PENGEMBANGAN SISTEM PELAPORAN GANGGUAN BERBASIS *WEB* DENGAN MENGGUNAKAN TEKNOLOGI *PROGRESSIVE WEB APPLICATION*

(Studi kasus: Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Ari Setiawan
NIM: 155150200111083

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

PENGESAHAN

PENGEMBANGAN SISTEM PELAPORAN GANGGUAN BERBASIS *WEB* DENGAN MENGGUNAKAN TEKNOLOGI *PROGRESSIVE WEB APPLICATION*
(Studi kasus: Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Ari Setiawan
NIM: 155150200111083

Skripsi ini telah diuji dan dinyatakan lulus pada
12 Maret 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing 2
Denny Sagita R., S.Kom, M.Kom	Agi Putra Kharisma, S.T, M.T
NIP: 19851124 201504 1 001	NIK: 2013048 604301 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

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 referensi.

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, 12 Maret 2019

Ari Setiawan

NIM: 155150200111083

PRAKATA

Assalamu'alaikum Wr. Wb. Puji dan syukur penulis panjatkan atas rahmat dan karunia Allah SWT sehingga penulis dapat menyelesaikan skripsi yang berjudul Pengembangan Sistem Pelaporan Gangguan Berbasis Web dengan Menggunakan Teknologi *Progressive Web Application* (Studi kasus: Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon). Skripsi ini diajukan sebagai salah satu syarat untuk mendapatkan gelar Sarjana Komputer di Fakultas Ilmu Komputer, Universitas Brawijaya.

Selama proses penyusunan skripsi, penulis banyak belajar dan mengaplikasikan ilmu yang telah penulis dapatkan dari perkuliahan. Semoga penulis dapat terus mengembangkan ilmu yang diperoleh selama perkuliahan dengan baik.

Dalam penyusunan skripsi ini penulis mendapat banyak dukungan dan bantuan dari berbagai pihak. Oleh sebab itu, penulis ingin menguncapkan terima kasih kepada:

1. Bapak Denny Sagita R., S.Kom, M.Kom. selaku Pembimbing I, yang telah membimbing dan mengarahkan penulis dalam proses penyelesaian skripsi.
2. Bapak Agi Putra Kharisma, S.T, M.T selaku Pembimbing II, yang telah membimbing dan mengarahkan penulis sehingga skripsi ini dapat penulis selesaikan.
3. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
5. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya.
6. Bapak dan Ibu dosen serta staff Fakultas Ilmu Komputer Universitas Brawijaya yang telah bersedia membagi ilmu dan arahan kepada penulis.
7. Bapak Arsadi dan Ibu Suhaeti sebagai kedua orang tua saya serta Lintang Ardianti sebagai adik saya yang senantiasa memberikan motivasi, semangat dan do'a.
8. Bapak Awal Arifianto beserta jajaran pegawai unit IT PT. Kereta Api Indonesia Daerah III Cirebon yang telah memberikan informasi untuk mendukung tugas akhir ini.
9. Diana Rachmawati yang selalu menjadi penyemangat dan memberikan motivasi bagi penulis untuk menyelesaikan skripsi ini.
10. Keluarga Departemen Pengembangan Ilmu dan Profesi yang selalu menjadi penyemangat bagi penulis untuk menyelesaikan skripsi ini.

11. Rekan-rekan EMIF (Eksekutif Mahasiswa Informatika) Fakultas Ilmu Komputer yang telah mendukung dan memotivasi.
12. Teman-teman satu angkatan Teknik Informasi 2015 yag selalu memberikan informasi, semangat, dorongan dan bantuan pikiran.
13. Semua pihak yang tidak dapat disebutkan satu – persatu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Penulis menyadari dalam proses penyusunan skripsi masih banyak terdapat kekurangan, oleh sebab itu kritik dan saran yang membangun sangat diperlukan dalam penulisan selanjutnya. Penulis berharap semoga skripsi ini memberikan manfaat kepada pihak yang membutuhkan.

Malang, 12 Maret 2019

Penulis

arisetiawan@student.ub.ac.id

ABSTRAK

Ari Setiawan, Pengembangan Sistem Pelaporan Gangguan Berbasis *Web* Dengan Menggunakan Teknologi *Progressive Web Application* (Studi kasus: Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon)

Pembimbing: Denny Sagita R., S.Kom, M.Kom dan Agi Putra Kharisma, S.T, M.T

PT. Kereta Api Indonesia (PERSERO) yang selanjutnya disingkat menjadi PT. KAI merupakan salah satu Badan Usaha Milik Negara yang menyediakan transportasi dengan menggunakan kereta api. PT. KAI memiliki 9 daerah operasi, daerah operasi III Cirebon salah satunya. Misi dari PT. KAI yaitu menyelenggarakan kegiatan bisnisnya dengan dilandasi oleh 4 pilar utama yaitu keselamatan, ketepatan waktu, pelayanan dan kenyamanan. Untuk mencapai 4 pilar tersebut unit IT PT. KAI daerah operasi III Cirebon membuat sebuah prosedur penanganan gangguan. Namun prosedur ini masih dilakukan secara konvensional, sehingga masih memiliki beberapa permasalahan seperti sering terjadi laporan gangguan yang tidak berhasil diterima, kurangnya informasi tentang petugas yang sedang bertugas, petugas tidak dapat melapor kepada pimpinan saat ingin pergi ke tempat gangguan dan belum adanya laporan perangkat yang keluar dari gudang penyimpanan. Untuk menangani masalah tersebut dibuatlah sebuah solusi berbentuk sebuah sistem yang memiliki fitur utama yaitu pelaporan gangguan, penanganan gangguan dan manajemen gudang penyimpanan. Penelitian dimulai dengan melakukan rekayasa kebutuhan yang mendapatkan 48 kebutuhan fungsional dan 1 kebutuhan non-fungsional sistem. Kemudian melakukan tahapan perancangan dan implementasi. Sistem ini dikembangkan dengan menggunakan bahasa *PHP*, *JavaScript*, *CSS* dan *HTML* yang menggunakan pola perancangan *MVC* (*Model, View, Controller)* berbasis *web*. Sistem ini dilengkapi dengan teknologi *progressive web application*. Sistem ini menggunakan *database MySQL*. Setelah melakukan implementasi, dilakukan pengujian terhadap sistem. Sistem ini telah berhasil diuji dengan pengujian unit, pengujian integrasi, pengujian validasi dan pengujian *compatibility*. Hasil dari pengujian sistem yang dibangun menghasilkan 100% valid untuk semua pengujian, yang berarti sistem yang dibuat telah sesuai dengan hasil analisis kebutuhan dan perancangan sebelumnya.

Kata kunci: *progressive web application*, pelaporan, gangguan, *web*, *model-view-controller*

ABSTRACT

Ari Setiawan, Pengembangan Sistem Pelaporan Gangguan Berbasis *Web* Dengan Menggunakan Teknologi *Progressive Web Application* (Studi kasus: Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon)

Supervisors: Denny Sagita R., S.Kom, M.Kom and Agi Putra Kharisma, S.T, M.T

PT. Kereta Api Indonesia (PERSERO), hereinafter abbreviated as PT. KAI is one of the Badan Usaha Milik Negara that provides transportation by train. PT. KAI has 9 operational areas, one of which is Cirebon operation area. The mission of PT. KAI is conducting business activities based on 4 main pillars, namely safety, punctuality, service and comfortability. To reach these 4 pillars, the IT unit of PT. KAI operating area III Cirebon makes a procedure for handling distrubance. But this procedure is still done conventionally, so that it still has several problems such as frequent reports of disturbances that are not successfully delivered, lack of information about officers who are on duty, officers unable to report to the leader when they want to go to place of distrubance and there are no reports of device coming out of the warehouse. To deal with this problem a solution was formed in the form of a system that has the main features disturbance reporting, disturbance handling and warehouse management. Research begins with requirement engineering which analyzed to 48 functional requirements and 1 non-functional requirement. The next step is to make a design and implementation. This system is developed using PHP, JavaScript, CSS and HTML using MVC design patterns (Model, View, Controller) based on web. This system is equipped with progressive web application technology. This system uses a MySQL database. After implementation is finished, testing of the system is carried out. This system has been successfully tested with unit testing, integration testing, validation testing and compatibility testing. The results of the system test is 100% valid for all tests, which means the system is compatible with analysis and design result.

Keywords: progressive web application, reporting, distrubance, web, model-view-controller

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Kepuasan pelanggan merupakan salah satu parameter yang dapat mengindikasikan baik tidaknya sebuah penyedia layanan. Semakin tinggi kualitas layanan maka akan semakin tinggi pula kepuasan pelanggan yang didapatkan (Khurshid, Naeem, Ejaz, Mukhtar, & Batool, 2012). Bagi penyedia layanan, *availability* atau ketersediaan menjadi salah satu faktor utama untuk meningkatkan kepuasan pelanggan. Ketika kegagalan layanan terjadi, cara perusahaan untuk pulih, dan kecepatan dimana mereka melakukannya, memiliki dampak penting pada respons pelanggan berikutnya (Chou, Hsu, & Goo, 2009).

Salah satu perusahaan yang menawarkan pelayanan jenis transportasi adalah PT. Kereta Api Indonesia (PERSERO). PT. Kereta Api Indonesia (PERSERO) yang selanjutnya disingkat menjadi PT. KAI merupakan salah satu Badan Usaha Milik Negara yang menyediakan transportasi dengan menggunakan kereta api. Misi dari PT. KAI yaitu menyelenggarakan kegiatan bisnisnya dengan dilandasi oleh 4 pilar utama yaitu keselamatan, ketepatan waktu, pelayanan dan kenyamanan (PT Kereta Api Indonesia (Persero), 2017). PT. KAI dibagi menjadi 9 Daerah Operasi, salah satunya adalah Daerah Operasi III Cirebon yang menaungi 32 stasiun di dalamnya. Untuk mewujudkan 4 pilar tersebut, unit IT PT. KAI Daerah Operasi III Cirebon memiliki prosedurnya dalam menangani gangguan yang terjadi di wilayah operasinya. Prosedurnya dengan menghubungi telepon kantor unit IT, setelah itu petugas unit IT yang sedang bertugas akan mengatasi laporan tersebut dengan dua cara yaitu yang pertama dengan memberikan pengarahan apa yang harus dilakukan oleh pelapor. Jika cara pertama tidak dapat menyelesaikan gangguan maka dilakukanlah cara yang kedua, yaitu *lintas*. *Lintas* merupakan prosedur mengunjungi langsung tempat gangguan (Arifianto, 2018). *Lintas* tidak dilakukan hanya jika terjadi gangguan saja, namun sudah ada *lintas* yang dijadwalkan setiap bulannya untuk melakukan *maintenance*.

Namun prosedur tersebut masih memiliki beberapa kekurangan yaitu yang pertama saat tidak ada satupun pegawai unit IT yang berada di kantor maka tidak ada yang mengangkat telepon sehingga gangguan tidak dapat segera terselesaikan. Yang kedua adalah kurangnya informasi tentang petugas yang sedang bertugas. Yang ketiga adalah masalah ketika suatu petugas ingin pergi melakukan *lintas* namun tak ada manajer atau asisten manajer sehingga petugas tersebut tidak dapat melapor jika ingin *lintas*. Dan masalah yang terakhir adalah saat belum adanya pencatatan barang keluar dari dalam gudang penyimpanan untuk pergantian perangkat seperti komputer dan printer saat penanganan gangguan.

Dari masalah yang sudah dipaparkan diatas, penulis memiliki sebuah alternatif solusi dengan melakukan pengembangan satu aplikasi yang bertujuan untuk membantu proses penanganan laporan gangguan yang masuk ke unit IT PT. Kereta Api Indonesia DAOP III Cirebon.

Aplikasi ini akan dikembangkan dengan menggunakan *platform web* yang menggunakan teknologi *Progressive Web Application* (PWA) dengan alasan sebagian besar aplikasi pada PT. KAI DAOP III menggunakan *platform web* sehingga untuk kedepannya akan lebih mudah untuk dapat mengintegrasikannya dan juga dengan menggunakan PWA dapat mempermudah pengguna untuk mengaksesnya serta meminimalisir biaya dalam melakukan *maintenance*.

Untuk mewujudkan solusi yang tepat guna yang diajukan oleh penulis diperlukan hasil analisis kebutuhan yang sesuai dengan kebutuhan pengguna, perancangan yang sesuai dengan hasil analisis kebutuhan, hasil implementasi yang sesuai dengan hasil perancangan dan dilakukan pengujian untuk memastikan sistem yang dibuat sesuai dengan kebutuhan user.

1.2 Rumusan Masalah

1. Bagaimanakah analisis kebutuhan yang dapat digunakan untuk Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon?
2. Bagaimanakah rancangan yang dapat digunakan untuk Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon?
3. Bagaimanakah implementasi dari Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon?
4. Bagaimanakah pengujian dari Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon?

1.3 Tujuan

1. Menganalisis kebutuhan Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon sesuai kebutuhan pengguna.
2. Merancang Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon yang sesuai dengan hasil analisis kebutuhan.
3. Membuat Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon.
4. Menguji Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon.

1.4 Manfaat

Bagi unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon untuk meningkatkan efektivitas dan efisiensi prosedur pelaporan dan penanganan gangguan pada Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon.

1.5 Batasan Masalah

Lingkungan Sistem Pelaporan Gangguan yang dibatasi hanya bisa digunakan oleh pegawai PT. Kereta Api Indonesia Daerah yang berada dibawah naungan PT. Kereta Api Indonesia Daerah Operasi III Cirebon dan dibatasi pada pelaporan gangguan, penanganan laporan gangguan, pelaporan *lintas*, informasi petugas unit IT yang sedang bertugas, informasi riwayat gangguan, penjadwalan kerja petugas unit IT dan pendataan barang keluar untuk kegiatan *lintas*.

1.6 Sistematika Pembahasan

Sistematika pembahasan laporan ditunjukkan guna memberikan gambaran serta uraian dari penelitian ini secara garis besar. Untuk mencapai tujuan yang diharapkan maka sistematika penulisan yang disusun dalam tugas akhir ini meliputi beberapa bab, sebagai berikut:

1. **BAB 1 PENDAHULUAN**

Bab ini akan berisi tentang latar belakang dibuatnya Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon, rumusan masalah, tujuan penelitian kenapa mengembangkan Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon, manfaat penelitian pengembangan Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon, batasan masalah yang berkaitan dengan pengembangan Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon, dan sistematika pembahasan yang dipergunakan dalam menyusun penelitian ini.

2. **BAB 2 LANDASAN KEPUSTAKAAN**

Bab ini berisi tentang kajian pustaka, dasar teori yang melandasi penulisan dan penelitian pengembangan Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon ini yaitu teori tentang PT. Kereta Api Indonesia, pengembangan perangkat lunak, tahapan pengembangan perangkat lunak, *Software Development Life Cycle* model *Waterfall, Unified Modifying Language* (UML), pola perancangan *Model-View-Controller*, teknologi pengembangan Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon serta pengujian yang dilakukan terhadap Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon.

3. **BAB 3 METODOLOGI**

Bab ini berisi tentang studi literatur atau teori yang digunakan dalam penelitian ini, tahap-tahap yang dilakukan pada saat melakukan rekayasa kebutuhan, tahap-tahap pada saat perancangan sistem, tahap-tahap pada implementasi sistem, tahap-tahap pengujian yang dilakukan pada sistem serta kesimpulan dan saran.

4. **BAB 4 REKAYASA KEBUTUHAN**

Bab ini berisi tentang hasil proses rekayasa kebutuhan pengguna aplikasi Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon mulai dari elisitasi kebutuhan hingga hasil analisis kebutuhan.

5. **BAB 5 PERANCANGAN DAN IMPLEMENTASI**

Bab ini berisi penjelasan mengenai rancangan sistem yang akan digunakan dalam pembangunan Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon dan pada bab ini juga akan menjelaskan gambaran dari sistem dan deskripsi sistem berdasarkan hasil analisis kebutuhan yang telah dilakukan pada bab sebelumnya.

6. **BAB 6 PENGUJIAN**

Bab ini berisi hasil pengujian dari beberapa pengujian yang telah dilakukan pada Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon.

7. **BAB 7 PENUTUP**

Bab ini akan berisi tentang kesimpulan dan saran yang merupakan hasil dari penelitian yang telah dilakukan sekaligus jawaban dari rumusan masalah yang telah dibuat.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Pada kajian pustaka akan mengandung penjelasan tentang penelitian-penelitian sebelumnya yang telah dilakukan dan membahas tema penelitian yang sejenis. Terdapat tiga penelitian yang membahas tentang tema yang sejenis dengan yang penulis bahas yaitu pertama penelitian yang dilakukan oleh Richa Amalia Permatasari. Penelitian ini berjudul "Pengembangan Sistem Aplikasi Pelaporan Masyarakat Berbasis *Web* di Kabupaten Pekalongan". Penelitian ini menjelaskan mengenai pengembangan Sistem Aplikasi Pelaporan Masyarakat pada kabupaten Pekalongan yang dapat membantu masyarakat Pekalongan untuk melapor langsung ke Organisasi Perangkat Daerah (OPD) yang terkait, dapat melihat respon yang diberikan OPD dan progres dari pemecahan masalah yang perlu ditindaklanjuti (Permatasari, Priyambadha, & Arwan, 2018).

Yang kedua penelitian yang dilakukan oleh Nanang Basir yang berjudul "Desain dan Implementasi Aplikasi Pelaporan Gangguan Sistem SCADA (LGS) Pada PT. PLN APD Bandung". Pada penelitian ini dijelaskan bahwa sistem ini dibangun untuk dapat membantu dalam proses pencatatan laporan gangguan sistem SCADA, validasi indikasi gangguan dari sistem SCADA, menyediakan laporan gangguan kepada pihak-pihak yang menggunakannya, meng*input*, mengedit serta menampilkan data-data yang berkaitan dengan formulir gangguan, perbaikan serta data-data lainnya yang digunakan oleh *user*. Pada kesimpulan penelitian ini menjelaskan bahwa aplikasi ini memudahkan petugas untuk mencari, meng*input* serta melakukan *update* data-data gangguan, *work order*, data perbaikan serta data-data lainnya dengan mudah dan cepat jika dibandingkan dengan sistem sebelumnya (Basir, 2005).

Yang ketiga adalah penelitian yang dilakukan oleh Muhammad Rasyid Ridho yang berjudul "Perbandingan Performa *Progressive Web Apps* dan *Mobile Web* Terkait Waktu Respon, Penggunaan Memori dan Penggunaan Media Penyimpanan". Pada penelitian ini dijelaskan bahwa *Progressive Web Apps* mampu mengungguli *Mobile Web* terkait waktu respon pada ukuran berkas dan *cache* yang cukup besar dan performa dari PWA akan semakin baik apabila frekuensi pengaksesan yang tidak hanya sekali saja (Ridho, Pinandito, & Dewi, 2018).

Dari ketiga penelitian yang menjadi kajian pustaka penelitian, dapat ditarik kesimpulan bahwa diharapkan nantinya Sistem Pelaporan Gangguan di Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon ini dapat meningkatkan efektivitas dan efisiensi proses pelaporan gangguan serta diharapkan laporan gangguan yang terjadi tepat sampai kepada petugas yang sedang bertugas. Selain itu, dengan penggunaan teknologi *Progressive Web Application* juga diharapkan bahwa aplikasi ini dapat memiliki waktu respon yang cepat dalam mengatasi ukuran berkas dan *cache* yang cukup besar.

2.2 Landasan Teori

2.2.1 PT. Kereta Api Indonesia

PT Kereta Api Indonesia (Persero) selanjutnya akan ditulis PT. KAI adalah salah satu Badan Usaha Milik Negara (BUMN) yang menyediakan layanan berbasis transportasi dengan menggunakan kereta api. Wilayah operasi PT. KAI saat ini berada di pulau Jawa, Aceh, Sumatera Utara, Sumatera Barat, Sumatera Selatan dan Lampung. PT. KAI melayani jasa angkutan penumpang dan barang.

Daerah Operasi Kereta Api Indonesia atau yang biasa disingkat DAOP KAI merupakan pembagian daerah operasi kereta api Indonesia dibawah lingkungan PT. KAI yang berada dibawah direksi PT. KAI yang dipimpin oleh seorang *Executive Vice President* (EVP). Ada 9 daerah operasi yaitu, Daerah Operasi I Jakarta, Daerah Operasi II Bandung, Daerah Operasi III Cirebon, Daerah Operasi IV Semarang, Daerah Operasi V Purwokerto, Daerah Operasi VI Yogyakarta, Daerah Operasi VII Madiun, Daerah Operasi VIII Surabaya, dan Daerah Operasi IX Jember.

2.2.2 Pengembangan Perangkat Lunak

Akan ada beberapa hal yang akan dibahas pada sub bab ini diantaranya adalah model pengembangan perangkat lunak itu sendiri, model *waterfall*, pendekatan dan pemodelan berorientasi objek.

2.2.2.1 Tahapan Pengembangan Perangkat Lunak

Ada banyak jenis model proses perangkat lunak yang mencakup metodologi konvensional serta inovatif seperti *waterfall, prototyping,* pengembangan *iterative* dan *incremental,* pengembangan spiral, *rapid application development*, dan *extreme programming*. Jenis metodologi ini berbagi beberapa tahap perkembangan berikut (Lee, 2014):

1. Tahap Analisis
2. Tahap Kebutuhan
3. Tahap Mendesain Sistem
4. Tahap Implementasi (*coding*)
5. Tahap Pengujian
6. Tahap *Deployment*
7. Tahap *Maintenance* dan memperbaiki *bug*

2.2.2.2 *Software Development Life Cycle* Model *Waterfall*

Menurut (Pressman, 2010), model pengembangan *waterfall* atau sering disebut juga dengan *classic life cycle* merupakan model pengembangan yang bersifat sistematis yang menggunakan pendekatan sekuensial. Pengembangan dengan model *waterfall* ini dimulai dengan proses *communication* (komunikasi) yang didalamnya terdapat proses pengumpulan kebutuhan dan kemudian dilanjutkan dengan proses *planning* (perencanaan), *modeling* (pemodelan),

construction (konstruksi) dan diakhiri dengan proses *deployment* (penyebaran). Berikut merupakan SDLC model *waterfall* dalam bentuk gambar pada Gambar 2.1.

Berikut ini merupakan penjelasan dari setiap tahapnya:

1. *Communication*

Langkah ini adalah proses analisis terhadap kebutuhan perangkat lunak, dan tahap untuk melakukan pengumpulan data dengan mengadakan pertemuan dengan pengguna, maupun megumpulkan data tambahan yang berasal dari jurnal, artikel, maupun dari *internet*.

2. *Planning*

Proses *planning* adalah lanjutan dari proses *communication*. Tahapan ini akan mengeluarkan hasil berupa dokumen kebutuhan pengguna atau bisa disebut dengan data yang berhubungan dengan keinginan pengguna dalam pembuatan perangkat lunak, termasuk rencana atau *plan* yang akan dilakukan.

3. *Modeling*

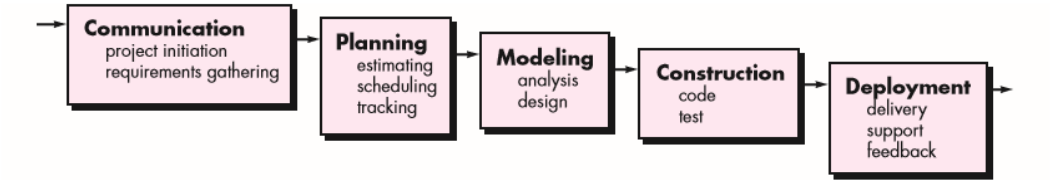
Dalam proses ini akan terjadi proses penerjemahan kebutuhan yang telah didapatkan pada tahap sebelumnya ke sebuah rancangan perangkat lunak yang dapat diperkirakan sebelum diterjemahkan ke dalam kode program. Tahap ini akan menghasilkan dokumen yang juga disebut *software requirement*.

4. *Construction*

Construction merupakan proses penerjemahan kebutuhan menjadi kode program. *Coding* atau bisa juga disebut proses pengkodean merupakan proses untuk menerjemahkan rancangan menjadi bahasa yang dapat dimengerti oleh komputer. Tahapan inilah yang merupakan tahapan nyata dalam mengerjakan suatu perangkat lunak, artinya penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan dengan melakukan pengujian terhadap sistem yang telah dibuat tadi. Tujuan dari pengujian itu sendiri adalah menemukan kesalahan-kesalahan yang ada pada sistem tersebut agar kemudian dapat diperbaiki.

5. *Deployment*

Tahapan ini bisa dikatakan tahap akhir dalam pembuatan sebuah perangkat lunak. Setelah melakukan analisis, desain atau perancangan dan pengkodean maka sistem yang berhasil dibuat akan mulai digunakan oleh pengguna. Perangkat lunak yang telah dibuat juga harus dijaga dengan melakukan pemeliharaan secara berkala.



Gambar 2.1 SDLC model *waterfall*

(Sumber: Pressman, 2010)

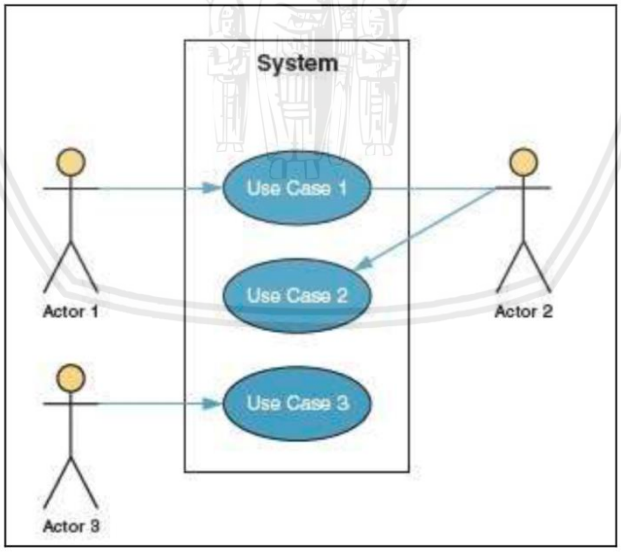
2.2.3 Unified Modifying Language (UML)

UML merupakan suatu kesepakatan pemodelan yang digunakan untuk mendeskripsikan atau menspesifikasikan sebuah *software* yang terkait dengan objek (Whitten & Bentley, 2007). Pada UML sendiri terdiri dari beberapa diagram, yaitu:

a. Use Case Diagram

Use case diagram merupakan diagram yang digunakan untuk menggambarkan interaksi sistem dengan pengguna (Whitten & Bentley, 2007). Jadi, diagram ini akan mendeskripsikan siapa saja yang akan menggunakan sistem dan bagaimana pengguna berinteraksi dengan sistem.

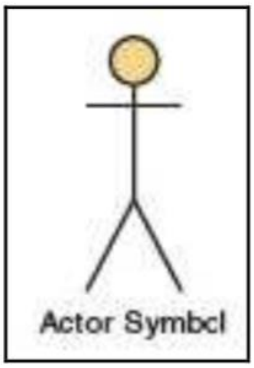
1. *Use case* menggambarkan fungsi sistem dari perspektif pengguna eksternal dan dengan cara dan terminologi yang mereka pahami. *Use case* digambarkan pada Gambar 2.2 dengan bentuk oval.



Gambar 2.2 Diagram *use case*

(Sumber: Whitten & Bentley, 2007)

2. Aktor merupakan sesuatu yang dapat berinteraksi dengan sistem. Simbol aktor digambarkan pada Gambar 2.3.



Gambar 2.3 Simbol aktor

(Sumber: Whitten & Bentley, 2007)

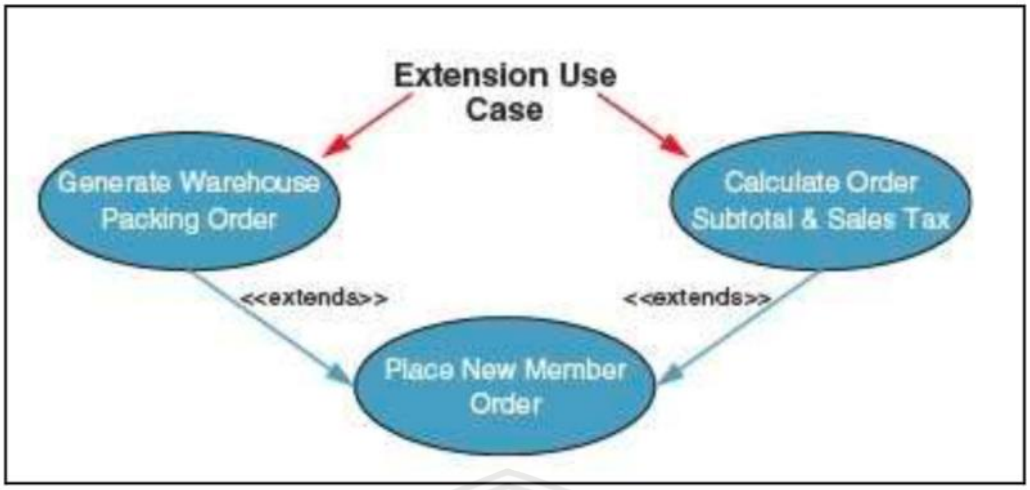
3. Hubungan (*Relationship)* pada *use case diagram* digambarkan sebagai sebuah garis antara dua simbol entah itu aktor maupun *use case*. Ada beberapa tipe hubungan atau *relationship* yang terdapat pada *use case diagram*, yaitu:

    1) Gabungan (*Association*): hubungan yang terjadi antara aktor dan *use case*. Gabungan digambarkan pada Gambar 2.4.

Gambar 2.4 Hubungan atau *relationship association*

(Sumber: Whitten & Bentley, 2007)

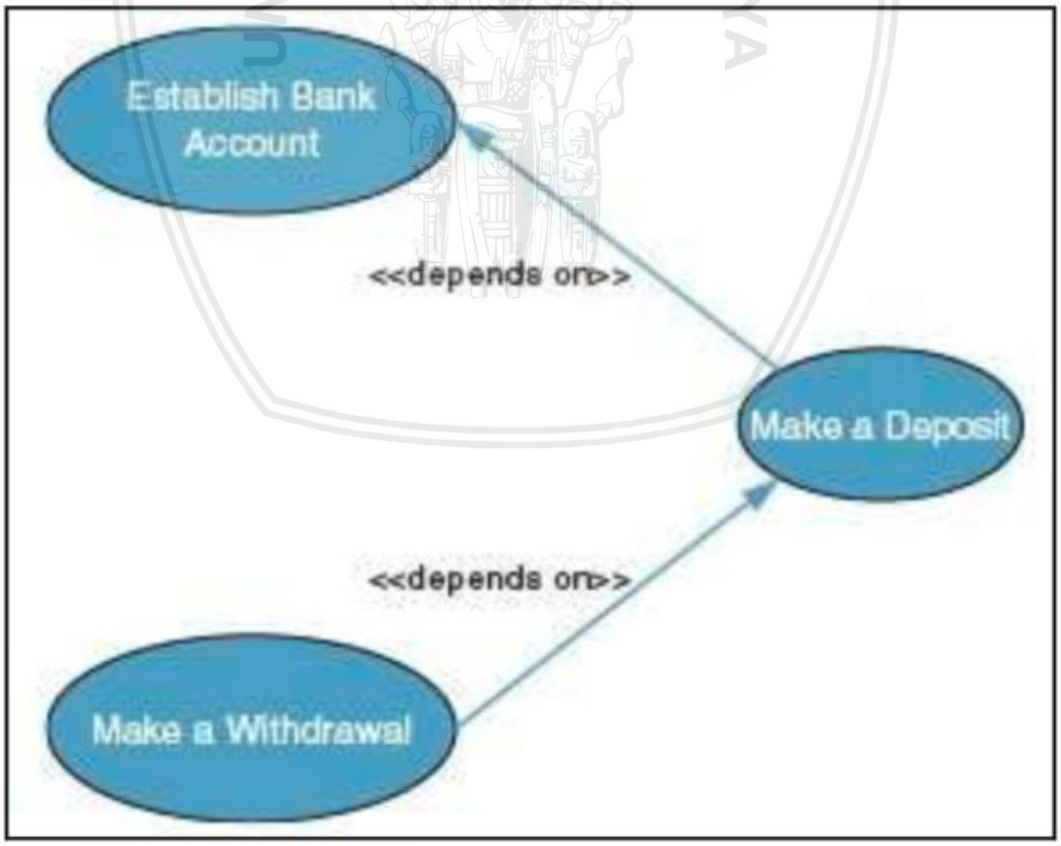
    2) *Extends:* hubungan yang terjadi antara *use case* yang terdiri dari langkah yang diekstraksi dari *use case* yang lebih kompleks untuk menyederhanakannya. *Extends* digambarkan pada Gambar 2.5.



Gambar 2.5 Hubungan atau *relationship extends*

(Sumber: Whitten & Bentley, 2007)

3) *Depends On:* hubungan yang terjadi antara *use case* yang memiliki ketergantungan pada *use case* yang lain untuk menetapkan rangkaian *use case* yang perlu dikembangkan. *Depends On* digambarkan pada Gambar 2.6.

