PENGEMBANGAN APLIKASI *MOBILE* PENGADUAN MASYARAKAT PADA DINAS PERHUBUNGAN KOTA MALANG MENGGUNAKAN FITUR *LOCATION BASED SERVICE* BERBASIS ANDROID

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
FEGI ERIYANI
NIM: 175150209111028

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

PENGEMBANGAN APLIKASI *MOBILE* PENGADUAN MASYARAKAT PADA DINAS PERHUBUNGAN KOTA MALANG MENGGUNAKAN FITUR *LOCATION BASED SERVICE* BERBASIS ANDROID

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
FEGI ERIYANI
NIM: 175150209111028

Skripsi ini telah diuji dan dinyatakan lulus pada
7 Februari 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Bayu Priyambadha, S.Kom, M.Kom.	Ir. Heru Nurwarsito, M.Kom.
NIP: 198209092008121004	NIP: 196504021990021001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 197105182003121001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 7 Februari 2019

Fegi Eriyani

NIM: 175150209111028

KATA PENGANTAR

Puji syukur penulis panjatkan atas kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan penulisan skripsi yang berjudul "Pengembangan Aplikasi *Mobile* Pengaduan Masyarakat pada Dinas Perhubungan Kota Malang menggunakan Fitur *Location Based Service* berbasis Android". Terselesaikannya penelitian ini tidak lepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, penulis menyampaikan ucapan terima kasih sebanyak-banyaknya kepada:

1. Kedua orang tua serta keluarga penulis yang telah memberikan doa, semangat, kasih sayang, perhatian serta dukungannya.
2. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer.
3. Bapak Bayu Priyambadha, S.Kom., M.Kom dan Bapak Ir. Heru Nurwarsito, M.Kom selaku dosen pembimbing skripsi yang telah meluangkan banyak waktu untuk memberikan ilmu, saran dan juga bimbingan dalam menyelesaikan skripsi ini.
4. Segenap keluarga besar Dinas Perhubungan Kota Malang yang telah memberikan penulis waktu untuk melakukan penelitian pada Dinas Perhubungan Kota Malang dan mendapatkan data-data yang dibutuhkan dalam penelitian ini.
5. Untuk teman dekat penulis, Robi Dwi Setiawan yang selalu memberikan semangat dan membantu penulis dalam menyelesaikan skripsi ini.
6. Rekan-rekan penulis di Universitas Brawijaya atas dorongan semangat dan bantuan ilmu yang diberikan.
7. Pihak-pihak lain yang ikut membantu dan memberi dukungan serta masukan yang sangat berarti dalam menyelesaikan penelitian ini.

Penulis menyadari dalam penyusunan skripsi masih terdapat banyak kekurangan dan kesalahan. Untuk itu penulis mengharapkan saran dan kritik yang bersifat membangun demi kesempurnaan skripsi ini. Akhir kata penulis mengucapkan terima kasih sebanyak-banyaknya kepada semua pihak yang telah membantu dalam penyusunan skripsi.

Malang, 7 Februari 2019

Penulis

fegieryni@gmail.com

ABSTRAK

Fegi Eriyani, Pengembangan Aplikasi *Mobile* Pengaduan Masyarakat pada Dinas Perhubungan Kota Malang menggunakan Fitur *Location Based Service* berbasis Android.

Pembimbing: (1) Bayu Priyambadha, S.Kom., M.Kom, (2) Ir. Heru Nurwarsito, M.Kom.

Pengaduan masyarakat merupakan elemen yang penting dalam suatu instansi daerah karena bertujuan untuk melihat keberhasilan kerja yang telah dilakukan, memperbaiki kekurangan dan menerima saran dari tugas yang sudah dilaksanakan. Pengaduan masyarakat biasanya sering terjadi pada bidang sarana fasilitas dan pelayanan umum, dimana bidang tersebut ditangani oleh Dinas Perhubungan. Pengaduan masyarakat yang ada belum tersampaikan dengan baik karena kurangnya informasi untuk hal pengaduan sehingga tidak mengetahui tempat atau sarana untuk menyampaikan keluhan atas buruknya pelayanan publik, karena tidak adanya alur yang jelas atas keluhan masyarakat. Maka pada penelitian ini akan mengembangkan sebuah aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang berbasis Android guna memudahkan masyarakat dalam melaporkan keluhannya.

Aplikasi ini menggunakan fitur *location based service* yang digunakan untuk memudahkan dalam pencarian lokasi pelapor dengan mengetahuinya melalui keberadaan posisi *smartphone* pelapor. *Location based service* ialah layanan informasi lokasi menampilkan lokasi geografis *smartphone* pengguna yang dapat diakses dengan memanfaatkan *smartphone* tersebut. Aplikasi pengaduan masyarakat dapat mengirimkan bukti keluhan yang terjadi seperti foto melalui aplikasi *smartphone* dengan mengambil foto tersebut dari kamera langsung ataupun dari *gallery smartphone* pelapor. Aplikasi *mobile* dirancang menggunakan pemodelan berorientasi *object* dan diimplementasikan dengan menggunakan bahasa pemrograman Java dengan *database* MySQL sedangkan web untuk administrator dibangun dengan menggunakan *framework* laravel. Aplikasi ini dilakukan pengujian dengan menggunakan *whitebox testing* untuk pengujian *unit* dan pengujian integrasi, *blackbox testing* untuk pengujian validasi serta menggunakan pengujian *usability.* Hasil pengujian pada aplikasi pengaduan masyarakat menyatakan bahwa aplikasi ini dapat berjalan sesuai kebutuhan dan dapat digunakan dengan mudah oleh pengguna dengan memiliki *score System Usability Scale* 83,75 atau dapat dikatakan *acceptable.*

Kata kunci: Pengaduan Masyarakat, Dinas Perhubungan Kota Malang, *Location Based Service*, *Smartphone*.

ABSTRACT

Fegi Eriyani, ***Development of the Mobile Public Complaint Application at the Department of Transportation in Malang City using Location Based Service Feature based on*** **Android*.***

***Thesis Adviser*: (1) Bayu Priyambadha, S.Kom., M.Kom, (2) Ir. Heru Nurwarsito, M.Kom.**

Public complaints are an important element in a regional agency because the aim to see the success of the work that has been done, improve deficiencies and receive advice from the tasks that have been carried out. Public complaints usually occur in the sector of facilities and public services, where the sector is handled by the Department of Transportation. Public complaints that have not been conveyed properly due to lack of information for complaints so that they do not know the place or means to complain about the poor public services, because there is no clear path for public complaints. So in this thesis will develop an Mobile Public Complaints Application at the Department of Transportation in Malang City based on Android *to facilitate the public in reporting complaints.*

The application using the location based service feature that is used to make it easier to find the location of the reporter by knowing it through the presence of the User's Smartphone position. Location based service is a location information service that displays the geographical location of a user's smartphone that can be accessed using the smartphone. Public complaints application can send proof of complaints that occur such as photos through the smartphone application by taking photos from the camera directly or from the user's smartphone gallery. The Mobile Application are designed using object-oriented modeling and implemented using the Java programming language with MySQL databases while the web for administrators implemented using the laravel framework. The application has been tested using whitebox testing for unit testing and integration testing, blackbox testing for validation testing, and also usability testing. The test results on the application of public complaints state that this application can run as needed and can be used easily by users by having a 83,75 System Usability Scale score or can be said to be acceptable.

Keywords*: Public Complaints, Department of Transportation in Malang City, Location Based Service, Smartphone.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Dinas Perhubungan memiliki tugas pokok dan fungsinya yang berada dibawah dan bertanggung jawab melalui Sekretaris Daerah kepada Walikota dalam hal melaksanakan unsur pemerintahan dalam bidang Perhubungan yang dipimpin oleh Kepala Dinas (Dishub, 2015). Dinas Perhubungan Kota Malang diarahkan untuk upaya penyelenggaraan pelayanan dalam bidang transportasi dan sarana umum lalu lintas guna mewujudkan pelayanan tersebut dalam menjamin keamanan, ketertiban, dan kenyamanan untuk masyarakat. Masyarakat memiliki peranan penting dalam membantu terciptanya keadaan tersebut. Namun untuk kualitas pelayanan dan sarana yang diberikan oleh Dinas Perhubungan di anggap masih sangat kurang sedangkan kualitas dari pelayanan publik yang diberikan dengan baik dapat digunakan sebagai kunci keberhasilan dalam pelayanan publik. Untuk meningkatkan pelayanan publik yang baik dengan memberikan kesempatan pada masyarakat untuk menyampaikan pengaduan berupa keluhan, saran ataupun kritik yang membangun tentang pelayanan yang diterima oleh masyarakat yang tidak sesuai dengan sebagaimana mestinya. Keluhan masyarakat tersebut biasanya sering terjadi di bidang sarana fasilitas umum dan pelayanan umum.

Pengaduan masyarakat merupakan elemen yang penting dalam suatu instansi daerah karena bertujuan untuk melihat keberhasilan kerja yang telah dilakukan, memperbaiki kekurangan dan menerima saran dari tugas yang sudah dilaksanakan. Dengan berkembang pesatnya teknologi saat ini membuat pelayanan publik mendorong untuk menggunakan media elektronik seperti halnya dengan pengaduan masyarakat pada Dinas Perhubungan Kota Malang. Pengaduan masyarakat pada Dinas Perhubungan terkait belum tersampaikan dengan baik sehingga membuat masyarakat kebingungan harus menyampaikan keluhan kepada siapa. Keluhan masyarakat yang sering terjadi seperti keluhan terkait sarana umum lalu lintas seperti rambu-rambu lalu lintas, kemacetan, parkir liar, dan sarana umum lalu lintas lainnya yang menjadi tanggung jawab Dinas Perhubungan (Prasetya, et al., 2013). Dengan banyaknya pengaduan lalu lintas yang ada, membuat masyarakat semakin bingung harus menyampaikan keluhan dimana dan kepada siapa karena tidak mengetahuinya tempat atau sarana untuk menyampaikan keluhan atas buruknya pelayanan publik, karena tidak adanya alur yang jelas atas keluhan masyarakat, kurangnya sosialiasi pun menjadikan masyarakat tidak mengerti instansi pemerintah yang mana yang dapat menindaklanjuti pengaduan yang dikeluhkan masyarakat. Sedangkan untuk pengaduan masyarakat Dinas Perhubungan Kota Malang hanya menggunakan layanan *Contact Center* 151, penerapan *e-mail* dan *web* resmi Dinas Perhubungan Kota Malang untuk mempermudah penyampaian informasi dan penyampaian keluhan, kritik, dan saran. Proses pengaduan tersebut dapat dilakukan dengan melaporkan pengaduan pada media-media yang telah disediakan oleh Dinas Perhubungan Kota Malang yang kemudian selanjutnya laporan tersebut akan

ditindaklanjuti oleh petugas lapangan Dinas Perhubungan Kota Malang. Meskipun telah disediakan sarana tersebut, hal ini masih dirasa kurang efektif karena kurang terkoordinasi dengan baik dan dari banyaknya pengaduan tidak bisa semua ditindaklanjuti karena kurangnya informasi yang tidak lengkap dan tidak terpercaya.

Terdapat sebuah penelitian yang sebelumnya pernah dilakukan pada tahun 2017, penelitian tersebut membahas mengenai berbagai pengaduan masyarakat tentang semua permasalahan sarana umum yang terjadi di Kota Malang. Pada penelitian tersebut meneliti tentang bagaimana merancang dan membangun sistem untuk melaporkan keluhan yang berjudul RANCANG BANGUN SISTEM PELAPORAN KELUHAN MASYARAKAT DENGAN FITUR *GEOTAGGING* BERBASIS ANDROID. Penelitian ini membahas mengenai pelaporan keluhan oleh masyarakat dengan menggunakan fitur *geotagging* untuk mempermudah pengguna untuk menemukan lokasi dengan menggunakan metode *point in polygon* yang dikelompokkan berdasarkan wilayah kecamatan dalam Kota Malang (Fauzia, 2017). Penelitian lainnya yaitu tentang jual beli produk pertanian. Aplikasi yang dibuat pada penelitian tersebut merupakan aplikasi untuk jual beli produk pertanian pada *smarthphone* berbasis Android, pada penelitian tersebut menerapkan metode *Location Based Service* untuk menampilkan lokasi secara geografis sesuai dengan letak *smartphone* pengguna sehingga memudahkan untuk memasarkan produk petani. Penelitian ini dilakukan pada tahun 2016 yang berjudul RANCANG BANGUN APLIKASI JUAL BELI PRODUK PERTANIAN MENGGUNAKAN PENDEKATAN *LOCATION BASED SERVICE* (Firdaus, 2016).

Sedangkan pada penelitian penulis lebih berfokus terhadap pelaporan keluhan masyarakat pada Dinas Perhubungan Kota Malang, pelaporan keluhan tersebut berkaitan dengan sarana umum lalu lintas sesuai dengan tugas Dinas Perhubungan, sehingga penelitian ini mengimplementasikan sebuah aplikasi pengaduan masyarakat Kota Malang untuk melaporkan keluhan pada Dinas Perhubungan Kota Malang dan dapat membantu Dinas Perhubungan dalam menindaklanjuti keluhan dengan cepat dan tanggap. Semua keluhan mengenai sarana umum lalu lintas seperti kemacetan, kerusakan lampu lalu lintas, kerusakan rambu-rambu lalu lintas, hilangnya marka jalan, parkir ilegal dan sarana umum lainnya yang menjadi tanggung jawab dari Dinas Perhubungan dimasukkan ke dalam sebuah aplikasi melalui sebuah foto yang diambil langsung dari kamera *smartphone*. Informasi dimuat dalam sebuah perangkat bergerak berbasis Android dengan memanfaatkan fitur *location based service*, sehingga memudahkan dalam pencarian lokasi pelapor dengan mengetahuinya melalui keberadaan posisi aplikasi *mobile* pelapor. *Location based service* adalah sebuah layanan informasi lokasi, untuk mengaksesnya dengan memanfaatkan *smartphone* secara *online* serta dapat menampilkan lokasi geografis *smartphone* pengguna, *location based service* ini memungkinkan pengguna mendapatkan informasi lokasi sesuai dengan kebutuhan (Anwar, et al., 2014). Sehingga penggunaannya dapat mempermudah dalam mengetahui lokasi keberadaan pelapor dan dapat mengirimkan foto keluhan yang terjadi tidak hanya

mengirimkan melalui *text* saja melainkan dapat melapor dengan mengirimkan bukti keluhan melalui aplikasi *smartphone*.

Penelitian yang dilakukan ini dirancang pada sebuah *smartphone* untuk aplikasi *mobile* berbasis Android. Dimana *smartphone* dapat digunakan dengan mudah dan sering dijumpai sehingga masyarakat sudah terbiasa menggunakan *smartphone* untuk membantu aktifitas sehari-hari. Pada tahun 2014, jumlah pengguna masih pada kisaran angka 38,3 juta. Sedangkan Tahun 2018 jumlah pengguna *smartphone* mencapai 100 juta pengguna dengan Negara paling banyak menggunakan *smartphone* ialah China, India, dan Amerika Serikat termasuk Indonesia berada pada urutan keempat dunia. Melihat jumlah pengguna sudah mencapai hampir setengah dari jumlah penduduk Indonesia secara keseluruhan, maka perkembangan *smartphone* di Indonesia sangat pesat. Perusahaan-perusahaan yang menjual perangkatnya dengan harga terjangkau merupakan alasan pendukung pertumbuhan *smartphone* di Indonesia (Symu, 2013). *Smartphone* atau perangkat bergerak merupakan suatu perangkat yang memiliki sistem operasi yang digunakan untuk mengoperasikannya, sistem operasi yang dimaksud ialah sistem operasi *smartphone* yang merupakan salah satu sistem operasi yang ada pada *smartphone*. Android adalah sistem operasi *smartphone* yang berkembang pesat yang ada pada sebuah perangkat bergerak yang terdapat sistem operasi dan aplikasi-aplikasi utama lainnya didalamnya (Juhara, 2016).

Aplikasi ini diharapkan dapat memudahkan masyarakat dalam menyampaikan pengaduan berupa keluhan, kritik, dan saran pada Dinas Perhubungan Kota Malang. Mengingat Dinas Perhubungan dalam tugasnya tidak terlepas dari pekerjaan lapangan yaitu mencakup seluruh Kota Malang maka perlu adanya sebuah aplikasi pengaduan yang bisa memberikan petunjuk jelas tentang kejadian dilapangan seperti foto dan lokasi pelaporan dengan menggunakan fitur *location based service* agar lokasi pelaporan lebih akurat sehingga Dinas Perhubungan dapat memberi umpan balik terhadap pengaduan masyarakat. Selain itu, aplikasi ini dapat memberikan *support* atau dukungan terhadap pengaduan masyarakat yang disampaikan oleh pelapor lain serta pelapor dapat melihat seluruh daftar dukungan terhadap dukungannya kepada pelapor lain. Berdasarkan pentingnya akan wadah pengaduan masyarakat tentang sarana umum lalu lintas diperlukan suatu aplikasi untuk menjembatani masyarakat dengan Dinas Perhubungan yaitu layanan pengaduan masyarakat berbasis Android yang akan diangkat pada skripsi ini dengan judul, "Pengembangan Aplikasi *Mobile* Pengaduan Masyarakat pada Dinas Perhubungan Kota Malang Menggunakan Fitur *Location Based Service* Berbasis Android".

1.2 Rumusan Masalah

Dari permasalahan pada latar belakang yang sudah dijelaskan, maka rumusan masalah tersebut sebagai berikut:

1. Bagaimana memberikan layanan dan fasilitas untuk menyampaikan keluhan kepada Dinas Perhubungan Kota Malang pada sebuah aplikasi *mobile* pengaduan masyarakat berbasis Android?

2. Bagaimana menerapkan fitur *Location Based Service* untuk mengetahui letak posisi yang akurat sesuai dengan keberadaan masyarakat yang melapor pada sebuah perangkat bergerak?
3. Bagaimana tingkat kemudahan aplikasi *mobile* pengaduan masyarakat pada Dinas Perhubungan Kota Malang berbasis Android yang mudah dimengerti dan mudah digunakan oleh pengguna?

1.3 Tujuan

Berdasarkan rumusan masalah yang telah dirumuskan, maka didapatkan tujuan penulis mengangkat topik ini sebagai berikut:

A. Tujuan Umum

Merancang dan membangun sebuah aplikasi pada *smartphone* berbasis Android dalam membantu masyarakat untuk memudahkan menyampaikan keluhan mengenai sarana umum lalu lintas kepada Dinas Perhubungan Kota Malang serta mempermudah pihak tersebut dalam menindaklanjuti keluhan masyarakat serta memberikan informasi kepada masyarakat jika keluhan yang dilaporkan sudah ditindaklanjuti oleh pihak terkait.

B. Tujuan Khusus

1. Memberikan layanan dan fasilitas pada aplikasi *mobile* untuk pengaduan masyarakat tentang sarana umum lalu lintas guna membantu masyarakat untuk menyampaikan keluhan pada Dinas Perhubungan Kota Malang.
2. Menerapkan fitur *location based service* untuk mengetahui letak posisi yang akurat sesuai dengan keberadaan masyarakat yang melapor pada sebuah perangkat bergerak.
3. Mengetahui tingkat kemudahan dari aplikasi *mobile* pengaduan masyarakat pada Dinas Perhubungan Kota Malang berbasis Android yang mudah dimengerti dan mudah digunakan oleh pengguna.

1.4 Manfaat

Penelitian ini dilakukan agar dapat memberikan manfaat dari beberapa sisi, adapun manfaatnya sebagai berikut:

A. Manfaat Bagi Penulis

Dapat menerapkan materi mata kuliah dan praktikum yang sudah ditempuh selama menjalankan perkuliahan di jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.

B. Manfaat Bagi Masyarakat

1. Membantu masyarakat untuk mempermudah dalam menyampaikan keluhan pada Dinas Perhubungan Kota Malang perihal keluhan sarana umum lalu lintas.

2. Memberikan peringatan kepada masyarakat yang melapor bahwa keluhan yang disampaikan sudah di tindaklanjuti dan diselesaikan membuat masyarakat ikut membantu membuat sarana umum lalu lintas Kota Malang menjadi lebih baik.
3. Membantu berperan aktif dalam penyampaian informasi secara cepat kepada Dinas Perhubungan Kota Malang.

C. Manfaat Bagi Dinas Perhubungan Kota Malang

1. Memudahkan Dinas Perhubungan Kota Malang untuk cepat dan tanggap dalam menindaklanjuti keluhan yang disampaikan masyarakat.
2. Membantu Dinas Perhubungan Kota Malang untuk mengetahui lokasi yang akurat sesuai dengan keberadaan pelapor dalam melaporkan keluhannya.
3. Mempermudah kinerja Dinas Perhubungan Kota Malang sehingga dapat meningkatkan pelayanan kualitas publik yang baik.

1.5 Ruang Lingkup Penelitian

Agar penelitian yang dilakukan dapat terarah dan tidak semakin meluas, maka perlu adanya ruang lingkup dan batasan masalah adalah sebagai berikut:

A. Ruang Lingkup

Ruang lingkup penelitian ini digunakan untuk mengarahkan penelitian agar fokus terhadap ruang lingkup yang sedang diteliti dan tidak meluas. Ruang lingkup pada penelitian ini adalah aplikasi yang dikembangkan pada Dinas Perhubungan di Kota Malang mencakup semua wilayah pada Kota Malang.

B. Batasan Masalah

Penelitian ini memiliki batasan masalah yang digunakan untuk memberi batasan agar penelitian mempunyai batasan masalah yang jelas dari topik yang sedang diteliti, batasan masalah tersebut sebagai berikut:

1. Aplikasi dibangun pada sebuah perangkat *mobile* berbasis Android.
2. Sistem yang digunakan untuk admin berbasis web dengan menggunakan *framework* Laravel.
3. Informasi yang dikirimkan masyarakat hanya berupa keluhan menggunakan deskripsi (*text*), foto, dan lokasi pelaporan.
4. Fitur *location based service* hanya untuk mengetahui lokasi pelapor dapat dilihat berdasarkan letak lokasi yang dilaporkan oleh pelapor.
5. Aplikasi ini hanya dapat menangani keluhan masyarakat tentang sarana umum lalu lintas yang menjadi tanggung jawab Dinas Perhubungan Kota Malang.
6. Pengaduan masyarakat antara lain yang terkait dengan permasalahan kondisi lalu lintas, rambu-rambu lalu lintas, jalan raya, dan fasilitas umum

lainnya yang sesuai dengan cakupan kasus yang ditangani Dinas Perhubungan Kota Malang.

1.6 Sistematika Penulisan

Penulisan penelitian dijelaskan berdasarkan garis besar pada setiap bab dari keseluruhan isi penulisan, gambaran tersebut terdiri dari bab – bab sebagai berikut:

BAB I PENDAHULUAN

Pada BAB I membahas tentang latar belakang masalah, rumusan masalah, tujuan umum dan khusus penelitian, manfaat penelitian dari berbagai sisi, ruang lingkup dan batasan masalah yang ada pada penelitian serta rencana kegiatan pada penelitian.

BAB II LANDASAN KEPUSTAKAAN

Pada BAB II menjelaskan mengenai kajian literatur dari penelitian yang sudah ada mengenai penelitian yang sama dan menjelaskan tentang dasar teori pendukung yang digunakan pada penelitian.

BAB III METODEOLOGI PENELITIAN

Pada BAB III membahas tentang langkah-langkah yang akan dilakukan dalam penelitian metode yang terdiri dari studi literatur, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis, serta kesimpulan dan saran.

BAB IV ANALISIS KEBUTUHAN SISTEM

Pada BAB IV menjelaskan mengenai analisis kebutuhan dari sistem yang akan dibuat dan berisi pemodelan dari perangkat lunak yang digunakan.

BAB V PERANCANGAN DAN IMPLEMENTASI SISTEM

Pada BAB V menjelaskan mengenai rancangan aplikasi yang akan dibangun pada penelitian dan membahas tentang implementasi atau hasil dari perancangan aplikasi yang telah dibuat pada perancangan.

BAB VI PENGUJIAN DAN ANALISIS HASIL

Pada BAB VI membahas mengenai pengujian yang dilakukan pada aplikasi serta hasil uji yang dihasilkan dari aplikasi tersebut.

BAB VII PENUTUP

Pada BAB VII memuat sebuah kesimpulan dari penelitian yang dilakukan. Serta memuat saran yang mungkin dapat dikembangkan lagi dalam pengembangan selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Bab landasan kepustakaan menjelaskan mengenai kajian literatur dan dasar teori yang berkaitan dengan penelitian. Kajian literatur membahas yang berkaitan dengan penelitian sebelumnya dengan penelitian yang sedang dilakukan. Dan dasar teori yang menjelaskan tentang teori-teori pendukung penelitian yang sedang dilakukan.

2.1 Kajian Literatur

Kajian literatur merupakan penjelasan penelitian-penelitian sebelumnya mengenai penggunaan *location based service* dan aplikasi pengaduan masyarakat berbasis Android. Tujuan di cantumkannya kajian literatur adalah untuk menjadikan sebagai studi literatur pada proses pengerjaan penelitian ini serta membantu penulis menganalisa kebutuhan penelitian. Penjelasan penelitian mengenai kajian literatur terdapat pada Tabel 2.1.

Tabel 2. 1 Tabel Kajian Literatur

No	Judul Penelitian	Nama Penulis	Perbandingan	
			Kajian Literatur	Skripsi Penulis
1.	Rancang Bangun Aplikasi *Mobile* Jual Beli Produk Pertanian Menggunakan Pendeketan *Location Based Service* (LBS)	(Firdaus, 2016)	Implementasi *location based service* yang digunakan untuk mengumpulkan informasi lokasi petani menjual produk pertanian yang sudah dipasarkan	Penggunan fitur *location based service* untuk aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang
2.	Pengembangan Aplikasi *Mobile* Pelaporan Keluhan Pelanggan Listrik	(Wiriyo, 2016)	Implementasi *location based service* yang digunakan untuk mengetahui keberadaan lokasi	Penggunan fitur *location based service* untuk aplikasi pengaduan masyarakat pada

	Menggunakan Fitur *Location Based Service* Berbasis Android		pelapor pelanggan listrik berbasis Android	Dinas Perhubungan Kota Malang
3.	Rancang Bangun Sistem Laporan Keluhan Masyarakat Dengan Fitur *Geotagging* Berbasis Android	(Fauzia, 2017)	Perancangan dan implementasi sistem laporan keluhan untuk masyarakat di Kota Malang dengan menggunakan fitur *geotagging* dan metode *point in polygon* untuk mengelompokkan keluhan berdasarkan wilayah	Penggunan fitur *location based service* untuk sistem pengaduan masyarakat pada Dinas Perhubungan Kota Malang

Penelitian pertama dilakukan oleh Firdaus dengan mengimplementasikan fitur *location based service* digunakan untuk mengumpulkan informasi lokasi petani menjual produk pertanian di Kota Malang yang berjudul "Rancang Bangun Aplikasi *Mobile* Jual Beli Produk Pertanian menggunakan Pendeketan *Location Based Service* (LBS)". Pada penelitian ini, penulis menggunakan teknologi *location based service* untuk membantu petani memasarkan produk pertanian secara lebih luas dengan mencari lokasi yang di temukan dari GPS *smartphone* dan menjangkau konsumen secara langsung serta membantu para pembeli dalam mencari bahan pangan yang telah dipasarkan oleh petani berdasarkan lokasi petani. Penulis memanfaatkan fitur *location based service* untuk mendapatkan lokasi geografis *user* berdasarkan lokasi yang ditemukan dari GPS *smartphone user.* Penelitian ini dapat dijadikan pandangan untuk melakukan tahap analisis kebutuhan pada penelitian yang sedang dilakukan dimana dengan memanfaatkan fitur *location*

based service. Perbedaannya penelitian yang sedang dilakukan menggunakan fitur *location based service* untuk pengaduan masyarakat pada Dinas Perhubungan Kota Malang untuk mengetahui lokasi pelapor secara akurat menggunakan perangkat *mobile* (Firdaus, 2016).

Penelitian kedua merupakan penelitian dilakukan oleh Sifasani Qalbina Fauzia yang berjudul "Rancang Bangun Sistem Laporan Keluhan Masyarakat Dengan Fitur *Geotagging* Berbasis Android". Pada penelitian tersebut, penulis menggunakan fitur *geotagging* untuk mempermudah pengguna untuk mencari lokasi pengguna dengan memanfaatkan metode *point in polygon*, lokasi yang ditemukan berupa koordinat *latitude* dan *longitude* yang dikelompokkan berdasarkan wilayah kecamatan Kota Malang dan berfungsi memberi *marker* pada lokasi yang akan dilaporkan. Penelitian ini dapat dijadikan referensi pada penelitian yang sedang dilakukan dimana penelitian ini serupa membahas yang bertopik pengaduan masyarakat. Namun perbedaan penelitian yang sedang dilakukan lebih berfokus pada pengaduan masyarakat untuk sarana umum lalu lintas pada Dinas Perhubungan Kota Malang dengan menggunakan perangakat bergerak (Fauzia, 2017).

Penelitian ketiga dilakukan oleh Fahmi Hammadi Wiriyo yang berjudul "Pengembangan Aplikasi *Mobile* Pelaporan Keluhan Pelanggan Listrik Menggunakan Fitur *Location Based Service* Berbasis Android". Penelitan tersebut, penulis memanfaatkan fitur *location based service* digunakan untuk mengetahui lokasi pelapor dengan mengetahuinya melalui keberadaan posisi *smartphone* yang digunakan *user* melaporkan keluhan pengguna listrik. Dari penelitian tersebut bisa diambil sebagai *literature review* sebagaimana pemanfaatan *location based service* untuk mencari lokasi pelapor dengan mengetahui keberadaannya melalui posisi *smartphone* pelapor yang dikembangkan pada penelitian tersebut dapat dikembangkan dalam penelitian yang dilakukan (Wiriyo, 2016).

2.2 Dinas Perhubungan Kota Malang

Peraturan Daerah Nomor 6 Tahun 2012 tentang Organisasi dan Tata Kerja Dinas Daerah dan Peraturan Walikota nomor 45 Tahun 2012 tentang Uraian Tugas Pokok, Fungsi dan Tata Kerja Dinas Perhubungan Kota Malang, maka Dinas Perhubungan memiliki tugas pokok dan fungsinya yang berada dibawah dan bertanggung jawab melalui Sekretaris Daerah kepada Walikota dalam hal melaksanakan unsur pemerintahan dalam bidang Perhubungan yang dipimpin oleh Kepala Dinas. Dalam melaksanakan tugas pokok pelaksanaan kebijakan daerah dibidang Perhubungan, Dinas Perhubungan mempunyai tugas antara lain merumuskan kebijakan teknis dibidang perhubungan yang meliputi teknis angkutan dan terminal, teknis lalu lintas dan parkir, teknis pengujian kendaraan bermotor, teknis perizinan, teknis pengendalian dan operasional berdasarkan peraturan perundang – undangan yang berlaku (Dishub, 2015).

Seiring bertambahnya jumlah penduduk Kota Malang yang semakin meningkat karena Kota Malang sebagai kota pendidikan serta akibat urbanisasi, maka Dinas Perhubungan Kota Malang perlu meningkatkan pelayanan khususnya dibidang

transportasi, hal ini merupakan tantangan sekaligus peluang dalam meningkatkan kualitas transportasi bagi Dinas Perhubungan, tantangan tersebut seperti dapat memenuhi keamanan dan kenyamanan masyarakat dalam berlalu lintas seperti dalam hal kemacetan, pemenuhan akan sarana dan prasarana lalu lintas yang mencukupi serta memenuhi kenyamanan masyarakat terhadap pelayanan perparkiran yang aman.

2.3 Keluhan dan Pengaduan Masyarakat

Keluhan masyarakat biasanya terjadi karena adanya suatu masalah pada lingkungan sekitar baik di bidang fasilitas umum dan layanan umum (lalu lintas, sekolah, kesehatan, jalan umum dan sebagainya), infrastruktur, sosial, dan lingkungan. Semua keluhan tersebut biasanya yang sering dilaporkan oleh masyarakat kepada instansi terkait. Tidak adanya alur yang jelas untuk pengaduan masyarakat merupakan alasan lain yang membuat masyarakat bingung untuk menyampaikan semua keluhan. Seperti halnya pada Dinas Perhubungan Kota Malang, sistem pengaduan masyarakat belum sepenuhnya diketahui secara optimal oleh masyarakat Kota Malang. Kurangnya sosialisasi kepada masyarakat menyebabkan mereka tidak mengetahui untuk melaporkan keluhannya kepada pemerintah.

Ada beberapa syarat yang harus dipenuhi oleh pengadu atau masyarakat jika ingin menyampaikan keluhan, karena jika tidak memenuhi syarat tersebut keluhan atau pengaduan yang disampaikan kepada pemerintah atau pihak terkait tidak akan ditanggapi. Syarat tersebut terdiri dari tiga jenis, syarat pertama yaitu tempat atau lokasi dari kejadian yang dilaporkan secara lengkap dan jelas, seperti nama jalan, nomor jalan, nama kelurahan, dan nama kecamatan. Syarat kedua ialah isi dari keluhan dan pengaduan yang ingin dilaporkan dan syarat yang ketiga adalah bukti-bukti pendukung seperti foto-foto mengenai permasalahan yang diadukan kepada pemerintah (Prasetya, et al., 2013).

2.4 Pelaporan Keluhan Masyarakat

Pelaporan keluhan masyarakat ialah laporan atau pernyataan atas pengaduan dan keluhan masyarakat mengenai masalah yang ada di sekitar. Laporan yang disampaikan dapat dilaporkan melalui berbagai media, seperti media sosial, media masa, maupun media telekomunikasi. Pelaporan keluhan masyarakat yang disampaikan kepada pemerintah daerah ataupun pihak terkait dapat dilakukan melalui berbagai macam bentuk. Berikut bentuk-bentuk tempat penyampaian keluhan yang disediakan antara lain (Wiyanto, 2011):

1. Melalui tatap muka secara langsung dengan catatan harus membuat janji untuk bertemu dengan staf atau pejabat yang berkaitan dengan penerimaan keluhan dengan cara mengunjungi kantor penyedia pelayanan
2. Melalui telepon saat jam kerja
3. Melalui *email*
4. Melalui surat menyurat

5. Melalui mesin faksimili kepada siapa keluhan ditujukan

2.5 Rekayasa Perangkat Lunak

Rekayasa Perangkat Lunak merupakan suatu bidang keilmuan yang mempelajari mengenai perekayasaan yang berhubungan dengan aspek yang berkaitan dengan proses pembangunan perangkat lunak dengan cara yang efeketif dan efisien yang dilihat dari segi waktu, biaya, dan tenaga. Rekayasa perangkat lunak berfokus pada setiap tahapan-tahapan pengembangan perangkat lunak mulai dari tahap awal spesifikasi sistem yaitu tahapan identifikasi sampai pemeliharaan sistem setelah digunakan yang dimana tahapan tersebut merupakan tahapan akhir dari pengembangan perangkat lunak (Sommervile, 2003).

2.6 *Software Development Life Cycle*

Software Developmet Life Cycle atau SDLC yang biasa disebut dengan siklus hidup atau tahapan-tahapan pengembangan perangkat lunak. SDLC merupakan sebuah metode pengembangan perangkat lunak yang digunakan untuk mendefinisikan sebuah pekerjaan dalam suatu sistem informasi yang dilakukan pada setiap proses tahapan yang harus diikuti untuk melakukan seluruh tahapan-tahapan seperti yang digunakan untuk menganalisa, merancang, mengimplementasikan, dan memelihara suatu perangkat lunak (Pressman, 2001).

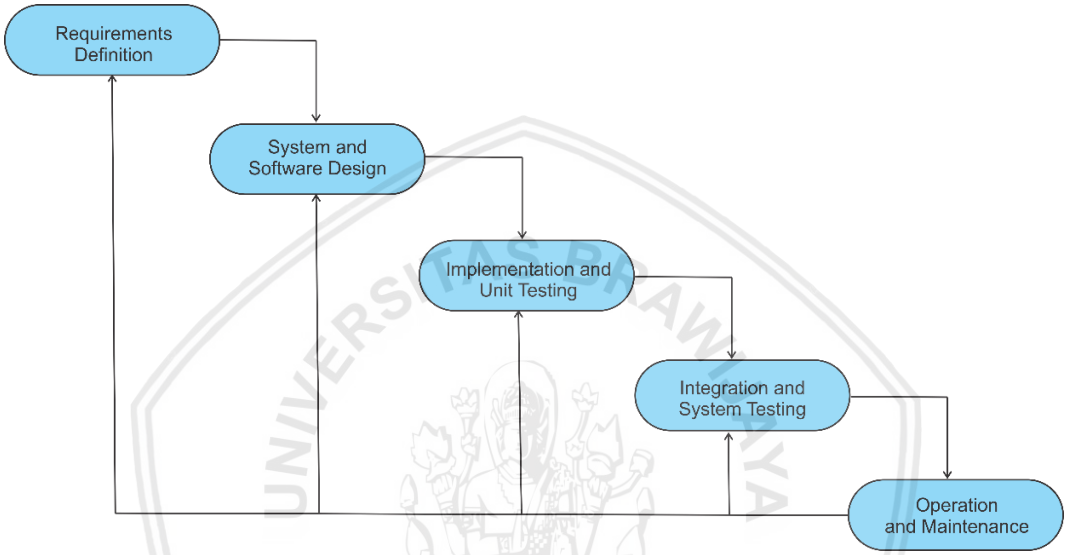
Software Development Life Cycle ialah urutan kegiatan perangkat lunak yang mungkin berbeda pada setiap tahapan-tahapan yang terjadi selama kegiatan pengembangan perangkat lunak (Gustafson, 2002). Pada sebuah metode SDLC terdapat beberapa model satu dari beberapa model tersebut ialah model *waterfall* dimana model tersebut yang akan peneliti pakai pada pengembangan aplikasi pengaduan masyarakat. Pengertian dan tahapan-tahapan pada model *waterfall* akan dijelaskan pada sub bab berikut ini.

2.6.1 Waterfall Model

Terdapat beberapa macam model SDLC dimana didefinisikan untuk diikuti setiap tahapan pada proses pengembangan perangkat lunak. Salah satu dari macam model SDLC yaitu *waterfall model* yang mana model yang bersifat klasik dibandingkan dengan model SDLC lainnya. *Waterfall model* disebut juga dengan model *linear* yang mana model SDLC ini diperkenalkan dan dipublikasikan sejak lama yang berawan dari proses *system engineering* secara *general* (Sommerville, 2011).

Model SDLC ini diberi nama *waterfall* karena proses yang terjadi pada tahapan ini sama seperti layaknya air terjun yakni tiap tahapan satu ke tahapan yang lainnya mengalir ke bawah seperti *waterfall* atau air terjun, oleh karena itu proses dari fase satu harus selesai terlebih dahulu sebelum berlanjut ke fase selanjutnya. *Waterfall model* merupakan model pengembangan perangkat lunak yang kebutuhan-kebutuhan yang harus ada pada perangkat lunak harus sudah pasti dan direncanakan dahulu sebelum memulai untuk mengerjakannya atau disebut

dengan *plan-driven*. Untuk itu model ini berfokus pada tahap awal berupa perencanaan yang harus disiapkan secara matang dan memastikan tidak ada perubahan pada kebutuhan-kebutuhan fungsional pada perangkat lunak dan memastikan tidak terdapat kecacatan pada tahap perancangan sebelum melanjutkannya pada tahapan pengembangan perangkat lunak. Pada model ini juga berfokus pada dokumentasi perangkat lunak pada tiap tahapannya sehingga akan menghasilkan dokumentasi dan kualitas perangkat lunak yang baik (Sommerville, 2011). Berikut merupakan diagram dari *waterfall model*, dapat ditunjukkan pada Gambar 2.1.



Gambar 2. 1 Diagram *Waterfall Model*

Sumber: Sommerville (2011)

Berikut fase-fase yang dimiliki oleh model *waterfall* dalam mengembangkan sebuah perangkat lunak:

- *Requirements Definition*

  Tahapan ini merupakan tahapan pertama pada *waterfall model*. Tahapan ini mendifinisikan kebutuhan-kebutuhan perangkat lunak yang akan dikembangkan, tahapan *requirements definition* ini disebut juga dengan SRS atau *Software Requirements Spesification*. Pada SRS ini terdapat deskripsi lengkap dari setiap kebutuhan perangkat lunak. Kebutuhan tersebut dijelaskan secara *detail* termasuk batasan-batasan pada perangkat lunak, dan tujuan yang diinginkan dari perangkat lunak. Dari kebutuhan-kebutuhan yang sudah dijelaskan tersebut akan dijadikan acuan pada tahap perancangan sistem (Sommerville, 2011). Kebutuhan-kebutuhan yang dimaksudkan ialah kebutuhan fungsional dan juga kebutuhan non fungsional dimana menghasilkan *use case diagram* dan *sequence diagram* yang digunakan untuk tahap pemodelan kebutuhan.

- *System and Software Design*

  Ketika proses *requirements definition* sudah dilakukan, maka tahapan selanjutnya ialah *system and software design* dimana pada tahap ini melakukan perancangan dari kebutuhan yang sudah didefinisikan pada tahapan sebelumnya. Tahapan ini membuat arsitektur dari kebutuhan keseluruhan sistem yang digunakan untuk menempatkan semua kebutuhan-kebutuhan pada perangkat lunak (Sommerville, 2011).

- *Implementation and Unit Testing*

  Dengan selesai tahapan *system* dan *software design* maka selanjutnya masuk ke tahapan *implementation and unit testing*. Tahapan ini mengimplementasikan perancangan atau desain yang telah dibuat sebelumnya untuk dijadikan sebuah algoritme dari program. Sedangkan *unit* testing digunakan untuk menguji dari setiap *unit* pada algoritme yang sudah dibuat tersebut dan memastikan sesuai dengan perancangan (Pressman, 2011).

- *Integration and SystemTesting*

  Setelah *implementation and unit testing* selesai maka tahapan selanjutnya ialah pengujian untuk menemukan *bug* atau kesalahan yang terdapat pada perangkat lunak. Pada tahapan ini melakukan verifikasi atau memastikan ada atau tidaknya kesalahan, kecatatan atau bug pada sistem yang dikembangkan. Tahapan *integration and system testing* ini memiliki *unit* program yang individu maupun yang saling berhubungan, untuk itu dilakukan pengujian integrasi sebagai kesatuan *unit* sistem yang telah selesai untuk memastikan jika kebutuhan perangkat lunak sudah memenuhi sesuai dengan yang diinginkan (Sommerville, 2011).

- *Operation and Maintenance*

  Tahapan yang terakhir ialah *operation and maintenance*, pada tahapan ini melakukan pemeliharaan terhadap perangkat lunak yang sudah dikembangkan dengan berkala (Sommerville, 2011).

2.7 *Smartphone*

Pada era teknologi seperti saat ini, *smartphone* merupakan alat komunikasi yang sering dijumpai dikalangan masyarakat. Hampir semua orang memiliki *smartphone* sebagai media komunikasi digital. *Smartphone* dapat membantu aktifitas sehari-hari yang digunakan dengan mudah dan sering dijumpai di masyarakat sehingga masyarakat sudah terbiasa menggunakannya. *Smartphone* merupakan media digital yang membantu aktifitas dalam berbagai bidang dan bersifat multifungsi karena *smartphone* sangat *fleksible* dimana mudah dibawa kemana saja sehingga membantu memenuhi keinginan penggunanya.

Smartphone ialah telepon yang sangat pintar dimana mempunyai kemampuan serupa dengan *personal* komputer. *Smartphone* memiliki kelebihan *mobile computing* yang termasuk ke dalam kategori sebagai *high end mobile phone*

dimana *smartphone* berbeda dengan ponsel biasa karena memiliki kelebihan *mobile computing* tersebut. Pertama kali *smartphone* muncul merupakan gabungan dari telepon dengan kamera atapun telepon genggam dengan fungsi dari *personal digital assistant* (PDA). Saat ini fungsi lainnya dari sebuah *smartphone* ialah memiliki video c*amera*, media player, dan internet broadband, GPS serta akses data WiFi (Williams & Sawyer, 2011).

2.8 Android

Android ialah salah satu dari banyak sistem operasi yang ada pada *smartphone*, sistem operasi yang berbasis Linux ini dengan mudah di lakukan gabungan untuk perangkat bergerak yang meliputi bagian dari *middleware*, dan sistem operasi. Sistem operasi Android dibuat dan dikembangkan oleh perusahaan Android Inc. yang kemudian perusahaan tersebut pada tahun 2005 dibeli oleh Google (Juhara, 2016).

Ada tiga bagian pada Android yang harus dimengerti dalam membangun aplikasi berbasis Android, yaitu (Pambudi, 2013).

1. *Intent* yaitu komponen pada Android dimana memiliki fungsi dapat memberikan notifikasi dan sekaligus menerima notifikasi tersebut.
2. *Service* merupakan komponen pada Android yang layanan atau *service* nya dapat digunakan pada *background* karena tidak memiliki antarmuka pengguna.
3. *Content Provider* ialah komponen yang dapat digunakan oleh aplikasi lain yang merupakan kumpulan data aplikasi secara spesifik.

Untuk itu Android adalah suatu sistem yang memiliki susunan yang terdiri dari beberapa bagian perangkat lunak. Jadi dapat diambil kesimpulan Android bukan hanya berdiri diatas suatu sistem atau sebuah *framework* (Pambudi, 2013).

2.9 Java

Bahasa pemrogaraman Java ialah salah satu dari banyak bahasa pemrograman yang berorientasikan objek, Java ditemukan perusahaan workstation UNIX (Sparc) oleh satu tim dari perusahaan Sun Microsystem. Platform *independent* merupakan tujuan dari ciptakannya bahasa pemrogaraman Java ini (tanpa mengkompilasi ulang yang digunakan pada berbagai jenis *hardware* lainnya), Java dibangun berdasarkan bahasa C++ namun dibanding C++, Java memiliki objek yang murni dan lebih sederhana. Java merupakan bahasa pemrograman yang *simple* dengan rancangan yang dibuat dengan mudah dapat dipahami dan digunakan dengan simple oleh seorang *programmer* pemula (Suarga, 2009).

Sebuah *interpreter* Java melakukan eksekusi program Java dimana Java ialah bahasa pemrograman yang diinterpresikan yang harus melakukan sistem *host* dalam melakukan instalasi. Javac adalah hasil dari *compiler* Java yang di interpreter dengan membaca kode yang dimasukkan dalam sebuah *file* yang disebut dengan Intrepeter Java. File tersebut kemudian dibaca oleh *compiler*

dengan kata lain disebut dengan kode *byte* Java. Untuk menjadikan Java sebagai bahasa pemrograman yang *portable* dan lengkap awalnya dibuat untuk *tools* website pada Java yang menghasilkan dari *compiler* Javac yang mempunyai ekstensi *class* Java. Sedangkan yang memiliki fungsi sebagai input untuk *compiler* Java memiliki ekstensi Java *file* kode *byte*. Java memiliki banyak kemiripan dengan bahasa pemrograman *C* dan *C++* yang sering dihubungkan. Ketika Java versi pertama rilis sekitar pada tahun 1995 dan langsung sukses di masyarakat (Mustaqim, 2010).

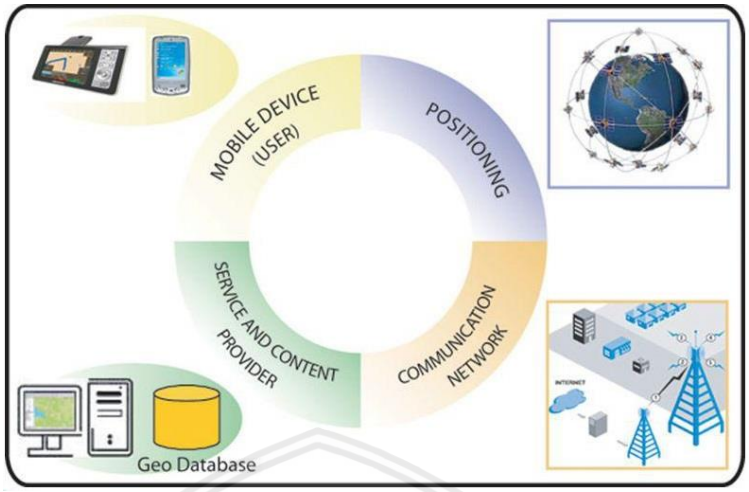
2.10 *Location Based Service*

LBS atau kepanjangan dari *Location Based Service* ialah sebuah layanan informasi yang dapat diakses melalui *smartphone* berbasis lokasi dengan menerapkan jaringan secara *online* serta dapat digunakan untuk mengetahui keberadaan pengguna sesuai dengan letak *smartphone* yang digunakan oleh pengguna sesuai dengan posisi pengguna. Android adalah salah satu *platform smartphone* yang dapat mengimplementasikan sistem *location based service*. Untuk itu, pengguna bisa mendapatkan informasi yang dibutuhkan dengan cara mereferensi posisi pengguna dengan memberi tahu penyedia layanan tersebut (Safaat, 2013). *Locaton based serviice* bisa dikatakan sebuah layanan dimana memiliki hubungan yang dipertemukan dalam tiga teknologi yaitu: *Mobile Devices*, *Internet Service,* dan *Geographic Information System* (Shiode, et al., 2004).

2.10.1 Komponen *Location Based Service*

Dalam LBS memiliki 5 komponen yang penting digambarkan pada Gambar 2.3. Komponen-komponen tersebut memiliki kegunaan (Steineger, et al., 2008):

1. *Mobile Devices* : Sebuah komponen *location based service* yang berfungsi untuk mendapatkan informsi yang dibutuhkan pengguna. Informasi yang didapatkan dapat berupa *text*, bentuk gambar, maupun suara.
2. *Communication Network* : Jaringan pada komunikasi dimana memiliki fungsi dalam mengirimkan informasi dan data yang dibutuhkan pengguna dan mengirimkannya kembali data dan informasi yang diminta kepada pengguna yang berasal dari *mobile* terminal ke *service provider*. Jaringan pada komunikasi seperti *Wireless Local Area Network* (WLAN), jaringan seluler (GSM, CDMA) atau *Wireles Wide Are Netwrk* (WWAN).
3. *Positioning Component* : Proses ini berfungsi untuk mengetahui letak posisi pengguna secara akurat dengan memproses layanan yang diberikan.
4. *Service and Application Provider* : Proses ini memberikan tawaran untuk bermacam-macam kualitas layanan dan memiliki tanggung jawab untuk memproses informasi dan data yang dibutuhkan pengguna.
5. *Data and Content Provider* : Proses pada tahap ini didapatkan dari *content provider* dan dari data yang digunakan untuk menyediakan layanan yang tidak selalu menyimpan semua informasi dan data.



Gambar 2. 2 Gambar Komponen Dasar *Location Based Service*

Sumber: Steiniger (2006)

2.11 *Global Positioning System* (GPS)

GPS atau *Global Positioning System* ialah alat yang digunakan dengan menggunakan bantuan satelit untuk alat navigasi. Sekitar berjumlah 4 sampai 12 buah yang dapat terhubung dengan satelit yang lainnya. Untuk memperhitungkan letak objek yang ada dibumi dengan satelit yang terhubung dapat menggunakan GPS. Ketinggian satelit sekitar 20.200 kilometer dan mempunyai bagian utama yaitu 24 buah satelit yang dapat mengorbit bumi, dibutuhkan sekitar kira-kira 12 jam untuk mengorbit bumi pada sebuah pembangkit energi listrik untuk memaksimalkan kinerjanya dimana satu satelit memiliki kecepatan 11.000 kilometer per jam dengan waktu yang cukup cepat. Untuk membangkitkan energi listrik perlu mengumpulkan energi matahari pada tiap satelit dalam panel-panel pada tiap satelit GPS (Ziad, 2013).

Untuk menjaga data agar lebih akurat yang dikirim ke *receiver* atau GPS berdasarkan akurasi data, sebuah satelit pada *receiver* tersebut harus selalu ada pada posisi orbit yang sesuai. Stasiun pengendali yang ada di bumi memantau keakuratan tiap satelit dengan letak orbit yang mengelilingi bumi. Dalam mencari posisi keakuratan pengguna yang didapat dari satelit berdasarkan informasi dilakukan dengan menggunakan perhitungan *triangulation* yang dilakukan oleh GPS *receiver*. Posisi pengguna yang sudah didapatkan dari proses perhitungan tersebut akan ditampilkan dalam peta eletronik (Mardani, 2012).

2.12 Google *Maps* API

Google *Maps* API merupakan layanan yang digunakan dengan memanfaatkan fitur dari Google Map. Google *Maps* disediakan oleh Google yang digunakan untuk melihat *maps* secara lengkap dan dapat digunakan secara tidak. Fitur yang dimiliki oleh Google *Maps* ialah dapat menemukan garis bujur dan lintang pada *maps*

untuk menemukan sebuah lokasi. Layanan Google *Maps* ini untuk digunakan dengan mengintegrasikan atau *embedded* ke aplikasi atau sistem yang akan dibuat. Untuk bahasa pemrograman yang digunakan pada Google *Maps* ini ialah bahasa pemrograman HTML atau *Hypertext Markup Languange* dengan ektensi *JavaScript* berbayar (Masykur, 2014).

Pada Google *Maps* dibutuhkan sebuah *key* atau sandi yang digunakan untuk bisa mengakses Google *Maps* API yakni dengan memanfaatkan API *key*, pada Google *Maps* memberikan fitur API atau *Application Programming Interface* yang berbentuk *library* untuk diintegrasikan dengan data yang dimiliki oleh pengguna. API atau *Application Programming Interface* itu sendiri merupakan *interface* atau antarmuka yang menghubungkan atau sebagai *interface* seperti aplikasi dan web yang dibuat. Sedangkan API *key* merupakan sandi atau *password* yang digunakan untuk melihat *maps* dari Google *Maps* yang berfungsi untuk mengintegrasikan pada aplikasi ataupun web agar mudah dikenal (Kindarto, 2008).

2.13 REST API

REST atau kepanjangan dari *Representational State Transfer* yang merupakan arsitektur pada perangkat lunak yang dijadikan sebagai media untuk komunikasi dan merupakan pengembangan atau pengimplementasian dari API. Dalam melakukan komunikasi dan pertukaran pada setiap data yakni menggunakan *Hypertext Transfer Protocol* atau biasa disebut dengan HTTP. Hasil yang dikirim pada REST API ini berformat JSON atau *JavaScript Object Notation* ataupun XML serta dapat diindentifikasikan dengan URI atau *Uniform Resource Identifier*. Format JSON atau XML tersebut berisi data yang akan digunakan kemudian klien mengakses *interface* yang sedang diproses. REST dapat juga digunakan sebagai antarmuka dari *Application Programming Interface* atau API untuk membantu mengakses *resource*. Terdapat *method-method* yang digunakan pada REST untuk mengaksess *resource* pada *server* dengan menggunakan HTTP *methode* yakni sebagai berikut (Raval & Gonsai, 2015).

a. GET merupakan *method* yang digunakan untuk membaca sumber data atau *resource*.
b. PUT merupakan *method* yang digunakan untuk memperbaharui sumber data yang sudah ada.
c. DELETE merupakan *method* yang digunakan untuk menghapus *resource*.
d. POST merupakan *method* yang digunakan untuk membuat sumber data atau *resource* baru.

2.14 Framework Laravel

Laravel merupakan framework atau bingkai kerja yang digunakan untuk pengembangan aplikasi yang berbasis web yang menerapkan fitur MVC (*Model View Controller*) yang menggunakan bahasa pemrograman PHP atau *Hypertext Prepocessor*. Fitur yang diberikan oleh framework laravel itu sendiri dapat membantu pengguna untuk membuat ataupun mengembangkan website dengan

memberikan kesederhanaan, kemudahan dan *fleksibel* dalam desainnya serta laravel merupakan framework yang mudah diakses.

Framework Laravel menerapkan fitur MVC (*Model View Controller*) dimana pola tersebut memungkinkan struktur dari kode program yang didapatkan memiliki susunan yang rapih dan jelas serta terstruktur. Pola MVC merupakan pola arsitektur sebagai pengembangan *software*. Pola MVC terdiri dari Model, View, dan Controller. Model digunakan sebagai tempat untuk memanipulasi data seperti logika data, mengelola data itu sendiri. View merupakan tampilan antarmuka yang berhubungan dengan *output* hasil antarmuka seperti gambar, teks, diagram maupun komponen lainnya. Sedangkan *Controller* merupakan penghubung antara model dan *view*, *controller* bertugas untuk menerima input yang diberikan oleh *view* kemudian meneruskannya ke model. *Controller* yang melakukan komunikasi antar model dan *view* dengan memanipulasi data pada model kemudian menampilkannya pada *view* berupa tampilan antarmuka (Caytiles & Lee, 2014).

2.15 MySQL

MySQL merupakan program *database server* yang dapat menerima dan mengirimkan data dengan cepat dengan menggunakan perintah-perintah SQL. SQL atau *Structured Query Languange* ialah salah satu bahasa pemrograman *database* yang popular dimana SQL merupakan bahasa *query* yang digunakan untuk membuat dan mengelola *database* berupa perintah-perintah DDL dan DML seperti menambah, mengubah, mencari, dan menghapus data dan lain sebagainya (Prasetya, 2010).

Terdapat 3 jenis bahasa yang dimiliki MySQL antara lain sebagai berikut:

1. DDL atau *Data Definition Language* ialah bahasa yang digunakan untuk membuat dan memanipulasi struktur *database* dan tabel.
2. DML atau *Data Manipulation Language* ialah bahasa yang digunakan untuk memanipulasi data pada sebuah *database*.
3. DCL atau *Data Control Language* ialah bahasa yang digunakan untuk menjaga keamanan *database*.

2.16 Unified Modeling Language

UML atau *Unified Modeling Language* merupakan bahasa pemodelan yang berisi notasi yang lengkap digunakan untuk membuat visualisasi model suatu sistem. Sistem biasanya berisi mengenai informasi dan fungsi tetapi secara normal dapat digunakan untuk memodelkan sistem komputer (Yasin, 2012).

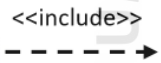
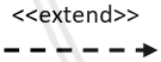
UML disebut dengan bahasa pemodelan bukan metode dimana bahasa pemodelan yang digunakan oleh sistem atau perangkat lunak yang berparadigma berorientasi objek. Bahasa pemodelan merupakan bagian tepenting dari metode. UML ialah bahasa pemodelan standar untuk penulisan *blueprint software* yang digunakan untuk memvisualisasikan, menspesifikasikan, pembentukan, dan pendokumentasian *tools* dari sistem perangkat lunak. Dengan adanya UML dapat

memberikan bahasa pemodelan visual yang ekspresif untuk mengembangkan sistem dan dapat saling menukar model dengan mudah (Rumbaugh, et al., 2005).

2.16.1 *Use Case Diagram*

Use Case Diagram merupakan diagram yang menggambarkan fungsionalitas dari sebuah sistem yang dapat menjelaskan mengenai deskripsi lengkap tentang interaksi yang terjadi pada aktor dengan sistem atau perangkat lunak. Sedangkan *Use case* ialah deskripsi tentang bagaimana sistem atau perangkat lunak berperilaku terhadap aktor. Sebuah *use case* dapat merepresentasikan sebuah interaksi antara aktor dengan sistem. Pada *use case diagram* terdapat beberapa istilah seperti aktor, *use case*, dan *use case relationship* (Yasin, 2012).

Tabel 2. 2 Penjelasan Simbol *Use Case Diagram*

No	Gambar	Nama	Keterangan
1.		*Actor*	Seseorang yang berinteraksi dengan sistem. Aktor dapat direpresentasikan oleh manusia, atau perangkat keras eksternal lainnya
2.		*Include*	*Include* menunjukkan *use case* seluruhnya adalah fungsionalitas dari *use case* lainnya. Yang bertujuan untuk penggunaan kembali sebuah *use case*
3.		*extend*	*Extend* ialah sebuah hubungan yang dapat memodifikasi *use case* dasar dengan menambah fungsional dari *use case* lainnya jika suatu kondisi terpenuhi.
4.		*Association*	Sebuah relasi atau koneksi antara aktor dan *use case*.
5.		*Use Case*	*Use case* merupakan spesifikasi urutan aksi-aksi yang ditampil sistem.

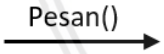
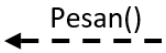
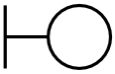
Sumber: Sukamto & Shalahudin (2014)

2.16.2 *Sequence Diagram*

Sequence Diagram merupakan diagram yang menggambarkan interaksi antar objek didalam maupun di sekitar sistem yang berupa pesan yang disusun kedalam suatu urutan waktu. *Sequence diagram* digunakan untuk menunjukkan rangkaian tahap demi tahap berupa pesan yang dikirimkan antar objek dan juga interaksi

antar objek dari suatu kegiatan untuk menghasilkan *output* tertentu. *Sequence diagram* sangat berhubungan dengan *use case diagram*, dimana *sequence diagram* ialah gambaran dari langkah-langkah yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case diagram*. Pada satu *use case diagram* akan menjadi satu *sequence diagram* (Sommervile, 2003).

Tabel 2. 3 Penjelasan Simbol *Sequence Diagram*

No	Gambar	Nama	Keterangan
1.		*Actor*	*Actor* biasa digunakan untuk mendefinisikan seorang pengguna yang mempunyai fungsi untuk mengirimkan dan/atau menerima pesan.
2.		*Object*	*Object* merupakan sesuatu yang digunakan secara urut untuk mengirimkan dan/atau menerima pesan.
3.		*Lifeline*	Menggambarkan kehidupan objek selama urutan.
4.		*Activation*	Terletak diatas *lifeline* yang berbentuk persegi panjang. Menandakan ketika suatu objek mengirim dan/atau menerima pesan.
5.		*Objek Message*	Objek yang memuat informasi-informasi untuk dikirim ke objek lainnya.
6.		*Return Message*	Menghasilkan suatu kembalian dari objek tertentu, arah panah mengarah ke objek yang menerima.
7.		*Message to self*	Menggambarkan pesan atau hubungan objek itu sendiri.
8.		*Boundary*	Digunakan untuk mengambarkan sebuah *form.*
9.		*Control Class*	*Control class* digunakan untuk menghubungkan *boundary* dengan tabel.

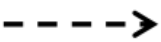
10.		*Entity Class*	Menggambarkan hubungan kegiatan yang akan dilakukan.

Sumber: Sukamto & Shalahudin (2014)

2.16.3 *Class Diagram*

Class Diagram ialah diagram yang mampu menjelaskan tipe dari objek sistem dan hubungan dengan objek lainnya dimana objek adalah nilai dari setiap atribut dan *entity class*. *Class diagram* merupakan diagram yang menggambarkan mengenai struktur dan deskripsi dari *class*, *package,* objek, dan hubungan satu dengan yang lainnya antara lain *composition*, asosiasi, pewarisan, dan lain sebagainya. *Class* itu sendiri ialah suatu hal yang menjadi *point* dari suatu permasalahan (Lethbridge & Laganiere, 2005).

Tabel 2. 4 Penjelasan Simbol *Class Diagram*

No	Gambar	Nama	Keterangan
1.		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk *(ancestor)*.
2.		*Nary Association*	*Symbol* yang digunakan sebagai upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3.		*Class*	*Class* menjelaskan mengenai atribut, operasi, dan *instance* dimana sebuah tempat untuk dibuatnya objek.
4.		*Collaboration*	Penjelasan dari urutan aktifitas yang ditampilkan sistem yang menghasilkan sesuatu yang terukur bagi suatu *actor*.
5.		*Realization*	Operasi yang dilakukan oleh suatu objek.
6.		*Dependency*	*Dependency* ialah hubungan dimana menunjukkan *class* mandiri yang bergantung dengan *class* lain yang tidak mandiri.

7.	———	*Association*	Merupakan hubungan yang menggambarkan antara objek satu dan objek lainnya.

Sumber: Sukamto & Shalahudin (2014)

2.17 Pengujian Perangkat Lunak

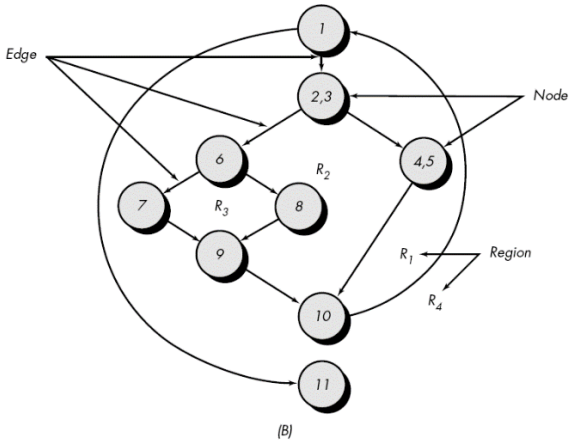
Pengujian perangkat lunak merupakan tahapan akhir dari suatu pengembangan perangkat lunak setelah selesai dibangun. Pengujian pada perangkat lunak dilakukan untuk menemukan *bug* atau *error* yang kemungkinan terjadi karena adanya kesalahan dari perangkat lunak. Tahapan pengujian perangkat lunak bertujuan untuk memastikan kelengkapan dari kebutuhan fungsional apakah sudah sesuai dengan tahapan analisis kebutuhan. Ketika sebuah perangkat lunak sudah digunakan oleh pengguna, perangkat lunak tersebut harus terbebas dari adanya kesalahan untuk itu perangkat lunak harus diuji terlebih dahulu untuk menemukan sebanyak mungkin kesalahan pada perangkat lunak sebelum diberikan kepada pengguna (Pressman, 2010).

Dalam penelitian ini akan menggunakan tiga pengujian untuk menguji aplikasi yang telah dibuat yakni menggunakan metode *whitebox testing* dengan pengujian yang digunakan ialah *basis path testing* dan pengujian integrasi dan metode *blackbox testing* dengan pengujian yang digunakan yakni pengujian validasi, dan pengujian terakhir yakni pengujian *usability*. Berikut akan menjelaskan masing-masing pengujian yang akan digunakan pada penelitian ini.

2.17.1 Whitebox Testing

Whitebox Testing atau pengujian kotak putih yang merupakan salah satu metode pengujian yang dilakukan untuk mendapatkan kasus uji dengan menggunakan algoritme atau kode program dalam pengujiannya. Pengujian ini dilakukan untuk mengetahui dan memastikan kesalahan-kesalahan yang ada pada algoritme atau kode program yang digunakan dalam membangun sistem (Argawal, et al., 2010).

Whitebox testing memiliki beberapa jenis pengujian, salah satu metode pengujian tersebut ialah *basis path testing*. *Basis path testing* merupakan pengujian *whitebox* yang bertujuan untuk mendapatkan *cyclomatic complexity* atau kompleksitas logika pada suatu sistem merupakan rumus perhitungan dalam menentukan tingkat kompleksitas suatu sistem. *Cyclomatic complexity* menjadi acuan untuk menentukan jalur program. Kompleksitas logika diperoleh berdasarkan diagram alir kode program atau algoritme atau yang biasa disebut dengan *flow graph*. Untuk melakukan *basis path testing* dengan membuat algoritme berdasarkan kode program, kemudian selanjutnya membuat *flow graph*, pada *flow graph* ini akan didapatkan *node-node flow graph*, setelah itu membuat *cyclomatic complexity* yang digunakan untuk menentukan *independent path*. *Independent path* digunakan untuk mencari jumlah jalur dari sebuah *flow graph* (Pressman, 2010). Berikut merupakan gambar *flow graph* yang ada pada *basis path testing* dapat dilihat pada Gambar 2.3.



Gambar 2. 3 *Flow Graph Basis Path Testing*

Sumber: Pressman (2010)

2.17.2 *Blackbox Testing*

Blackbox Testing atau juga disebut dengan pengujian tingkah laku atau *Behavioral Testing* merupakan pengujian yang dilakukan pada kebutuhan fungsional dari suatu sistem. Pengujian ini dilakukan mengacu berdasarkan *scenario* yang telah dibuat untuk prosedur uji. Pada pengujian *blackbox* memungkinkan *developer* yang membuat sistem untuk mendapatkan kondisi pada masukan yang akan melakukan eksekusi kebutuhan pada sistem. Sehingga *blackbox testing* dilakukan pengujian pada tampilan *layout* sistem untuk mengetahui dari kondisi *input* maupun *output* sudah sesuai dengan sistem yang diinginkan (Pressman, 2010).

Blackbox testing dan *whitebox testing* merupakan pengujian yang saling berkaitan satu sama lain dimana pengujian *blackbox* digunakan sebagai pelengkap dari pengujian *whitebox* melainkan bukan untuk pengganti dari pengujian *whitebox*, untuk itu jika melakukan *whitebox testing* pada suatu sistem maka perlu juga melakukan *blackbox testing* agar melengkapi proses pengujian sistem. Yang dilakukan untuk melakukan pengujian *blackbox* yakni harus terlebih dahulu membuat *scenario* dimana *scenario* tersebut dibuat dengan *scenario* dengan kondisi benar dan kondisi salah, dengan itu *tester* dapat mengetahui hasil yang didapatkan dari kebutuhan fungsional tersebut sesuai atau tidak dengan yang diinginkan. *Blackbox testing* melakukan pengujian kepada seluruh kebutuhan-kebutuhan fungsional maupun kebutuhan non fungsional dari analisis kebutuhan yang sudah didefinisikan.

2.17.3 Pengujian *Usability*

Pengujian *usability* atau pengujian kebergunaan merupakan pengujian yang dilakukan untuk memberikan penilaian oleh pengguna mengenai evaluasi hasil akhir dari sebuah perangkat lunak. Pengujian *usability* dilakukan dengan tujuan untuk mengetahui kemudahan perangkat lunak yang sudah diimplementasikan, seberapa besar kemudahan interaksi antarmuka yang dapat digunakan oleh *user*

terhadap sistem. Tujuan dari mengetahuinya kemudahan dari sistem ialah untuk mengetahui apakah sistem tersebut memiliki sifat efektif untuk penggunanya. Melakukan pengujian ini dilakukan dengan tujuan untuk memastikan bahwa sistem sudah memenuhi salah satu kebutuhan non fungsional yang sudah dijelaskan pada bab perancangan yakni kebutuhan non fungsional yang ada pada aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang mengenai *usability* sistem (Santoso, et al., 2015).

Untuk melakukan pengujian *usability* ini menggunakan sebuah metode kuesioner. Kuesioner ini nantinya akan diberikan kepada pengguna aplikasi untuk kasus ini yakni memberikan kuesioner kepada pengguna atau masyarakat Kota Malang terhadap aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang. Terdapat banyak jenis kuesioner untuk pengujian *usability* salah satunya ialah SUS atau *System Usability Scale* dimana metode kuesioner SUS ini yang akan digunakan pada kasus ini. *System Usability Scale* merupakan metode kuesioner untuk pengujian usability yang bersifat valid diantara jenis metode kuesioner lainnya (Santoso, et al., 2015).

Untuk itu, pengujian *usability* ini dilakukan kepada masyarakat Kota Malang dengan menggunakan metode kuesioner atau disebut dengan *SUS* atau *System Usability Scale*. Berikut merupakan parameter dari kuesioner yang digunakan pada aplikasi pengaduan masyarakat yang terdiri dari lima parameter, yakni sebagai berikut:

1. Learnabilitas (*Learnability*)
2. Efisiensi (*Eficiency*)
3. Memorabilitas (*Memorability*)
4. Kepuasan (*Satisfaction*)
5. Efektifitas (*Effectiveness*)

2.17.4 *System Usability Scale* (SUS)

System Usability Scale atau SUS merupakan metode kuesioner yang digunakan untuk melakukan pengujian *usability*. Pada kuesioner SUS ini terdapat beberapa jenis pernyataan yang digunakan pada kueioner yakni pernyatan negatif dan positif. Untuk pernyataan pertama yakni pernyataan positif akan menjadi pernyataan pada kuesioner yang terletak di pernyataan bernomor ganjir begitupun sebaliknya, untuk pernyataan kedua atau pernyataan ganji akan menjadi pernyataan pada kuesioner yang terletak di pernyataan bernomor genap (Sauro, 2015).

Langkah-langkah dan perhitungan yang digunakan pada kuesioner SUS ini yaitu seperti berikut.

1. Menghitung *score* pernyataan yang memiliki ketentuan sebagai berikut:
    a. Pernyataan bernilai positif atau pernyataan yang dimasukkan pada pernyataan bernomor ganjil memiliki rumus seperti berikut:

Score = x – 1

b. Pernyataan bernilai negatif atau pernyataan yang dimasukkan pada pernyataan bernomor genap memiliki rumus seperti berikut:

Score = 5 – x

Keterangan:

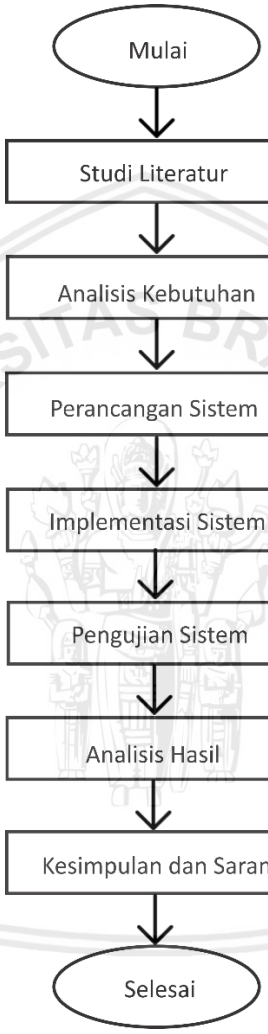
x = nilai dari responden yang mengisi kuesioner

2. Menjumlahkan *score* yang didapatkan pada setiap pernyataan sehingga didapat *score* akhir SUS ialah hasil dari menjumlahkan *score* setiap pernyataan dikali dengan 2.5.

Terdapat beberapa kategori yang digunakan untuk mengetahui hasil akhir SUS apakah sistem atau aplikasi mudah digunakan dan mudah diterima oleh penggunanya pada pengujian *usability* ini. *Score* SUS mulai dari 0-50 memiliki arti "Tidak dapat diterima" atau termasuk kategori "*Not Acceptable*" oleh penggunanya. *Score* SUS 51-70 memiliki arti "Sebagian yang dapat menerima" atau termasuk kategori "*Marginal*" oleh pengguna. Dan untuk *score* SUS 71-100 memiliki arti "Dapat Diterima" atau masuk kedalam kategori "*Acceptabl*e" oleh penggunanya (Brooke, 2013).

BAB 3 METODOLOGI PENELITIAN

Bab metodologi membahas mengenai penjelasan metodologi yang berkaitan dengan penelitian. Penjelasan tersebut antara lain ialah studi literatur, analisis kebutuhan, pengumpulan data, perancangan sistem, implementasi, pengujian, analisis serta penarikan kesimpulan dan saran. Langkah-langkah metodologi dapat dijelaskan pada diagram alir seperti pada Gambar 3.1.



Gambar 3. 1 Gambar Diagram Alir Penelitian

3.1 Studi Literatur

Studi literatur dapat dijadikan sebagai pandangan untuk menyelesaikan permasalahan yang sesuai berdasarkan dasar teori dengan topik yang ada pada penelitian yang sedang dilakukan. Kegiatan studi literatur ini mengkaji teori yang dilakukan baik dari sumber buku, jurnal, atau pada sebuah penelitian yang pernah dilakukan sebelumnya, sesuai dengan topik pembicaraan pada penelitian yang sedang dilakukan. Dasar teori yang terdapat pada penelitian ini adalah sebagai berikut:

1. Dinas Perhubungan
2. Keluhan dan Pengaduan Masyarakat
3. Pelaporan Keluhan Masyarakat
4. Rekayasa Perangkat Lunak
5. *Software Development Life Cycle*
6. *Smartphone*
7. Android
8. Java
9. *Location Based Service*
10. *Global Positioning System* (GPS)
11. Google *Maps* API
12. REST API
13. Framework Laravel
14. Pemodelan *Object Oriented*
    a. *Use Case Diagram*
    b. *Sequence Diagram*
    c. *Class Diagram*
15. Pengujian Perangkat Lunak
    a. *Whitebox Testing*
    b. *Blackbox Testing*
    c. Pengujian *Usability*
    d. *System Usability Scale* (SUS)

3.2 Analisis Kebutuhan

Tahapan analisis kebutuhan merupakan tahapan yang dibutuhkan untuk melakukan pengumpulan data dan informasi untuk memperoleh informasi yang digunakan untuk kebutuhan pada pengembangan sistem. Tujuan menganalisis kebutuhan ini agar pengimplementasian sistem yang diinginkan dapat berjalan dengan baik pada setiap kebutuhan fungsional maupun kebutuhan non fungsional, proses analisis kebutuhan ini sangat perlu diperlukan pada suatu penelitian. Aspek yang diperlukan dalam analisis kebutuhan seperti rencana dari proyek yang akan dikerjakan, mengatur strategi dalam mengatasi resiko dari management penelitian dan dapat mengendalikan perubahan pada setiap kebutuhan yang menjadi kemungkinan akan terus bertambah. Tahapan analisis kebutuhan ini akan menganalisis permasalahan mengenai pengaduan masyarakat pada Dinas Perhubungan Kota Malang, untuk itu pada penelitian ini akan melakukan analisis kebutuhan pada Dinas Perhubungan Kota Malang. Proses analisis kebutuhan ini dilakukan dengan melakukan proses elisitasi kebutuhan, spesifikasi kebutuhan, dan pemodelan kebutuhan.

3.2.1 Elisitasi Kebutuhan

Elisitasi kebutuhan merupakan tahapan yang dilakukan dengan mengumpulkan kebutuhan-kebutuhan yang dibutuhkan oleh sistem yang akan dikembangkan yang diambil dari stakeholder atau pemangku kebutuhan dimana yang memiliki keterkaitan antara permasalahan yang terjadi dengan sistem yang

akan dikembangkan, pada penelitian ini yang berperan sebagai pemangku kebutuhan yakni petugas Dinas Perhubungan Kota Malang dan masyarakat Kota Malang yang menggunakan sistem dan sebagai ruang lingkup dari penelitian. Agar semua kebutuhan pada sistem dapat dijalankan dengan baik maka perlu dilakukan pengumpulan data yang dilakukan berdasarkan kebutuhan fungsional dan kebutuhan non fungsional pada sistem. Pengumpulan data yang dilakukan pada penelitian ini ialah dengan menggunakan metode wawancara.

Wawancara merupakan metode pengumpulan data yang dilakukan agar memperoleh data dan informasi yang dibutuhkan dalam memenuhi kebutuhan pada sistem baik kebutuhan fungsional dan non fungsional. Wawancara dilakukan kepada masyarakat Kota Malang dan petugas Dinas Perhubungan Kota Malang. Informasi yang akan digali mengenai permasalahan yang terjadi di lapangan yang pernah dialami masyarakat terkait pelaporan pada Dinas Perhubungan Kota Malang, tugas pokok dari Dinas Perhubungan Kota Malang, data dan informasi tentang tata cara untuk menyampaikan pengaduan yang berupa keluhan oleh masyarakat Kota Malang, data tentang keluhan yang sering dilaporkan oleh masyarakat, data tentang langkah atau tindakan yang dilakukan Dinas Perhubungan Kota Malang dalam menindaklanjuti keluhan yang dilaporkan serta hal lainnya yang merupakan kegiatan yang dilakukan dan berlangsung pada Dinas Perhubungan Kota Malang.

3.2.2 Spesifikasi Kebutuhan

Setelah melakukan tahap elisitasi kebutuhan yakni pengumpulan data yang dilakukan dengan wawancara, tahap selanjutnya ialah spesifikasi kebutuhan. Spesifikasi kebutuhan ialah tahapan analisis kebutuhan yang dilakukan setelah proses elisitasi kebutuhan. Tujuan adanya spesifikasi kebutuhan untuk menjelaskan secara *detail* kebutuhan-kebutuhan yang harus ada pada sistem yang sudah dijelaskan pada tahap elisitasi kebutuhan. Tahapan spesifikasi kebutuhan dilakukan pada aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang untuk mendukung kebutuhan fungsional maupun non fungsional pada sistem yang akan dikembangkan dapat berjalan dengan baik sesuai dengan yang diinginkan dimana proses ini dilakukan untuk memperoleh kebutuhan-kebutuhan pada aplikasi pengaduan masyarakat baik untuk kebutuhan fungsional maupun kebutuhan non fungsional.

3.2.3 Pemodelan Kebutuhan

Pemodelan kebutuhan ialah salah satu tahapan dari analisis kebutuhan. Pemodelan kebutuhan dilakukan setelah melakukan tahap spesifikasi kebutuhan. Tahap pemodelan kebutuhan dilakukan untuk memastikan kebutuhan-kebutuhan yang sudah didapatkan dan dijelaskan pada tahapan sebelumnya benar adanya dan kebutuhan tersebut sudah jelas. Setiap kebutuhan fungsional maupun kebutuhan non fungsional yang sudah dijelaskan selanjutnya dimodelkan dengan mengintrepretasikan kebutuhan tersebut kedalam bentuk pemodelan kebutuhan dimana pemodelan kebutuhan yang digunakan pada penelitian aplikasi pengaduan masyarakat ini dengan menggunakan pemodelan berorientasi *object*,

oleh karena itu pemodelan kebutuhan yang sudah didefinisikan dengan menggunakan *use case diagram* dan *use case scenario*.

3.3 Perancangan Sistem

Setelah melakukan analisis kebutuhan tahapan selanjutnya pada metodologi penelitian ialah merancang sistem yang akan dibangun. Merancang sistem yang akan dibuat mengacu pada tahap analisis yang telah dilakukan sebelumnya. Terdapat 3 perancangan yang akan dibangun yaitu perancangan basis data, perancangan sistem, dan perancangan antar muka. Perancangan basis data dengan membuat *Entity Relationship Diagram* (ERD). ERD adalah cara yang digunakan untuk merepresentasikan data atau objek yang dibuat yang terdiri entitas, atribut, dan hubungan antar entitas.

Pada pencangan sistem menggunakann pendekatan *object rriented*. Tahap perancangan sistem pada pengembangan sistem *object oriented* adalah pembuatan diagram-diagram perancangan seperti *sequence diagram*, dan *class diagram*. Hasil dari rancangan yang dibuat harus sesuai dengan langkah kerja dari sistem yang dikembangkan. Sedangkan untuk perancangan antar muka ialah pembuatan *wireframing* untuk langkah awal seperti pembuatan tampilan aplikasi tetapi belum masuk ke dalam tahap pembuatan *mock up*, kemudian langkah selanjutnya ialah pembuatan *mock up* dimana pada tahap ini memulai untuk menggunakan warna dan gambar pada rancangan tampilan. Langkah selanjutnya dari tahap perancangan antar muka adalah mengimplementasikan rancangan *mock up* ke *layout* XML Android. Pada tahapan perancangan sistem dilakukan beberapa perancangan yang terdiri dari perancangan *sequence diagram*, *class diagram*, perancangan basis data, perancangan komponen berupa perancangan algoritme, dan perancangan antarmuka.

3.3.1 *Sequence Diagram*

Sequence Diagram digunakan untuk menunjukkan rangkaian tahap demi tahap berupa pesan yang dikirimkan antar objek dan juga interaksi antar objek dari suatu kegiatan untuk menghasilkan *output* tertentu. *Sequence diagram* merupakan diagram yang menggambarkan interaksi antar objek yang berupa pesan yang disusun kedalam suatu urutan waktu. *Sequence diagram* merupakan salah satu perancangan yang masuk ke dalam *object oriented*. Sehingga pada perancangan ini terlihat interaksi antar objek.

3.3.2 *Class Diagram*

Class Diagram ialah diagram yang mampu menjelaskan tipe dari objek sistem dan hubungan dengan objek lainnya dimana objek adalah nilai dari setiap atribut dan *entity class*. *Class diagram* pula merupakan salah satu perancangan yang masuk ke dalam perancangan berorientasi *object oriented*. Dengan dirancangnya *class diagram* dapat dilihat *class-class* yang berisi *method* yang digunakan dalam membangun aplikasi pengaduan masyarakat dimana *class-class* tersebut saling berhubungan.

3.3.3 Perancangan Basis Data

Aplikasi pengaduan masyarakat menerapkan perancangan basis data ke dalam *Entity Relationship Diagram* atau ERD dimana dengan melakukan perancangan menggunakan ERD ini tabel-tabel yang dibutuhkan dalam membangun aplikasi ini diimplementasikan dengan baik pada sistem. Perancangan yang digambarkan pada perancangan basis data ini ialah *entity* serta atribut yang ada dan tersimpan berdasarkan pada data yang ada di aplikasi pengaduan masyarakat yang sudah didapatkan dari proses pengumpulan data yang dilakukan pada tahapan analisis kebutuhan sebelumnya.

3.3.4 Perancangan Algoritme

Perancangan komponen dilakukan pada tahapan perancangan sistem untuk aplikasi pengaduan masyarakat ini. Perancangan komponen dilakukan untuk menjelaskan algoritme yang sesuai dengan kode program dimana sebelum mengimplementasikan kode program tersebut harus terlebih dahulu melakukan perancangan algoritme ini. Perancangan algoritme yang digunakan pada penelitian ini akan dirancang dalam *pseudocode*. *Pseudocode* akan menjelaskan gambaran algoritme ataupun kode program yang digunakan dalam membangun aplikasi pengaduan masyarakat. Sehingga dengan adanya perancangan *pseudocode* ini dapat dijadikan sebagai acuan atau gambaran dalam membuat kode program yang dilakukan pada tahap implementasi sistem.

3.3.5 Perancangan Antarmuka

Tampilan antarmuka ialah gambaran tampilan dari sebuah sistem. Tahapan awal dari perancangan antarmuka ini yakni pembuatan *wireframing* untuk langkah awal seperti pembuatan tampilan aplikasi tetapi belum masuk ke dalam tahap pembuatan *mock up*, kemudian langkah selanjutnya ialah pembuatan *mock up* dimana pada tahap ini memulai untuk menggunakan warna dan gambar pada rancangan tampilan. Langkah selanjutnya dari tahap perancangan antar muka adalah mengimplementasikan rancangan *mock up* ke *layout* pada IDE Android Studio dengan menggunakan format XML Android.

3.4 Implementasi

Tahap implementasi mengacu pada perancangan sistem yang telah dibuat. Perancangan yang telah selesai dibuat merupakan tahapan yang akan dilakukan implementasi dimana implementasi ini harus sesuai dengan rancangan yang dibuat berdasarkan dengan kebutuhan fungsional dan non fungsional agar aplikasi sesuai yang diinginkan.

Implementasi sistem ini dibagi menjadi dua yaitu untuk *user* atau pengguna yang digunakan oleh masyarakat sebagai pelapor serta oleh pegawai Dinas Perhubungan Kota Malang sebagai admin:

1. Implementasi sistem untuk pelapor atau masyarakat dilakukan dalam sebuah aplikasi *mobile* berbasis Android. Untuk dapat

mengimplementasikan kedalam aplikasi *mobile* diperlukan suatu alat atau piranti dalam membangun aplikasi tersebut, yaitu Android studio. Android *studio* adalah editor untuk membuat aplikasi untuk *platform* Android dengan bahasa pemrograman yang digunakan adalah Java. Untuk mengimplementasikan basis data menggunakan basis data MySQL, dan menggunakan fitur *LBS*. *Location based service* ialah sebuah layanan untuk mengetahui posisi seseorang berdasarkan keberadaan *smartphone* yang digunakan dengan berbasis lokasi.

2. Implementasi sistem untuk admin atau pegawai Dinas Perhubungan yaitu berbasis web dengan menggunakan *framework laravel* serta *tools* lain yang mendukung proses implementasi.

3.5 Pengujian

Untuk melakukan proses apakah tahapan analisis kebutuhan dan merancang aplikasi sudah terpenuhi yang harus dilakukan ialah melakukan uji coba atau pengujian untuk mengetahui apakah aplikasi tersebut terdeteksi *bug* atau tidak serta mengetahui hasil implementasi yang dibuat. Dengan tujuan untuk meminimalisir *bug* yang ada sebelum dibagikan ke masyarakat karena jika sistem mengalami kegagalan dapat mengakibatkan kerugian. Dalam pengujian aplikasi *mobile* pengaduan masyarakat menggunakan pengujian dengan metode *blackbox testing*, *whitebox testing,* dan juga pengujian *usability.* Untuk pengujian *whitebox* menggunakan metode pengujian unit dan pengujian integrasi, untuk pengujian *blackbox* menggunakan metode pengujian validasi serta pengujian *usability* menggunakan metode kuesioner SUS atau *System Usability Scale*.

Untuk memastikan apakah semua kebutuhan telah terpenuhi pada sistem yang sudah selesai dibangun perlu dilakukan pengujian dengan *blackbox testing*. *Blackbox testing* akan memastikan kebutuhan sistem yang telah dijelaskan pada tahap analisis kebutuhan sudah diimplementasikan dan sesuai dengan perancangan dengan menggunakan pengujian validasi. Pengujian selanjutnya yang digunakan pada penelitian adalah pengujian dengan metode *whitebox testing* yaitu *basis path testing* dan *integrase testing* dimana metode *basis path testing* ini digunakan untuk mengecek jalur algoritme untu mendapatkan *independent path* dan metode *cyclomatic complexity* untuk mengetahui tingkat kerumitan sistem.

Setelah aplikasi sudah selesai dibuat langkah selanjutnya melakukan pengujian untuk menguji aplikasi tersebut mengenai tingkat kemudahan aplikasi atau bisa disebut dengan pengujian *usability* dengan menggunakan metode kuesioner, penulis melakukan pengujian kuesioner dengan memberi kuesioner kepada 20 orang atau yang disebut dengan responden. Kuesioner yang diberikan terdapat nilai dari 1 sampai 5 dimana yang biasa disebut dengan skala *likert*, responden dapat mengisi atau memberikan penilaian dari 1 sampai 5 mengenai kemudahan aplikasi yang dibuat, pengujian yang didefinisikan tersebut merupakan pengujian usabilitas.

3.6 Analisis Hasil

Setelah dilakukan pengujian langkah selanjutnya ialah tahap menganalisis. Pada tahap ini menganalisis hasil berdasarkan pengujian yang telah dilakukan. Tahap analisis merupakan pernyataan dari sebuah hasil penelitian yang digunakan untuk menganalisis penelitian yang berupa implementasi dari hasil yang sesuai dengan tujuan penelitian serta analisis hasil mampu memberikan jawaban dari permasalahan pada rumusan maasalah.

Analisis hasil telah mendapatkan jawaban permaslahan rumusan masalah, analisis yang harus dilakukan penulis ialah:

1. Bagaimana memberikan layanan dan fasilitas pada aplikasi *mobile* untuk pengaduan masyarakat tentang sarana umum lalu lintas guna membantu masyarakat untuk menyampaikan keluhan pada Dinas Perhubungan Kota Malang?
2. Bagaimana menerapkan fitur *Location Based Service* untuk mengetahui letak posisi yang akurat sesuai dengan keberadaan masyarakat yang melapor pada sebuah perangkat bergerak?
3. Bagaimana tingkat kemudahan aplikasi *mobile* pengaduan masyarakat pada Dinas Perhubungan Kota Malang berbasis Android yang mudah dimengerti dan mudah digunakan oleh pengguna?

3.7 Kesimpulan dan Saran

Tahap akhir dari penelitian setelah terselesaikannya penelitian tersebut mulai dari studi literatur, mengumpulkan data yang dibutuhkan, menganalisis kebutuhan, perancangan sistem, pengimplementasian, pengujian sistem, dan analisis hasil maka diambil kesimpulan serta dapat memberikan saran dari hasil yang diperoleh pada penelitian yang digunakan untuk menjawab permasalahan pada penelitian. Kesimpulan didapat dari hasil analisis pengujian yang telah dilakukan. Kesimpulan juga dapat memberikan saran dimana penyampaian saran dapat dilakukan untuk memberikan suatu pertimbangan dan pandangan untuk penelitian berikutnya dalam memperbaiki kekurangan dan kesalahan yang ada.

BAB 4 ANALISIS KEBUTUHAN SISTEM

Bab ini membahas mengenai analisis kebutuhan yang ada pada sistem pengaduan masyarakat pada Dinas Perhubungan Kota Malang dengan menggunakan fitur *location based service* berbasis Android. Analisis kebutuhan merupakan tahapan yang pertama kali dilakukan dalam mengembangkan sebuah sistem, tahapan analisis kebutuhan analisis dilakukan untuk menentukan kebutuhan fungsional atau kebutuhan yang harus ada pada sistem maupun kebutuhan non-fungsional pada sistem.

Pada bab analisis kebutuhan sistem ini menjelaskan mengenai siapa yang berperan dalam sistem pengaduan masyarakat pada Dinas Perhubungan Kota Malang. Selain itu, pada bab ini dibuat pemodelan-pemodelan kebutuhan yang digunakan untuk membantu dalam memahami secara detil mengenai kebutuhan-kebutuhan yang ada pada sistem dimana pemodelan tersebut seperti *use case* diagram dan *use case scenario*.

4.1 Analisis Kebutuhan

Pada tahapan analisis kebutuhan ini menjelaskan mengenai gambaran umum pada sistem, identifikasi aktor, dan penjelasan mengenai kebutuhan pada sistem. Analisis kebutuhan bertujuan untuk menjelaskan dan menggambarkan kebutuhan-kebutuhan yang harus ada pada sistem guna memenuhi kebutuhan pengguna.

4.1.1 Penggalian Kebutuhan Sistem

Hasil wawancara yang telah dilakukan untuk mengidentifikasi permasalahan mengenai kurangnya sarana untuk menyampaikan keluhan maka didapatkan hasil bahwa Dinas Perhubungan Kota Malang sudah menerapkan sistem informasi untuk pekerjaannya, salah satunya ialah sudah mengembangkan sistem informasi untuk pengaduan masyarakat dengan menggunakan website dan sosial media. Namun media tersebut kurang efektif dalam melakukan pengaduan dikarenakan kurangnya respon yang cepat dari pihak Dinas Perhubungan Kota Malang. Pada website dan sosial media tersebut hanya terdapat deskripsi dari masalah yang terjadi tanpa mengetahui foto dan lokasi yang terpercaya. Alasan lain yang menjadikan kurang efektif ialah karena kurang terkoordinasi dengan baik pengaduan tersebut dan dari banyaknya pengaduan tidak bisa semua ditindaklanjuti karena kurangnya informasi yang tidak lengkap dan tidak terpercaya. Sedangkan pada Dinas Perhubungan Kota Malang sendiri hanya memiliki satu admin yang mengkoordinasi mengenai pengaduan masyarakat ini dimana admin tersebut merupakan bukan pekerjaan utama yang dilakukan melainkan pekerjaan sampingan yang mana seorang admin tersebut memiliki dua pekerjaan sekaligus, permasalahan tersebut merupakan faktor internal kurang cepat dan tanggapnya keluhan di tindaklanjuti karena kurangnya sumber daya manusia pada Dinas Perhubungan Kota Malang.

Dari permasalahan-permasalahan yang terjadi, penulis mengetahui bahwa permasalahan terjadi pada faktor eksternal ialah proses penyampaian keluhan tanpa informasi yang jelas seperti tidak adanya foto dan lokasi keluhan yang terpercaya, untuk itu penulis mengembangkan suatu aplikasi untuk Pengaduan Masyarakat yang dapat memasukkan foto dan lokasi kejadian dalam satu aplikasi untuk memudahkan tugas dari seorang admin Dinas Perhubungan Kota Malang. *stakeholder* disana menginginkan aplikasi untuk pengaduan masyarakat selain dengan memasukkan deskripsi dari keluhan bisa dengan memasukkan foto dan lokasi tempat kejadian agar memudahkan pihak Dinas Perhubungan yang bertugas dilapangan untuk menindaklanjuti keluhan yang dilaporkan dimana dengan adanya informasi yang jelas keluhan dari masyarakat akan ditindaklanjuti dengan cepat dan tanggap. Dan menginginkan tampilan untuk halaman admin agar mudah digunakan atau *user friendly* dengan adanya deskripsi, foto, dan lokasi dari permasalahan yang dilaporkan. Untuk hasil wawancara dalam proses penggalian kebutuhan sistem ini dapat dilihat pada Lampiran A Hasil Wawancara.

4.1.2 Standar Operasional Prosedur (SOP) Pengaduan Masyarakat pada Dinas Perhubungan Kota Malang

Pada Dinas Perhubungan Kota Malang memiliki standar operasional prosedur untuk pengaduan masyarakat. Pengaduan masyarakat disini bersifat pengaduan tentang pelayanan yang menjadi wewenang Dinas Perhubungan Kota Malang. Alur untuk melakukan pengaduan pada Dinas Perhubungan Kota Malang itu sendiri dengan melakukan proses pengaduan melalui website dengan mengisikan pada form secara jelas dan benar, dapat melalui sosial media seperti twitter dan Instagram. Kemudian laporan yang masuk akan diverifikasi oleh admin dan selanjutnya admin akan meneruskan atau memberitahukan laporan yang masuk pada pejabat berwenang kemudian jika telah diverifikasi oleh pejabat tersebut yang sudah di koordinasi selanjunya laporan akan diteruskan pada petugas terkait yang menangani pelaporan sesuai dengan bidangnya untuk dilakukan penanganan terhadap laporan yang disampaikan oleh pelapor atau masyarakat.

4.1.2.1 Alur dan Standar Pelayanan Pengaduan Dinas Perhubungan Kota Malang

Berikut merupakan alur dan standar pelayanan pengaduan Dinas Perhubungan Kota Malang dapat ditunjukkan pada Gambar 4.1.

1. Hak Penerima Pelayanan
    - Menyampaikan berupa keluhan, kritik, pernyataan tidak puas
    - Memperoleh tanggapan
    - Mengetahui hasil pengaduan

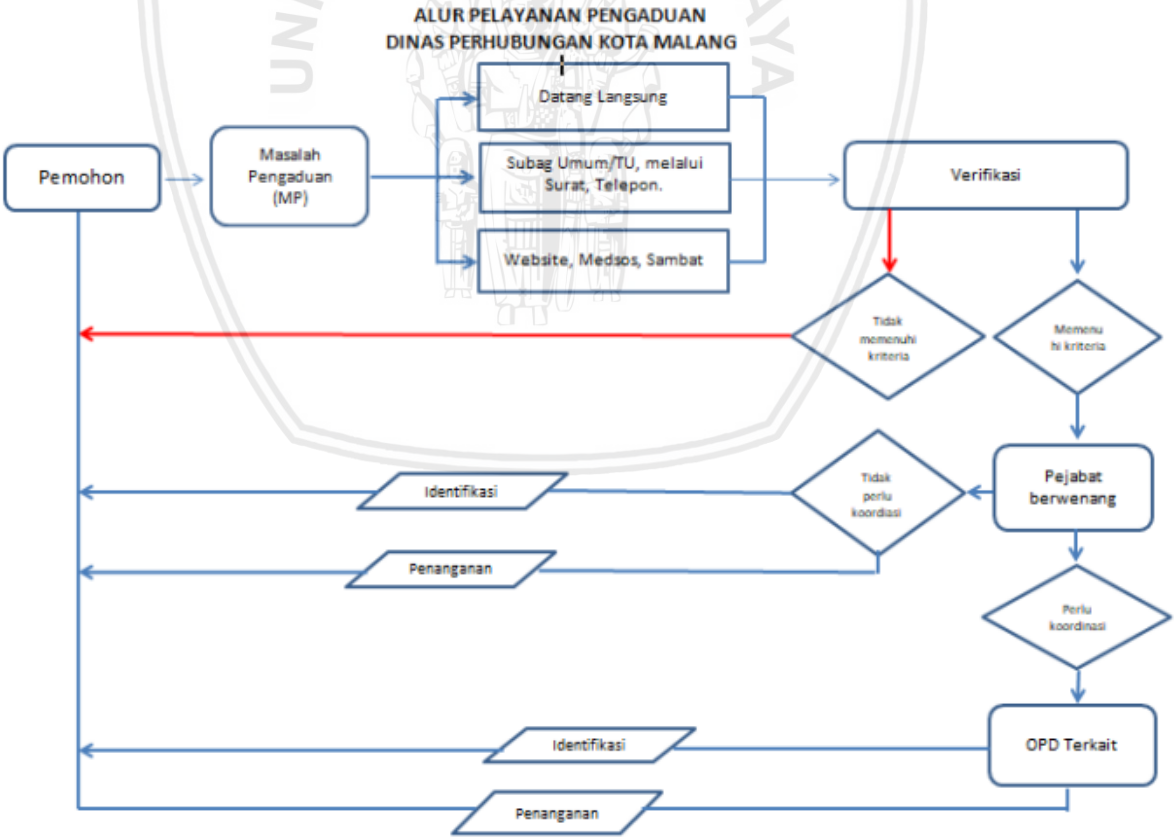
2. Cara Penyampaian Pengaduan
    - Datang langsung
    - Melalui surat
    - Melalui media cetak/eletronik/telepon Dinas

- Website Dinas Perhubungan Kota Malang

3. Tugas Pelayanan
    - Menerima pengaduan
    - Registrasi
    - Menindaklanjuti menyampaikan ke bidang terkait
    - Menyampaikan hasil penyelesaian ke pemohon

4. Waktu Penyelesaian Tanpa Perlu Koordinasi
    - Paling lama 3 hari kerja sejak pengaduan diterima harus identifikasi atau klarifikasi
    - Paling lama 7 hari kerja sejak pengaduan diterima harus menyampaikan hasil

5. Waktu Penyelesaian Perlu Koordinasi
    - Paling lama 7 hari kerja sejak pengaduan diterima harus identifikasi atau klarifikasi
    - Paling lama 30 hari kerja sejak pengaduan diterima harus menyampaikan hasil



Gambar 4. 1 Standar Operasional Prosedur Pengaduan Masyarakat pada Dinas Perhubungan Kota Malang

4.1.3 Gambaran Umum Sistem

Gambaran umum pada sistem pengaduan masyarakat pada Dinas Perhubungan Kota Malang dengan fitur *location based service* berbasis Android terdiri dari deskripsi sistem dan lingkungan sistem.

4.1.3.1 Deskripsi Sistem

Aplikasi yang akan dibangun berdasarkan permasalahan yang ada pada latar belakang masalah dimana masalah-masalah tersebut yang berkaitan dengan fasilitas umum yang ditangani oleh Dinas Perhubungan yang membuat masyarakat kebingungan harus menyampaikan keluhannya kepada siapa, lalu kurangnya wadah dan fasilitas untuk menyampaikan pengaduan oleh masyarakat kepada Dinas Perhubungan Kota Malang menjadi permasalahan lain dan kurangnya informasi lokasi kejadian masalah yang dilaporkan tidak lengkap dan tidak akurat merupakan permasalahan lain yang membuat keluhan tersebut tidak ditindaklanjuti oleh Dinas Perhubungan Kota Malang sehingga pentingnya informasi lokasi kejadian masalah untuk diketahui segera oleh Dinas Perhubungan Kota Malang guna mempercepat dalam penanganan keluhan dari masyarakat. Untuk itu, dibutuhkan suatu aplikasi yang digunakan sebagai media untuk mempermudah masyarakat dalam menyampaikan keluhan terkait dengan masalah fasilitas umum yang menjadi tugas Dinas Perhubungan Kota Malang. Aplikasi ini diberi nama "SIGAP" yang merupakan singkatan dari Aplikasi Pengaduan Masyarakat pada Dinas Perhubungan dan sesuai dengan kinerja Dinas Perhubungan Kota Malang yang selalu sigap dalam menjalankan tugasnya. Dengan adanya aplikasi ini, diharapkan masyarakat ikut berperan aktif dalam meningkatkan pelayanan *public* yang baik dengan cara menyampaikan pengaduan berupa keluhan ataupun kritik yang membangun tentang pelayanan yang diterima oleh masyarakat agar Dinas Perhubungan pun dapat dengan mudah merespon dan menindaklanjuti permasalahan yang terjadi dengan cepat.

Fitur utama yang ada aplikasi ini ialah proses melaporkan keluhan yang berupa deskripsi (*text*) dan dapat menyertakan dengan foto sesuai dengan kategori permasalahan yang akan dilaporkan. Informasi-informasi tersebut digunakan untuk mempermudah permasalahan yang terjadi, foto keluhan yang diambil pengguna, dan penentuan lokasi pelaporan. Penentuan lokasi keluhan menggunakan fitur *location based service* dengan memanfaatkan fitur *Google Maps* dan *GPS* yang dapat menentukan lokasi pelapor dengan tepat berdasarkan informasi yang didapat dari fitur tersebut. Untuk pihak Dinas Perhubungan yang ingin mengetahui lokasi akurat dari lokasi pelaporan dengan menggunakan fitur *location based service*, fitur ini digunakan untuk memudahkan dalam pencarian lokasi pelapor dengan mengetahuinya melalui keberadaan posisi aplikasi *mobile* pelapor. Selain itu, aplikasi ini dapat melihat informasi mengenai *history* daftar *support* atau dukungan yang pernah dilakukan untuk mendukung laporan pengguna lain.

Masyarakat atau *user* dapat memberikan dukungan kepada laporan *user* lain dengan cara memberikan *support* pada foto maupun deskripsi laporan. Dimana

jika laporan tersebut memiliki banyak *support* menunjukkan laporan tersebut yang menjadi permasalahan yang banyak dan sering dialami oleh masyarakat Kota Malang sehingga butuh ditindaklanjuti dengan segera.

4.1.3.2 Lingkungan Sistem

Aplikasi pengaduan masyarakat pada Dinas Perhubunan Kota Malang menggunakan fitur *location based service* berbasis Android membutuhkan lingkungan sistem untuk berjalannya aplikasi ini. Aplikasi ini menggunakan perangkat berupa *smartphone* khususnya dengan menggunakan sistem operasi Android. *Smartphone* dipilih karena dapat digunakan dengan mudah dimana saja dan kapan saja, dan juga *smartphone* banyak dimiliki oleh masyarakat.

4.2 Identifikasi Aktor

Identifikasi aktor dilakukan karena memiliki tujuan untuk mengetahui aktor siapa saja yang berperan dalam sistem. Identifikasi aktor merupakan cara untuk menentukan orang atau aktor maupun sistem yang berinteraksi dengan sistem lain sesuai dengan kebutuhan masing-masing tiap aktor tersebut. Aktor yang terlibat pada aplikasi pengaduan masyarakat ini pengguna atau masyarakat dan Admin atau Dinas Perhubungan Kota Malang. Identifikasi aktor dan deskripsinya dapat dilihat pada Tabel 4.1.

Tabel 4. 1 Identifikasi Aktor

No	Aktor	Deskripsi
1.	Pengguna	Pengguna merupakan aktor yang memiliki hak akses untuk *register* dan *login* yang digunakan sebagai autentifikasi.
2.	Pelapor	Pelapor merupakan aktor yang sudah melakukan registrasi dan *login* ke sistem sehingga dapat melakukan proses pengaduan masyarakat pada aplikasi
3.	Admin	Admin merupakan aktor yang memiliki hak akses penuh terhadap sistem dan dapat mengelola data keluhan dari pelapor.

4.3 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional merupakan tahapan untuk menganalisis kebutuhan-kebutuhan apa saja yang harus dimiliki dan dipenuhi pada sistem. Analisis kebutuhan fungsional ini menjelaskan mengenai aktor, kode fungsi, dan deskripsi kebutuhannya. Kebutuhan fungsional ini berkaitan dengan fitur yang dimiliki oleh sistem. Berikut merupakan daftar kebutuhan fungsional yang harus dimiliki oleh sistem dapat dilihat pada Tabel 4.2 sampai Tabel 4.4.

Tabel 4. 2 Daftar Kebutuhan Fungsional Pengguna

No	Kode	Nama *Use Case*	Deskripsi
1.	SG-1-001	*Register* Pelapor	Pengguna melakukan registrasi untuk mengakses aplikasi.
2.	SG-1-002	*Login* Pelapor	Pengguna melakukan *login* guna masuk kedalam aplikasi.

Tabel 4. 3 Daftar Kebutuhan Fungsional Pelapor

No.	Kode	Nama *Use Case*	Deskripsi
1.	SG-2-001	Melihat Daftar Laporan	Sistem dapat menampilkan seluruh laporan keluhan yang ada pada daftar laporan.
2.	SG-2-002	Menambah Laporan	Sistem dapat melakukan penambahan laporan baru.
3.	SG-2-003	Memilih Fitur *Camera*	Sistem dapat menampilkan fitur *camera* untuk mengambil foto.
4.	SG-2-004	Memilih Fitur *Gallery*	Sistem dapat menampilkan fitur g*allery* untuk memilih foto dari g*allery*.
5.	SG-2-005	Mengubah Laporan	Sistem dapat melakukan perubahan pada laporan yang disampaikan.
6.	SG-2-006	Melihat *Detail* Laporan	Sistem dapat menampilkan *detail* dari laporan yang berisi *marker* tentang deskripsi, foto, kategori keluhan, dan lokasi pelaporan.
7.	SG-2-007	*Support* Keluhan	Sistem dapat melakukan *support* atau mendukung keluhan yang dilaporkan oleh pelapor lain.
8.	SG-2-008	Melihat *Profile*	Sistem dapat menampilkan *detail profile* dari pelapor yakni nama, foto, data diri, dan riwayat pelaporan keluhan.
9.	SG-2-009	Mengubah *Profile*	Sistem dapat melakukan perubahan pada *profile* pengguna.

10.	SG-2-010	Melihat Laporan Pribadi	Sistem dapat menampilkan laporan pribadi milik pelapor itu sendiri yang berupa foto, kategori keluhan, lokasi, dan deskripsi laporan.
11.	SG-2-011	Melihat Daftar Dukungan	Sistem dapat menampilkan seluruh dukungan terhadap laporan yang didukung.
12.	SG-2-012	Melihat *Trending* Laporan	Sistem dapat menampilkan seluruh *trending* laporan yang berdasarkan s*upport* terbanyak.
13.	SG-2-013	*Logout*	Pelapor melakukan *logout* untuk keluar dari sistem.

Tabel 4. 4 Daftar Kebutuhan Fungsional Admin

No.	Kode	Nama *Use Case*	Deskripsi
1.	SG-3-001	*Login*	Admin melakukan *login* untuk masuk kedalam sistem.
2.	SG-3-002	Melihat Jumlah Laporan	Sistem dapat menampilkan jumlah laporan yang masuk pada sistem.
3.	SG-3-003	Melihat *Trending* Laporan	Sistem dapat menampilkan seluruh *trending* laporan yang berisi deskripsi, foto, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh.
4.	SG-3-004	Melihat *Detail Trending*	Sistem dapat menampilkan detail seluruh *trending* laporan yang berisi deskripsi laporan, *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab.
5.	SG-3-005	Membagikan Lokasi Kejadian *Trending* Laporan	Sistem dapat menampilkan alamat lokasi kejadian yang menjadi *trendin*g laporan untuk dibagikan kepada penanggungjawab.

6.	SG-3-006	Mencari *Trending* Laporan	Sistem dapat melakukan pencarian terhadap laporan yang menjadi *trending*.
7.	SG-3-007	Melihat Laporan Berdasarkan Status *Waiting*	Sistem dapat menampilkan seluruh laporan berdasarkan status *waiting* yang berisi deskripsi, foto, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh.
8.	SG-3-008	Melihat *Detail* Laporan Status *Waiting*	Sistem dapat menampilkan *detail* seluruh laporan berdasarkan status *waiting* yang berisi deskripsi laporan, map dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab.
9.	SG-3-009	Mengganti Laporan Status *Waiting*	Sistem dapat melakukan perubahan pada status laporan yang berstatus *waiting.*
10.	SG-3-010	Membagikan Lokasi Kejadian Status *Waiting*	Sistem dapat menampilkan alamat lokasi kejadian yang berstatus *waiting* untuk dibagikan kepada penanggungjawab.
11.	SG-3-011	Mencari Laporan Status *Waiting*	Sistem dapat melakukan pencarian terhadap laporan yang berstatus *waiting*.
12.	SG-3-012	Melihat Laporan Berdasarkan Status *OnProgress*	Sistem dapat menampilkan seluruh laporan berdasarkan status *onprogress* yang berisi deskripsi, foto, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh.
13.	SG-3-013	Melihat *Detail* Laporan Status *OnProgress*	Sistem dapat menampilkan detail seluruh laporan berdasarkan status *onprogress*.
14.	SG-3-014	Mengganti Laporan Status *OnProgress*	Sistem dapat melakukan perubahan pada status laporan yang berstatus *onprogress*.