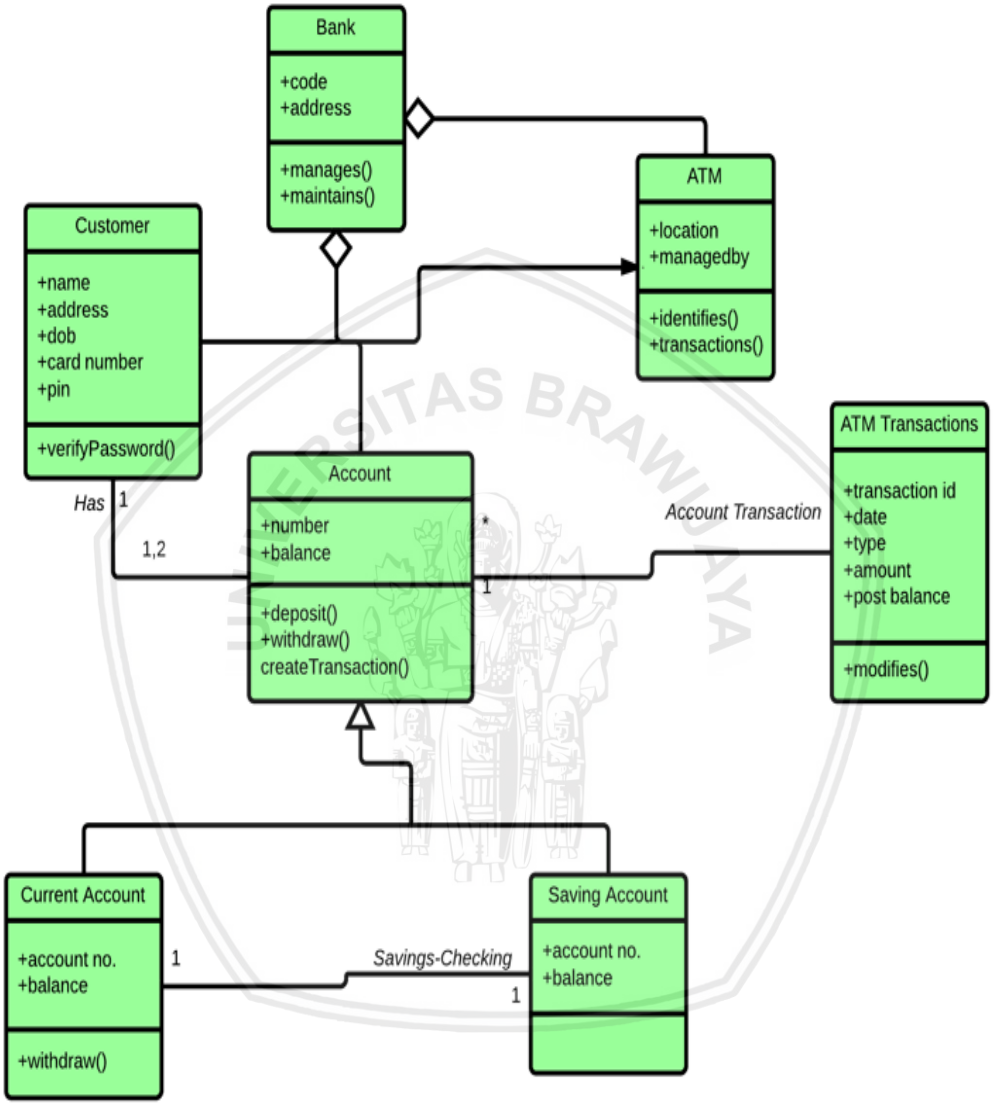


Gambar 2.6 Hubungan atau *relationship depends on*

(Sumber: Whitten & Bentley, 2007)

b. Class Diagram

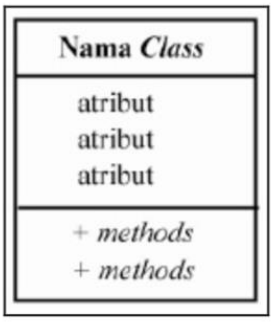
Class diagram adalah diagram yang menggambarkan objek yang ada pada sistem yang akan dibangun beserta hubungannya (Whitten & Bentley, 2007). Contoh *Class Diagram* digambarkan pada Gambar 2.11.



Gambar 2.7 Contoh *class diagram*

(Sumber: Guru99, 2018)

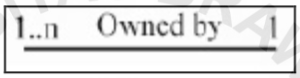
1) *Class* atau kelas direpresentasikan sebagai sebuah persegi yang memiliki tiga bagian. Bagian yang pertama adalah bagian atas yang merupakan bagian nama dari *class*. Bagian yang kedua adalah bagian tengah merupakan atribut *class*. Dan bagian akhir yang mendefinisikan fungsi atau *behaviour* yang dimiliki oleh sebuah *class*. *Class* digambarkan pada Gambar 2.7.



Gambar 2.8 *Class*

(Sumber: Whitten & Bentley, 2007)

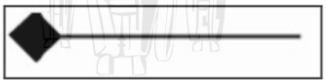
2) *Association* merupakan hubungan yang umum terjadi diantara dua *class* dan direpresentasikan dengan garis yang menghubungkan dua *class*. Ada beberapa tipe-tipe hubungan yaitu, *one-to-one, one-to-many, many-to-many. Association* digambarkan pada Gambar 2.8.



Gambar 2.9 *Association*

(Sumber: Whitten & Bentley, 2007)

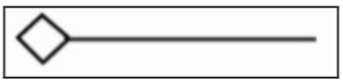
3) *Composition* merupakan hubungan yang terjadi jika sebuah *class* tidak dapat berdiri sendiri dan harus termasuk bagian dari *class* lain. Sebuah hubungan *composition* direpresentasikan dengan garis dengan ujung berbentuk jajar genjang berwarna hitam. *Composition* digambarkan pada Gambar 2.9.



Gambar 2.10 *Composition*

(Sumber: Whitten & Bentley, 2007)

4) *Agregation* merupakan hubungan yang terjadi jika sebuah *class* bisa berdiri sendiri meskipun termasuk bagian dari *class* lain. *Agregation* digambarkan pada Gambar 2.10.



Gambar 2.11 *Agregation*

(Sumber: Whitten & Bentley, 2007)

5) UML menyediakan 3 tingkat *visibility* atau cara pengaksesan, yaitu:

1. *Public*, direpresentasikan menggunakan tanda “+”.

2. *Protected*, direpresentasikan menggunakan tanda “#”.

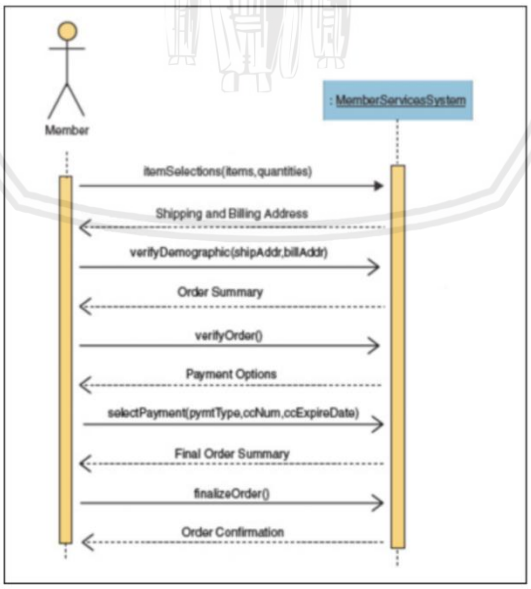
3. *Private*, direpresentasikan menggunakan tanda "-'.

c. *Sequence* Diagram

Sequence diagram merupakan diagram yang menggambarkan cara objek-objek berinteraksi melalui pesan dalam proses eksekusi dari suatu *use case* (Whitten & Bentley, 2007). Contoh *Class Diagram* digambarkan pada Gambar 2.12.

Berikut ini merupakan notasi-notasi yang ada pada *sequence diagram*, yaitu:

1. *Actor*, merupakan aktor yang memulai proses eksekusi *use case* yang direpresentasikan dengan simbol aktor pada *use case*.
2. *System*, direpresentasikan dengan sebuah kotak yang menggambarkan sistem yang sedang berjalan.
3. *Lifelines*, direpresentasikan dengan garis putus-putus vertikal dari simbol sistem dan aktor.
4. *Activation bars*, direpresentasikan dengan balok yang ada di atas *lifelines*, yang menunjukkan periode waktu ketika terdapat interaksi secara aktif.
5. *Input messages*, direpresentasikan dengan anak panah mendatar dari aktor menuju sistem yang menggambarkan pesan masuk.
6. *Output messages*, direpresentasikan dengan anak panah mendatar dengan garis putus-putus dari sistem menuju aktor yang menggambarkan pesan keluar.
7. *Receiver Actor*, direpresentasikan dengan aktor lain yang menerima pesan dari sistem.
8. *Frame*, direpresentasikan dengan kotak yang berisi pesan terpisah untuk menggambarkan perulangan, *alternative*, atau pilihan.



Gambar 2.12 S*equence* diagram

(Sumber: Whitten & Bentley, 2007)

2.2.4 *Model-View-Controller*

Model-View-Controller merupakan salah satu dari sekian banyak pola perancangan yang memiliki 3 komponen utama yang digunakan untuk membangun aplikasi, yaitu *model*, *view*, dan *controller*. *Model* merupakan bagian yang biasanya berhubungan langsung dengan *database* yang fungsinya adalah untuk melakukan manipulasi data, menangani validasi dari bagian *controller*, tapi tidak berhubungan dengan bagian *view*. *View* merupakan bagian yang berfungsi untuk menerima dan merepresentasikan data kepada pengguna. Sedangkan *controller* merupakan jembatan pengubung antara bagian *model* dan bagian *view* yang berfungsi untuk menerima *request* dan data dari *user* yang kemudian *controller* akan menentukan apa yang seharusnya diproses oleh aplikasi (Riyandwyana & Mukhlason, 2012).

2.2.5 *Progressive Web Application*

Progressive Web Apps (PWA) adalah satu *set* standar baru yang dianjurkan oleh *Google Web Fundamentals group* yang berupaya menjembatani kesenjangan antara aplikasi *native* dan aplikasi *web* dengan memperkenalkan fitur-fitur seperti *offline support*, sinkronisasi latar belakang, dan pemasangan layar beranda ke *web* (Biørn-Hansen, Majchrzak, & Grønli, 2017).

Progressive Web Apps (PWA) adalah aplikasi berbasis *web* yang memuat halaman *web* atau situs *web* biasa tetapi menawarkan fungsionalitas pengguna seperti dapat bekerja secara *offline*, *push notification* dan akses perangkat keras yang tersedia untuk aplikasi *native*. PWA ini sendiri dikembangkan oleh Google. *Progressive Web Apps* menggunakan beberapa teknologi diantaranya *Hypertext Transfer Protocol Secure* (HTTPS), *Manifest* dan *Service Workers*.

Menurut (Lepage, 2018), PWA memiliki beberapa karakteristik, yaitu:

a) ***Progressive*** - Bekerja untuk setiap pengguna, terlepas dari pilihan *browser* karena dibuat dengan peningkatan progresif sebagai prinsip inti.

b) ***Responsive*** - Sesuai dengan berbagai faktor bentuk: *desktop*, seluler, tablet, atau apa pun yang berikutnya.

c) ***Connectivity independent*** - Ditingkatkan dengan *service worker* untuk bekerja di luar atau pada jaringan dengan kualitas rendah.

d) ***App-like*** - Terasa seperti aplikasi, karena model *shell* aplikasi memisahkan fungsi aplikasi dari konten aplikasi.

e) ***Fresh*** - Selalu *up-to-date* berkat proses pembaruan *service worker*.

f) ***Safe*** - Ditayangkan melalui HTTPS untuk mencegah pengintaian dan memastikan konten belum dirusak.

g) ***Discoverable*** - Diidentifikasi sebagai "aplikasi" berkat cakupan W3C dan ruang lingkup pendaftaran *service worker*, yang memungkinkan mesin pencari menemukannya.

h) ***Re-engageable*** - Membuat keterlibatan ulang mudah melalui fitur seperti *push notification*.

i) ***Installable*** - Memungkinkan pengguna menambahkan aplikasi yang paling berguna ke layar beranda tanpa kerumitan toko aplikasi.

j) ***Linkable*** - Mudah berbagi aplikasi melalui URL, tidak memerlukan instalasi yang rumit.

Progressive Web Apps memiliki tiga kebutuhan yang harus dipenuhi, yaitu (Malavolta, 2016):

1. Harus dilayani melalui *Hypertext Transfer Protocol Secure* (HTTPS).
2. Memiliki *web app manifest*.
3. Memiliki setidaknya satu *service worker.*

Hypertext Transfer Protocol Secure

Hypertext Transfer Protocol Secure (HTTPS) adalah hasil pengembangan dari HTTP [RFC2616] dengan menambahkan tindakan keamanan berorientasi saluran dengan menggunakan SSL, dan penerusnya TLS [RFC2246] (Rescorla, 2000). Hal ini dapat meningkatkan keamanan men*transfer* halaman *web* melalui jaringan komputer (mis., *Internet*) dengan mencegah *man-in-the-middle attack*, yang mana *man-in-the-middle attacks* dapat dicapai oleh seseorang yang memotret identitas mereka untuk bertindak sebagai penerima yang dituju antara dua pihak komunikasi. Dengan TLS atau SSL, hal semacam *man-in-the-middle-attack* dapat terhambat dengan mengenkripsi data dan memverifikasi identitas *server* terhadap pihak ketiga, yaitu *certificate authority* (CA). CA menerima informasi identitas bersama dengan kunci publik dari *server* dan memeriksa ini terhadap kunci pribadi yang disimpan di *server*nya sendiri. Jika mereka cocok, mereka secara digital menandatangani bahwa mereka menyetujui identitas *server* ke klien.

Web App Manifest

Web app manifest adalah *file* JSON sederhana yang memberi tahu *browser* tentang aplikasi *web* Anda dan bagaimana seharusnya berperilaku ketika 'diinstal' pada perangkat seluler atau *desktop* pengguna (Gaunt & Kinlan, 2018).

Service Worker

Service worker adalah *script* yang dijalankan oleh *browser* di latar belakang atau *background*, terpisah dari halaman *web*, berfungsi untuk membuka pintu ke fitur yang tidak memerlukan halaman *web* atau interaksi pengguna. Sekarang, *service worker* sudah memiliki fitur seperti *push notification* dan *background sync*. *Service worker* digunakan untuk mencegah dan menangani permintaan jaringan, termasuk mengelola *cache* secara terprogram (Gaunt, 2018).

2.2.6 Metode Pengujian

2.2.6.1 *Black-box Testing*

Black-box testing merupakan pengujian yang berfokus terhadap persyaratan dari perangkat lunak yang memungkinkan penguji untuk mendapatkan *set* kondisi *input* yang secara keseluruhan melakukan persyaratan fungsional untuk sebuah program (Pressman, 2010). Dalam kategori yang diuji *black-box testing* antara lain fungsi yang tidak sesuai dengan fungsional ataupun fungsi yang hilang dan antara lain antarmuka, struktur data, perilaku (*behavior*) dan inisialisasi dan pemutusan. *Black-box testing* bertujuan untuk menemukan kesalahan-kesalahan fungsi yang tidak benar atau belum ada, kesalahan antarmuka, kesalahan struktur data atau akses *database*, kesalahan kinerja, kesalahan inisialisasi dan kesalahan terminasi.

Black-box testing, disebut juga sebagai *behavioural testing*, dikhususkan pada kebutuhan fungsional perangkat lunak. Teknik *black-box testing* digunakan untuk mengetahui kesesuaian fungsi tertentu dari suatu produk dengan tujuan awal didesainnya produk tersebut. Pengujian *black-box* dilaksanakan untuk menemukan *error* ketika skrip dieksekusi. *Black-box testing* melakukan pencarian *error* dalam kategori (Pressman, 2010):

1. Fungsi yang salah atau fungsi yang tidak lengkap.
2. Antar muka yang *error*.
3. *Error* dalam melakukan akses *database* eksternal atau akses data struktur.
4. *Error* dalam *behavior* atau kinerja.
5. *Error* dalam inisialisasi dan *termination*.

Menurut (Pressman, 2010) *black-box testing* tidak digunakan untuk menggantikan *white-box testing*, namun merupakan pendekatan yang digunakan untuk melengkapi *white-box testing* dengan menemukan *error* yang berada pada kelas yang berbeda dari *white-box testing*. *Black-box testing* cenderung dilaksanakan pada langkah terakhir karena pengujian ini tidak mempedulikan struktur kontrol.

Black-box testing dilakukan dengan membuat beberapa *test case* yang sudah didesain untuk menemukan *error* pada level validasi perangkat lunak. Di setiap *case* yang diberikan, diharapkan dapat menemukan *error* sebanyak mungkin dengan waktu dan usaha seminimal mungkin. Setiap *test case* dapat dibuat agar dapat melakukan uji coba terhadap setiap fungsi yang dibuat, kemudian akan divalidasi apakah telah beroperasi sepenuhnya dan disaat yang berasamaan dilakukan pencarian *error* yang mungkin terjadi. *Black-box testing* dilakukan pada bagian antarmuka dengan memberikan sekumpulan kondisi *input* yang akan dijalankan secara keseluruhan dalam kebutuhan fungsi program.

Menurut (Guru99, 2019b) berikut ini adalah beberapa strategi yang digunakan dalam *black-box testing*:

a) *Equivalence Class Testing*: Hal ini digunakan untuk meminimalkan jumlah kasus uji yang mungkin ke tingkat optimal sambil mempertahankan cakupan uji yang wajar.
b) *Boundary Value Testing*: Pengujian nilai batas difokuskan pada nilai pada batas. Teknik ini menentukan apakah rentang nilai tertentu dapat diterima oleh sistem atau tidak. Ini sangat berguna dalam mengurangi jumlah kasus uji. Ini sebagian besar cocok untuk sistem di mana *input* berada dalam rentang tertentu.
c) *Decision Table Testing*: Sebuah tabel keputusan menempatkan penyebab dan efeknya dalam matriks. Ada kombinasi unik di setiap kolom.

2.2.6.2 *White-box Testing*

White-box testing adalah pendekatan dalam pengujian program dimana tes didasarkan pada pengetahuan tentang struktur program dan komponennya (Pressman, 2010). Menurut (Guru99, 2019a) *basis path testing* merupakan salah satu teknik dari *whitebox testing* yang dapat menghasilkan *logical complexity* (kompleksitas logis) dari rancangan dengan cara prosedural dan menggunakan pengukuran ini sebagai dasar kumpulan jalur yang akan dieksekusi. *Test case* yang dihasilkan dapat menjamin eksekusi setiap jalur program paling tidak dilakukan sekali selama pengujian.

Cyclomatic Complexity merupakan *software metric* yang berfungsi untuk mengukur kompleksitas suatu program dengan mengukur jalur independen melalui kode program. Jalur independen merupakan jalur yang paling tidak memiliki 1 *edge* yang belum pernah dilewati sebelumnya oleh jalur yang lain. Notasi *flowgraph* untuk program telah ditetapkan. Beberapa *node* dihubungkan melalui *edges*. Dibawah ini adalah *flow* diagram untuk pernyataan seperti *if-else, While, until, dan normal sequence*. Notasi *flowgraph* digambarkan pada Gambar 2.13.

Kompleksitas suatu program dapat dicari dengan menggunakan rumus:

$$V(G) = E - N + 2$$

E = Jumlah *edge.*

N = Jumlah *node.*

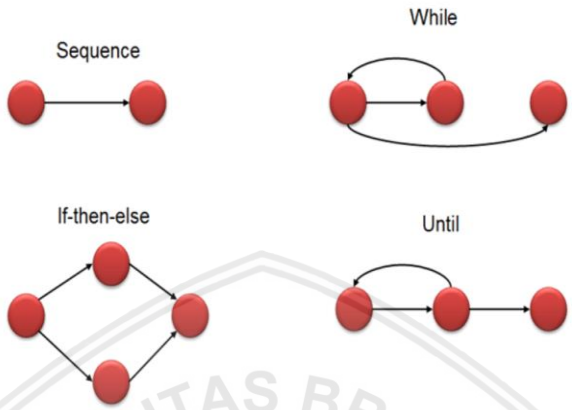
Atau dengan menggunakan rumus:

$$V(G) = P + 1$$

P = Jumlah *predicate node* (node yang berisi kondisi).

Cyclomatic Complexity juga dapat dihitung secara manual pada suatu program yang bersifat sederhana. Diperlukan sebuah *Automated tools* jika program yang akan dihitung kompleksitasnya merupakan program yang sangat kompleks yang melibatkan banyak *flow graph*. Pada tabel dibawah ini merupakan

tabel yang memberikan gambaran kompleksitas dan arti dari nilai V(G). Nilai *Complexity Number* itu sendiri memiliki makna disetiap nilainya. Makna itu berdasarkan struktur dan penulisannya, tingkat *testability*-nya dan tingkat biaya dan usahanya. Berikut ini pada Tabel 2.1 dijelaskan makna dari setiap nilai dari *Complexity Number*.



Gambar 2.13 Notasi *flow graph*

(Sumber: Guru99, 2019)

Tabel 2.1 Makna *complexity number*

Complexity Number (Nilai Kompleksitas)	*Meaning* (Arti)
1 – 10	Struktur dan penulisan dapat dikatakan baik. Bersifat h*igh testability.* Biaya dan usaha rendah.
10 – 20	Kode dapat dikatakan kompleks. Bersifat *medium testability.* Biaya dan usaha medium.
20 – 40	Kode dapat dikatakan sangat kompleks. Bersifat *low testability.* Biaya dan usaha tinggi.
>40	Tidak semuanya bisa diuji. Biaya dan usaha sangat tinggi.

(Sumber: Guru99, 2019)

2.2.6.3 *Integration Testing*

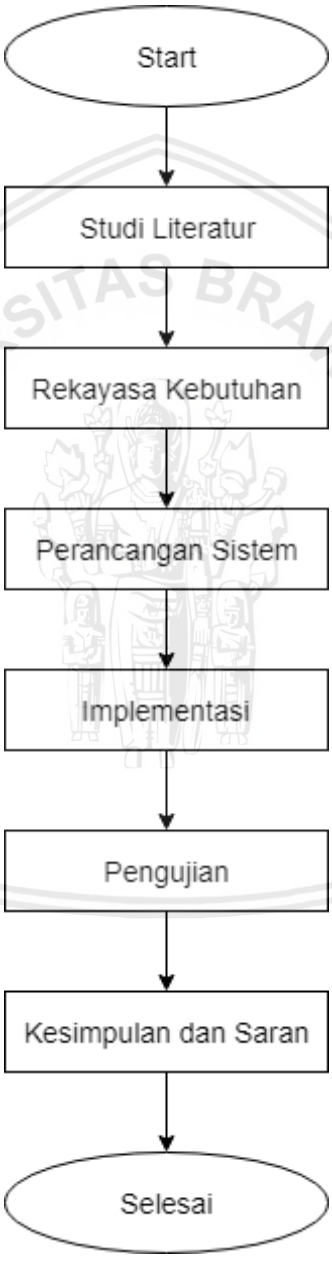
Pengujian integrasi adalah teknik sistematis untuk membangun struktur program sementara pada saat yang sama melakukan tes untuk mengungkap kesalahan yang terkait dengan *interfacing*. Tujuannya adalah untuk mengambil komponen yang diuji unit dan membangun struktur program yang telah ditentukan oleh desain (Pressman, 2010). Dalam pengujian integrasi ada dua cara untuk melakukannya, yaitu secara non-inkremental atau bing bang dan secara inkremental. Integrasi non-inkremental dilakukan setelah dengan cara semua modul digabung seluruhnya. Sedangkan integrasi inkremental dilakukan setiap ada modul baru yang dibangun.

2.2.6.4 *Compatibility Testing*

Compatibility testing memastikan antarmuka yang konsisten secara fungsional di seluruh *platform* perangkat keras dan perangkat lunak. Misalnya, antarmuka tipe *Windows* mungkin berbeda secara *visual* tergantung pada lingkungan implementasi, tetapi perilaku pengguna dasar yang sama harus menghasilkan hasil yang sama terlepas dari standar antarmuka klien. (Pressman, 2010).

BAB 3 METODOLOGI

Bab ini berisi mengenai metode penelitian yang digunakan dalam proses pengembangan Sistem Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon. Beberapa tahapan dalam metodologi penelitian yang digunakan yaitu studi literatur, rekayasa kebutuhan, perancangan sistem, implementasi, pengujian dan pengambilan kesimpulan dan saran. Metodologi yang dilakukan dalam penelitian ini digambarkan pada Gambar 3.1.



Gambar 3.1 Diagram alir metodologi penelitian

3.1 Studi Literatur

Studi literatur merupakan penjelasan dari dasar teori yang digunakan dalam penelitian yang pernah dilakukan sebelumnya, buku, jurnal, dan beberapa literatur yang berasal dari *internet* untuk mendukung penulisan tugas akhir.

Berikut ini merupakan teori-teori pendukung yang digunakan pada penelitian yang saya lakukan:

1. PT. Kereta Api Indonesia.
2. Rekayasa perangkat lunak.
3. *Software development life cycle waterfall model.*
4. *Progressive web application.*
5. Pengujian perangkat lunak.
6. *White-box testing.*
7. *Black-box testing.*
8. *Integration testing.*
9. *Compatibility testing.*
10. *Model-view-controller*.

3.2 Rekayasa Kebutuhan

Rekayasa kebutuhan dilakukan dengan dua tahap yaitu elisitasi kebutuhan dan analisis kebutuhan.

a. Elisitasi kebutuhan

Elisitasi kebutuhan untuk penelitian ini dilakukan di unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon. Pengumpulan data dilakukan dengan cara wawancara terstrukur dan observasi. Wawancara terstruktur sendiri yaitu dengan memberikan pertanyaan-pertanyaan pada responden. Pada wawancara yang dilakukan ini saya mewawancarai asisten manajer dari unit IT di PT. Kereta Api Indonesia Daerah Operasi III Cirebon yaitu bapak Awal Arifianto.

b. Analisis kebutuhan

Dari hasil wawancara dan observasi yang dilakukan akan didapatkan permasalahan yang terjadi pada prosedur pelaporan gangguan. Permasalahan yang didapat itu kemudian dianalisis sehingga akan mendapatkan kebutuhan-kebutuhan yang nantinya harus dimiliki oleh sistem yang akan dibangun. Tahapan ini sangat penting, karena jika ditemukan kesalahan pada pendefinisian kebutuhan maka akan mempengaruhi semua tahapan setelah analisis kebutuhan karena terjadinya perbedaan antara apa yang dibutuhkan oleh *user* dengan *software* yang nantinya dihasilkan.

Pada pembuatan Sistem Pelaporan Gangguan di Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon ini harus dilakukan studi literatur. Studi

literatur yang digunakan adalah hasil wawancara yang telah dihasilkan dari proses elisitasi kebutuhan dan referensi-referensi baik dari buku, jurnal ilmiah, *internet* dan laporan ilmiah yang mendukung penelitian yang akan penulis kerjakan. Setiap kebutuhan fungsional diberikan kode yang akan dijelaskan pada tabel 3.1.

Tabel 3.1 Kode kebutuhan dan definisinya

Kode	Definisi
SPG_F_XX_YY	Pendefinisian kebutuhan fungsional.
SPG_NF_XX_YY	Pendefinisian kebutuhan non-fungsional.

Keterangan:

SPG : singkatan dari Sistem Pelaporan Gangguan.

F : menunjukkan kebutuhan fungsional.

NF : menunjukkan kebutuhan non-fungsional.

XX : menunjukkan nomor kebutuhan.

YY : menunjukkan nomor spesifikasi dari sebuah kebutuhan.

Didalam penelitian ini penulis menggunakan pemodelan dengan pendekatan *object-oriented*, oleh karena itu dalam pemodelan ini nantinya akan dihasilkan *use-case diagram* dan *use-case scenario*.

3.3 Perancangan Sistem

Perancangan sistem merupakan tahapan yang dilakukan untuk mengubah kebutuhan menjadi suatu model perangkat lunak. Perancangan sistem dilakukan setelah selesai melakukan tahapan rekayasa kebutuhan. Pada tahapan perancangan ini pertama penulis akan membuat rancangan arsitektur dari sistem pelaporan gangguan ini. Kemudian dalam merancang sistem pelaporan gangguan ini penulis menggunakan pendekatan *object-oriented*, oleh karena itu perancangan sistem akan menghasilkan *sequence diagram*, *class diagram*, perancangan komponen, kemudian dilanjutkan dengan membuat *conceptual data model* (CDM) yang merupakan hasil dari perancangan *database* dan perancangan antarmuka. *Sequence diagram* dibuat berdasarkan *use case* dan *use case scenario* yang telah dibuat pada tahap rekayasa kebutuhan.

3.4 Implementasi

Implementasi sistem merupakan tahapan pembangunan sistem yang mengacu pada hasil dari perancangan sistem. Proses implementasi meliputi pemilihan bahasa pemrograman yang digunakan dan melakukan proses pengkodean atau *coding*. *Coding* merupakan sebuah proses yang dilakukan untuk merubah hasil perancangan sistem yang telah dibuat menjadi bentuk kode program. Pola perancangan yang saya gunakan adalah *Model-View-Controller*

(MVC). Sistem akan dibuat pada *platform web*, menggunakan *framework CodeIgniter* dan menerapkan teknologi *Progressive Web Application* dengan menambahkan *Service Workers* dan *Web App Manifest* serta membuat sistem pelaporan gangguan ini dapat berjalan diatas HTTPS. Bahasa yang digunakan adalah bahasa pemrograman PHP untuk membuat *back-end* dan menggunakan bahasa pemrograman javascript, html dan css untuk membuat *front-end*.

Dalam menggunakan pola perancangan MVC ini terdiri dari 3 modul, yaitu *Model*, *View*, dan *Controller*. *View* merupakan antarmuka dari sistem yang dapat dilihat oleh pengguna sistem. *Model* merupakan pengelola *database* yang berisi memasukkan, membaca, mengedit dan menghapus data yang ada pada *database*. Dan *controller* yang merupakan penghubung antara *model* dan *view*.

3.5 Pengujian

Tujuan dari tahap pengujuan sistem adalah untuk mengetahui apakah sistem yang sudah kita buat sudah sesuai dengan apa yang diharapkan oleh pengguna. Apabila ditemukan kesalahan ataupun kekurangan maka akan dilakukan perbaikan pada sistem yang telah dibuat.

Pada penelitian ini pengujian yang akan dilakukan yaitu *verification testing, validation testing* dan pengujian *compatibility*. Pada *verification testing* akan dilakukan pengujian unit dan pengujian integrasi yang akan dilakukan dengan menggunakan *white-box testing* jenis *basis path testing*. Pada *basis path testing* akan dilakukan pengujian terhadap kode program berdasarkan algoritme yang terdapat pada setiap fungsi di setiap kelas. Pengujian unit pada penelitian ini dilakukan dengan tiga sampel uji. Pengujian integrasi dilakukan untuk menguji keterhubungan antar kelas. Pada penelitian ini pengujian integrasi akan menggunakan pendekatan *top-down*. Pada pengujian integrasi juga akan dilakukan dengan satu sampel uji. Pengujian validasi atau *validation testing* akan dilakukan dengan menggunakan *black-box testing* dengan teknik *equivalence partitioning*. Dengan teknik ini, kebutuhan yang telah didefinisikan pada tahap analisis kebutuhan akan diuji dari masukan yang diberikan dan hasil keluaran sistem, apakah sistem telah berjalan sesuai dengan yang diharapkan. Untuk *compatibility testing* akan digunakan kakas bantu yaitu perangkat lunak *SortSite* 5. Pengujian c*ompatibility* akan dilakukan pada *browser edge, firefox, safari, opera* dan *chrome*. Pengujian *compatibility* akan menggunakan beberapa parameter yaitu menggunakan tag HTML dan atribut yang kompatibel pada browser, menggunakan aturan CSS yang didukung oleh browser, tidak ada *element* yang hilang, dan tidak ada properti *JavaScript* dan *DOM* dan fungsi yang memicu bug pada browser.

3.6 Kesimpulan dan Saran

Proses pengambilan kesimpulan dilakukan setelah semua tahapan pengembangan perangkat lunak telah selesai dilakukan. Kesimpulan didapatkan dari hasil pengujian dan analisis terhadap sistem yang telah dibangun. Saran ditulis untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan

penulisan dan juga untuk memberikan pertimbangan untuk melakukan pengembangan lebih lanjut terhadap perangkat lunak yang telah dibuat.

BAB 4 REKAYASA KEBUTUHAN

Rekayasa kebutuhan merupakan tahapan yang dilakukan pertama kali dalam mengembangkan perangkat lunak. Pada bagian ini akan menjelaskan tahapan dalam menentukan kebutuhan pada sistem yang akan dibuat dan menentukan aktor yang akan berinteraksi dengan sistem pada saat sistem telah selesai dikembangkan.