15.	SG-3-015	Membagikan Lokasi Kejadian Status *OnProgress*	Sistem dapat menampilkan alamat lokasi kejadian yang berstatus *onprogress* untuk dibagikan kepada penanggungjawab.
16.	SG-3-016	Mencari Laporan Status *OnProgress*	Sistem dapat melakukan pencarian terhadap laporan yang berstatus *onprogress.*
17.	SG-3-017	Melihat Laporan Berdasarkan Status *Done*	Sistem dapat menampilkan seluruh laporan berdasarkan status *done* yang berisi deskripsi, foto, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh.
18.	SG-3-018	Melihat *Detail* Laporan Status *Done*	Sistem dapat menampilkan *detail* seluruh laporan berdasarkan status *done*.
19.	SG-3-019	Mengganti Laporan Status *Done*	Sistem dapat melakukan perubahan pada status laporan yang berstatus *done*.
20.	SG-3-020	Mencari Laporan Status *Done*	Sistem dapat melakukan pencarian terhadap laporan yang berstatus *done*.
21.	SG-3-021	Melihat Laporan Berdasarkan Status *Block*	Sistem dapat menampilkan seluruh laporan berdasarkan status *block* yang berisi deskripsi, foto, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh.
22.	SG-3-022	Melihat *Detail* Laporan Status *Block*	Sistem dapat menampilkan *detail* seluruh laporan berdasarkan status *block*.
23.	SG-3-023	Mengganti Laporan Status *Block*	Sistem dapat melakukan perubahan pada status laporan yang berstatus *block*.
24.	SG-3-024	Mencari Laporan Status *Block*	Sistem dapat melakukan pencarian terhadap laporan yang berstatus *block*.
25.	SG-3-025	*Logout*	Admin melakukan *logout* untuk keluar dari sistem.

4.4 Analisis Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakaner kebutuhan yang ada pada sistem yang tidak wajib ada namun kebutuhan tersebut sangat diperlukan. Analisis kebutuhan non-fungsional ini ialah melakukan proses analisis untuk kebutuhan non-fungsional yang ada pada sistem pengaduan masyarakat pada Dinas Perhubungan Kota Malang, proses ini menjelaskan mengenai kode dan deskripsi dari fitur kebutuhan tersebut. Dapat dilihat pada Tabel 4.5 tentang daftar kebutuhan non-fungsional.

Tabel 4. 5 Daftar Kebutuhan Non-Fungsional

No	Kode	Nama	Deskripsi
1.	SG-4-001	*Usability*	Sistem mudah digunakan oleh penggunanya dengan target hasil *score* SUS > 71
2.	SG-4-002	*Security*	Sistem tidak dapat menampilkan *password* pada respon akhir atau *end point*

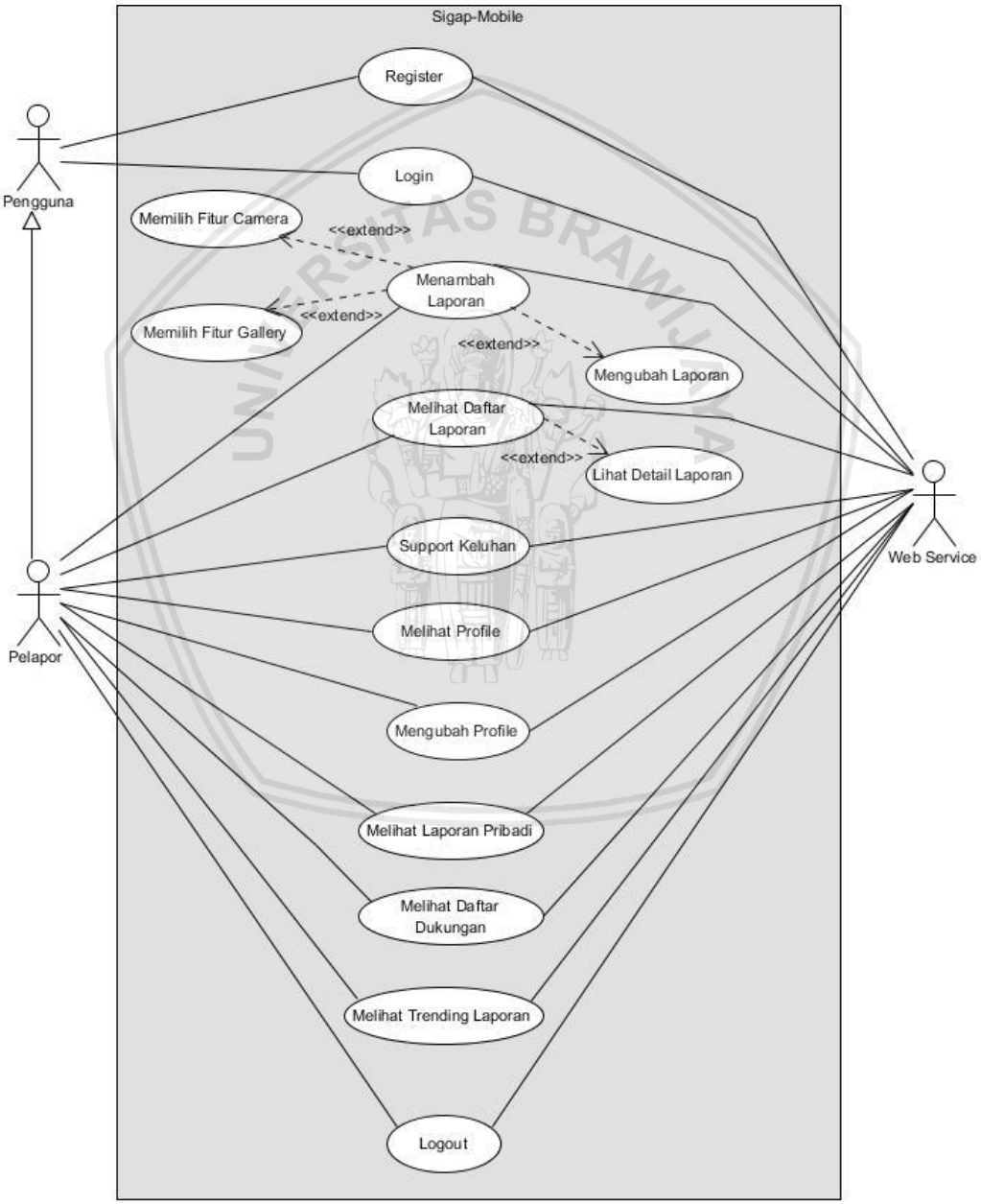
4.5 Pemodelan Kebutuhan

4.5.1 *Use Case Diagram*

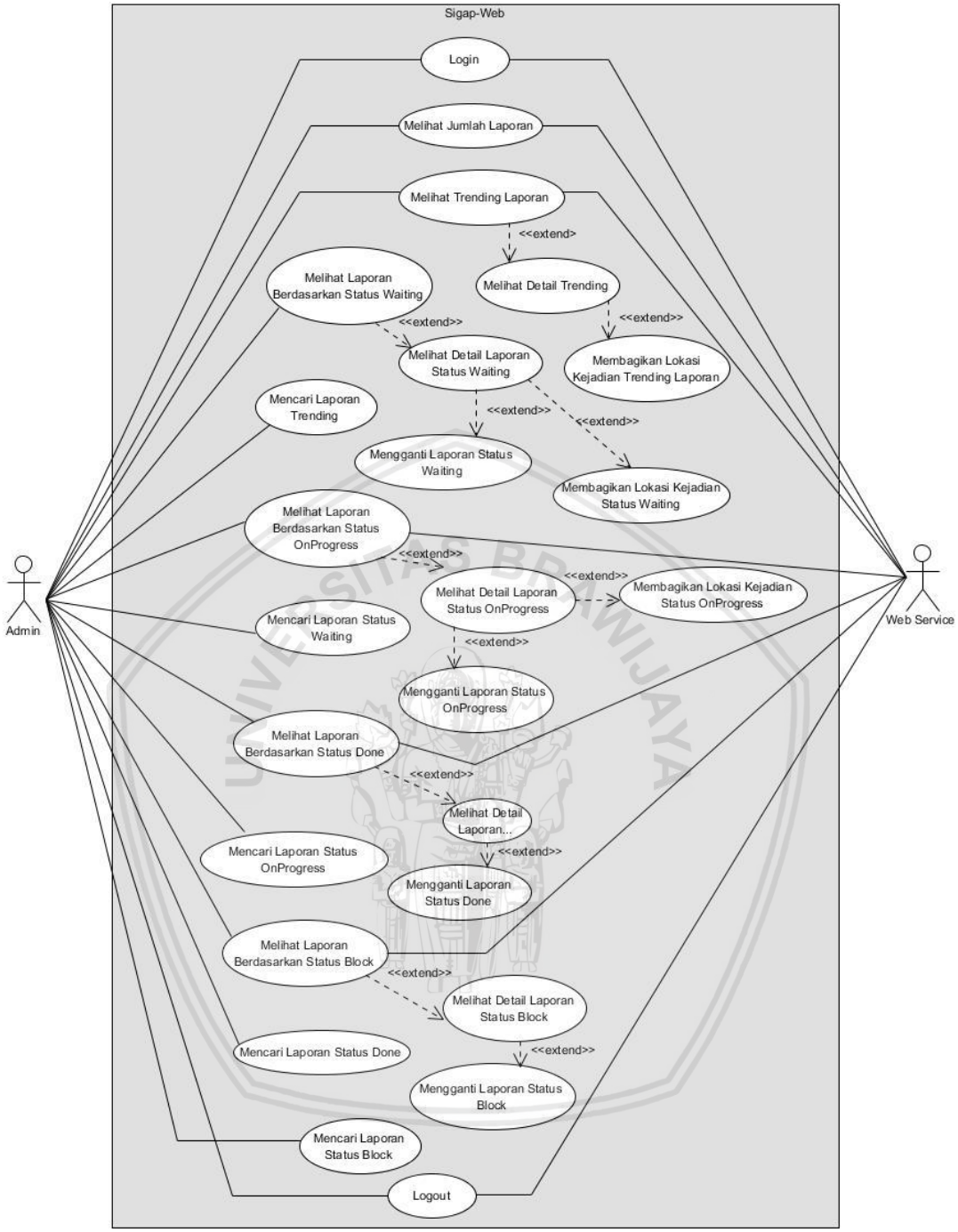
Use Case Diagram merupakan diagram yang menggambarkan fungsionalitas dari sebuah sistem yang dapat menjelaskan mengenai deskripsi lengkap tentang interaksi yang terjadi pada aktor dengan sistem atau perangkat lunak (Yasin, 2012). Aktor yang digambarkan merupakan pengguna dari sistem sedangkan *use case* ialah deskripsi tentang bagaimana sistem atau perangkat lunak berperilaku terhadap aktor.

Use case diagram dapat menjelaskan fungsi yang dapat dilakukan oleh aktor dan harus dipenuhi oleh sistem agar pembaca dapat mengetahui dan memahami fungsionalitas dari sistem. Pada aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang ini dispesialisasikan menjadi tiga aktor yaitu aktor *user* yang dibagi menjadi tiga jenis yakni yang pertama *user* sebagai pengguna, keda sebagai pelapor, dan yang kedua ialah *user* sebagai pelapor dan aktor lainnya yakni admin. *Use case diagra*m pada penelitian ini terdapat dua *use case diagram*, *use case diagram* yang pertama yakni *use case diagram* untuk *mobile app* yang digunakan oleh aktor pengguna, pelapor atau masyarakat yang digunakan untuk melaporkan pengaduannya melalui aplikasi *mobile* dan yang kedua ialah *use case diagram* untuk *web* yang digunakan oleh admin atau pihak Dinas Perhubungan Kota Malang untuk menerima laporan yang masuk dari pelapor. Untuk mengubungan dua sistem antara *mobile app* dengan *web* yakni dengan menggunakan *web service* yang digunakan sebagai penghubung antara dua sistem tersebut maka untuk itu *web service* disini dijadikan aktor karena sebagai penghubung antar dua sistem yang berbeda yakni *mobile app* dengan *web*. *Use*

case diagram merupakan kebutuhan fungsional yang dapat dilakukan oleh pengguna yang dilihat dari sisi aktor yaitu pengguna dan pelapor serta aktor admin atau Dinas Perhubungan Kota Malang. Pada masing-masing *use case diagram* terdapat *web service* karena digunakan sebagai penghubung antar dua sistem yang berbeda. Dapat dilihat pada gambar dibawah ini merupakan *use case diagram* untuk kebutuhan sistem dari aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang. Pada Gambar 4.2 merupakan Gambar *use case diagram* untuk *mobile app* yakni untuk aktor pengguna dan pelapor dan Gambar 4.3 merupakan Gambar *use case diagram* untuk *web* untuk *administrator* atau untuk pihak Dinas Perhubungan Kota Malang.



Gambar 4. 2 ***Use Case Diagram Mobile App***



Gambar 4. 3 *Use Case Diagram Web*

4.5.2 *Use Case Scenario*

Use Case Scenario dapat digunakan untuk menjelaskan secara *detail* kebutuhan fungsional pada sistem dari *use case diagram* karena dari setiap *use case* dalam diagram tersebut dapat dijelaskan lebih lanjut mengenai perilakunya yang dimasukkan pada *use case scenario* ini. Pada *use case scenario* berisi nama dan kode *use case*, aktor yang berperan dalam *use case*, tujuan dari *use case*, kondisi awal yang harus dipenuhi, alur utama *use case*, alur alternatif, dan kondisi

akhir yang diharapkan. Berikut merupakan *use case scenario* dari masing-masing *use case* dapat dilihat pada Tabel 4.6 sampai Tabel 4.45.

Tabel 4. 6 *Use Case Scenario Register* Pelapor

Use Case untuk *Register* Pelapor	
Kode	SG-1-001
Objective	*Use case* ini menjelaskan bagaimana pengguna melakukan *register*
Actor	Pengguna
Pre-condition	Pengguna masuk ke halaman "*Register"*
Main Flow	*1.* Pengguna menginputkan form untuk *register* yang terdiri dari *username, email*, dan *password* lalu menekan *button register* 2. Sistem menampilkan berupa pesan jika proses *register* berhasil yakni "*Register Successful*" lalu sistem masuk ke halaman untuk *login* dengan memasukkan *email* dan *username* yang sudah didaftarkan pada saat register
Alternatif Flow	1. Jika *username* yang diinputkan sudah ada maka akan muncul pesan bahwa *username* sudah ada dan digunakan yakni "*User Already Exist*!" 2. Jika penulisan *email* salah maka akan muncul pesan jika format *email* yang dimasukkan salah 3. Jika *username, email, password* kosong maka akan muncul pesan bahwa *username, email, dan password* tidak boleh kosong
Post-condition	Pengguna mempunyai akun untuk melakukan *login* sebagai pengguna pada aplikasi *mobile*

Tabel 4. 7 *Use Case Scenario Login* Pelapor

Use Case untuk *Login* Pelapor	
Kode	SG-1-002
Objective	*Use case* ini menjelaskan bagaimana pengguna melakukan *login*
Actor	Pengguna
Pre-condition	Pengguna masuk ke halaman "*Login"*

Main Flow	1. Pengguna memasukkan *email* dan *password* pada halaman *login* untuk melakukan *login* selanjutnya pengguna menekan *botton login* 2. Sistem menampilkan berupa pesan jika proses login berhasil yakni "*Login Successful*" lalu sistem masuk ke halaman daftar laporan atau menu *home*
Alternatif Flow	Jika *email* dan *password* yang dimasukkan tidak ada atau belum terdaftar maka akan muncul pesan jika *email* dan *password* tidak sesuai yakni "*You Have Entered An Invalid Email or Password!"*
Post-condition	Pengguna berhasil melakukan proses *login* pada aplikasi *mobile*

Tabel 4. 8 *Use Case Scenario* Melihat Daftar Laporan

Use Case untuk Melihat Daftar Laporan	
Kode	SG-2-001
Objective	*Use case* ini menjelaskan bagaimana pelapor melihat daftar laporan dari pelapor lain
Actor	Pelapor
Pre-condition	Pelapor sudah melakukan *login* ke sistem
Main Flow	1. Pelapor menekan tombol "Daftar Laporan" 2. Sistem dapat menampilkan halaman daftar laporan yang terdiri dari seluruh daftar laporan yang dilaporkan oleh pengguna lain
Alternatif Flow	Apabila tidak ada laporan baru maka sistem akan menampilkan pesan bahwa "Tidak Ada Laporan Untuk Saat Ini"
Post-condition	Pelapor melihat seluruh daftar laporan dari pengguna lain

Tabel 4. 9 *Use Case Scenario* Menambah Laporan

Use Case untuk Menambah Laporan	
Kode	SG-2-002
Objective	*Use case* ini menjelaskan bagaimana pelapor menambah laporan baru
Actor	Pelapor
Pre-condition	1. Pelapor menuju halaman "Upload Laporan"

	2. Pelapor sudah mengambil foto kejadian yang digunakan untuk foto laporan yang ingin dilaporkan
Main Flow	1. Pelapor memilih kategori keluhan berdasarkan permasalahan yang ingin dilaporkan 2. Pelapor mememasukkan deskripsi dari laporan yang ingin dilaporkan 3. Pelapor menekan tombol "*Pick Location*" untuk memasukkan lokasi kejadian 4. Sistem dapat menampilkan *map* 5. Pelapor memilih lokasi kejadian yang ingin dilaporkan dengan menggerakkan *marker* atau memilih tombol lokasi saat ini 6. Pelapor menekan tombol ceklis atau selesai jika lokasi sudah sesuai dengan lokasi yang ingin dilaporkan 7. Sistem akan menampilkan alamat lokasi kejadian sesuai dengan lokasi yang dipilih sebelumnya 8. Pelapor menekan *botton upload* untuk menggunggah laporan yang berupa foto, kategori keluhan, deskripsi, dan lokasi yang ingin di unggah 9. Sistem akan menampilkan pesan jika upload laporan berhasil yakni "Data Berhasil Disimpan, Harap Tunggu Proses Selanjutnya" 10. Sistem akan menyimpan data laporan yang di *upload* ke dalam basis data sistem 11. Sistem akan menampilkan data laporan yang sudah di *upload* pada halaman daftar laporan
Alternatif Flow	Apabila foto, kategori keluhan, deskripsi, atau lokasi tidak di isi atau kosong dan pelapor menekan tombol *upload* maka akan muncul pesan jika kategori keluhan, deskripsi, dan lokasi tidak boleh kosong yakni "Data Tidak Berhasil Disimpan, Harap Coba Kembali"
Post-condition	Pelapor berhasil membuat laporan baru dan laporan akan muncul pada halaman daftar laporan

Tabel 4. 10 *Use Case Scenario* Memilih Fitur *Camera*

Use Case untuk Melihat Fitur *Camera*	
Kode	SG-2-003

Objective	*Use case* ini menjelaskan bagaimana pelapor dapat memilih fitur *camera* untuk mengambil foto dari *camera smartphone* pelapor
Actor	Pelapor
Pre-condition	Pelapor sudah melakukan *login* ke sistem
Main Flow	1. Pelapor menekan tombol "*Report*" 2. Sistem dapat menampilkan halaman "*Report*" yang digunakan untuk membuat laporan baru 3. Pelapor memilih fitur "*Camera"* 4. Sistem akan membuka c*amera* dari *smartphone* pelapor 5. Pelapor mengambil foto kejadian yang diambil dari *camera smartphone* pengguna untuk membuat laporan baru 6. Sistem akan menampilkan foto kejadian yang sudah diambil pada halaman menu "*Camera*" 7. Pelapor menekan tombol *next* atau selanjutnya jika foto kejadian sudah benar untuk dilaporkan
Alternatif Flow	1. Jika pelapor belum mengambil foto kejadian dari *camera* dan menekan *botton next* atau selanjutnya maka akan muncul pesan jika "Kamu Belum Memilih Gambar" 2. Jika pelapor sudah mengambil foto kejadian yang diambil dari menu "*Gallery"* maka akan muncul pesan jika "Kamu Telah Memilih Foto Dari *Gallery*" 3. Jika pelapor menekan tombol *back* atau kembali maka proses mengambil foto kejadian dari *camera smartphone* pengguna tidak akan dilakukan
Post-condition	Pelapor berhasil mengambil foto kejadian yang diambil dengan memilih fitur *camera*

Tabel 4. 11 *Use Case Scenario* Memilih Fitur *Gallery*

Use Case untuk Melihat Fitur *Gallery*	
Kode	SG-2-004
Objective	*Use case* ini menjelaskan bagaimana pelapor dapat memilih fitur *gallery* untuk mengambil gambar yang sudah disimpan pada *gallery smartphone*
Actor	Pelapor
Pre-condition	Pelapor sudah melakukan *login* ke sistem

Main Flow	1. Pelapor menekan tombol "*Report*" 2. Sistem dapat menampilkan halaman "*Report"* yang digunakan untuk membuat laporan baru 3. Pelapor memilih fitur *"Gallery"* 4. Sistem akan membuka *gallery smartphone* pelapor 5. Pelapor memilih foto kejadian yang disimpan pada *gallery smartphone* pengguna untuk membuat laporan baru 6. Sistem akan menampilkan foto kejadian yang sudah dipilih pada halaman menu "*Gallery"* 7. Pelapor menekan tombol *next* atau selanjutnya jika foto kejadian sudah benar untuk dilaporkan
Alternatif Flow	1. Jika pelapor belum mengambil foto kejadian dari *gallery* dan menekan *botton next* atau selanjutnya maka akan muncul pesan jika "Kamu Belum Memilih Gambar" 2. Jika pelapor sudah mengambil foto kejadian yang diambil dari menu "*Camera*" maka akan muncul pesan jika "Kamu Telah Memilih Foto Dari *Camera*" 3. Jika pelapor menekan tombol *back* atau kembali maka proses memilih foto kejadian dari *gallery smartphone* pengguna tidak akan dilakukan
Post-condition	Pelapor berhasil memilih foto kejadian yang sudah disimpan pada *gallery smartphone* dengan memilih fitur *gallery*

Tabel 4. 12 *Use Case Scenario* Mengubah Laporan

Use Case untuk Mengubah Laporan	
Kode	SG-2-005
Objective	*Use case* ini menjelaskan pelapor dapat mengubah laporan yang sudah dilaporkan namun jika masih berstatus *waiting*
Actor	Pelapor
Pre-condition	1. Pelapor memilih laporan yang akan diubah 2. Pelapor menuju halaman "*Edit My Laporan"*
Main Flow	1. Pelapor mengubah dan mengisi laporan yang diinginkan sesuai dengan laporan permasahalan yakni foto, kategori keluhan, deskripsi, dan lokasi kejadian yang dapat dilakukan perubahan

	2. Pelapor menekan *botton save* untuk menggunggah laporan yang sudah diubah berupa foto, kategori keluhan, deskripsi, dan lokasi yang ingin di unggah 3. Sistem akan menampilkan pesan jika laporan berhasil diubah yakni "Data Berhasil Di *Update*" 4. Sistem akan menyimpan data laporan yang sudah di ubah ke dalam basis data sistem 5. Sistem akan memperbaharui data laporan yang sudah di ubah pada menu "Laporan" yang terdapat pada halaman "*Profile"*
Alternatif Flow	1. Jika foto, kategori keluhan, deskripsi, atau lokasi tidak di isi atau kosong dan pelapor menekan tombol *save* maka akan muncul pesan jika kategori keluhan, deskripsi, dan lokasi tidak boleh kosong yakni "Data Gagal Di *Update*" 2. Laporan yang dapat dilakukan perubahan hanya laporan yang masih berstatus *waiting* sedangkan laporan yang sudah berstatus *onprogress* atau *done* tidak dapat dilakukan perubahan laporan 3. Jika pelapor menekan tombol *back* atau kembali maka proses mengubah laporan pribadi tidak akan dilakukan
Post-condition	Pelapor berhasil mengubah laporan pribadi dan laporan yang sudah diubah akan diperbarui pada menu "Laporan" yang terdapat pada halaman "*Profile*"

Tabel 4. 13 *Use Case Scenario* Melihat *Detail* Laporan

Use Case untuk Melihat *Detail* Laporan	
Kode	SG-2-006
Objective	*Use case* ini menjelaskan bagaimana pelapor melihat *detail* dari pelaporan keluhan
Actor	Pelapor
Pre-condition	Pelapor sudah melakukan *login* ke sistem
Main Flow	1. Pelapor menekan tombol "Daftar Laporan" 2. Sistem dapat menampilkan halaman daftar laporan yang terdiri dari seluruh daftar laporan yang dilaporkan oleh pengguna lain 3. Pelapor memilih laporan yang diinginkan untuk melihat secara *detail* dari laporan tersebut

	4. Sistem akan menampilkan halaman "*Detail* Laporan" yang berisi foto kejadian, kategori keluhan, deskripsi, tanggal posting, status laporan, jumlah *support* keluhan, dan juga *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian
Alternatif Flow	Jika pada menu "Daftar Laporan" tidak terdapat laporan maka akan muncul pesan "Tidak Ada Laporan Untuk Saat Ini"
Post-condition	Pelapor dapat melihat *detail* laporan yang diinginkan dari laporan pengguna lain

Tabel 4. 14 *Use Case Scenario Support* Keluhan

Use Case untuk *Support* Keluhan	
Kode	SG-2-007
Objective	*Use sase* ini menjelaskan pengguna dapat s*upport* atau mendukung laporan keluhan pelapor lain
Actor	Pelapor
Pre-condition	Pelapor sudah melakukan *login* ke sistem
Main Flow	1. Pelapor menekan tombol "Daftar Laporan" 2. Sistem dapat menampilkan halaman daftar laporan yang terdiri dari seluruh daftar laporan yang dilaporkan oleh pengguna lain 3. Pelapor memilih laporan yang diinginkan untuk melihat secara *detail* dari laporan tersebut 4. Sistem akan menampilkan halaman "*Detail* Laporan" yang berisi foto kejadian, kategori keluhan, deskripsi, tanggal posting, status laporan, jumlah *support* keluhan, dan juga *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian 5. Pelapor menekan tombol *love* yang digunakan untuk *support* keluhan 6. Sistem dapat menampilkan jumlah dukungan laporan dan sistem mengubah tombol *love* menjadi berwarna abu-abu
Alternatif Flow	Jika pelapor menekan tombol *love* yang sudah didukung atau menekan tombol *love* yang berwarna abu-abu maka akan muncul pesan jika "Dukung Laporan DiBatalkan"
Post-condition	*Support* keluhan pada laporan berhasil ditambahkan

Tabel 4. 15 *Use Case Scenario* Melihat *Profile*

Use Case untuk Melihat *Profile*	
Kode	SG-2-008
Objective	*Use case* ini menjelaskan pelapor dapat melihat *profile* dari pelapor itu sendiri
Actor	Pelapor
Pre-condition	Pelapor sudah melakukan *login* ke sistem
Main Flow	1. Pelapor memilih menu "Biodata" pada halaman *Profile* 2. Sistem dapat menampilkan halaman "Biodata" pengguna yang terdiri dari nama, *email,* dan tanggal lahir pengguna
Alternatif Flow	Jika pelapor gagal melihat *profile* maka akan muncul pesan jika "Terjadi Kesalahan, Data Gagal Di Proses"
Post-condition	Pelapor berhasil melihat *profile* miliknya sendiri

Tabel 4. 16 *Use Case Scenario* Mengubah *Profile*

Use Case untuk Mengubah *Profile*	
Kode	SG-2-009
Objective	*Use case* ini menjelaskan pelapor dapat mengubah p*rofile* dari pelapor itu sendiri
Actor	Pelapor
Pre-condition	Pelapor sudah melakukan *login* ke sistem
Main Flow	*1.* Pelapor memilih menu "Biodata" pada halaman *Profile* 2. Sistem akan menampilkan halaman "Biodata" dari pengguna yang terdiri dari nama, *email*, dan tanggal lahir pengguna 3. Pelapor menekan tombol *edit* pada halaman "Biodata" yang berbentuk *icon edit* 4. Sistem dapat menampilkan halaman untuk merubah *profile* pelapor 5. Pelapor mengubah dan mengisi biodata pengguna yang akan diubah, yang dapat dilakukan perubahan pada biodata yakni nama, alamat, *password*, dan juga tanggal lahir pengguna dan selanjutnya pelapor menekan tombol ceklis atau selesai jika selesai melakukan perubahan

	6. Sistem akan menampilkan pesan jika biodata berhasil diubah yakni “Data Berhasil Di *Update*” 7. Sistem akan menyimpan data biodata yang sudah di ubah ke dalam basis data sistem 8. Sistem akan memperbarui data biodata yang sudah di ubah pada menu “Biodata” yang terdapat pada halaman “*Profile*”
Alternatif Flow	1. Jika nama, alamat, *password*, dan juga tanggal lahir pengguna tidak di isi atau kosong dan pelapor menekan tombol ceklis atau selesai maka akan muncul pesan jika nama, alamat, *password*, dan juga tanggal lahir pengguna tidak boleh kosong 2. Jika pelapor menekan tombol *back* atau kembali maka proses mengubah laporan pribadi tidak akan dilakukan
Post-condition	Pelapor berhasil mengubah biodata *profile* miliknya sendiri

Tabel 4. 17 *Use Case Scenario* Melihat Laporan Pribadi

Use Case untuk Melihat Laporan	
Kode	SG-2-010
Objective	*Use case* ini digunakan untuk melihat laporan pribadi yang sudah dilaporkan oleh pelapor itu sendiri
Actor	Pelapor
Pre-condition	Pelapor sudah melakukan *login* ke sistem
Main Flow	1. Pelapor memilih menu “Laporan” pada halaman *Profile* 2. Sistem dapat menampilkan halaman “Laporan” pengguna yang terdiri dari seluruh daftar laporan yang pernah dilaporkan baik yang berstatus *waiting, onprogress* maupun *done*
Alternatif Flow	Jika pelapor belum pernah membuat laporan pada sistem maka akan muncul pesan “Kamu Belum Melaporkan Sesuatu”
Post-condition	Pelapor berhasil melihat seluruh daftar laporan miliknya sendiri yang pernah dilaporkan

Tabel 4. 18 *Use Case Scenario* Melihat Daftar Dukungan

Use Case untuk Melihat Daftar Dukungan	
Kode	SG-2-011

Objective	*Use case* ini digunakan untuk melihat daftar dukungan pribadi yang pernah dilakukan terhadap dukungan laporan pengguna lain
Actor	Pelapor
Pre-condition	Pelapor sudah melakukan *login* ke sistem
Main Flow	1. Pelapor menekan tombol "Daftar Dukungan" 2. Sistem dapat menampilkan halaman "Daftar Dukungan" yang terdiri dari seluruh daftar dukungan yang pernah didukung terhadap laporan yang dilaporkan oleh pengguna lain
Alternatif Flow	Jika pelapor tidak mendukung laporan maka akan muncul pesan jika "Kamu Tidak Mendukung Laporan 3 Bulan Ini"
Post-condition	Pelapor dapat melihat seluruh daftar dukungan yang pernah di dukung terhadap laporan lain

Tabel 4. 19 *Use Case Scenario* Melihat *Trending* Laporan

Use Case untuk Melihat *Trending* Laporan	
Kode	SG-2-012
Objective	*Use case* ini digunakan untuk melihat *trending* dari laporan berdasarkan s*upport* terbanyak
Actor	Pelapor
Pre-condition	Pelapor sudah melakukan *login* ke sistem
Main Flow	1. Pelapor menekan tombol "*Trending* Laporan" 2. Sistem dapat menampilkan halaman "*Trending* Laporan" yang terdiri dari seluruh laporan yang menjadi *trending* dalam waktu 3 hari terakhir karena memiliki jumlah *support* terbanyak
Alternatif Flow	1. Jika laporan memiliki jumlah *support* paling banyak maka laporan tersebut berada paling atas diantara laporan lainnya yang terdapat pada halaman "*Trending* Laporan" 2. Jika tidak ada *trending* laporan maka sistem akan menampilkan pesan jika "Tidak Ada *Trending* Untuk 3 Hari Ini"
Post-condition	Pelapor dapat melihat *trending* laporan berdasarkan jumlah *support* terbanyak

Tabel 4. 20 *Use Case Scenario Logout*

Use Case untuk *Logout*	
Kode	SG-2-013
Objective	*Use case* ini digunakan untuk melakukan *logout*
Actor	Pelapor
Pre-condition	Pelapor sudah melakukan *login* ke sistem
Main Flow	*1.* Pelapor memilih menu "Biodata" pada halaman *Profile* 2. Sistem akan menampilkan halaman "Biodata" dari pengguna 3. Pelapor menekan tombol *close* atau *logout* pada halaman "Biodata" 4. Sistem berhasil keluar dari sistem 5. Sistem menampilkan halaman *login*
Alternatif Flow	-
Post-condition	Pelapor dapat melakukan proses *logout* dari sistem

Tabel 4. 21 *Use Case Scenario Login* Admin

Use Case untuk *Login* Admin	
Kode	SG-3-001
Objective	*Use case* ini menjelaskan admin dapat melakukan *login* pada halaman *login* admin
Actor	Admin
Pre-condition	Admin masuk ke halaman "*Login"*
Main Flow	1. Admin memasukkan *email* dan *password* pada halaman *login* untuk melakukan *login* selanjutnya 2. Admin menekan botton *login* 3. Sistem menampilkan halaman utama atau halaman *dashboard*
Alternatif Flow	Apabila *email* dan *password* yang dimasukkan *invalid* atau tidak terdapat pada sistem, maka admin tidak dapat melakukan login dan gagal masuk ke halaman *dashboard*
Post-condition	Admin berhasil melakukan proses *login* ke sistem

Tabel 4. 22 *Use Case Scenario* Melihat Jumlah Laporan

Use Case untuk Melihat Jumlah Laporan	
Kode	SG-3-002
Objective	*Use case* ini menjelaskan admin dapat melihat jumlah dari daftar laporan masyarakat yang masuk ke dalam sistem
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "*Dashboard*" 2. Sistem dapat menampilkan halaman jumlah laporan yang masuk terdiri dari jumlah *trending* laporan, jumlah laporan status *waiting*, status *onprogress*, status *done*, serta status *block*
Alternatif Flow	-
Post-condition	Admin dapat melihat jumlah laporan yang masuk ke sistem terdiri dari jumlah *trending* laporan, jumlah laporan status *waiting*, status *onprogress*, status *done*, serta status *block*

Tabel 4. 23 *Use Case Scenario* Melihat *Trending* Laporan

Use Case untuk Melihat *Trending* Laporan	
Kode	SG-3-003
Objective	*Use case* ini menjelaskan admin dapat melihat *trending* laporan yang masuk ke sistem
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	*1.* Admin menekan tombol *"Trending"* 2. Sistem dapat menampilkan halaman daftar laporan yang menjadi *trending* terdiri dari deskripsi laporan, foto laporan, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh laporan
Alternatif Flow	Apabila tidak ada *trending* laporan maka sistem akan menampilkan pesan "*No matching records found*"
Post-condition	Admin dapat melihat daftar *trending* laporan yang masuk pada sistem

Tabel 4. 24 *Use Case Scenario* Melihat *Detail Trending*

Use Case untuk Melihat *Detail Trending*	
Kode	SG-3-004
Objective	*Use case* ini menjelaskan bagaimana admin melihat *detail* dari *trending* laporan yang masuk ke sistem
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "*Trending*" *2.* Sistem dapat menampilkan halaman *trending* laporan yang terdiri dari seluruh daftar laporan yang menjadi *trending* 3. Admin memilih laporan yang diinginkan untuk melihat secara *detail* dari *trending* laporan tersebut 4. Sistem akan menampilkan halaman *"Detail Trending* Laporan" yang berisi deskripsi laporan, *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab
Alternatif Flow	Jika pada menu "*Trending*" tidak ada trending laporan maka sistem akan menampilkan pesan "*No matching records found*"
Post-condition	Admin dapat melihat *detail* laporan yang menjadi *trending*

Tabel 4. 25 *Use Case Scenario* Membagikan Lokasi Kejadian *Trending* Laporan

Use Case untuk Membagikan Lokasi Kejadian *Trending* Laporan	
Kode	SG-3-005
Objective	*Use case* ini menjelaskan admin membagikan lokasi kejadian dari *trending* laporan kepada penanggung jawab
Actor	Admin
Pre-condition	1. Admin sudah melakukan *login* ke sistem 2. Admin masuk ke halaman "*Trending*"
Main Flow	1. Admin menekan tombol "*Detail*" pada *trending* laporan yang dipilih 2. Sistem akan menampilkan halaman "*Detail Trending* Laporan" yang berisi deskripsi laporan, *map* yang

	ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab 3. Admin menekan tombol "*Share*" pada halaman "*Detail Trending* Laporan" 4. Sistem menampilkan pesan berupa *pop up* yang berisi deskripsi laporan dan lokasi kejadian yang sudah disalin 5. Admin menekan tombol "Ok" jika berhasil melakukan proses salinan terhadap lokasi kejadian
Alternatif Flow	-
Post-condition	Admin dapat membagikan lokasi kejadian yang menjadi *trending* laporan kepada petugas penanggung jawab

Tabel 4. 26 *Use Case Scenario* Mencari *Trending* Laporan

Use Case untuk Mencari *Trending* Laporan	
Kode	SG-3-006
Objective	*Use case* ini menjelaskan admin dapat mencari *trending* laporan yang ada pada halaman "*Trending*"
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "*Trending*" 2. Sistem dapat menampilkan halaman daftar laporan yang menjadi *trending* terdiri dari deskripsi laporan, foto laporan, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh laporan 3. Admin mencari laporan yang menjadi *trending* pada kolom pencarian 4. Sistem akan menampilkan laporan yang dicari oleh admin
Alternatif Flow	Jika laporan yang dicari oleh admin tidak terdapat pada sistem maka sistem menampilkan pesan "*No matching records found*"
Post-condition	Admin dapat mencari laporan yang menjadi *trending* laporan

Tabel 4. 27 *Use Case Scenario* Melihat Laporan Berdasarkan Status *Waiting*

Use Case untuk Melihat Laporan Berdasarkan Status *Waiting*	
Kode	SG-3-007

Objective	*Use case* ini menjelaskan admin dapat melihat laporan berdasarkan status *waiting* atau masih menunggu untuk ditindaklanjuti
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "*Waiting*" 2. Sistem dapat menampilkan halaman daftar laporan yang memiliki status *waiting* dimana laporan berisi deskripsi laporan, foto laporan, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh laporan
Alternatif Flow	Apabila tidak ada laporan yang berstatus *waiting* maka sistem akan menampilkan pesan "*No matching records found*"
Post-condition	Admin dapat melihat daftar laporan berdasarkan status *waiting*

Tabel 4. 28 *Use Case Scenario* Melihat *Detail* Laporan Status *Waiting*

Use Case untuk Melihat *Detail* Laporan Status *Waiting*	
Kode	SG-3-008
Objective	*Use case* ini menjelaskan bagaimana admin melihat *detail* dari laporan berdasarkan status *waiting*
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "*Waiting*" *2.* Sistem dapat menampilkan halaman "*Waiting*" laporan yang terdiri dari seluruh daftar laporan yang berstatus *waiting* *3.* Admin menekan tombol "Proses" pada laporan yang ingin dilihat secara *detail* 4. Sistem akan menampilkan halaman "*Detail* Laporan Berstatus *Waiting"* yang berisi deskripsi laporan, ganti status laporan, *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab
Alternatif Flow	Jika pada menu "*Waiting*" tidak ada laporan yang berstatus waiting maka sistem akan menampilkan pesan "*No matching records found"*

Post-condition	Admin dapat melihat *detail* laporan berdasarkan status *waiting*

Tabel 4. 29 *Use Case Scenario* Mengganti Laporan Status *Waiting*

Use Case untuk Mengganti Laporan Status *Waiting*	
Kode	SG-3-009
Objective	*Use case* ini menjelaskan admin dapat mengganti status laporan yang berstatus *waiting*
Actor	Admin
Pre-condition	Admin masuk ke halaman "*Waiting*"
Main Flow	1. Admin menekan tombol "Proses" pada laporan yang dipilih 2. Sistem akan menampilkan halaman "*Detail* Laporan Berstatus *Waiting*" yang berisi deskripsi laporan, ganti status laporan, *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab 3. Admin mengganti status laporan pada halaman "*Detail* Laporan Berstatus *Waiting*" 4. Sistem menampilkan status laporan yang dipilih 5. Admin menekan tombol "*Save*" jika status sudah terganti
Alternatif Flow	1. Jika laporan masih dalam status pengerjaan maka admin mengganti dengan status "*OnProgress*" 2. Jika laporan sudah selesai ditindaklanjuti maka admin mengganti dengan status "*Done*" 3. Jika laporan yang masuk ke sistem tidak sesuai dengan ketentuan pelaporan maka admin mengganti dengan status "*Block*"
Post-condition	Admin berhasil mengganti status laporan yang berstatus *waiting*

Tabel 4. 30 *Use Case Scenario* Membagikan Lokasi Kejadian Status *Waiting*

Use Case untuk Membagikan Lokasi Kejadian Status *Waiting*	
Kode	SG-3-010

Objective	*Use case* ini menjelaskan admin membagikan lokasi kejadian dari laporan yang berstatus *waiting* kepada penanggung jawab
Actor	Admin
Pre-condition	1. Admin sudah melakukan *login* ke sistem 2. Admin masuk ke halaman "*Waiting*"
Main Flow	1. Admin menekan tombol "Detail" pada laporan yang dipilih 2. Sistem akan menampilkan halaman "*Detail* Laporan Berstatus *Waiting*" yang berisi deskripsi laporan, *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab 3. Admin menekan tombol "*Share*" pada halaman "*Detail* Laporan Berstatus *Waiting*" 4. Sistem menampilkan pesan berupa *pop up* yang berisi deskripsi laporan dan lokasi kejadian yang sudah disalin 5. Admin menekan tombol "Ok" jika berhasil melakukan proses salinan terhadap lokasi kejadian
Alternatif Flow	-
Post-condition	Admin dapat membagikan lokasi kejadian berdasarkan laporan yang berstatus *waiting*

Tabel 4. 31 *Use Case Scenario* Mencari Laporan Status *Waiting*

Use Case untuk Mencari Laporan Status *Waiting*	
Kode	SG-3-011
Objective	*Use case* ini menjelaskan admin dapat mencari laporan berdasarkan status *waiting*
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "*Waiting*" 2. Sistem dapat menampilkan halaman daftar laporan yang berstatus *waiting* terdiri dari deskripsi laporan, foto laporan, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh laporan 3. Admin mencari laporan yang berstatus *waiting* pada kolom pencarian 4. Sistem akan menampilkan laporan yang dicari oleh admin

Alternatif Flow	Jika laporan yang dicari oleh admin tidak terdapat pada sistem maka sistem menampilkan pesan "*No matching records found*"
Post-condition	Admin dapat mencari laporan berdasarkan status *waiting*

Tabel 4. 32 *Use Case Scenario* Melihat Laporan Berdasarkan Status *OnProgress*

Use Case untuk Melihat Laporan Berdasarkan Status *OnProgress*	
Kode	SG-3-012
Objective	*Use case* ini menjelaskan admin dapat melihat laporan berdasarkan status *onprogress* atau sedang ditindaklanjuti
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "*Progress*" 2. Sistem dapat menampilkan halaman daftar laporan yang memiliki status *onprogress* dimana laporan berisi deskripsi laporan, foto laporan, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh laporan
Alternatif Flow	Apabila tidak ada laporan yang berstatus *onprogress* maka sistem akan menampilkan pesan "*No matching records found*"
Post-condition	Admin dapat melihat daftar laporan berdasarkan status *onprogress*

Tabel 4. 33 *Use Case Scenario* Melihat *Detail* Laporan Status *OnProgress*

Use Case untuk Melihat *Detail* Laporan Status *OnProgress*	
Kode	SG-3-013
Objective	*Use case* ini menjelaskan bagaimana admin melihat d*etail* dari laporan berdasarkan status *onprogress*
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "*Progress*" 2. Sistem dapat menampilkan halaman "*Progress*" laporan yang terdiri dari seluruh daftar laporan yang berstatus *onprogress* *3.* Admin menekan tombol "Proses" pada laporan yang ingin dilihat secara *detail*

	4. Sistem akan menampilkan halaman "*Detail* Laporan Berstatus *OnProgress*" yang berisi deskripsi laporan, ganti status laporan, *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab
Alternatif Flow	Jika pada menu "*Progress*" tidak ada laporan yang berstatus *onprogress* maka sistem akan menampilkan pesan "*No matching records found"*
Post-condition	Admin dapat melihat *detail* laporan berdasarkan status *onprogress*

Tabel 4. 34 *Use Case Scenario* Mengganti Laporan Status *OnProgress*

Use Case untuk Mengganti Laporan Status *OnProgress*	
Kode	SG-3-014
Objective	*Use case* ini menjelaskan admin dapat mengganti status laporan yang berstatus *onprogress*
Actor	Admin
Pre-condition	Admin masuk ke halaman "*Progress*"
Main Flow	1. Admin menekan tombol "Proses" pada laporan yang dipilih 2. Sistem akan menampilkan halaman "*Detail* Laporan Berstatus *OnProgress*" yang berisi deskripsi laporan, ganti status laporan, *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab 3. Admin mengganti status laporan pada halaman "*Detail* Laporan Berstatus *OnProgress*" 4. Sistem menampilkan status laporan yang dipilih 5. Admin menekan tombol "*Save*" jika status sudah terganti
Alternatif Flow	1. Jika laporan masih menunggu untuk pengerjaan maka admin mengganti dengan status "*Waiting*" 2. Jika laporan sudah selesai ditindaklanjuti maka admin mengganti dengan status "*Done*" 3. Jika laporan yang masuk ke sistem tidak sesuai dengan ketentuan pelaporan maka admin mengganti dengan status "*Block*"

Post-condition	Admin berhasil mengganti status laporan yang berstatus *onprogress*

Tabel 4. 35 *Use Case Scenario* Membagikan Lokasi Kejadian Status *OnProgress*

Use Case untuk Membagikan Lokasi Kejadian Status *OnProgress*	
Kode	SG-3-015
Objective	*Use case* ini menjelaskan admin membagikan lokasi kejadian dari laporan yang berstatus *onprogress* kepada penanggung jawab
Actor	Admin
Pre-condition	1. Admin sudah melakukan *login* ke sistem 2. Admin masuk ke halaman "*Progress*"
Main Flow	1. Admin menekan tombol "Proses" pada laporan yang dipilih 2. Sistem akan menampilkan halaman "*Detail* Laporan Berstatus *OnProgress*" yang berisi deskripsi laporan, *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab 3. Admin menekan tombol "*Share*" pada halaman "*Detail* Laporan Berstatus *OnProgress*" 4. Sistem menampilkan pesan berupa *pop up* yang berisi deskripsi laporan dan lokasi kejadian yang sudah disalin 5. Admin menekan tombol "*Ok*" jika berhasil melakukan proses salinan terhadap lokasi kejadian
Alternatif Flow	-
Post-condition	Admin dapat membagikan lokasi kejadian berdasarkan laporan yang berstatus *onprogress*

Tabel 4. 36 *Use Case Scenario* Mencari Laporan Status *OnProgress*

Use Case untuk Mencari Laporan Status *OnProgress*	
Kode	SG-3-016
Objective	*Use case* ini menjelaskan admin dapat mencari laporan berdasarkan status *onprogress*
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem

Main Flow	1. Admin menekan tombol "*Progress*" 2. Sistem dapat menampilkan halaman daftar laporan yang berstatus *onprogress* terdiri dari deskripsi laporan, foto laporan, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh laporan 3. Admin mencari laporan yang berstatus *onprogress* pada kolom pencarian 4. Sistem akan menampilkan laporan yang dicari oleh admin
Alternatif Flow	Jika laporan yang dicari oleh admin tidak terdapat pada sistem maka sistem menampilkan pesan "*No matching records found*"
Post-condition	Admin dapat mencari laporan berdasarkan status *onprogress*

Tabel 4. 37 *Use Case Scenario* Melihat Laporan Berdasarkan Status *Done*

Use Case untuk Melihat Laporan Berdasarkan Status *Done*	
Kode	SG-3-017
Objective	*Use case* ini menjelaskan admin dapat melihat laporan berdasarkan status *done* atau selesai ditindaklanjuti
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "*Done*" 2. Sistem dapat menampilkan halaman daftar laporan yang memiliki status *done* dimana laporan berisi deskripsi laporan, foto laporan, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh laporan
Alternatif Flow	Apabila tidak ada laporan yang berstatus *done* maka sistem akan menampilkan pesan "*No matching records found*"
Post-condition	Admin dapat melihat daftar laporan berdasarkan status *done*

Tabel 4. 38 *Use Case Scenario* Melihat *Detail* Laporan Status *Done*

Use Case untuk Melihat *Detail* Laporan Status *Done*	
Kode	SG-3-018
Objective	*Use case* ini menjelaskan bagaimana admin melihat d*etail* dari laporan berdasarkan status *done*
Actor	Admin

Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "*Done*" 2. Sistem dapat menampilkan halaman "*Done*" laporan yang terdiri dari seluruh daftar laporan yang berstatus *done* *3.* Admin menekan tombol "Proses" pada laporan yang ingin dilihat secara *detail* 4. Sistem akan menampilkan halaman "*Detail* Laporan Berstatus *Done"* yang berisi deskripsi laporan, ganti status laporan, *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab
Alternatif Flow	Jika pada menu "*Done*" tidak ada laporan yang berstatus done maka sistem akan menampilkan pesan "*No matching records found"*
Post-condition	Admin dapat melihat *detail* laporan berdasarkan status *done*

Tabel 4. 39 *Use Case Scenario* Mengganti Laporan Status *Done*

Use Case untuk Mengganti Laporan Status *Done*	
Kode	SG-3-019
Objective	*Use case* ini menjelaskan admin dapat mengganti status laporan yang berstatus *done*
Actor	Admin
Pre-condition	Admin masuk ke halaman "*Done*"
Main Flow	1. Admin menekan tombol "Proses" pada laporan yang dipilih 2. Sistem akan menampilkan halaman "*Detail* Laporan Berstatus *Done*" yang berisi deskripsi laporan, ganti status laporan, *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab 3. Admin mengganti status laporan pada halaman "*Detail* Laporan Berstatus *Done*" 4. Sistem menampilkan status laporan yang dipilih 5. Admin menekan tombol "*Save*" jika status sudah terganti
Alternatif Flow	1. Jika laporan masih menunggu untuk pengerjaan maka admin mengganti dengan status "*Waiting*"

	2. Jika laporan masih dalam proses pengerjaan maka admin mengganti dengan status "*OnProgress*" 3. Jika laporan yang masuk ke sistem tidak sesuai dengan ketentuan pelaporan maka admin mengganti dengan status *"Block"*
Post-condition	Admin berhasil mengganti status laporan yang berstatus *done*

Tabel 4. 40 *Use Case Scenario* Mencari Laporan Status *Done*

Use Case untuk Mencari Laporan Status *Done*	
Kode	SG-3-020
Objective	*Use case* ini menjelaskan admin dapat mencari laporan berdasarkan status *done*
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "*Done*" 2. Sistem dapat menampilkan halaman daftar laporan yang berstatus *done* terdiri dari deskripsi laporan, foto laporan, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh laporan 3. Admin mencari laporan yang berstatus *done* pada kolom pencarian 4. Sistem akan menampilkan laporan yang dicari oleh admin
Alternatif Flow	Jika laporan yang dicari oleh admin tidak terdapat pada sistem maka sistem menampilkan pesan "*No matching records found"*
Post-condition	Admin dapat mencari laporan berdasarkan status *done*

Tabel 4. 41 *Use Case Scenario* Melihat Laporan Berdasarkan Status *Block*

Use Case untuk Melihat Laporan Berdasarkan Status *Block*	
Kode	SG-3-021
Objective	*Use case* ini menjelaskan admin dapat melihat laporan berdasarkan status *block* atau laporan tidak sesuai dengan ketentuan
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem

Main Flow	1. Admin menekan tombol "*Block*" 2. Sistem dapat menampilkan halaman daftar laporan yang memiliki status *block* dimana laporan berisi deskripsi laporan, foto laporan, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh laporan
Alternatif Flow	Apabila tidak ada laporan yang berstatus block maka sistem akan menampilkan pesan "*No matching records found*"
Post-condition	Admin dapat melihat daftar laporan berdasarkan status *block*

Tabel 4. 42 *Use Case Scenario* Melihat *Detail* Laporan Status *Block*

Use Case untuk Melihat *Detail* Laporan Status *Block*	
Kode	SG-3-022
Objective	*Use case* ini menjelaskan bagaimana admin melihat d*etail* dari laporan berdasarkan status *block*
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "*Block*" 2. Sistem dapat menampilkan halaman "*Block*" laporan yang terdiri dari seluruh daftar laporan yang berstatus *block* 3. Admin menekan tombol "Proses" pada laporan yang ingin dilihat secara *detail* 4. Sistem akan menampilkan halaman "*Detail* Laporan Berstatus *Block*" yang berisi deskripsi laporan, ganti status laporan, *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab
Alternatif Flow	Jika pada menu "*Block*" tidak ada laporan yang berstatus *block* maka sistem akan menampilkan pesan "*No matching records found*"
Post-condition	Admin dapat melihat *detail* laporan berdasarkan status *block*

Tabel 4. 43 *Use Case Scenario* Mengganti Laporan Status *Block*

Use Case untuk Mengganti Laporan Status *Block*	
Kode	SG-3-023
Objective	*Use case* ini menjelaskan admin dapat mengganti status laporan yang berstatus *block*

Actor	Admin
Pre-condition	Admin masuk ke halaman "*Block*"
Main Flow	1. Admin menekan tombol "*Detail*" pada laporan yang dipilih 2. Sistem akan menampilkan halaman "*Detail* Laporan Berstatus *Block*" yang berisi deskripsi laporan, ganti status laporan, *map* yang ditunjukkan dengan adanya *marker* yang sesuai dengan lokasi kejadian serta nama penanggung jawab 3. Admin mengganti status laporan pada halaman "*Detail* Laporan Berstatus *Block*" 4. Sistem menampilkan status laporan yang dipilih 5. Admin menekan tombol "*Save*" jika status sudah terganti
Alternatif Flow	1. Jika laporan masih menunggu untuk pengerjaan maka admin mengganti dengan status "*Waiting*" 2. Jika laporan masih dalam proses pengerjaan maka admin mengganti dengan status "*OnProgress*" 3. Jika laporan sudah selesai ditindaklanjuti maka admin mengganti dengan status *"Done*"
Post-condition	Admin berhasil mengganti status laporan yang berstatus *block*

Tabel 4. 44 *Use Case Scenario* Mencari Laporan Status *Block*

Use Case untuk Mencari Laporan Status *Block*	
Kode	SG-3-024
Objective	*Use case* ini menjelaskan admin dapat mencari laporan berdasarkan status *block*
Actor	Admin
Pre-condition	Admin sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "*Block*" 2. Sistem dapat menampilkan halaman daftar laporan yang berstatus *block* terdiri dari deskripsi laporan, foto laporan, alamat, tanggal post, pelapor, dan jumlah dukungan yang diperoleh laporan 3. Admin mencari laporan yang berstatus *block* pada kolom pencarian 4. Sistem akan menampilkan laporan yang dicari oleh admin

Alternatif Flow	Jika laporan yang dicari oleh admin tidak terdapat pada sistem maka sistem menampilkan pesan "*No matching records found*"
Post-condition	Admin dapat mencari laporan berdasarkan status *block*

Tabel 4. 45 *Use Case Scenario Logout*

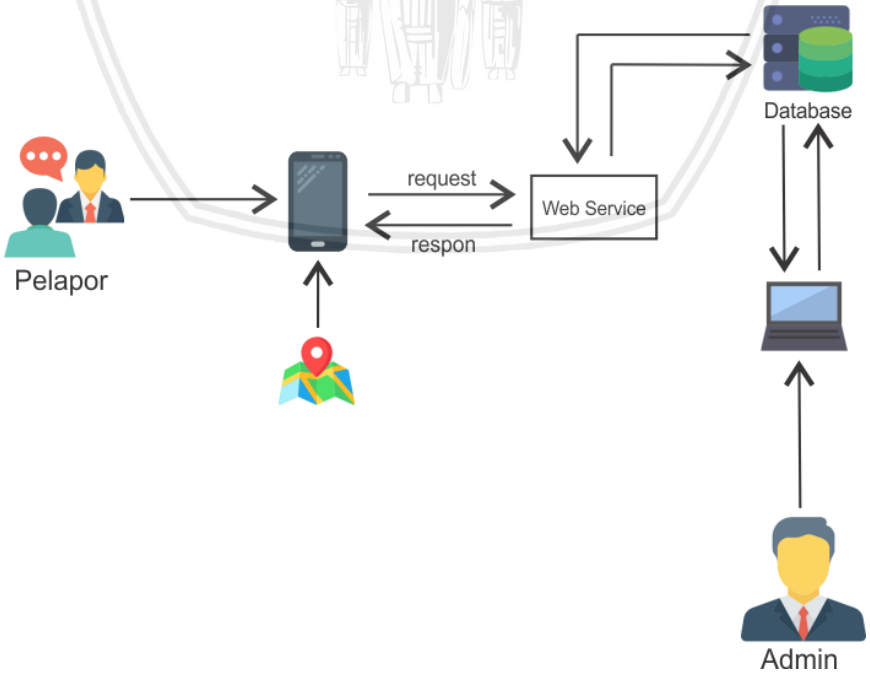
Use Case untuk *Logout*	
Kode	SG-3-025
Objective	*Use case* ini digunakan untuk melakukan *logout*
Actor	Admin
Pre-condition	Pelapor sudah melakukan *login* ke sistem
Main Flow	1. Admin menekan tombol "Logout" pada halaman *Dashboard* 2. Sistem berhasil keluar dari sistem 3. Sistem menampilkan halaman l*ogin*
Alternatif Flow	-
Post-condition	Admin dapat melakukan proses *logout* dari sistem

BAB 5 PERANCANGAN DAN IMPLEMENTASI SISTEM

5.1 Perancangan Sistem

Perancangan sistem merupakan tahapan yang dilakukan berdasarkan hasil dari setelah melakukan proses analisis kebutuhan dimana perancangan sistem ini merupakan tahapan lanjutan dari bab sebelumnya yakni analisis kebutuhan sistem. Perancangan umum sistem menjabarkan mengenai proses kerja dari aktor, basis data, dan perangkat bergerak yang digunakan untuk mengakses basis data tersebut. Tahapan perancangan aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang terdiri dari beberapa bagian, yakni perancangan arsitektur diagram, perancangan komponen, perancangan basis data, dan perancangan antarmuka.