4.1 Elisitasi Kebutuhan

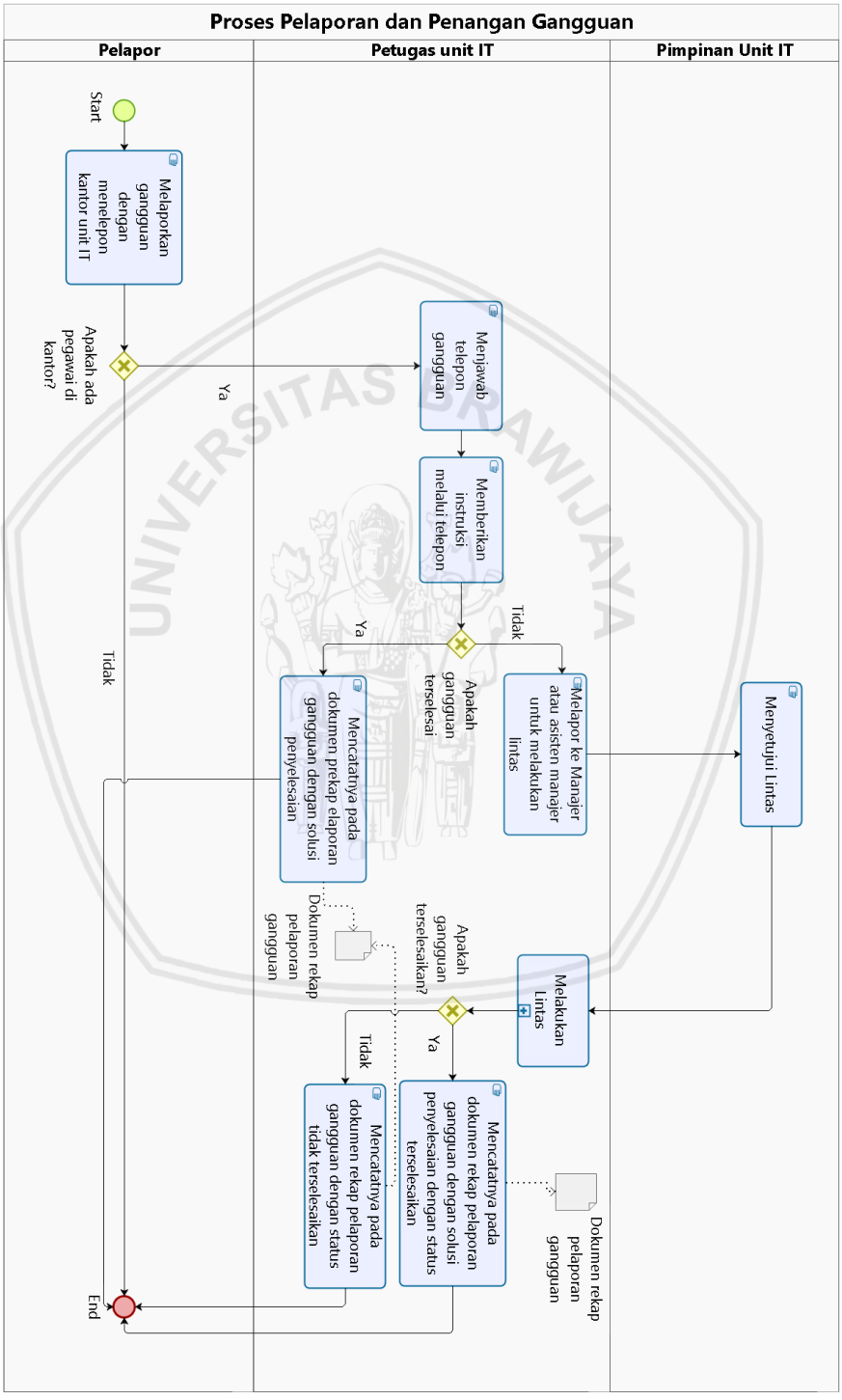
Elisitasi kebutuhan merupakan proses pertama dalam rekayasa kebutuhan. Tujuan dilakukannya elisitasi kebutuhan adalah untuk memahami masalah yang terjadi pada proses pelaporan gangguan di unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon. Kemudian, dari permasalahan yang didapatkan akan diajukan alternatif solusi yang dapat membantu menyelesaikan permasalahan yang terjadi.

Teknik elisitasi yang digunakan dalam penelitian ini adalah wawancara dan observasi. Proses wawancara dilakukan untuk mengetahui permasalahan yang terjadi, siapa saja orang yang terlibat, dan proses bisnis yang terjadi pada unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon yang hasilnya akan didapatkan fitur-fitur dan aktor-aktor yang terlibat dalam sistem. Sedangkan, proses observasi dilakukan dengan mengamati secara langsung proses yang terjadi di unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon yang nantinya juga akan digunakan untuk mendapatkan fitur-fitur yang digunakan dalam sistem.

Berdasarkan hasil wawancara dan observasi didapatkan proses bisnis dari pelaporan dan penanganan gangguan yang ada pada unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon. Proses bisnis pelaporan dan penanganan gangguan sebelum adanya sistem yang akan dibangun bisa dilihat pada Gambar 4.1. Proses pelaporan dan penanganan dimulai dengan pelapor melaporkan gangguan dengan cara menelepon ke telepon kantor. Bila diangkat petugas unit IT akan memberikan instruksi terlebih dahulu. Jika instruksi tersebut belum dapat mengatasi gangguannya, maka petugas unit IT akan melapor untuk melakukan *lintas* pada pimpinan unit IT (manajer atau asisten manajer) yang sedang berada di kantor.

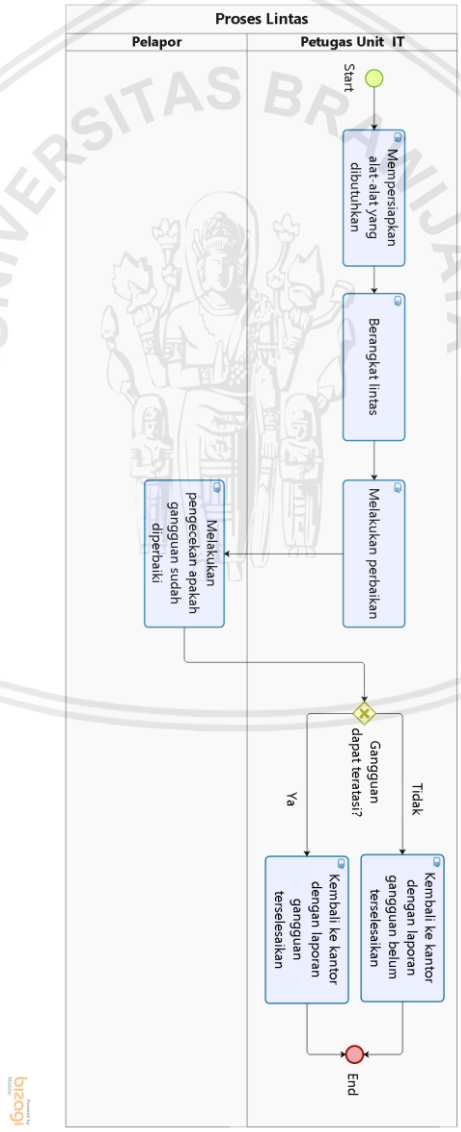
Setelah mendapatkan izin dari manajer atau asisten manajer, petugas unit IT akan menuju tempat gangguan dan memperbaikinya. Setelah gangguan selesai diperbaiki maka akan dicatat pada dokumen rekap pelaporan gangguan. Bila gangguan dapat terselesaikan di dokumen rekap pelaporan gangguan disertakan solusi yang digunakan dalam mengatasi gangguan tersebut. Sedangkan apabila gangguan tidak dapat terselesaikan, pada dokumen pelaporan gangguan dituliskan status tidak terselesaikan. Apabila saat pelapor melaporkan gangguan namun tidak ada yang mengangkat telepon itu, berdasarkan wawancara yang dilakukan, pelapor akan menelepon lagi ke kantor unit IT hingga laporan gangguan diterima oleh unit IT. Masalah yang terjadi disini yaitu saat tidak ada satupun petugas unit IT yang berada di kantor maka tidak ada yang mengangkat telepon

sehingga gangguan tidak dapat segera terselesaikan. Yang kedua adalah kurangnya informasi tentang petugas yang sedang bertugas. Yang ketiga adalah masalah ketika suatu petugas ingin pergi melakukan *lintas* namun tak ada manajer atau asisten manajer sehingga petugas tersebut tidak dapat melapor jika ingin *lintas*.



Gambar 4.1 Proses bisnis *as-is* pelaporan dan penanganan gangguan

Berdasarkan hasil wawancara juga didapatkan proses bisnis pada kegiatan *lintas* yang dilakukan oleh petugas unit IT. Prosesnya dimulai dengan mempersiapkan alat-alat yang sekiranya dibutuhkan. Setelah itu petugas unit IT berangkat *lintas* menuju tempat terjadinya gangguan. Setelah sampai di tempat gangguan petugas unit IT akan memperbaiki gangguan. Pelapor akan melakukan pengecekan apakah gangguan dapat terselesaikan atau tidak. Jika gangguan terselesaikan maka petugas unit IT akan kembali ke kantor dengan laporan gangguan berhasil terselesaikan, sebaliknya jika gangguan belum terselesaikan maka petugas IT akan kembali ke kantor dengan laporan gangguan belum terselesaikan. Masalah yang terjadi disini adalah belum adanya pendataan barang atau perangkat keluar dari gudang penyimpanan yang telah digunakan dalam penanganan gangguan. Gambar 4.2 menggambarkan proses bisnis *as-is* pelaksanaan *lintas*.



Gambar 4.2 Proses bisnis *as-is* pelaksanan *lintas*

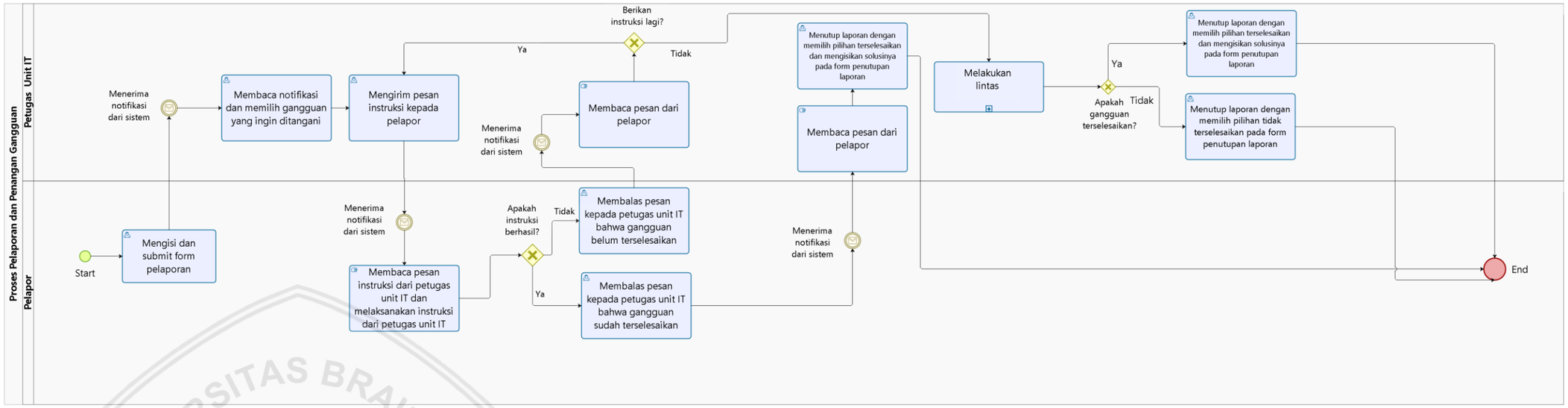
4.2 Analisis Kebutuhan

Berdasarkan hasil elisitasi kebutuhan akan didapatkan proses bisnis *to-be* atau proses bisnis setelah adanya sistem, identifikasi aktor dan daftar kebutuhan fungsional yang nantinya akan dimodelkan menjadi *use-case diagram* dan *use-case scenario*. Proses bisnis pelaporan dan penanganan gangguan setelah terdapat sistem pada dilihat pada Gambar 4.3.

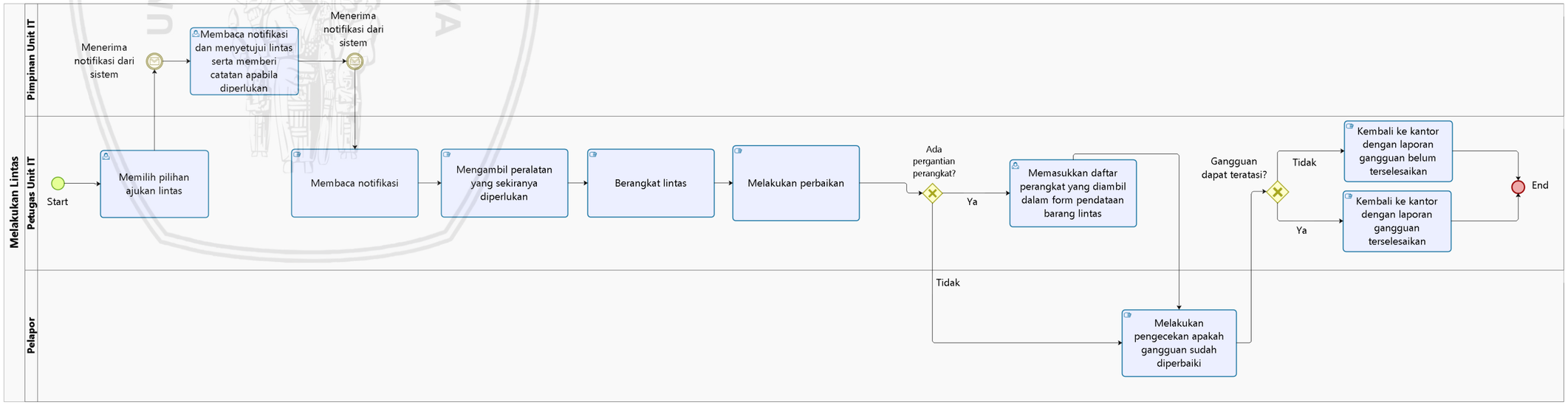
Proses pelaporan dimulai dengan pelapor yang telah melakukan *login* ke dalam sistem mengisi *form* pelaporan. Kemudian sistem akan menyimpan laporan tersebut ke dalam *database* dan akan memberikan notifikasi kepada petugas unit IT yang sedang bertugas pada waktu laporan masuk. Petugas unit IT yang sedang bertugas pada hari itu akan memilih gangguan yang akan ditangani yang nantinya akan memberikan notifikasi kepada pelapor bahwa gangguan yang dilaporkan sedang ditangani. Lalu petugas unit IT akan memberikan instruksi melalui pesan. Apabila instruksi tersebut sudah dapat menyelesaikan gangguan maka petugas unit IT akan menutup laporan dengan menuliskan solusinya pada *form* penutupan laporan. Apabila dengan instruksi yang diberikan gangguan belum terselesaikan maka petugas unit IT akan melakukan *lintas*.

Gambar 4.4 menggambarkan proses bisnis *to-be* melakukan *lintas*. Prosesnya dimulai dengan memilih pilihan ajukan untuk *lintas* yang nantinya sistem akan memberitahukan manajer atau asisten manajer dengan notifikasi bahwa petugas tersebut akan melakukan *lintas*. Setelah disetujui oleh pimpinan unit IT, petugas tersebut menerima notifikasi bahwa *lintas* sudah disetujui. Setelah itu petugas unit IT mempersiapkan alat dan perangkat yang sekiranya dibutuhkan langsung menuju tempat gangguan dan mengatasi gangguan. Apabila ada perangkat yang diganti maka petugas unit IT akan mendata perangkat tersebut pada *form* pendataan barang *lintas*.

Setelah itu pelapor akan melakukan pengecekan apakah gangguan telah teratasi atau belum teratasi. Jika gangguan terselesaikan maka petugas unit IT akan kembali ke kantor dengan laporan gangguan telah terselesaikan, sebaliknya jika gangguan belum terselesaikan maka petugas unit IT akan kembali ke kantor dengan laporan gangguan tidak terselesaikan. Apabila gangguan telah teratasi maka petugas unit IT akan menutup laporan dengan menuliskan solusinya pada *form* penutupan laporan dengan memilih status terselesaikan, sebaliknya jika gangguan tidak teratasi maka petugas unit IT akan menutup laporan dengan memilih status tidak terselesaikan.



Gambar 4.3 Proses bisnis *to-be* pelaporan dan penanganan gangguan



Gambar 4.4 Proses bisnis *to-be* pelaksanaan *lintas*

4.2.1 Identifikasi Aktor

Berdasarkan hasil dari elisitasi kebutuhan, aktor yang terdapat pada sistem adalah pengguna, pimpinan unit IT, petugas unit IT, dan pelapor. Berikut ini adalah hasil identifikasi aktor ditampilkan pada Tabel 4.1.

Tabel 4.1 Tabel identifikasi aktor

Aktor	Deskripsi
Pengguna	Pengguna merupakan orang yang belum terautentikasi oleh sistem.
Pimpinan Unit IT	Pimpinan unit IT terdiri dari manajer dan asisten manajer. Pimpinan unit IT di dalam sistem memiliki peran untuk menambahkan akun, melihat daftar akun, melihat jadwal kerja, membuat jadwal kerja, melihat permohonan *lintas*, memberi izin *lintas*, melakukan manajemen gudang penyimpanan dan melihat daftar dan grafik riwayat gangguan yang terjadi.
Petugas Unit IT	Petugas unit IT merupakan orang yang bertugas untuk mengatasi laporan gangguan yang masuk ke unit IT. Pada sistem petugas unit IT memiliki peran diantaranya untuk melihat laporan gangguan yang masuk, menangani laporan gangguan yang masuk, melihat laporan gangguan yang ditangani, mengajukan *lintas*, melihat daftar *lintas* yang telah diajukan, melihat *detail* laporan gangguan, mengirim pesan, menambahkan barang pada daftar barang yang digunakan untuk *lintas*, melakukan manajemen gudang penyimpanan, menutup laporan dan melihat daftar dan grafik riwayat gangguan yang terjadi.
Pelapor	Pelapor merupakan pegawai PT. Kereta Api Indonesia Daerah Operasi III Cirebon selain pegawai unit IT. Dalam sistem pelapor memiliki peran untuk melihat daftar petugas unit IT yang sedang aktif, membuat laporan

	gangguan, mencetak *e-ticket* pelaporan gangguan, melihat laporan gangguan yang telah terlaporkan, melihat *detail* laporan gangguan, mengirim pesan, dan membatalkan pelaporan gangguan.

4.2.2 Daftar Kebutuhan Fungsional

Kebutuhan fungsional merupakan fungsi yang harus dimiliki oleh sistem yang akan dibangun. Kebutuhan fungsional pada sistem yang akan dibangun didapat dari hasil wawancara dengan asisten manajer unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon dan proses observasi yang telah dilakukan. Kebutuhan yang telah terdaftarkan telah divalidasi untuk menunjukkan kebutuhan fungsional dalam sistem telah benar dan lengkap sesuai dengan kebutuhan aktor yang terlibat didalam sistem dan diverifikasi menunjukkan bahwa kebutuhan yang telah didefinisikan mudah diidentifikasi, lengkap, dan terbebas dari ambigu. Berikut ini kebutuhan fungsional pada Sistem Pelaporan Gangguan di Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon dijelaskan pada Tabel 4.2.

Tabel 4.2 Daftar kebutuhan fungsional

No	*Use Case*	Aktor	Spesifikasi	Kode
1.	*Login*	Pengguna	Sistem harus dapat melakukan autentikasi terhadap pengguna sebagai Pimpinan unit IT, Petugas unit IT, atau Pelapor.	SPG_F_01
			Halaman *login* terdiri dari *field* untuk NIPP dan *password* dan satu tombol *"Login"*.	SPG_F_01_01
			Field NIPP hanya bisa diisi dengan angka.	SPG_F_01_02
			Jika *field* NIPP dan *password* tidak diisi maka akan menampilkan pesan peringatan.	SPG_F_01_03
			Jika NIPP atau *password* tidak terdaftar akan menampilkan pesan peringatan.	SPG_F_01_04

No	*Use Case*	Aktor	Spesifikasi	Kode
2.	Menambahkan Akun	Pimpinan Unit IT	Sistem harus dapat menyediakan fungsi untuk menambahkan akun baru di dalam sistem.	SPG_F_02
			Field yang ada pada halaman tambah akun adalah NIPP, nama lengkap, unit kerja, jabatan, kedudukan, nomor telepon, *email*, tipe akun dan satu tombol bertuliskan "Tambahkan Akun"	SPG_F_02_01
			Pada *field* kedudukan terdapat pilihan kantor DAOP, stasiun arjawinangun, stasiun babakan, stasiun bangoduwa, stasiun brebes, stasiun bulakamba, stasiun cangkring, stasiun cikaum, stasiun cilegeh, stasiun cipunegara, stasiun cirebon, stasiun cirebon prujakan, stasiun haurgeulis, stasiun jatibarang, stasiun kadokangabus, stasiun kertasemaya, stasiun ketanggungan, stasiun larangan, stasiun losari, stasiun luwung, stasiun pabuaran, stasiun pasirbungur, stasiun pegadenbaru, stasiun pringkasap, stasiun sindanglaut, stasiun songgom, stasiun tanjung, stasiun tanjungrasa, stasiun tegalsari, stasiun terisi, dan stasiun waruduwur.	SPG_F_02_02

No	*Use Case*	Aktor	Spesifikasi	Kode
			Pada *field* akun terdapat 3 pilihan yaitu Pimpinan unit IT, Petugas unit IT dan Pelapor.	SPG_F_02_03
			Semua *field* wajib terisi, jika tidak terisi akan menampilkan pesan "*Please fill out this field*".	SPG_F_02_04
3.	Menambahkan Akun Dengan Impor *File*	Pimpinan Unit IT	Sistem harus dapat menyediakan fungsi untuk menambahkan akun dengan cara impor *file*.	SPG_F_03
			File yang dapat diimpor adalah *file* yang berekstensi .xls dan .xlsx	SPG_F_03_01
			Pada halaman menambah akun dengan impor terdapat *link download* format *file*.	SPG_F_03_02
			Pada halaman menambah akun dengan impor terdapat tombol yang bertuliskan "Tambahkan Akun".	SPG_F_03_03
			Pada halaman menambah akun dengan impor terdapat tabel nomor kedudukan.	SPG_F_03_04
			Jika ekstensi *file* bukan .xls atau .xlsx maka akan menampilkan pesan "*File* yang dimasukkan harus memiliki ekstensi .xlsx,.xls".	SPG_F_03_05
4.	Melihat Daftar Akun	Pimpinan Unit IT	Sistem harus dapat menyediakan fungsi untuk menampilkan daftar akun yang ada di dalam sistem.	SPG_F_04
			Kolom yang ditampilkan dalam tabel adalah	SPG_F_04_01

No	*Use Case*	Aktor	Spesifikasi	Kode
			nomor, NIPP, nama lengkap, unit kerja, jabatan, nomor telepon, tipe *user*, *email* dan tombol "*delete*".	
			Pada halaman ini terdapat satu *dropdown field* yang digunakan untuk memilih berapa jumlah data yang ditampilkan.	SPG_F_04_02
			Pada halaman ini juga terdapat *field* yang digunakan untuk mencari data berdasarkan NIPP, nama lengkap, unit kerja, jabatan, nomor telepon, tipe *user* atau *email.*	SPG_F_04_03
5.	Menghapus Akun	Pimpinan Unit IT	Sistem harus dapat menyediakan fungsi untuk menghapus akun yang sudah terdaftar di dalam sistem.	SPG_F_05
6.	Melihat Jadwal Kerja	Pimpinan Unit IT, Petugas Unit IT	Sistem harus dapat menyediakan fungsi untuk menampilkan jadwal bulanan pegawai unit IT.	SPG_F_06
			Pada tabel akan diberi warna kuning jika status kerja "K", warna merah jika status kerja "L" dan warna hijau jika status kerja "P/R".	SPG_F_06_01
			Pada halaman jadwal kerja terdapat satu *field* bulan dan satu *button* yang digunakan untuk melakukan *filter* berdasarkan bulan.	SPG_F_06_02
			Data yang ditampilkan adalah NIPP, Nama dan tanggal pada bulan yang	SPG_F_06_03

No	*Use Case*	Aktor	Spesifikasi	Kode
			dipilih yang berisi status kerja dari setiap pegawai unit IT.	
			Jika aktor yang terautentikasi merupakan pimpinan unit IT maka ada satu tombol yang bertuliskan "Tambah Jadwal".	SPG_F_06_04
7.	Menambahkan Jadwal Kerja	Pimpinan Unit IT	Sistem harus menyediakan fungsi untuk menambahkan jadwal kerja bulanan.	SPG_F_07
			Jika berhasil menambahkan jadwal kerja akan menampilkan pesan "Berhasil membuat jadwal kerja".	SPG_F_07_01
8.	Melihat Daftar Permohonan *Lintas*	Pimpinan Unit IT	Sistem harus menyediakan fungsi untuk menampilkan permohonan *lintas* yang masuk.	SPG_F_08
			Kolom yang ditampilkan pada tabel adalah nomor, kode gangguan, NIPP petugas, lokasi gangguan, jenis gangguan dan tombol "izinkan".	SPG_F_08_01
			Pada halaman ini terdapat satu *dropdown field* yang digunakan untuk memilih berapa jumlah data yang ditampilkan.	SPG_F_08_02
			Pada halaman ini juga terdapat *field* yang digunakan untuk mencari data berdasarkan kode gangguan, NIPP petugas,	SPG_F_08_03

No	*Use Case*	Aktor	Spesifikasi	Kode
			lokasi gangguan atau jenis gangguan.	
9.	Memberi Izin Permohonan *Lintas*	Pimpinan Unit IT	Sistem harus menyediakan fungsi untuk memberi izin terhadap permohonan *lintas* yang masuk.	SPG_F_09
			Ketika tombol izinkan ditekan maka akan muncul modal izinkan *lintas.*	SPG_F_09_01
			Di dalam modal izinkan *lintas* terdapat satu *field* yang digunakan untuk menambahkan catatan *lintas.*	SPG_F_09_02
			Sistem akan mengirimkan notifikasi ketika *lintas* sudah disetujui.	SPG_F_09_03
10.	Melihat Daftar Petugas Aktif	Pelapor	Sistem harus menyediakan fungsi untuk melihat daftar petugas yang sedang aktif atau bertugas.	SPG_F_10
			Informasi yang ditampilkan adalah nama lengkap, NIPP, nomor telepon dan *email* petugas.	SPG_F_10_01
11.	Membuat Laporan Gangguan	Pelapor	Sistem harus menyediakan fungsi untuk membuat laporan gangguan.	SPG_F_11
			Pada halaman buat laporan gangguan harus terdapat *field* tipe gangguan, deskripsi gangguan dan lokasi gangguan.	SPG_F_11_01
			Pada *field* tipe gangguan terdapat 4 pilihan yaitu	SPG_F_11_02

No	*Use Case*	Aktor	Spesifikasi	Kode
			Gangguan Jaringan, Gangguan *Hardware*, Gangguan Operasional, dan Gangguan Perjalanan.	
			Pada *field* lokasi gangguan terdapat pilihan kantor DAOP, stasiun arjawinangun, stasiun babakan, stasiun bangoduwa, stasiun brebes, stasiun bulakamba, stasiun cangkring, stasiun cikaum, stasiun cilegeh, stasiun cipunegara, stasiun cirebon, stasiun cirebon prujakan, stasiun haurgeulis, stasiun jatibarang, stasiun kadokangabus, stasiun kertasemaya, stasiun ketanggungan, stasiun larangan, stasiun losari, stasiun luwung, stasiun pabuaran, stasiun pasirbungur, stasiun pegadenbaru, stasiun pringkasap, stasiun sindanglaut, stasiun songgom, stasiun tanjung, stasiun tanjungrasa, stasiun tegalsari, stasiun terisi, dan stasiun waruduwur.	SPG_F_11_03
12.	Mengunduh *E-Ticket* Pelaporan Gangguan	Pelapor	Sistem harus menyediakan fungsi untuk mengunduh *E-Ticket* Pelaporan Gangguan.	SPG_F_12
			E-Ticket terunduh dalam bentuk *file* berekstensi PDF.	SPG_F_12_01

No	*Use Case*	Aktor	Spesifikasi	Kode
			Pada *E-Ticket* terdapat informasi NIPP pelapor, nama lengkap, waktu lapor, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan serta terdapat *QR code* yang mengandung informasi kode gangguan.	SPG_F_12_02
13.	Melihat Daftar Laporan Gangguan yang Telah Terlaporkan	Pelapor	Sistem harus menyediakan fungsi untuk melihat daftar laporan gangguan yang telah dilaporkan oleh aktor.	SPG_F_13
			Pada halaman lihat laporan gangguan terlaporkan pada tabel terdapat kolom yang berisi nomor, kode gangguan, NIPP pelapor, NIPP petugas, jenis gangguan, lokasi gangguan, waktu lapor, status, tombol unduh *e-ticket*, tombol *detail*, dan tombol batalkan.	SPG_F_13_01
			Pada halaman ini terdapat satu *dropdown field* yang digunakan untuk memilih berapa jumlah data yang ditampilkan.	SPG_F_13_02
			Pada halaman ini juga terdapat *field* yang digunakan untuk mencari data berdasarkan kode gangguan, NIPP pelapor, NIPP petugas, jenis gangguan, lokasi gangguan, waktu lapor, atau status.	SPG_F_08_03

No	*Use Case*	Aktor	Spesifikasi	Kode
14.	Melihat Detail Laporan Gangguan yang Telah Terlaporkan	Pelapor	Sistem harus dapat menyediakan fungsi untuk melihat *detail* dari sebuah laporan gangguan yang telah terlaporkan oleh aktor.	SPG_F_14
			Pada halaman *detail* laporan gangguan terdapat bagian *detail* laporan gangguan yang menampilkan NIPP petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status dan solusi.	SPG_F_14_01
			Pada halaman *detail* laporan gangguan terdapat bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan.	SPG_F_14_02
			Pada halaman *detail* laporan gangguan terdapat bagian pendataan barang *lintas* yang memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan.	SPG_F_14_03
			Pada bagian pendataan barang *lintas* terdapat tabel dengan kolom nama barang dan kode barang.	SPG_F_14_04

No	*Use Case*	Aktor	Spesifikasi	Kode
15.	Membatalkan Laporan Gangguan	Pelapor	Sistem harus dapat menyediakan fungsi untuk membatalkan suatu laporan gangguan.	SPG_F_15
16.	Mengirim Pesan	Pelapor, Petugas Unit IT	Sistem harus dapat menyediakan fungsi untuk mengirimkan pesan.	SPG_F_16
17.	Melihat Daftar Laporan Gangguan Masuk	Petugas Unit IT	Sistem harus dapat menyediakan fungsi untuk menampilkan laporan gangguan yang masuk ke dalam unit IT.	SPG_F_17
			Pada tabel laporan gangguan masuk menampilkan kolom nomor, kode gangguan, NIPP pelapor, jenis gangguan, deskripsi, lokasi gangguan, waktu lapor, dan tombol “tangani”.	SPG_F_17_01
			Pada halaman ini terdapat satu *dropdown field* yang digunakan untuk memilih berapa jumlah data yang ditampilkan.	SPG_F_17_02
			Pada halaman ini juga terdapat *field* yang digunakan untuk mencari data berdasarkan kode gangguan, NIPP pelapor, jenis gangguan, deskripsi, lokasi gangguan, atau waktu lapor.	SPG_F_17_03
18.	Menangani Laporan Gangguan	Petugas Unit IT	Sistem harus dapat menyediakan fungsi untuk menangani laporan gangguan.	SPG_F_18
19.	Melihat Daftar Laporan	Petugas Unit IT	Sistem harus dapat menyediakan fungsi untuk	SPG_F_19

No	*Use Case*	Aktor	Spesifikasi	Kode
	Gangguan yang Ditangani		menampilkan laporan gangguan yang ditangani.	
			Pada tabel laporan gangguan yang ditangani harus terdapat kolom nomor, kode gangguan, NIPP pelapor, jenis gangguan, lokasi gangguan, waktu lapor, status dan tombol "*detail*".	SPG_F_19_01
			Pada halaman ini terdapat satu *dropdown field* yang digunakan untuk memilih berapa jumlah data yang ditampilkan.	SPG_F_18_02
			Pada halaman ini juga terdapat *field* yang digunakan untuk mencari data berdasarkan kode gangguan, NIPP pelapor, jenis gangguan, lokasi gangguan, atau waktu lapor.	SPG_F_18_03
20.	Melihat *Detail* Laporan Gangguan yang Ditangani	Petugas Unit IT	Sistem harus dapat menyediakan fungsi untuk melihat *detail* laporan bagi petugas.	SPG_F_20
			Pada halaman *detail* laporan gangguan terdapat bagian *detail* laporan gangguan yang menampilkan NIPP pelapor, NIPP Petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, solusi, tombol "Ajukan *Lintas*" dan tombol "Tutup Laporan".	SPG_F_20_01