Pada perancangan sistem membahas pula mengenai arsitektur sistem pada pengembangan aplikasi pengaduan masyarakat dimana pada arsitektur menjelaskan bagaimana pelapor atau masyarakat dapat melaporkan pengaduannya dengan mengisikan *form* seperti foto kejadian, deskripsi, kategori keluhan, dan juga lokasi pelaporan pada aplikasi melalui *smartphone* dengan mengaktifkan GPS kemudian dikirimkan kepada admin atau Dinas Perhubungan Kota Malang melalui *web service* yang tersimpan pada *database* dan laporan akan diterima oleh admin. Dan admin dapat melakukan respon terhadap laporan tersebut pada pelapor melalui *web service* dan yang tersimpan terlebih dahulu pada *database* dan respon atau status dari laporan dapat diterima oleh pelapor melalui *smartphone*. Berikut merupakan arsitektur sistem pada pengembangan aplikasi pengaduan masyarakat ini dapat dilihat pada Gambar 5.1 seperti berikut.



Gambar 5. 1 Arsitektur Sistem Pengaduan Masyarakat

5.1.1 Perancangan Arsitektur Diagram

Perancangan arsitektur diagram ini merupakan perancangan yang menjelaskan secara *detail* mengenai *sequence diagram* yakni digunakan untuk menunjukkan rangkaian tahap demi tahap berupa pesan yang dikirimkan antar objek. Pada perancangan arsitektur ini akan menjelaskan tiga *sequence diagram* diantaranya *sequence diagram* untuk menambah laporan, mengubah laporan pribadi, dan melihat laporan berdasarkan status *waiting*, selain itu pada perancangan arsitektur ini akan membahas tentang *class diagram* yakni diagram yang mampu menjelaskan objek sistem dan hubungan dengan objek lainnya. *Class diagram* yang ada perancangan ini yakni *class diagram* dari *mobile app* dan juga *class diagram* dari web.

5.1.1.1 Sequence Diagram

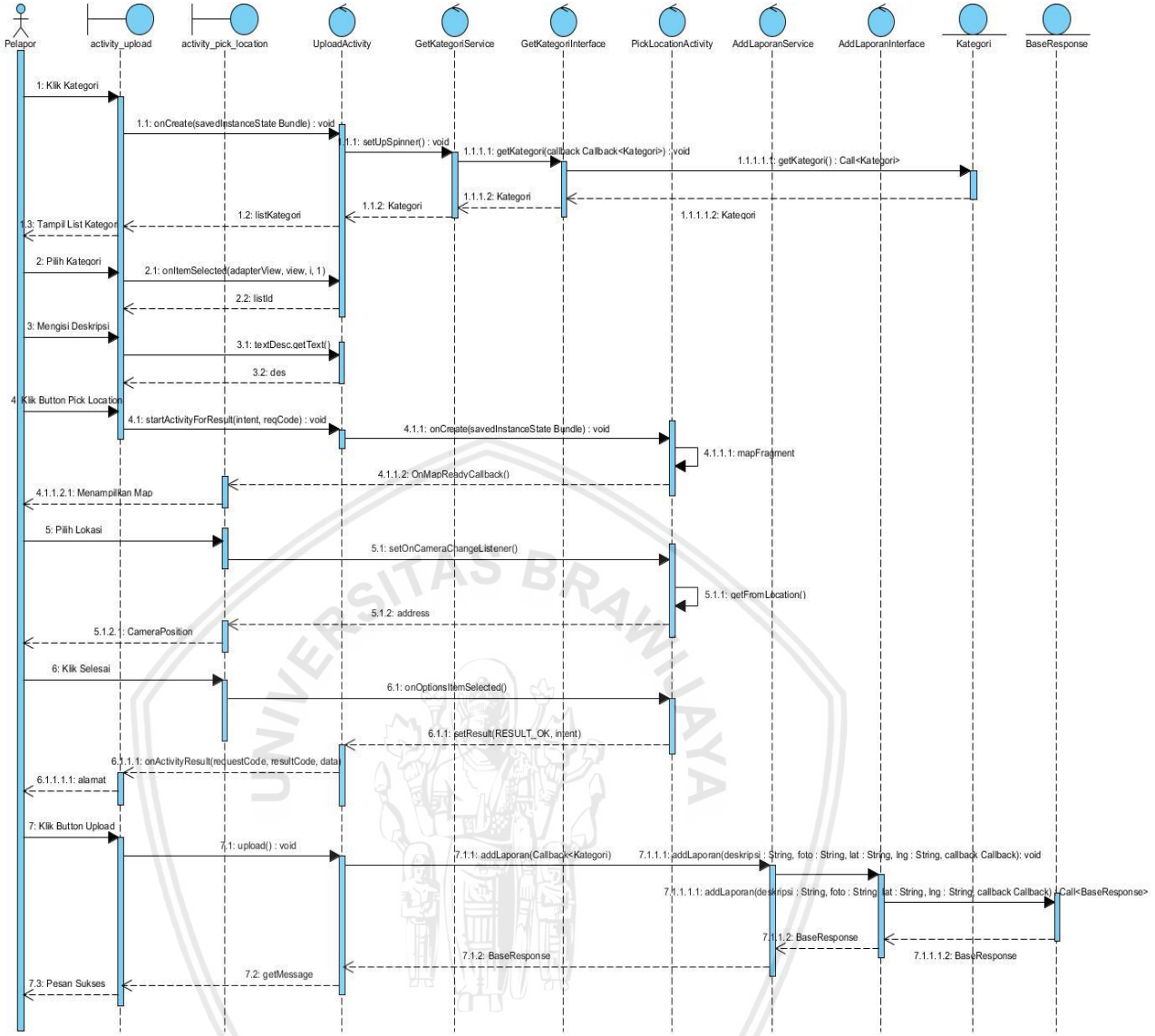
Dalam perancangan *sequence diagram* menjelaskan mengenai tiga *sequence diagram* yang digunakan untuk pengembangan pengaduan masyarakat pada Dinas Perhubungan Kota Malang yakni *sequence diagram* untuk menambah laporan, mengubah laporan pribadi, dan melihat laporan berdasarkan status *waiting*. Berikut merupakan *sequence diagram* yang dapat dilihat pada Gambar 5.2 sampai Gambar 5.4.

5.1.1.1.1. Sequence Diagram Menambah Laporan

Sequence diagram menambah laporan pelapor terdapat pada Gambar 5.2 yang digambarkan dengan menggunakan *lifeline* yang digunakan untuk menggambarkan pertukaran pesan antar objek. Pada *sequence diagram* menambah laporan ini diasumsikan telah mengambil foto maka dari itu pada *sequence diagram* ini hanya dimulai dari proses menginputkan *form* laporan.

Sequence digaram menambah laporan terdiri dari sebelas *lifeline* yakni satu aktor, dua *view*, enam *controller*, dan dua model data. Yang pertama kali dikirim pesan untuk memulai aktifitas untuk menambah laporan adalah pelapor memilih kategori keluhan yang berupa spinner dengan cara *klik spinner* kategori keluhan. *Spinner* kategori tersebut terdapat pada tampilan activity_upload, pelapor yang melakukan proses memilih kategori melakukan proses dengan mengirim pesan pada UploadActivity dengan mengirim *method* onCreate() kemudian UploadActivity mengirim pesan pada GetKategoriService dengan mengirim *method* setUpSpinner() dalam memilih kategori setelah itu GetKategoriService melanjutkan proses dengan mengirim *method* getKategori() untuk mendapatkan kategori pada GetKategoriInterface, selanjutnya *controller* ini mendapatkan dari model Laporan dengan mengirim pesan getKategori(). Model Laporan ini me*return* atau mengembalikan nilai pada GetKategoriInterface berupa Kategori. Pada GetKategoriInterface mengembalikan nilai pada GetKategoriService berupa Kategori, dan juga *controller* ini mengembalikan nilai kembalian kepada *controller* UploadActivity berupa Kategori. Setelah itu yang akan ditampilkan pada *user* atau pelapor berupa *list* kategori keluhan yang diambil dari model data Laporan.

Selanjutnya jika *list* kategori sudah ditampilkan maka pelapor memilih kategori yang diinginkan sesuai dengan permasalahan laporannya dengan cara memilih kategori keluhan pada *view* activity_upload kemudian *view* ini mengirim pesan memilih kategori pada *controller* UploadActivity dengan mengirim pesan onItemSelected(adapterView, view, I, 1) lalu UploadActivity mengembalikan nilai listId pada *user* dengan kategori keluhan yang sudah dipilih berdasarkan id. Selanjutnya pelapor mengisi deskripsi laporan pada tampilan yang sama yakni activity_upload, setelah itu activity_upload melakukan proses dengan mengirim pesan pada UploadActivity berupa *method* textDesc.getText() dan mengembalikan nilai pada *user* berupa des yakni deskripsi. Setelah itu pelapor memilih lokasi permasalahan dengan cara menekan tombol *pick location* pada *view* activity_upload setelah itu *user* akan diarahkan pada *view* activity_pick_location berupa *map* lalu *view* ini mengirim pesan untuk menampilkan *map* tersebut berupa *operation* startActivityFor*Result*(intent, reqCode) pada UploadActivity. Lalu *controller* UploadActivity mengirim pesan pada *controller* PickLocationActivity yakni onCreate() serta pada *controller* PickLocationActivity melakukan proses *self message* atau melakukan proses pada *class* itu sendiri berupa mapFragment. Lalu PickLocationActivity mengembalikan nilai pada *user* akan ditampilkan *map* untuk memilih lokasi. Setelah itu pelapor memilih lokasi pada *view* yang sama yakni activity_pick_location dan *view* tersebut akan diteruskan pada PickLocationActivity dengan mengirim pesan setOnCameraChangeListener() dan pada *controller* PickLocationActivity melakukan proses *self message* berupa getFromLocation() lalu PickLocationActivity mengembalikan nilai pada activity_pick_location berupa *address* dan pada *user* akan ditampilan *map* berupa *marke*r yang dapat digerakkan yakni CameraPosition. Jika *user* sudah mendapatkan lokasi yang diinginkan kemudian *user* memilih *button ceklis* atau selesai pada *view* activity_pick_location dan akan diteruskan pada *controller* PickLocationActivity berupa *operation* onOptionItemSelected() dan *controller* tersebut mengembalikan nilai pada *controller* UploadActivity berupa *method* set*Result*(*RESULT*_OK, intent) dan *user* akan menerima tampilan berupa alamat yang sudah dipilih pada activity_upload. Kemudian *user* menekan tombol *upload* pada view activity_upload dan akan melakukan proses dengan mengirim pesan pada UploadActivity yakni upload() kemudian pesan teresbut akan melanjutkan pada *controller* AddLaporanService berupa addLaporan() dan *controller* AddLaporanService akan meneruskannya pada *controller* AddLaporanInterface berupa addLaporan() dan *controller* ini mnegirim pesan berupa laporan pada model BaseResponse berupa addLaporan() kemudian model data ini mengembalikan nilai pada AddLaporanInterface berupa BaseResponse dan juga controller tersebut juga mengembalikan nilai pada AddLaporanService berupa BaseResponse lalu *controller* AddLaporanService mengembalikan nilai pada UploadActivity berupa BaseResponse setelah proses menambah laporan sukses dilakukan maka pada user akan ditampilkan pesan sukses.



Gambar 5. 2 *Sequence Diagram* Menambah Laporan

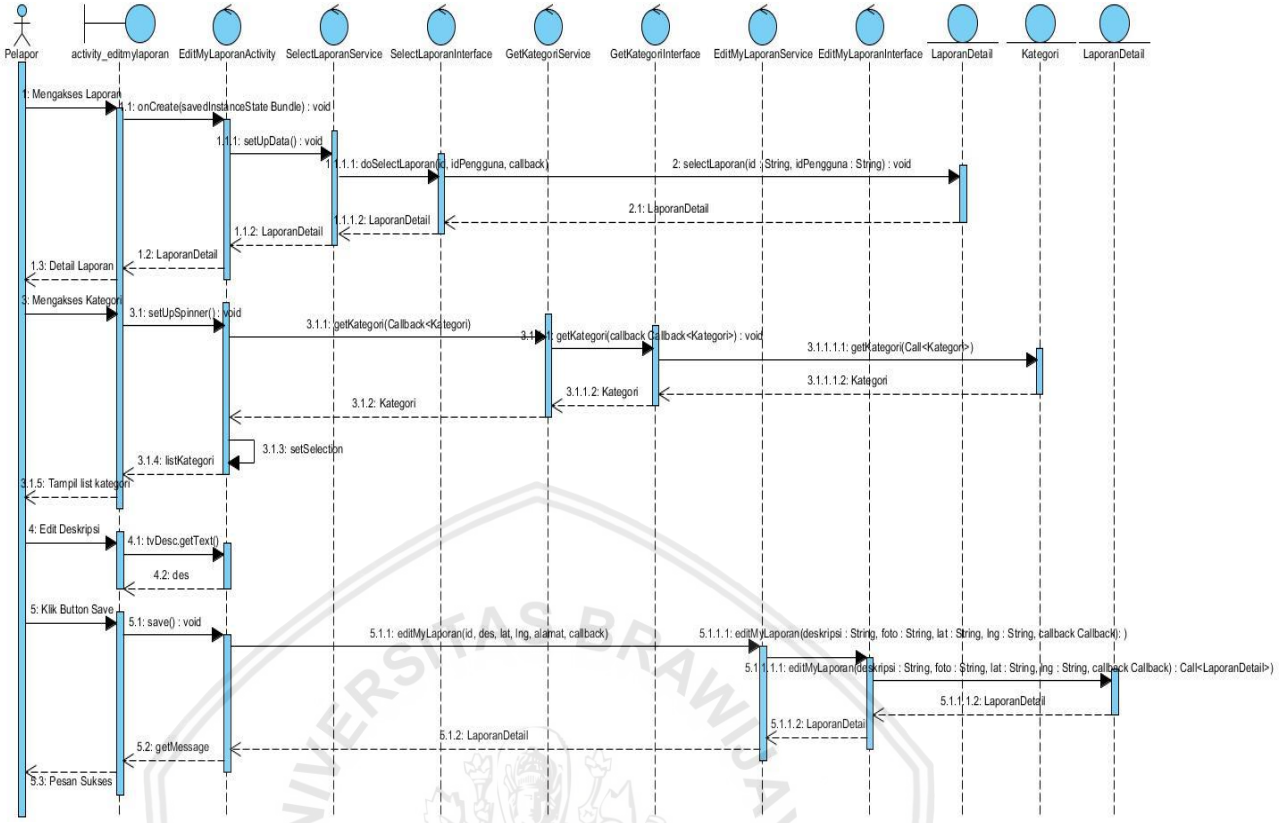
5.1.1.1.2. Sequence Diagram Mengubah Laporan

Sequence diagram mengubah laporan akan merepresentasikan *use case* dari *use case* mengubah laporan yang mengacu pada *use case scenario*. Pada Gambar 5.3 menggambarkan dua belas buah *lifeline* untuk merepresentasikan pertukaran pesan antar objek satu dengan yang lainnya. *Lifeline* tersebut terdiri dari satu aktor, tujuh *controller*, dan juga tiga model data. Satu aktor tersebut yakni pelapor atau masyarakat, satu *boundary* ialah activity_editmylaporan, tujuh controller ialah *controller* EditMyLaporanActivity, *controller* SelectLaporanService, *controller* SelectLaporanInterface, GetKategoriService, GetKategoriInterface, EditMyLaporanService, EditMyLaporanInterface serta tiga model yakni LaporanDetail, Kategori, dan LaporanDetail. Proses mengubah laporan pertama kali dilakukan oleh aktor pelapor dengan mengakses laporan pada *view* activity_editmylaporan kemudian *view* tersebut akan diteruskan pada *controller* EditMyLaporanActivity berupa *method* onCreate() kemudian *controller* tersebut

akan mengirim pesan pada *controller* SelectLaporanService dengan mengirim pesan setUpData() dan SelectLaporanService meneruskan pesan pada SelectLaporanInterface berupa doSelectLaporan() dan *controller* SelectLaporanInterface mengirim pesan pada model data LaporanDetail berupa selectLaporan() lalu model tersebut mengembalikan nilai pada SelectLaporanInterface berupa LaporanDetail dan *controller* teresbut mengirim nilai kembalian pada SelectLaporanInterface berupa LaporanDetail lalu SelectLaporanService menerima nilai kembalian dari SelectLaporanInterface berupa LaporanDetail selanjutnya SelectLaporanService mengembalikan nilai pada EditMyLaporanActivity berupa LaporanDetail dan yang akan ditampilkan pada *user* atau pelapor yakni Data Laporan yang akan diubah.

Data laporan yang muncul pada *user* akan menampilkan kategori yang sudah dipilih sebelum dilakukan perubahan maka terjadi proses setUpSpinner untuk mengakses *spinner* pada controller EditMyLaporanActivity lalu *controller* tersebut akan meneruskan pesan berupa getKategori() pada *controller* GetKategoriService dan GetKategoriService akan mengirim pesan pada GetKategoriInterface berupa getKategori() dan GetKategoriInterface akan mengirim pesan pada model data Kategori berupa getKategori() dan model data tersebut mengembalikan nilai pada GetKategoriInterface berupa Kategori dan juga GetKategoriInterface mengembalikan nilai pada GetKategoriService berupa Kategori selanjutnya *controller* GetKategoriService akan mengembalikan nilai pada EditMyLaporanActivity berupa Kategori dan pada *controller* EditMyLaporanActivity melakukan proses *self message* berupa setSelection tentang kategori yang dipilih kemudian pada tampilan activity_editmylaporan akan menampilkan kategori keluhan yang sudah pernah dipilih sebelumnya.

Pada kasus ini akan dilakukan proses untuk mengubah deskripsi laporan maka *user* atau pelapor melakukan perubahan pada kolom deskripsi pada tvDesc.getText() melalui *controller* EditMyLaporanActivity dan *controller* tersebut melakukan pengembalian nilai berupa des atau deskripsi laporan yang sudah diubah. Kemudian *user* menekan *button save* pada *view* activity_editmylaporan kemudian *view* tersebut akan mengirim pesan pada EditMyLaporanActivity berupa method save() lalu EditMyLaporanActivity akan mengirim pesan pada EditMyLaporanService yaitu editMyLaporan() lalu controller tersebut akan meneruskan pesan pada EditMyLaporanInterface berupa editMyLaporan() lalu EditMyLaporanInterface mengirim pesan pada model LaporanDetail berupa editMyLaporan() selanjutnya model data tersebut mengembalikan nilai pada EditMyLaporanInterface berupa LaporanDetail, lalu EditMyLaporanInterface juga mengembalikan nilai pada EditMyLaporanService berupa LaporanDetail jadi *user* akan mendapatkan pesan sukses jika proses mengubah laporan sukses dilakukan. Jika ingin mengubah laporan dengan mengubah gambar atau foto kejadian maka proses untuk perubahan gambar dengan melakukan pengambilan yang hanya bisa dilakukan melalui *gallery smartphone* pelapor. Berikut merupakan *sequence diagram* untuk mengubah laporan.

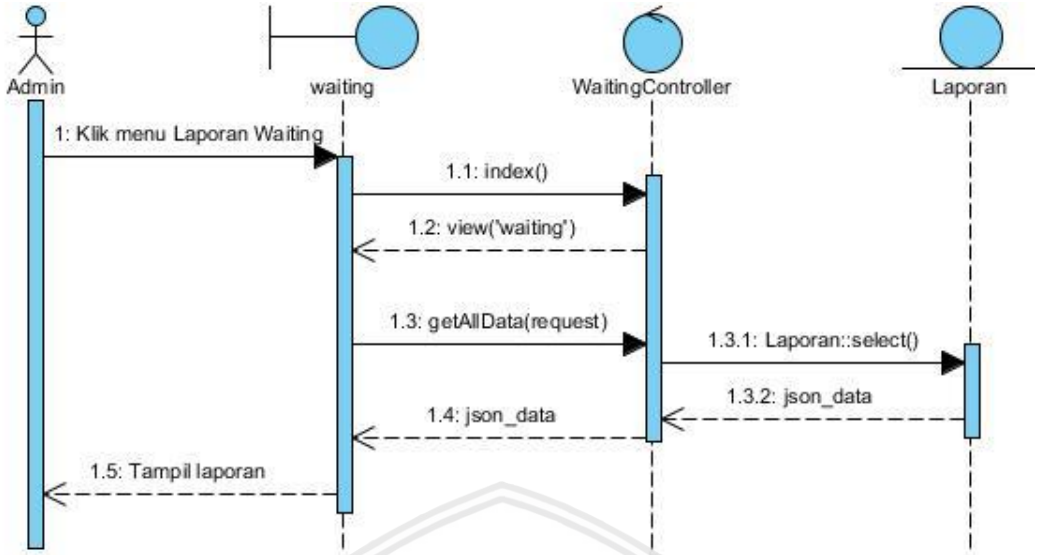


Gambar 5. 3 *Sequence Diagram* Mengubah Laporan

5.1.1.1.3. Sequence Diagram Melihat Laporan Berdasarkan Status *Waiting*

Sequence diagram melihat laporan berdasarkan status *waiting* akan digambarkan berdasarkan *use case* melihat laporan berdasarkan status *waiting* yang sesuai dengan *use case scenario* pada Tabel 4.27 tentang *use case scenario* melihat laporan berdasarkan status *waiting*. Gambar 5.4 menunjukkan *sequence diagram* melihat laporan berdasarkan status *waiting*.

Sequence diagram yang ada pada Gambar 5.4 memiliki empat buah *lifeline*. *Lifeline* tersebut terdiri dari satu *lifeline* aktor, satu *boundary* yakni *waiting*. Dan juga terdiri dari satu *controller* yakni *WaitingController,* dan satu entitas yaitu entitas Laporan. Tahapan pertama yang dilakukan ialah *administrator* atau pihak Dinas Perhubungan Kota Malang dengan klik menu laporan pada *boundary waiting* kemudian pada *boundary* tersebut mengirim pesan pada WaitingController berupa *method* index() lalu *controller* WaitingController mengirim pesan berupa *view* ('waiting') selanjutnya jika halaman laporan *waiting* sudah ditampilkan maka akan dilakukan proses getAllData(request) pada WaitingController dan akan mengirim pesan berupa *query* berupa Laporan::select() pada model data Laporan. Selanjutnya model Laporan mengembalikan nilai berupa json_data pada *controller* WaitingController lalu *controller* tersebut mengembalikan nilai berupa json_data pada *boundary waiting* maka admin akan mendapatkan tampilan berupa seluruh daftar laporan status *waiting*. Gambar 5.4 merupakan *sequence diagram* untuk melihat laporan berdasarkan status *waiting*.



Gambar 5. 4 *Sequence Diagram* Melihat Laporan Berdasarkan Status *Waiting*

5.1.1.2 Class Diagram

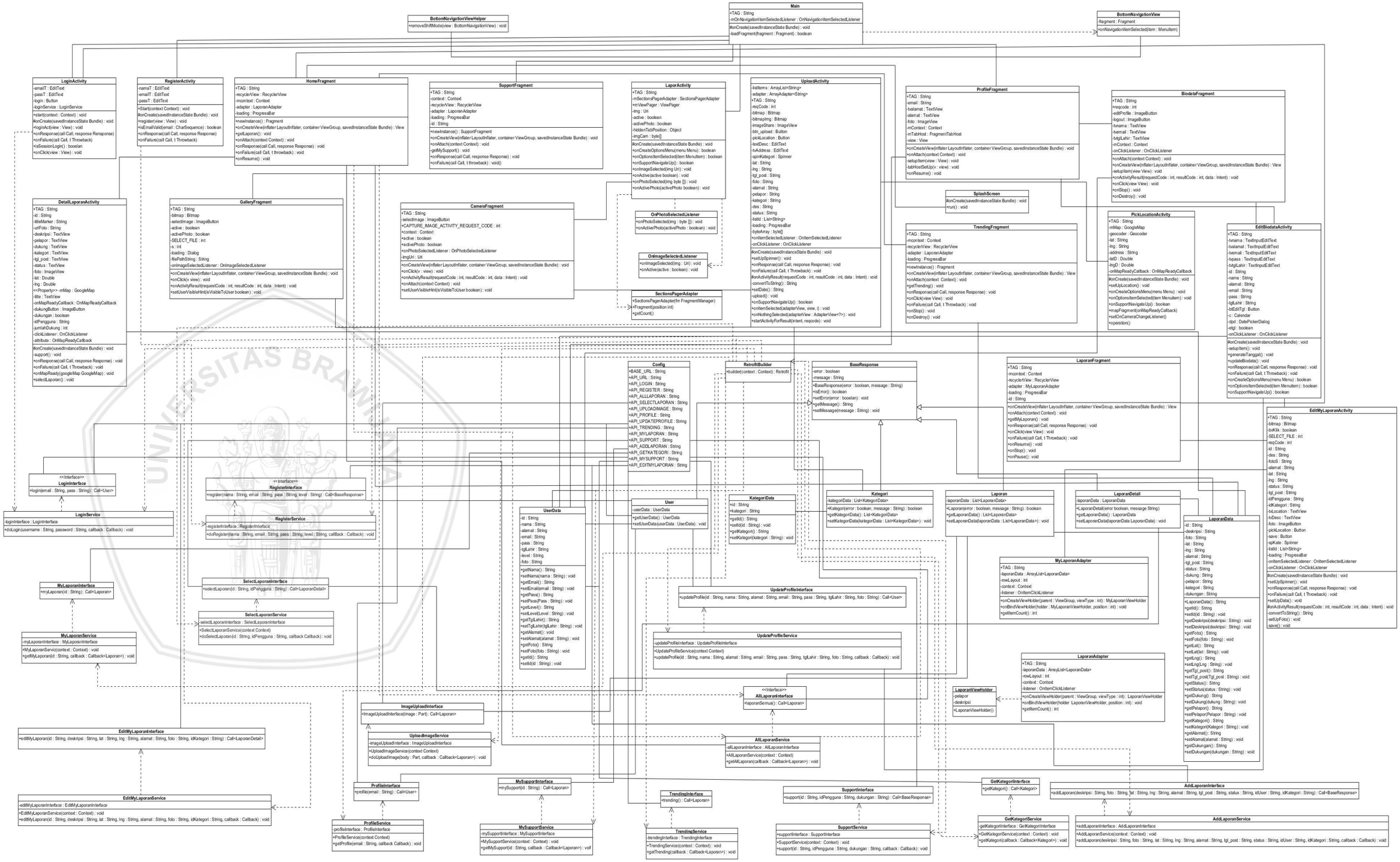
Dalam subbab ini akan menjelaskan mengenai *class diagram* yakni diagram yang mampu menjelaskan objek sistem dan hubungan dengan objek lainnya. Sebuah *class-class* pada diagram ini membentuk aplikasi *mobile* untuk pengaduan masyarakat pada Dinas Perhubungan Kota Malang dan juga hubungan antar *class-class* tersebut. *Class diagram* pada penelitian ini terdapat dua *class diagram* yakni *class diagram* untuk *mobile app* yang digunakan oleh pelapor atau masyarakat dan *class diagram* untuk web yang digunakan oleh admin atau pihak Dinas Perhubungan Kota Malang. Berikut merupakan penjabaran dari *class diagram* dapat ditunjukkan pada Gambar 5.5 untuk *class diagram mobile app* dan Gambar 5.6 merupakan *class diagram* untuk web atau yang digunakan untuk admin.

Pada *class diagram* yang ditunjukkan pada Gambar 5.5 untuk *class diagram mobile apps* terdapat 63 *class* dalam pembuatan aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang yakni terdiri dari *activity, fragment, interface,* dan model. Dimana *activity class* merupakan *class* yang berhubungan dan berinteraksi dengan *layout* antara *user* dan *layout* antarmuka. Pada *class diagram* terdapat 10 *Activity* yakni DetailLaporanActivity, EditBiodataActivity, EditMyLaporanActivity, Main, *LoginActivity*, *RegisterActivity*, *UploadActivity*, LaporActivity, SplashScreen, dan *PickLocationActivity*. *Class* lain yang berhubungan antara interaksi *user* dengan antarmuka selain *activity* ialah *fragment* dimana *fragment* ini merupakan *layout* yang berada pada sebuah *activity* yang mana *fragment* ini tidak dapat melakukan proses sendiri karena berada pada sebuah *activity*, pada *class diagram* yang dirancang terdapat delapan *fragment* yaitu *HomeFragment*, *CameraFragment*, *GalleryFragment*, *BiodataFragment*, dan juga *ProfileFragment*, *LaporanFragment, SupportFragment,* dan juga TrendingFragment. Fragment tersebut akan mengambil interaksi dari *user* dan meneruskannya kepada *model*. *Model-model* yang terdapat pada *class diagram* ini diantaranya BaseResponse, Laporan,

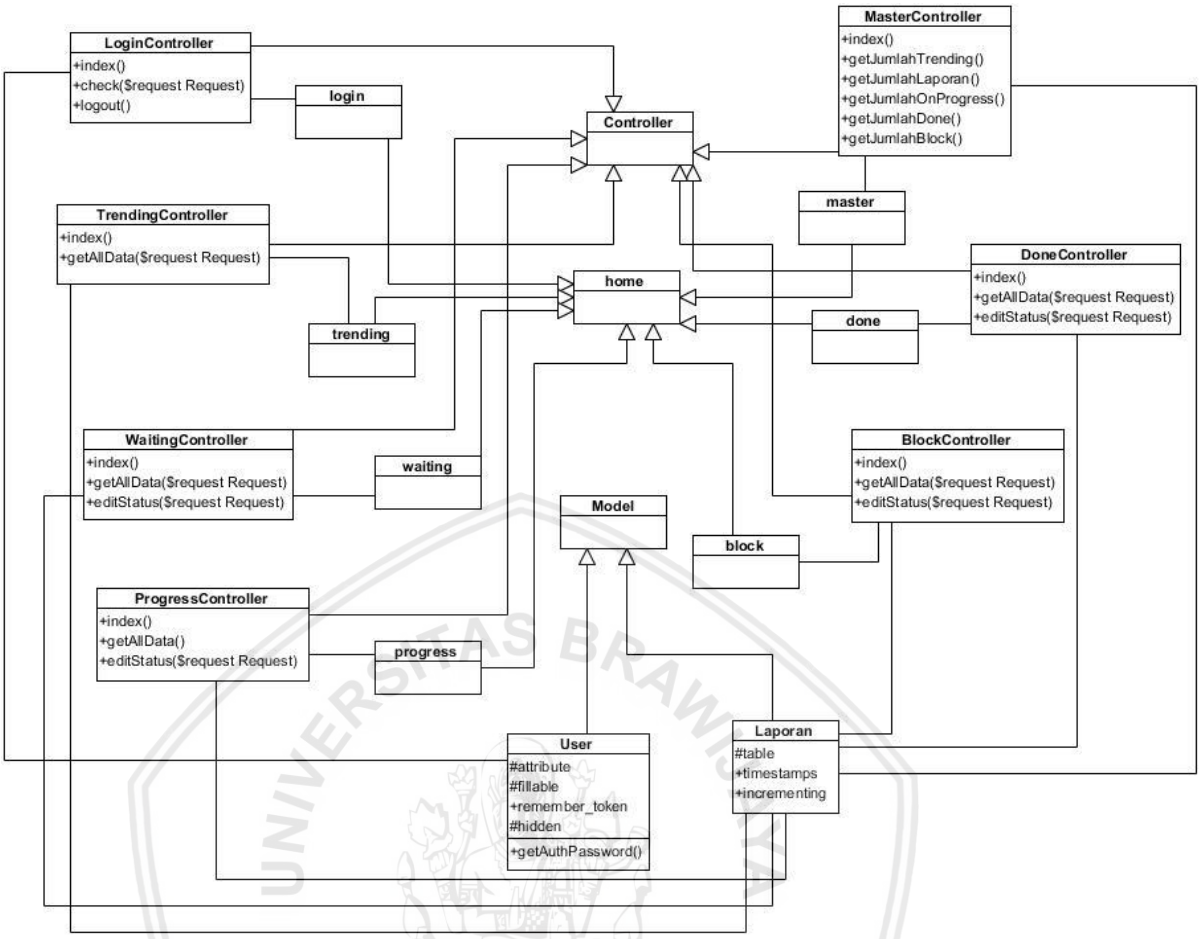
LaporanData, Laporan*Detail*, *User, User*Data, Kategori, dan juga KategoriData. Pada *Class User* berfungsi untuk mengelola proses yang berkaitan dengan *class UserData* sedangkan *class* Laporan berfungsi untuk mengelola proses dan memanggil *method* pada *class LaporanData* serta pada *class* Kategori berfungsi untuk mengelola proses yang berkaitan dengan *class* KategoriData.

Pada *class diagram* ini terdapat beberapa *class*. *Class* yang pertama yakni *class Login*Activity dimana *class* tersebut memiliki hubungan dengan Main yakni *class* Main adalah *class* yang dituju setelah melakukan proses *login*. Untuk melakukan proses *login* tersebut digunakan sebuah *class* berupa *class* model untuk mendapatkan data *user* pada *class User*Data yang digunakan untuk menjalankan *controller* tersebut. Pada *class diagram* aplikasi pengaduan masyarakat ini memiliki *class* utama yang digunakan untuk menu-menu yang ada pada aplikasi ini yakni menu DaftarLaporan yang digunakan untuk melihat seluruh daftar laporan yang dilaporkan oleh masyarakat menu ini menggunakan *class* HomeFragment dengan mengambil data dari model LaporanData. Menu yang kedua ialah menu *Trending*, *class* yang digunakan yakni *class Trending*Fragment dimana *class* ini dapat digunakan untuk melihat *trending* laporan yang memiliki banyak *support* dari masyarakat lain yang memiliki keluhan yang sama dengan pelapor yang melaporkan laporan yang menjadi *trending* tersebut. Menu yang ketiga yakni menu tambah laporan yang menggunakan *class* LaporFragment, pada menu ini memiliki beberapa *activity* dan *fragment* dalam membangun menu ini yakni *Gallery*Fragment dan *Camera*Fragment, serta UploadActivity. Menu ini digunakan untuk menambah laporan kemudian dihubungkan dengan model LaporanData. Dan menu yang terakhir ialah menu *Profile*, pada menu ini digunakan untuk mengelola data-data *Profile* atau biodata dari pelapor.

Gambar 5.6 merupakan *class diagram* pada web yang merupakan *class-class* yang digunakan untuk meyusun sistem pengaduan masyarakat untuk admin. Pada *class diagram web* ini memiliki 19 *class* dalam pembangunan website sistem pengaduan masyarakat. Dimana pada *class diagram web* memiliki 8 *class controller* yakni controller, MasterController, LoginController, TrendingController, WaitingController, ProgressController, DoneController, dan juga BlockController. Pada *class diagram web* juga memiliki *class view* sebanyak 8 *view* yang terdiri *home, master, login, trending, waiting, progress, done*, dan juga *block*. Lalu pada *class diagram* pada web memiliki 2 model data yang digunakan yakni *user* dan laporan. *Controller* merupakan *controller* yang dijadikan *controller* utama maka dari itu masing-masing controller berasosiasi dengan *controller* ini. Untuk *view* pun sama halnya seperti *controller*, *home* merupakan *view* utama sehingga *home*, *master, login, trending, waiting, progress, done*, dan *block* saling berasosiasi dengan *home*. Dan juga model *user* dan laporan berhubungan dengan model utama. LoginController berhubungan dengan *view home* dan juga berhubungan model *User*. TrendingController memiliki hubungan dengan *view trending* dan model data Laporan, dan begitupun dengan *controller* lainnya.



Gambar 5. 5 *Class Diagram Mobile App*



Gambar 5. 6 *Class Diagram Web* Pengaduan Masyarakat

5.1.2 Perancangan Komponen

Perancangan komponen pada aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang ini menjelaskan mengenai proses yang ada dalam komponen sistem, dimana proses tersebut akan tersusun dari adanya sebuah *class*. Sebuah subsistem dari tiap komponen akan dijelaskan rinciannya secara *detail* pada suatu perangkat lunak dalam perancangan komponen ini. Pada perancangan komponen akan dijelaskan beberapa komponen yang merupakan algoritam utama dari aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang yang merupakan fungsi utama dari sistem ini yakni *method upload()* dari *class UploadActivity*, *method save()* dari *class EditMyLaporanActivity*, dan *method editStatus()* dari *class WaitingController*. Berikut merupakan perancangan komponen secara *detail* dari sistem pengaduan masyarakat dapat dilihat pada Subbab 5.1.2.1 sampai 5.1.2.3.

5.1.2.1 Perancangan Komponen *Method "upload"* dari *Class "UploadActivity"*

Untuk membuat algoritme menambah laporan menggunakan *method upload()* dari *class UploadActivity*. *Method* ini digunakan untuk membangun aplikasi dengan fitur menambah laporan, *method upload*() berasal dari *class UploadActivity* yang berisi getdes, getalamat, foto convert to string, set date, get

id from UserData yang digunakan dalam menambah laporan dimana jika parameter-parameter tersebut berhasil dimasukkan dan *object call* dan *response* tidak kosong makan BaseResponse akan mengirimkan *response* dan laporan yang sudah ditambahkan tersebut akan masuk pada halaman main dan jika *object call* dan *response* atau parameter ada yang kosong makan akan muncul pesan "Terjadi Kesalahan, Silahkan Ulangi Kembali!" setelah itu proses menambah laporan selesai. Perancangan komponen menambah laporan dapat ditunjukkan pada Tabel 5.1.

Tabel 5. 1 Perancangan Algoritme *Method* "*upload*" dari *Class* "*UploadActivity*"

No	*Pseudocode*
1	`implementation method upload()`
2	`   getdes to string();`
3	`   foto convert to string();`
4	`   getalamat to string();`
5	`   setdate tgl_post();`
6	`   getid pelapor from UserData();`
7	
8	`   addLaporanService = implementation method addLaporan with parameter des, foto, lat, lng, alamat, tgl_post, status, pelapor, kategori`
9	`      if (!call = null && !response = null)`
10	`          BaseResponse send response.body();`
11	`          start activity with main page();`
12	`      else`
13	`          show message "Terjadi Kesalahan, Silahkan Ulangi Kembali!"`
14	`      endif`

Berikut merupakan penjelasan dari *pseudocode* dengan *method upload()* dari *class UploadActivit*y dapat ditunjukkan pada Tabel 5.2.

Tabel 5. 2 Penjelasan *Pseudocode Method* "*upload*" dari *Class* "*UploadActivity*"

Baris	Penjelasan
1	Implementasi *method* upload
2 - 7	Kondisi apabila deskripsi, foto, alamat, tanggal post sudah terisi
8	Mengambil API addLaporan
9 - 11	Kondisi jika *object call* dan *response* tidak kosong maka akan mengirimkan *response* dan laporan yang ditambahkan akan masuk ke halaman main
12 - 13	Kondisi apabila *object call* dan *response* kosong makan akan muncul pesan "Terjadi Kesalahan, Silahkan Ulangi Kembali!"
14	Proses selesai

5.1.2.2 Perancangan Komponen *Method "save" dari Class "EditMyLaporanActivity"*

Untuk merancang algoritme mengubah biodata menggunakan *method* "save" dari *class* "EditMyLaporanActivity". Pada *method* save ini berasal dari *class* EditMyLaporanActivity yang berisi *method* untuk melakukan *update* laporan pribadi dengan melakukan implementasi dari *method* save yang berisi getdes, getalamat, foto convert to string. Jika parameter-parameter tersebut berhasil dimasukkan dan *object call* dan *response* tidak kosong makan BaseResponse akan mengirimkan *response* dan muncul pesan sukses dan jika *object call* dan *response* atau parameter ada yang kosong makan akan muncul pesan "Terjadi Kesalahan!" setelah itu proses mengubah laporan pribadi selesai dilakukan. Perancangan komponen *edit* laporan pribadi pada *method* save dan *class* EditMyLaporanActivity dapat ditunjukkan pada Tabel 5.3.

Tabel 5. 3 Perancangan Algoritme *Method* "save" dari *Class* "EditMyLaporanActivity"

No	*Pseudocode*
1	`implementation method save()`
2	`   getdes to string();`
3	`   foto convert to string();`
4	`   getalamat to string();`
5	
6	`   editMyLaporanService = implementation method editMyLaporan with parameter id, des, fotoS, lat, lng, alamat, idKategori`
7	`      if (!call = null && !response = null)`
8	`         BaseResponse send response.body();`
9	`      else`
10	`         show message "Terjadi Kesalahan"`
11	`      endif`

Tabel 5.4 merupakan penjelasan dari algoritme mengubah laporan pribadi dengan *method* "updateBiodata" dari *class* "EditBiodataActivity"

Tabel 5. 4 Penjelasan *Pseudocode Method* "updateBiodata" dari *Class* "EditBiodataActivity"

Baris	Penjelasan
1	Implementasi *method* save
2 - 5	Kondisi apabila deskripsi, foto, dan alamatsudah terisi
6	Mengambil API editMyLaporan
7 - 8	Kondisi jika *object call* dan *response* tidak kosong maka akan mengirimkan *response*
9 - 10	Kondisi apabila *object call* dan *response* kosong makan akan muncul pesan "Terjadi Kesalahan!"

11	Proses selesai

5.1.2.3 Perancangan Komponen *Method "editStatus()"* dari *Class "WaitingController"*

Perancangan komponen untuk melihat laporan berdasarkan status waiting dirancang berdasarkan *method* "editStatus" dari *class* "WaitingController". Fitur melihat laporan berdasarkan status *waiting* dimana menu laporan *waiting* ini dapat melihat semua laporan-laporan yang dilaporkan masyarakat yang berstatus *waiting*. Tabel 5.5 merupakan perancangann algoritme untuk melihat laporan berdasarkan status *waiting*.

Tabel 5. 5 Perancangan Algoritme *Method* "editStatus()" dari *Class* "WaitingController"

No	*Pseudocode*
1	`implementasi method editStatus()`
2	`    request id;`
3	`    request status;`
4	`    select laporan from model Laporan;`
5	
6	`    if(laporans != null){`
7	`        show message "Data Berhasil di Proses";`
8	`        response id;`
9	`        response status;`
10	`    else`
11	`        show message "Data Tidak Berhasil di Proses";`
12	`    endif`
13	`endif`

Tabel 5.6 menjelaskan proses dari *pseudocode* dengan *method* "editStatus" yang berada pada *class* "WaitingController".

Tabel 5. 6 Penjelasan *Pseudocode Method* "editStatus" dari *Class* "WaitingController"

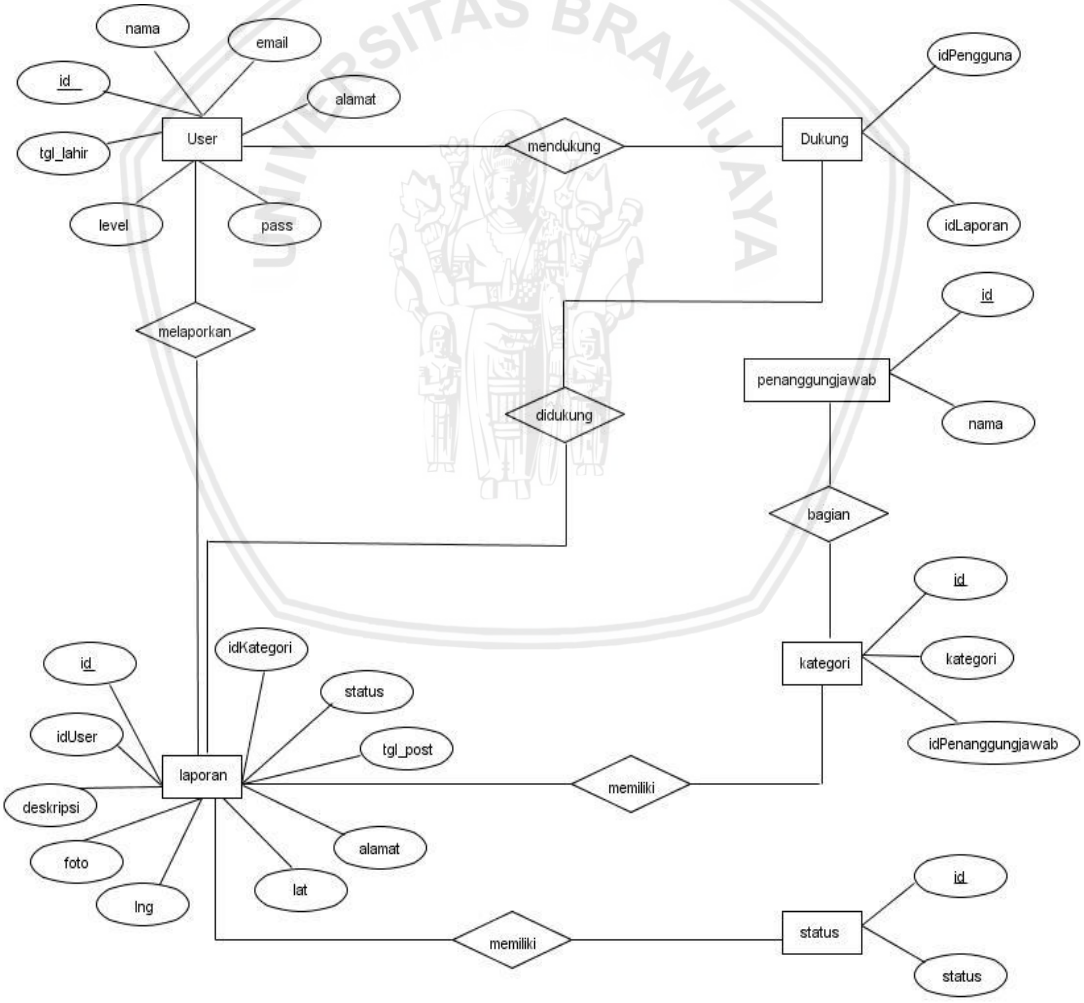
Baris	Penjelasan
1	Implementasi *method* editStatus
2 - 3	*Request* id dan status
4 - 5	Proses mengambil data laporan dari model data Laporan
6	Kondisi jika laporan yang dipilih tidak kosong
7 - 9	Menampilkan pesan "Data Berhasil di Proses" pada id dan status yang dipilih
10 - 12	Kondisi jika laporan yang dipilih kosong maka akan muncul pesan "Data Tidak Berhasil di Proses"

13	Proses selesai

5.1.3 Perancangan Basis Data

Perancangan data pada aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang berbasis *mobile* menggunakan ERD atau *Entity Relationship Diagram*. Gambar perancangan data dapat dilihat pada Gambar 5.7.

Perancangan basis data pada Gambar 5.7 menunjukkan terdapat enam tabel atau entitas yakni tabel *user*, laporan, dukung, penanggung jawab, status, dan kategori. Entitas pertama ialah entitas *user*, entitas ini digunakan untuk menyimpan data pengguna yang mengakses aplikasi. Entitas ini memiliki relasi dengan entitas laporan yakni memiliki relasi *one to many* dimana *user* memiliki banyak laporan dan entitas dukung yakni memiliki entitas *one to many* juga karena *user* dapat banyak mendukung laporan.



Gambar 5. 7 *Entity Relationship Diagram*

Entitas selanjutnya ialah entitas status, entitas ini digunakan untuk menyimpan data mengenai nama-nama status laporan yang ada pada aplikasi ini. Entitas status

memiliki relasi dengan entitas laporan yaitu memiliki relasi *one to one* dimana laporan hanya memiliki satu status. Entitas ketiga ialah entitas laporan, entitas ini merupakan entitas untuk menyimpan data tentang laporan keluhan yang diunggah oleh pelapor. Entitas laporan memiliki relasi dengan entitas dukung, entitas kategori serta entitas status. Dimana relasi antara entitas laporan dengan entitas dukung yakni memiliki relasi *one to many*, sedangkan entitas laporan dengan entitas kategori memiliki relasi *one to many* karena pada laporan hanya memiliki satu kategori serta relasi antara entitas laporan dengan entitas status ialah one to many laporan hanya memiliki satu status laporan. Entitas penanggung jawab merupakan tabel yang berisi nama-nama dari penanggung jawab yang menindaklanjuti laporan berdasarkan jenis pekerjaan dengan kategori laporan. Lalu entitas berikutnya ialah entitas kategori, pada entitas ini berisi nama-nama kategori dari setiap permasalahan yang terjadi dimana entitas ini memiliki relasi dengan entitas penanggung jawab yakni relasi *one to one* karena satu penanggung jawab hanya menangani satu laporan berdasarkan kategori. Dan entitas yang terakhir ialah entitas dukung, entitas ini berisi data-data id pengguna dengan id laporan yang didapatkan dari id pada entitas *User* dan juga entitas Laporan.

Setiap entitas-entitas yang sudah dijabarkan merupakan representasi tabel yang akan dibuat pada *database*. Sedangkan atribut pada tiap entitas merupakan representasi *field* dari tabel pada basis data. Tabel *user* merupakan representasi dari entitas *user* yang digunakan untuk masuk ke aplikasi dengan memasukkan nama dan *password*. Untuk dapat mengetahui secara *detail* perancangan struktur tabel pada *database* ini dapat dilihat pada Tabel 5.7.

Tabel 5. 7 Perancangan Struktur Tabel *User*

No.	Nama *Field*	Tipe Data	Keterangan
1.	Id	Varchar	Id yang dimiliki pengguna
2.	Nama	Varchar	Nama dari pengguna yang mengakses aplikasi
3.	Alamat	Varchar	Alamat dari pengguna yang mengakses aplikasi
4.	email	Varchar	*Email* yang dimiliki pengguna untuk melakukan r*egister*
5.	*pass*	Varchar	*Password* yang dimiliki pengguna untuk dapat l*ogin*
6.	tgl_lahir	Date	Tanggal lahir dari pengguna yang mengakses aplikasi
7.	level	Integer	Level merupakan hak akses untuk pengguna yang membedakan pelapor dan admin

Rancangan struktur tabel laporan dapat dilihat pada Tabel 5.8. Struktur tabel ini representasi dari entitas laporan. Atribut yang ada pada entitas laporan akan menjadi *field* pada tabel laporan. Rancangan struktur tabel laporan dapat dilihat pada tabel berikut.

Tabel 5. 8 Perancangan Struktur Tabel Laporan

No.	Nama *Field*	Tipe Data	Keterangan
1.	id	Varchar	Id yang dimiliki laporan
2.	deskripsi	Varchar	Deskripsi untuk menjelaskan keluhan yang dilaporkan
3.	foto	Text	Foto untuk laporan yang akan diunggah berisi url dari foto tersebut
4.	lat	Double	Koordinat untuk lokasi dalam mengunggah foto pada garis lintang
5.	lng	Double	Koordinat untuk lokasi dalam mengunggah foto pada garis bujur
6.	alamat	Varchar	Alamat yang digunakan untuk melaporkan lokasi kejadian yang ingin dilaporkan
7.	tgl_post	Date	Tanggal pada saat mengunggah laporan
8.	status	Integer	Status dari laporan apakah sudah ditindaklanjuti atau belum
9.	id*User*	Varchar	Id yang dimiliki pengguna
10.	idKategori	Varchar	Id untuk kategori keluhan

Struktur tabel selanjutnya ialah tabel status yang dapat dijabarkan pada Tabel 5.9. Tabel Status ini digunakan untuk menyimpan data status dari laporan seperti laporan dengan status *waiting, onprogress, done*, dan *block*. Terdapat dua *field* pada entitas tabel ini yakni *id,* dan status.

Tabel 5. 9 Perancangan Struktur Tabel Status

No.	Nama *Field*	Tipe Data	Keterangan
1.	id	Varchar	Id dari status laporan

2.	status	Varchar	Status dari laporan yang dilaporkan

Tabel kategori merupakan representasi dari entitas kategori yang digunakan untuk memilih kategori keluhan untuk melaporkan berdasarkan permasalahan yang ingin dilaporkan. Untuk dapat mengetahui secara *detail* perancangan struktur tabel pada *database* ini dapat dilihat pada Tabel 5.10.

Tabel 5. 10 Perancangan Struktur Tabel Kategori

No.	Nama *Field*	Tipe Data	Keterangan
1.	id	Varchar	Id dari kategori
2.	kategori	Varchar	Kategori keluhan yang digunakan untuk memilih kategori dalam melaporkan keluhan
3.	idPenanggungjawab	Varchar	Id yang dimiliki oleh penanggungjawab yang menindaklanjuti laporan

Tabel 5.11 merupakan tabel dukung dimana representasi dari entitas dukung yang digunakan untuk mendukung laporan yang dilaporkan oleh pelapor lain. Tabel dukung memiliki dua *field* yang berisi idLaporan dan idPengguna. idLaporan merupakan id yang didapatkan dari id pada tabel *user* dan idPengguna merupakan id yang didapatkan dari id pada tabel laporan. Tabel berikut merupakan penjelasan secara *detail* dari masing-masing *field* beserta tipe data yang digunakan.

Tabel 5. 11 Perancangan Struktur Tabel Dukung

No.	Nama *Field*	Tipe Data	Keterangan
1.	idLaporan	Varchar	Id yang dimiliki oleh laporan
2.	idPengguna	Varchar	Id yang dimiliki oleh pelapor

Tabel penanggungjawab digunakan untuk menyimpan data dari penanggungjawab yang menindaklanjuti laporan yang disampaikan oleh masyarakat. Untuk dapat melihat secara *detail* struktur tabel dari tabel penanggungjawab dapat dilihat pada Tabel 5.12.

Tabel 5. 12 Perancangan Struktur Tabel Penanggungjawab

No.	Nama *Field*	Tipe Data	Keterangan
1.	id	Varchar	Id yang dimiliki oleh penanggungjawab

2.	nama	Varchar	Nama dari penanggungjawab yang menindaklanjuti laporan dari masyarakat

5.1.4 Perancangan Antarmuka

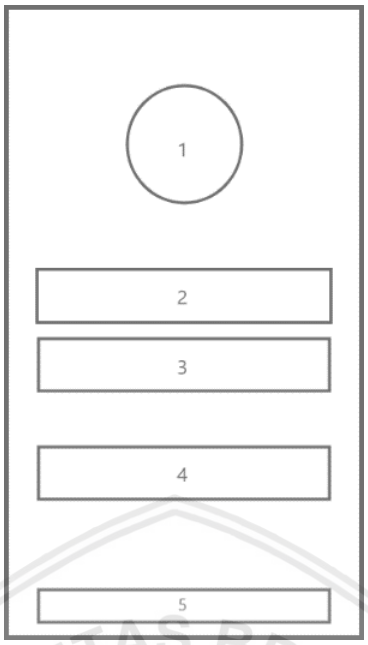
Perancangan antarmuka ini akan menjelaskan mengenai rancangan antarmuka dari aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang menggunakan fitur *location based service*. Pada perancangan antarmuka ini terdiri dari dua sistem yakni sistem yang digunakan oleh pelapor atau masyarakat dengan menggunakan *mobile app* sedangkan sistem lainnya menggunakan *web* yang nantinya akan digunakan oleh admin atau pihak dari Dinas Perhubungan Kota Malang. Seluruh halaman yang ada pada *mobile app* maupun *web* nantinya ada pada dua sistem tersebut untuk itu dilakukan rancangan terlebih dahulu, dengan demikian dengan adanya rancangan antarmuka ini digunakan agar menjaga konsistensi dari rancangan antarmuka dengan pada saat implementasi pada sistem. Rancangan antarmuka tersebut digambarkan dengan perancangan *layout* berupa *wireframe*.