No	*Use Case*	Aktor	Spesifikasi	Kode
			Pada halaman *detail* laporan gangguan terdapat bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan.	SPG_F_20_02
			Pada halaman *detail* laporan gangguan terdapat bagian pendataan barang *lintas* yang memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan.	SPG_F_20_03
			Pada bagian pendataan barang *lintas* terdapat tabel dengan kolom nama barang dan kode barang.	SPG_F_20_04
			Jika status laporan gangguan adalah "Melakukan *Lintas*" maka pada bagian pendataan barang terdapat tombol "Tambah Barang" yang digunakan untuk menambah daftar barang.	SPG_F_20_05
21.	Mengajukan *Lintas*	Petugas Unit IT	Sistem harus dapat menyediakan fungsi untuk mengajukan *lintas* kepada Pimpinan Unit IT.	SPG_F_21
22.	Menutup Laporan	Petugas Unit IT	Sistem harus dapat menyediakan fungsi untuk menutup laporan gangguan.	SPG_F_22
			Pada *form* tutup laporan terdapat *field* jenis status	SPG_F_22_01

No	*Use Case*	Aktor	Spesifikasi	Kode
			gangguan, solusi dan dua tombol yaitu simpan dan tidak.	
			Pada *field* status gangguan terdiri dari dua pilihan yaitu terselesaikan dan tidak terselesaikan.	SPG_F_22_02
			Field solusi akan tampil jika *field* status gangguan bernilai terselesaikan.	SPG_F_22_03
23.	Melihat Daftar *Lintas*	Petugas Unit IT	Sistem harus dapat menyediakan fungsi untuk melihat daftar *lintas* yang telah diajukan oleh aktor.	SPG_F_23
			Pada tabel daftar *lintas* harus ditampilkan kolom nomor, kode gangguan, NIPP pelapor, lokasi gangguan, jenis gangguan, status *lintas* dan catatan.	SPG_F_23_01
			Pada halaman ini terdapat satu *dropdown field* yang digunakan untuk memilih berapa jumlah data yang ditampilkan.	SPG_F_23_02
			Pada halaman ini juga terdapat *field* yang digunakan untuk mencari data berdasarkan kode gangguan, NIPP pelapor, lokasi gangguan, jenis gangguan, status *lintas* atau catatan.	SPG_F_23_03
24.	Menambahkan Barang yang Telah Digunakan	Petugas Unit IT	Sistem harus menyediakan fungsi untuk menambahkan barang yang telah digunakan.	SPG_F_24
			Jumlah barang yang dimasukkan tidak boleh lebih dari stok barang yang	SPG_F_24_01

No	*Use Case*	Aktor	Spesifikasi	Kode
			ada di gudang penyimpanan.	
			Dalam *form* penambahan barang yang telah digunakan terdapat *field* untuk mengisi jumlah barang dan untuk kode dari setiap barang.	SPG_F_24_02
25.	Melihat Daftar Barang di Gudang Penyimpanan	Pimpinan Unit IT, Petugas Unit IT	Sistem harus menyediakan fungsi untuk melihat daftar barang di gudang penyimpanan.	SPG_F_25
			Pada tabel harus menampilkan kolom nomor, nama barang, kuantitas barang, tombol *update* dan tombol hapus.	SPG_F_25_01
			Pada halaman manajemen gudang penyimpanan terdapat satu *dropdown field* yang digunakan untuk memilih berapa jumlah data yang ditampilkan.	SPG_F_24_02
			Pada halaman manajemen gudang penyimpanan juga terdapat *field* yang digunakan untuk mencari data berdasarkan nama barang atau kuantitas barang.	SPG_F_24_03
			Pada halaman manajemen gudang penyimpanan terdapat satu tombol yang bertuliskan "Tambah Barang".	SPG_F_24_04
26.	Menambah Barang di Daftar Barang	Pimpinan Unit IT, Petugas Unit IT	Sistem harus menyediakan fungsi untuk menambah barang pada daftar barang di gudang.	SPG_F_26

No	*Use Case*	Aktor	Spesifikasi	Kode
	Gudang Penyimpanan			
			Field yang harus tersedia adalah nama barang dan kuantitas barang serta tombol “simpan”.	SPG_F_26_01
			Nilai minimal *field* kuantitas adalah 0.	SPG_F_26_02
27.	Meng*update* Barang di Daftar Barang Gudang Penyimpanan	Pimpinan Unit IT, Petugas Unit IT	Sistem harus menyediakan fungsi untuk meng*update* barang pada daftar barang di gudang penyimpanan.	SPG_F_27
			Field yang harus tersedia adalah nama barang dan kuantitas barang serta tombol “*Update*” dan tombol “Tidak”.	SPG_F_27_01
			Setiap *field* memiliki nilai *default* sama dengan nilai sebelum data barang diedit.	SPG_F_27_02
			Nilai minimal *field* kuantitas adalah 0.	SPG_F_27_03
28.	Menghapus Barang dari Daftar Barang Gudang Penyimpanan	Pimpinan Unit IT, Petugas Unit IT	Sistem harus menyediakan fungsi untuk menghapus barang pada daftar barang di gudang penyimpanan.	SPG_F_28
			Sistem harus menampilkan pesan konfirmasi sebelum proses penghapusan dilakukan.	SPG_F_28_01
29.	Melihat Daftar Riwayat Laporan Gangguan	Pimpinan Unit IT, Petugas Unit IT	Sistem harus menyediakan fungsi untuk menampilkan daftar riwayat laporan gangguan.	SPG_F_29

No	*Use Case*	Aktor	Spesifikasi	Kode
			Pada halaman daftar riwayat laporan gangguan data yang ditampilkan pada tabel adalah nomor, NIPP pelapor, NIPP petugas, jenis gangguan, lokasi, status dan tombol *detail.*	SPG_F_29_01
			Pada halaman daftar riwayat laporan gangguan juga terdapat *form filter* yang terdiri dari *field* tanggal awal, *field* tanggal akhir dan satu tombol "*filter*".	SPG_F_29_02
			Pada halaman ini terdapat satu *dropdown field* yang digunakan untuk memilih berapa jumlah data yang ditampilkan.	SPG_F_29_03
			Pada halaman ini juga terdapat *field* yang digunakan untuk mencari data berdasarkan kode gangguan, NIPP pelapor, NIPP petugas, lokasi gangguan, jenis gangguan, atau status.	SPG_F_29_04
30.	Mengekspor Daftar Riwayat Laporan Gangguan	Pimpinan Unit IT, Petugas Unit IT	Sistem harus menyediakan fungsi untuk mengekspor daftar riwayat laporan gangguan dalam bentuk *file* berekstensi .xlsx	SPG_F_30
31.	Melihat *Detail* Riwayat Laporan Gangguan	Pimpinan Unit IT, Petugas Unit IT	Sistem harus menyediakan fungsi untuk melihat *detail* dari sebuah riwayat laporan gangguan.	SPG_F_31
			Pada halaman *detail* laporan gangguan terdapat bagian *detail*	SPG_F_31_01

No	*Use Case*	Aktor	Spesifikasi	Kode
			laporan gangguan yang menampilkan NIPP pelapor, NIPP petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, dan solusi.	
			Pada halaman *detail* laporan gangguan terdapat bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan.	SPG_F_31_02
			Pada halaman *detail* laporan gangguan terdapat bagian pendataan barang *lintas* yang memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan yang berupa tabel dengan kolom nama barang dan kode barang.	SPG_F_31_03
32.	Melihat Profil	Pimpinan Unit IT, Petugas Unit IT, Pelapor	Sistem harus menyediakan fungsi untuk melihat profil.	SPG_F_32
			Data yang ditampilkan adalah nama lengkap, NIPP, unit kerja, jabatan, kedudukan, *email* dan nomor telepon.	SPG_F_32_01
			Pada halaman profil terdapat tombol edit	SPG_F_32_02

No	*Use Case*	Aktor	Spesifikasi	Kode
			profil, edit foto profil dan edit *password.*	
33.	Mengedit Profil	Pimpinan Unit IT, Petugas Unit IT, Pelapor	Sistem harus dapat mengedit profil pelapor, pimpinan unit IT dan pegawai unit IT.	SPG_F_33
			Profil pengguna yang dapat dirubah adalah nama lengkap, unit kerja, jabatan, kedudukan, dan nomor telepon.	SPG_F_33_01
34.	Mengedit *Password*	Pimpinan Unit IT, Petugas Unit IT, Pelapor	Sistem harus menyediakan fungsi untuk melakukan edit *password.*	SPG_F_34
			Pada *form* edit *password field* harus ada *field password* lama, *password* baru dan ulangi *password* baru.	SPG_F_34_01
35.	Mengedit Foto Profil	Pimpinan Unit IT, Petugas Unit IT, Pelapor	Sistem harus menyediakan fungsi untuk melakukan edit foto profil.	SPG_F_35
			Pada *form* edit foto profil ada satu inputan yang bertipe *file,* tombol batal dan tombol simpan.	SPG_F_35_01
			File yang dimasukkan harus berekstensi *image.*	SPG_F_35_02
36.	Melihat Grafik Riwayat Laporan Gangguan Bulanan	Pimpinan Unit IT, Petugas Unit IT	Sistem harus menyediakan fungsi untuk menampilkan grafik riwayat laporan gangguan berdasarkan bulan.	SPG_F_36
			Pada halaman grafik riwayat laporan gangguan	SPG_F_36_01

No	*Use Case*	Aktor	Spesifikasi	Kode
			bulanan terdapat *form filter* yang terdiri dari satu *input*-an untuk bulan dan tahun dan satu tombol Pilih.	
			Pada halaman grafik riwayat laporan gangguan bulanan terdapat grafik berbentuk *bar* yang menampilkan jumlah gangguan yang terjadi pada setiap stasiun pada bulan yang dipilih.	SPG_F_36_02
			Pada halaman grafik riwayat laporan gangguan bulanan terdapat grafik berbentuk *pie* yang menampilkan jumlah setiap jenis gangguan pada bulan yang dipilih.	SPG_F_36_03
37.	*Logout*	Pimpinan Unit IT, Petugas Unit IT, Pelapor	Sistem harus dapat mengeluarkan aktor dari sistem.	SPG_F_37
38.	Mencari Akun	Pimpinan Unit IT	Sistem harus dapat menyediakan fungsi untuk mencari akun berdasarkan NIPP, nama lengkap, unit kerja, jabatan, nomor telepon, tipe *user*, atau *email*.	SPG_F_38
39.	Mengekspor Daftar Akun	Pimpinan Unit IT	Sistem harus dapat menyediakan fungsi untuk mengekspor data akun.	SPG_F_39
			Daftar akun akan diekspor dalam bentuk *file excel.*	SPG_F_39_01
40.	Mencari Permohonan *Lintas*	Pimpinan Unit IT	Sistem harus dapat menyediakan fungsi untuk mencari permohonan lintas berdasarkan kode	SPG_F_40

No	*Use Case*	Aktor	Spesifikasi	Kode
			gangguan, NIPP petugas, lokasi gangguan, atau jenis gangguan.	
41.	Mencari Laporan Gangguan yang Telah Terlaporkan	Pelapor	Sistem harus dapat menyediakan fungsi untuk mencari laporan gangguan yang telah terlaporkan berdasarkan kode gangguan, NIPP pelapor, NIPP petugas, jenis gangguan, lokasi gangguan, waktu lapor, atau status.	SPG_F_41
42.	Mencari Laporan Gangguan yang Ditangani	Petugas Unit IT	Sistem harus dapat menyediakan fungsi untuk mencari laporan gangguan yang ditangani berdasarkan kode gangguan, NIPP pelapor, NIPP petugas, jenis gangguan, lokasi gangguan, waktu lapor, atau status.	SPG_F_42
43.	Mencari *Lintas*	Petugas Unit IT	Sistem harus dapat menyediakan fungsi untuk mencari *lintas* berdasarkan kode gangguan, NIPP pelapor, lokasi gangguan, jenis gangguan, status *lintas* atau catatan.	SPG_F_43
44.	Mencari Barang	Petugas Unit IT, Pimpinan Unit IT	Sistem harus dapat meyediakan fungsi untuk mencari barang berdasarkan nama barang atau kuantitas.	SPG_F_44
45.	Mencari Riwayat Laporan Gangguan	Petugas Unit IT, Pimpinan Unit IT	Sistem harus dapat menyediakan fungsi untuk mencari riwayat laporan gangguan berdasarkan kode gangguan, NIPP pelapor, NIPP petugas,	SPG_F_45

No	*Use Case*	Aktor	Spesifikasi	Kode
			lokasi gangguan, jenis gangguan, atau status.	
46.	Melihat Grafik Riwayat Laporan Gangguan Tahunan	Petugas Unit IT, Pimpinan Unit IT	Sistem harus dapat menyediakan fungsi untuk menampilkan grafik *bar* yang menunjukkan banyaknya gangguan setiap tahunnya.	SPG_F_46
47.	*Filter* Daftar Riwayat Laporan Gangguan Berdasarkan Tanggal	Petugas Unit IT, Pimpinan Unit IT	Sistem harus dapat menyediakan fungsi untuk melakukan *filter* terhadap daftar riwayat laporan gangguan berdasarkan tanggal.	SPG_F_47
			Terdapat dua *input*an yaitu tanggal awal dan tanggal akhir dan satu tombol *filter.*	SPG_F_47_01
			Tanggal pada tanggal akhir tidak boleh kurang dari tanggal awal.	SPG_F_47_02
48.	Menampilkan Grafik Riwayat Laporan Bulanan Berdasarkan Bulan	Petugas Unit IT, Pimpinan Unit IT	Sistem harus dapat menyediakan fungsi untuk menampilkan grafik riwayat laporan bulanan berdasarkan bulan.	SPG_F_48
			Terdapat satu buat *input*an yatu *input*an bulan tahun.	SPG_F_48_01

4.2.3 Daftar Kebutuhan Non-Fungsional

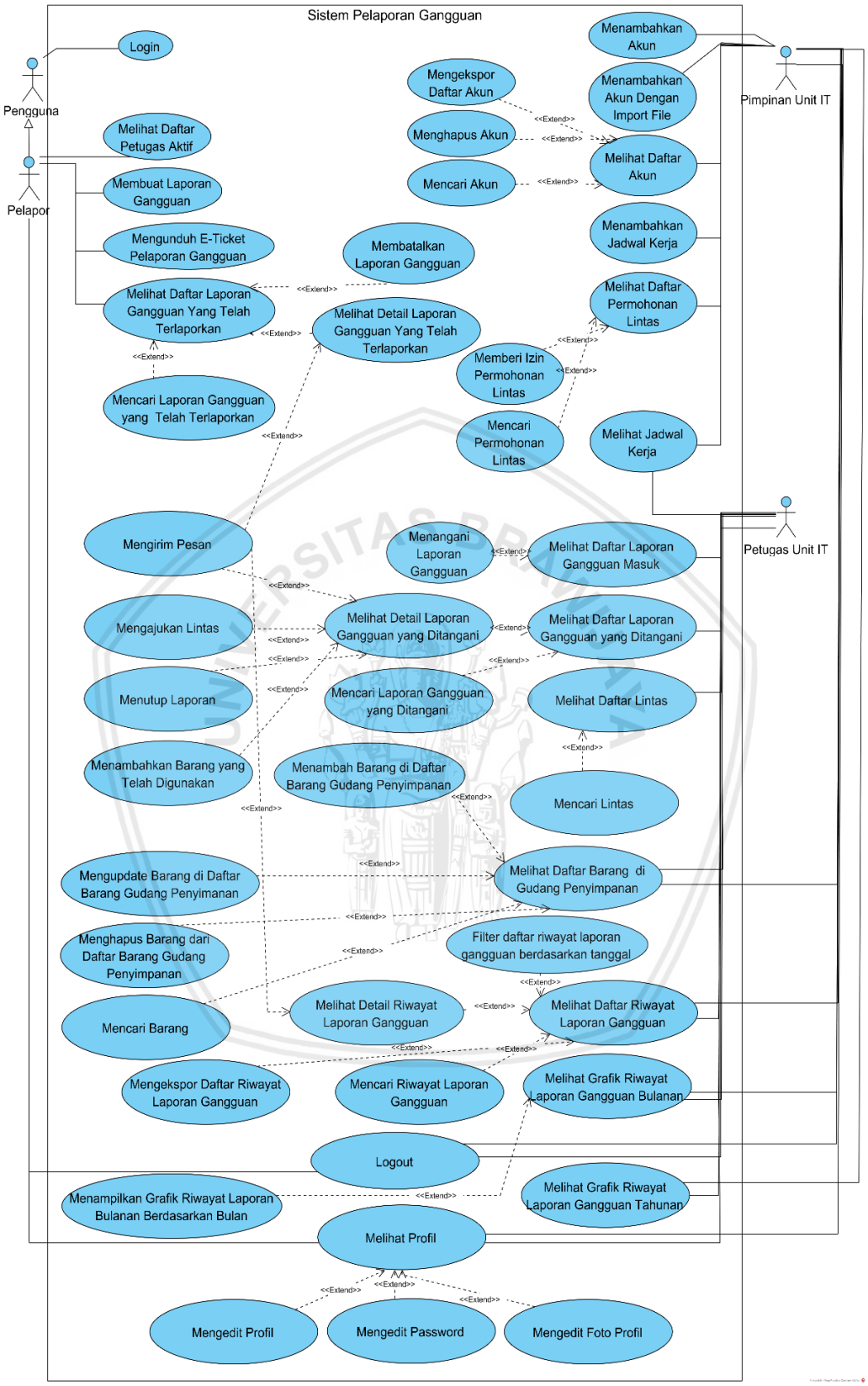
Kebutuhan non-fungsional merupakan kebutuhan yang dapat mendukung berjalannya fungsi yang terdapat didalam sistem. Berikut daftar kebutuhan non-fungsional yang harus dimiliki oleh sistem yang dapat dilihat pada Tabel 4.3.

Tabel 4.3 Daftar kebutuhan non-fungsional

No	Kode Fungsi	Parameter	Deskripsi
1.	SPG_NF_01	*Compatibility*	Sistem kompatibel pada *browser edge, firefox, safari, opera* dan *chrome.*
	SPG_NF_01_1		Parameter yang digunakan dalam pengujian ini adalah menggunakan tag HTML dan atribut yang kompatibel pada browser, menggunakan aturan CSS yang didukung oleh browser, tidak ada *element* yang hilang, dan tidak ada properti *JavaScript* dan *DOM* dan fungsi yang memicu bug pada browser.

4.2.4 *Use Case Diagram*

Use case diagram merupakan diagram yang digunakan untuk merepresentasikan perilaku sistem yang akan dibangun dan aktor yang terlibat. *Use case diagram* dibuat dengan melihat dari kebuthan fungsional yang sebelumya telah terdefinisi. Dalam *use case diagram* sistem pelaporan gangguan ini terdapat 4 aktor yaitu pengguna, pimpinan unit IT, petugas unit IT dan pelapor. *Use case* yang terdapat dalam *use case diagram* sistem pelaporan gangguan ini adalah sebanyak 48 *use case*. Berikut *use case diagram* dari sistem pelaporan gangguan dapat dilihat pada Gambar 4.5.



Gambar 4.5 *Use-case diagram*

4.2.5 *Use Case Scenario*

Setelah membuat *use case diagram*, waktunya untuk menjabarkan alur kerja atau langkah-langkah kerja tiap *use case* yang telah didefinisikan sebelumnya dengan menggunakan *use case scenario*. Penjelasan langkah-langkah setiap *use case* dapat dilihat pada setiap tabel *use case scenario*nya pada Tabel 4.4 hingga Tabel 4.51.

4.2.5.1 *Login* – SPG_F_01

Use case scenario login dapat dilihat pada Tabel 4.4. Aktor yang terlibat dalam *use case* ini adalah pengguna. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case login*. Terdapat 3 *alternative flow* yaitu apabila *field* NIPP tidak diisi, apabila *field password* tidak diisi dan apabila NIPP atau *password* yang dimasukkan salah.

Tabel 4.4 *Use case scenario login*

Login	
Actor	Pengguna
Objective	Aktor telah dikenali sistem sebagai aktor yang valid dan menampilkan halaman utama.
Pre-Condition	Aktor telah berada pada halaman *login.*
Main Flow	1. Sistem menampilkan *form login* yang terdiri NIPP dan *password.* 2. Aktor mengisi *form login* secara lengkap. 3. Aktor menekan tombol *login.* 4. Sistem melakukan validasi masukan dari aktor. 5. Sistem menampilkan halaman utama.
Alternative Flow	2. Aktor tidak mengisi *form* secara lengkap: a. Jika *field* NIPP tidak terisi maka sistem akan menampilkan pesan “NIPP wajib diisi terdiri dari angka”. b. Jika *field password* tidak terisi maka sistem akan menampilkan pesan *“Password* wajib diisi”*.*

	4. Jika validasi bernilai tidak valid maka akan menampilkan pesan "NIPP atau *password* yang anda masukkan salah, silahkan coba lagi" dan akan menampilkan kembali halaman *login.*
Post Condition	Sistem menampilkan halaman utama.

4.2.5.2 *Logout* – SPG_F_37

Use case scenario logout dapat dilihat pada Tabel 4.5. Aktor yang terlibat dalam *use case* ini adalah Pelapor, Pimpinan Unit IT, dan Petugas Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case logout*.

Tabel 4.5 *Use case scenario logout*

Logout	
Actor	Pelapor, Pimpinan Unit IT, Petugas Unit IT.
Objective	Aktor keluar dari sistem dan dikenali sebagai aktor yang tidak valid.
Pre-Condition	Aktor telah melakukan *login* atau masuk ke dalam sistem.
Main Flow	1. Aktor menekan foto profil di pojok kanan atas. 2. Sistem akan menampilkan menu *dropdown.* 3. Aktor menekan tombol *logout.*
Alternative Flow	-
Post Condition	Sistem menampilkan halaman *login*.

4.2.5.3 Melihat Daftar Akun – SPG_F_04

Use case scenario Melihat Daftar Akun dapat dilihat pada Tabel 4.6. Aktor yang terlibat dalam *use case* ini adalah Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat Daftar Akun.

Tabel 4.6 *Use case scenario* melihat daftar akun

Melihat Daftar Akun	
Actor	Pimpinan Unit IT.
Objective	Untuk melihat data akun yang telah terdaftar.

Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor memilih menu Administrasi. 2. Sistem menampilkan menu *dropdown.* 3. Aktor memilih menu Daftar Akun. 4. Sistem akan menampilkan halaman daftar akun yang akan menampilkan tabel yang didalamnya terdapat data nomor, NIPP, nama lengkap, jabatan, unit kerja, kedudukan, nomor telepon, tipe *user*, dan *email* dari setiap akun serta tombol untuk menghapus akun tersebut.
Alternative Flow	-
Post Condition	Sistem akan menampilkan halaman daftar akun yang akan menampilkan tabel yang didalamnya terdapat data nomor, NIPP, nama lengkap, jabatan, unit kerja, kedudukan, nomor telepon, tipe *user*, dan *email* dari setiap akun serta tombol untuk menghapus akun tersebut.

4.2.5.4 Menambahkan Akun – SPG_F_02

Use case scenario Menambahkan Akun dapat dilihat pada Tabel 4.7. Aktor yang terlibat dalam *use case* ini adalah Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Menambahkan Akun. Terdapat 2 *alternative flow* yaitu apabila *field input*an tidak diisi dengan lengkap dan saat NIPP sudah digunakan.

Tabel 4.7 *Use case scenario* menambahkan akun

Menambahkan Akun	
Actor	Pimpinan Unit IT.
Objective	Untuk menambahkan akun pada sistem.
Pre-Condition	Aktor telah masuk ke dalam sistem dan telah berada pada halaman tambah akun.

Main Flow	1. Sistem menampilkan *form* tambah akun yang terdiri dari NIPP, nama lengkap, unit kerja, jabatan, kedudukan, nomor telepon, *email*, tipe akun. 2. Aktor mengisi *input* NIPP. 3. Sistem melakukan pengecekan *input*-an NIPP dari aktor. 4. Sistem menampilkan pesan "NIPP tersedia". 5. Aktor mengisi *form* tambah akun secara lengkap dan tanpa pesan kesalahan. 6. Aktor menekan tombol tambahkan akun. 7. Sistem menyimpan data akun dan menampilkan pesan "Berhasil membuat akun baru".
Alternative Flow	3. Jika NIPP sudah terdaftar maka akan menampilkan pesan "NIPP sudah terdaftar". 4. Jika aktor mengisi *form* tambah akun tidak lengkap dan NIPP sudah terdaftar dalam sistem akan *disable* tombol "Tambahkan Akun".
Post Condition	Sistem menyimpan data akun dan menampilkan pesan "Berhasil membuat akun baru".

4.2.5.5 Menambahkan Akun Dengan Impor *File* – SPG_F_03

Use case scenario Menambahkan Akun Dengan Impor *File* dapat dilihat pada Tabel 4.8. Aktor yang terlibat dalam *use case* ini adalah Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case scenario* Menambahkan Akun Dengan Impor *File*. Terdapat 4 *alternative flow* yaitu apabila *field input*an tidak diisi dengan lengkap, jika *field input*an diisi tidak dengan *file excel*, jika di dalam *file* tidak terdapat data dan saat ada NIPP yang sudah terdaftar di dalam sistem.

Tabel 4.8 *Use case scenario* menambahkan akun dengan impor *file*

Menambahkan Akun Dengan Impor *File*

Actor	Pimpinan Unit IT.
Objective	Untuk menambahkan akun pada sistem.
Pre-Condition	Aktor telah masuk ke dalam sistem dan telah berada pada halaman tambah akun dengan impor.
Main Flow	1. Sistem menampilkan *form* tambah akun dengan impor *file* yang terdiri dari *input*an *file.* 2. Aktor mengunduh format yang disediakan. 3. Aktor mengisi data setiap baris dengan lengkap. 4. Aktor mengisi *form* tambah akun dengan impor *file* dengan lengkap dan sesuai format. 5. Aktor menekan tombol tambahkan akun. 6. Sistem melakukan validasi. 7. Sistem menyimpan data akun dan menampilkan pesan "Akun berhasil diimpor".
Alternative flow	4. Jika *form* tambah akun dengan impor *file* tidak diisi dengan lengkap tapi sudah menekan tombol tambahkan akun maka akan menampilkan pesan "*Please select a file*". 4. Jika *form* tambah akun dengan impor *file* diisi dengan bukan *file* berekstensi xls atau xlsx maka akan menampilkan pesan "File yang dimasukkan harus memiliki ekstensi .xlsx,.xls" 6. Jika terdapat NIPP yang terdaftar maka akan menampilkan pesan "Impor akun gagal dilakukan, NIPP (NIPP yang terdaftar) sudah terdaftar dalam sistem".

	6. Jika tidak terdapat data di dalam *file excel* tersebut maka akan menampilkan pesan "Tidak ada data di dalam *file excel*".
Post Condition	Sistem menyimpan data akun dan menampilkan pesan "Akun berhasil diimpor".

4.2.5.6 Menghapus Akun – SPG_F_05

Use case scenario Menghapus Akun dapat dilihat pada Tabel 4.9. Aktor yang terlibat dalam *use case* ini adalah Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Menghapus Akun. Ada satu *alternative flow* yaitu ketika tombol tidak ditekan.

Tabel 4.9 *Use case scenario* menghapus akun

Menghapus Akun	
Actor	Pimpinan Unit IT.
Objective	Untuk menghapus akun pada sistem.
Pre-Condition	Aktor telah masuk ke dalam sistem dan telah berada pada halaman daftar akun.
Main Flow	1. Aktor menekan tombol hapus pada baris akun yang ingin dihapus. 2. Sistem menampilkan modal konfirmasi. 3. Aktor menekan tombol Ya. 4. Sistem menghapus data akun pada *database* dan menampilkan halaman daftar akun.
Alternative Flow	4. Jika aktor menekan tombol Tidak maka akan menutup modal konfirmasi.
Post Condition	Sistem menghapus data akun pada *database* dan menampilkan halaman daftar akun.

4.2.5.7 Melihat Jadwal Kerja – SPG_F_06

Use case scenario Melihat Jadwal Kerja dilihat pada Tabel 4.10. Aktor yang terlibat dalam *use case* ini adalah Pimpinan Unit IT dan Petugas Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat Jadwal Kerja.

Terdapat satu *alternative flow* yaitu pada saat jadwal kerja pada bulan tersebut belum dibuat oleh Pimpinan IT.

Tabel 4.10 *Use case scenario* melihat jadwal kerja

Melihat Jadwal Kerja	
Actor	Pimpinan Unit IT, Petugas Unit IT.
Objective	Untuk menampilkan jadwal kerja bulanan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor memilih menu Jadwal Kerja. 2. Sistem menampilkan jadwal kerja yaitu NIPP, nama lengkap, semua tanggal pada bulan tersebut, dan jadwal masing masing pegawai di setiap tanggalnya.
Alternative Flow	2. Jika jadwal bulan tersebut belum dibuat maka sistem menampilkan "*No data available in table*" pada tabel.
Post Condition	Sistem menampilkan jadwal kerja yaitu NIPP, nama lengkap, semua tanggal pada bulan tersebut, dan jadwal masing masing pegawai di setiap tanggalnya.

4.2.5.8 Menambahkan Jadwal Kerja – SPG_F_07

Use case scenario Menambahkan Jadwal Kerja dilihat pada Tabel 4.11. Aktor yang terlibat dalam *use case* ini adalah Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Menambahkan Jadwal Kerja.

Tabel 4.11 *Use case scenario* menambahkan jadwal kerja

Menambahkan Jadwal Kerja	
Actor	Pimpinan Unit IT.
Objective	Untuk menambahkan jadwal kerja.
Pre-Condition	Aktor telah masuk ke dalam sistem dan berada pada halaman jadwal kerja.
Main Flow	1. Aktor menekan tombol tambah jadwal.

	2. Sistem akan mengambil daftar pegawai. 3. Sistem akan mengambil daftar pimpinan. 4. Sistem akan mengambil daftar petugas. 5. Sistem akan menghapus jadwal kerja bulan yang telah terdaftar. 6. Sistem akan menginisialisasi hari kerja dan libur setiap pegawai. 7. Sistem akan menginisialisasi jadwal dinas posko untuk pemimpin. 8. Sistem akan menginisialiasi jadwal *ready on call* atau piket malam petugas. 9. Sistem menyimpan jadwal kerja, menampilkan pesan "Berhasil membuat jadwal kerja" dan menampilkan halaman jadwal kerja.
Alternative Flow	-
Post Condition	Sistem menyimpan jadwal kerja, menampilkan pesan "Berhasil membuat jadwal kerja" dan menampilkan halaman jadwal kerja.

4.2.5.9 Melihat Daftar Permohonan *Lintas* – SPG_F_08

Use case scenario Melihat Daftar Permohonan *Lintas* dilihat pada Tabel 4.12. Aktor yang terlibat dalam *use case* ini adalah Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat Daftar Permohonan *Lintas*. Terdapat satu *alternative flow* yaitu ketika tidak ada data yang dapat ditampilkan dari *database*.

Tabel 4.12 *Use case scenario* melihat daftar permohonan *lintas*

Melihat Daftar Permohonan *Lintas*	
Actor	Pimpinan Unit IT.
Objective	Untuk melihat daftar permohonan *lintas*.

Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor menekan menu Permohonan *Lintas.* 2. Sistem akan mengambil data dari *database.* 3. Sistem akan menampilkan halaman daftar permohonan *lintas* yang terdapat tabel daftar permohonan *lintas* yang memiliki kolom kode gangguan, NIPP petugas, lokasi gangguan, jenis gangguan dan tombol izinkan.
Alternative Flow	3. Jika tidak ada data yang dapat ditampilkan dari *database* maka akan menampilkan pesan "*No data available in table"* pada tabel.
Post Condition	Sistem akan menampilkan halaman daftar permohonan *lintas* yang terdapat tabel daftar permohonan *lintas* yang memiliki kolom kode gangguan, NIPP petugas, lokasi gangguan, jenis gangguan dan tombol izinkan.

4.2.5.10 Memberi Izin Permohonan *Lintas* – SPG_F_09

Use case scenario Memberi Izin Permohonan *Lintas* dilihat pada Tabel 4.13. Aktor yang terlibat dalam *use case* ini adalah Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Memberi Izin Permohonan *Lintas*. Terdapat satu *alternative flow* yaitu pada saat tombol batal ditekan.

Tabel 4.13 *Use case scenario* memberi izin permohonan *lintas*

Memberi Izin Permohonan *Lintas*	
Actor	Pimpinan Unit IT.
Objective	Untuk memberikan izin terhadap permohonan *lintas.*
Pre-Condition	Aktor telah masuk ke dalam sistem dan berada pada halaman daftar permohonan *lintas.*
Main Flow	1. Aktor memilih laporan *lintas* yang akan diizinkan.

	2. Aktor menekan tombol “Izinkan”. 3. Sistem akan menampilkan modal izinkan *lintas.* 4. Aktor mengisi catatan. 5. Aktor menekan tombol Izinkan. 6. Sistem meng*update* status *lintas* dan status gangguan dan menampilkan halaman daftar permohonan *lintas.*
Alternative Flow	5. Jika aktor menekan tombol batal maka akan menutup modal izinkan *lintas.*
Post Condition	Sistem meng*update* status *lintas* dan status gangguan dan menampilkan halaman daftar permohonan *lintas.*

4.2.5.11 Melihat Daftar Petugas Aktif – SPG_F_10

Use case scenario Melihat Daftar Petugas Aktif dilihat pada Tabel 4.14. Aktor yang terlibat dalam *use case* ini adalah Pelapor. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat Daftar Petugas Aktif.

Tabel 4.14 *Use case scenario* melihat daftar petugas aktif

Melihat Daftar Petugas Aktif	
Actor	Pelapor.
Objective	Untuk melihat daftar petugas yang sedang aktif atau sedang bekerja.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor memilih menu pelaporan. 2. Sistem akan menampilkan menu *dropdown.* 3. Aktor memilih menu daftar petugas aktif. 4. Sistem akan menampilkan halaman lihat petugas aktif yang berisi data pegawai yang sedang aktif dengan data nama lengkap, NIPP, *email* dan nomor telepon.
Alternative Flow	-

Post Condition	Sistem akan menampilkan halaman lihat petugas aktif yang berisi data pegawai yang sedang aktif dengan data nama lengkap, NIPP, *email* dan nomor telepon.

4.2.5.12 Membuat Laporan Gangguan – SPG_F_11

Use case scenario Membuat Laporan Gangguan dilihat pada Tabel 4.15. Aktor yang terlibat dalam *use case* ini adalah Pelapor. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Membuat Laporan Gangguan. Terdapat satu *alternative flow* yaitu pada saat *input* tidak diisi lengkap.

Tabel 4.15 *Use case scenario* membuat laporan gangguan

Membuat Laporan Gangguan	
Actor	Pelapor.
Objective	Untuk membuat laporan gangguan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor memilih menu pelaporan. 2. Sistem akan menampilkan menu *dropdown.* 3. Aktor memilih menu lapor gangguan. 4. Sistem akan menampilkan halaman buat laporan gangguan dengan *form* buat laporan gangguan dengan *input* tipe gangguan, deskripsi gangguan dan lokasi gangguan. *5.* Aktor mengisi *form* buat laporan gangguan dengan lengkap. 6. Aktor menekan tombol laporkan gangguan. 7. Sistem men*generate* kode gangguan dan mengecek kode gangguan. 8. Sistem mengambil data petugas aktif. 9. Sistem menyimpan laporan gangguan pada *database,*

	menampilkan *detail* laporan gangguan dan memberikan notifikasi kepada petugas aktif dan menampilkan halaman laporan gangguan berhasil dilaporkan.
Alternative Flow	5. Jika *input* tidak diisi dengan lengkap maka akan menampilkan pesan "*Please fill out this field".*
Post Condition	Sistem menyimpan laporan gangguan pada *database,* menampilkan *detail* laporan gangguan dan memberikan notifikasi kepada petugas aktif dan menampilkan halaman laporan gangguan berhasil dilaporkan.

4.2.5.13 Mengunduh *E-Ticket* Pelaporan Gangguan – SPG_F_12

Use case scenario Mengunduh *E-Ticket* Pelaporan Gangguan dilihat pada Tabel 4.16. Aktor yang terlibat dalam *use case* ini adalah Pelapor. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mengunduh *E-Ticket* Pelaporan Gangguan.

Tabel 4.16 *Use case scenario* mengunduh *e-ticket* pelaporan gangguan

Mengunduh *E-Ticket* Pelaporan Gangguan	
Actor	Pelapor.
Objective	Untuk mengunduh *e-ticket* pelaporan gangguan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor memilih menu pelaporan. 2. Sistem akan menampilkan menu *dropdown.* 3. Aktor memilih menu daftar laporan gangguan. 4. Sistem akan menampilkan halaman daftar laporan gangguan terlaporkan. 5. Aktor memilih laporan gangguan yang ingin diunduh *e-ticket*nya dengan menekan tombol "Unduh *E-Ticket*".

	6. *E-Ticket* akan terunduh ke perangkat aktor.
Alternative Flow	-
Post Condition	*E-Ticket* akan terunduh ke perangkat actor.

4.2.5.14 Melihat Daftar Laporan Gangguan Yang Telah Terlaporkan – SPG_F_13

Use case scenario Melihat Daftar Laporan Gangguan Yang Telah Terlaporkan dilihat pada Tabel 4.17. Aktor yang terlibat dalam *use case* ini adalah Pelapor. *Use case scenario* ini menggambarkan alur yang terjadi pada *use* Melihat Daftar Laporan Gangguan Yang Telah Terlaporkan. Terdapat satu *alternative flow* yaitu ketika tidak ada data yang dapat ditampilkan.

Tabel 4.17 *Use case scenario* melihat daftar laporan gangguan yang telah terlaporkan

Melihat Daftar Laporan Gangguan Yang Telah Terlaporkan	
Actor	Pelapor.
Objective	Untuk melihat daftar laporan gangguan yang telah terlaporkan oleh aktor.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor memilih menu pelaporan. 2. Sistem akan menampilkan menu *dropdown.* 3. Aktor memilih menu daftar laporan gangguan. 4. Sistem mengambil data laporan gangguan yang terlaporkan dari *database.* 5. Sistem akan menampilkan halaman daftar laporan gangguan terlaporkan yang terdapat tabel yang berisi nomor, kode gangguan, NIPP petugas, jenis gangguan, lokasi gangguan, waktu lapor, status, tombol unduh *e-ticket*, tombol lihat *detail*, dan tombol batalkan dari setiap laporan gangguan.

Alternative Flow	5. Jika tidak ada data yang dapat ditampilkan sistem akan menampilkan halaman lihat daftar laporan gangguan yang terlaporkan dengan tabel yang berisi "*No data available in table*".
Post Condition	Sistem akan menampilkan halaman daftar laporan gangguan terlaporkan yang terdapat tabel yang berisi nomor, kode gangguan, NIPP petugas, jenis gangguan, lokasi gangguan, waktu lapor, status, tombol unduh *e-ticket*, tombol lihat *detail*, dan tombol batalkan dari setiap laporan gangguan.

4.2.5.15 Melihat *Detail* Laporan Gangguan yang Telah Terlaporkan – SPG_F_14

Use case scenario Melihat *Detail* Laporan Gangguan yang Telah Terlaporkan dilihat pada Tabel 4.18. Aktor yang terlibat dalam *use case* ini adalah Pelapor. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat *Detail* Laporan Gangguan yang Telah Terlaporkan.

Tabel 4.18 *Use case scenario* melihat *detail* laporan gangguan yang telah terlaporkan

Melihat *Detail* Laporan Gangguan yang Telah Terlaporkan	
Actor	Pelapor.
Objective	Untuk melihat *detail* laporan gangguan yang telah terlaporkan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman daftar laporan gangguan terlaporkan.
Main Flow	1. Aktor memilih laporan gangguan yang ingin dilihat *detail*nya. 2. Sistem akan menampilkan halaman *detail* laporan gangguan yang terdapat bagian *detail* laporan gangguan yang menampilkan NIPP pelapor, NIPP Petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, solusi, bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang

	menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan dan bagian pendataan barang *lintas* yang memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan.
Alternative Flow	-
Post Condition	Sistem akan menampilkan halaman *detail* laporan gangguan yang terdapat bagian *detail* laporan gangguan yang menampilkan NIPP pelapor, NIPP Petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, solusi, bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan dan bagian pendataan barang *lintas* yang memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan.

4.2.5.16 Membatalkan Laporan Gangguan – SPG_F_15

Use case scenario Membatalkan Laporan Gangguan dilihat pada Tabel 4.19. Aktor yang terlibat dalam *use case* ini adalah Pelapor. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Membatalkan Laporan Gangguan. Terdapat satu *alternative flow* yaitu ketika tombol tidak ditekan.

Tabel 4.19 *Use case scenario* membatalkan laporan gangguan

Membatalkan Laporan Gangguan.	
Actor	Pelapor.
Objective	Untuk membatalkan laporan gangguan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman daftar laporan gangguan terlaporkan.

Main Flow	1. Aktor memilih laporan gangguan yang ingin dibatalkan. 2. Aktor menekan tombol batalkan. 3. Sistem akan menampilkan modal batalkan laporan gangguan. 4. Aktor menekan tombol Ya. 5. Sistem akan meng*update* status laporan gangguan menjadi dibatalkan dan akan menampilkan halaman daftar laporan gangguan terlaporkan.
Alternative Flow	4. Jika aktor menekan tombol tidak maka sistem akan menutup modal batalkan laporan gangguan.
Post Condition	Sistem akan meng*update* status laporan gangguan menjadi dibatalkan dan akan menampilkan halaman lihat daftar laporan gangguan terlaporkan.

4.2.5.17 Mengirim Pesan – SPG_F_16

Use case scenario Mengirim Pesan dilihat pada Tabel 4.20. Aktor yang terlibat dalam *use case* ini adalah Pelapor dan Petugas Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mengirim Pesan.

Tabel 4.20 *Use case scenario* mengirim pesan

Mengirim Pesan	
Actor	Pelapor dan Petugas Unit IT.
Objective	Untuk mengirim pesan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan berada pada halaman *detail* laporan gangguan.
Main Flow	1. Aktor memasukan isi pesan. 2. Aktor menekan tombol kirim. 3. Sistem akan menyimpan pesan dalam *database*, menampilkannya dalam bagian pesan dan memberikan notifikasi kepada penerima.
Alternative Flow	-

Post Condition	Sistem akan menyimpan pesan dalam *database*, menampilkannya dalam bagian pesan dan memberikan notifikasi kepada penerima.

4.2.5.18 Melihat Daftar Laporan Gangguan Masuk – SPG_F_17

Use case scenario Melihat Daftar Laporan Gangguan Masuk dilihat pada Tabel 4.21. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat Daftar Laporan Gangguan Masuk. Terdapat satu *alternative flow* yaitu saat tidak ada laporan gangguan masuk atau tidak ada data yang dapat ditampilkan.

Tabel 4.21 *Use case scenario* melihat daftar laporan gangguan masuk

Melihat Daftar Laporan Gangguan Masuk	
Actor	Petugas Unit IT.
Objective	Untuk melihat daftar laporan gangguan masuk.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor memilih menu penanganan. 2. Sistem akan menampilkan menu *dropdown.* 3. Aktor memilih submenu Laporan Gangguan Masuk. 4. Sistem mengambil data laporan gangguan yang masuk dari *database.* 5. Sistem akan menampilkan halaman daftar laporan gangguan masuk yang menampilkan tabel dengan kolom nomor, kode gangguan, NIPP pelapor, jenis gangguan, deskripsi, lokasi gangguan, waktu lapor, dan tombol “tangani” dari setiap laporan gangguan.
Alternative Flow	5. Jika tidak ada laporan gangguan yang masuk atau tidak ada data yang dapat ditampilkan maka sistem akan menampilkan “*No data available in tabel*” pada tabel.

Post Condition	Sistem akan menampilkan halaman daftar laporan gangguan masuk yang menampilkan nomor, kode gangguan, NIPP pelapor, jenis gangguan, deskripsi, lokasi gangguan, waktu lapor, dan tombol "tangani" dari setiap laporan gangguan.

4.2.5.19 Menangani Laporan Gangguan – SPG_F_18

Use case scenario Menangani Laporan Gangguan dilihat pada Tabel 4.22. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Menangani Laporan Gangguan. Terdapat satu *alternative flow* yaitu saat tombol tidak ditekan.

Tabel 4.22 *Use case scenario* menangani laporan gangguan

Menangani Laporan Gangguan.	
Actor	Petugas Unit IT.
Objective	Untuk menangani laporan gangguan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan telah berada pada halaman daftar laporan gangguan masuk.
Main Flow	1. Aktor memilih laporan gangguan yang ingin ditangani. 2. Aktor menekan tombol tangani. 3. Sistem akan menampilkan modal tangani laporan gangguan. 4. Aktor menekan tombol Ya. 5. Sistem akan meng*update* status laporan gangguan menjadi sedang ditangani, memberikan notifikasi kepada pelapor dan akan menampilkan halaman daftar laporan gangguan ditangani.
Alternative Flow : tombol tidak ditekan	4. Jika aktor menekan tombol tidak maka sistem akan menutup modal tangani laporan gangguan.
Post Condition	Sistem akan meng*update* status laporan gangguan menjadi sedang ditangani, memberikan notifikasi kepada pelapor dan akan

	menampilkan halaman daftar laporan gangguan ditangani.

4.2.5.20 Melihat Daftar Laporan Gangguan yang Ditangani – SPG_F_19

Use case scenario Melihat Daftar Laporan Gangguan yang Ditangani dilihat pada Tabel 4.23. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat Daftar Laporan Gangguan yang Ditangani. Terdapat satu *alternative flow* yaitu saat tidak ada laporan gangguan yang sedang ditangani atau tidak ada data yang dapat ditampilkan.

Tabel 4.23 *Use case scenario* melihat daftar laporan gangguan yang ditangani

Melihat Daftar Laporan Gangguan yang Ditangani	
Actor	Petugas Unit IT.
Objective	Untuk melihat daftar laporan gangguan yang ditangani.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor memilih menu penanganan. 2. Sistem akan menampilkan menu *dropdown.* 3. Aktor memilih submenu Laporan Gangguan Ditangani. 4. Sistem mengambil data laporan gangguan yang ditangani dari *database.* 5. Sistem akan menampilkan halaman daftar laporan gangguan ditangani yang menampilkan tabel dengan kolom nomor, kode gangguan, NIPP pelapor, jenis gangguan, lokasi gangguan, waktu lapor, status dan tombol "*detail*" dari setiap laporan gangguan.
Alternative Flow	5. Jika tidak ada laporan gangguan yang ditangani atau tidak ada data yang dapat ditampilkan maka sistem akan menampilkan "*No data available in tabel*" pada tabel.

Post Condition	Sistem akan menampilkan halaman daftar laporan gangguan ditangani yang menampilkan nomor, kode gangguan, NIPP pelapor, jenis gangguan, lokasi gangguan, waktu status, lapor dan tombol "*detail*" dari setiap laporan gangguan.

4.2.5.21 Melihat *Detail* Laporan Gangguan yang Ditangani – SPG_F_20

Use case scenario Melihat *Detail* Laporan Gangguan yang Ditangani dilihat pada Tabel 4.24. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat *Detail* Laporan Gangguan yang Ditangani.

Tabel 4.24 *Use case scenario* melihat *detail* laporan gangguan yang ditangani

Melihat *Detail* Laporan Gangguan yang Ditangani	
Actor	Petugas Unit IT.
Objective	Untuk melihat *detail* laporan gangguan yang ditangani.
Pre-Condition	Aktor telah masuk ke dalam sistem dan telah berada di halaman daftar laporan gangguan ditangani.
Main Flow	1. Aktor memilih laporan gangguan yang akan dilihat *detail*nya dengan menekan tombol *detail* pada data riwayat yang ingin dilihat *detail*nya. 2. Sistem akan menampilkan halaman *detail* laporan gangguan yang terdapat bagian *detail* laporan gangguan yang menampilkan NIPP pelapor, NIPP Petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, solusi, tombol "Ajukan *Lintas*" dan tombol "Tutup Laporan", bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan dan bagian pendataan barang *lintas* yang memperlihatkan barang atau

	perangkat apa saja yang sudah digunakan untuk menangani gangguan.
Alternative Flow	-
Post Condition	Sistem akan menampilkan halaman *detail* laporan gangguan yang terdapat bagian *detail* laporan gangguan yang menampilkan NIPP pelapor, NIPP Petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, solusi, tombol "Ajukan *Lintas*" dan tombol "Tutup Laporan", bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan dan bagian pendataan barang *lintas* yang memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan.

4.2.5.22 Mengajukan *Lintas* – SPG_F_21

Use case scenario Mengajukan *Lintas* dilihat pada Tabel 4.25. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mengajukan *Lintas*. Terdapat satu *alternatif flow* yaitu ketika saat tombol tidak ditekan.

Tabel 4.25 *Use case scenario* mengajukan *lintas*

Mengajukan *Lintas*	
Actor	Petugas Unit IT.
Objective	Untuk mengajukan *lintas.*
Pre-Condition	Aktor telah masuk ke dalam sistem dan telah berada di halaman *detail* dari sebuah laporan gangguan.
Main Flow	1. Aktor menekan tombol ajukan *lintas.* 2. Sistem akan menampilkan modal ajukan *lintas*. 3. Aktor menekan tombol Ya.

	4. Sistem akan menyimpan permohonan *lintas*, meng*update* status dari laporan gangguan menjadi mengajukan *lintas*, mengirimkan notifikasi kepada pimpinan Unit IT dan menampilkan kembali halaman *detail* dari laporan tersebut.
Alternative Flow	4. Jika aktor menekan tombol tidak maka sistem akan menutup modal ajukan *lintas.*
Post Condition	Sistem akan menyimpan permohonan *lintas*, meng*update* status dari laporan gangguan menjadi mengajukan *lintas*, mengirimkan notifikasi kepada pimpinan Unit IT dan menampilkan kembali halaman *detail* dari laporan tersebut.

4.2.5.23 Menutup Laporan – SPG_F_22

Use case scenario Menutup Laporan dilihat pada Tabel 4.26. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Menutup Laporan. Terdapat tiga *alternatif flow* yaitu ketika saat pilihan status gangguan bernilai tidak terselesaikan, ketika *field* status tidak diisi dan ketika tombol tidak ditekan.

Tabel 4.26 *Use case scenario* menutup laporan

Menutup Laporan	
Actor	Petugas Unit IT.
Objective	Untuk menutup laporan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan telah berada di halaman *detail* dari sebuah laporan gangguan.
Main Flow	1. Aktor menekan tombol tutup laporan. 2. Sistem akan menampilkan modal *form* tutup laporan yang berisikan *field* status. 3. Aktor memilih pilihan terselesaikan. 4. Sistem menampilkan *field* solusi.

	5. Aktor mengisikan solusi yang digunakan untuk mengatasi gangguan tersebut. 6. Aktor menekan tombol simpan. 7. Sistem menyimpan solusi, meng*update* status gangguan menjadi terselesaikan dan menampilkan halaman lihat daftar laporan gangguan ditangani.
Altertnative Flow	3. Jika aktor memilih pilihan tidak terselesaikan: a. Sistem tidak akan menampilkan *field* solusi. b. Sistem meng*update* status gangguan menjadi tidak terselesaikan dan menampilkan halaman lihat daftar laporan gangguan ditangani. 5. Aktor tidak mengisi *field* status maka akan menampilkan pesan *"Please select an item in the list".*
Post Condition	Sistem menyimpan solusi, meng*update* status gangguan menjadi terselesaikan dan menampilkan halaman lihat daftar laporan gangguan ditangani.

4.2.5.24 Melihat Daftar *Lintas* – SPG_F_23

Use case scenario Melihat Daftar *Lintas* dilihat pada Tabel 4.27. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat Daftar *Lintas*. Terdapat satu *alternatif flow* yaitu ketika tidak ada daftar *lintas* atau tidak ada data yang ditampilkan.

Tabel 4.27 *Use case scenario* melihat daftar *lintas*

Melihat Daftar *Lintas*	
Actor	Petugas Unit IT.
Objective	Untuk melihat daftar *lintas.*
Pre-Condition	Aktor telah masuk ke dalam sistem dan berada pada halaman utama.
Main Flow	1. Aktor memilih menu penanganan.

	2. Sistem menampilkan menu *dropdown.* 3. Aktor memilih submenu daftar *lintas*. 4. Sistem menampilkan halaman daftar *lintas* yang menampilkan tabel dengan kolom nomor, kode gangguan, NIPP pelapor, lokasi gangguan, jenis gangguan, status *lintas* dan catatan.
Altertnative Flow	4. Jika tidak ada daftar *lintas* atau tidak ada data yang ditampilkan maka sistem menampilkan "*No data available in tabel*" pada tabel.
Post Condition	Sistem menampilkan halaman daftar *lintas* yang menampilkan tabel dengan kolom nomor, kode gangguan, NIPP pelapor, lokasi gangguan, jenis gangguan, status *lintas* dan catatan.

4.2.5.25 Menambahkan Barang yang Telah Digunakan – SPG_F_24

Use case scenario Menambahkan Barang yang Telah Digunakan dilihat pada Tabel 4.28. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Menambahkan Barang yang Telah Digunakan. Terdapat tiga *alternative flow* yaitu ketika aktor memilih barang yang telah terdaftar pada *form* tambah barang *lintas*, ketika aktor mengisi jumlah barang lebih dari jumlah tersedia, dan ketika aktor tidak mengisi lengkap *form* tambah barang *lintas*.

Tabel 4.28 *Use case scenario* menambahkan barang yang telah digunakan

Menambahkan Barang yang Telah Digunakan	
Actor	Petugas Unit IT.
Objective	Untuk menambahkan barang yang telah digunakan pada suatu *lintas.*
Pre-Condition	Aktor telah masuk ke dalam sistem dan telah berada di halaman *detail* dari sebuah laporan gangguan.
Main Flow	1. Aktor menekan tombol tambah barang. 2. Sistem menampilkan modal daftar barang yang menampilkan tabel

	daftar barang yang terdiri dari kolom nama barang dan jumlah tersedia. 3. Aktor memilih barang-barang yang ingin dipilih. 4. Sistem menampilkan *form* tambah barang *lintas* yang terdiri dari nama barang, nama kode barang, kuantitas barang dan kode barang. 5. Aktor mengisi dengan lengkap *form* tambah barang *lintas.* 6. Aktor menekan tombol simpan. 7. Sistem menyimpan daftar barang ke dalam *database* dan menampilkan kembali halaman *detail* dari laporan gangguan tersebut.
Altertnative Flow	3. Jika aktor memilih barang yang sudah ada pada *form* tambah barang *lintas* akan menampilkan pesan "Perhatian! Nama barang sudah ada di dalam daftar". 5. Jika aktor mengisi jumlah barang lebih dari jumlah tersedia maka akan menampilkan "Perhatian! Anda tidak bisa melakukan pengambilan barang dengan jumlah LEBIH BESAR dari ketersediaan barang. Jumlah barang tersedia adalah (jumlah tersedia)". 5. Jika aktor tidak mengisi dengan lengkap *form* tambah barang *lintas* maka akan menampilkan pesan *"Please fill out this field".*
Post Condition	Sistem menyimpan daftar barang ke dalam *database* dan menampilkan kembali halaman *detail* dari laporan gangguan tersebut.

4.2.5.26 Melihat Daftar Barang di Gudang Penyimpanan – SPG_F_25

Use case scenario Melihat Daftar Barang di Gudang Penyimpanan dilihat pada Tabel 4.29. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT dan

Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat Daftar Barang di Gudang Penyimpanan. Terdapat satu *alternative flow* yaitu jika tidak terdapat barang pada daftar barang gudang penyimpanan atau tidak ada data untuk ditampilkan.

Tabel 4.29 *Use case scenario* melihat daftar barang di gudang penyimpanan

Melihat Daftar Barang di Gudang Penyimpanan	
Actor	Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk melihat daftar barang di gudang penyimpanan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor memilih menu Manajemen Gudang. 2. Sistem menampilkan halaman manajemen gudang penyimpanan yang didalamnya terdapat tabel yang memiliki kolom nomor nama barang, kuantitas, tombol *update* dan tombol hapus.
Altertnative Flow : jika tidak terdapat barang pada daftar barang gudang penyimpanan	2. Jika tidak terdapat barang pada daftar barang gudang penyimpan atau tidak ada data yang dapat ditampilkan maka sistem menampilkan "*No data available in tabel*" pada tabel.
Post Condition	Sistem menampilkan halaman manajemen gudang penyimpanan yang didalamnya terdapat tabel yang memiliki kolom nomor nama barang, kuantitas, tombol *update* dan tombol hapus.

4.2.5.27 Menambah Barang di Daftar Barang Gudang Penyimpanan – SPG_F_26

Use case scenario Menambah Barang di Daftar Barang Gudang Penyimpanan dilihat pada Tabel 4.30. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Menambah Barang di Daftar Barang Gudang Penyimpanan. Terdapat dua *alternative flow* yaitu saat tombol tidak ditekan dan *form* tambah barang tidak diisi lengkap.

Tabel 4.30 *Use case scenario* menambah barang di daftar barang gudang penyimpan

Menambah Barang di Daftar Barang Gudang Penyimpan	
Actor	Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk menambah barang di daftar barang gudang penyimpanan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan telah berada pada halaman manajemen gudang penyimpanan.
Main Flow	1. Aktor menekan tombol tambah barang. 2. Sistem menampilkan modal tambah barang yang berisikan *form* tambah barang yang terdiri dari beberapa *input*an yaitu nama barang dan kuantitas barang. 3. Aktor mengisi *form* tambah barang secara lengkap. 4. Aktor menekan tombol tambah. 5. Sistem akan menyimpan barang pada *database*, menampilkan pesan "Berhasil menambahkan barang" dan menampilkan halaman daftar barang gudang penyimpanan.
Altertnative Flow	3. Jika aktor tidak mengisi *form* dengan lengkap maka akan menampilkan pesan "*Please fill out this field*". 4. Jika aktor menekan tombol batal maka sistem akan menutup modal tambah barang.
Post Condition	Sistem akan menyimpan barang pada *database*, menampilkan pesan "Berhasil menambahkan barang" dan menampilkan halaman daftar barang gudang penyimpanan.

4.2.5.28 Meng*update* Barang di Daftar Barang Gudang Penyimpanan – SPG_F_27

Use case scenario Meng*update* Barang di Daftar Barang Gudang Penyimpanan dilihat pada Tabel 4.31. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Meng*update* Barang di Daftar Barang Gudang Penyimpanan. Terdapat dua *alternative flow* yaitu saat tombol tidak ditekan dan saat *form update* barang tidak diisi lengkap.

Tabel 4.31 *Use case scenario* meng*update* barang di daftar barang gudang penyimpanan

Meng*update* Barang di Daftar Barang Gudang Penyimpanan	
Actor	Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk meng*update* barang di daftar barang gudang penyimpanan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan telah berada pada halaman manajemen gudang penyimpanan.
Main Flow	1. Aktor memilih barang yang akan di*update.* 2. Aktor menekan tombol edit. 3. Sistem menampilkan modal edit barang yang berisikan *form* tambah barang yang terdiri dari beberapa *input*an yaitu nama barang dan kuantitas barang dengan nilai sama dengan data barang sebelumnya. 4. Aktor mengisikan *field* yang akan diedit. 5. Aktor menekan tombol *update.* 6. Sistem akan meng*update* data barang pada *database*, menampilkan pesan "Berhasil meng*update* barang" dan menampilkan halaman manajemen gudang penyimpanan.
Altertnative Flow	4. Jika aktor mengosongkan *field* pada *form* maka sistem akan menampilkan pesan "*Please fill out this field*".

	5. Jika aktor menekan tombol tidak maka akan menutup modal edit barang.
Post Condition	Sistem akan meng*update* data barang pada *database*, menampilkan pesan "Berhasil meng*update* barang" dan menampilkan halaman manajemen gudang penyimpanan.

4.2.5.29 Menghapus Barang di Daftar Barang Gudang Penyimpanan – SPG_F_28

Use case scenario Menghapus Barang di Daftar Barang Gudang Penyimpanan dilihat pada Tabel 4.32. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Menghapus Barang di Daftar Barang Gudang Penyimpanan. Terdapat satu *alternative flow* yaitu saat tombol tidak ditekan.

Tabel 4.32 *Use case scenario* menghapus barang di daftar barang gudang penyimpanan

Menghapus Barang di Daftar Barang Gudang Penyimpanan	
Actor	Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk menghapus barang di daftar barang gudang penyimpanan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan telah berada pada halaman manajemen gudang penyimpanan.
Main Flow	1. Aktor memilih barang yang akan dihapus. 2. Aktor menekan tombol hapus. 3. Sistem menampilkan modal hapus barang. 4. Aktor menekan tombol hapus. 5. Sistem akan menghapus data barang dari *database*, menampilkan pesan "Berhasil menghapus barang" dan menampilkan halaman manajemen gudang penyimpanan.

Altertnative Flow	4. Jika aktor menekan tombol tidak maka sistem menutup modal hapus barang.
Post Condition	Sistem akan menghapus data barang dari *database*, menampilkan pesan "Berhasil menghapus barang" dan menampilkan halaman manajemen gudang penyimpanan.

4.2.5.30 Melihat Daftar Riwayat Laporan Gangguan – SPG_F_29

Use case scenario Melihat Daftar Riwayat Laporan Gangguan dilihat pada Tabel 4.33 Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat Daftar Riwayat Laporan Gangguan. Terdapat satu *alternative flow* yaitu jika tidak terdapat riwayat laporan gangguan.

Tabel 4.33 *Use case scenario* melihat daftar riwayat laporan gangguan

Melihat Daftar Riwayat Laporan Gangguan	
Actor	Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk melihat daftar riwayat laporan gangguan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor memilih menu Riwayat Gangguan. 2. Sistem menampilkan *dropdown.* 3. Aktor memilih riwayat laporan gangguan. 4. Sistem menampilkan *dropdown.* 5. Aktor memilih daftar riwayat. 6. Sistem akan menampilkan halaman daftar riwayat laporan gangguan yang terdiri dari *form filter* yang terdiri dari *input*an tanggal awal dan tanggal akhir serta tabel yang berisi kolom nomor, kode gangguan, NIPP pelapor, NIPP petugas, jenis gangguan, lokasi, status dan tombol *detail.*

Altertnative Flow :	4. Jika tidak terdapat riwayat laporan gangguan maka sistem menampilkan "*No data available in tabel*" pada tabel.
Post Condition	Sistem akan menampilkan halaman daftar riwayat laporan gangguan yang terdiri dari *form filter* yang terdiri dari *input*an tanggal dari dan tanggal sampai serta tabel yang berisi kolom nomor, kode gangguan, NIPP pelapor, NIPP petugas, jenis gangguan, lokasi, status dan tombol *detail.*

4.2.5.31 Mengekspor Daftar Riwayat Laporan Gangguan – SPG_F_30

Use case scenario Mengekspor Daftar Riwayat Laporan Gangguan dilihat pada Tabel 4.34. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mengekspor Daftar Riwayat Laporan Gangguan.

Tabel 4.34 *Use case scenario* mengekspor daftar riwayat laporan gangguan

Mengekspor Daftar Riwayat Laporan Gangguan	
Actor	Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk mengekspor daftar riwayat laporan gangguan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sudah berada pada halaman daftar riwayat laporan gangguan.
Main Flow	1. Aktor menekan tombol "*Eksport to Excel*". 2. Komputer aktor akan mengunduh *file* hasil ekspor.
Altertnative Flow	-
Post Condition	Komputer aktor akan mengunduh *file* hasil ekspor.

4.2.5.32 Melihat *Detail* Riwayat Laporan Gangguan – SPG_F_31

Use case scenario Melihat *Detail* Riwayat Laporan Gangguan dilihat pada Tabel 4.35. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat *Detail* Riwayat Laporan Gangguan.

Tabel 4.35 *Use case scenario* melihat *detail* riwayat laporan gangguan

Melihat *Detail* Riwayat Laporan Gangguan	
Actor	Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk melihat *detail* riwayat laporan gangguan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sudah berada pada halaman daftar riwayat laporan gangguan.
Main Flow	1. Aktor memilih riwayat laporan gangguan yang ingin dilihat *detail*nya dengan menekan tombol *detail.* 2. Sistem akan menampilkan halaman *detail* laporan gangguan yang terdapat bagian *detail* laporan gangguan yang menampilkan NIPP pelapor, NIPP Petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, dan solusi, bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan dan bagian pendataan barang *lintas* yang memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan.
Altertnative Flow	-
Post Condition	Sistem akan menampilkan halaman *detail* laporan gangguan yang terdapat bagian *detail* laporan gangguan yang menampilkan NIPP pelapor, NIPP Petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, dan solusi, bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang

	menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan dan bagian pendataan barang *lintas* yang memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan.

4.2.5.33 Melihat Profil – SPG_F_32

Use case scenario Melihat Profil dilihat pada Tabel 4.36. Aktor yang terlibat dalam *use case* ini adalah Pelapor, Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat Profil.

Tabel 4.36 *Use case scenario* melihat profil

Melihat Profil	
Actor	Pelapor, Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk melihat profil.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor menekan foto profil yang ada pada pojok kanan. 2. Sistem menampilkan menu *dropdown.* 3. Aktor memilih profil. 4. Sistem menampilkan halaman profil yang menampilkan nama lengkap, NIPP, unit kerja, jabatan, kedudukan, *email*, nomor telepon aktor, tombol edit profil, tombol edit foto profil, dan tombol edit *password*.
Altertnative Flow	-
Post Condition	Sistem menampilkan halaman profil yang menampilkan nama lengkap, NIPP, unit kerja, jabatan, kedudukan, *email*, nomor telepon aktor, tombol edit profil, tombol edit foto profil, dan tombol edit *password*.