5.1.4.1 Perancangan Antarmuka *Mobile App*

Perancangan antarmuka yang pertama untuk rancangan *layout* aplikasi pengaduan masyarakat untuk *mobile app* yang nantinya akan digunakan oleh pelapor atau masyarakat. Pada perancangan antarmuka untuk aplikasi *mobile* pengaduan masyarakat ini terdapat beberapa halaman. Dimana terdiri dari lima halaman menu utama, seperti menu daftar laporan yang berisi seluruh laporan-laporan yang dilaporkan oleh masyarakat, menu *trending* laporan yang digunakan untuk melihat laporan yang menjadi *trending*, menu *support* ialah menu yang digunakan untuk melihta laporan yang pernah di dukung oleh pengguna, menu menambah laporan ialah menu untuk menambah laporan baru dengan mengambil gambar terlebih dahulu, dan menu terakhir ialah menu *profile* yang mengelola biodata dan laporan pengguna tersebut. Berikut merupakan perancangan antarmuka untuk *mobile app* yang dapat dilihat pada Gambar 5.8 sampai Gambar 5.19.

1. Halaman *Login*

Halaman *login* untuk *mobile app* dapat dilihat pada Gambar 5.8. Halaman *login* ini merupakan halaman awal yang pertama muncul dari *mobile app* pada saat aplikasi tersebut dijalankan setelah sebelumnya terdapat *splash*. Pengguna yang akan menggunakan aplikasi ini harus melakukan *login* terlebih dahulu dengan menggunakan *email* dan *password* yang sudah didaftarkan sebelumnya. Pada halaman *login* ini merupakan sebuah *form* untuk mengisi *email* dan *password* dimana terdapat dua kolom untuk memasukkan *email* dan *password* dan juga terdapat satu tombol berupa *button* untuk menjalankan *login* tersebut setelah memasukkan *email* dan *password*. Berikut merupakan gambar perancangan antarmuka *login mobile app*.



Gambar 5. 8 Perancangan Antarmuka Halaman *Login Mobile App*

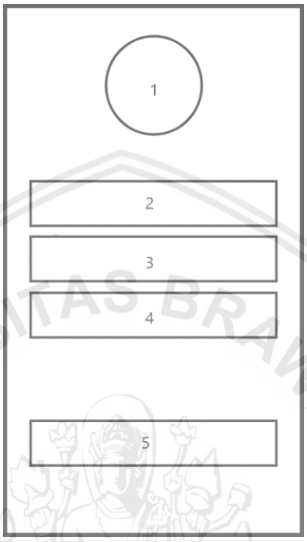
Berikut merupakan penjelasan dari Gambar 5.8 mengenai perancangan antarmuka untuk halaman *login* pada *mobile app*, penjelasan berikut akan menjelaskan secara rinci berdasarkan nama objek dan tipe yang digunakan, perancangan dapat ditunjukkan pada Tabel 5.13.

Tabel 5. 13 Penjelasan Perancangan Antarmuka Halaman *Login Mobile App*

No	Nama Objek	Tipe	Keterangan
1	Logo Aplikasi	Gambar	Logo aplikasi yakni logo dari Dinas Perhubungan Kota Malang
2	*Email*	*Text*	Kolom yang berupa *text* untuk memasukkan *email* pengguna untuk masuk pada aplikasi pengaduan masyarakat
3	*Password*	*Text*	Kolom yang berupa *text* untuk memasukkan *password* pengguna pada saat l*ogin*
4	*Login*	Tombol	Tombol untuk *login* atau untuk mengarahkan ke halaman *home*
5	*Register*	*Text*	*Text* untuk mengarahkan ke halaman *register* jika belum memiliki akun

2. Halaman *Register*

Halaman *register* ini merupakan halaman ketika pengguna akan membuat akun untuk dapat mengakses aplikasi ini dengan mengakses halaman *register* ini yang terletak pada halaman *login*. Pengguna yang akan mendaftarkan diri pada halaman *register* ini dengan memasukkan data diri yang dibutuhkan. Pengguna pada halaman ini harus menginputkan data diri seperti, *username*, *email*, dan *password* yang akan digunakan untuk mengakses aplikasi ini. Perancangan antarmuka untuk halaman *register* terdapat pada Gambar 5.9 seperti berikut.



Gambar 5. 9 Perancangan Antarmuka Halaman *Register Mobile App*

Tabel 5.14 merupakan penjelasan dari perancangan antarmuka halaman *register* pada *mobile app* berdasarkan pada Gambar 5.9. Penjelasan berikut menjelaskan secara *detail* dari masing-masing bagian berdasarkan nama objek dan tipe yang membangun halaman antarmukan untuk *register* pada *mobile app* sebagai berikut.

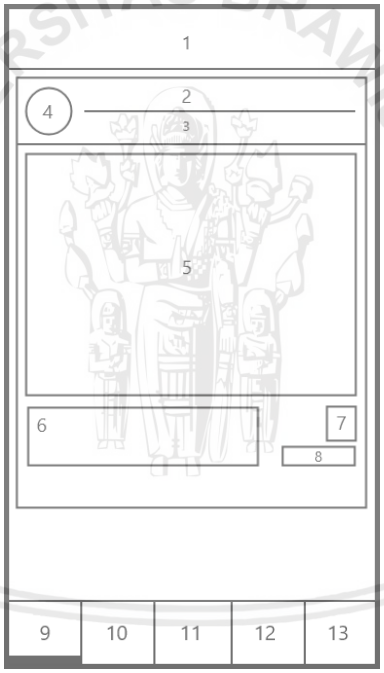
Tabel 5. 14 Penjelasan Perancangan Antarmuka Halaman *Register Mobile App*

No	Nama Objek	Tipe	Keterangan
1	Logo Aplikasi	Gambar	Logo aplikasi yakni logo dari Dinas Perhubungan Kota Malang
2	*User*name	*Text*	Kolom yang berupa *text* untuk memasukkan *username* pengguna baru pada saat r*egister*
3	*Email*	*Text*	Kolom yang berupa *text* untuk memasukkan *email* pengguna baru pada halaman *register*
4	*Password*	*Text*	Kolom yang berupa *text* untuk memasukkan *password* pengguna baru pada saat *register*

6	*Register*	Tombol	Tombol untuk melakukan r*egister* atau untuk mengarahkan ke halaman *home*

3. Halaman Daftar Laporan

Rancangan antarmuka pada aplikasi ini terdiri dari beberapa menu, menu yang pertama yakni menu daftar laporan. Setelah pengguna melakukan *login* maka pengguna akan masuk pada halaman daftar laporan yang mana halaman ini merupakan halaman pertama yang muncul ketika selesai melakukan *login*. Halaman daftar laporan ini berisi halaman yang didalamnya terdapat laporan-laporan mengenai keluhan yang disampaikan oleh seluruh pengguna. Pada laporan yang dibagikan terdapat foto, nama, dan alamat dari pengguna yang melaporkan. Halaman daftar laporan ini akan menampilkan laporan-laporan yang terbaru dari laporan yang disampaikan masyarakat. Berikut merupakan perancangan antarmuka untuk halaman daftar laporan pada Gambar 5.10.



Gambar 5. 10 Perancangan Antarmuka Halaman Daftar Laporan

Berikut merupakan penjelasan dari Gambar 5.10 mengenai perancangan antarmuka untuk halaman daftar laporan pada m*obile app* dapat dilihat pada Tabel 5.15.

Tabel 5. 15 Penjelasan Perancangan Antarmuka Halaman Daftar Laporan

No	Nama Objek	Tipe	Keterangan
1	Navbar	Navbar	Navbar aplikasi Sigap untuk halaman daftar laporan

2	Nama Pengguna	*Text*	*Text* yang menunjukkan nama pengguna yang melaporkan pengaduan
3	Alamat	*Text*	*Text* yang menunjukkan alamat dari lokasi permasalahan yang dilaporkan
4	Foto Pengguna	Gambar	Foto dari pengguna yang melaporkan pengaduan
5	Foto Laporan	Gambar	Foto laporan yang dilaporkan oleh pengguna mengenai permasalahan yang terjadi
6	Deskripsi Laporan	*Text*	*Text* yang berisi deskripsi dari laporan yang dilaporkan secara d*etail*
7	Status Laporan	*Text*	*Text* yang berisi status dari laporan yang disampaikan
8	Jumlah *Suppor*t Keluhan	*Text*	*Text* yang berisi jumlah *support* keluhan pada suatu laporan
9	Menu Daftar Laporan	*Navigation Tabs*	Menu ini berisi seluruh daftar laporan yang pengguna lain laporkan terdiri dari nama pengguna, foto pengguna, alamat, foto keluhan, dan s*upport* keluhan
10	Menu *Trending*	*Navigation Tabs*	Menu ini berisi laporan-laporan yang memiliki jumlah s*upport* terbanyak yang terdiri dari sepuluh laporan dengan s*upport* terbanyak
11	Menu Tambah Laporan	*Navigation Tabs*	Menu digunakan untuk menambah laporan dengan mengambil foto terlebih dahulu
12	Menu Daftar Dukungan	*Navigation Tabs*	Menu ini menunjukkan daftar dari laporan-laporan yang pernah mendukung pada laporan lain
13	Menu *Profile*	*Navigation Tabs*	Menu *profile* berisi data diri dari pengguna dan daftar laporan yang dilaporkan oleh pengguna itu sendiri

4. Halaman *Trending* Laporan

Halaman *trending* laporan dapat diakses ketika pengguna memilih menu *trending* laporan dimana menu ini merupakan menu yang kedua pada perancangan antarmuka aplikasi ini. Halaman *trending* laporan dapat menampilkan laporan yang mendapatkan dukungan dari pelapor lain. Laporan-laporan ini akan tersusun sesuai dengan laporan dengan dukungan terbanyak. Halaman *trending* laporan dirancang untuk masyarakat jika masyarakat tersebut memiliki permasalahan yang sama dengan pelapor yang sudah menyampaikan laporan tersebut terlebih dahulu. Perancangan antarmuka halaman *trending* laporan digambarkan pada Gambar 5.11 yakni sebagai berikut.

Gambar 5. 11 Perancangan Antarmuka Halaman *Trending* Laporan

Halaman antarmuka untuk *trending* laporan ini sama seperti halaman antarmuka pada halaman *home* atau daftar laporan, jika terdapat *trending* maka halaman laporan seperti pada Gambar 5.11 namun jika tidak ada *trending* laporan dalam waktu tiga hari maka halaman antarmuka untuk halaman *trending* yakni kosong dan muncul pesan "Tidak Ada *Trending* Untuk 3 Hari Ini". Penjelasan berikut menjelaskan secara *detail* dari masing-masing bagian berdasarkan nama objek dan tipe yang membangun halaman antarmuka untuk halaman *trending* laporan pada *mobile app* sebagai berikut dapat dilihat pada Tabel 5.16.

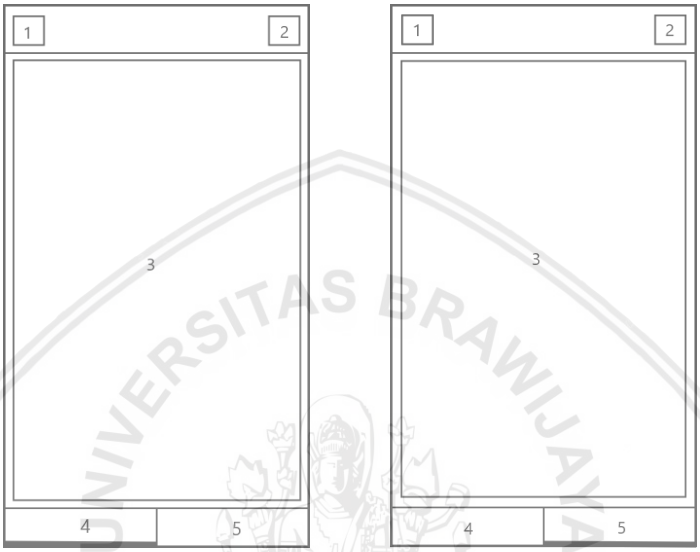
Tabel 5. 16 Penjelasan Perancangan Antarmuka Halaman *Trending* Laporan

No	Nama Objek	Tipe	Keterangan
1	Navbar	Navbar	Navbar aplikasi Sigap untuk halaman daftar laporan

2	Nama Pengguna	*Text*	*Text* yang menunjukkan nama pengguna yang melaporkan pengaduan
3	Alamat	*Text*	*Text* yang menunjukkan alamat dari lokasi permasalahan yang dilaporkan
4	Foto Pengguna	Gambar	Foto dari pengguna yang melaporkan pengaduan
5	Foto Laporan	Gambar	Foto laporan yang dilaporkan oleh pengguna mengenai permasalahan yang terjadi
6	Deskripsi Laporan	*Text*	*Text* yang berisi deskripsi dari laporan yang dilaporkan secara d*etail*
7	Status Laporan	*Text*	*Text* yang berisi status dari laporan yang disampaikan
8	Jumlah *Suppor*t Keluhan	*Text*	*Text* yang berisi jumlah *support* keluhan pada suatu laporan
9	Menu Daftar Laporan	*Navigation Tabs*	Menu ini berisi seluruh daftar laporan yang pengguna lain laporkan terdiri dari nama pengguna, foto pengguna, alamat, foto keluhan, dan s*upport* keluhan
10	Menu *Trending*	*Navigation Tabs*	Menu ini berisi laporan-laporan yang memiliki jumlah s*upport* terbanyak yang terdiri dari sepuluh laporan dengan s*upport* terbanyak
11	Menu Tambah Laporan	*Navigation Tabs*	Menu digunakan untuk menambah laporan dengan mengambil foto terlebih dahulu
12	Menu Daftar Dukungan	*Navigation Tabs*	Menu ini menunjukkan daftar dari laporan-laporan yang pernah mendukung pada laporan lain
13	Menu *Profile*	*Navigation Tabs*	Menu *profile* berisi data diri dari pengguna dan daftar laporan yang dilaporkan oleh pengguna itu sendiri

5. Halaman Menambah Laporan

Menu yang ketiga pada perancangan antarmuka ini ialah menu menambah laporan. Untuk membuat laporan baru pelapor terlebih dahulu mengambil gambar untuk dapat melaporkan keluhan. Untuk mengambil gambar pada aplikasi ini dapat mengakses kamera dari *smartphone* yang digunakan atau dari *gallery smartphone*. Pada halaman laporan keluhan ini digunakan untuk bukti terhadap laporan yang disampaikan kepada pihak Dinas Perhubungan Kota Malang. Gambar 5.12 menggambarkan halaman laporan keluhan untuk mengunggah gambar.



Gambar 5. 12 Perancangan Antarmuka Halaman untuk *Gallery* (kiri) dan Perancangan Antarmuka Halaman *Camera* (kanan)

Tabel 5.17 menunjukkan penjelasan perancangan antarmuka dari halaman untuk *gallery* dan antarmuka halaman *camera* yang mengacu pada Gambar 5.12.

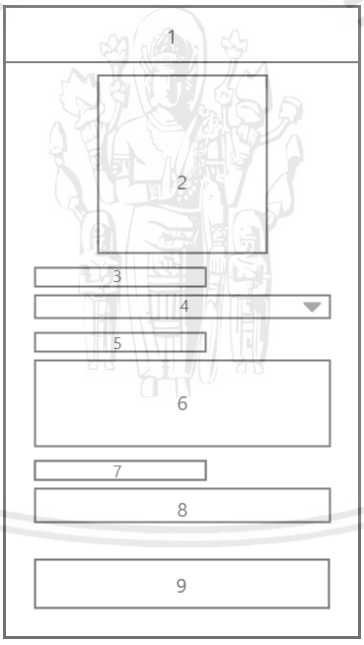
Tabel 5. 17 Penjelasan Perancangan Antarmuka Halaman untuk *Gallery* dan Halaman *Camera*

No	Nama Objek	Tipe	Keterangan
1	*Back*	Gambar	*Icon* yang digunakan jika pengguna ingin kembali ke halaman sebelumnya
2	*Next*	Gambar	*Icon* yang digunakan untuk mengarahkan ke halaman selanjutnya yakni halaman upload
3	*Gallery*/*Camera*	Gambar	Gambar yang digunakan untuk mengambil foto laporan dari g*allery smartphone* maupun dari c*amera* langsung
4	*Gallery*	Menu Tab	Menu yang menunjukkan jika pengguna ingin mengambil foto

			dari g*allery* s*martphone* yang sudah tersimpan sebelumnya
5	*Camera*	Menu Tab	Menu yang menunjukkan jika pengguna ingin mengambil foto dari c*amera* s*martphone* langsung

6. Halaman *Upload* Laporan

Ketika pelapor sudah mengambil gambar baik dari *gallery* maupun dari *camera smartphone* maka akan masuk pada halaman *upload* laporan ini. Pada halaman *upload* laporan terdapat foto yang sudah diambil lalu pelapor dapat memilih kategori keluhan berdasarkan masalah yang dilaporkan, terdapat tombol *map* untuk memilih lokasi kejadian lalu akan diarahkan pada *map* berdasarkan *latitude* dan *longitude* yang sudah ditentukan dan juga dapat memiliki lokasi berdasarkan posisi pengguna saat itu dan selanjutnya memasukkan deskripsi setelah itu pelapor dapat menekan tombol *upload* yang terletak dibawah halaman *upload* laporan. Gambar 5.13 menggambarkan perancangan antarmuka halaman *upload* laporan.



Gambar 5. 13 Perancangan Antarmuka Halaman *Upload* Laporan

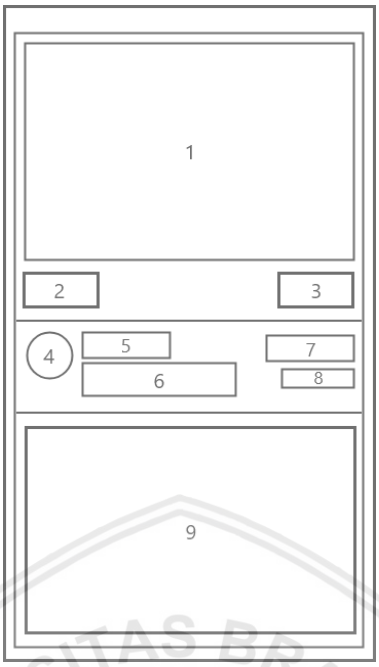
Gambar 5.13 yakni mengenai perancangan antarmuka untuk halaman upload laporan pada *mobile app*, penjelasan dibawah dapat menjelaskan struktur yang membangun aplikasi untuk halaman upload laporan yang terdiri dari nama objek, jenis struktur yang digunakan serta keterangan dari jenis dan nama objeknya. Penjelasan dari perancangan antarmuka halaman *upload* laporan terdapat pada Tabel 5.18, yaitu sebagai berikut.

Tabel 5. 18 Penjelasan Perancangan Antarmuka Halaman *Upload* Laporan

No	Nama Objek	Tipe	Keterangan
1	Navbar	Navbar	Navbar aplikasi Sigap untuk halaman upload laporan
2	Foto Laporan	Gambar	Foto laporan yang akan dilaporkan dimana telah diambil dari g*allery* maupun c*amera* pada halaman sebelumnya
3	Kategori Keluhan	*Text*	*Text* yang menunjukkan keterangan kategori keluhan
4	*Dropdown* Kategori	*Dropdown*	Fitur yang digunakan untuk memilih kategori dari keluhan yang akan dilaporkan berdasarkan tugas dari Dinas Perhubungan
5	Deskripsi	*Text*	*Text* yang berisi keterangan dari deskripsi
6	Deskripsi Laporan	*Text*	*Text* yang digunakan untuk memasukkan deskripsi laporan yang akan disampaikan
7	Lokasi	*Text*	*Text* yang berisi keterangan dari lokasi
8	Lokasi Keluhan	*Text*	Kolom yang berisi *text* untuk menampilkan lokasi dari keluhan yang akan dilaporkan
9	*Upload* Laporan	Tombol	Tombol yang digunakan untuk mengarahkan laporan masuk ke *database* dan menu daftar laporan

7. Halaman *Detail* Laporan

Halaman *detail* laporan dapat dilihat ketika pengguna ingin mengetahui secara *detail* dari laporan dengan menekan laporan yang diinginkan. Halaman ini akan menampilkan laporan secara *detail* seperti foto beserta deskripsi lengkap dari laporan, dan juga lokasi dari laporan tersebut. Pada halaman *detail* laporan ini dapat menampilkan *map* dari lokasi permasalahan yang dilaporkan yang mana terdapat *marker* pada tepat pada lokasi permasalahan. Pada halaman *detail* laporan pelapor dapat melakukan *support* keluhan terhadap laporan yang dilaporkan oleh pelapor lain dengan cara menekan tombol *love*. Terdapat pula jumlah *support* keluhan tersebut. Halaman *detail* laporan dapa dilihat pada Gambar 5.14 seperti berikut.



Gambar 5. 14 Perancangan Antarmuka Halaman *Detail* Laporan

Berikut merupakan penjelasan dari Gambar 5.14 mengenai perancangan antarmuka untuk halaman *detail* laporan pada *mobile app*, dapat dilihat pada Tabel 5.19.

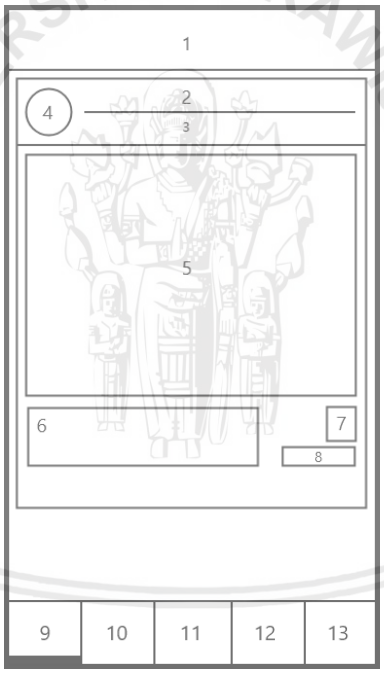
Tabel 5. 19 Penjelasan Perancangan Antarmuka Halaman *Detail* Laporan

No	Nama Objek	Tipe	Keterangan
1	Foto Laporan	Gambar	Foto laporan yang dilaporkan oleh pengguna mengenai permasalahan yang terjadi dapat dilihat secara lebih jelas dan *detail*
2	Kategori Keluhan	*Text*	*Text* yang menunjukkan kategori dari foto keluhan yang dilaporkan
3	*Support* Keluhan	Gambar	*Icon* yang digunakan untuk melakukan *support* atau dukung laporan pengguna lain dengan cara mengklik *icon* tersebut
4	Foto Pengguna	Gambar	Foto dari pengguna yang melaporkan pengaduan
5	Nama Pengguna	*Text*	*Text* yang menunjukkan nama pengguna yang melaporkan pengaduan
6	Deskripsi Laporan	*Text*	*Text* yang berisi deskripsi dari laporan yang dilaporkan secara *detail*

7	Status Laporan	*Text*	*Text* yang berisi status dari laporan yang disampaikan
8	Tanggal Posting	*Date*	*Text* yang berisi tanggal posting dari laporan
9	*Map* Laporan	*Map*	*Map* ini menunjukkan lokasi pengaduan yang dilaporkan oleh pengguna

8. Halaman Daftar Dukungan

Halaman daftar dukungan merupakan halaman yang digunakan untuk melihat daftar dari laporan-laporan pribadi yang pernah dilakukan untuk mendukung sebuah laporan yang dilaporkan oleh pengguna lain. Tampilan halaman daftar dukungan ini sama seperti halaman daftar laporan atau menu *home*. Namun perbedaannya halaman daftar dukungan ini hanya berisi laporan-laporan yang pernah didukung. Gambar 5.15 merupakan halaman daftar dukungan.



Gambar 5. 15 Perancangan Antarmuka Halaman Daftar Dukungan

Berikut merupakan penjelasan dari Gambar 5.15 mengenai perancangan antarmuka untuk halaman daftar dukungan pada *mobile app*, dapat dilihat pada Tabel 5.20.

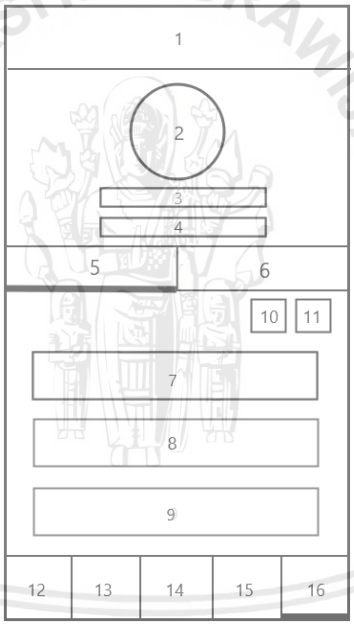
Tabel 5. 20 Penjelasan Perancangan Antarmuka Halaman Daftar Dukungan

No	Nama Objek	Tipe	Keterangan
1	Navbar	Navbar	Navbar aplikasi Sigap untuk halaman daftar laporan

2	Nama Pengguna	*Text*	*Text* yang menunjukkan nama pengguna yang melaporkan pengaduan
3	Alamat	*Text*	*Text* yang menunjukkan alamat dari lokasi permasalahan yang dilaporkan
4	Foto Pengguna	Gambar	Foto dari pengguna yang melaporkan pengaduan
5	Foto Laporan	Gambar	Foto laporan yang dilaporkan oleh pengguna mengenai permasalahan yang terjadi
6	Deskripsi Laporan	*Text*	*Text* yang berisi deskripsi dari laporan yang dilaporkan secara d*etail*
7	Status Laporan	*Text*	*Text* yang berisi status dari laporan yang disampaikan
8	Jumlah *Suppor*t Keluhan	*Text*	*Text* yang berisi jumlah *support* keluhan pada suatu laporan
9	Menu Daftar Laporan	*Navigation Tabs*	Menu ini berisi seluruh daftar laporan yang pengguna lain laporkan terdiri dari nama pengguna, foto pengguna, alamat, foto keluhan, dan s*upport* keluhan
10	Menu *Trending*	*Navigation Tabs*	Menu ini berisi laporan-laporan yang memiliki jumlah s*upport* terbanyak yang terdiri dari sepuluh laporan dengan s*upport* terbanyak
11	Menu Tambah Laporan	*Navigation Tabs*	Menu digunakan untuk menambah laporan dengan mengambil foto terlebih dahulu
12	Menu Daftar Dukungan	*Navigation Tabs*	Menu ini menunjukkan daftar dari laporan-laporan yang pernah mendukung pada laporan lain
13	Menu *Profile*	*Navigation Tabs*	Menu *profile* berisi data diri dari pengguna dan daftar laporan yang dilaporkan oleh pengguna itu sendiri

9. Halaman *Profile* Pelapor

Gambar 5.16 dan Gambar 5.17 merupakan gambar dari halaman *profile* pelapor. Menu ini merupakan menu kelima pada aplikasi pengaduan masyarakat. Pada halaman *profile* pelapor terdapat dua tab yang digunakan dimana tab yang pertama menampilkan data diri dari pelapor tersebut dan tab yang kedua menampilkan seluruh laporan yang pernah diunggah oleh pengguna itu sendiri. Pada menu tab data diri pelapor dapat mengubah data dirinya dan pada menu tab laporan pelapor juga dapat mengubah laporannya sebelum ditindaklanjuti oleh admin. Pada halaman berikut ini merupakan halaman data diri dari pengguna akun dimana pada halaman data diri ini terdapat informasi-informasi dari pengguna seperti nama lengkap, *email*, *password*, tanggal lahir, dan juga alamat. Pada halaman data diri ini juga terdapat dua menu yang memiliki fungsi untuk *edit* biodata dan juga *logout*. Dan tab yang kedua yakni halaman laporan pribadi yang berisi seluruh daftar laporan pribadi yang pernah dilaporkan oleh pelapor tersebut. Untuk halaman data diri dapat ditunjukkan pada Gambar 5.16 dan perancangan antarmuka halaman laporan pribadi dapat dilihat pada Gambar 5.17.



Gambar 5. 16 Perancangan Antarmuka Halaman Data Diri

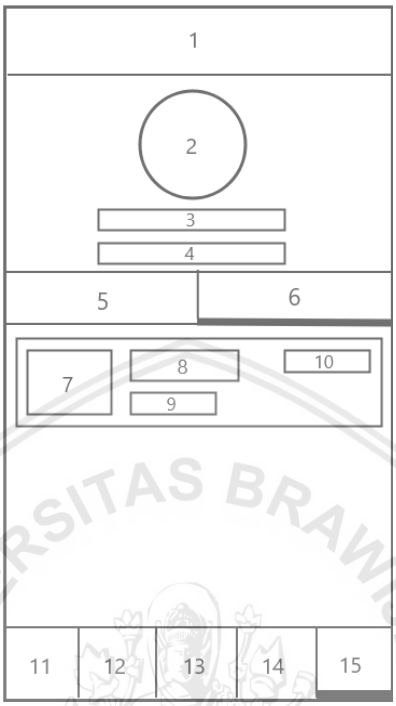
Tabel 5.21 menunjukkan penjelasan perancangan antarmuka dari halaman data diri yang mengacu pada Gambar 5.16.

Tabel 5. 21 Penjelasan Perancangan Antarmuka Halaman Data Diri

No	Nama Objek	Tipe	Keterangan
1	Navbar	Navbar	Navbar aplikasi Sigap untuk halaman *profile* pengguna
2	Foto Pengguna	Gambar	Foto dari pengguna yang memiliki yang sudah melakukan *login*

3	Nama Pengguna	*Text*	*Text* menunjukkan nama pengguna yang sudah melakukan *login*
4	Alamat	*Text*	*Text* yang menunjukkan alamat dari pengguna
5	Data Diri	Menu Tab	Menu yang digunakan untuk melihat data diri dari pengguna yang melakukan *login*
6	Laporan	Menu Tab	Menu yang digunakan untuk melihat laporan pribadi dari pengguna yang melakukan *login*
7	Nama	*Text*	*Text* yang menunjukkan nama lengkap dari pengguna yang *login*
8	*Email*	*Text*	*Text* yang menunjukkan *email* dari pengguna
9	*Password*	*Text*	*Text* yang berisi *password* dari pengguna tersebut
10	*Edit*	Gambar	Fitur untuk mengubah data diri dari pengguna yang mengarahkan ke halaman *edit profile*
11	*Logout*	Gambar	Fitur untuk melakukan *logout* atau keluar dari sistem
12	Menu Daftar Laporan	*Navigation Tabs*	Menu ini berisi seluruh daftar laporan yang pengguna lain laporkan terdiri dari nama pengguna, foto pengguna, alamat, foto keluhan, dan s*upport* keluhan
13	Menu *Trending*	*Navigation Tabs*	Menu ini berisi laporan-laporan yang memiliki jumlah s*upport* terbanyak yang terdiri dari sepuluh laporan dengan s*upport* terbanyak
14	Menu Tambah Laporan	*Navigation Tabs*	Menu digunakan untuk menambah laporan dengan mengambil foto terlebih dahulu
15	Menu Daftar Dukungan	*Navigation Tabs*	Menu ini menunjukkan daftar dari laporan-laporan yang pernah mendukung pada laporan lain
16	Menu *Profile*	*Navigation Tabs*	Menu *profile* berisi data diri dari pengguna dan daftar laporan yang

			dilaporkan oleh pengguna itu sendiri



Gambar 5. 17 Perancangan Antarmukan Halaman Laporan Pribadi

Berikut merupakan penjelasan dari Gambar 5.17 mengenai perancangan antarmuka untuk halaman laporan pribadi pada *mobile app*, dapat dilihat pada Tabel 5.22.

Tabel 5. 22 Penjelasan Perancangan Antarmukan Halaman Laporan Pribadi

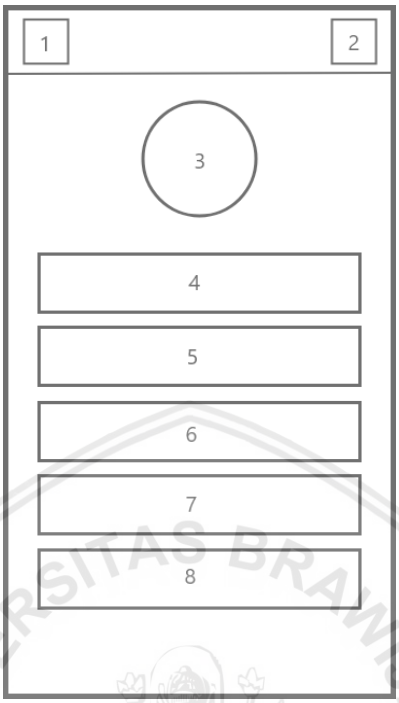
No	Nama Objek	Tipe	Keterangan
1	Navbar	Navbar	Navbar aplikasi Sigap untuk halaman *profile* pengguna
2	Foto Pengguna	Gambar	Foto dari pengguna yang memiliki yang sudah melakukan *login*
3	Nama Pengguna	*Text*	*Text* menunjukkan nama pengguna yang sudah melakukan *login*
4	Alamat	*Text*	*Text* yang menunjukkan alamat dari pengguna
5	Data Diri	Menu Tab	Menu yang digunakan untuk melihat data diri dari pengguna yang melakukan *login*

6	Laporan	Menu Tab	Menu yang digunakan untuk melihat laporan pribadi dari pengguna yang melakukan *login*
7	Foto Laporan	Gambar	Foto laporan yang dilaporkan oleh pengguna mengenai permasalahan yang terjadi
8	Deskripsi Laporan	*Text*	*Text* yang berisi deskripsi dari laporan yang dilaporkan secara *detail*
9	Jumlah *Support* Keluhan	*Text*	*Text* yang berisi jumlah *support* keluhan pada suatu laporan
10	Tanggal Posting	*Date*	*Text* yang berisi tanggal posting dari laporan
11	Menu Daftar Laporan	*Navigation Tabs*	Menu ini berisi seluruh daftar laporan yang pengguna lain laporkan terdiri dari nama pengguna, foto pengguna, alamat, foto keluhan, dan s*upport* keluhan
12	Menu *Trending*	*Navigation Tabs*	Menu ini berisi laporan-laporan yang memiliki jumlah s*upport* terbanyak yang terdiri dari sepuluh laporan dengan s*upport* terbanyak
13	Menu Tambah Laporan	*Navigation Tabs*	Menu digunakan untuk menambah laporan dengan mengambil foto terlebih dahulu
14	Menu Daftar Dukungan	*Navigation Tabs*	Menu ini menunjukkan daftar dari laporan-laporan yang pernah mendukung pada laporan lain
15	Menu *Profile*	*Navigation Tabs*	Menu *profile* berisi data diri dari pengguna dan daftar laporan yang dilaporkan oleh pengguna itu sendiri

10. Halaman Mengubah *Profile*

Halaman mengubah *profile* merupakan halaman yang dilakukan untuk mengubah *profile* pengguna dan sudah melakukan proses *login* pada saat awal membuka aplikasi *mobile*. Halaman mengubah *profile* dilakukan setelah menekan *tab profile* dan menekan *icon edit* pada halaman *tab profile*. Pada halaman mengubah *profile* ini pengguna dapat mengubah nama, *password*, alamat, dan juga tanggal lahir pada kolom yang terdapat pada halaman mengubah *profile*.

Perancangan halaman mengubah *profile* terdapat pada Gambar 5.18 seperti berikut.



Gambar 5. 18 Perancangan Antarmuka Halaman Mengubah *Profile*

Penjelasan dari Gambar 5.18 mengani perancangan antarmuka halaman mengubah *profile* pada aplikasi *mobile* dapat dijelaskan pada Tabel 5.23 yang berisi penjelasan dari komponen-komponen yang menyusun halaman mengubah *profile.*

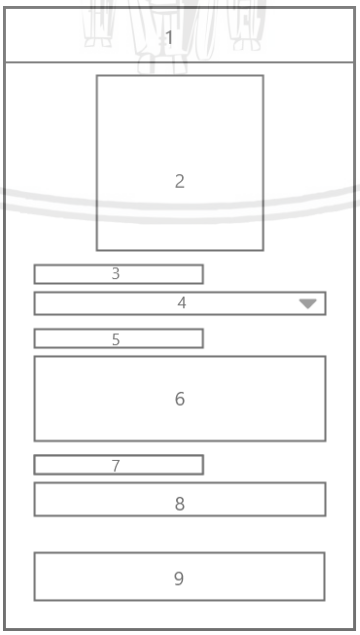
Tabel 5. 23 Penjelasan Perancangan Antarmukan Halaman Mengubah *Profile*

No	Nama Objek	Tipe	Keterangan
1	*Back*	Gambar	*Icon* yang digunakan jika pengguna ingin kembali ke halaman sebelumnya
2	Selesai	Gambar	*Icon* ceklis atau selesai yang digunakan jika mengubah profile sudah selesai dilakukan
3	Foto Pengguna	Gambar	Foto dari pengguna yang memiliki yang sudah melakukan *login*
4	Nama Pengguna	*Text*	Kolom yang berisi *text* nama pengguna, jika ingin mengubah nama pengguna dapat mengubah pada kolom ini

5	*Email*	*Text*	Kolom yang berisi *text email*, jika ingin mengubah *email* dapat mengubah pada kolom *email* ini
6	*Password*	*Text*	Jika ingin mengubah *password* akun aplikasi *mobile* pengaduan masyarakat dapat dilakukan dengan cara mengubah *password* pada kolom ini
7	Alamat	*Text*	Kolom yang berisi *text* untuk alamat, jika ingin mengubah alamat dapat mengubah pada kolom ini
8	Tanggal Lahir	*Date*	Jika ingin mengubah tanggal lahir dapat mengubah tanggal lahir pada kolom ini

11. Halaman Mengubah Laporan

Halaman mengubah laporan dapat diakses dengan menekan halaman tab laporan terlebih dahulu pada halaman *profile* pengguna. Halaman mengubah laporan digunakan untuk mengubah laporan pribadi yang sudah dilaporkan namun belum ditindaklanjuti. Untuk mengakses halaman mengubah laporan pribadi ini pengguna harus memilih laporan diubah setelah itu akan masuk pada halaman mengubah laporan. Laporan yang dapat diubah yakni gambar, kategori keluhan, deskripsi, dan juga lokasi kejadian. Perancangan antarmuka halaman mengubah laporan pribadi dapat dilihat pada Gambar 5.19 seperti berikut.



Gambar 5. 19 Perancangan Antarmuka Halaman Mengubah Laporan

Gambar 5.19 yakni mengenai perancangan antarmuka untuk halaman mengubah laporan pribadi pada aplikasi *mobile*, penjelasan dari perancangan antarmuka halaman mengubah laporan terdapat pada Tabel 5.24, yaitu sebagai berikut.

Tabel 5. 24 Penjelasan Perancangan Antarmuka Halaman Mengubah Laporan

No	Nama Objek	Tipe	Keterangan
1	Selesai	Gambar	*Icon* ceklis atau selesai yang digunakan jika mengubah laporan pribadi sudah selesai dilakukan
2	Foto Laporan	Gambar	Foto laporan yang akan dilaporkan dimana telah diambil dari g*allery* maupun c*amera* pada halaman sebelumnya
3	Kategori Keluhan	*Text*	*Text* yang menunjukkan keterangan kategori keluhan
4	*Dropdown Kategori*	*Dropdown*	Fitur yang digunakan untuk memilih kategori dari keluhan yang akan dilaporkan berdasarkan tugas dari Dinas Perhubungan
5	Deskripsi	*Text*	*Text* yang berisi keterangan dari deskripsi
6	Deskripsi Laporan	*Text*	*Text* yang digunakan untuk memasukkan deskripsi laporan yang akan disampaikan
7	Lokasi	*Text*	*Text* yang berisi keterangan dari lokasi
8	Lokasi Keluhan	*Text*	Kolom yang berisi *text* untuk menampilkan lokasi dari keluhan yang akan dilaporkan
9	*Upload* Laporan	Tombol	Tombol yang digunakan untuk mengarahkan laporan masuk ke *database* dan menu daftar laporan

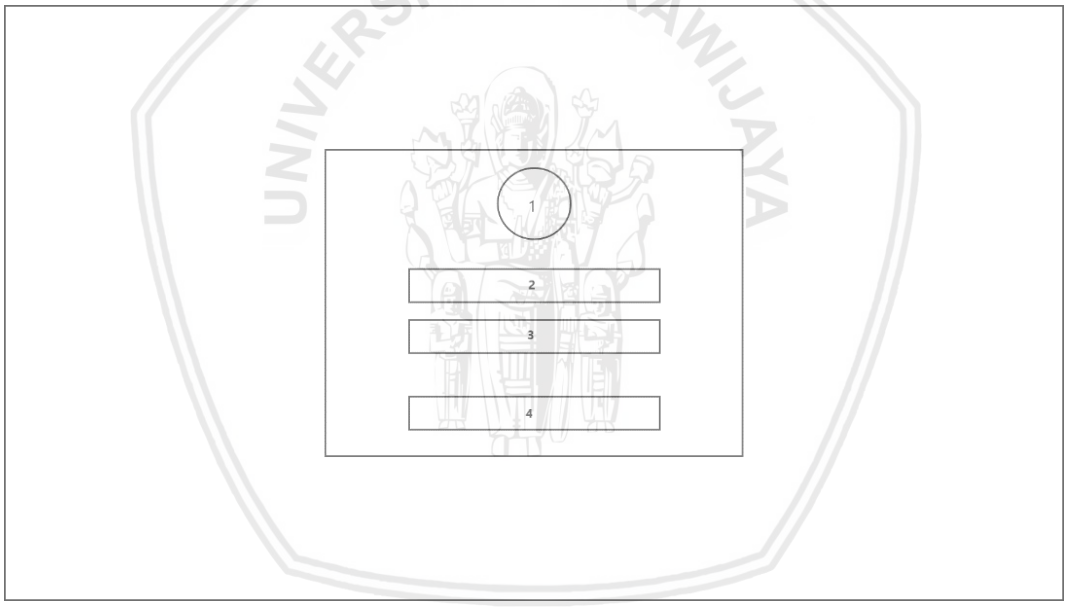
5.1.4.2 Perancangan Antarmuka *Web* Pengaduan Masyarakat

Perancangan antarmuka yang kedua ialah perancangan antarmuka untuk *web* yang nantinya digunakan untuk admin atau pihak Dinas Perhubungan Kota Malang. Halaman web ini terdiri dari halaman halaman *login*, halaman *dashboard*, dan halaman laporan dimana halaman laporan ini terdiri dari empat submenu seperti menu *trending* laporan, laporan *waiting*, laporan *onprogress*, laporan *done* serta laporan *block*. Pada perancangan antarmuka untuk web ini hanya digunakan untuk mengelola laporan yang masuk yang dilaporkan oleh masyarakat Kota

Malang, dapat melihat lokasi permasalahan secara akurat, mengelola status laporan mulai dari *waiting, onprogress, done*, dan *block*. Pada halaman web yang digunakan oleh admin ini dapat menampilkan *map* dari lokasi kejadian dan terdapat kategori keluhan pada map tersebut. Perancangan ini digunakan untuk memudahkan dalam proses implementasi sistem. Perancangan antarmuka web dapat dilihat pada Gambar 5.20 sampai dengan Gambar 5.25.

1. Halaman *Login*

Seperti pada perancangan antarmuka untuk *mobile* pada perancangan antarmuka web terdapat halaman *login* seperti pada Gambar 5.20. Gambar tersebut merupakan halaman *login* untuk web yang mana halaman awal yang muncul ketika admin mengakses web pengaduan masyarakat. Pengguna harus melakukan *login* terlebih dahulu pada halaman ini untuk dapat mengakses web tersebut. Halaman *login* terdapat *form* untuk memasukkan *email* dan *password* dan terdapat tombol untuk dapat *login* pada web. Pengguna yang akan mengakses dapat memasukkan *email* dan *password* yang dimilik untuk dapat melakukan aktifitas pada web.



Gambar 5. 20 Perancangan Antarmuka Halaman *Login*

Berikut merupakan penjelasan dari Gambar 5.20 mengenai perancangan antarmuka untuk halaman *login* pada web, penjelasan berikut menjelaskan secara rinci dari komponen-komponen yang membangun perancangan antarmuka untuk halaman *login*, perancangan tersebut dapat ditunjukkan pada Tabel 5.25.

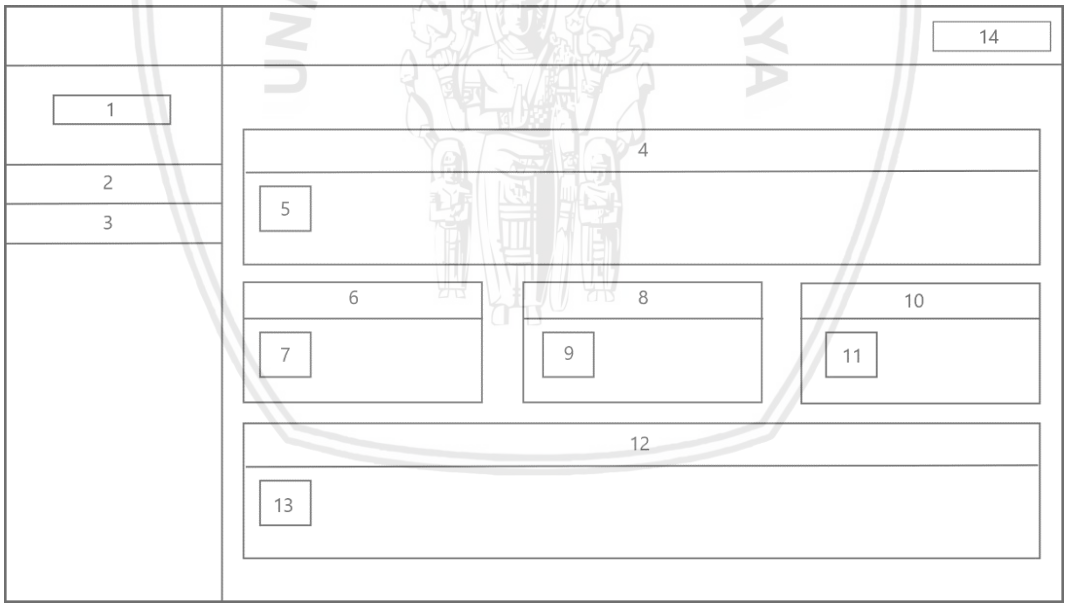
Tabel 5. 25 Penjelasan Perancangan Antarmuka Halaman *Login*

No	Nama Objek	Tipe	Keterangan
1	Logo Aplikasi	Gambar	Logo aplikasi yakni logo dari Dinas Perhubungan Kota Malang

2	*User*name	*Text*	Kolom yang berupa *text* untuk memasukkan *uername* pengguna pada saat l*ogin*
3	*Password*	*Text*	Kolom yang berupa *text* untuk memasukkan *password* pengguna pada saat *login*
4	*Login*	Tombol	Tombol untuk *login* atau untuk mengarahkan ke halaman *home*

2. Halaman *Dashboard*

Halaman *dashboard* merupakan halaman pertama yang muncul setelah *administrator* melakukan login pada sistem. Halaman *dashboard* ini menampilkan total jumlah laporan yang masuk pada sistem. Dimana laporan-laporan yang dimaksudkan tersebut ialah menampilkan jumlah *trending* yang masuk, seluruh laporan yang masuk, jumlah laporan yang berstatus *waiting, onprogress*, dan juga *done* serta menampilkan jumlah *block* terhadap laporan yang masuk pada sistem. Perancangan antarmuka halaman *dashboard* pada website ini dapat dilihat pada Gambar 5.21.



Gambar 5. 21 Perancangan Antarmuka Halaman *Dashboard*

Penjelasan antarmuka untuk halaman *dashboard* tersebut dapat dilihat pada Tabel 5.26 seperti berikut.

Tabel 5. 26 Penjelasan Perancangan Antarmuka Halaman *Home*

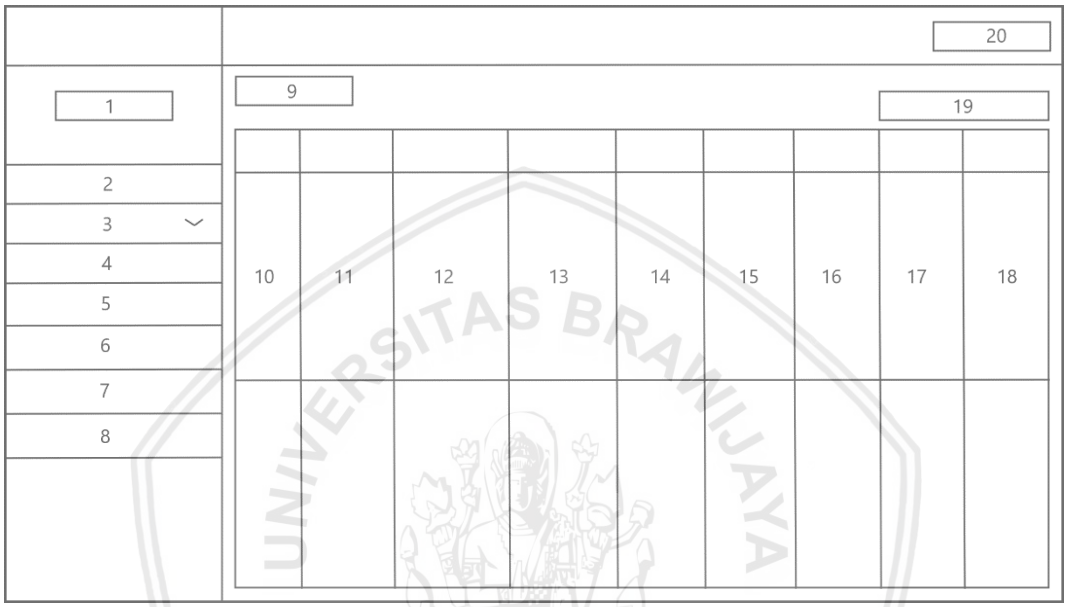
No	Nama Objek	Tipe	Keterangan
1	Nama Admin	*Text*	*Text* yang menunjukkan nama dari admin yang melakukan *login*

2	Menu *Dashboard*	Menu Navbar	Nama menu pada website dengan nama menu *dashboard*
3	Menu Laporan	Menu Navbar	Nama menu pada website dengan nama menu laporan yang berisi seluruh laporan yang masuk pada sistem
4	*Trending*	*Text*	Judul menu yang ada pada menu *dashboard* yakni *trending*
5	Jumlah *Trending*	*Text*	*Text* yang menampilkan jumlah total *trending* laporan yang masuk
6	Laporan	*Text*	Judul menu yang ada pada menu *dashboard* yakni laporan
7	Jumlah Laporan	*Text*	*Text* yang menampilkan jumlah total laporan yang masuk pada sistem
8	Status *OnProgress*	*Text*	Judul status menu yang ada pada menu *dashboard* yakni *onprogress*
9	Jumlah *OnProgress*	*Text*	*Text* yang menampilkan jumlah total laporan yang berstatus *OnProgress* yang masuk pada sistem
10	Status *Done*	*Text*	Judul status menu yang ada pada menu *dashboard* yakni *Done*
11	Jumlah *Done*	*Text*	*Text* yang menampilkan jumlah total laporan yang berstatus *Done* atau laporan selesai ditindaklanjuti
12	Status *Block*	*Text*	Judul status menu yang ada pada menu *dashboard* yakni *Block*
13	Jumlah *Block*	*Text*	*Text* yang menampilkan jumlah total laporan yang berstatus *Block* atau laporan yang dilaporkan tidak sesuai dengan ketentuan
14	Logout	Gambar	*Icon* yang digunakan untuk melakukan *logout* atau keluar dari sistem

3. Halaman *Trending* Laporan

Halaman *trending* laporan merupakan halaman kedua setelah halaman *dashboard*. Pada halaman laporan ini terdiri dari beberapa menu lagi yakni menu

Trending Laporan, Laporan *Waiting*, Laporan *OnProgress*, Laporan *Done*, dan juga menu *Block*. Halaman *trending* laporan untuk *administrator* ini digunakan untuk menampilkan laporan-laporan yang masuk yang menjadi *trending*. Laporan yang menjadi *trending* tersebut berdasarkan laporan yang memiliki jumlah *support* terbanyak. Pada halaman *trending* laporan terdapat daftar laporan mulai dari deskripsi, foto laporan, alamat kejadian, tanggal posting, kategori keluhan, status dari laporan, nama pelapor, dan juga jumlah dukung yang didapatkan. Gambar 5.22 menunjukkan perancangan antarmuka untuk halaman *trending* laporan.



Gambar 5. 22 Perancangan Antarmuka Halaman *Trending* Laporan

Tabel 5.27 menunjukkan penjelasan perancangan antarmuka dari halaman *trending* laporan untuk sisi *administrator* yang mengacu pada Gambar 5.22.

Tabel 5. 27 Penjelasan Perancangan Antarmuka Halaman *Trending* Laporan

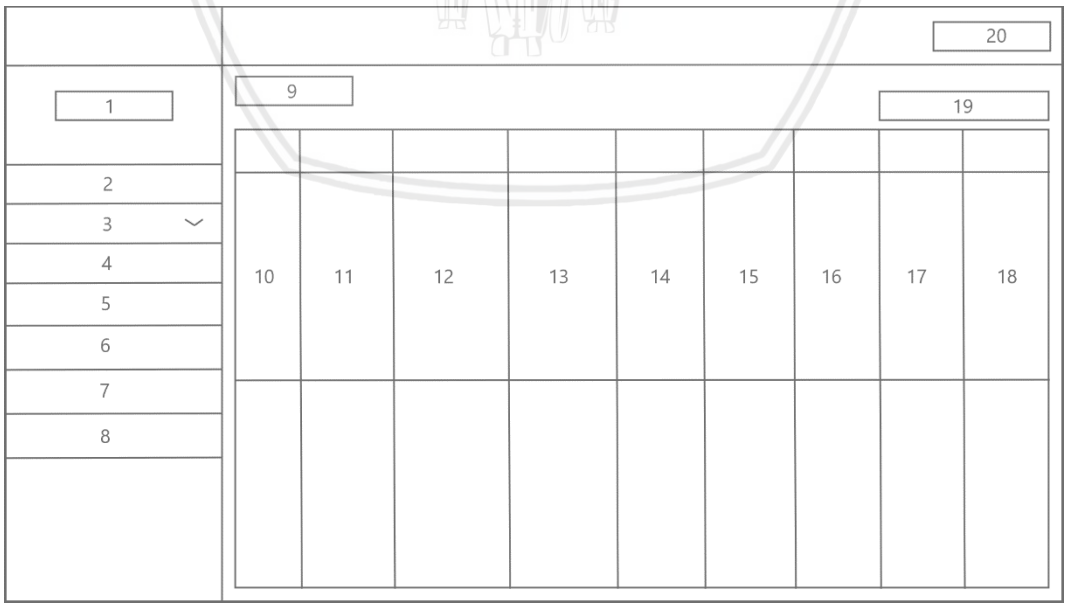
No	Nama Objek	Tipe	Keterangan
1	Nama Admin	*Text*	*Text* yang menunjukkan nama dari admin yang melakukan *login*
2	Menu *Dashboard*	Menu Navbar	Nama menu pada website dengan nama menu *dashboard*
3	Menu Laporan	Menu Navbar	Nama menu pada website dengan nama menu laporan yang berisi seluruh laporan yang masuk pada sistem
4	Menu *Trending* Laporan	Menu Navbar	Nama menu pada website dengan nama menu *trending* laporan yang berisi seluruh laporan yang menjadi *trending*

5	Menu Status *Waiting*	Menu Navbar	Nama menu dengan nama menu status *waiting* yang berisi seluruh laporan yang masuk yang berstatus *waiting*
6	Menu Status *OnProgress*	Menu Navbar	Nama menu dengan nama menu status *onprogress* yang berisi seluruh laporan yang masuk yang berstatus *onprogress* atau sedang diperbaiki
7	Menu Status *Done*	Menu Navbar	Nama menu dengan nama menu status *done* yang berisi seluruh laporan yang masuk yang berstatus *done* atau selesai diperbaiki
8	Menu Status *Bloack*	Menu Navbar	Nama menu dengan nama menu status *block* yang berisi laporan-laporan yang masuk namun foto laporan tidak sesuai dengan permasalahan
9	Status Halaman	*Text*	*Text* yang menunjukkan status halaman dari menu yang sedang dijalankan
10	Deskripsi	*Text*	*Text* yang menunjukkan deskripsi secara *detail* dari laporan yang masuk dari seorang pelapor
11	Foto Laporan	Gambar	Gambar yang menunjukkan foto laporan yang masuk mengenai situasi dan kondisi lokasi
12	Alamat	*Text*	*Text* yang menunjukkan alamat atau lokasi dari tempat kejadian pelapor menyampaikan permasalahan tersebut
13	Tanggal Post	*Date*	*Text* yang menunjukkan tanggal pelaporan
14	Kategori Keluhan	*Text*	*Text* yang menunjukkan kategori keluhan yang dipilih oleh pelapor
15	Status	*Text*	*Text* yang menunjukkan status laporan dari pengguna yang melaporkan yang di kelola oleh admin

16	Nama Pelapor	*Text*	*Text* yang menunjukkan nama pelapor yang melaporkan keluhan
17	Dukung Laporan	*Text*	*Text* yang menampilkan jumlah dukungan yang didapatkan oleh laporan
18	Ganti Status Laporan	*Botton*	*Botton* yang digunakan untuk mengganti status laporan
19	*Search* Laporan	Kolom	Kolom yang digunakan untuk melakukan pencarian terhadap *trending* laporan
20	*Logout*	Gambar	*Icon* yang digunakan untuk melakukan *logout* atau keluar dari sistem

4. Halaman Laporan Status *Waiting*

Halaman laporan dengan status *waiting* merupakan halaman yang menampilkan secara keseluruhan laporan-laporan yang memiliki status *waiting*. Sama halnya seperti halaman *trending* laporan, halaman laporan status *waiting* ini memiliki halaman antarmuka sama seperti halaman *trending* laporan tersebut. Pada halaman laporan status *waiting* ini terdapat daftar laporan berstatus *waiting* mulai dari deskripsi, foto laporan, alamat kejadian, tanggal posting, kategori keluhan, status dari laporan, nama pelapor, dan juga jumlah dukung yang didapatkan. Perancangan antarmuka halaman laporan status *waiting* dapat dilihat pada Gambar 5.23.