4.2.5.34 Mengedit Profil – SPG_F_33

Use case scenario Mengedit Profil dilihat pada Tabel 4.37. Aktor yang terlibat dalam *use case* ini adalah Pelapor, Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mengedit Profil. Terdapat dua *alternative flow* yaitu ketika tombol batal ditekan dan ketika aktor mengosongkan *field input* dari *form* edit profil.

Tabel 4.37 *Use case scenario* mengedit profil

Mengedit Profil	
Actor	Pelapor, Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk mengedit profil.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sudah pada halaman profil.
Main Flow	1. Aktor menekan tombol edit profil. 2. Sistem akan menampilkan *section* edit profil yang terdiri dari *input*an nama lengkap, unit kerja, jabatan, kedudukan, *email*, dan nomor telepon dari aktor yang nilainya sama dengan data profil sebelum diedit. 3. Aktor mengedit data profil yang ingin diedit. 4. Aktor menekan tombol simpan. 5. Sistem meng*update* profil pengguna pada *database* dan menampilkan halaman profil.
Altertnative Flow	3. Ketika aktor mengosongkan *input*an maka sistem akan menampilkan pesan "*Please fill out this field*". 4. Ketika aktor menekan tombol batal maka sistem akan menampilkan *section* profil dan menutup *section* edit profil.
Post Condition	Sistem meng*update* profil pengguna pada *database* dan menampilkan halaman profil.

4.2.5.35 Mengedit *Password* – SPG_F_34

Use case scenario Mengedit *Password* dilihat pada Tabel 4.38. Aktor yang terlibat dalam *use case* ini adalah Pelapor, Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mengedit *Password*. Terdapat empat *alternative flow* yaitu ketika tombol batal ditekan, ketika password lama tidak sesuai, ketika *field password* baru kurang dari 6 karakter dan ketika *field password* baru tidak sesuai dengan *field* ulangi *password* baru.

Tabel 4.38 *Use case scenario* mengedit *password*

Mengedit *Password*	
Actor	Pelapor, Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk mengedit *password.*
Pre-Condition	Aktor telah masuk ke dalam sistem dan sudah pada halaman profil.
Main Flow	1. Aktor menekan tombol edit *password.* 2. Sistem menampilkan modal edit *password* yang didalamnya terdapat *form* edit *password* yang didalamnya terdapat *field password* lama, *password* baru dan ulangi *password* baru. 3. Aktor mengisi *field password* lama sesuai dengan password sebelum diedit. 4. Aktor mengisi *field password* baru lebih dari atau sama dengan 6 karakter. 5. Aktor mengisi *field* ulangi *password* baru sesuai dengan *field password* baru. 6. Aktor menekan tombol simpan. 7. Sistem meng*update password* aktor pada *database* dan menampilkan halaman profil.
Altertnative Flow	3. Jika aktor mengisi *field password* lama tidak sesuai dengan *password* sebelum diedit maka sistem akan

	menampilkan pesan "*Password* tidak sesuai". 4. Jika aktor mengisi *field password* baru kurang dari 6 maka sistem akan menampilkan pesan "*Password* harus memiliki minimal 6 karakter". 5. Jika aktor mengisi *field* ulangi *password* tidak sesuai dengan *field password* baru maka sistem akan menampilkan pesan "*Password* baru dan ulangi *password* tidak sesuai". 6. Jika aktor menekan tombol batal maka sistem akan menutup modal edit *password.*
Post Condition	Sistem meng*update password* aktor pada *database* dan menampilkan halaman profil.

4.2.5.36 Mengedit Foto Profil – SPG_F_35

Use case scenario Mengedit Foto Profil dilihat pada Tabel 4.39. Aktor yang terlibat dalam *use case* ini adalah Pelapor, Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mengedit Foto Profil. Terdapat dua *alternative flow* yaitu ketika tombol batal ditekan dan ketika *form* edit foto profil tidak diisi lengkap.

Tabel 4.39 *Use case scenario* mengedit foto profil

Mengedit Foto Profil	
Actor	Pelapor, Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk mengedit foto profil.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sudah pada halaman profil.
Main Flow	1. Aktor menekan tombol edit foto profil. 2. Sistem menampilkan modal edit foto profil yang berisi *form* edit foto profil yang terdiri dari satu *field input*an yaitu foto baru. 3. Aktor memilih foto profil baru. 4. Aktor menekan tombol simpan.

	5. Sistem meng*update* foto profil aktor pada *database* dan menampilkan halaman profil.
Altertnative Flow	2. Jika aktor tidak mengisikan dengan lengkap *form* edit profil maka sistem akan menampilkan pesan "*Please select a file*". 4. Jika aktor menekan tombol batal maka sistem akan menutup modal edit foto profil.
Post Condition	Sistem meng*update* foto profil aktor pada *database* dan menampilkan halaman profil.

4.2.5.37 Melihat Grafik Riwayat Laporan Gangguan Bulanan – SPG_F_36

Use case scenario Melihat Grafik Riwayat Laporan Gangguan Bulanan dilihat pada Tabel 4.40. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat Grafik Riwayat Laporan Gangguan Bulanan.

Tabel 4.40 *Use case scenario* melihat grafik riwayat laporan gangguan bulanan

Melihat Grafik Riwayat Laporan Gangguan Bulanan	
Actor	Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk melihat grafik riwayat laporan gangguan bulanan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor memilih menu Riwayat Laporan Gangguan. 2. Sistem menampilkan *dropdown.* 3. Aktor memilih grafik riwayat. 4. Sistem akan menampilkan *dropdown.* 5. Aktor memilih Grafik Perbulan. 6. Sistem akan menampilkan halaman grafik riwayat laporan gangguan perbulan yang terdiri dari grafik riwayat laporan gangguan bulanan per stasiun dan grafik riwayat jenis

	gangguan pada laporan gangguan bulanan pada bulan saat itu.
Altertnative Flow	-
Post Condition	Sistem akan menampilkan halaman grafik riwayat laporan gangguan perbulan yang terdari dari grafik riwayat laporan gangguan bulanan per stasiun dan grafik riwayat jenis gangguan pada laporan gangguan bulanan pada bulan saat itu.

4.2.5.38 Mencari Akun – SPG_F_38

Use case scenario Mencari Akun dilihat pada Tabel 4.41. Aktor yang terlibat dalam *use case* ini adalah Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mencari Akun. Ada satu *alternative flow* yaitu ketika tidak ada akun yang cocok sesuai *keyword* yang dimasukkan.

Tabel 4.41 *Use case scenario* mencari akun

Mencari Akun	
Actor	Pimpinan Unit IT.
Objective	Untuk mencari akun pada daftar akun.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman daftar akun.
Main Flow	1. Aktor memasukkan *keyword* di dalam *input search.* 2. Sistem mencari data akun yang memiliki kesamaan terhadap *keyword* pada NIPP, nama lengkap, unit kerja, jabatan, nomor telepon, tipe atau *email.* 3. Sistem menampilkan data akun pada tabel daftar akun.
Altertnative Flow	3. Jika sistem tidak menemukan akun yang mengandung kesamaan dengan *keyword* yang dimasukkan maka sistem akan menampilkan "*No data available in tabel*" pada tabel.
Post Condition	Sistem menampilkan data akun pada tabel daftar akun.

4.2.5.39 Mengekspor Daftar Akun – SPG_F_39

Use case scenario Mengekspor Daftar Akun dilihat pada Tabel 4.42. Aktor yang terlibat dalam *use case* ini adalah Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mengekspor Daftar Akun.

Tabel 4.42 *Use case scenario* mengekspor daftar akun

Mengekspor Daftar Akun	
Actor	Pimpinan Unit IT.
Objective	Untuk mengunduh data akun ke dalam perangkat aktor.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman daftar akun.
Main Flow	1. Aktor menekan tombol *Export to Excel.* 2. Sistem akan mengekspor data akun menjadi *file excel* dan perangkat aktor akan mengunduh *file* tersebut.
Altertnative Flow	-
Post Condition	Sistem akan mengekspor data akun menjadi *file excel* dan perangkat aktor akan mengunduh *file* tersebut.

4.2.5.40 Mencari Permohonan *Lintas* – SPG_F_40

Use case scenario Mencari Permohonan *Lintas* dilihat pada Tabel 4.43. Aktor yang terlibat dalam *use case* ini adalah Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mencari Permohonan *Lintas*. Ada satu *alternative flow* yaitu ketika tidak ada data yang cocok sesuai *keyword* yang dimasukkan.

Tabel 4.43 *Use case scenario* mencari permohonan *lintas*

Mencari Permohonan *Lintas*	
Actor	Pimpinan Unit IT.
Objective	Untuk mencari permohonan *lintas* pada daftar permohonan *lintas.*
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman daftar permohonan *lintas.*

Main Flow	1. Aktor memasukkan *keyword* di dalam *input search.* 2. Sistem mencari data permohonan *lintas* yang memiliki kesamaan terhadap *keyword* pada kode gangguan, NIPP petugas, lokasi gangguan, atau jenis gangguan. 3. Sistem menampilkan data permohonan *lintas* pada tabel daftar permohonan *lintas.*
Altertnative Flow	3. Jika sistem tidak menemukan permohonan *lintas* yang mengandung kesamaan dengan *keyword* yang dimasukkan maka sistem akan menampilkan "*No data available in tabel*" pada tabel.
Post Condition	Sistem menampilkan data permohonan *lintas* pada tabel daftar permohonan *lintas.*

4.2.5.41 Mencari Laporan Gangguan yang telah Terlaporkan – SPG_F_41

Use case scenario Mencari Laporan Gangguan yang telah Terlaporkan dilihat pada Tabel 4.44. Aktor yang terlibat dalam *use case* ini adalah Pelapor. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mencari Laporan Gangguan yang telah Terlaporkan. Ada satu *alternative flow* yaitu ketika tidak ada data yang cocok sesuai *keyword* yang dimasukkan.

Tabel 4.44 *Use case scenario* mencari laporan gangguan yang telah terlaporkan

Mencari Laporan Gangguan yang telah Terlaporkan	
Actor	Pelapor.
Objective	Untuk mencari laporan gangguan yang terlaporkan pada daftar laporan gangguan yang terlaporkan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman daftar laporan gangguan yang terlaporkan.
Main Flow	1. Aktor memasukkan *keyword* di dalam *input search.* 2. Sistem mencari data permohonan *lintas* yang memiliki kesamaan

	terhadap *keyword* pada kode gangguan, NIPP pelapor, NIPP petugas, lokasi gangguan, jenis gangguan, waktu lapor atau status. 3. Sistem menampilkan data daftar laporan gangguan yang terlaporkan pada tabel daftar laporan gangguan yang terlaporkan.
Altertnative Flow	3. Jika sistem tidak menemukan permohonan *lintas* yang mengandung kesamaan dengan *keyword* yang dimasukkan maka sistem akan menampilkan "*No data available in tabel*" pada tabel.
Post Condition	Sistem menampilkan data daftar laporan gangguan yang terlaporkan pada tabel daftar laporan gangguan yang terlaporkan.

4.2.5.42 Mencari Laporan Gangguan yang Ditangani – SPG_F_42

Use case scenario Mencari Laporan Gangguan yang Ditangani dilihat pada Tabel 4.45. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mencari Laporan Gangguan yang Ditangani. Ada satu *alternative flow* yaitu ketika tidak ada data yang cocok sesuai *keyword* yang dimasukkan.

Tabel 4.45 *Use case scenario* mencari laporan gangguan yang ditangani

Mencari Laporan Gangguan yang Ditangani	
Actor	Petugas Unit IT.
Objective	Untuk mencari laporan gangguan yang ditangani pada daftar laporan gangguan ditangani.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman daftar laporan gangguan ditangani.
Main Flow	1. Aktor memasukkan *keyword* di dalam *input search.* 2. Sistem mencari data laporan gangguan yang ditangani yang memiliki kesamaan terhadap *keyword* pada kode gangguan, NIPP

	pelapor, lokasi gangguan, jenis gangguan, waktu lapor atau status. 3. Sistem menampilkan data daftar laporan gangguan yang ditangani pada tabel daftar laporan gangguan yang ditangani.
Altertnative Flow	3. Jika sistem tidak menemukan laporan gangguan yang ditangani yang mengandung kesamaan dengan *keyword* yang dimasukkan maka sistem akan menampilkan "*No data available in tabel*" pada tabel.
Post Condition	Sistem menampilkan data daftar laporan gangguan yang ditangani pada tabel daftar laporan gangguan yang ditangani.

4.2.5.43 Mencari *Lintas* – SPG_F_43

Use case scenario Mencari *Lintas* dilihat pada Tabel 4.46. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mencari *Lintas*. Ada satu *alternative flow* yaitu ketika tidak ada data yang cocok sesuai *keyword* yang dimasukkan.

Tabel 4.46 *Use case scenario* mencari *lintas*

Mencari *Lintas*	
Actor	Petugas Unit IT.
Objective	Untuk mencari data *lintas* pada daftar *lintas.*
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman daftar *lintas.*
Main Flow	1. Aktor memasukkan *keyword* di dalam *input search.* 2. Sistem mencari data *lintas* yang memiliki kesamaan terhadap *keyword* pada kode gangguan, NIPP pelapor, lokasi gangguan, jenis gangguan, status *lintas* atau catatan. 3. Sistem menampilkan data *lintas* pada tabel daftar *lintas.*

Altertnative Flow	3. Jika sistem tidak menemukan *lintas* yang mengandung kesamaan dengan *keyword* yang dimasukkan maka sistem akan menampilkan "*No data available in tabel*" pada tabel.
Post Condition	Sistem menampilkan data *lintas* pada tabel daftar *lintas.*

4.2.5.44 Mencari Barang – SPG_F_44

Use case scenario Mencari Barang dilihat pada Tabel 4.47. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mencari Barang. Ada satu *alternative flow* yaitu ketika tidak ada data yang cocok sesuai *keyword* yang dimasukkan.

Tabel 4.47 *Use case scenario* mencari barang

Mencari Barang	
Actor	Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk mencari data barang pada daftar barang gudang penyimpanan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman daftar barang gudang penyimpanan.
Main Flow	1. Aktor memasukkan *keyword* di dalam *input search.* 2. Sistem mencari data barang yang memiliki kesamaan terhadap nama barang atau kuantitas. 3. Sistem menampilkan data barang pada tabel daftar barang gudang penyimpanan.
Altertnative Flow	3. Jika sistem tidak menemukan barang yang mengandung kesamaan dengan *keyword* yang dimasukkan maka sistem akan menampilkan "*No data available in tabel*" pada tabel.
Post Condition	Sistem menampilkan data barang pada tabel daftar barang gudang penyimpanan.

4.2.5.45 Mencari Riwayat Laporan Gangguan – SPG_F_45

Use case scenario Mencari Riwayat Laporan Gangguan dilihat pada Tabel 4.48. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Mencari Riwayat Laporan Gangguan. Ada satu *alternative flow* yaitu ketika tidak ada data yang cocok sesuai *keyword* yang dimasukkan.

Tabel 4.48 *Use case scenario* mencari riwayat laporan gangguan

Mencari Riwayat Laporan Gangguan	
Actor	Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk mencari data riwayat laporan gangguan pada daftar riwayat laporan gangguan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman daftar riwayat laporan gangguan.
Main Flow	1. Aktor memasukkan *keyword* di dalam *input search.* 2. Sistem mencari data riwayat laporan gangguan yang memiliki kesamaan terhadap kode barang, NIPP pelapor, NIPP petugas, jenis gangguan, lokasi, atau status. 3. Sistem menampilkan data riwayat laporan gangguan pada tabel daftar riwayat laporan gangguan.
Altertnative Flow	3. Jika sistem tidak menemukan riwayat laporan gangguan yang mengandung kesamaan dengan *keyword* yang dimasukkan maka sistem akan menampilkan "*No data available in tabel*" pada tabel.
Post Condition	Sistem menampilkan data riwayat laporan gangguan pada tabel daftar riwayat laporan gangguan.

4.2.5.46 Melihat Grafik Riwayat Laporan Gangguan Tahunan – SPG_F_46

Use case scenario Melihat Grafik Riwayat Laporan Gangguan Tahunan dilihat pada Tabel 4.49. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Melihat Grafik Riwayat Laporan Gangguan Tahunan.

Tabel 4.49 *Use case scenario* melihat grafik riwayat laporan gangguan tahunan

Melihat Grafik Riwayat Laporan Gangguan Tahunan	
Actor	Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk melihat grafik riwayat laporan gangguan bulanan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman utama.
Main Flow	1. Aktor memilih menu Riwayat Laporan Gangguan. 2. Sistem menampilkan *dropdown.* 3. Aktor memilih grafik riwayat. 4. Sistem akan menampilkan *dropdown.* 5. Aktor memilih Grafik Pertahun. 6. Sistem akan menampilkan halaman grafik riwayat laporan gangguan tahunan berdasarkan jenis gangguan pertahunnya.
Altertnative Flow	-
Post Condition	Sistem akan menampilkan halaman grafik riwayat laporan gangguan tahunan berdasarkan jenis gangguan pertahunnya.

4.2.5.47 *Filter* Daftar Riwayat Laporan Gangguan Berdasarkan Tanggal – SPG_F_47

Use case scenario Filter Daftar Riwayat Laporan Gangguan Berdasarkan Tanggal dilihat pada Tabel 4.50. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case Filter* Daftar Riwayat Laporan Gangguan Berdasarkan Tanggal. Terdapat satu *alternative flow* yaitu pada saat tidak ada data pada *range* yang diberikan.

Tabel 4.50 *Use case scenario filter* daftar riwayat laporan gangguan berdasarkan tanggal

Filter Daftar Riwayat Laporan Gangguan Berdasarkan Tanggal	
Actor	Petugas Unit IT, Pimpinan Unit IT.

Objective	Untuk melakukan *filter* pada daftar riwayat laporan gangguan berdasarkan tanggal.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman daftar riwayat laporan gangguan.
Main Flow	1. Aktor mengisi *field input* tanggal awal. 2. Aktor mengisi *field input* tanggal akhir. 3. Aktor menekan tombol *filter.* 4. Sistem mencari data riwayat laporan gangguan yang berada diantara tanggal awal dan tanggal akhir. 5. Sistem menampilkan data riwayat laporan gangguan yang berada diantara tanggal awal dan tanggal akhir pada tabel daftar riwayat laporan gangguan.
Altertnative Flow	5. Jika tidak ada data pada *range* yang diberikan maka sistem akan menampilkan "*No data available in tabel*" pada tabel.
Post Condition	Sistem menampilkan data riwayat laporan gangguan yang berada diantara tanggal awal dan tanggal akhir pada tabel daftar riwayat laporan gangguan.

4.2.5.48 Menampilkan Grafik Riwayat Laporan Bulanan Berdasarkan Bulan – SPG_F_48

Use case scenario Menampilkan Grafik Riwayat Laporan Bulanan Berdasarkan Bulan dilihat pada Tabel 4.51. Aktor yang terlibat dalam *use case* ini adalah Petugas Unit IT dan Pimpinan Unit IT. *Use case scenario* ini menggambarkan alur yang terjadi pada *use case* Menampilkan Grafik Riwayat Laporan Bulanan Berdasarkan Bulan. Terdapat satu *alternative flow* yaitu ketika pada bulan tersebut tidak ada data riwayat laporan gangguan.

Tabel 4.51 *Use case scenario* menampilkan grafik riwayat laporan bulanan berdasarkan bulan

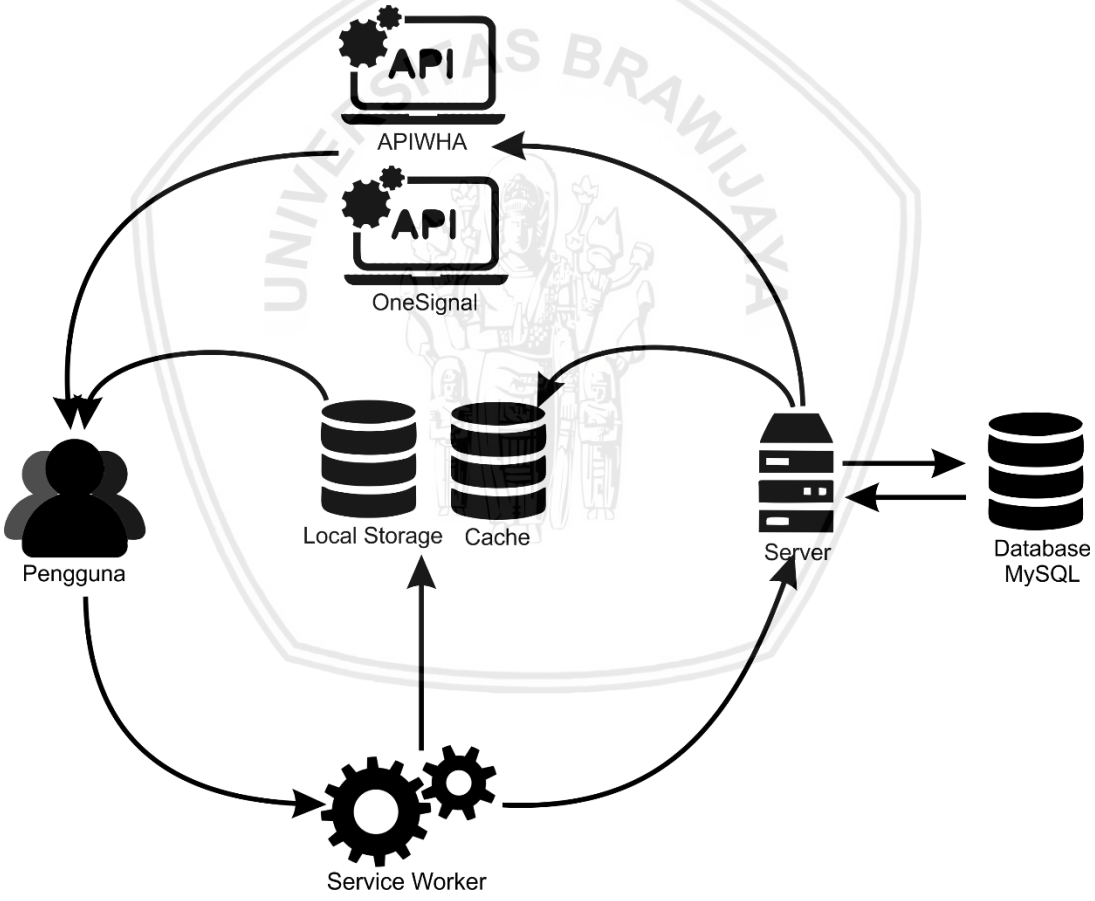
Menampilkan Grafik Riwayat Laporan Bulanan Berdasarkan Bulan	
Actor	Petugas Unit IT, Pimpinan Unit IT.
Objective	Untuk menampilkan grafik riwayat laporan bulanan berdasarkan bulan.
Pre-Condition	Aktor telah masuk ke dalam sistem dan sedang berada pada halaman grafik riwayat laporan gangguan perbulan.
Main Flow	1. Aktor mengisi *field input* bulan tahun. 2. Aktor menekan tombol pilih. 3. Sistem melakukan proses *generate* grafik riwayat laporan gangguan pada bulan yang sesuai dengan *input*an aktor. 4. Sistem menampilkan halaman grafik riwayat laporan gangguan perbulan berdasarkan *input*an bulan tahun yang dimasukan.
Altertnative Flow	4. Jika tidak ditemukan riwayat laporan gangguan pada bulan dan tahun yang di*input*kan oleh aktor maka sistem akan menampilkan grafik kosong.
Post Condition	Sistem menampilkan halaman grafik riwayat laporan gangguan perbulan berdasarkan *input*an bulan tahun yang dimasukan.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab ini akan menjelaskan hasil perancangan dan implementasi dari sistem pelaporan gangguan. Tahap-tahap perancangan yang dilakukan adalah perancangan arstiktur sistem, perancangan *sequence diagram*, perancangan *class diagram*, perancangan komponen, perancangan *database* dan perancangan antarmuka. Sedangkan pada bagian implementasi akan menjelaskan tentang spesifikasi sistem yang digunakan untuk membuat sistem baik hardware maupun software, implementasi komponen, implementasi database dan implementasi antarmuka.

5.1 Perancangan Sistem

5.1.1 Perancangan Arsitektur Sistem



Gambar 5.1 Aristektur Sistem

Sistem pelaporan gangguan dibuat dengan berbasis *web* yang menggunakan teknologi *progressive web application* yang mengaplikasikan *application shell architecture*. *Platform web* dipilih berdasarkan hasil elisitasi yang mengatakan bahwa sebagian besar sistem pada unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon menggunakan *web*, sehingga diharapkan dengan membangun sistem pelaporan gangguan dengan berbasis *web* maka akan memudahkan dalam proses

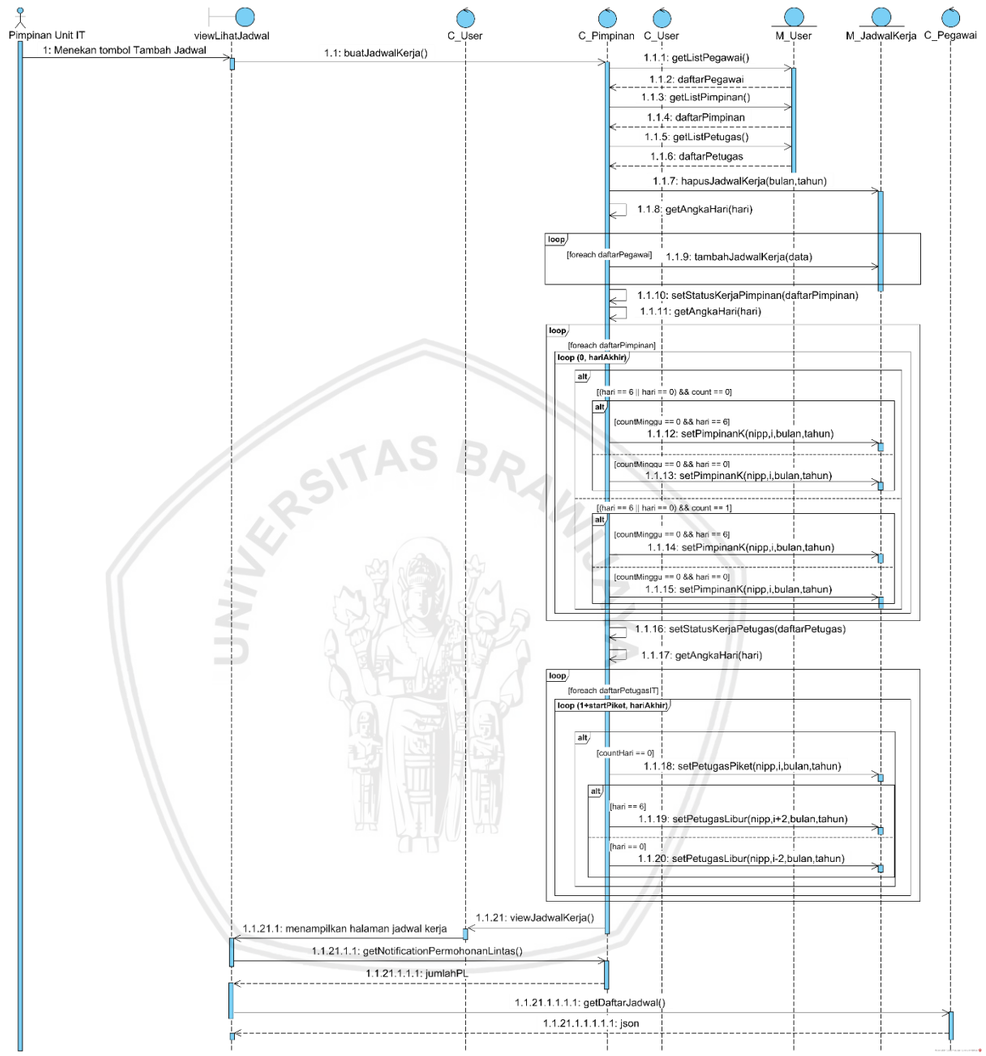
integrasi dengan sistem yang ada. Kemudian alasan mengapa menggunakan teknologi *progressive web application* adalah karena untuk memberikan kemudahan akses dari aplikasi sendiri, melihat dengan keunggulan yang dimilikinya yang dapat membuat *web* dapat dioperasikan layaknya aplikasi *native* dan mendukung jaringan yang kurang stabil. Hal ini akan sangat mendukung untuk kegiatan *lintas* yang dilakukan petugas unit IT.

Teknologi *progressive web application* tidak jauh berbeda dengan *web* pada umunya, hanya saja *progressive web application* harus memiliki 3 syarat utama yaitu *web app manifest, service worker*s dan harus dijalankan diatas HTTPS. *Web app manifest* merupakan *file JSON* yang memberitahukan *browser* tentang aplikasi *web* yang dijalankan seperti bagaimana harusnya berperilaku ketika dilakukan instalasi pada perangkat selular atau *desktop*, ini diperlukan untuk memunculkan permintaan "Tambahkan ke Layar Beranda". *Service workers* digunakan untuk melakukan *caching* pada *browser* sehingga dapat mempercepat waktu pengaksesan selanjutnya.

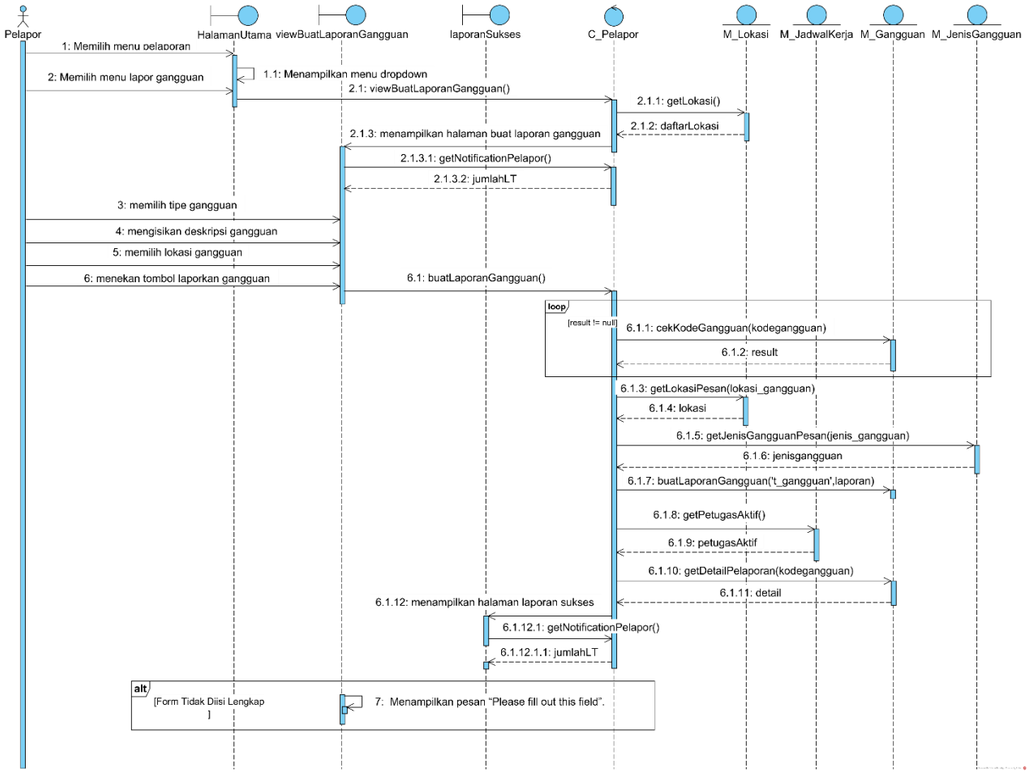
Application shell itu sendiri adalah HTML, CSS, dan JavaScript minimal yang mendukung antarmuka pengguna. *Application shell architecture* membagi bagian inti infrastruktur aplikasi dan *user interface* dari data. Semua *user interface* dan infrastruktur di *cache* secara lokal dengan menggunakan *service workers* sehingga pada pengaksesan selanjutnya, alih-alih harus memuat semuanya, *progressive web application* hanya perlu mengambil data yang diperlukan. Hal ini memungkinkan sistem ini dapat meningkatkan kecepatan pengaksesan dan memberikan keuntungan *native application* seperti *instant loading* dan *regular update* tanpa harus membutuhkan toko aplikasi atau *app store*. *User interface* dan infrastruktur aplikasi akan disimpan pada *cache storage* pada *browser* sedangkan data-data yang bersifat dinamis akan disimpan pada *local storage* yang ada pada *browser* setiap kali dilakukan *fetching* data. Hal ini bertujuan supaya aplikasi tetap bisa berjalan ketika dalam keadaan jaringan yang kurang baik atau mungkin tidak terdapat jaringan sama sekali.