Gambar 5. 23 Perancangan Antarmuka Halaman Laporan Status *Waiting*

Tabel 5.28 menunjukkan penjelasan perancangan antarmuka dari halaman untuk laporan yang memiliki status *waiting* untuk sisi *administrator* yang mengacu pada Gambar 5.23.

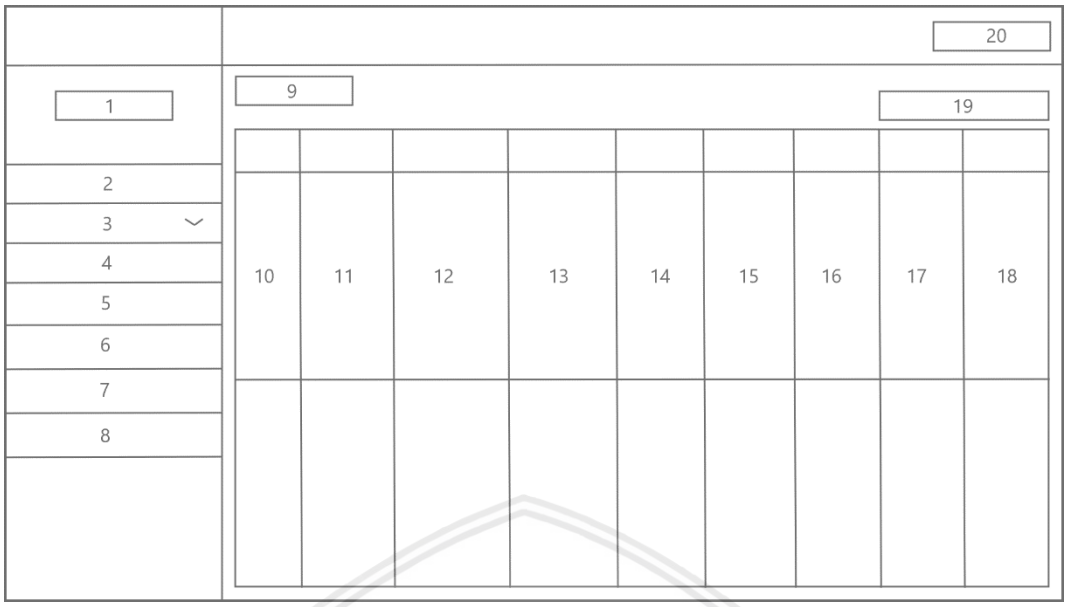
Tabel 5. 28 Penjelasan Perancangan Antarmuka Halaman Laporan Status *Waiting*

No	Nama Objek	Tipe	Keterangan
1	Nama Admin	*Text*	*Text* yang menunjukkan nama dari admin yang melakukan *login*
2	Menu *Dashboard*	Menu Navbar	Nama menu pada website dengan nama menu *dashboard*
3	Menu Laporan	Menu Navbar	Nama menu pada website dengan nama menu laporan yang berisi seluruh laporan yang masuk pada sistem
4	Menu *Trending* Laporan	Menu Navbar	Nama menu pada website dengan nama menu *trending* laporan yang berisi seluruh laporan yang menjadi *trending*
5	Menu Status *Waiting*	Menu Navbar	Nama menu dengan nama menu status *waiting* yang berisi seluruh laporan yang masuk yang berstatus *waiting*
6	Menu Status *OnProgress*	Menu Navbar	Nama menu dengan nama menu status *onprogress* yang berisi seluruh laporan yang masuk yang berstatus *onprogress* atau sedang diperbaiki
7	Menu Status *Done*	Menu Navbar	Nama menu dengan nama menu status *done* yang berisi seluruh laporan yang masuk yang berstatus *done* atau selesai diperbaiki
8	Menu Status *Bloack*	Menu Navbar	Nama menu dengan nama menu status *block* yang berisi laporan-laporan yang masuk namun foto laporan tidak sesuai dengan permasalahan
9	Status Halaman	*Text*	*Text* yang menunjukkan status halaman dari menu yang sedang dijalankan

10	Deskripsi	*Text*	*Text* yang menunjukkan deskripsi secara *detail* dari laporan yang masuk dari seorang pelapor
11	Foto Laporan	Gambar	Gambar yang menunjukkan foto laporan yang masuk mengenai situasi dan kondisi lokasi
12	Alamat	*Text*	*Text* yang menunjukkan alamat atau lokasi dari tempat kejadian pelapor menyampaikan permasalahan tersebut
13	Tanggal Post	*Date*	*Text* yang menunjukkan tanggal pelaporan
14	Kategori Keluhan	*Text*	*Text* yang menunjukkan kategori keluhan yang dipilih oleh pelapor
15	Status	*Text*	*Text* yang menunjukkan status laporan dari pengguna yang melaporkan yang di kelola oleh admin
16	Nama Pelapor	*Text*	*Text* yang menunjukkan nama pelapor yang melaporkan keluhan
17	Dukung Laporan	*Text*	*Text* yang menampilkan jumlah dukungan yang didapatkan oleh laporan
18	Ganti Status Laporan	*Botton*	*Botton* yang digunakan untuk mengganti status laporan
19	*Search* Laporan	Kolom	Kolom yang digunakan untuk melakukan pencarian terhadap laporan *waiting*
20	*Logout*	Gambar	*Icon* yang digunakan untuk melakukan *logout* atau keluar dari sistem

5. Halaman Laporan Status *OnProgress*

Halaman laporan dengan status *onprogress* merupakan halaman yang menampilkan secara keseluruhan laporan-laporan yang memiliki status *onprogress*. Pada halaman laporan status *onprogress* ini terdapat daftar laporan berstatus *onprogress* mulai dari deskripsi, foto laporan, alamat kejadian, tanggal posting, kategori keluhan, status dari laporan, nama pelapor, dan juga jumlah dukung yang didapatkan. Perancangan antarmuka halaman laporan status *onprogress* dapat dilihat pada Gambar 5.24.



Gambar 5. 24 Perancangan Antarmuka Halaman Laporan Status *OnProgress*

Tabel 5.29 menunjukkan penjelasan perancangan antarmuka dari halaman untuk laporan yang memiliki status *onprogress* yang mengacu pada Gambar 5.24.

Tabel 5. 29 Penjelasan Perancangan Antarmuka Halaman Laporan Status *OnProgress*

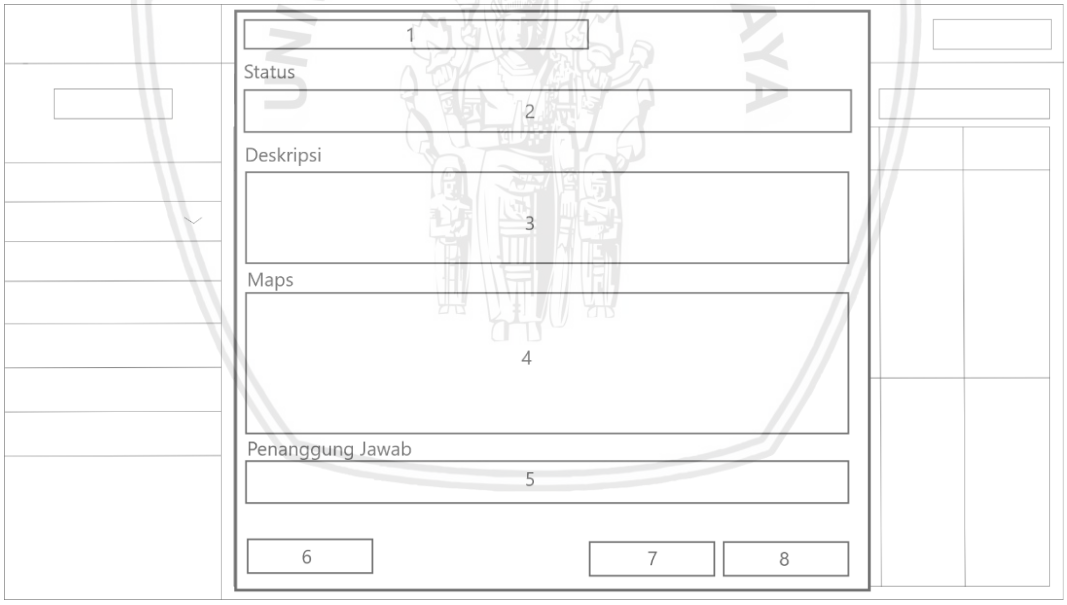
No	Nama Objek	Tipe	Keterangan
1	Nama Admin	*Text*	*Text* yang menunjukkan nama dari admin yang melakukan *login*
2	Menu *Dashboard*	Menu Navbar	Nama menu pada website dengan nama menu *dashboard*
3	Menu Laporan	Menu Navbar	Nama menu pada website dengan nama menu laporan yang berisi seluruh laporan yang masuk pada sistem
4	Menu *Trending* Laporan	Menu Navbar	Nama menu pada website dengan nama menu *trending* laporan yang berisi seluruh laporan yang menjadi *trending*
5	Menu Status *Waiting*	Menu Navbar	Nama menu dengan nama menu status *waiting* yang berisi seluruh laporan yang masuk yang berstatus *waiting*
6	Menu Status *OnProgress*	Menu Navbar	Nama menu dengan nama menu status *onprogress* yang berisi seluruh laporan yang masuk yang

			berstatus *onprogress* atau sedang diperbaiki
7	Menu Status *Done*	Menu Navbar	Nama menu dengan nama menu status *done* yang berisi seluruh laporan yang masuk yang berstatus *done* atau selesai diperbaiki
8	Menu Status *Bloack*	Menu Navbar	Nama menu dengan nama menu status *block* yang berisi laporan-laporan yang masuk namun foto laporan tidak sesuai dengan permasalahan
9	Status Halaman	*Text*	*Text* yang menunjukkan status halaman dari menu yang sedang dijalankan
10	Deskripsi	*Text*	*Text* yang menunjukkan deskripsi secara *detail* dari laporan yang masuk dari seorang pelapor
11	Foto Laporan	Gambar	Gambar yang menunjukkan foto laporan yang masuk mengenai situasi dan kondisi lokasi
12	Alamat	*Text*	*Text* yang menunjukkan alamat atau lokasi dari tempat kejadian pelapor menyampaikan permasalahan tersebut
13	Tanggal Post	*Date*	*Text* yang menunjukkan tanggal pelaporan
14	Kategori Keluhan	*Text*	*Text* yang menunjukkan kategori keluhan yang dipilih oleh pelapor
15	Status	*Text*	*Text* yang menunjukkan status laporan dari pengguna yang melaporkan yang di kelola oleh admin
16	Nama Pelapor	*Text*	*Text* yang menunjukkan nama pelapor yang melaporkan keluhan
17	Dukung Laporan	*Text*	*Text* yang menampilkan jumlah dukungan yang didapatkan oleh laporan
18	Ganti Status Laporan	*Botton*	*Botton* yang digunakan untuk mengganti status laporan

19	*Search* Laporan	Kolom	Kolom yang digunakan untuk melakukan pencarian terhadap laporan berstatus *onprogress*
20	*Logout*	Gambar	*Icon* yang digunakan untuk melakukan *logout* atau keluar dari sistem

6. Halaman Ganti Status Laporan

Halaman ganti status laporan merupakan halaman yang digunakan untuk mengganti status laporan yang masuk pada sistem. Pada halaman ganti status laporan ini berisi id dari laporan yang akan diganti statusnya, status yang akan diganti oleh admin, deskripsi dari laporan, *map* berupa lokasi pelaporan, dan juga nama penanggung jawab petugas yang menangani permasalahan berdasarkan kategori atau bidang keluhannya. Pada halaman ini juga terdapat *botton share* yang digunakan untuk membagikan alamat lokasi pelaporan untuk dibagikan kepada penanggung jawab yang sesuai dengan bidangnya serta terdapat *botton save* yang digunakan untuk menyimpan perubahan yang terjadi. Perancangan antarmuka halaman ganti status laporan dapat dilihat pada Gambar 5.25.



Gambar 5. 25 Perancangan Antarmuka Halaman Ganti Status Laporan

Tabel 5.30 menunjukkan penjelasan secara *detail* perancangan antarmuka dari halaman untuk mengganti status laporan yang mengacu pada Gambar 5.25.

Tabel 5. 30 Penjelasan Perancangan Antarmuka Halaman Ganti Status Laporan

No	Nama Objek	Tipe	Keterangan
1	Id Laporan	*Text*	*Text* yang menunjukkan id dari laporan yang akan diganti statusnya

2	Status Laporan	*Dropdown*	*Dropdown* yang berisi jenis status yang digunakan untuk laporan ini yakni status *waiting, onprogress, done* serta *block*
3	Deskripsi	*Text*	*Text* yang menunjukkan deskripsi secara *detail* dari laporan yang masuk dari seorang pelapor
4	*Maps*	*Map*	*Map* yang menunjukkan lokasi kejadian yang dilaporkan ditunjukkan dengan adanya marker pada *map* tersebut
5	Penanggung Jawab	*Text*	*Text* yang menunjukkan nama penanggung jawab yang menangani permasalahan berdasarkan kategori atau bidang keluhannya
6	*Share*	*Botton*	*Botton share* yang digunakan untuk membagikan lokasi permasalahan kepada penanggung jawab melalui *social media* yang digunakan oleh penanggung jawab tersebut pada sebuah *smartphone* yang dimiliki
7	Batal	*Botton*	*Botton cancel* digunakan jika tidak ingin jadi melakukan perubahan status laporan
8	*Save*	*Botton*	*Botton save* yang digunakan untuk menyimpan perubahan status laporan jika status sudah selesai diganti

5.2 Implementasi Sistem

Implementasi sistem merupakan tahapan dari terselesaikannya sebuah proses perancangan sistem. Dimana implementasi sistem merupakan tahapan selanjutnya dari bab ini yang saling terkait yakni merupakan bab perancangan dan implementasi sistem. Implementasi dapat dilakukan berdasarkan hasil yang telah didapatkan pada proses analisis kebutuhan dan perancangan sistem. Kebutuhan-kebutuhan yang sudah didapatkan akan di interpretasikan pada tahapan ini. Implementasi sistem akan menjabarkan mengenai spesifikasi sistem untuk membuat aplikasi, implementasi dari kode program, implementasi basis data, dan juga implementasi antarmuka yang sudah dirancang pada tahap perancangan sistem.

5.2.1 Spesifikasi Sistem

Pada subbab ini menjelaskan mengenai spesifikasi yang akan digunakan dalam membangun aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang berbasis Android. Penjelasan spesifikasi sistem ini akan dibagi menjadi dua yakni spesifikasi perangkat keras dan spesifikasi perangkat lunak. Perangkat keras yang digunakan ialah laptop yang digunakan sebagai media dalam mengimplementasikan kode program *smartphone* yang digunakan untuk media *debug* aplikasi pengaduan masyarakat. Untuk spesifikasi perangkat keras laptop pada dapat dilihat pada Tabel 5.31 dan spesifikasi perangkat keras *smartphone* pada Tabel 5.32. Dan spesifikasi perangkat lunak dapat dilihat pada Tabel 5.33.

Tabel 5. 31 Spesifikasi Perangkat Keras Laptop

Keterangan	Spesifikasi
System Model	ASUS A455L
Processor	Intel(R) Intel Core i5 CPU @ 2.20GHz
Memory	8GB
Display	14.0" WideScreen HD (1366x768)

Tabel 5. 32 Spesifikasi Perangkat Keras *Smartphone*

Keterangan	Spesifikasi
System Model	ASUS ZENFONE 5
Processor	Dual-core 1.2 GHz
Memory	16 GB, 2 GB RAM
Display	IPS Sharp/AUO, Adreno 306

Tabel 5. 33 Spesifikasi Perangkat Lunak

Keterangan	Spesifikasi
Operating System	Windows 10 Pro 64 bit
Programming Language	Java, PHP
DBMS	MySQL
Software	Android Studio 3.0, Notepad ++, Sublime Text 3

Spesifikasi sistem operasi yang digunakan pada pengembangan aplikasi pengaduan masyarakat ini dapat dilihat pada Tabel 5.34 seperti berikut.

Tabel 5. 34 Sistem Operasi

Nama Komponen	Spesifikasi
Sistem Operasi Pengembangan Sistem	Windows 10 Pro 64 bit
Sistem Operasi Server	Windows 10 Pro 64 bit

5.2.2 Implementasi Kode Program

Berdasarkan perancangan komponen yang telah dirancang pada tahapan perancangan sistem yang telah dirancang sebelumnya akan digunakan pada implementasi kode program pada proses implementasi sistem. Perancangan komponen tersebut berupa algoritme-algoritme yang akan diubah kedalam bentuk *pseudocode*, pada bagian implementasi kode program ini *pseudocode* yang telah dibuat akan diimplmentasikan dalam bentuk bahasa pemrograman yakni bahasa pemrograman Java. Algoritme yang akan diimplementasikan pada bagian ini ialah *Method upload()* dari *Class UploadActivity*, *Method save()* dari *Class EditMyLaporanActivity*, dan *Method editStatus()* dari *Class WaitingController*. Berikut merupakan implementasi kode program masing-masing *method* yang digunakan untuk membangun aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang berbasis Android.

5.2.2.1 Implementasi Kode Program *Method "upload"* dari *Class "UploadActivity"*

Untuk membuat algoritme dari *Method* upload dan *Class* UploadActivity berdasarkan implementasi dari *method* dan *class* tersebut. *Method* ini digunakan untuk membangun aplikasi dengan fitur menambah laporan, *method upload*() berasal dari *class UploadActivity* yang berisi getdes, getalamat, foto convert to string, set date, get id from UserData yang digunakan dalam menambah laporan dimana jika parameter-parameter tersebut berhasil dimasukkan dan *object call* dan *response* tidak kosong makan BaseResponse akan mengirimkan *response* dan laporan yang sudah ditambahkan tersebut akan masuk pada halaman main dan jika *object call* dan *response* atau parameter ada yang kosong makan akan muncul pesan "Terjadi Kesalahan, Silahkan Ulangi Kembali!" setelah itu proses menambah laporan selesai. Tabel 5.35 merupakan kode program dari *method* upload dari class UploadActivity.

Nama *Class* : UploadActivity

Nama *Method* : upload

Tabel 5. 35 Kode Program *Method* "upload" dari *Class* "UploadActivity"

No	Kode Program

1
2
3
	`private void upload() {`
	`  des = textDesc.getText().toString();`

```
4       if(TextUtils.isEmpty(des)) {
5           textDesc.setError("Deskripsi Tidak Boleh Kosong!");
6       return;
7       }
8
9       foto = convertToString();
10      alamat = tvAddress.getText().toString();
11      if (TextUtils.isEmpty(alamat)){
12          tvAddress.setError("Pick Lokasi Terlebih Dahulu!");
13      return;
14      }
15
16      tgl_post = setDate();
        User user = PrefUtil.getUser(this,
17  PrefUtil.USER_SESSION);
18      pelapor = user.getUserData().getId();
19
20      loading.setVisibility(View.VISIBLE);
        AddLaporanService addLaporanService = new
21  AddLaporanService(this);
        addLaporanService.addLaporan(des, foto, lat, lng,
    alamat, tgl_post, status, pelapor, kategori, new Callback()
22  {
23          @Override
            public void onResponse(Call call, Response response)
24  {
                BaseResponse baseResponse = (BaseResponse)
25  response.body();
26              Log.d(TAG, baseResponse.getMessage());
                Toast.makeText(UploadActivity.this,
27  baseResponse.getMessage(), Toast.LENGTH_LONG).show();
28
29              loading.setVisibility(View.GONE);
                Intent intent = new Intent(UploadActivity.this,
30  Main.class);
31  intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
32              startActivity(intent);
33          }
34
35          @Override
36          public void onFailure(Call call, Throwable t) {
37
38          }
39      });
    }
```

Tabel 5.36 merupakan penjelasan dari kode program dari *method* "upload" dari *class* "*Upload*Activity".

Tabel 5. 36 Penjelasan Kode Program *Method* "upload" dari *Class* "*Upload*Activity"

Baris	Penjelasan
1	Implementasi *method* upload
2	Kondisi jika deskripsi tidak kosong

3 - 7	Kondisi jika deskripsi kosong maka akan muncul pesan “Deskripsi Tidak Boleh Kosong!”
8	*Convert foto to string*
9	Kondisi jika alamat tidak kosong
10 - 14	Kondisi jika alamat kosong maka akan muncul pesan “*Pick Location* Terlebih Dahulu!”
15	*set tgl_post*
16 - 18	*getId* pelapor dari UserData
19	*Visible loading = visible*
20 - 21	Proses melakukan implementasi *Method* addLaporan pada *Class* addLaporanService dengan parameter des, foto, lat, lng, alamat, tgl_post, status, pelapor, kategori
22 - 23	Proses memanggil *Method* on*Response* yang berisikan *Response* dari addLaporanService
24 - 33	Proses jika onResponse berhasil yakni dengan menampilkan *message* dari base response
34 - 37	Proses jika gagal melakukan proses memanggil *Method* on*Response* dan menampilkan *message error*
38 - 39	Proses selesai

5.2.2.2 Implementasi Kode Program *Method “save”* dari *Class “EditMyLaporanActivity”*

Untuk merancang algoritme mengubah biodata menggunakan *method* “save” dari *Class* “EditMyLaporanActivity” yang dilihat berdasarkan implementasi kode program dari *method* dan *class* tersebut. Pada *method* save ini berasal dari *class* EditMyLaporanActivity yang berisi *method* untuk melakukan *update* laporan pribadi dengan melakukan implementasi dari *method* save yang berisi getdes, getalamat, foto *convert to string*. Jika parameter-parameter tersebut berhasil dimasukkan dan *object call* dan *response* tidak kosong makan BaseResponse akan mengirimkan *response* dan muncul pesan sukses dan jika *object call* dan *response* atau parameter ada yang kosong makan akan muncul pesan “Terjadi Kesalahan!“ setelah itu proses mengubah laporan pribadi selesai dilakukan. Implementasi kode program ini digunakan untuk mengubah laporan yang pernah dilakukan oleh pelapor atau masyarakat itu sendiri jika pada laporan yang sudah dilaporkan terdapat kesalahan maka dapat dilakukan *edit* laporan ini namun dengan laporan tersebut masih bersifat status *waiting*. Implementasi kode program dari *method* save dari *class* EditMyLaporanActivity dapat dilihat pada Tabel 5.37.

Nama *Class* : EditMyLaporanActivity

Nama *Method* : save

Tabel 5. 37 Kode Program *Method* "save" dari *Class* "EditMyLaporanActivity"

No	Kode Program
1	`private void save() {`
2	`    loading.setVisibility(View.VISIBLE);`
3	`    des = tvDesc.getText().toString();`
4	`    alamat = tvLocation.getText().toString();`
5	`    if (bvKlik == false){`
6	`        fotoS = "";`
7	`    }else if(bvKlik == true){`
8	`        fotoS = convertToString();`
9	`    }`
10	
11	`    EditMyLaporanService editMyLaporanService = new EditMyLaporanService(this);`
12	`    editMyLaporanService.editMyLaporan(id, des, lat, lng, alamat, fotoS, idKategori, new Callback() {`
13	`        @Override`
14	`        public void onResponse(Call call, Response response) {`
15	`            BaseResponse baseResponse = (BaseResponse) response.body();`
16	`            Log.d(TAG, baseResponse.getMessage());`
17	`            Toast.makeText(EditMyLaporanActivity.this, baseResponse.getMessage(), Toast.LENGTH_LONG).show();`
18	`            loading.setVisibility(View.GONE);`
19	
20	`            finish();`
21	`        }`
22	
23	`        @Override`
24	`        public void onFailure(Call call, Throwable t) {`
25	`            Log.d(TAG, "Error :" + t);`
26	`            Toast.makeText(EditMyLaporanActivity.this, "Terjadi Kesalahan", Toast.LENGTH_LONG).show();`
27	`        }`
28	`    });`
29	`}`

Berikut merupakan penjelasan dari kode program dengan *method* "save" dari *class* "EditMyLaporanActivity" dapat ditunjukkan pada Tabel 5.38.

Tabel 5. 38 Penjelasan Kode Program *Method* "save" dari *Class* "EditMyLaporanActivity"

Baris	Penjelasan
1	Implementasi *method* save
2	*Loading visibility = visible*
3	Kondisi apabila deskripsi tidak kosong
4	Kondisi apabila alamat tidak kosong

5 - 6	Kondisi apabila foto kosong
7 - 10	Kondisi apabila foto tidak kosong
11 - 12	Proses melakukan implementasi *method* editMyLaporan pada c*lass* editMyLaporanService dengan parameter id, des, fotoS, lat, lng, alamat, idKategori
13 - 14	Proses memanggil *method* on*Response* yang berisikan *Response* dari editMyLaporanService
15 - 22	Proses jika onResponse berhasil yakni dengan menampilkan *message* dari base response
23 - 27	Proses jika gagal melakukan proses memanggil *method* on*Response* dan menampilkan *message error*
28 - 29	Proses selesai

5.2.2.3 Implementasi Kode Program *Method* "editStatus ()" dari *Class* "WaitingController"

Perancangan komponen untuk melihat laporan berdasarkan status *waiting* dirancang berdasarkan *Method* "editStatus" dari *Class* "WaitingController". Fitur melihat laporan berdasarkan status *waiting* dimana menu laporan *waiting* ini dapat melihat semua laporan-laporan yang dilaporkan masyarakat yang berstatus *waiting*. Tabel 5.39 merupakan implementasi kode program untuk melihat laporan berdasarkan status *waiting*.

Nama *Class* : WaitingController

Nama *Method* : *editStatus*

Tabel 5. 39 Kode Program *Method* "*editStatus*" dari *Class* "WaitingController"

No	Kode Program
1	`function editStatus(Request $request){`
2	`$id = $request->input("idLaporan");`
3	`$status = $request->input("status");`
4	
5	`$laporan = Laporan::select('laporan.id','laporan.deskripsi', 'laporan.foto', 'laporan.lat',`
6	`'laporan.lng', 'laporan.alamat', 'laporan.tgl_post', 'kategori.kategori', 'laporan.status', 'user.nama AS pelapor',`
7	`DB::raw('count(dukung.idLaporan) As dukung'))`
8	`->groupBy('laporan.id')`
9	`->leftJoin('user', 'user.id', '=', 'laporan.idUser')`
10	`->leftJoin('kategori', 'kategori.id', '=', 'laporan.idKategori')`
11	`->leftJoin('dukung', 'dukung.idLaporan', '=', 'laporan.id')`
12	

```
            ->leftJoin('status',      'status.id',      '=',
13  'laporan.status')
14              ->where('laporan.id','=',"$id")
15              ->update(['laporan.status' => $status]);
16
17          if($laporan){
18              $response["error"] = FALSE;
19              $response["message"] = "Data Berhasil di Proses";
20              $response["data"]["id"]= $id;
21              $response["data"]["status"]= $status;
22          }
23
24          return $response;
25      }
```

Tabel 5.40 merupakan penjelasan dari kode program dengan *method* "*editStatus*" dari *Class* "WaitingController".

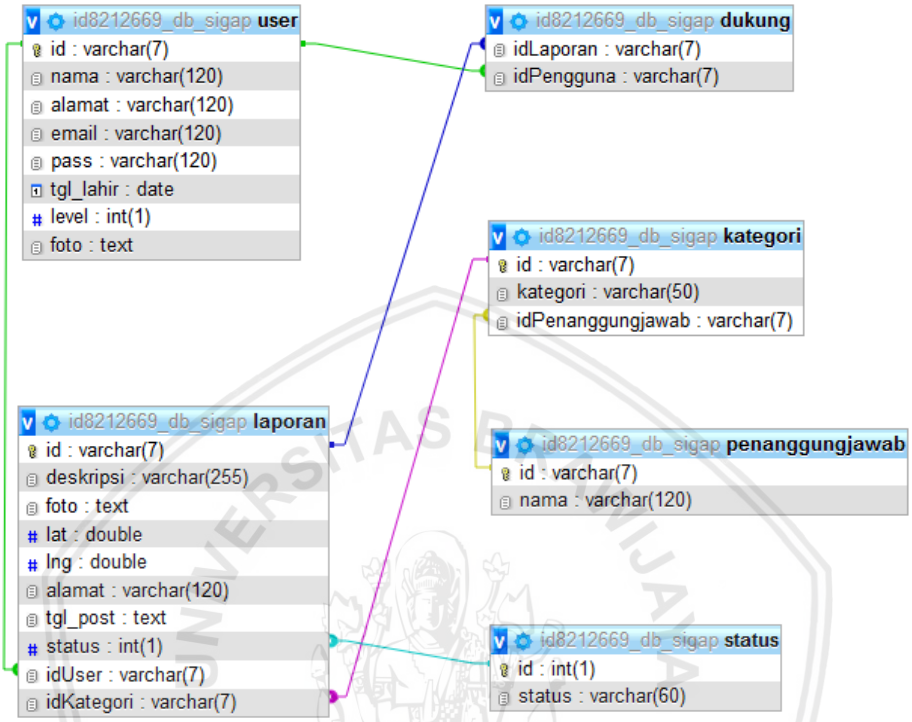
Tabel 5. 40 Penjelasan Kode Program *Method* "*editStatus*" dari *Class* "WaitingController"

Baris	Penjelasan
1	Implementasi *method* editStatus memiliki parameter *request*
2 - 4	*Request* id dan status dari idLaporan dan status
5 - 15	Deklarasi *variable* laporan sebagai *query* untuk mengganti status laporan
16 - 22	Proses yang terjadi apabila *select query* mengganti status berhasil dilakukan maka akan muncul pesan "Data Berhasil di Proses"
23 - 24	Proses yang terjadi apabila *select query* mengganti status tidak berhasil dilakukan maka akan muncul pesan "Data Tidak Berhasil di Proses"
25	Proses selesai

5.2.3 Implementasi Basis Data

Implementasi basis data dibuat berdasarkan pada perancangan ERD atau *Entity Relationship Diagram*. Implementasi basis data yang digunakan menggunakan basis data MySQL. Gambar 5.26 merupakan diagram yang menggambarkan semua tabel yang digunakan pada basis data aplikasi pengaduan masyarakat berbasis Android beserta kolom dan tipe datanya pada setiap kolom. Dimana tabel-tabel yang berkaitan akan dihubungkan dengan suatu relasi. Berikut merupakan diagram dari implementasi basis data. Semua tabel yang diimplementasikan mengacu berdasarkan perancangan basis data yang sudah dibuat sebelumnya pada tahapan perancangan basis data sistem. Pada perancangan basis data terdiri dari enam tabel yakni tabel *user*, tabel laporan,

tabel kategori, tabel penanggungjawab, tabel dukung, dan tabel status. Dan pada implementasi basis data diimplementasikan dalam diagram fisik yang memiliki relasi antar tabel yang mengacu pada pada ERD pada Gambar 5.6.



Gambar 5. 26 Implementasi Basis Data

Berdasarkan Tabel 5.7 pada perancangan yang merupakan tabel *user* memiliki lima *field* yang akan diimplementasikan yang berisi data-data *user* meliputi id, nama, alamat, email, pass, tgl_lahir, dan level. *Email* dan *password user* digunakan untuk menginputkan pada saat *login*. Sedangkan untuk id merupakan *auto increment* yang secara otomatis diatur. Selain itu Id merupakan *foreign key* pada tabel lain yakni tabel laporan dan tabel dukung dimana kedua tabel tersebut memiliki hubungan dalam proses mendukung laporan keluhan untuk pengguna yang mendukung laporan pengguna lainnya. Mengacu pada perancangan Tabel 5.8 yakni tabel laporan akan digunakan pada implementasi yang terdiri terdiri dari sepuluh *field* dimana tabel tersebut berisi data laporan yang didapatkan dari proses melakukan pengaduan. Tabel Laporan memiliki satu *primary key* yaitu id. Tabel laporan juga memiliki *foreign key* yaitu id*User* yang merupakan penghubung antara tabel laporan dengan tabel *user*. Selain terhubung dengan tabel *user*, tabel laporan terhubung dengan tabel kategori dan tabel dukung. Tabel kategori itu sendiri akan menampung kategori-kategori permasalahan lalu lintas yang menjadi tanggungjawab Dinas Perhubungan Kota Malang, tabel ini terdiri dari tiga *field* yakni id, kategori, dan idPenanggungjawab, yang menghubungkan tabel laporan dan tabel kategori yakni idKategori yang ada pada tabel laporan.

Tabel kategori mengacu kepada perancangan basis data pada Tabel 5.10. Tabel dukung berisi data dukungan terhadap laporan yang disampaikan, tabel ini terdiri dari dua *field* yakni idLaporan dan idPengguna dimana idLaporan merupakan *foreign key* dari tabel laporan dan idPengguna merupakan *foreign key* dari tabel *user*. Tabel dukung mengacu berdasarkan perancangan basis data pada Tabel 5.11 yang akan digunakan pada implementasi ini.

Tabel lainnya yakni tabel penanggungjawab, tabel ini mengacu pada perancangan basis data yang terdapat pada Tabel 5.12. Tabel ini berisi data penanggungjawab yang menangani laporan yang masuk yang bertanggung jawab terhadap laporan yang ada untuk menangani permasalahan langsung ke lokasi kejadian. Tabel penanggungjawab terdiri dari dua *field* yakni id dan nama dimana id pada Tabel ini memiliki *foreign key* pada tabel lain yaitu tabel kategori yang memiliki hubungan berkaitan dengan penanggungjawab yang menangani sesuai dengan kategori permasalahan ataupun berdasarkan bidang pekerjaan.

5.2.4 Implementasi Antarmuka

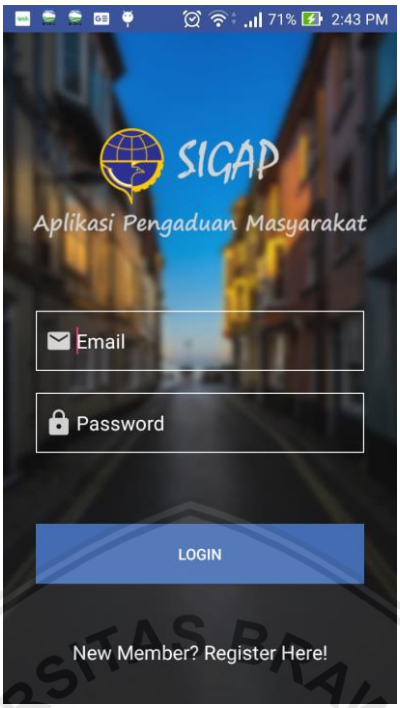
Implementasi antarmuka yang ada pada bagian ini merupakan antarmuka sistem hasil pada proses perancangan yang telah dirancang sebelumnya. Implementasi antarmuka pada aplikasi pengaduan masyarakat terdapat beberapa tampilan antarmuka yang digunakan untuk sisi pengguna maupun *administrator*. Subbab ini akan menjelaskan mengenai bagian tampilan antarmuka tersebut dengan masing-masing menu yang ada pada aplikasi ini dengan menyertakan hasil *screenshot* dari aplikasi yang telah dibuat.

5.2.4.1 Implementasi Antarmuka *Mobile App*

Implementasi antarmuka yang pertama untuk implementasi *layout* aplikasi pengaduan masyarakat untuk *mobile app* yang nantinya akan digunakan oleh pelapor atau masyarakat. Pada implementasi antarmuka untuk aplikasi *mobile* pengaduan masyarakat ini terdapat beberapa halaman. Dimana terdiri dari lima halaman menu utama, seperti menu daftar laporan yang berisi seluruh laporan-laporan yang dilaporkan oleh masyarakat, menu *trending* laporan yang digunakan untuk melihat laporan yang menjadi *trending*, menu *support* ialah menu yang digunakan untuk melihat laporan yang pernah di dukung oleh pengguna, menu menambah laporan ialah menu untuk menambah laporan baru, dan menu terakhir ialah menu *profile* yang mengelola biodata dan laporan pengguna tersebut. Berikut merupakan implementasi antarmuka untuk *mobile app* yang dapat dilihat pada Gambar 5.27 sampai Gambar 5.37.

1. Implementasi Antarmuka Halaman *Login*

Halaman *login* untuk *mobile app* dapat dilihat pada Gambar 5.27. Halaman *login* ini merupakan halaman awal yang pertama muncul dari *mobile app*. Pengguna yang akan menggunakan aplikasi ini harus melakukan *login* terlebih dahulu dengan menggunakan *email* dan *password* yang dimiliki. Pada halaman *login* ini merupakan sebuah *form* untuk mengisi *email* dan *password*. Berikut merupakan implementasi antarmuka *login mobile app*.



Gambar 5. 27 Implementasi Antarmuka Halaman *Login*

2. Implementasi Antarmuka Halaman *Register*

Halaman *register* ini merupakan halaman ketika pengguna akan membuat akun dengan mengakses halaman *register* ini yang terletak pada halaman *login*. Pengguna yang akan mendaftarkan diri pada halaman *register* ini dengan menginputkan data diri seperti, *username*, *email*, dan *password*. Implementasi antarmuka untuk halaman *register* terdapat pada Gambar 5.28 seperti berikut.

Gambar 5. 28 Implementasi Antarmuka Halaman *Register*

3. Implementasi Antarmuka Halaman Daftar Laporan

Setelah pengguna melakukan *login* maka pengguna akan masuk pada halaman daftar laporan. Halaman daftar laporan ini berisi halaman laporan yang disampaikan oleh seluruh pengguna. Pada laporan yang dibagikan terdapat foto, nama, dan lokasi kejadian, foto beserta deskripsi dari laporan. Implementasi antarmuka pada aplikasi ini terdiri dari beberapa menu, menu yang pertama yakni menu daftar laporan. Setelah pengguna melakukan *login* maka pengguna akan masuk pada halaman daftar laporan yang mana halaman ini merupakan halaman pertama yang muncul ketika selesai melakukan *login*. Halaman daftar laporan ini berisi halaman yang didalamnya terdapat laporan-laporan mengenai keluhan yang disampaikan oleh seluruh pengguna. Pada laporan yang dibagikan terdapat foto, nama, dan alamat dari pengguna yang melaporkan. Halaman daftar laporan ini akan menampilkan laporan-laporan yang terbaru dari laporan yang disampaikan masyarakat. Berikut merupakan implementasi antarmuka untuk halaman daftar laporan pada Gambar 5.29.

Gambar 5. 29 Implementasi Antarmuka Halaman Daftar Laporan

4. Implementasi Antarmuka Halaman *Trending* Laporan

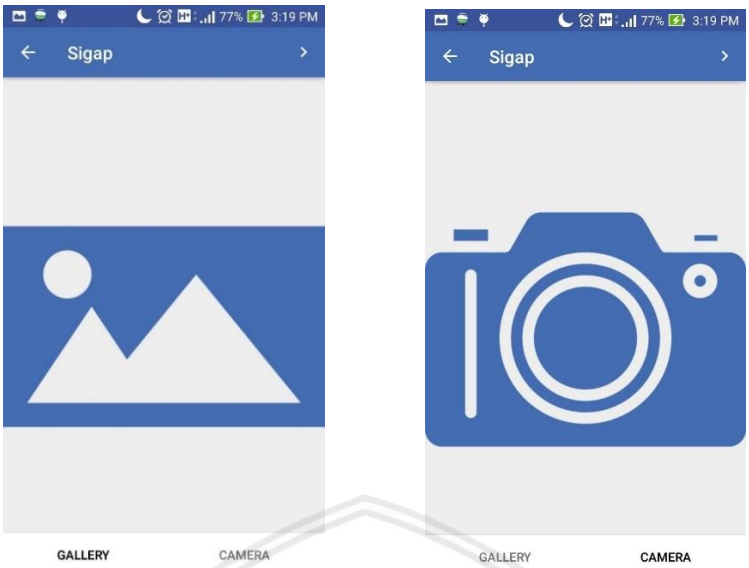
Halaman *trending* laporan dapat diakses ketika pengguna memilih menu *trending* laporan dimana menu ini merupakan menu yang kedua pada aplikasi ini. Laporan-laporan ini akan tersusun sesuai dengan laporan dengan dukungan terbanyak. Laporan yang termasuk ke dalam *trending* laporan ini memiliki jumlah dukungan terbanyak dimana hanya sepuluh laporan yang memiliki jumlah dukungan terbanyak. Untuk itu halaman *trending* laporan ini digunakan oleh masyarakat untuk saling *support* terhadap laporan yang dialami di Kota Malang.

Halaman *trending* laporan dapat menampilkan laporan yang mendapatkan dukungan dari pelapor lain. Laporan-laporan ini akan tersusun sesuai dengan laporan dengan dukungan terbanyak. Halaman *trending* laporan digunakan untuk masyarakat jika masyarakat tersebut memiliki permasalahan yang sama dengan pelapor yang sudah menyampaikan laporan tersebut terlebih dahulu. Perancangan antarmuka halaman *trending* laporan digambarkan pada Gambar 5.30 yakni sebagai berikut.

Gambar 5. 30 Implementasi Antarmuka Halaman *Trending* Laporan

5. Implementasi Antarmuka Halaman Menambah Laporan

Menu yang ketiga pada perancangan antarmuka ini ialah menu menambah laporan. Menu ini merupakan halaman yang digunakan untuk membuat laporan keluhan baru. Untuk membuat laporan baru pelapor terlebih dahulu mengambil gambar untuk dapat melaporkan keluhan. Untuk mengambil gambar pada aplikasi ini dapat mengakses kamera dari *smartphone* yang digunakan ketika pelapor ingin mengambil gambar secara langsung dari *smartphone* tetapi jika pelapor ingin mengunggah gambar yang ada pada *gallery smartphone* maka pelapor dapat memilih foto yang ada pada *gallery*. Proses pengambilan gambar untuk pengaduan masyarakat hanya digunakan salah satu gambar yang diambil baik dari kamera *smatphone* pelapor maupun dari *gallery smartphone* yang gambarnya sudah di simpan terlebih dahulu. Gambar 5.31 menggambarkan halaman laporan keluhan untuk mengunggah gambar.



Gambar 5. 31 Implementasi Antarmuka Halaman untuk *Gallery* (kiri) dan Implementasi Antarmuka Halaman *Camera* (kanan)

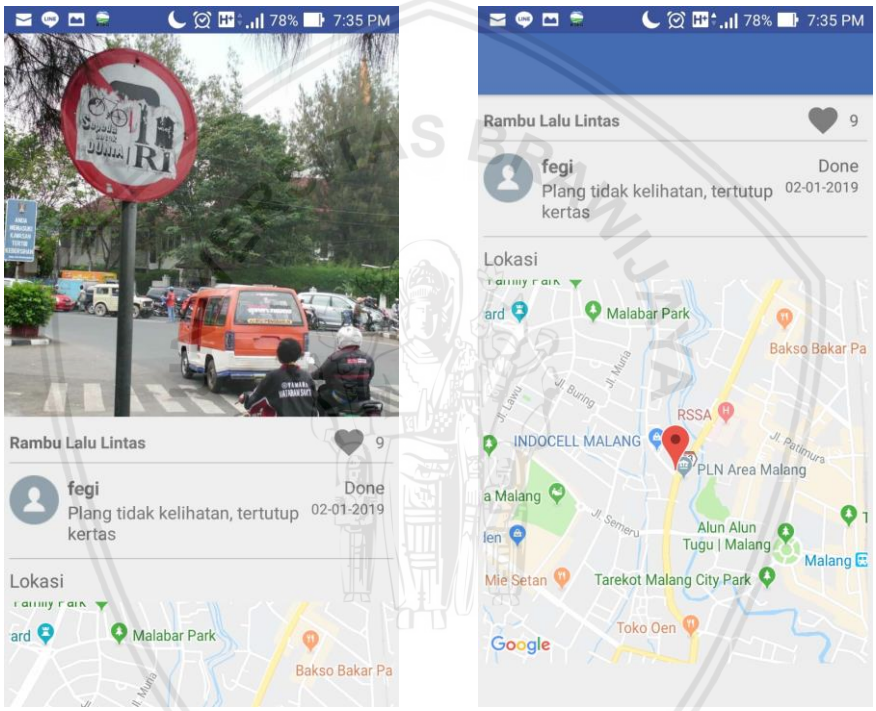
6. Implementasi Antarmuka Halaman *Upload* Laporan

Halaman ini dapat diakses ketika pelapor sudah mengambil gambar pada halaman sebelumnya. Pada halaman *upload* laporan terdapat foto yang sudah diambil oleh pelapor lalu pelapor dapat memilih kategori keluhan berdasarkan masalah yang dilaporkan, terdapat tombol *map* untuk memilih lokasi kejadian dan memasukkan deskripsi dari permasalahan setelah itu pelapor dapat menekan tombol *upload* yang terletak dibawah halaman *upload* laporan. Gambar 5.32 menggambarkan perancangan antarmuka halaman *upload* laporan.

Gambar 5. 32 Implementasi Antarmuka Halaman *Upload* Laporan

7. Implementasi Antarmuka Halaman *Detail* Laporan

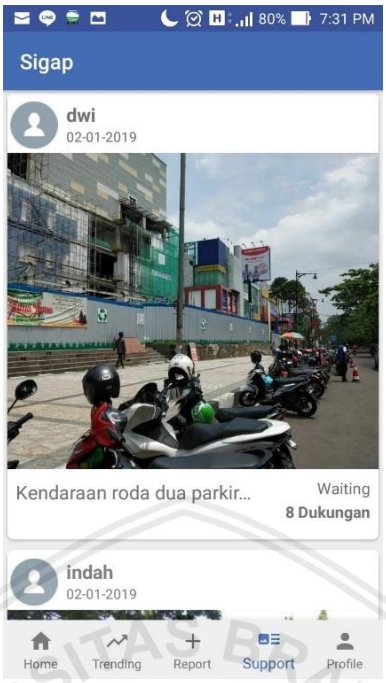
Halaman *detail* laporan dapat dilihat ketika pengguna ingin mengetahui secara *detail* dari laporan dengan mengklik laporan yang diinginkan. Halaman ini akan menampilkan laporan secara *detail* seperti foto beserta deskripsi lengkap dari laporan, dan juga lokasi dari laporan tersebut. Terdapat pula *user* yang melaporkan laporan tersebut. Pada halaman *detail* laporan ini dapat melihat foto permasalahan yang terjadi secara jelas dan dapat melihat lokasi kejadian secara akurat dengan adanya map pada halaman *detail* laporan ini. Untuk itu masyarakat dapat melihat dan mengetahui lokasi yang dimaksudkan pada laporan tersebut. Pada halaman *detail* laporan masyarakat dapat melakukan proses *support* laporan terhadap laporan yang dilaporkan oleh pelapor lain. Halaman *detail* laporan dapa dilihat pada Gambar 5.33 seperti berikut.



Gambar 5. 33 Implementasi Antarmuka Halaman *Detail* Laporan

8. Implementasi Antarmuka Halaman Daftar Dukungan

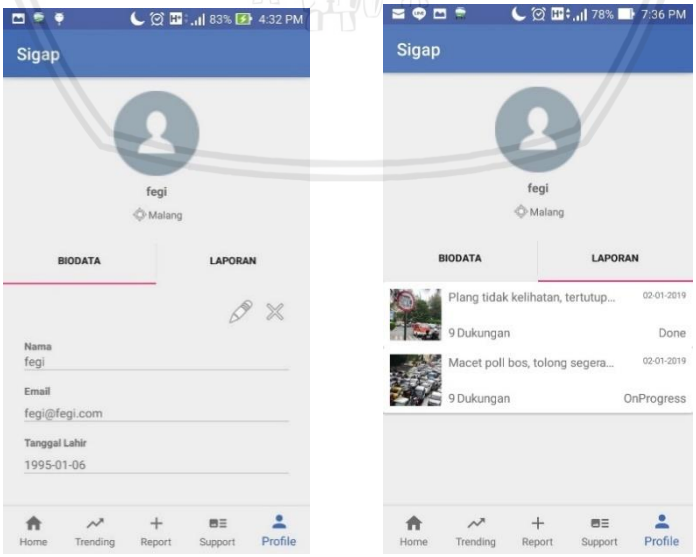
Halaman daftar dukungan merupakan halaman yang digunakan untuk melihat daftar dari laporan-laporan pribadi yang pernah dilakukan untuk mendukung sebuah laporan yang dilaporkan oleh pengguna lain karena memiliki permasalahan-permasalahan yang dialami sama dengan pengguna lain. Tampilan halaman daftar dukungan ini sama seperti halaman daftar laporan atau menu *home*. Namun perbedaannya halaman daftar dukungan ini hanya berisi laporan-laporan yang pernah didukung. Pada halaman daftar dukungan dapat diarahkan juga pada halaman detail laporan yang didukungnya. Seperti dapat melihat laporan apa yang didukung, lokasi kejadian, deskripsi, kategori, foto pelaporan, serta nama pelapor yang melaporkan pengaduan. Gambar 5.34 merupakan halaman daftar dukungan.



Gambar 5. 34 Implementasi Antarmuka Halaman Daftar Dukungan

9. Implementasi Antarmuka Halaman *Profile* Pelapor

Gambar 5.35 merupakan gambar dari halaman *profile* pelapor. Pada halaman *profile* pelapor terdapat dua tab yang digunakan dimana tab yang pertama menampilkan data diri dari pelapor tersebut dan tab yang kedua menampilkan seluruh laporan yang pernah diunggah oleh pengguna itu sendiri. Pada halaman data diri ini juga terdapat dua menu yang memiliki fungsi untuk *edit* biodata dan juga *logout*. Untuk halaman data diri dan halaman laporan pribadi dapat ditunjukkan pada Gambar 5.35.



Gambar 5. 35 Implementasi Antarmuka Halaman untuk Biodata Pengguna (kiri) dan Implementasi Antarmuka Halaman Laporan Pengguna (kanan)

10. Implementasi Antarmuka Halaman Mengubah *Profile*

Halaman mengubah *profile* merupakan halaman yang dilakukan untuk mengubah *profile* pengguna yang memiliki akun dan sudah melakukan proses *login* pada saat awal membuka aplikasi *mobile* ini. Halaman mengubah *profile* dilakukan setelah menekan *tabs profile* dan menekan *icon edit* pada halaman tab *profile*. Pada halaman mengubah *profile* ini pengguna dapat mengubah nama, *password*, alamat, dan juga tanggal lahir pada kolom yang terdapat pada halaman mengubah *profile*. Namun pada halaman mengubah profile ini tidak dapat melakukan perubahan pada foto profil dari pengguna dan tidak dapat melakukan perubahan pada email yang telah di daftarkan. Implementasi halaman mengubah *profile* terdapat pada Gambar 5.36 seperti berikut.

Gambar 5. 36 Implementasi Antarmuka Halaman Mengubah *Profile*

11. Implementasi Antarmuka Halaman Mengubah Laporan

Halaman mengubah laporan dapat diakses dengan menekan halaman tab laporan terlebih dahulu pada halaman *profile* pengguna. Halaman mengubah laporan digunakan untuk mengubah laporan pribadi yang sudah dilaporkan namun belum ditindaklanjuti oleh pihak Dinas Perhubungan Kota Malang. Halaman mengubah laporan ini berupa *listview* yang berisi laporan-laporan yang pernah dilaporkan oleh pengguna yang sudah melakukan *logi*n. Untuk mengakses halaman mengubah laporan pribadi ini pengguna harus memilih laporan yang ada pada *listview* untuk diubah setelah itu akan masuk pada halaman mengubah laporan. Laporan yang dapat diubah yakni gambar, kategori keluhan, deskripsi, dan juga lokasi kejadian. Implementasi antarmuka halaman mengubah laporan pribadi dapat dilihat pada Gambar 5.37 seperti berikut.

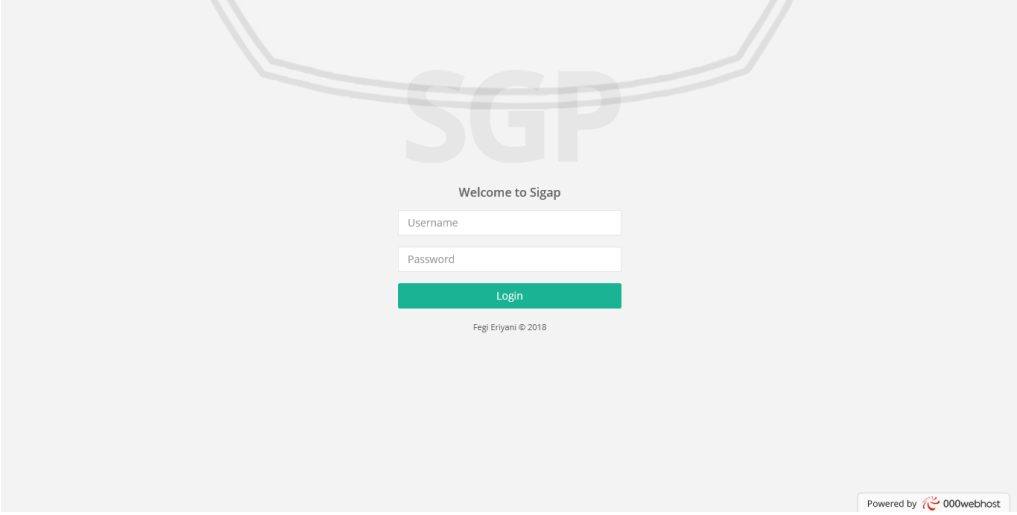
Gambar 5. 37 Implementasi Antarmuka Halaman Mengubah Laporan

5.2.4.2 Implementasi Antarmuka Web

Implementasi antarmuka yang kedua ialah perancangan antarmuka untuk *web* yang akan digunakan untuk admin atau pihak Dinas Perhubungan Kota Malang. Halaman web ini terdiri dari beberapa halaman. Implementasi antarmuka web dapat dilihat pada Gambar 5.38 sampai dengan Gambar 5.45.

1. Implementasi Antarmuka Halaman *Login*

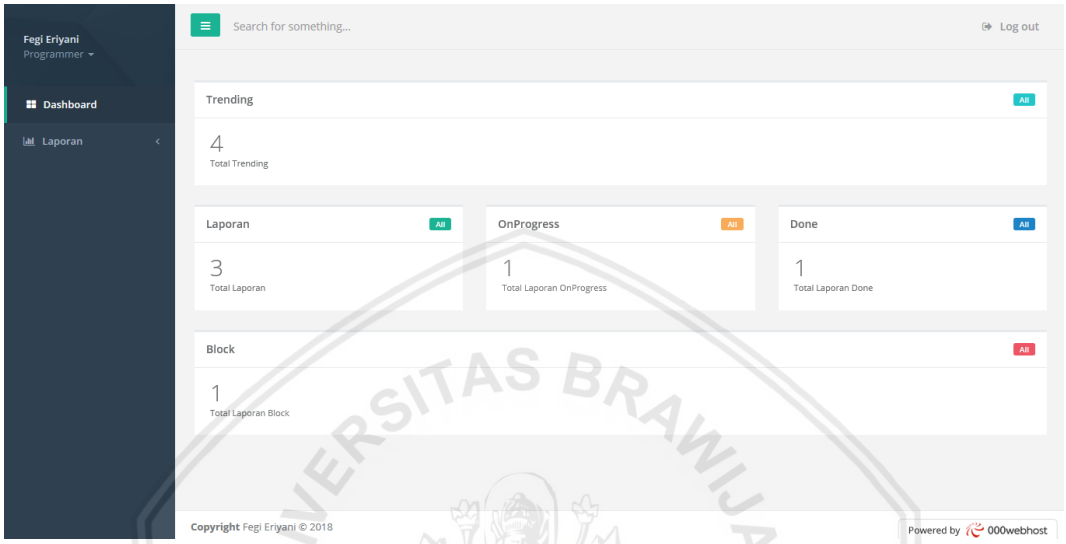
Gambar 5.38 merupakan halaman *login* untuk web yang mana halaman awal yang muncul ketika admin mengakses web pengaduan masyarakat. Pengguna harus melakukan *login* terlebih dahulu pada halaman ini untuk dapat mengakses web tersebut.



Gambar 5. 38 Implementasi Antarmuka Halaman *Login*

2. Implementasi Antarmuka Halaman *Dashboard*

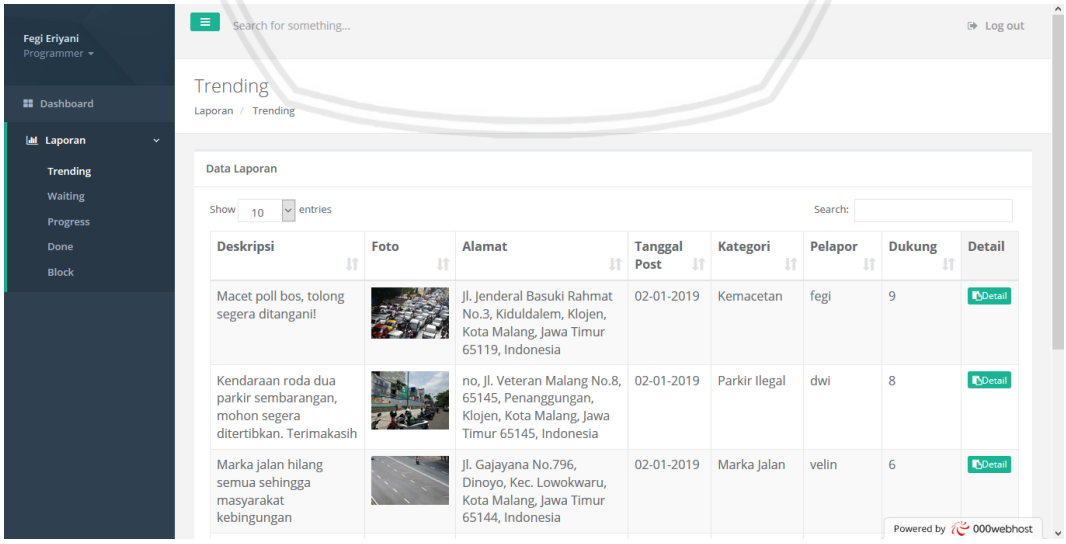
Halaman *dashboard* ini menampilkan total jumlah laporan yang masuk pada sistem. Dimana laporan-laporan yang dimaksudkan tersebut ialah menampilkan jumlah *trending* yang masuk, seluruh laporan yang masuk, jumlah laporan yang berstatus *waiting, onprogress, done* serta *block*. Implementasi antarmuka halaman *dashboard* pada website ini dapat dilihat pada Gambar 5.39.



Gambar 5. 39 Implementasi Antarmuka Halaman *Dashboard*

3. Implementasi Antarmuka Halaman *Trending* Laporan

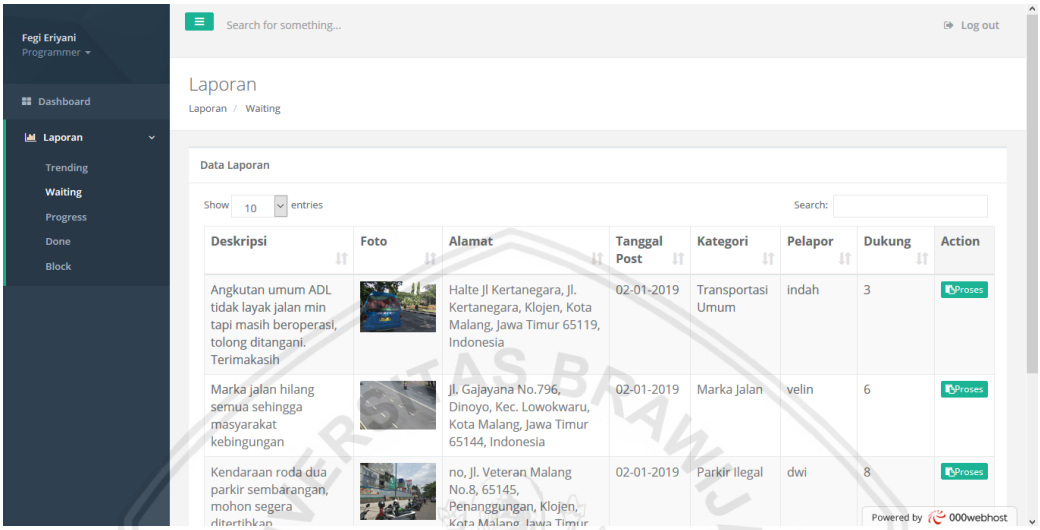
Halaman *trending* laporan untuk *administrator* ini digunakan untuk menampilkan laporan-laporan yang masuk yang menjadi *trending*. Laporan yang menjadi *trending* tersebut berdasarkan laporan yang memiliki jumlah *support* terbanyak. Gambar 5.40 menunjukkan implementasi antarmuka untuk halaman *trending* laporan.



Gambar 5. 40 Implementasi Antarmuka Halaman *Trending* Laporan

4. Implementasi Halaman Laporan Status *Waiting*

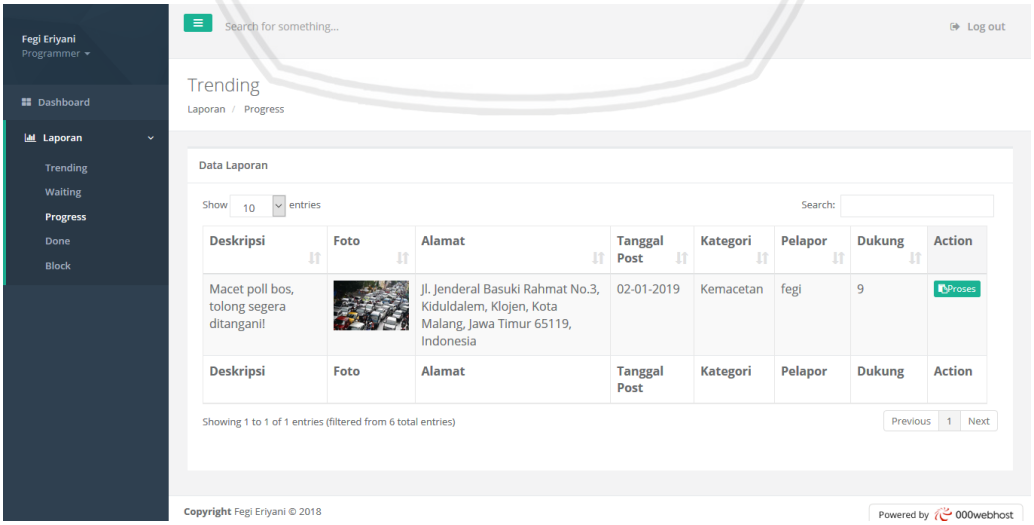
Halaman laporan dengan status *waiting* merupakan halaman yang menampilkan secara seluruh laporan yang memiliki status *waiting*. Halaman laporan status *waiting* ini memiliki halaman antarmuka sama seperti halaman *trending* laporan. Implementasi antarmuka halaman laporan status *waiting* dapat dilihat pada Gambar 5.41.



Gambar 5. 41 Implementasi Antarmuka Halaman Laporan Status *Waiting*

5. Implementasi Antarmuka Halaman Laporan Status *OnProgress*

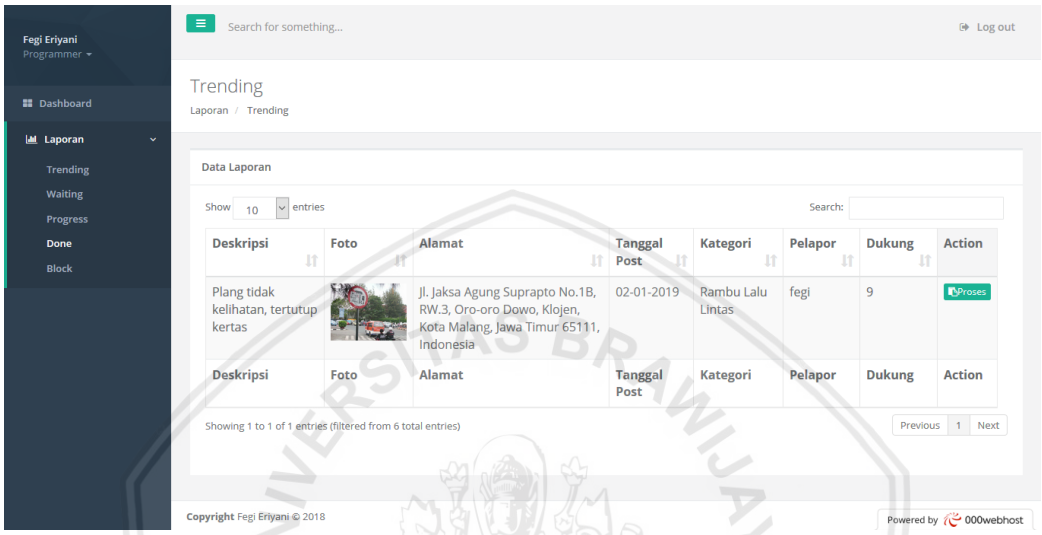
Halaman laporan dengan status *onprogress* merupakan halaman yang menampilkan secara keseluruhan laporan yang memiliki status *onprogress*. Pada halaman ini terdapat daftar laporan berstatus *onprogress* mulai dari deskripsi, foto, alamat, tanggal posting, kategori keluhan, status laporan, nama pelapor, dan juga jumlah dukung. Implementasi antarmuka halaman laporan status onprogress dapat dilihat pada Gambar 5.42.



Gambar 5. 42 Implementasi Antarmuka Halaman Laporan Status *OnProgress*

6. Implementasi Antarmuka Halaman Laporan Status *Done*

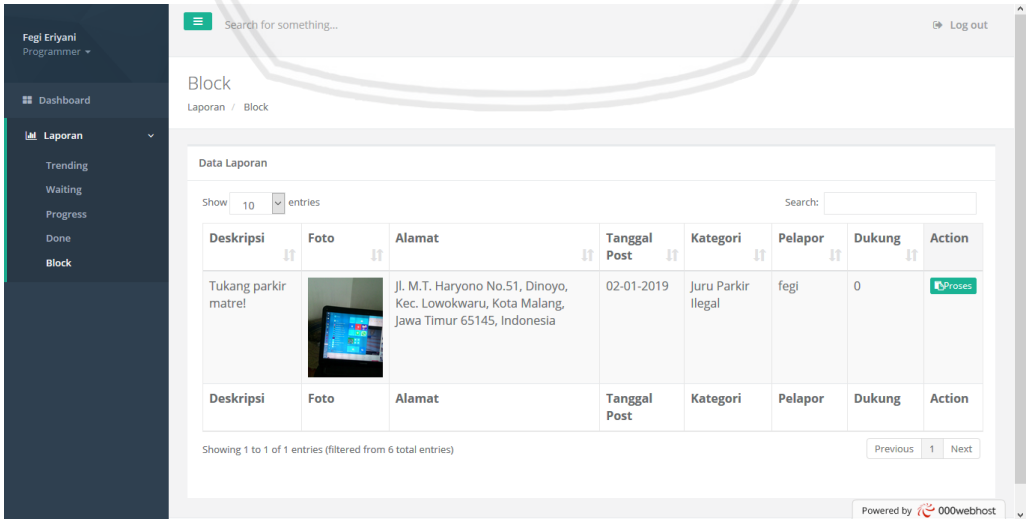
Halaman laporan dengan status *done* merupakan halaman yang menampilkan laporan yang memiliki status *done*. Pada halaman laporan status *done* ini terdapat daftar laporan berstatus *done* mulai dari deskripsi, foto laporan, alamat kejadian, tanggal posting, kategori keluhan, status dari laporan, nama pelapor, dan juga jumlah dukung yang didapatkan. Implementasi antarmuka halaman laporan status *done* dapat dilihat pada Gambar 5.43.



Gambar 5. 43 Implementasi Antarmuka Halaman Laporan Status *Done*

7. Implementasi Antarmuka Halaman Laporan Status *Block*

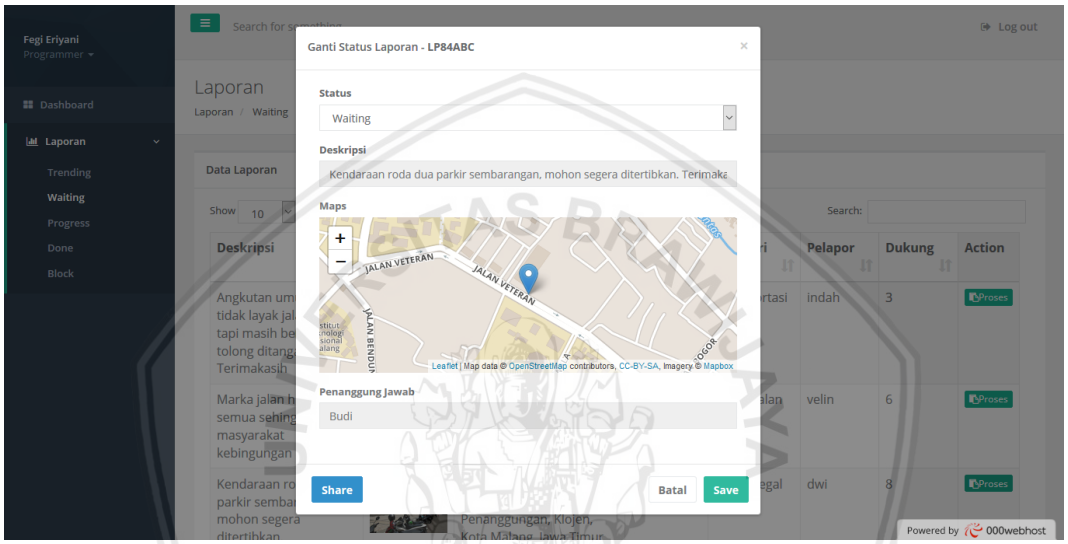
Sama halnya seperti halaman laporan status lainnya, halaman laporan status *block* ini terdiri dari deskripsi, foto laporan, alamat kejadian, tanggal posting, kategori keluhan, status dari laporan, nama pelapor, dan juga jumlah dukung. Implementasi antarmuka halaman laporan status *block* dapat dilihat pada Gambar 5.44.



Gambar 5. 44 Implementasi Antarmuka Halaman Laporan Status *Block*

8. Implementasi Antarmuka Halaman Ganti Status Laporan

Halaman ganti status laporan merupakan halaman yang digunakan untuk mengganti status laporan yang masuk pada sistem. Pada halaman ganti status laporan ini berisi status yang akan diganti oleh admin, deskripsi laporan, lokasi pelaporan, dan juga nama penanggung jawab petugas yang menangani permasalahan. Pada halaman ini juga terdapat *botton share* yang digunakan untuk membagikan alamat lokasi pelaporan untuk dibagikan kepada penanggung jawab yang sesuai dengan bidangnya serta terdapat *botton save* yang digunakan untuk menyimpan perubahan yang terjadi. Implementasi antarmuka halaman ganti status laporan dapat dilihat pada Gambar 5.45.



Gambar 5. 45 Implementasi Antarmuka Halaman Ganti Status Laporan

BAB 6 PENGUJIAN SISTEM

Pengujian sistem merupakan tahap yang dilakukan ketika tahap implementasi selesai dilakukan. Bab pengujian sistem ini mengacu kepada peracangan dan implementasi aplikasi pengaduan masyarakat yang telah dibuat. Tujuan dari melakukan sebuah pengujian sistem yakni untuk memastikan implementasi yang telah dibuat sesuai dengan analisis kebutuhan yang diperlukan dan rancangan sistem yang terdapat pada tahap perancangan sistem, selain itu dengan adanya pengujian ini dilakukan untuk menemukan *bug* atau *error* yang terjadi pada sistem. Pengujian pada aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang menggunakan pengujian *whitebox*, pengujian *blackbox, dan* pengujian *usability,* untuk pengujian *whitebox* yakni dengan menggunakan pengujian *unit* dan pengujian integrasi, dan pengujian *blackbox* menggunakan pengujian validasi. Hasil pengujian yang sudah dilakukan kemudian mendapatkan hasil yang sesuai dengan kebutuhan fungsional, maka sistem layak digunakan dan dapat memenuhi kebutuhan *software*.

6.1 Pengujian *Unit*

Pengujian *unit* merupakan salah satu pengujian dari metode pengujian *whitebox* dengan menggunakan *basis path testing* sebagai teknik yang digunakannya. Pengujian *unit* berfokus pada *unit* atau komponen dari tiap *unit* pada *software* yang telah diimplementasikan. *Basis path testing* ialah metode yang digunakan untuk mengecek jalur algoritme untu mendapatkan *independent path* dan metode *cyclomatic complexity* untuk mengetahui tingkat kerumitan sistem. *Basis path testing* ini dilakukan mengacu kepada perancangan komponen berupa algoritme yang telah dibuat pada bab sebelumnya. Pada pengujian *basis path* dilakukan untuk memastikan agar tiap jalur setidaknya hanya dilakukan satu kali. Berikut merupakan tahapan untuk melakukan *basis path testing* untuk mendapatakan *test case* pada pengujian ini, yakni sebagai berikut:

1. Berdasarkan perancangan berupa algoritme yang telah dibuat pada tahap perancangan maka dapat dilakukan pembuatan *flow graph* mengacu pada algoritme tersebut.
2. Setelah *flow graph* telah dibuat selanjutnya menentukan *cyclomatic complexity* mengacu *flow graph* yang telah dibuat.
3. Ketika *cyclomatic complexity* telah ditentukan selanjutnya dapat menentukan *independent path* berdasarkan *flow graph* dimana *independent path* ini digunakan untuk membuat kasus uji.

6.1.1 Pengujian *Unit* pada *Method* "upload()" dari *Class* "*Upload*Activity"

Tabel 6.1 merupakan *pseudocode* yang sudah dirancang pada bab perancangan yang mengacu pada implementasi kode program pada Tabel 5.34 yang digunakan untuk pengujian.

Tabel 6. 1 *Pseudocode Method* "upload()" dari *Class* "*Upload*Activity"

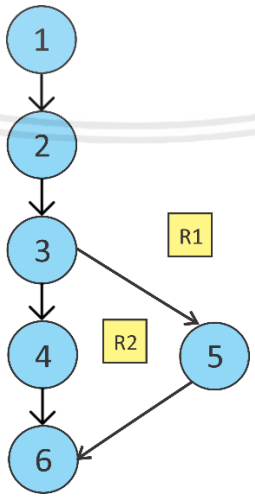
Node	*Pseudocode*
1	`implementation method upload()`
1	`  getdes to string();`
1	`  foto convert to string();`
1	`  getalamat to string();`
1	`  setdate tgl_post();`
1	`  getid pelapor from UserData();` **1**
2	`  addLaporanService = implementation method addLaporan with parameter des, foto, lat, lng, alamat, tgl_post, status, pelapor, kategori` **2**
3	`  if (!call = null && !response = null)` **3**
4	`  BaseResponse send response.body();`
4	`  start activity with main page();` **4**
5	`  else` **5**
	`  show message "Terjadi Kesalahan, Silahkan Ulangi Kembali!"`
6	`  endif` **6**

6.1.1.1 Basis Path Testing

a) *Flow Graph*
Flow graph untuk pengujian *unit* ini terdapat pada Gambar 6.1.

b) *Cyclomatic Complexity* (V(G))
- V(G) = jumlah *region* = 2
- V(G) = jumlah *edge* - jumlah *node* + 2 = 6 - 6 + 2 = 2
- V(G) = jumlah *predicate node* + 1 = 1 + 1 = 2

c) *Independent Path*
- Jalur 1 : 1-2-3-4-6
- Jalur 2 : 1-2-3-5-6



Gambar 6. 1 *Flow Graph Method* "upload()" dari *Class* "*Upload*Activity"

Berikut merupakan kasus uji serta hasil dari pengujian *method* "upload()" dari *Class* "*Upload*Activity" yang telah dilakukan dapat dilihat pada Tabel 6.2.

Tabel 6. 2 Hasil pengujian *Unit Method* "upload()" dari *Class* "*Upload*Activity"

No.	No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1.	1	-idUser = "2" (idUser = terdaftar) -foto = "http://sigapapp.000webhostapp.com/img/laporan/LP0D" des = "Plang tidak kelihatan, tertutup kertas" -kategori keluhan = "Rambu Lalu Lintas" -alamat = "jalan bunga kumis kucing No. 27"	Data laporan baru tersimpan dalam *database* dan ditampilkan pada halaman *main*	Data laporan baru tersimpan dalam *database* dan ditampilkan pada halaman *main*	Valid
2.	2	-Callback = *failure* -informasi *message error* ditampilkan	Tidak dapat mengakses halaman *upload* laporan dan muncul *message error*	Tidak dapat mengakses halaman *upload* laporan dan muncul *message error*	Valid

6.1.1.2 Analisis Hasil Pengujian Unit pada Method "upload()" dari Class "UploadActivity"

Analisis hasil dari pengujian *unit method* upload() dari *class Upload*Activity menghasilkan nilai *cyclomatic complexity* sebanyak 2 nilai yang mana akan menentukan jalur *independent* yang berarti memiliki 2 jalur *independent* dimana jalur tersebut dapat digunakan sebagai acuan untuk kasus uji pada pengujian *unit* ini. Kasus uji yang didapatkan pada pengujian *unit* ini berkaitan dengan menambah laporan baru dengan memasukkan semua inputan yang ada pada *form* UploadActivity dan untuk kasus uji yang ialah tidak dapat mengakses halaman UploadActivity. Untuk itu hasil pengujian yang didapatkan pada pengujian *unit* ini untuk keduanya berstatus valid.

Berdasarkan dua kasus uji yang sudah dijelaskan dan sudah dilakukan pengujian dengan menggunakan pengujian *unit* dari *method* upload() dari *class Upload*Activity maka diperoleh hasil untuk kedua kasus uji tersebut valid. Dapat dikatakan pengujian *unit* pada *method* upload() dari *class Upload*Activity melakukan pengujian terhadap salah satu fitur yang ada pada aplikasi *mobile* pengaduan masyarakat yakni menambah laporan atau menyampaikan keluhan yang dialami. Dengan demikian pengujian *unit* ini dapat menjawab pertanyaan

untuk rumusan masalah yang terdapat pada poin pertama mengenai fitur layanan dan fasilitas yang ada pada aplikasi *mobile* untuk pengaduan masyarakat tentang sarana umum lalu lintas guna membantu masyarakat untuk menyampaikan keluhan pada Dinas Perhubungan Kota Malang.

6.1.2 Pengujian *Unit* pada *Method* "save" dari *Class* "EditMyLaporanActivity"

Berikut merupakan *pseudocode* yang sudah dirancang pada bab perancangan yang mengacu pada implementasi kode program pada Tabel 5.36 yang digunakan untuk pengujian *unit*. *Pseudocode* tersebut dapat dilihat pada Tabel 6.3.

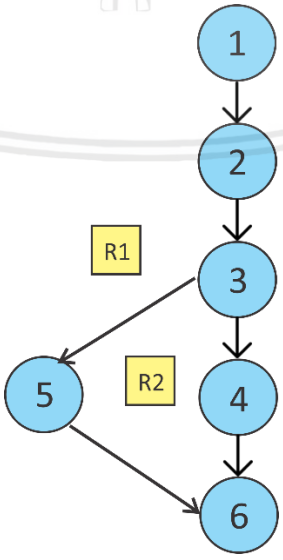
Tabel 6. 3 *Pseudocode Method* "save" dari *Class* "EditMyLaporanActivity"

Node	*Pseudocode*
1	`implementation method save()`
1	`  getdes to string();`
1	`  foto convert to string();` **1**
1	`  getalamat to string();`
2	`  editMyLaporanService = implementation method editMyLaporan with parameter id, des, fotoS, lat, lng, alamat, idKategori` **2**
3	`  if (!call = null && !response = null)` **3**
4	`  BaseResponse send response.body();` **4**
5	`  else`
5	`  show message "Terjadi Kesalahan"` **5**
6	`  endif` **6**

6.1.2.1 Basis Path Testing

a) *Flow Graph*

Flow graph untuk pengujian *unit* init terdapat pada Gambar 6.2.



Gambar 6. 2 *Flow Graph Method* "save" dari *Class* "EditMyLaporanActivity"

b) *Cyclomatic Complexity* (V(G))
   - V(G) = jumlah *region* = 2
   - V(G) = jumlah *edge* - jumlah *node* + 2 = 6 - 6 + 2 = 2
   - V(G) = jumlah *predicate node* + 1 = 1 + 1 = 2

c) *Independent Path*
   - Jalur 1 : 1-2-3-4-6
   - Jalur 2 : 1-2-3-5-6

Berikut merupakan kasus uji serta hasil dari pengujian *method* "save" dari *class* "EditMyLaporanActivity" yang telah dilakukan berdasarkan *independent path* yang dapat dilihat pada Tabel 6.4.

Tabel 6. 4 Hasil pengujian *Unit Method* "save" dari *Class* "EditMyLaporanActivity"

No.	No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1.	1	-mengubah deskripsi -idUser = "2" (idUser = terdaftar) -des = "Plang tidak terlihat membuat pengendara tidak mengetahui plang tersebut" -kategori keluhan = "Rambu Lalu Lintas"	Data laporan yang sudah diubah tersimpan dalam *database*	Data laporan yang sudah diubah tersimpan dalam *database*	Valid
2.	2	-Callback = *failure* -informasi message *error* ditampilkan	Tidak dapat mengakses halaman edit laporan dan muncul *message error*	Tidak dapat mengakses halaman edit laporan dan muncul *message error*	Valid

6.1.2.2 Analisis Hasil Pengujian Unit pada Method "save" dari Class "EditMyLaporanActivity"

Analisis hasil dari pengujian *unit method* save() dari *class* EditMyLaporanActivity menghasilkan nilai *cyclomatic complexity* sebanyak 2 yang mana akan menentukan jalur *independent* yang dapat digunakan sebagai acuan untuk kasus uji pada pengujian *unit* ini. Kasus uji yang didapatkan pada pengujian *unit* ini berkaitan dengan mengubah laporan pribadi dimana kasus uji pertama pelapor mengubah deskripsi dari laporan yang sebelumnya dan hasilnya berhasil dilakukan dan kasus uji yang kedua pelapor tidak dapat mengakses halaman *edit* laporan. Setelah dilakukan pengujian pada kasus uji tersebut maka hasil pengujian yang didapatkan pada pengujian *unit* ini berstatus valid.

Berdasarkan dua kasus uji yang sudah dijelaskan dan sudah dilakukan pengujian dengan menggunakan pengujian *unit* dari *method* save() dari *class* EditMyLaporanActivity maka diperoleh hasil untuk kedua kasus uji tersebut valid. Dapat dikatakan pengujian *unit* pada *method* save() dari *class* EditMyLapoanActivity melakukan pengujian terhadap salah satu fitur yang ada pada aplikasi *mobile* pengaduan masyarakat yakni fitur untuk mengubah laporan pribadi. Dengan demikian pengujian *unit* ini dapat menjawab pertanyaan untuk rumusan masalah yang terdapat pada poin pertama mengenai fitur layanan dan fasilitas yang ada pada aplikasi *mobile* untuk pengaduan masyarakat tentang sarana umum lalu lintas guna membantu masyarakat untuk menyampaikan keluhan pada Dinas Perhubungan Kota Malang. Karena mengubah laporan pribadi merupakan salah satu fitur yang ada pada aplikasi pengaduan masyarakat ini.

6.1.3 Pengujian *Unit* pada *Method* "*editStatus*()" dari *Class* "WaitingController"

Berikut merupakan *pseudocode* yang sudah dirancang pada bab perancangan yang mengacu pada implementasi kode program pada Tabel 5.38 yang digunakan untuk pengujian *unit*. *Pseudocode* tersebut dapat dilihat pada Tabel 6.5.

Tabel 6. 5 *Pseudocode Method* "editStatus()" dari *Class* "WaitingController"

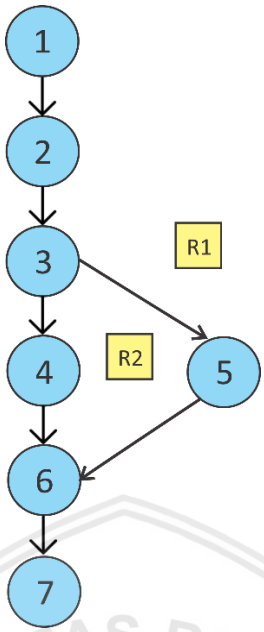
Node	*Pseudocode*
1	implementasi method editStatus()
1	request id; **1**
1	request status;
2	select laporan from model Laporan; **2**
3	if(laporans != null){ **3**
4	show message "Data Berhasil di Proses"; **4**
4	response id;
4	response status;
5	else
5	show message "Data Tidak Berhasil di Proses"; **5**
6	endif **6**
7	endif **7**

6.1.3.1 Basis Path Testing

a) *Flow Graph*
Flow graph untuk pengujian *unit* ini terdapat pada Gambar 6.3.

b) *Cyclomatic Complexity* (V(G))
- V(G) = jumlah *region* = 2
- V(G) = jumlah *edge* - jumlah *node* + 2 = 7 - 7 + 2 = 2
- V(G) = jumlah *predicate node* + 1 = 1 + 1 = 2

c) *Independent Path*
- Jalur 1 : 1-2-3-4-6-7
- Jalur 2 : 1-2-3-5-6-7



Gambar 6. 3 *Flow Graph Method* "editStatus()" dari *Class* "WaitingController"

Berikut merupakan kasus uji serta hasil dari pengujian *method* "editStatus()" dari *class* "WaitingController" yang telah dilakukan dapat dilihat pada Tabel 6.6.

Tabel 6. 6 Hasil Pengujian *Unit Method* "editStatus()" dari *Class* "WaitingController"

No.	No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1.	1	-mengubah status *waiting* -idUser = "2" (idUser = terdaftar) -status = "*OnProgress*"	Status laporan yang sudah diubah tersimpan dalam *database*	Status laporan yang sudah diubah tersimpan dalam *database*	Valid
2.	2	-Callback = *failure* -informasi message *error* ditampilkan	Tidak dapat mengakses halaman ganti status laporan dan muncul *message error*	Tidak dapat mengakses halaman ganti status laporan dan muncul *message error*	Valid

6.1.3.2 Analisis Hasil Pengujian Unit pada Method "editStatus()" dari Class "WaitingController"

Analisis hasil dari pengujian *unit method* editStatus() dari *class* WaitingController menghasilkan nilai *cyclomatic complexity* sebanyak 2 yang mana akan menentukan jalur *independent*, pada pengujian *unit* ini menghasilkan 2 jalur *independent* dimana jalur *independent* tersebut dapat digunakan sebagai acuan untuk kasus uji pada pengujian *unit* ini. Setiap jalur *independent* tersebut

memiliki kasus uji yang digunakan untuk memastikan agar setiap jalur dapat dijalankan dengan benar. Kasus uji yang pertama ialah melakukan perubahan pada status laporan waiting yang diubah menjadi status laporan *onprogress* dan setelah berhasil dilakukan akan muncul pesan "Data Berhasil di Proses". Untuk itu hasil pengujian yang didapatkan pada pengujian *unit* ini untuk keduanya berstatus valid.

Kasus uji yang sudah dijelaskan dengan menggunakan pengujian *unit* dari *method* editStatus() dari *class* WaitingController maka diperoleh hasil untuk kedua kasus uji tersebut valid. Dapat dikatakan pengujian *unit* pada *method* editStatus() dari *class* WaitingController melakukan pengujian terhadap salah satu fitur yang ada pada aplikasi *mobile* pengaduan masyarakat yakni *method* tersebut dibangun untuk mengganti status laporan oleh admin. Dengan demikian pengujian *unit* ini dapat menjawab pertanyaan untuk rumusan masalah yang terdapat pada poin pertama mengenai fitur layanan dan fasilitas yang ada pada aplikasi *mobile* untuk pengaduan masyarakat tentang sarana umum lalu lintas guna membantu masyarakat untuk menyampaikan keluhan pada Dinas Perhubungan Kota Malang.

6.2 Pengujian Integrasi

Pengujian Integrasi merupakan pengujian yang dilakukan dengan menggunakan metode *whitebox testing*. Pengujian integrasi dilakukan pada arsitektur sebuah *software* dimana pengujian ini berkonsentrasi pada konstruksi dan desain perangkat lunak itu sendiri. Pengujian integrasi pada aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang terdapat pada Tabel 6.7, pada tabel tersebut berisi kasus uji yang akan dilakukan pada pengujian integrasi pada aplikasi ini.

Tabel 6. 7 Identifikasi dan Kasus Uji Aplikasi Pengaduan Masyarakat

No	Nama *Class*	Nama *Method*	Tujuan
1	UploadActivity	upload()	Menambah laporan pengaduan baru pada *database*
	AddLaporanService	addLaporan()	
2	EditMyLaporanActivity	save()	Mengubah laporan pribadi yang masih memiliki status *waiting* pada *database*
	EditMyLaporanService	editMyLaporan()	
3	PickLocationActivity	onCreate()	Memasukkan lokasi pelaporan dengan menggunakan *location based service*
	PickLocationActivity	OnMapReadyCallback()	

6.2.1 Hasil pengujian Integrasi Nomor 1

Berikut merupakan hasil dari identifikasi pada pengujian integrasi yang telah dilakukan dapat dilihat pada Tabel 6.8, pada tabel tersebut merupakan hasil dari pengujian ingerasi kasus uji nomor 1.

Tabel 6. 8 Hasil Pengujian Integrasi Nomor 1

Nomor Uji	1
***Input* Pertama**	"nama" : fegi "foto" : http://sigapapp.000 webhostapp.com/img/laporan/LP0D "kategori keluhan" : Kemacetan "deskripsi" : Macet poll, tolong segera ditangani! "alamat" : Jl. Jenderal Basuki Rahmat No.3, Kiduldalem, Klojen
***Method* dari *Class* UploadActivity**	upload()
***Output* pertama atau *input* kedua**	"nama" : fegi "foto" : http://sigapapp.000 webhostapp.com/img/laporan/LP0D "kategori keluhan" : Kemacetan "deskripsi" : Macet poll, tolong segera ditangani! "alamat" : Jl. Jenderal Basuki Rahmat No.3, Kiduldalem, Klojen
***Method* dari *Class* AddLaporanService**	addLaporan()
Expected Result	Laporan pengaduan baru terekam pada *database* sesuai dengan data laporan yang diinputkan
Result	Laporan pengaduan baru terekam pada *database* sesuai dengan data laporan yang diinputkan
Status	Valid

Analisis Hasil Pengujian Integrasi Nomor Uji 1

Analisis hasil dari pengujian integrasi ini antara *class* UploadActivity dan *class* AddLaporanService memperlihatkan jika dua kelas tersebut dapat berintegrasi dengan baik. Inputan dari *class* UploadActivity mampu diproses pada *class* AddLaporanService oleh karena itu diperoleh *output* yang sesuai dengan yang diinginkan. Pengujian yang dilakukan pada dua *class* tersebut menghasilkan status yang valid.

Berdasarkan pengujian integrasi yang sudah dilakukan pengujian dengan melakukan pengujian dari dua *class* tersebut memiliki status valid. Hubungan antar kedua *class* ini mendapatkan fitur yang diinginkan yakni fitur menambah laporan pengaduan baru dimana fitur ini merupakan salah satu fitur yang ada pada

aplikasi *mobile* pengaduan masyarakat. Dengan demikian pengujian integrasi antar dua *class* ini dapat menjawab pertanyaan untuk rumusan masalah yang terdapat pada poin pertama mengenai fitur layanan dan fasilitas yang ada pada aplikasi *mobile* untuk pengaduan masyarakat tentang sarana umum lalu lintas guna membantu masyarakat untuk menyampaikan keluhan pada Dinas Perhubungan Kota Malang.

6.2.2 Hasil Pengujian Integrasi Nomor 2

Berikut merupakan hasil dari identifikasi pada pengujian integrasi yang telah dilakukan dapat dilihat pada Tabel 6.9, pada Tabel tersebut merupakan hasil dari pengujian ingerasi kasus uji nomor 2.

Tabel 6. 9 Hasil Pengujian Integrasi Nomor 2

Nomor Uji	2
***Input* Pertama**	"nama" : fegi "foto" : http://sigapapp.000 webhostapp.com/img/laporan/LPC9 "kategori keluhan" : Parkir Ilegal "deskripsi" : parkir sembarangan "alamat" : Jl. Raya Sumbersari No.558, Ketawanggede
***Method* dari *Class* EditMyLaporanActivity**	save()
***Output* pertama atau *input* kedua**	"nama" : fegi "foto" : http://sigapapp.000 webhostapp.com/img/laporan/LPC9 "kategori keluhan" : Parkir Ilegal "deskripsi" : parkir sembarangan "alamat" : Jl. Raya Sumbersari No.558, Ketawanggede
Method* dari *Class* *EditMyLaporanService	editMyLaporan()
Expected Result	Laporan pengaduan yang sudah di ubah terekam pada *database* sesuai dengan data laporan yang di ubah
Result	Laporan pengaduan yang sudah di ubah terekam pada *database* sesuai dengan data laporan yang di ubah
Status	Valid

Analisis Hasil Pengujian Integrasi Nomor Uji 2

Analisis hasil dari pengujian integrasi ini antara *class* EditMyLaporanActivity dan *class* *EditMyLaporanService* memperlihatkan jika dua kelas tersebut dapat

berintegrasi dengan baik. Berdasarkan pengujian integrasi yang sudah dijelaskan dan sudah dilakukan pengujian dengan melakukan pengujian dari dua *class* EditMyLaporanActivity dan *class EditMyLaporanService* memiliki status valid. Hubungan antar kedua *class* ini mendapatkan fitur yang diinginkan yakni fitur mengubah laporan pribadi merupakan salah satu fitur yang ada pada aplikasi *mobile* pengaduan masyarakat. Dengan demikian pengujian integrasi antar dua *class* ini dapat menjawab pertanyaan untuk rumusan masalah yang terdapat pada poin pertama mengenai fitur layanan dan fasilitas yang ada pada aplikasi *mobile* untuk pengaduan masyarakat tentang sarana umum lalu lintas guna membantu masyarakat untuk menyampaikan keluhan pada Dinas Perhubungan Kota Malang.

6.2.3 Hasil pengujian Integrasi Nomor 3

Tabel 6.10 merupkan tabel yang berisi hasil dari identifikasi pada pengujian integrasi yang telah dilakukan, pada tabel tersebut merupakan hasil dari pengujian ingerasi kasus uji nomor 3.

Tabel 6. 10 Hasil Pengujian Integrasi Nomor 3

Nomor Uji	3
***Input* Pertama**	"alamat" : Jl. Jaksa Agung Suprapto No.1B, RW.3, Oro-oro Dowo
***Method* dari *Class* PickLocationActivity**	onCreate()
Output pertama atau *input* kedua	"alamat" : Jl. Jaksa Agung Suprapto No.1B, RW.3, Oro-oro Dowo
***Method* dari *Class* PickLocationActivity**	OnMapReadyCallback()
Expected Result	Menambahkan data lokasi baru denan menggunakan *location based service*
Result	Menambahkan data lokasi baru denan menggunakan *location based service*
Status	Valid

Analisis Hasil Pengujian Integrasi Nomor Uji 3

Analisis hasil dari pengujian integrasi dari *class* PickLocationActivity ialah memperlihatkan jika *method* yang ada pada *class* yang sama dapat berintegrasi dengan baik. Berdasarkan pengujian integrasi yang sudah dilakukan pengujian dengan melakukan pengujian dari *class* yang sama pada *class* PickLocationActivity yang memiliki status valid. *Class* PickLocationActivity digunakan untuk menambah lokasi baru yang ada kaitannya dengan *location based service*. Dengan demikian pengujian integrasi *class* PickLocationActivity ini dapat menjawab pertanyaan untuk rumusan masalah yang terdapat pada poin kedua mengenai implementasi

fitur *location based service* pada aplikasi *mobile* untuk pengaduan masyarakat guna membantu masyarakat untuk menyampaikan keluhan pada Dinas Perhubungan Kota Malang.

6.3 Pengujian Validasi

Pengujian validasi merupakan salah satu pengujian dari *blackbox testing*. Pengujian ini mengecek semua kebutuhan-kebutuhan baik kebutuhan fungsional maupun kebutuhan non fungsional yang sudah didefinisikan apakah sudah berjalan pada sistem yang telah dibuat. Keberhasilan pengujian validasi yakni dapat dilihat dari sistem yang telah dibangun mampu memenuhi keseluruhan kebutuhan sesuai dengan alur *scenario* yang sudah dibuat.

6.3.1 Pengujian Validasi *Register*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional *register* yang ada pada aplikasi pengaduan masyarakat berdasarkan *scenario* yang telah dirancang, pengujian tersebut dapat ditunjukkan pada Tabel 6.11.

Tabel 6. 11 Pengujian Validasi *Register*

Kode Kebutuhan	SG-1-001
Nama Kasus Uji	*Register*
Prosedur	1. Mengakses halaman *Register* 2. Memasukkan nama, email, *password* pada halaman *Register*, kemudian menekan tombol "*Register*"
Expected Result	Sistem menampilkan *message* proses *register* berhasil dan mengarahkan pengguna masuk ke halaman *login*
Result	Sistem menampilkan *message* proses *register* berhasil dan mengarahkan pengguna masuk ke halaman *login*
Status	Valid

Analisis Hasil Pengujian Validasi *Register*

Analisis hasil dari pengujian validasi *register* ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji *register*. Pengujian tersebut digunakan untuk memastikan fitur *register* ini dapat berjalan dengan baik pada aplikasi pengaduan masyarakat.

6.3.2 Pengujian Validasi *Login*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional *login* yang ada pada aplikasi pengaduan masyarakat, pengujian tersebut dapat ditunjukkan pada Tabel 6.12.

Tabel 6. 12 Pengujian Validasi *Login*

Kode Kebutuhan	SG-1-002
Nama Kasus Uji	*Login*
Prosedur	1. Mengakses halaman *Login* 2. Memasukkan *email* dan *password* pada halaman *Login*, kemudian menekan tombol "*Login*"
Expected Result	Sistem menampilkan *message* jika proses *login* berhasil dan mengarahkan pengguna masuk ke halaman *home*
Result	Sistem menampilkan *message* jika proses *login* berhasil dan mengarahkan pengguna masuk ke halaman *home*
Status	Valid

Analisis Hasil Pengujian Validasi *Login*

Analisis hasil dari pengujian validasi *login* ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji *login*. Pengujian tersebut digunakan untuk memastikan fitur *login* ini dapat berjalan dengan baik pada aplikasi *mobile* yang digunakan oleh pelapor.

6.3.3 Pengujian Validasi Melihat Daftar Laporan

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat daftar laporan yang ada pada aplikasi pengaduan masyarakat, pengujian tersebut dapat ditunjukkan pada Tabel 6.13.

Tabel 6. 13 Pengujian Validasi Melihat Daftar Laporan

Kode Kebutuhan	SG-2-001
Nama Kasus Uji	Melihat Daftar Laporan
Prosedur	1. Mengakses halaman *home* atau halaman daftar laporan yang berisi semua laporan yang dilaporkan
Expected Result	Sistem menampilkan halaman *home* dan menampilkan daftar laporan yang ada berupa nama pelapor, tanggal post, foto laporan, deskripsi laporan, dan juga jumlah dukungan terhadap laporan
Result	Sistem menampilkan halaman *home* dan menampilkan daftar laporan yang ada berupa nama pelapor, tanggal post, foto laporan, deskripsi laporan, dan juga jumlah dukungan terhadap laporan
Status	Valid

Analisis Hasil Pengujian Validasi Melihat Daftar Laporan

Analisis hasil dari pengujian validasi melihat daftar laporan ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji melihat daftar laporan. Pengujian tersebut digunakan untuk memastikan fitur melihat daftar laporan ini dapat berjalan dengan baik pada aplikasi *mobile*.

6.3.4 Pengujian Validasi Menambah Laporan

Pengujian validasi menambah laporan yang dilakukan untuk kebutuhan fungsional menambah laporan yang ada pada aplikasi ini yang dibuat berdasarkan pada *use case scenario*. Pengujian validasi menambah laporan ini dapat dilihat pada Tabel 6.14.

Tabel 6. 14 Pengujian Validasi Menambah Laporan

Kode Kebutuhan	SG-2-002
Nama Kasus Uji	Menambah Laporan
Prosedur	1. Mengakses halaman upload laporan 2. Aktor sudah mengambil foto laporan
Expected Result	Sistem menambahkan laporan baru yang berisi foto laporan, kategori keluhan, deskripsi laporan, dan lokasi kejadian
Result	Sistem menambahkan laporan baru yang berisi foto laporan, kategori keluhan, deskripsi laporan, dan lokasi kejadian
Status	Valid

Analisis Hasil Pengujian Validasi Menambah Laporan

Analisis hasil dari pengujian validasi untuk menambah laporan baru ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji menambah laporan. Pengujian tersebut digunakan untuk memastikan fitur menambah laporan baru dapat berjalan dengan baik pada aplikasi *mobile*.

6.3.5 Pengujian Validasi Memilih Fitur *Camera*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional memilih camera untuk mengambil gambar yang digunakan untuk melaporkan, pengujian tersebut dapat ditunjukkan pada Tabel 6.15.

Tabel 6. 15 Pengujian Validasi Memilih Fitur *Camera*

Kode Kebutuhan	SG-2-003
Nama Kasus Uji	Memilih Fitur *Camera*
Prosedur	1. Aktor mengakses menu tambah laporan

	2. Aktor memilih fitur *Camera*
Expected Result	Sistem membuka *camera smartphone* pengguna untuk mengambil foto
Result	Sistem membuka *camera smartphone* pengguna untuk mengambil foto
Status	Valid

Analisis Hasil Pengujian Validasi Memilih Fitur *Camera*

Analisis hasil dari pengujian validasi memilih fitur *camera* ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji memilih fitur *camera*. Pengujian tersebut digunakan untuk memastikan fitur memilih fitur *camera* ini dapat berjalan dengan baik pada aplikasi *mobile* yang digunakan oleh pelapor.

6.3.6 Pengujian Validasi Memilih Fitur *Gallery*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional memilih fitur *gallery* untuk mengambil gambar, pengujian tersebut dapat ditunjukkan pada Tabel 6.16.

Tabel 6. 16 Pengujian Validasi Memilih Fitur *Gallery*

Kode Kebutuhan	SG-2-004
Nama Kasus Uji	Memilih Fitur *Gallery*
Prosedur	1. Aktor mengakses menu tambah laporan 2. Aktor memilih fitur *Gallery*
Expected Result	Sistem membuka *gallery smartphone* pengguna untuk mengambil foto
Result	Sistem membuka *gallery smartphone* pengguna untuk mengambil foto
Status	Valid

Analisis Hasil Pengujian Validasi Memilih Fitur *Gallery*

Analisis hasil dari pengujian validasi memilih fitur *gallery* ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji memilih fitur *gallery*. Pengujian tersebut digunakan untuk memastikan fitur memilih fitur *gallery* ini dapat berjalan dengan baik pada aplikasi *mobile* yang digunakan oleh pelapor.

6.3.7 Pengujian Validasi Mengubah Laporan

Tabel 6.17 merupakan tabel untuk pengujian validasi mengubah laporan pribadi. Mengubah laporan merupakan salah satu kebutuhan fungsional yang ada pada aplikasi pengaduan masyarakat.

Tabel 6. 17 Pengujian Validasi Mengubah Laporan

Kode Kebutuhan	SG-2-005
Nama Kasus Uji	Mengubah Laporan
Prosedur	1. Aktor memilih laporan yang akan diubah 2. Aktor mengakses halaman "Edit My Laporan"
Expected Result	Sistem mengubah laporan pribadi dan tersimpan dalam *database*
Result	Sistem mengubah laporan pribadi dan tersimpan dalam *database*
Status	Valid

Analisis Hasil Pengujian Validasi Mengubah Laporan

Analisis hasil dari pengujian validasi mengubah laporan pribadi menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur. Pengujian tersebut digunakan untuk memastikan fitur mengubah laporan dapat berjalan dengan baik pada aplikasi *mobile*.

6.3.8 Pengujian Validasi Lihat *Detail* Laporan

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional lihat detail dari laporan yang ada pada aplikasi pengaduan masyarakat, pengujian tersebut dapat ditunjukkan pada Tabel 6.18.

Tabel 6. 18 Pengujian Validasi Lihat *Detail* Laporan

Kode Kebutuhan	SG-2-006
Nama Kasus Uji	Lihat *Detail* Laporan
Prosedur	*1.* Mengakses halaman *home* 2. Aktor mengakses foto laporan 3. Aktor mengakses *Detail* laporan
Expected Result	Sistem menampilkan foto laporan serta menampilkan melihat laporan yang berisi foto laporan, deskripsi secara *detail, support*, dan lokasi laporan
Result	Sistem menampilkan foto laporan serta menampilkan melihat laporan yang berisi foto laporan, deskripsi secara *detail, support*, dan lokasi laporan
Status	Valid

Analisis Hasil Pengujian Validasi Lihat *Detail* Laporan

Analisis hasil dari pengujian validasi lihat *detail* laporan ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji lihat *detail*

laporan. Pengujian tersebut digunakan untuk memastikan fitur lihat *detail* laporan ini dapat berjalan dengan baik pada aplikasi *mobile* pengaduan masyarakat.

6.3.9 Pengujian Validasi *Support* Keluhan

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melakukan *support* keluhan terhadap laporan yang ada pada aplikasi pengaduan masyarakat, pengujian tersebut dapat ditunjukkan pada Tabel 6.19.

Tabel 6. 19 Pengujian Validasi *Support* Keluhan

Kode Kebutuhan	SG-2-007
Nama Kasus Uji	*Support* Keluhan
Prosedur	1. Aktor mengakses menu daftar laporan 2. Aktor memilih laporan 3. Aktor melakukan *Support* keluhan pada laporan
Expected Result	Sistem menampilkan penambahan *support* keluhan pada laporan yang dipilih
Result	Sistem menampilkan penambahan *support* keluhan pada laporan yang dipilih
Status	Valid

Analisis Hasil Pengujian Validasi *Support* Keluhan

Analisis hasil dari pengujian validasi *support* keluhan ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji *support* keluhan. Pengujian tersebut digunakan untuk memastikan fitur *support* keluhan ini dapat berjalan dengan baik pada aplikasi *mobile* pengaduan masyarakat.

6.3.10 Pengujian Validasi Melihat *Profile*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat *profile* atau biodata pribadi berdasarkan *scenario* yang telah dirancang sebelumnya, pengujian tersebut dapat ditunjukkan pada Tabel 6.20.

Tabel 6. 20 Pengujian Validasi Melihat *Profile*

Kode Kebutuhan	SG-2-008
Nama Kasus Uji	Melihat *Profile*
Prosedur	1. Aktor mengakses menu *Profile*
Expected Result	Sistem menampilkan menu *Profile*
Result	Sistem menampilkan menu *Profile*
Status	Valid

Analisis Hasil Pengujian Validasi Melihat *Profile*

Analisis hasil dari pengujian validasi melihat *profile* ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji melihat *profile*. Pengujian tersebut digunakan untuk memastikan fitur melihat *profile* ini dapat berjalan dengan baik pada aplikasi *mobile* pengaduan masyarakat.

6.3.11 Pengujian Validasi Mengubah *Profile*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional mengubah *profile* yang ada pada aplikasi pengaduan masyarakat, pengujian tersebut dapat ditunjukkan pada Tabel 6.21.

Tabel 6. 21 Pengujian Validasi Mengubah *Profile*

Kode Kebutuhan	SG-2-009
Nama Kasus Uji	Mengubah *Profile*
Prosedur	1. Aktor mengakses menu *Edit* pada halaman Menu *Profile* 2. Pelapor melakukan perubahan pada *Profile* yang berisi *username, password*, tanggal lahir, dan alamat kemudian menekan tombol *Update*
Expected Result	Sistem menampilkan pesan bahwa *profile* pelapor berhasil dirubah dan menampilkan halaman *profile*
Result	Sistem menampilkan pesan bahwa *profile* pelapor berhasil dirubah dan menampilkan halaman *profile*
Status	Valid

Analisis Hasil Pengujian Validasi Mengubah *Profile*

Analisis hasil dari pengujian validasi mengubah *profile* ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji mengubah *profile*. Pengujian tersebut digunakan untuk memastikan fitur mengubah *profile* ini dapat berjalan dengan baik pada aplikasi *mobile*.

6.3.12 Pengujian Validasi Melihat Laporan Pribadi

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat laporan pribadi yang pernah dilaporkan pada aplikasi pengaduan masyarakat, pengujian tersebut dapat ditunjukkan pada Tabel 6.22.

Tabel 6. 22 Pengujian Validasi Melihat Laporan Pribadi

Kode Kebutuhan	SG-2-010
Nama Kasus Uji	Melihat Laporan Pribadi
Prosedur	1. Aktor mengakses menu *Profile*

	2. Aktor mengakses *tab* menu laporan
Expected Result	Sistem menampilkan daftar laporan pribadi yang pernah dilaporkan
Result	Sistem menampilkan daftar laporan pribadi yang pernah dilaporkan
Status	Valid

Analisis Hasil Pengujian Validasi Melihat Laporan Pribadi

Analisis hasil dari pengujian validasi melihat laporan pribadi ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji melihat laporan pribadi. Pengujian tersebut digunakan untuk memastikan fitur melihat laporan pribadi ini dapat berjalan dengan baik pada aplikasi *mobile* pengaduan masyarakat.

6.3.13 Pengujian Validasi Melihat Daftar Dukungan

Tabel 6.23 merupakan tabel untuk pengujian validasi melihat daftar dukungan yang telah didukung. Pengujian validasi tersebut dapat dilihat seperti berikut.

Tabel 6. 23 Pengujian Validasi Daftar Dukungan

Kode Kebutuhan	SG-2-011
Nama Kasus Uji	Melihat Daftar Dukungan
Prosedur	1. Aktor mengakses halaman “Daftar Dukungan”
Expected Result	Sistem menampilkan daftar dukungan laporan yang pernah dilakukan
Result	Sistem menampilkan daftar dukungan laporan yang pernah dilakukan
Status	Valid

Analisis Hasil Pengujian Validasi Melihat Daftar Dukungan

Analisis hasil dari pengujian validasi melihat daftar dukungan menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur. Pengujian tersebut digunakan untuk memastikan fitur melihat daftar dukungan dapat berjalan dengan baik pada aplikasi *mobile*.

6.3.14 Pengujian Validasi Melihat *Trending* Laporan

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat *trending* laporan yang ada pada aplikasi pengaduan masyarakat, pengujian tersebut dapat ditunjukkan pada Tabel 6.24.

Tabel 6. 24 Pengujian Validasi Melihat *Trending* Laporan

Kode Kebutuhan	SG-2-012

Nama Kasus Uji	Melihat *Trending* Laporan
Prosedur	1. Aktor mengakses menu *Trending* laporan
Expected Result	Sistem menampilkan daftar *trending* laporan dengan 10 *trending* teratas
Result	Sistem menampilkan daftar *trending* laporan dengan 10 *trending* teratas
Status	Valid

Analisis Hasil Pengujian Validasi Melihat *Trending* Laporan

Analisis hasil dari pengujian validasi melihat *trending* laporan ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji melihat *trending* laporan. Pengujian tersebut digunakan untuk memastikan fitur melihat *trending* laporan ini dapat berjalan dengan baik pada aplikasi *mobile* pengaduan masyarakat.