Untuk memberikan notifikasi kepada pengguna sistem, penulis menggunakan bantuan API dari *OneSignal* dan API dari *APIWHA*. API *OneSignal* digunakan karena dapat mendukung sistem untuk meberikan notifikasi kepada pengguna tertentu saja. Notifikasi yang diberikan oleh *OneSignal* ini berbentuk *web push notification*. *APIWHA* juga digunakan untuk memberikan notifikasi, namun bedanya notifikasi yang diberikan dalam bentuk pesan yang disampaikan melalui aplikasi *WhatsApp*. Hal ini dipilih berdasarkan dari hasil elisitasi yang mendapatkan bahwa semua pegawai unit IT menggunakan aplikasi *WhatsApp*. Untuk membuat aplikasi ini dapat berjalan diatas HTTPS, penulis menggunakan jasa pemberi sertifikat *Let's Encrypt* yang disediakan oleh penyedia *hosting* yang digunakan. Untuk gambaran arsitektur sistem dapat dilihat pada Gambar 5.1.

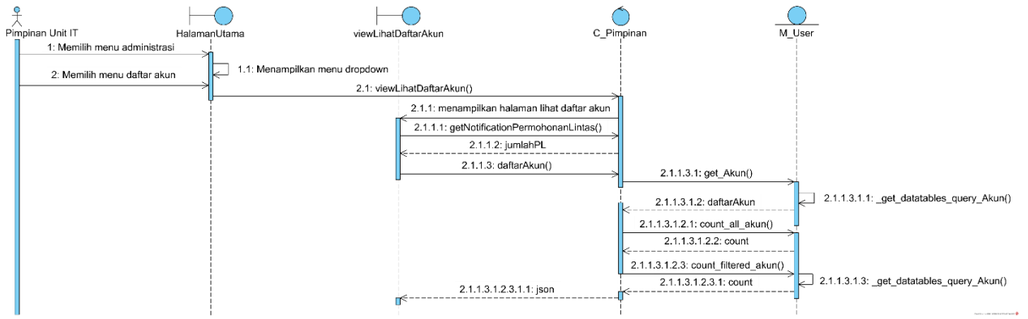
5.1.2 Perancangan *Sequence Diagram*



Gambar 5.2 *Sequence diagram* menambahkan jadwal kerja



Gambar 5.3 *Sequence diagram* membuat laporan gangguan



Gambar 5.4 *Sequence diagram* melihat daftar akun

Perancangan *sequence diagram* bertujuan untuk menggambarkan perilaku antara objek di dalam bentuk diagram alir. *Sequence diagram* mengacu pada *usecase diagram* dan *usecase scenario*. Pada penelitian ini penulis membuat 48 *sequence diagram*, namun hanya 3 *sequence diagram* yang dicantumkan dalam skripsi ini.

5.1.2.1 *Sequence Diagram* Menambahkan Jadwal Kerja

Sequence diagram menambahkan jadwal kerja digambarkan pada Gambar 5.2. Untuk penjelasan dari setiap nomor *sequence* yang ada pada *sequence diagram* menambahkan jadwal kerja akan dijelaskan pada Tabel 5.1. *Sequence diagram* menambahkan jadwal kerja dimulai saat aktor menekan tombol "Tambah Jadwal" di kelas *view* viewLihatJadwal yang akan memanggil fungsi buatJadwalKerja() pada kelas *controller* C_Pimpinan. Setelah itu pada kelas *controller* C_Pimpinan terjadi pemanggilan fungsi getListPegawai() dari kelas *model* M_User yang akan mengembalikan variabel daftarPegawai. Setelah itu dilanjutkan dengan pemanggilan fungsi getListPimpinan() dari kelas *model* M_User yang akan mengembalikan variabel daftarPimpinan. Lalu dilanjutkan dengan memanggil fungsi daftarPegawaiIT() dari kelas *model* M_User yang akan mengembalikan variabel daftarPegawai. Fungsi-fungsi yang dipanggil tadi digunakan untuk mendapatkan daftar seluruh pegawai berdasarkan jabatannya.

Setelah itu dipanggil *method* hapusJadwalKerja() dari kelas *model* M_JadwalKerja yang digunakan untuk menghapus jadwal kerja sebelumnya pada bulan tersebut yang sudah ada. Setelah itu memanggil fungsi getAngkaHari() yang digunakan untuk mengambil bentuk angka dari suatu hari. Setelah itu akan memulai perulangan untuk menginisialisasi hari kerja dan hari libur dengan menggunakan fungsi tambahJadwalKerja() yang ada pada kelas M_JadwalKerja.

Fungsi setStatusKerjaPimpinan() yang ada pada kelas M_JadwalKerja digunakan untuk menginisialisasi jadwal dinas posko. Dinas posko disini adalah berjaga di kantor pada hari sabtu dan hari minggu. Sedangkan fungsi setStatusKerjaPegawai() pada kelas M_JadwalKerja digunakan untuk menginisiasliasi jadwal *ready on call* atau piket malam. Pemanggilan fungsi getNotificationPermohonanLintas() dilakukan untuk mengambil jumlah notifikasi permohonan lintas yang masuk. Pemanggilan fungsi getDaftarJadwal() digunakan untuk mengambil daftar jadwal kerja untuk ditampilkan.

Tabel 5.1 Tabel penjelasan *sequence diagram* menambahkan jadwal kerja

No	Nomor *Sequence*	Keterangan
1.	1	Pengguna melakukan interaksi dengan sistem dengan cara menekan tombol Tambah Jadwal.
2.	1.1	Pemanggilan fungsi buatJadwalKerja() yang merupakan fungsi utama untuk menambahkan jadwal kerja.

No	Nomor *Sequence*	Keterangan
3.	1.1.1	Pemanggilan fungsi getListPegawai() yang digunakan untuk mengambil daftar pegawai di unit IT.
4.	1.1.2	daftarPegawai merupakan nilai kembalian dari fungsi getListPegawai yang berisi data semua pegawai unit IT.
5.	1.1.3	Pemanggilan fungsi getListPimpinan() yang digunakan untuk mengambil daftar pimpinan di unit IT.
6.	1.1.4	daftarPimpinan merupakan nilai kembalian dari fungsi getListPimpinan() yang berisi data semua pimpinan di unit IT.
7.	1.1.5	Pemanggilan fungsi getListPetugas() yang digunakan untuk mengambil daftar petugas di unit IT.
8.	1.1.6	daftarPetugas merupakan nilai kembalian dari fungsi getListPetugas() yang berisi data semua petugas di unit IT.
9.	1.1.7	Pemanggilan fungsi hapusJadwalKerja() yang digunakan untuk menghapus bila sudah ada jadwal kerja pada bulan tersebut.
10.	1.1.8	Pemanggilan fungsi getAngkaHari() untuk menentukan angka dari hari awal bulan.
11.	1.1.9	Pemanggilan fungsi tambahJadwalKerja() yang digunakan untuk menge*set* jadwal kerja dan jadwal libur seluruh pegawai unit iT.
12.	1.1.10	Pemanggilan fungsi setStatusKerjaPimpinan() yang digunakan untuk menge*set* status piket posko mingguan pimpinan.
13.	1.1.11	Pemanggilan fungsi getAngkaHari() untuk menentukan angka dari hari awal bulan.
14.	1.1.12	Pemanggilan *method* setPimpinanK() yang digunakan untuk menge*set* status kerja menjadi ‘K’.
15.	1.1.13	Pemanggilan *method* setPimpinanK() yang digunakan untuk menge*set* status kerja menjadi ‘K’.

No	Nomor *Sequence*	Keterangan
16.	1.1.14	Pemanggilan *method* setPimpinanK() yang digunakan untuk menge*set* status kerja menjadi 'K'.
17.	1.1.15	Pemanggilan *method* setPimpinanK() yang digunakan untuk menge*set* status kerja menjadi 'K'.
18.	1.1.16	Pemanggilan fungsi setStatusKerjaPetugas() yang digunakan untuk menge*set* status *ready on call* atau piket malam petugas.
19.	1.1.17	Pemanggilan fungsi getAngkaHari() untuk menentukan angka dari hari awal bulan.
20.	1.1.18	Pemanggilan *method* setPetugasPiket() yang digunakan untuk menge*set* status kerja petugas menjadi 'P/R'.
21.	1.1.19	Pemanggilan *method* setPetugasLibur() yang digunakan untuk menge*set* status kerja petugas menjadi 'L'.
22.	1.1.20	Pemanggilan *method* setPetugasLibur() yang digunakan untuk menge*set* status kerja petugas menjadi 'L'.
23.	1.1.21	Pemanggilan fungsi viewJadwalKerja() yang digunakan untuk menampilkan halaman jadwal kerja.
24.	1.1.21.1	Sistem menampilkan halaman jadwal kerja.
25.	1.1.21.1.1	Pemanggilan fungsi getNotificationPermohonanLintas() dilakukan untuk mengambil jumlah notifikasi permohonan lintas yang masuk
26	1.1.21.1.1.1	Nilai kembalian dari pemanggilan fungsi getNotificationPermohonanLintas()
27	1.1.21.1.1.1.1	Pemanggilan fungsi getDaftarJadwal() digunakan untuk mengambil daftar jadwal kerja untuk ditampilkan
28	1.1.21.1.1.1.1.1	Nilai kembalian dari pemanggilan fungsi getDaftarJadwal()

5.1.2.2 *Sequence Diagram* Membuat Laporan Gangguan

Sequence diagram membuat laporan gangguan digambarkan pada Gambar 5.3. Untuk penjelasan dari setiap nomor *sequence* yang ada pada *sequence diagram* membuat laporan gangguan akan dijelaskan pada Tabel 5.2. *Sequence diagram* membuat laporan dimulai dengan memilih menu pelaporan yang akan membuat sistem menampilkan menu *dropdown*. Setelah menu *dropdown* keluar kita pilih menu lapor gangguan yang akan memanggil fungsi viewBuatLaporanGangguan() pada kelas C_Pelapor yang berfungsi untuk menampilkan halaman buat laporan gangguan. Pada halaman buat laporan gangguan terdapat pemanggilan fungsi getNotificationPelapor() untuk mengambil jumlah laporan gangguan yang belum selesai diatasi.

Setelah itu aktor memilih tipe gangguan, mengisikan deskripsi gangguan, memilih lokasi gangguan dan menekan tombol laporkan gangguan yang akan memanggil fungsi buatLaporanGangguan() pada kelas C_Pelapor. Di dalam fungsi buatLaporanGangguan() terdapat proses *generate* kode gangguan yang akan dilanjutkan dengan pemanggilan fungsi cekKodeGangguan() yang terdapat pada kelas M_Gangguan yang digunakan untuk mengecek apakah kode tersebut sudah digunakan untuk laporan gangguan lain atau belum. Kemudian akan dipanggil fungsi getLokasiPesan() pada kelas M_Lokasi yang digunakan untuk mengambil nama lokasi untuk pengiriman pesan yang dilanjutkan dengan pemanggilan fungsi getJenisGangguanPesan() pada kelas M_JenisGangguan yang digunakan untuk mengambil nama jenis gangguan untuk pengiriman pesan. Lalu setelah itu dipanggil *method* buatLaporanGangguan() pada kelas M_Gangguan yang berguna untuk menyimpan laporan gangguan pada *database*. Lalu jika nilai kembalian dari *method* buatLaporanGangguan() lebih dari atau sama dengan satu maka setelah itu ada pemanggilan fungsi getPetugasAktif() milik kelas M_JadwalKerja yang digunakan untuk mencari petugas unit IT yang sedang bekerja untuk dikirimkan notifikasi. Yang terakhir terdapat pemanggilan fungsi getDetailPelaporan() yang digunakan untuk mengambil *detail* laporan untuk ditampilkan pada halaman laporan sukses. Jika nilai kembalian dari fungsi buatLaporanGangguan() lebih dari satu maka sistem akan menampilkan halaman laporan gangguan sukses, selain itu maka sistem akan menampilkan halaman buat laporan gangguan kembali.

Tabel 5.2 Tabel penjelasan *sequence diagram* membuat laporan gangguan

No	Nomor *Sequence*	Keterangan
1.	1	Aktor memilih menu pelaporan.
2.	1.1	Sistem akan menampilkan menu *dropdown.*
3.	2	Aktor memilih menu lapor gangguan.
4.	2.1	Pemanggilan fungsi viewBuatLaporanGangguan() yang digunakan untuk menampilkan halaman buat laporan gangguan.

No	Nomor *Sequence*	Keterangan
5.	2.1.1	Pemanggilan fungsi getLokasi() yang digunakan untuk mengambil nama-nama lokasi gangguan yang terdaftar dalam sistem.
6.	2.1.2	daftarLokasi adalah nilai kembalian dari pemanggilan fungsi getLokasi() yang berisi nama-nama lokasi gangguan yang terdaftar dalam sistem.
7.	2.1.3	Sistem menampilkan halaman buat laporan gangguan.
8.	2.1.3.1	Pemanggilan fungsi getNotificationPelapor() untuk mengambil jumlah laporan gangguan yang belum terselesaikan
9.	2.1.3.2	Nilai kembalian dari pemanggilan fungsi getNotificationPelapor()
10.	3	Aktor memilih tipe gangguan.
11.	4	Aktor mengisikan deskripsi gangguan.
12.	5	Aktor memilih lokasi gangguan.
13.	6	Aktor menekan tombol Laporkan Gangguan.
14.	6.1	Pemanggilan fungsi buatLaporanGangguan() yang digunakan untuk memulai proses pelaporan gangguan.
15.	6.1.1	Pemanggilan fungsi cekKodeGangguan() yang digunakan untuk mengecek apakah kode yang telah di*generate* oleh sistem sudah terdaftar dalam *database* atau belum.
16.	6.1.2	result merupakan nilai kembalian dari pemanggilan fungsi cekKodeGangguan() yang berisi jumlah data yang memiliki kodegangguan yang sama.
17.	6.1.3	Pemanggilan fungsi getLokasiPesan() yang digunakan untuk mengambil nama lokasi untuk pengiriman pesan notifikasi.
18.	6.1.4	lokasi merupakan nilai kembalian dari pemanggilan fungsi getLokasiPesan() yang berisi nama lokasi untuk pengiriman pesan notifikasi.

No	Nomor *Sequence*	Keterangan
19.	6.1.5	Pemanggilan fungsi getJenisGangguanPesan() yang digunakan untuk mengambil nama jenis gangguan untuk pengiriman pesan notifikasi.
20.	6.1.6	Jenisgangguan merupakan nilai kembalian dari pemanggilan fungsi getJenisGangguanPesan() yang berisi nama jenis gangguan untuk pengiriman pesan notifikasi.
21.	6.1.7	Pemanggilan fungsi buatLaporanGangguan() yang digunakan untuk menyimpan data laporan gangguan kedalam *database.*
22.	6.1.8	Pemanggilan fungsi getPetugasAktif() yang digunakan untuk mengambil data petugas yang sedang aktif atau sedang bekerja saat itu.
23.	6.1.9	Nilai kembalian dari pemanggilan fungsi getPetugasAktif() yang berisi data petugas yang sedang aktif atau sedang bekerja saat itu.
24.	6.1.10	Pemanggilan fungsi getDetailPelaporan() yang digunakan untuk mengambil *detail* dari sebuah laporan gangguan.
25.	6.1.11	detail merupakan nilai kembalian dari fungsi getDetailPelaporan() yang berisi *detail* dari sebuah laporan gangguan.
26.	6.1.12	Sistem menampilkan halaman laporan sukses.
27.	6.1.12.1	Pemanggilan fungsi getNotificationPelapor() untuk mengambil jumlah laporan gangguan yang belum terselesaikan
28	6.1.12.1.1	Nilai kembalian dari pemanggilan fungsi getNotificationPelapor()
29	6.2	Menampilkan pesan "*Please fill out this field".*

5.1.2.3 *Sequence Diagram* Melihat Daftar Akun

Sequence diagram melihat daftar akun digambarkan pada Gambar 5.4. Untuk penjelasan dari setiap nomor *sequence* yang ada pada *sequence diagram* melihat daftar akun akan dijelaskan pada Tabel 5.3. *Sequence diagram* melihat daftar akun dimulai dengan memilih menu administrasi pada halaman utama. Lalu setelah sistem menampilkan menu *dropdown* pilih menu daftar akun. Dengan memilih menu daftar akun maka akan menjalankan fungsi viewLihatDaftarAkun() milik kelas C_Pimpinan. Kemudian sistem akan menampilkan halaman lihat daftar akun. Pada saat memuat halaman lihat daftar akun, ada dua *script javascript* yang

dijalankan yaitu yang pertama pemanggilan fungsi getNotificationPermohonanLintas() dilakukan untuk mengambil jumlah notifikasi permohonan lintas yang masuk. Yang kedua pemanggilan fungsi daftarAkun() milik kelas C_Pimpinan.

Selanjutnya didalam fungsi tersebut terdapat pemanggilan fungsi get_Akun() milik kelas M_User yang bertujuan untuk mengambil data *user* di *database*. Pada fungsi get_Akun() dipanggil fungsi _get_datatables_query_Akun() yang digunakan untuk mendapatkan *query*. Setelah data *user* didapatkan, pada fungsi daftarAkun() terdapat pemanggilan fungsi count_all_akun() milik kelas M_User yang bertujuan untuk mengambil jumlah semua akun. Selanjutnya dipanggil fungsi count_filetered_akun() milik kelas M_User yang nantinya didalam fungsi itu akan dipanggil kembali fungsi _get_datatables_query_Akun(). Kegunaan fungsi count_fileterd_akun() ini adalah untuk mendapatkan jumlah data yang ter*filter* dari total semua jumlah akun.

Tabel 5.3 Tabel penjelasan *sequence diagram* melihat daftar akun

No	Nomor *Sequence*	Keterangan
1.	1	Aktor memilih menu administrasi.
2.	1.1	Sistem menampilkan menu *dropdown.*
3.	2	Aktor memilih menu daftar akun.
4.	2.1	Pemanggilan fungsi viewLihatDaftarAkun yang digunakan untuk menampilkan halaman lihat daftar akun.
5.	2.1.1	Sistem menampilkan halaman daftar akun.
6.	2.1.1.1	Pemanggilan fungsi getNotificationPermohonanLintas() dilakukan untuk mengambil jumlah notifikasi permohonan lintas yang masuk
7.	2.1.1.2	Nilai kembalian dari pemanggilan fungsi getNotificationPermohonanLintas()
6.	2.1.1.3	Pemanggilan fungsi daftarAkun() yang digunakan untuk memulai proses mengambil daftar akun dari *database*. Fungsi ini dipanggil menggunakan AJAX.
8.	2.1.1.3.1	Pemanggilan fungsi get_Akun() yang digunakan untuk mengambil data dari *database*.
9.	2.1.1.3.1.1	Pemanggilan fungsi _get_datatables_query_Akun() yang berfungsi sebagai inisialisasi *query database.*

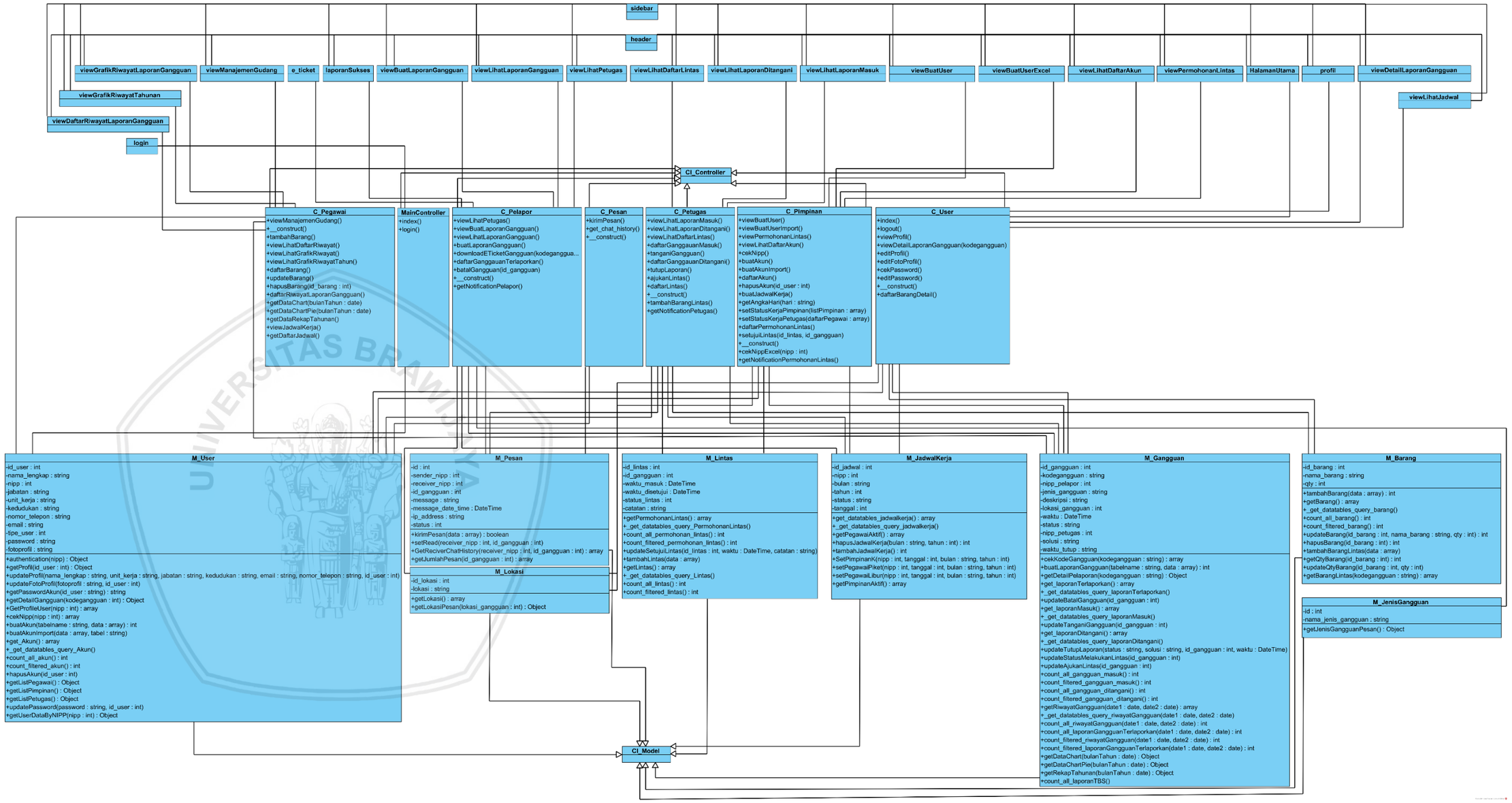
No	Nomor *Sequence*	Keterangan
10.	2.1.1.3.1.2	daftarAkun merupakan nilai kembalian dari pemanggilan fungsi get_Akun() yang berisi daftar akun yang ada di *database.*
11.	2.1.1.3.1.2.1	Pemanggilan fungsi count_all_akun() yang digunakan untuk mengambil jumlah data akun yang ada pada *database.*
12.	2.1.1.3.1.2.2	count merupakan nilai kembalian dari pemanggilan fungsi count_all_akun() yang berisi jumlah data akun yang ada pada *database.*
13.	2.1.1.3.1.2.3	Pemanggilan fungsi count_filtered_akun() yang digunakan untuk mengambil jumlah data akun yang ada pada *database* yang sudah ter*filter* oleh *input*an aktor pada *datatables.*
14.	2.1.1.3.1.3	Pemanggilan fungsi _get_datatables_query_Akun() yang berfungsi sebagai inisialisasi *query database.*
15.	2.1.1.3.1.2.3.1	count merupakan nilai kembalian dari pemanggilan fungsi count_ filtered _akun() yang berisi jumlah data akun yang ada pada *database* yang sudah ter*filter* oleh *input*an aktor pada *datatables.*
16.	2.1.1.3.1.2.3.1.1	Json merupakan hasil pemanggilan dari fungsi daftarAkun() yang akan ditampilkan pada *datatables.*

5.1.3 Perancangan *Class Diagram*

Dalam bagian perancangan *class diagram* ini akan menunjukkan objek yang terdapat pada suatu sistem serta relasi antara objek-objek tersebut. Selain itu *class diagram* juga menggambarkan atribut dan operasi yang dimiliki oleh setiap objek tersebut. Pada Gambar 5.5 digambar *class diagram* Sistem Pelaporan Gangguan.

Perancangan *class diagram* sistem Pelaporan Gangguan ini dibuat dengan menerapkan pola perancangan MVC (*Model-View-Controller*). Pada *class diagram* pada Gambar 5.5 terdapat kelas-kelas *view* saling berasosiasi dengan kelas-kelas *controller*, kelas-kelas *controller* yang saling berasosisasi dengan kelas *view* dan juga ada kelas *model*. Ada 23 kelas *view* yaitu *header* yang digunakan sebagai bagian *header*, *sidebar* yang digunakan sebagai bagian *menu bar*, *e-ticket* yang digunakan sebagai tampilan untuk membuat *e-ticket* pelaporan gangguan, laporanSukses yang digunakan untuk halaman ketika laporan berhasil terlaporkan, viewBuatLaporanGangguan yang digunakan sebagai tampilan untuk membuat

laporan gangguan, viewLihatLaporanGangguan yang digunakan untuk menampilkan laporan yang sudah terlaporkan, viewLihatPetugas yang digunakan untuk melihat daftar petugas yang sedang aktif atau bertugas, viewLihatDaftarLintas yang digunakan untuk melihat *lintas* yang sudah diajukan, viewLihatLaporanDitangani yang digunakan untuk melihat daftar laporan yang sedang ditangani, viewLihatLaporanMasuk yang digunakan untuk melihat laporan yang masuk ke unit IT, viewBuatUser yang digunakan untuk membuat akun baru, viewBuatUserExcel yang digunakan untuk membuat akun baru dengan *import* excel, viewLihatDaftarAkun yang digunakan untuk melihat daftar akun, viewPermohonanLintas yang digunakan untuk melihat daftar permohonan *lintas*, HalamanUtama yang digunakan untuk menampilkan halaman utama sistem, profil yang digunakan untuk menampilkan profil pengguna, viewDetailLaporanGangguan yang digunakan untuk menampilkan dari sebuah laporan, *login* yang digunakan untuk menampilkan *form login*, viewDaftarRiwayatLaporanGangguan yang digunakan untuk menampilkan daftar riwayat laporan gangguan, viewGrafikRiwayatLaporanGangguan yang digunakan untuk menampilkan grafik riwayat laporan gangguan perbulanviewGrafikRiwayatTahunan yang digunakan untuk menampilkan grafik riwayat laporan gangguan tahunan, viewManajemenGudang yang digunakan untuk menampilkan daftar barang gudang penyimpanan dan viewLihatJadwal yang digunakan untuk menampilkan halaman jadwal kerja. Pada *class diagram* tersebut terdapat 7 buah kelas *controller* yaitu C_Pelapor, C_Pesan, C_Pimpinan, C_User, C_Pegawai dan MainController. Kelas-kelas *controller* tersebut penulis pisahkan berdasarkan aktor yang menggunakannya. Pada *class diagram* tersebut terdapat 8 buah kelas *model* yaitu M_User, M_Lintas, M_Gangguan, M_JadwalKerja, M_Lokasi, M_Barang, M_JenisGangguan dan M_Pesan. Kelas-kelas model yang disebutkan tadi merupakan representasi entitas-entitas yang ada pada *database*.



Gambar 5.5 C*lass diagram* sistem pelaporan gangguan

5.1.4 Perancangan Komponen

Dalam perancangan komponen ini akan menjelaskan proses yang terjadi pada beberapa bagian dari sub-sistem untuk menjalankan fungsionalitas sistem. Perancangan komponen yang dibuat digunakan sebagai acuan dalam membuat kode program pada tahap implementasi. Dalam perancangan komponen ini, akan dijelaskan 3 perancangan komponen yaitu fungsi buatAkunImport(), fungsi buatLaporanGangguan() dan fungsi cekNipp().

5.1.4.1 Perancangan Komponen Fungsi buatAkunImport()

Berikut ini merupakan perancangan komponen untuk fungsi buatAkunImport() yang dapat dilihat pada Tabel 5.4. Dalam perancangan komponen ini dilakukan proses pengambilan data dari *file excel* yang nantinya akan disimpan di dalam *database*.

Nama Kelas : C_Pimpinan

Nama Fungsi : buatAkunImport()

Tabel 5.4 *Pseudocode* fungsi buatAkunImport()

***Pseudocode* fungsi buatAkunImport()**	
1	Mulai
2	Inisialisasi variabel path = path dari file excel Inisialisasi variabel object = hasil pemanggilan fungsi load(path) Inisialisasi variabel data dengan tipe array Inisialisasi nipp_terdaftar dengan tipe array
3	FOREACH setiap workshet yang ada di file excel
4	Inisialisasi highestRow = hasil pemanggilan fungsi getHighestRow()
5	FOR nilai awal row = 2 hingga row <= highestRow, row++
6	Inisialisasi variabel nipp = hasil dari pemanggilan fungsi getCellByColumnAndRow(0, row) untuk mengambil nilai pada kolom 0 pada row tersebut Inisialisasi variabel nama_lengkap = hasil dari pemanggilan fungsi getCellByColumnAndRow(1, row) untuk mengambil nilai pada kolom 1 pada row tersebut Inisialisasi variabel unit_kerja = hasil dari pemanggilan fungsi getCellByColumnAndRow(2, row) untuk mengambil nilai pada kolom 2 pada row tersebut Inisialisasi variabel jabatan = hasil dari pemanggilan fungsi getCellByColumnAndRow(3, row) untuk mengambil nilai pada kolom 3 pada row tersebut Inisialisasi variabel kedudukan = hasil dari pemanggilan fungsi getCellByColumnAndRow(4, row) untuk mengambil nilai pada kolom 4 pada row tersebut

***Pseudocode* fungsi buatAkunImport()**	
	Inisialisasi variabel nomor_telepon = hasil dari pemanggilan fungsi getCellByColumnAndRow(5, row) untuk mengambil nilai pada kolom 5 pada row tersebut Inisialisasi variabel tipe_user = hasil dari pemanggilan fungsi getCellByColumnAndRow(6, row) untuk mengambil nilai pada kolom 6 pada row tersebut Inisialisasi variabel email = hasil dari pemanggilan fungsi getCellByColumnAndRow(7, row) untuk mengambil nilai pada kolom 7 pada row tersebut Inisialisasi variabel fotoprofil = assets/img/fotoprofil/defaultprofile.jpg Inisialisasi variabel password = $2y$10$KKxo3PjBIqeBykuVAeqmM.DRK5xyPVGXVUjZVh6amZUpO9Prbymci Inisialisasi variabel count dengan nilai sama dengan nilai kembalian dari pemanggilan fungsi cekNippExcel(nipp)
7	IF nipp != null
8	IF count == 0
9	Inisialisasi variabel data dengan nilai array yang berisikan nipp, nama_lengkap, unit_kerja, jabatan, kedudukan, nomor_telepon, tipe_user, email, fotoprofil, dan password
10	ELSE Pemanggilan method array_push(nipp_terdaftar,nipp)
11	END IF
12	ELSE Continue
13	END IF
14	END FOR
15	END FOREACH
16	IF nipp_data != null
17	IF nipp_terdaftar != null
18	Menampilkan pesan impor akun gagal karena ada nipp yang telah terdaftar
19	ELSE Pemanggilan fungsi untuk menyimpan data akun ke dalam database Menampilkan pesan akun berhasil diimpor
20	END IF
21	ELSE Menampilkan pesan tidak ada data di dalam file excel
22	END IF

***Pseudocode* fungsi buatAkunImport()**	
23	Selesai

5.1.4.2 Perancangan Komponen Fungsi buatLaporanGangguan()

Berikut ini merupakan perancangan komponen untuk fungsi buatLaporanGangguan() yang dapat dilihat pada Tabel 5.5. Dalam perancangan komponen ini dilakukan proses pengambilan *input*an dari pelapor yang nantinya akan disimpan dalam *database*.