6.3.15 Pengujian Validasi *Logout*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional *logout* yang ada pada aplikasi pengaduan masyarakat berdasarkan *scenario* sebelumnya, pengujian tersebut dapat ditunjukkan pada Tabel 6.25.

Tabel 6. 25 Pengujian Validasi *Logout*

Kode Kebutuhan	SG-2-013
Nama Kasus Uji	*Logout*
Prosedur	1. Aktor mengakses menu *Profile* 2. Aktor mengakses tab menu biodata 3. Aktor menekan tombil *Logout*
Expected Result	Sistem keluar dari aplikasi
Result	Sistem keluar dari aplikasi
Status	Valid

Analisis Hasil Pengujian Validasi *Logout*

Analisis hasil dari pengujian validasi *logout* ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji *logout*. Pengujian tersebut digunakan untuk memastikan fitur *logout* ini dapat berjalan dengan baik pada aplikasi *mobile* pengaduan masyarakat.

6.3.16 Pengujian Validasi *Login* Admin

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional *login* yang ada pada aplikasi pengaduan masyarakat berdasarkan

scenario yang telah dirancang, pengujian tersebut dapat ditunjukkan pada Tabel 6.26.

Tabel 6. 26 Pengujian Validasi *Login* Admin

Kode Kebutuhan	SG-3-001
Nama Kasus Uji	*Login*
Prosedur	1. Aktor mengakses halaman *Login* 2. Aktor memasukkan nama, dan *password* pada halaman *Login* 3. Aktor menekan tombol *Login*
Expected Result	Sistem mengarahkan pengguna masuk ke halaman *home* jika *login* berhasil
Result	Sistem mengarahkan pengguna masuk ke halaman *home* jika *login* berhasil
Status	Valid

Analisis Hasil Pengujian Validasi *Login*

Analisis hasil dari pengujian validasi *login* ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji *login*. Pengujian tersebut digunakan untuk memastikan fitur *login* ini dapat berjalan dengan baik pada website pengaduan masyarakat yang digunakan oleh admin.

6.3.17 Pengujian Validasi Melihat Jumlah Laporan

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat jumlah laporan yang ada pada sistem pengaduan masyarakat berdasarkan *scenario* sebelumnya, pengujian tersebut dapat ditunjukkan pada Tabel 6.27.

Tabel 6. 27 Pengujian Validasi Melihat Jumlah Laporan

Kode Kebutuhan	SG-3-002
Nama Kasus Uji	Melihat Jumlah Laporan
Prosedur	Aktor mengakses halaman *dashboard*
Expected Result	Sistem menampilkan jumlah laporan yang masuk terdiri dari *trending* laporan, laporan *waiting*, laporan *onprogress*, laporan *done*, dan laporan *block*
Result	Sistem menampilkan jumlah laporan yang masuk terdiri dari *trending* laporan, laporan *waiting*, laporan *onprogress*, laporan *done*, dan laporan *block*
Status	Valid

Analisis Hasil Pengujian Validasi Melihat Jumlah Laporan

Pengujian validasi melihat jumlah laporan ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji melihat jumlah laporan. Pengujian tersebut digunakan untuk memastikan fitur melihat jumlah laporan ini dapat berjalan dengan baik pada website pengaduan masyarakat yang digunakan oleh admin.

6.3.18 Pengujian Validasi Melihat *Trending* Laporan

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat *trending* laporan yang ada pada sistem pengaduan masyarakat berdasarkan *scenario* yang telah dirancang, pengujian tersebut dapat ditunjukkan pada Tabel 6.28.

Tabel 6. 28 Pengujian Validasi Melihat *Trending* Laporan

Kode Kebutuhan	SG-3-003
Nama Kasus Uji	Melihat *Trending* Laporan
Prosedur	1. Aktor mengakses halaman laporan 2. Aktor memilih menu *trending* laporan
Expected Result	Sistem menampilkan daftar laporan yang menjadi *trending* yang memiliki jumlah *support* terbanyak
Result	Sistem menampilkan daftar laporan yang menjadi *trending* yang memiliki jumlah *support* terbanyak
Status	Valid

Analisis Hasil Pengujian Validasi Melihat *Trending* Laporan

Pengujian validasi melihat *trending* laporan ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji melihat *trending* laporan. Pengujian tersebut digunakan untuk memastikan fitur melihat *trending* laporan ini dapat berjalan dengan baik pada website pengaduan masyarakat yang digunakan oleh admin.

6.3.19 Pengujian Validasi Melihat *Detail Trending*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat *detail trending* laporan yang ada pada sistem pengaduan masyarakat berdasarkan *scenario* sebelumnya, pengujian tersebut dapat ditunjukkan pada Tabel 6.29.

Tabel 6. 29 Pengujian Validasi Melihat *Detail Trending*

Kode Kebutuhan	SG-3-004
Nama Kasus Uji	Melihat *Detail Trending*
Prosedur	1. Aktor mengakses menu *trending* laporan

	2. Aktor memilih salah satu laporan yang menjadi *trending*
Expected Result	Sistem menampilkan *detail* laporan yang menjadi *trending* seperti *detail* dari deskripsi, lokasi, status, dan nama penanggung jawab
Result	Sistem menampilkan *detail* laporan yang menjadi *trending* seperti *detail* dari deskripsi, lokasi, status, dan nama penanggung jawab
Status	Valid

Analisis Hasil Pengujian Validasi Melihat *Detail Trending*

Pengujian validasi melihat *detail trending* laporan ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji. Pengujian tersebut digunakan untuk memastikan fitur melihat *detail trending* dapat berjalan dengan baik pada website pengaduan masyarakat yang digunakan oleh admin.

6.3.20 Pengujian Validasi Membagikan Lokasi Kejadian *Trending* Laporan

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional membagikan lokasi kejadian *trending* laporan yang ada pada sistem pengaduan masyarakat berdasarkan *scenario* yang telah dirancang, pengujian tersebut dapat ditunjukkan pada Tabel 6.30.

Tabel 6. 30 Pengujian Validasi Membagikan Lokasi Kejadian *Trending* Laporan

Kode Kebutuhan	SG-3-005
Nama Kasus Uji	Membagikan Lokasi Kejadian *Trending* Laporan
Prosedur	1. Aktor menekan tombol "*Detail*" pada salah satu laporan trending 2. Aktor menekan tombol "Share" pada halaman "*Detail Trending* Laporan"
Expected Result	Sistem menampilkan pesan berupa *pop up* yang berisi deskripsi laporan dan lokasi kejadian yang sudah disalin
Result	Sistem menampilkan pesan berupa *pop up* yang berisi deskripsi laporan dan lokasi kejadian yang sudah disalin
Status	Valid

Analisis Hasil Pengujian Validasi Membagikan Lokasi Kejadian *Trending* Laporan

Pengujian validasi membagikan lokasi kejadian *trending* laporan ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji membagikan lokasi kejadian *trending*. Pengujian tersebut digunakan untuk

memastikan fitur tersebut dapat berjalan dengan baik pada website pengaduan masyarakat yang digunakan.

6.3.21 Pengujian Validasi Mencari *Trending* Laporan

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional mencari *trending* laporan yang ada pada sistem pengaduan masyarakat berdasarkan *scenario* sebelumnya, pengujian tersebut dapat ditunjukkan pada Tabel 6.31.

Tabel 6. 31 Pengujian Validasi Mencari *Trending* Laporan

Kode Kebutuhan	SG-3-006
Nama Kasus Uji	Mencari *Trending* Laporan
Prosedur	1. Mengakses halaman *trending* laporan 2. Mencari laporan yang menjadi *trending*
Expected Result	Sistem menampilkan laporan yang dicari oleh aktor
Result	Sistem menampilkan laporan yang dicari oleh aktor
Status	Valid

Analisis Hasil Pengujian Validasi Mencari *Trending* Laporan

Pengujian validasi mencari *trending* laporan ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji mencari *trending* laporan. Pengujian tersebut digunakan untuk memastikan fitur mencari laporan yang menjadi *trending* dapat berjalan dengan baik.

6.3.22 Pengujian Validasi Melihat Laporan Berdasarkan Status *Waiting*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat laporan berdasarkan status *waiting* yang ada pada sistem pengaduan masyarakat berdasarkan *scenario*, pengujian tersebut dapat ditunjukkan pada Tabel 6.32.

Tabel 6. 32 Pengujian Validasi Melihat Laporan Berdasarkan Status *Waiting*

Kode Kebutuhan	SG-3-007
Nama Kasus Uji	Melihat Laporan Berdasarkan Status *Waiting*
Prosedur	1. Aktor mengakses halaman laporan 2. Aktor memilih menu laporan *waiting*
Expected Result	Sistem menampilkan daftar laporan yang berstatus *waiting*
Result	Sistem menampilkan daftar laporan yang berstatus *waiting*
Status	Valid

Analisis Hasil Pengujian Validasi Melihat Laporan Berdasarkan Status *Waiting*

Pengujian validasi melihat laporan berstatus *waiting* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji melihat laporan berdasarkan status *waiting*. Pengujian ini digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik pada website.

6.3.23 Pengujian Validasi Melihat *Detail* Laporan Status *Waiting*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat *detail* laporan berstatus *waiting* yang ada pada sistem pengaduan masyarakat, pengujian tersebut dapat ditunjukkan pada Tabel 6.33.

Tabel 6. 33 Pengujian Validasi Melihat *Detail* Laporan Status *Waiting*

Kode Kebutuhan	SG-3-008
Nama Kasus Uji	Melihat *Detail* Laporan Status *Waiting*
Prosedur	1. Aktor mengakses menu laporan *waiting* 2. Aktor memilih salah satu laporan yang berstatus *waiting*
Expected Result	Sistem menampilkan *detail* laporan yang berstatus *waiting*
Result	Sistem menampilkan *detail* laporan yang berstatus *waiting*
Status	Valid

Analisis Hasil Pengujian Validasi Melihat *Detail* Laporan Status *Waiting*

Pengujian validasi melihat *detail* laporan status *waiting* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji melihat *detail* laporan berstatus *waiting*. Pengujian digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik pada website pengaduan masyarakat yang digunakan.

6.3.24 Pengujian Validasi Mengganti Status Laporan *Waiting*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional mengganti status laporan berstatus *waiting* yang ada pada sistem berdasarkan *scenario* yang telah dirancang sebelumnya, pengujian tersebut dapat ditunjukkan pada Tabel 6.34.

Tabel 6. 34 Pengujian Validasi Mengganti Status Laporan *Waiting*

Kode Kebutuhan	SG-3-009
Nama Kasus Uji	Mengganti Status Laporan *Waiting*
Prosedur	*1.* Aktor menekan tombol “Proses” pada salah satu laporan berstatus *waiting* 2. Aktor mengganti status laporan
Expected Result	Sistem menampilkan status laporan yang sudah diganti

Result	Sistem menampilkan status laporan yang sudah diganti
Status	Valid

Analisis Hasil Pengujian Validasi Mengganti Laporan Status *Waiting*

Pengujian validasi mengganti laporan status *waiting* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji. Pengujian digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik pada website pengaduan masyarakat yang digunakan.

6.3.25 Pengujian Validasi Membagikan Lokasi Kejadian Status *Waiting*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional membagikan lokasi kejadian status *waiting* yang ada pada sistem pengaduan masyarakat, pengujian tersebut dapat ditunjukkan pada Tabel 6.35.

Tabel 6. 35 Pengujian Validasi Membagikan Lokasi Kejadian Status *Waiting*

Kode Kebutuhan	SG-3-010
Nama Kasus Uji	Membagikan Lokasi Kejadian Status *Waiting*
Prosedur	1. Aktor menekan tombol "Proses" pada salah satu laporan berstatus *waiting* 2. Aktor menekan tombol "Share" pada halaman "Detail Laporan *Waiting*"
Expected Result	Sistem menampilkan pesan berupa *pop up* yang berisi deskripsi laporan dan lokasi kejadian yang sudah disalin
Result	Sistem menampilkan pesan berupa *pop up* yang berisi deskripsi laporan dan lokasi kejadian yang sudah disalin
Status	Valid

Analisis Hasil Pengujian Validasi Membagikan Lokasi Kejadian Status *Waiting*

Pengujian validasi membagikan lokasi kejadian status *waiting* ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji membagikan lokasi kejadian status *waiting*. Pengujian tersebut digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik.

6.3.26 Pengujian Validasi Mencari Laporan Status *Waiting*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional mencari laporan berstatus *waiting* yang ada pada sistem pengaduan masyarakat berdasarkan *scenario* sebelumnya, pengujian tersebut dapat ditunjukkan pada Tabel 6.36.

Tabel 6. 36 Pengujian Validasi Mencari Laporan Status *Waiting*

Kode Kebutuhan	SG-3-011

Nama Kasus Uji	Mencari Laporan Status *Waiting*
Prosedur	1. Mengakses halaman laporan *waiting* 2. Mencari laporan berstatus *waiting*
Expected Result	Sistem menampilkan laporan berstatus *waiting* yang dicari oleh aktor
Result	Sistem menampilkan laporan berstatus *waiting* yang dicari oleh aktor
Status	Valid

Analisis Hasil Pengujian Validasi Mencari Laporan Status *Waiting*

Pengujian validasi mencari laporan status *waiting* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji mencari laporan status *waiting*. Pengujian tersebut digunakan untuk memastikan fitur mencari laporan yang berstatus *waiting* dapat berjalan dengan baik pada website pengaduan masyarakat.

6.3.27 Pengujian Validasi Melihat Laporan Berdasarkan Status *OnProgress*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat laporan berdasarkan status *onprogress* yang ada pada sistem pengaduan masyarakat berdasarkan *scenario*, pengujian tersebut dapat ditunjukkan pada Tabel 6.37.

Tabel 6. 37 Pengujian Validasi Melihat Laporan Berdasarkan Status *OnProgress*

Kode Kebutuhan	SG-3-012
Nama Kasus Uji	Melihat Laporan Berdasarkan Status *OnProgress*
Prosedur	1. Aktor mengakses halaman laporan 2. Aktor memilih menu laporan *onprogress*
Expected Result	Sistem menampilkan daftar laporan yang berstatus *onprogress*
Result	Sistem menampilkan daftar laporan yang berstatus *onprogress*
Status	Valid

Analisis Hasil Pengujian Validasi Melihat Laporan Berdasarkan Status *OnProgress*

Pengujian validasi melihat laporan berstatus *onprogress* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji melihat laporan berdasarkan status *onprogress*. Pengujian ini digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik pada website pengaduan masyarakat.

6.3.28 Pengujian Validasi Melihat Detail Laporan Status *OnProgress*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat *detail* laporan berstatus *onprogress* berdasarkan *scenario* yang telah dirancang sebelumnya, pengujian tersebut dapat ditunjukkan pada Tabel 6.38.

Tabel 6. 38 Pengujian Validasi Melihat *Detail* Laporan Status *OnProgress*

Kode Kebutuhan	SG-3-013
Nama Kasus Uji	Melihat Detail Laporan Status *Onprogress*
Prosedur	1. Aktor mengakses menu laporan *onprogress* 2. Aktor memilih salah satu laporan yang berstatus *onprogress*
Expected Result	Sistem menampilkan *detail* laporan yang berstatus *onprogress* atau sedang diproses
Result	Sistem menampilkan *detail* laporan yang berstatus *onprogress* atau sedang diproses
Status	Valid

Analisis Hasil Pengujian Validasi Melihat *Detail* Laporan Status *OnProgress*

Pengujian validasi melihat detail laporan status *onprogress* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji melihat *detail* laporan berstatus *onprogress*. Pengujian digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik pada website pengaduan masyarakat yang digunakan.

6.3.29 Pengujian Validasi Mengganti Status Laporan *OnProgress*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional mengganti status laporan berstatus *onprogress* yang ada pada sistem pengaduan masyarakat berdasarkan *scenario* yang telah dibuat, pengujian tersebut dapat ditunjukkan pada Tabel 6.39.

Tabel 6. 39 Pengujian Validasi Mengganti Status Laporan *OnProgress*

Kode Kebutuhan	SG-3-014
Nama Kasus Uji	Mengganti Status Laporan *OnProgress*
Prosedur	1. Aktor menekan tombol “Proses” pada salah satu laporan berstatus *onprogress* 2. Aktor mengganti status laporan
Expected Result	Sistem menampilkan status laporan yang sudah diganti
Result	Sistem menampilkan status laporan yang sudah diganti

Status	Valid

Analisis Hasil Pengujian Validasi Mengganti Laporan Status *OnProgress*

Pengujian validasi mengganti laporan status *onprogress* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji. Pengujian digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik pada website pengaduan masyarakat yang digunakan.

6.3.30 Pengujian Validasi Membagikan Lokasi Kejadian Status *OnProgress*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional membagikan lokasi kejadian status *onprogress* yang ada pada sistem pengaduan masyarakat, pengujian tersebut dapat ditunjukkan pada Tabel 6.40.

Tabel 6. 40 Pengujian Validasi Membagikan Lokasi Kejadian Status *OnProgress*

Kode Kebutuhan	SG-3-015
Nama Kasus Uji	Membagikan Lokasi Kejadian Status *OnProgress*
Prosedur	*1.* Aktor menekan tombol "Proses" pada salah satu laporan berstatus *onprogress* 2. Aktor menekan tombol "Share" pada halaman "Detail Laporan OnProgress"
Expected Result	Sistem menampilkan pesan berupa *pop up* yang berisi deskripsi laporan dan lokasi kejadian yang sudah disalin
Result	Sistem menampilkan pesan berupa *pop up* yang berisi deskripsi laporan dan lokasi kejadian yang sudah disalin
Status	Valid

Analisis Hasil Pengujian Validasi Membagikan Lokasi Kejadian Status *OnProgress*

Pengujian validasi membagikan lokasi kejadian status *onprogress* ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji. Pengujian tersebut digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik pada website untuk admin.

6.3.31 Pengujian Validasi Mencari Laporan Status *OnProgress*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional mencari laporan berstatus *onprogress* yang ada pada sistem pengaduan masyarakat berdasarkan *scenario* yang telah dirancang sebelumnya, pengujian tersebut dapat ditunjukkan pada Tabel 6.41.

Tabel 6. 41 Pengujian Validasi Mencari Laporan Status *OnProgress*

Kode Kebutuhan	SG-3-016

Nama Kasus Uji	Mencari Laporan Status *OnProgress*
Prosedur	*1.* Mengakses halaman laporan *onprogress* 2. Mencari laporan berstatus *onprogress*
Expected Result	Sistem menampilkan laporan berstatus *onprogress* yang dicari oleh aktor
Result	Sistem menampilkan laporan berstatus *onprogres*s yang dicari oleh aktor
Status	Valid

Analisis Hasil Pengujian Validasi Mencari Laporan Status *OnProgress*

Pengujian validasi mencari laporan status *onprogress* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji mencari laporan status *onprogress*. Pengujian tersebut digunakan untuk memastikan fitur mencari laporan yang berstatus *onprogress* dapat berjalan dengan baik pada website pengaduan masyarakat.

6.3.32 Pengujian Validasi Melihat Laporan Berdasarkan Status *Done*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat laporan berdasarkan status *done* yang ada pada sistem pengaduan masyarakat berdasarkan *scenario* yang telah dirancang sebelumnya, pengujian tersebut dapat ditunjukkan pada Tabel 6.42.

Tabel 6. 42 Pengujian Validasi Melihat Laporan Berdasarkan Status *Done*

Kode Kebutuhan	SG-3-017
Nama Kasus Uji	Melihat Laporan Berdasarkan Status *Done*
Prosedur	1. Aktor mengakses halaman laporan 2. Aktor memilih menu laporan *done*
Expected Result	Sistem menampilkan daftar laporan yang berstatus *done*
Result	Sistem menampilkan daftar laporan yang berstatus *done*
Status	Valid

Analisis Hasil Pengujian Validasi Melihat Laporan Berdasarkan Status *Done*

Pengujian validasi melihat laporan berstatus *done* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji melihat laporan berdasarkan status *done*. Pengujian ini digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik.

6.3.33 Pengujian Validasi Melihat *Detail* Laporan Status *Done*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat *detail* laporan berstatus *done* yang ada pada sistem pengaduan

masyarakat berdasarkan *scenario* yang telah dirancang, pengujian tersebut dapat ditunjukkan pada Tabel 6.43.

Tabel 6. 43 Pengujian Validasi Melihat Detail Laporan Status *Done*

Kode Kebutuhan	SG-3-018
Nama Kasus Uji	Melihat *Detail* Laporan Status *Done*
Prosedur	1. Aktor mengakses menu laporan *done* 2. Aktor memilih salah satu laporan yang berstatus *done*
Expected Result	Sistem menampilkan *detail* laporan yang berstatus *done* atau sudah selesai ditindaklanjuti
Result	Sistem menampilkan *detail* laporan yang berstatus *done* atau sudah selesai ditindaklanjuti
Status	Valid

Analisis Hasil Pengujian Validasi Melihat *Detail* Laporan Status *Done*

Pengujian validasi melihat *detail* laporan status *done* menghasilkan status valid. Pengujian digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik pada website pengaduan masyarakat yang digunakan.

6.3.34 Pengujian Validasi Mengganti Status Laporan *Done*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional mengganti status laporan berstatus *done* yang ada pada sistem pengaduan masyarakat berdasarkan *scenario* yang telah dirancang sebelumnya, pengujian tersebut dapat ditunjukkan pada Tabel 6.44.

Tabel 6. 44 Pengujian Validasi Mengganti Status Laporan *Done*

Kode Kebutuhan	SG-3-019
Nama Kasus Uji	Mengganti Status Laporan *Done*
Prosedur	1. Aktor menekan tombol “Proses” pada salah satu laporan berstatus *done* 2. Aktor mengganti status laporan
Expected Result	Sistem menampilkan status laporan yang sudah diganti
Result	Sistem menampilkan status laporan yang sudah diganti
Status	Valid

Analisis Hasil Pengujian Validasi Mengganti Laporan Status *Done*

Pengujian validasi mengganti laporan status *done* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji mengganti laporan berstatus *done*. Pengujian digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik pada website pengaduan masyarakat yang digunakan.

6.3.35 Pengujian Validasi Mencari Laporan Status *Done*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional mencari laporan berstatus *done* yang ada pada sistem pengaduan masyarakat, pengujian tersebut dapat ditunjukkan pada Tabel 6.45.

Tabel 6. 45 Pengujian Validasi Mencari Laporan Status *Done*

Kode Kebutuhan	SG-3-020
Nama Kasus Uji	Mencari Laporan Status *Done*
Prosedur	1. Mengakses halaman laporan *done* 2. Mencari laporan berstatus *done*
Expected Result	Sistem menampilkan laporan berstatus *done* yang dicari oleh aktor
Result	Sistem menampilkan laporan berstatus *done* yang dicari oleh aktor
Status	Valid

Analisis Hasil Pengujian Validasi Mencari Laporan Status *Done*

Pengujian validasi mencari laporan status *done* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji mencari laporan status *done*. Pengujian tersebut digunakan untuk memastikan fitur mencari laporan yang berstatus *done* dapat berjalan dengan baik pada website pengaduan masyarakat.

6.3.36 Pengujian Validasi Melihat Laporan Berdasarkan Status *Block*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat laporan berdasarkan status *block* yang ada pada sistem pengaduan masyarakat berdasarkan *scenario*, pengujian tersebut dapat ditunjukkan pada Tabel 6.46.

Tabel 6. 46 Pengujian Validasi Melihat Laporan Berdasarkan Status *Block*

Kode Kebutuhan	SG-3-021
Nama Kasus Uji	Melihat Laporan Berdasarkan Status *Block*
Prosedur	1. Aktor mengakses halaman laporan 2. Aktor memilih menu laporan *block*
Expected Result	Sistem menampilkan daftar laporan yang berstatus *block*
Result	Sistem menampilkan daftar laporan yang berstatus *block*
Status	Valid

Analisis Hasil Pengujian Validasi Melihat Laporan Berdasarkan Status *Block*

Pengujian validasi melihat laporan berstatus *block* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji melihat laporan berdasarkan status *block.* Pengujian ini digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik pada website pengaduan masyarakat.

6.3.37 Pengujian Validasi Melihat *Detail* Laporan Status *Block*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional melihat *detail* laporan berstatus *block* yang ada pada sistem pengaduan masyarakat, pengujian tersebut dapat ditunjukkan pada Tabel 6.47.

Tabel 6. 47 Pengujian Validasi Melihat *Detail* Laporan Status *Block*

Kode Kebutuhan	SG-3-022
Nama Kasus Uji	Melihat Detail Laporan Status *Block*
Prosedur	1. Aktor mengakses menu laporan *block* 2. Aktor memilih salah satu laporan yang berstatus *block*
Expected Result	Sistem menampilkan *detail* laporan yang berstatus *block* atau melaporkan tidak sesuai ketentuan
Result	Sistem menampilkan *detail* laporan yang berstatus *block* atau melaporkan tidak sesuai ketentuan
Status	Valid

Analisis Hasil Pengujian Validasi Melihat *Detail* Laporan Status *Block*

Pengujian validasi melihat *detail* laporan status *block* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji melihat *detail* laporan berstatus *block*. Pengujian digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik.

6.3.38 Pengujian Validasi Mengganti Status Laporan *Block*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional mengganti status laporan berstatus *block* yang ada pada sistem pengaduan masyarakat berdasarkan *scenario* yang telah dirancang, pengujian tersebut dapat ditunjukkan pada Tabel 6.48.

Tabel 6. 48 Pengujian Validasi Mengganti Status Laporan *Block*

Kode Kebutuhan	SG-3-023
Nama Kasus Uji	Mengganti Status Laporan *Block*
Prosedur	1. Aktor menekan tombol “Proses” pada salah satu laporan berstatus *block* 2. Aktor mengganti status laporan

Expected Result	Sistem menampilkan status laporan yang sudah diganti
Result	Sistem menampilkan status laporan yang sudah diganti
Status	Valid

Analisis Hasil Pengujian Validasi Mengganti Laporan Status *Block*

Pengujian validasi mengganti laporan status *block* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji mengganti laporan berstatus *block*. Pengujian digunakan untuk memastikan fitur tersebut dapat berjalan dengan baik pada website pengaduan masyarakat yang digunakan.

6.3.39 Pengujian Validasi Mencari Laporan Status *Block*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional mencari laporan yang berstatus *block* yang ada pada sistem pengaduan masyarakat berdasarkan *scenario*, pengujian tersebut dapat ditunjukkan pada Tabel 6.49.

Tabel 6. 49 Pengujian Validasi Mencari Laporan Status *Block*

Kode Kebutuhan	SG-3-024
Nama Kasus Uji	Mencari Laporan Status *Block*
Prosedur	1. Mengakses halaman laporan *block* 2. Mencari laporan berstatus *block*
Expected Result	Sistem menampilkan laporan berstatus *block* yang dicari oleh aktor
Result	Sistem menampilkan laporan berstatus *block* yang dicari oleh aktor
Status	Valid

Analisis Hasil Pengujian Validasi Mencari Laporan Status *Block*

Pengujian validasi mencari laporan status *block* menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji mencari laporan status *block*. Pengujian tersebut digunakan untuk memastikan fitur mencari laporan yang berstatus *block* atau laporan yang tidak sesuai dengan ketentuan untuk pelaporan dapat berjalan dengan baik pada website pengaduan masyarakat.

6.3.40 Pengujian Validasi *Logout*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan fungsional *logout administrator* yang ada pada sistem pengaduan masyarakat atau pada website yang digunakan berdasarkan *scenario* yang telah dirancang sebelumnya, pengujian tersebut dapat ditunjukkan pada Tabel 6.50.

Tabel 6. 50 Pengujian Validasi *Logout* Admin

Kode Kebutuhan	SG-3-025
Nama Kasus Uji	*Logout*
Prosedur	1. Aktor mengakses halaman *dashboard* 2. Aktor menekan tombil l*ogout*
Expected Result	Sistem melakukan proses *logout* keluar dari sistem
Result	Sistem melakukan proses *logout* keluar dari sistem
Status	Valid

Analisis Hasil Pengujian Validasi *Logout*

Analisis hasil dari pengujian validasi *logout* ini menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji *logout*. Pengujian tersebut digunakan untuk memastikan fitur *logout* ini dapat dilakukan dengan baik pada website pengaduan masyarakat yang digunakan oleh admin.

6.3.41 Pengujian Validasi Sistem Tidak Dapat Menampilkan *Password* pada Respon Akhir atau *End Point*

Berikut merupakan pengujian validasi yang dilakukan untuk kebutuhan non fungsional yakni sistem tidak dapat menampilkan *password* pada setiap respon akhir yang ada pada aplikasi pengaduan masyarakat maupun sistem pengaduan masyarakat yang digunakan admin yang mengacu pada *scenario* yang telah dirancang sebelumnya, pengujian tersebut dapat ditunjukkan pada Tabel 6.51.

Tabel 6. 51 Pengujian Validasi Sistem Tidak Dapat Menampilkan *Password* pada Respon Akhir atau *End Point*

Kode Kebutuhan	SG-4-001
Nama Kasus Uji	Sistem Tidak Dapat Menampilkan *Password* pada Respon Akhir atau *End Point*
Prosedur	Melakukan pengecekan pada setiap respon akhir atau *end point*
Expected Result	Pada semua *end point* atau respon akhir tidak menampilkan *password*
Result	Pada semua *end point* atau respon akhir tidak menampilkan *password*
Status	Valid

Analisis Hasil Pengujian Validasi Sistem Tidak Dapat Menampilkan *Password* pada Respon Akhir atau *End Point*

Analisis hasil dari pengujian validasi sistem tidak dapat menampilkan *password* pada setiap respon akhir atau *end point* baik pada aplikasi *mobile* maupun pada website yang digunakan oleh admin yang menghasilkan status valid yang dilakukan melalui *scenario* yang ada pada prosedur uji non fungsional tersebut.

6.4 Pengujian *Usability*

Pengujian *Usability* atau pengujian kebergunaan merupakan pengujian yang dilakukan untuk memberikan penilaian oleh pengguna mengenai evaluasi hasil akhir dari sebuah perangkat lunak. Pengujian *usability* dilakukan dengan tujuan untuk mengetahui kemudahan perangkat lunak yang sudah diimplementasikan, seberapa besar kemudahan interaksi antarmuka yang dapat digunakan oleh *user* terhadap sistem. Tujuan dari mengetahuinya kemudahan dari sistem ialah untuk mengetahui apakah sistem tersebut bersifat efisien, dan efisien. Melakukan pengujian ini dilakukan dengan tujuan untuk memastikan bahwa sistem sudah memenuhi salah satu kebutuhan non fungsional yang sudah dijelaskan pada bab perancangan yakni kebutuhan non fungsional yang ada pada aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang mengenai *usability* sistem. Untuk itu, pengujian *usability* ini dilakukan kepada masyarakat Kota Malang dengan menggunakan metode kuesioner atau disebut dengan *SUS* atau *System Usability Scale*. Berikut merupakan pernyataan kuesioner untuk aplikasi *mobile* pengaduan masyarakat.

Tabel 6. 52 Pernyataan Kuesioner pada Aplikasi Pengaduan Masyarakat

No	Pernyataan	Pilihan Jawaban
1	Saya akan menggunakan aplikasi pengaduan masyarakat untuk melaporkan masalah-masalah yang ada di jalan seperti kemacetan, rambu-rambu lalu lintas, dan lain sebagainya.	☐ ☐ ☐ ☐ ☐ 1 2 3 4 5
2	Menurut saya, terdapat fitur yang tidak penting.	☐ ☐ ☐ ☐ ☐ 1 2 3 4 5
3	Menurut saya, aplikasi pengaduan masyarakat ini mudah digunakan.	☐ ☐ ☐ ☐ ☐ 1 2 3 4 5
4	Saya membutuhkan bantuan dari orang lain untuk menggunakan aplikasi pengaduan masyarakat ini.	☐ ☐ ☐ ☐ ☐ 1 2 3 4 5
5	Menurut saya, fitur-fitur yang ada pada aplikasi pengaduan masyarakat sudah terintegrasi dengan baik.	☐ ☐ ☐ ☐ ☐ 1 2 3 4 5

6	Menurut saya, terdapat beberapa hal yang tidak konsisten pada aplikasi pengaduan masyarakat.	☐ ☐ ☐ ☐ ☐ 1 2 3 4 5
7	Menurut saya, masyarakat dapat cepat mengerti sehingga dapat dengan mudah menggunakan aplikasi pengaduan masyarakat untuk melaporkan keluhan.	☐ ☐ ☐ ☐ ☐ 1 2 3 4 5
8	Menurut saya, aplikasi pengaduan masyarakat ini tidak praktis digunakan untuk melaporkan pengaduan masyarakat.	☐ ☐ ☐ ☐ ☐ 1 2 3 4 5
9	Saya yakin bisa menggunakan aplikasi pengaduan masyarakat.	☐ ☐ ☐ ☐ ☐ 1 2 3 4 5
10	Saya perlu mempelajari banyak hal sebelum menggunakan aplikasi pengaduan masyarakat.	☐ ☐ ☐ ☐ ☐ 1 2 3 4 5

Keterangan Tabel 6.52:

1 = Sangat Tidak Setuju

2 = Tidak Setuju

3 = Netral

4 = Setuju

5 = Sangat Setuju

Tabel 6. 53 Hasil Rekapitulasi Kuesioner pada Aplikasi Pengaduan Masyarakat

Responden	Pernyataan									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	4	3	5	3	4	2	5	1	5	5
2	4	2	4	2	4	3	4	3	5	2
3	4	1	5	2	4	1	5	1	5	2
4	5	1	5	1	5	1	5	2	5	1
5	5	1	5	2	5	1	5	2	5	1
6	5	1	4	2	4	2	4	2	4	1
7	3	2	4	2	4	1	5	2	4	2
8	4	1	4	2	4	1	4	2	4	1
9	4	3	4	4	4	3	4	2	4	2

10	5	1	5	1	4	2	4	1	5	1
11	4	1	5	2	4	1	5	1	5	2
12	5	4	5	1	4	3	5	2	5	1
13	5	3	4	1	4	2	5	1	5	3
14	4	1	5	1	5	2	4	1	5	3
15	5	1	4	4	5	2	4	1	4	5
16	4	2	4	1	4	1	5	2	5	1
17	5	3	5	3	4	4	5	5	5	4
18	4	2	4	2	3	2	3	2	4	3
19	4	1	5	1	5	1	5	5	5	1
20	4	2	5	1	3	1	5	1	5	2

Tabel 6. 54 *Score* dari Kuesioner *SUS* pada Aplikasi Pengaduan Masyarakat

Responden	**Pernyataan**										**Jumlah**	***Score SUS***
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**		
1	3	2	4	2	3	3	4	4	4	0	29	72.5
2	3	3	3	3	3	2	3	2	4	3	29	72.5
3	3	4	4	3	3	4	4	4	4	3	36	90
4	4	4	4	4	4	4	4	3	4	4	39	97.5
5	4	4	4	3	4	4	4	3	4	4	38	95
6	4	4	3	3	3	3	3	3	3	4	33	82.5
7	2	3	3	3	3	4	4	3	3	3	31	77.5
8	3	4	3	3	3	4	3	3	3	4	33	82.5
9	3	2	3	1	3	2	3	3	3	3	26	65
10	4	4	4	4	3	3	3	4	4	4	37	92.5
11	3	4	4	3	3	4	4	4	4	3	36	90
12	4	1	4	4	3	2	4	3	4	4	33	82.5
13	4	2	3	4	3	3	4	4	4	2	33	82.5
14	3	4	4	4	4	3	3	4	4	2	35	87.5
15	4	4	3	1	4	3	3	4	3	0	29	72.5
16	3	3	3	4	3	4	4	3	4	4	35	87.5

17	5	1	5	1	4	2	4	1	5	1	33	82.5
18	3	4	4	3	3	4	4	4	4	3	36	90
19	3	4	4	4	4	4	4	0	4	4	35	87.5
20	3	3	4	4	2	4	4	4	4	3	35	87.5
											Total *Score SUS*	1677
											Rata-Rata *Score SUS*	83.75

Keterangan Tabel 6.53 dan 6.54:
Responden : Orang yang memberikan jawaban pada kuesioner
Pernyataan : Pernyataan kuesioner

Rekapitulasi hasil kuesioner untuk aplikasi pengaduan masyarakat yang terdapat pada Tabel 6.53. Pada Tabel tersebut merupakan nilai yang diberikan oleh responden pada setiap pernyataannya. Setelah itu nilai yang telah didapatkan dari responden dihitung untuk mencari jumlah keseluruhan nilai dan total *score* kuesioner dengan menggunakan rumus dari *System Usability Scale*. Perhitungan *score* tersebut ialah pada pernyataan ganjil dikurangi 1. Sedangkan pernyataan genap diperoleh dengan mengurangi 5 dari nilai jawaban yang diberikan responden.

Tabel 6.54 merupakan Tabel *Score* dari *SUS* dari kuesioner aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang. Kolom yang ada antara kolom Responden dan kolom Pernyataan merupakan kolom dari nilai-nilai yang diberikan oleh pengguna terhadap sistem. Sedangkan kolom jumlah diperoleh dari jumlah total nilai yang diberikan oleh responden. Sehingga untuk mendapatkan *score SUS* pada Tabel 6.54 diperoleh dari nilai pada kolom jumlah dikali dengan 2,5. Sedangkan untuk mendapatkan nilai total *score SUS* diperoleh dari seluruh nilai *score SUS* dijumlahkan maka akan didapat total *score SUS*. Lalu nilai total *score* yang telah diperoleh selanjutnya dibagikan dengan dengan jumlah responden. Sehingga diperoleh rata-rata total *score SUS* dari kuesioner pada aplikasi pengaduan masyarakat dengan total *score SUS* sebesar 1677 dari total keseluruhan 20 responden sehingga rata-rata yang didapatkan yakni 83,75 yang memiliki arti dapat digunakan dan diterima oleh pengguna atau bisa disebut juga dengan *acceptable*.

6.5 Pembahasan Hasil Pengujian

Pembahasan hasil pengujian menjelaskan mengenai pembahasan dari keseluruhan hasil pengujian yang telah dilakukan. Pengujian yang telah dilakukan pada sistem ini yakni pengujian *unit*, pengujian integrasi, pengujian validasi, dan pengujian *usability*. Setelah pengujian-pengujian tersebut telah dilakukan maka akan diperoleh analisis dari masing-masing pengujian. Berikut merupakan pembahasan hasil pengujian *unit*, pengujian integrasi, pengujian validasi, dan pengujian *usability*.

6.5.1 Hasil Pengujian *Unit*

Pengujian pada aplikasi pengaduan masyarakat telah selesai dilakukan yakni dengan melakukan pengujian *whitebox* dengan *basis path testing* dan pengujian integrasi, selain itu melakukan juga pengujian *blackbox* dengan pengujian validasi, untuk itu hasil pengujian yang didapatkan setelah melakukan pengujian-pengujian tersebut yakni masing-masing berstatus valid pada *test case* tiap pengujian, maka dari itu aplikasi pengaduan masyarakat yang telah dibuat layak digunakan proses penyampian pengaduan masyarakat pada Dinas Perhubungan Kota Malang. Pengujian pertama melakukan pengujian dengan metode *whitebox testing* yakni *basis path testing*, pengujian ini dilakukan pada tiga *method* dari aplikasi ini seperti *method* "*upload*" dari *class* "*UploadActivity*", *method* "save" dari *class* "*EditMyLaporanActivity*", *dan method* "*editStatus*" dari *class* "*WaitingController*". Untuk pengujian *unit* pada ketiga *method* tersebut diperoleh 2 *test case* dimana masing-masing *test case* tersebut bersifat valid.

Ketiga pengujian *unit* yang dilakukan tersebut memiliki korelasi antar *class* yang digunakan untuk membangun fitur-fitur yang ada pada aplikasi pengaduan masyarakat. Fitur-fitur yang dilakukan pengujian tersebut yakni fitur menambah laporan pengaduan baru, mengubah laporan pribadi, dan mengganti status laporan yang dilakukan oleh admin. Untuk itu fitur-fitur ini dapat digunakan dengan baik, dengan demikian pengujian *unit* pada ketiga *method* yang dilakukan tersebut dapat membantu permasalahan yang ada pada rumusan masalah poin pertama yang mana aplikasi *mobile* pengaduan masyarakat dapat membantu masyarakat untuk menyampaikan keluhan kepada Dinas Perhubungan Kota Malang dengan fitur-fitur yang telah dilakukan pengujian *unit* ini.

6.5.2 Hasil Pengujian Integrasi

Hasil Pengujian kedua yang telah dilakukan yakni pengujian integrasi pada 3 pasang *class* untuk aplikasi pengaduan masyarakat ini yaitu *class* "UploadActivity" dan *class* "AddLaporanService" dengan masing-masing *class* tersebut memiliki *method* yang dipasangkan yakni *method* "*upload*" dengan *method* "*addLaporan*", *class* kedua ialah dari *class* "*EditMyLaporanActivity*" dan *class* "*EditMyLaporanService*" dimana masing-masing *method* pada tiap *class* ialah *method* "*save*" dan *method* "*editMyLaporan()*", dan *class* yang ketiga yaitu *class* "*PickLocationActivity*" dimana *method* yang saling berhubungan yakni *method* "*onCreate*" dengan *method* "*OnMapReadyCallback()*", maka jumlah kasus uji pada pengujian integrasi sebanyak 3 kasus uji dimana ketiga kasus uji tersebut memiliki status valid.

Berdasarkan hasil pengujian integrasi yang dilakukan pada aplikasi pengaduan masyarakat ini terdiri dari masing-masing pengujian integrasi bernilai valid untuk setiap kasus uji. Ketiga pengujian *unit* yang dilakukan tersebut memiliki korelasi antar *class* yang digunakan untuk membangun fitur-fitur yang ada pada aplikasi pengaduan masyarakat. Fitur-fitur yang dilakukan pengujian tersebut yakni menambah laporan pengaduan baru, mengubah laporan pribadi serta menambah lokasi baru yang akan digunakan untuk mengetahui lokasi pelapor dengan fitur

location based service. Untuk itu fitur-fitur ini dapat digunakan dengan baik, dengan demikian pengujian integrasi antar *class* yang dilakukan tersebut dapat membantu permasalahan yang ada pada rumusan masalah poin kedua yang mana aplikasi *mobile* pengaduan masyarakat dapat mengimplementasikan *fitur location based service* pada aplikasi ini untuk mempermudah dalam menemukan lokasi yang akurat mengenai lokasi pelapor.

6.5.3 Hasil Pengujian Validasi

Hasil pengujian validasi yang telah dilakukan pada aplikasi pengaduan masyarakat ini ialah pengujian dengan metode *blackbox testing* dengan pengujian validasi. Pengujian validasi dilakukan berdasarkan kebutuhan fungsional yang ada pada aplikasi ini dengan melakukan pengujian pada setiap *scenario* kebutuhan fungsional yang telah dibuat. Kebutuhan fungsional pada aplikasi pengaduan masyarakat menghasilkan kasus uji sebanyak 41 kasus dimana keseluruhan kasus uji tersebut bersifat valid.

Hasil pengujian validasi yang telah dilakukan terhadap 41 kasus uji untuk kebutuhan fungsional pada aplikasi pengaduan masyarakat memiliki status valid. Dimana salah satu kasus uji yang ada pada pengujian validasi ini ialah dilakukan pengujian untuk semua fitur berdasarkan kebutuhan fungsional dan non fungsional pada aplikasi ini dimana fitur-fitur tersebut dapat memberikan wadah, layanan serta fasilitas dalam menyampaikan pengaduan masyarakat. Sehingga dari hasil pengujian validasi ini dapat membantu permasalahan yang ada pada rumusan masalah poin pertama mengenai aplikasi *mobile* pengaduan masyarakat dapat memberikan wadah serta layanan untuk mempermudah masyarakat dalam menyampaikan keluhan pada Dinas Perhubungan Kota Malang mengenai permasalahan layanan publik.

6.5.4 Hasil Pengujian *Usability*

Hasil pengujian *usability* untuk aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang didapatkan *score* kuesioner dari metode *SUS* atau *System Usability Scale* ialah 83,75. Nilai yang didapatkan dari *SUS* tersebut ialah kategori *acceptable* atau bisa dikatakan aplikasi *mobile* pengaduan masyarakat dapat diterima dan layak untuk digunakan oleh pengguna atau masyarakat Kota Malang, dikarenakan nilai 83,75 masuk dalam kategori *acceptable* dimana *score* tersebut berada dalam nilai 71 sampai dengan 100. Dengan demikian hasil pengujian *usability* untuk aplikasi *mobile* pengaduan masyarakat pada Dinas Perhubungan Kota Malang berdasarkan hasil yang didapatkan bahwa aplikasi *mobile* ini dapat digunakan dengan mudah untuk pengguna atau masyarakat Kota Malang sehingga kebutuhan non fungsional yang ada pada tahap perancangan telah terpenuhi dengan baik.

Berdasarkan pengujian *usability* yang telah dilakukan dengan cara melakukan pengujian terhadap kemudahan aplikasi pengaduan masyarakat kepada pengguna dengan menggunakan kuesioner dengan metode *System Usability Scale* atau *SUS* didapatkan hasil bahwa aplikasi pengaduan masyarakat ini dapat dimengerti dan dapat diterima oleh penggunanya dengan mendapatkan *score* kuesioner sebanyak

83,75 yang mana termasuk ke dalam golongan *acceptable* atau dapat diterima oleh masyarakat. Dengan demikian berdasarkan hasil analisis pengujian *usability* yang telah dilakukan dapat membantu menjawab pertanyaan dari rumusan masalah pada poin ketiga mengenai tingkat kemudahan atau *usability* aplikasi pengaduan masyarakat berbasis Android.

BAB 7 PENUTUP

7.1 Kesimpulan

Mengacu pada hasil analisis kebutuhan, perancangan, implementasi, dan pengujian yang telah dilakukan pada penelitian ini, maka dari itu dapat disimpulkan beberapa poin yang dapat menjawab rumusan masalah. Kesimpulan tersebut sebagai berikut:

1. Dengan terselesaikannya pengimplementasian aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang diperoleh bahwa aplikasi ini dapat memberikan wadah dan fasilitas pada aplikasi *mobile* untuk pengaduan masyarakat tentang sarana umum lalu lintas guna membantu masyarakat untuk menyampaikan keluhan pada Dinas Perhubungan Kota Malang dengan telah dilakukannya pengujian *unit* dan pengujian validasi dimana dari hasil analisis pengujian *unit* dan pengujian validasi didapatkan fitur-fitur yang ada pada aplikasi pengaduan masyarakat yang memiliki status valid. Sehingga aplikasi pengaduan masyarakat ini dapat dikatakan mampu membantu dalam menyampaikan pengaduan kepada Dinas Perhubungan Kota Malang.
2. Hasil implementasi pada penelitian yang telah dilakukan ini diperoleh implementasi fitur *location based service* pada aplikasi *mobile*. Untuk membantu dalam pengimplementasian fitur *location based service* diperlukan koneksi internet dan GPS untuk mengetahui lokasi kejadian yang dilaporkan. Berdasarkan pengujian integrasi yang telah dilakukan mengenai pengimplementasian fitur *location based service* aplikasi *mobile* pengaduan masyarakat ini dapat membantu pihak Dinas Perhubungan Kota Malang dalam hal penggunaan fitur *location based service* untuk mengetahui lokasi akurat dari lokasi pelaporan.
3. Tingkat kemudahan dari aplikasi *mobile* pengaduan masyarakat berbasis android yang telah dilakukan pengujian dengan melakukan pengujian *usability* dengan menggunakan kuesioer dengan metode *SUS* atau *System Usability Scale* memperoleh *score* kuesioner sebesar 83,75. *Score* yang diperoleh tersebut dapat dikategorikan aplikasi yang *acceptable* atau dapat diterima dan mudah digunakan oleh penggunanya. Sehingga aplikasi *mobile* pengaduan masyarakat pada Dinas Perhubungan Kota Malang dapat diterima oleh masyarakat atau penggunanya.

7.2 Saran

Berdasarkan beberapa kesimpulan pada penelitian yang telah dijabarkan, didapatkan beberapa poin saran untuk aplikasi pengaduan masyarakat pada Dinas Perhubungan Kota Malang menggunakan fitur *Location Based Service* berbasis Android yang digunakan untuk pengembangan selanjutnya, yakni sebagai berikut:

1. Aplikasi ini perlu ditambahkan fitur notifikasi untuk pelapor agar pelapor mengetahui status laporannya secara *real time* apakah sudah ditindaklanjuti.

2. Aplikasi dapat ditambahkan fitur yang memudahkan komunikasi antara pengguna satu dengan pengguna lainnya seperti fitur *comment*.
3. Aplikasi pengaduan masyarakat juga dapat ditambahkan fitur untuk memfasilitasi antara admin atau penanggung jawab proses pengaduan masyarakat dengan petugas lapangan yang menindaklanjuti pengaduan untuk saling berkomunikasi dan memonitoring mengenai laporan yang masuk.
4. Aplikasi pengaduan masyarakat dapat ditambahkan fitur untuk memasukkan foto laporan lebih dari satu foto agar lebih memperjelas pengaduan yang ingin dilaporkan.

DAFTAR PUSTAKA

Anwar, B., Jaya, H. & Kusuma, P. I., 2014. Implementasi Location Based Service berbasis Android untuk Mengetahui Posisi User. *Jurnal Ilmiah SAINTIKOM,* Volume 13, p. 2.

Argawal, B. B., Tayal, S. P. & Gupta, M., 2010. *Software Engineering & Testing.* 1st ed. Massachusetts: Jones and Bartlett.

Brooke, J., 2013. SUS-A Quick And Dirty Usability Scale. *Usability Evaluation In Industry 189,* Volume 194, pp. 4-7.

Caytiles, R. D. & Lee, S., 2014. *A Review of an MVC Framework based Software Development.* s.l.:International Journal of Software Engineering and Its Application.

Dishub, K. M., 2015. Perubahan Rencana Strategis (RENSTRA). In: D. P. K. Malang, ed. *Perubahan Rencana Strategis 2014-2018.* Malang: Dinas Perhubungan Kota Malang, p. 1.

Fauzia, S. Q., 2017. Rancang Bangun Sistem Pelaporan Keluhan Masyarakat Dengan Fitur Geotagging Berbasis Android. *Repositori Jurnal Mahasiswa PTIIK UB,* Volume 9, p. 15.

Firdaus, F. I., 2016. Rancang Bangun Aplikasi Mobile Jual Beli Produk Pertanian Menggunakan Pendekatan Location Based Service. *Repositori Jurnal Mahasiswa PTIIK UB,* Volume 7, p. 9.

Gustafson, D., 2002. *Software Engineering.* New York: McGraw Hill.

Juhara, Z. P., 2016. *Panduan Lengkap Pemrograman ANDROID.* Yogyakarta: ANDI.

Kindarto, A., 2008. *Asyik Berinternet dengan Beragam Layanan Google.* Yogyakarta: Penerbit Andi.

Lethbridge, T. & Laganiere, R., 2005. *Object-oriented Software Engineering: Practical Software Development Using UML and Java.* 2nd ed. London: McGraw-Hill.

Mardani, A., 2012. Sistem Informasi Geografis Pelaporan Masyarakat (Sigma) Berbasis Foto Geotag. *Naskah Publikasi.*

Masykur, F., 2014. Implementasi Sistem Informasi Geografis Menggunakan Google *Maps* API dalam Pemetaan Asal Mahasiswa. *Jurnal SIMETRIS ,* Volume 5 Nomor 2, pp. 181-186.

Nugroho, A., 2010. *Rekayasa Perangkat Lunak Berorientasi Objek dengan Metode USDP.* Jakarta: Andi Offset.

Pambudi, A., 2013. Implementasi Model Perangkat Lunak Pelayanan Informasi Kegiatan Belajar Mengajar Tingkat SLTA degan Berbasis Operating System Android. *Jurnal Ilmu Komputer ,* Volume 9, p. 2.

P., D. & M., 2013. *Analisis Pengelolaan Pengaduan Masyarakat dalam Rangka Pelayanan Publik (Studi pada Dinas Komunikasi dan Informatika Kota Malang).* Malang: s.n.

Perhubungan, K. M. D., 2015. Perubahan Rencana Strategis (RENSTRA). In: *Perubahan Rencana Strategis (RENSTRA).* Malang: s.n.

Prasetya, A. H., 2010. *Cepat Kuasai PHP dan MySQL.* Yogyakarta: CV Andi.

Prasetya, D. R., Domai, T. & Mindarti , L. I., 2013. ANALISIS PENGELOLAAN PENGADUAN MASYARAKAT DALAM RANGKA PELAYANAN PUBLIK (Studi Pada Dinas Komunikasi dan Informatika Kota Malang). *Jurusan Adminsitrasi Publik, Fakultas Ilmu Administrasi, Universitas Brawijaya, Malang ,* Volume 2, pp. 1151-1158.

Pressman, R. s., 2010. *Software Engineering: a practitioner's approach.* 7th ed. New York: McGraw-Hill.

Pressman, R. S., 2011. *Software engineering: a practitioner's approach.* 7th ed. New York: McGraw-Hill.

Raval, R. & Gonsai, A., 2015. Performance Analysis and Design of a Mobile Web Services on Cloud Servers. *International Journal of Emerging Technology and Advanced Engineering ,* Volume 5(9), pp. 104-113.

Rumbaugh, I, J. & Booch, G., 2005. *The Unified Modeling Language reference manual.* 2 ed. Boston: Addison-Wesley.

Safaat, N., 2013. *Aplikasi Berbasis Android.* Bandung: Informatika.

Santoso, P. I., H.N, I. A. & Ferdiana, R., 2015. Pengujian Usability Menggunakan System Usability Scale. *IPTEK-KOM,* Volume 17, pp. 31-38.

Sauro, J., 2015. Measuring Usability With The System Usability Scale (SUS). *MeasuringU.*

Shiode, et al., 2004. The Impact and Penetration of Location-Based Services. *UCL Centre for advanced spatial analysis,* Volume 50, pp. 1-16.

S., N. M. & E. A., 2008. Foundations of Location Based Services. *Project CarouCHe,* pp. 1-28.

Sommervile, I., 2003. *Software Engineering: Rekayasa Perangkat Lunak jilid 1.* 6 ed. Jakarta: Penerbit Erlangga.

Sommerville, I., 2011. *Software engineering.* 9 ed. London: Addison-Wesley.

Steiniger, S., Neun, M. & Edwardes, A., 2006. *Foundations of Location Based Service.* V.1.0 ed. Bristol, USA: Taylor & Francis.

Suarga, 2009. *Dasar Pemrograman Komputer dalam Bahasa Java.* Yogyakarta: ANDI.

Sukamto, R. & Shalahuddin, M., 2014. *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek.* 2nd ed. Bandung: Informatika.

Symu, A., 2013. *Peringkat OS Smartphone Berdasarkan Prediksi IDC.* s.l., [ONLINE] Tersedia di : <https://www.tabloidpulsa.co.id/news/6451-peringkat-os-smartphone-berdasarkan-prediksi-idc>.

Wiriyo, F. H., 2016. Pengembangan Aplikasi Mobile Pelaporan Keluhan Pelanggan Listrik Menggunakan Fitur Location Based Service Berbasis Android. *Repositori Jurnal Mahasiswa PTIIK UB,* Volume 8, p. 24.

Wiyanto, 2011. *Pengelolaan Komplain (Keluhan) Masyarakat Dalam Mewujudkan Tata Pamong Yang Baik (Good Governance) Di Kota Semarang.* Universitas Negeri Semarang ed. Semarang: S1.

W., S. & S., 2011. *Pengelolaan Komplain (Keluhan) Masyarakat Dalam Mewujudkan Tata Pamong Yang Baik (Good Governance) Di Kota Semarang.* Universitas Negeri Semarang ed. Semarang: S1.

Yasin, V., 2012. *Rekayasa Perangkat Lunak Berorientasi Objek.* Jakarta: Penerbit Mitra Wacana Media.

Ziad, I., 2013. Rancang Bangun Pelacak Lokasi dengan Teknologi GPS. *Jurnal Teknologi dan Informatika,* Volume 3, p. 1.