Nama Kelas : C_Pelapor

Nama Fungsi : buatLaporanGangguan()

Tabel 5.5 *Pseudocode* fungsi buatLaporanGangguan()

***Pseudocode* fungsi buatLaporanGangguan()**	
1	Mulai
2	Set waktu lokal menjadi waktu Indonesia Inisialisasi nipp_pelapor = variabel nipp dari session user yang sedang login Inisialisasi waktu = hasil kembalian dari pemanggilan fungsi date('Y-m-d H:i:s') Inisialisasi variabel kodegangguan = hasil kembalian dari pemanggilan fungsi random_string('alnum',6) Inisialiasasi variabel result = hasil kembalian dari pemanggilan fungsi cekKodeGangguan(kodegangguan)
3	WHILE result !=null
4	Inisialisasi variabel kodegangguan = hasil kembalian dari pemanggilan fungsi random_string('alnum',6) Inisialiasasi variabel result = hasil kembalian dari pemanggilan fungsi cekKodeGangguan(kodegangguan)
5	END WHILE
6	Inisialisasi variabel jenis_gangguan = nilai dari inputan form dengan nilai variabel name = jenis_gangguan Inisialisasi variabel deskripsi = nilai dari inputan form dengan nilai variabel name = deskripsi Inisialisasi variabel lokasi_gangguan = nilai dari inputan form dengan nilai variabel name = lokasi gangguan_gangguan Inisialisasi variabel status dengan nilai = "Belum Ditangani" Inisialisasi variabel array laporan = array("nipp_pelapor"=> nipp_pelapor, "kodegangguan" => kodegangguan, "jenis_gangguan" => jenis_gangguan, "deskripsi" => deskripsi,

***Pseudocode* fungsi buatLaporanGangguan()**	
	"lokasi_gangguan"=> lokasi_gangguan, "waktu" => waktu, "status" => status); Inisialisasi variabel lokasi dengan nilai sama dengan nilai hasil kembalian dari pemanggilan fungsi getLokasiPesan() Inisialisasi variabel jenisgangguan dengan nilai sama dengan nilai hasil kembalian dari pemanggilan fungsi getJenisGangguanPesan() Inisialisasi variabel message dengan nilai "Ada jenisgangguan terjadi di lokasi dengan NIPP pelapor: nipp pelapor pada waktu Pemanggilan metod buatLaporanGangguan() dengan parameter variabel laporan Inisialisasi variabel url dengan nilai sama dengan nilai kembalian dari fungsi site_url() Inisialisasi variabel petugasAktif sama dengan hasil kembalian dari pemanggilan fungsi getPetugasAktif()
7	FOREACH variabel petugasAktif
8	Mengirim notifikasi kepada petugas aktif
9	END FOREACH
10	Inisialisasi data['laporan'] = hasil pemanggilan dari method getDetailPelaporan(kodegangguan) Menampilkan halaman laporanSukses
11	Selesai

5.1.4.3 Perancangan Komponen Fungsi cekNipp()

Berikut ini merupakan perancangan komponen untuk fungsi cekNipp() yang dapat dilihat pada Tabel 5.6. Dalam perancangan komponen ini dilakukan proses pengecekan antara NIPP yang di*input*kan oleh Pimpinan Unit IT dengan NIPP yang ada pada *database*.

Nama Kelas : C_Pimpinan

Nama Fungsi : cekNipp()

Tabel 5.6 *Pseudocode* fungsi cekNipp()

***Pseudocode* fungsi cekNipp()**	
1	Mulai
2	Inisialisasi nipp = nilai inputan Pimpinan Inisialisasi exist = kembalian dari pemanggilan fungsi cekNipp(nipp) Inisialisasi count = hasil pemanggilan fungsi count(exists)

***Pseudocode* fungsi cekNipp()**	
	Inisialisasi data['state'] = 0
3	IF count > 0
4	Inisialisasi data['state'] = 1
5	ELSE Inisialisasi data['state'] = 0
6	END IF
7	Cetak hasil pemanggilan fungsi json_encode(data)
8	Selesai

5.1.5 Perancangan *Database*

Perancangan *database* pada sistem pelaporan gangguan ini digambarkan pada Gambar 5.6 yang menggambarkan *Conceptual Data Model* dari sistem pelaporan gangguan unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon. Pada perancangan *database* sistem pelaporan gangguan terdapat 9 entitas yaitu t_gangguan, t_jadwalkerja, t_lintas, t_lokasi, t_pesan, t_barang, t_gangguan_barang, t_jenisgangguan dan t_user. Entitas t_gangguan memiliki 11 atribut dan berhubungan dengan entitas t_gangguan_barang, t_jenisgangguan, t_lintas, entitas t_lokasi, entitas t_user dan entitas t_pesan. Entitas t_jadwalkerja memiliki 6 atribut dan memiliki hubungan dengan entitas t_user. Entitas t_lintas memiliki 6 atribut dan memiliki hubungan dengan entitas t_gangguan. Entitas t_lokasi memiliki 2 atribut dan memiliki hubungan dengan entitas t_gangguan dan t_user. Entitas t_pesan memiliki 8 atribut dan memiliki hubungan dengan entitas t_user dan entitas t_gangguan. Entitas t_user memiliki 11 atribut dan memiliki hubungan dengan entitas t_gangguan, entitas t_jadwalkerja, t_lokasi dan entitas t_pesan. Entitas t_barang memiliki 3 atribut dan memiliki hubungan dengan entitas t_gangguan_barang. Entitas t_gangguan barang memilik 3 atribut dan memiliki hubungan dengan t_barang dan t_gangguan. Entitas t_jenisganggaun memiliki 2 atribut dan memiliki hubungan dengan entitas t_gangguan. Keterangan dari setiap atribut pada setiap entitas dapat dilihat pada Tabel 5.7.

Tabel 5.7 Penjelasan atribut dari entitas pada *conceptual data model*

No	Nama Atribut	Nama Entitas	Keterangan
1.	id_jadwal	t_jadwalkerja	*Primary key* dari entitas t_jadwalkerja, yang diinisialisasi secara inkremen.
2.	nipp	t_jadwalkerja	Merupakan atribut yang menandakan pegawai yang memiliki jadwal tersebut.

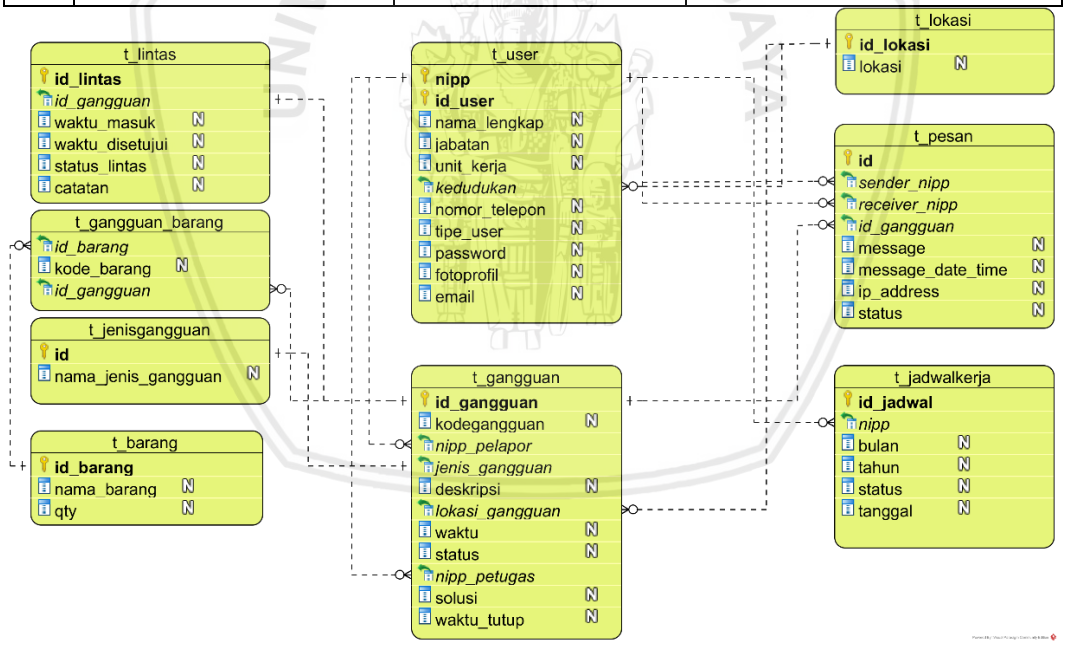
No	Nama Atribut	Nama Entitas	Keterangan
3.	bulan	t_jadwalkerja	Merupakan atribut yang menandakan pada bulan apa jadwal tersebut.
4.	tahun	t_jadwalkerja	Merupakan atribut yang menandakan pada tahun apa jadwal tersebut.
5.	status	t_jadwalkerja	Merupakan atribut yang menandakan status kerja pada jadwal tersebut.
6.	tanggal	t_jadwalkerja	Merupakan atribut yang menandakan pada tanggal berapa jadwal tersebut.
7.	id_lokasi	t_lokasi	Merupakan *primary key* dari entitas t_lokasi sebagai penanda lokasi dan diinisialisasi secara inkremen.
8.	lokasi	t_lokasi	Merupakan atribut yang menjelaskan nama dari suatu lokasi.
9.	id_lintas	t_lintas	Merupakan *primary key* dari entitas t_lintas sebagai penanda *lintas* dan diinisialisasi secara inkremen.
10.	id_gangguan	t_lintas	Merupakan *foreign key* dari entitas t_gangguan yang menyatakan gangguan mana yang terhubung dengan *lintas* tersebut.
11.	waktu_masuk	t_lintas	Merupakan atribut yang menjelaskan kapan pengajuan *lintas* masuk.
12.	waktu_disetujui	t_lintas	Merupakan atribut yang menjelaskan kapan pengajuan *lintas* disetujui.

No	Nama Atribut	Nama Entitas	Keterangan
13.	status_lintas	t_lintas	Merupakan atribut yang menjelaskan status dari sebuah pengajuan *lintas.*
14.	catatan	t_lintas	Merupakan atribut yang menjelaskan catatan dari pimpinan unit IT untuk sebuah *lintas.*
15.	id	t_pesan	Merupakan *primary key* dari entitas t_pesan sebagai penanda pesan dan diinisialisasi secara inkremen.
16.	sender_nipp	t_pesan	Merupakan atribut yang merupakan *foreign key* dari entitas t_user yang menyatakan nipp dari pengirim pesan.
17.	receiver_nipp	t_pesan	Merupakan atribut yang merupakan *foreign key* dari entitas t_user yang menyatakan nipp dari penerima pesan.
18.	id_gangguan	t_pesan	Merupakan *foreign key* dari entitas t_gangguan yang menyatakan gangguan mana yang terhubung dengan pesan tersebut.
19.	message	t_pesan	Atribut yang menjelaskan isi dari suatu pesan.
20.	message_date_time	t_pesan	Atribut yang menjelaskan kapan suatu pesan dikirim.
21.	ip_address	t_pesan	Atribut yang menjelaskan *ip address* dari pengirim pesan.
22.	status	t_pesan	Atribut yang menjelaskan status dari sebuah pesan.
23.	id_user	t_user	Merupakan *primary key* dari entitas t_user sebagai

No	Nama Atribut	Nama Entitas	Keterangan
			penanda *user* dan diinisialisasi secara inkremen.
24.	nipp	t_user	Atribut yang juga merupakan *primary key* pada entitas t_user yang menampilkan nomor induk kepegawaian.
25.	nama_lengkap	t_user	Atribut yang berisi nama lengkap dari seorang pegawai.
26.	jabatan	t_user	Atribut yang berisi jabatan dari seorang pegawai.
27.	unit_kerja	t_user	Atribut yang berisi unit_kerja dari seorang pegawai.
28.	kedudukan	t_user	Atribut yang berisi kedudukan dari seorang pegawai.
29.	nomor_telepon	t_user	Atribut yang berisi nomor telepon dari seorang pegawai.
30.	tipe_user	t_user	Atribut yang berisi tipe *user* dari seorang pegawai.
31.	password	t_user	Atribut yang berisi *password* dari seorang pegawai.
32.	fotoprofil	t_user	Atribut yang berisi *path* dari foto profil dari seorang pegawai.
33.	email	t_user	Atribut yang berisi alamat *email* dari seorang pegawai.
34.	id_gangguan	t_gangguan	Merupakan *primary key* dari entitas t_gangguan sebagai penanda gangguan dan diinisialisasi secara inkremen.

No	Nama Atribut	Nama Entitas	Keterangan
35.	kodegangguan	t_gangguan	Atribut yang berisi kodegangguan dari sebuah laporan gangguan.
36.	nipp_pelapor	t_gangguan	Atribut yang berisi nomor induk pegawai pelapor.
37.	jenis_gangguan	t_gangguan	Atribut yang berisi jenis gangguan dari sebuah laporan gangguan.
38.	deskripsi	t_gangguan	Atribut yang berisi deskripsi gangguan dari sebuah laporan gangguan.
39.	lokasi_gangguan	t_gangguan	Merupakan *foreign key* dari entitas t_lokasi yang berisi lokasi terjadinya sebuah laporan gangguan.
40.	waktu	t_gangguan	Atribut yang berisi kapan sebuah laporan gangguan masuk.
41.	status	t_gangguan	Atribut yang berisi status penanganan sebuah laporan gangguan.
42.	nipp_petugas	t_gangguan	Atribut yang berisi nomor induk pegawai petugas yang mengatasi sebuah laporan gangguan.
43.	solusi	t_gangguan	Atribut yang berisi solusi dari sebuah laporan gangguan.
44.	waktu_tutup	t_gangguan	Atribut yang berisi kapan sebuah laporan gangguan ditutup.
45.	id_barang	t_barang	Atribut yang berisi id dari suatu barang.
46.	nama_barang	t_barang	Atribut yang berisi nama dari suatu barang.
47.	qty	t_barang	Atribut yang berisi jumlah tersedia dari suatu barang.

No	Nama Atribut	Nama Entitas	Keterangan
48.	id_barang	t_gangguan_barang	Atribut yang merupakan *foreign key* dari entitas t_barang yang menyimpan id dari barang.
49.	kode_barang	t_gangguan_barang	Atribut yang berisi kode barang dari sebuah barang.
50.	id_gangguan	t_gangguan_barang	Atribut yang merupakan *foreign key* dari entitas t_gangguan yang menyimpan id gangguan.
51.	id	t_jenisgangguan	Atribut yang berisi id dari setiap nama jenis gangguan.
52.	nama_jenis_gangguan	t_jenisgangguan	Atribut yang berisi nama jenis gangguan.



Gambar 5.6 *Conceptual data model*

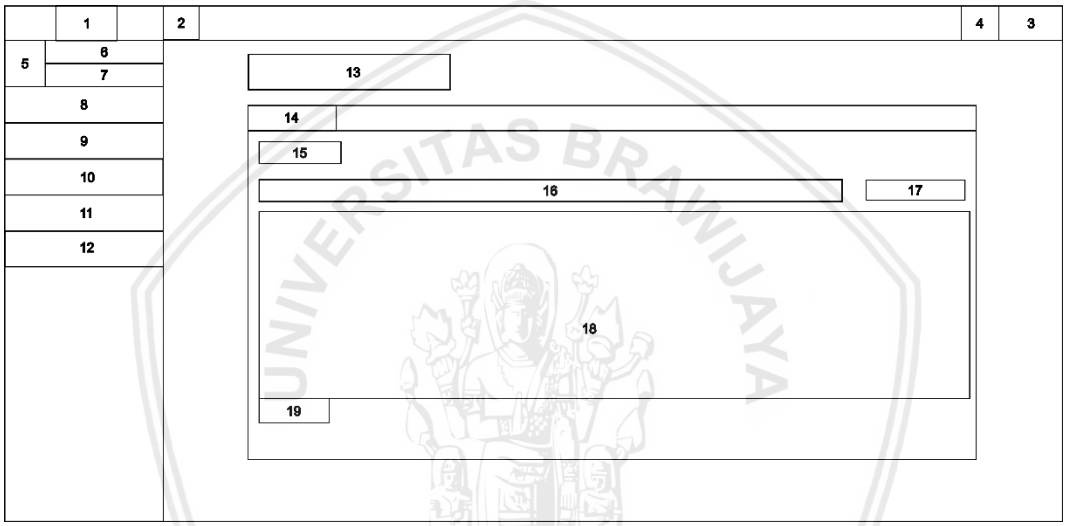
5.1.6 Perancangan Antarmuka

Sistem pelaporan gangguan ini menggunakan antarmuka berbasis *web* yang memiliki tampilan *responsive* mengikuti *device* yang mengakses. Namun disini penulis hanya menggambarkan tampilan saat diakses pada *desktop*. Pada penelitian ini, hanya 3 perancangan antarmuka yang penulis tampilkan yaitu perancangan antarmuka halaman jadwal kerja pada tampilan pimpinan unit IT,

perancangan antarmuka halaman buat laporan gangguan, dan perancangan antarmuka halaman detail laporan gangguan pada tampilan pelapor.

5.1.6.1 Perancangan Antarmuka Halaman Jadwal Kerja Pada Tampilan Pimpinan Unit IT

Perancangan antarmuka halaman jadwal kerja pada tampilan pimpinan unit IT dapat dilihat pada Gambar 5.7. Terdapat 19 komponen didalamnya mulai dari logo aplikasi, foto profil, nama pengguna yang sedang *login*, menu *sidebar* dan lain-lain. Penjelasan mengenai komponen-komponen yang ada pada perancangan antarmuka halaman jadwal kerja pada tampilan pimpinan unit IT dapat dilihat pada Tabel 5.8. Di tabel tersebut akan dijelaskan nama komponen, tipe komponen dan keterangan dari setiap komponen.



Gambar 5.7 Perancangan antarmuka halaman jadwal kerja pada tampilan pimpinan unit IT

Tabel 5.8 Penjelasan perancangan antarmuka halaman jadwal kerja pada tampilan pimpinan unit IT

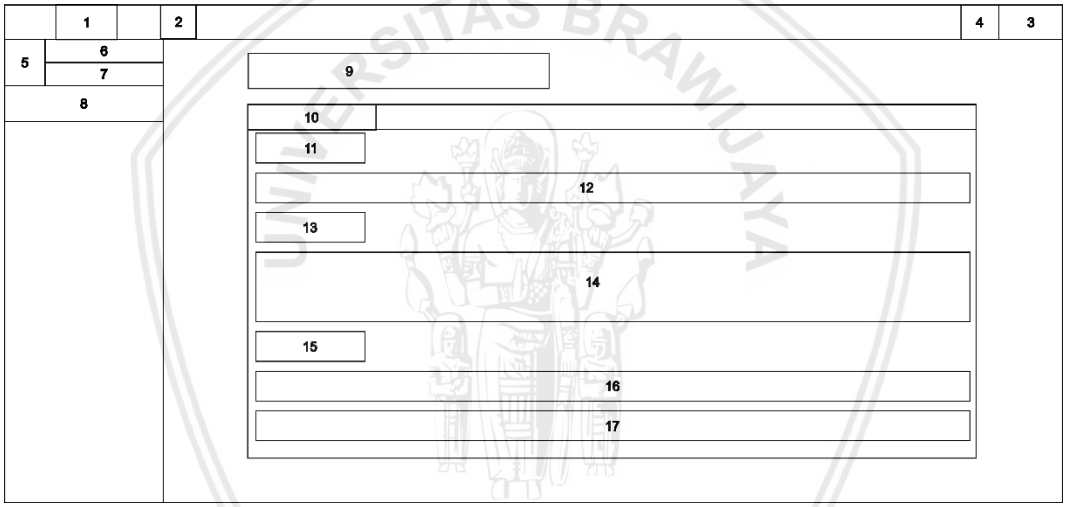
No	Nama Komponen	Tipe	Keterangan
1.	Logo Aplikasi	*Image*	Menampilkan logo aplikasi berupa logo PT. Kereta Api Indonesia.
2.	Tombol Menu	*Link*	Untuk menampilkan dan menyembunyikan menu *sidebar*.
3.	Nama Pengguna *Login*	*Span*	Untuk menampilkan nama lengkap dari pengguna yang sedang *login.*

No	Nama Komponen	Tipe	Keterangan
4.	Foto Profil	*Image*	Untuk menampilkan foto profil dari pengguna yang sedang *login.*
5.	Foto Profil *Sidebar*	*Image*	Untuk menampilkan foto profil dari pengguna yang sedang *login.*
6.	Nama Pengguna *Login Sidebar*	Paragraf	Untuk menampilkan nama lengkap dari pengguna yang sedang *login.*
7.	Unit Kerja dan Jabatan	*Heading* 6	Untuk menampilkan unit kerja dan jabatan dari pengguna yang sedang *login.*
8.	Menu Administrasi	*Link*	Untuk membuka *dropdown* menu administrasi.
9.	Menu Jadwal Kerja	*Link*	Untuk membuka halaman jadwal kerja.
10.	Menu Permohonan *Lintas*	*Link*	Untuk membuka halaman permohonan *lintas.*
11.	Menu Manajemen Gudang	*Link*	Untuk membuka halaman manajemen gudang penyimpanan.
12.	Menu Riwayat Laporan Gangguan	*Link*	Untuk menampilkan *dropdown* menu riwayat gangguan.
13.	Nama Halaman	*Heading* 2	Untuk menampilkan nama halaman.
14.	Nama Panel	*Heading* 3	Untuk menampilkan nama panel.
15.	Tombol Tambah Jadwal	*Link*	Untuk menjalankan fungsi tambah jadwal kerja.
16.	*Input* Bulan Tahun	*Input*	Untuk memasukkan nama bulan dan tahun yang ingin dilihat jadwalnya.
17.	Tombol Pilih	*Button*	Untuk menjalankan fungsi menampilkan jadwal sesuai dengan *input* bulan.

No	Nama Komponen	Tipe	Keterangan
18.	Tabel Jadwal	Tabel	Untuk menampilkan jadwal kerja dari pegawai unit IT.
19.	Tombol *Export to Excel*	*Button*	Untuk mengekspor jadwal kerja dalam bentuk *Excel*.

5.1.6.2 Perancangan Antarmuka Halaman Buat Laporan Gangguan

Perancangan antarmuka halaman buat laporan gangguan dapat dilihat pada Gambar 5.8. Terdapat 17 komponen didalamnya mulai dari logo aplikasi, foto profil, nama pengguna yang sedang *login*, menu *sidebar* dan lain-lain. Penjelasan mengenai komponen-komponen yang ada pada perancangan antarmuka halaman buat laporan gangguan dapat dilihat pada Tabel 5.9. Di tabel tersebut akan dijelaskan nama komponen, tipe komponen dan keterangan dari setiap komponen.



Gambar 5.8 Perancangan antarmuka halaman buat laporan gangguan

Tabel 5.9 Penjelasan perancangan antarmuka halaman buat laporan gangguan

No	Nama Komponen	Tipe	Keterangan
1.	Logo Aplikasi	*Image*	Menampilkan logo aplikasi berupa logo PT. Kereta Api Indonesia.
2.	Tombol Menu	*Link*	Untuk menampilkan dan menyembunyikan menu *sidebar.*
3.	Nama Pengguna *Login*	*Span*	Untuk menampilkan nama lengkap dari pengguna yang sedang *login.*

No	Nama Komponen	Tipe	Keterangan
4.	Foto Profil	*Image*	Untuk menampilkan foto profil dari pengguna yang sedang *login.*
5.	Foto Profil *Sidebar*	*Image*	Untuk menampilkan foto profil dari pengguna yang sedang *login.*
6.	Nama Pengguna *Login Sidebar*	Paragraf	Untuk menampilkan nama lengkap dari pengguna yang sedang *login.*
7.	Unit Kerja dan Jabatan	*Heading* 6	Untuk menampilkan unit kerja dan jabatan dari pengguna yang sedang *login.*
8.	Menu Pelaporan	*Link*	Untuk membuka *dropdown* menu pelaporan.
9.	Nama Halaman	*Heading* 2	Untuk menampilkan nama halaman.
10.	Nama Panel	*Heading* 3	Untuk menampilkan nama panel.
11.	Legenda *Input* Tipe Gangguan	*Legend*	Untuk menampilkan legenda untuk *input* tipe gangguan.
12.	*Input* Tipe Gangguan	*Input – Select*	Untuk memasukkan tipe gangguan.
13.	Legenda *Input* Deskripsi Gangguan	*Legend*	Untuk menampilkan legenda untuk *input* deskripsi gangguan.
14.	*Input* Deskripsi Gangguan	*Input – Textarea*	Untuk memasukkan deskripsi gangguan.
15.	Legenda *Input* Lokasi Gangguan	*Legend*	Untuk menampilkan legenda untuk *input* lokasi gangguan.
16.	*Input* Lokasi Gangguan	*Input – Select*	Untuk memasukkan lokasi gangguan.
17.	Tombol Laporkan Gangguan	*Button*	Untuk menjalankan fungsi yang digunakan untuk menyimpan data laporan gangguan.

5.1.6.3 Perancangan Antarmuka Halaman *Detail* Laporan Gangguan Pada Tampilan Pelapor

Perancangan antarmuka halaman *detail* laporan gangguan pada tampilan pelapor dapat dilihat pada Gambar 5.9. Terdapat 18 komponen didalamnya mulai dari logo aplikasi, foto profil, nama pengguna yang sedang *login*, menu *sidebar* dan lain-lain. Penjelasan mengenai komponen-komponen yang ada pada perancangan antarmuka halaman *detail* laporan gangguan pada tampilan pelapor dapat dilihat pada Tabel 5.10. Di tabel tersebut akan dijelaskan nama komponen, tipe komponen dan keterangan dari setiap komponen.

Gambar 5.9 Perancangan antarmuka halaman *detail* laporan gangguan pada tampilan pelapor

Tabel 5.10 Penjelasan perancangan antarmuka halaman *detail* laporan gangguan pada tampilan pelapor

No	Nama Komponen	Tipe	Keterangan
1.	Logo Aplikasi	*Image*	Menampilkan logo aplikasi berupa logo PT. Kereta Api Indonesia.
2.	Tombol Menu	*Link*	Untuk menampilkan dan menyembunyikan menu *sidebar.*
3.	Nama Pengguna *Login*	*Span*	Untuk menampilkan nama lengkap dari pengguna yang sedang *login.*
4.	Foto Profil	*Image*	Untuk menampilkan foto profil dari pengguna yang sedang *login.*
5.	Foto Profil *Sidebar*	*Image*	Untuk menampilkan foto profil dari pengguna yang sedang *login.*
6.	Nama Pengguna *Login Sidebar*	Paragraf	Untuk menampilkan nama lengkap dari pengguna yang sedang *login.*
7.	Unit Kerja dan Jabatan	*Heading* 6	Untuk menampilkan unit kerja dan jabatan dari pengguna yang sedang *login.*
8.	Menu Pelaporan	*Link*	Untuk membuka *dropdown* menu pelaporan.
9.	Nama Halaman	*Heading* 2	Untuk menampilkan nama halaman.
10.	Nama Panel *Detail* Laporan Gangguan	*Heading* 3	Untuk menampilkan nama panel *detail* laporan gangguan.
11.	Tabel *Detail* Laporan Gangguan	Tabel	Untuk menampilkan *detail* dari sebuah laporan gangguan.
12.	Nama Panel Pesan Gangguan	*Heading* 3	Untuk menampilkan nama panel pesan gangguan.
13.	Layar Pesan	*Div*	Untuk menampilkan pesan antara petugas dan pelapor.

No	Nama Komponen	Tipe	Keterangan
14.	Isi Pesan	*Input – Text*	Untuk menuliskan isi pesan yang ingin dikirm.
15.	Tombol Kirim	*Button*	Untuk menjalankan fungsi untuk menyimpan data pesan ke dalam *database.*
16.	Nama Panel Pendataan Barang *Lintas*	*Heading* 3	Untuk menampilkan nama panel pendataan barang *lintas.*
17.	Nama Tabel Barang Terdaftar	*Heading* 4	Untuk menampilkan nama tabel barang terdaftar.
18.	Tabel Barang Terdaftar	Tabel	Untuk menampilkan data barang yang telah terdaftar untuk sebuah penanganan gangguan.

5.2 Implementasi Sistem

5.2.1 Spesifikasi Sistem

Berikut ini adalah spesifikasi sistem yang penulis gunakan untuk melakukan perancangan dan pembangunan sistem ini meliputi perangkat lunak dan perangkat keras.

5.2.1.1 Spesifikasi *Hardware*

Spesifikasi perangkat yang digunakan untuk mengembangkan sistem pelaporan gangguan dapat dilihat pada Tabel 5.11.

Tabel 5.11 Spesifikasi *hardware*

Nama Komponen	Spesifikasi
System Model	ASUS X442UQR
Processor	Intel® Core™ i5-8250 CPU @ 1.60GHz (8 CPUs), ~ 1,8 GHz
Memory	8192MB RAM
Display	NVIDIA GeForce 940MX
Hard Disk	1 TB
Resolusi Layar	1920 x 1080 (32 bit) (60Hz)

5.2.1.2 Spesifikasi *Software*

Spesifikasi perangkat lunak yang digunakan untuk mengembangkan sistem pelaporan gangguan dapat dilihat pada Tabel 5.12.

Tabel 5.12 Spesifikasi *software*

Nama Komponen	Spesifikasi
Sistem Operasi	Windows 10 Home Single Language 64-bit
Editor Perancangan	Visual Paradigm 14.2
Editor Pemrograman	Visual Studio Code Versi 1.29.1
Bahasa Pemrograman	PHP, Javascrip, CSS, dan HTML
DBMS	MySQL
Aplikasi Pendukung Sistem	XAMPP v3.2.2 WhatsApp Messenger Microsoft Word 2016 CorelDraw X8
Peramban	Google Chrome versi 70.0.3538.110 Microsoft Edge 44.17763.1.0 Mozilla Firefox 64.0.2

5.2.2 Implementasi Komponen

5.2.2.1 Implementasi Komponen buatAkunImport()

Berikut ini merupakan hasil implementasi komponen untuk fungsi buatAkunImport() yang dapat dilihat pada Tabel 5.13. Dalam implementasi komponen ini dilakukan proses pengambilan data dari *file excel* yang nantinya akan disimpan di dalam *database*.

Nama Kelas : C_Pimpinan

Nama Fungsi : buatAkunImport()

Tabel 5.13 Implementasi komponen fungsi buatAkunImport()

Implementasi komponen fungsi buatAkunImport()

`public function buatAkunImport(){`
`$path = $_FILES["format_tambah"]["tmp_name"];`
`$object = PHPExcel_IOFactory::load($path);`
`$data = array();`
`$nipp_terdaftar = array();`
`foreach ($object->getWorksheetIterator() as $worksheet) {`

Implementasi komponen fungsi buatAkunImport()

```
            $highestRow = $worksheet->getHighestRow();
            for ($row = 2; $row <= $highestRow; $row++) {
                $nipp    =    $worksheet->getCellByColumnAndRow(0,
$row)->getValue();
                $nama_lengkap            =            $worksheet-
>getCellByColumnAndRow(1, $row)->getValue();
                $unit_kerja              =            $worksheet-
>getCellByColumnAndRow(2, $row)->getValue();
                $jabatan  =  $worksheet->getCellByColumnAndRow(3,
$row)->getValue();
                $kedudukan               =            $worksheet-
>getCellByColumnAndRow(4, $row)->getValue();
                $nomor_telepon           =            $worksheet-
>getCellByColumnAndRow(5, $row)->getValue();
                $tipe_user               =            $worksheet-
>getCellByColumnAndRow(6, $row)->getValue();
                $email   =   $worksheet->getCellByColumnAndRow(7,
$row)->getValue();
                $fotoprofil                                      =
'assets/img/fotoprofil/defaultprofile.jpg';
                $password                                        =
'$2y$10$KKxo3PjBIqeBykuVAeqmM.DRK5xyPVGXVUjZVh6amZUpO9PRbymci';
                $count = $this->cekNippExcel($nipp);
                if ($nipp != null) {
                    if($count==0){
                        $data[] = array(
                            'nipp' => $nipp,
                            'nama_lengkap' => $nama_lengkap,
                            'unit_kerja' => $unit_kerja,
                            'jabatan' => $jabatan,
                            'kedudukan' => $kedudukan,
                            'nomor_telepon' => $nomor_telepon,
                            'tipe_user'=> $tipe_user,
                            'email' => $email,
                            'fotoprofil' => $fotoprofil,
                            'password' => $password
                        );
                    }else{
                        array_push($nipp_terdaftar,$nipp);
                    }
                }
```

Implementasi komponen fungsi buatAkunImport()

```
                else{
                    continue;
                }
            }
        }
        if($nipp_terdaftar != null){
            ?> <script language="JavaScript">alert("Impor akun
gagal dilakukan, NIPP <?php echo join(', ', $nipp_terdaftar); ?>
sudah terdaftar dalam sistem.");
            document.location='<?php                          echo
site_url('C_Pimpinan/viewBuatUserImport');?>'</script><?php
        }else{
            if($data!=null){
                $this->M_User->buatAkunImport($data, "t_user");
                ?>  <script  language="JavaScript">alert('Akun
Berhasil Diimpor.');
                document.location='<?php                          echo
site_url('C_Pimpinan/viewBuatUserImport');?>'</script><?php
            }else{
            ?> <script language="JavaScript">alert('Tidak ada
data di dalam file excel.');
            document.location='<?php                          echo
site_url('C_Pimpinan/viewBuatUserImport');?>'</script><?php
            }
        }
    }
```

5.2.2.2 Implementasi Komponen buatLaporanGangguan()

Berikut ini merupakan hasil implementasi komponen untuk fungsi buatLaporanGangguan() yang dapat dilihat pada Tabel 5.14. Dalam komponen ini dilakukan proses pengambilan *input*an dari pelapor yang nantinya akan disimpan dalam *database*.

Nama Kelas : C_Pelapor

Nama Fungsi : buatLaporanGangguan()

Tabel 5.14 Implementasi komponen fungsi buatLaporanGangguan()

Implementasi komponen fungsi buatLaporanGangguan()

```
public function buatLaporanGangguan(){
        setlocale(LC_ALL, 'IND');
        $nipp_pelapor = $this->session->userdata('nipp');
        $waktu = date('Y-m-d H:i:s');
```

Implementasi komponen fungsi buatLaporanGangguan()

```
        $kodegangguan = random_string('alnum',6);
        $result               =               $this->M_Gangguan-
>cekKodeGangguan($kodegangguan);

        while($result != null){
            $kodegangguan = random_string('alnum',6);
            $result                              =$this->M_Gangguan-
>cekKodeGangguan($kodegangguan);
        }

        $jenis_gangguan             =             $this->input-
>post('jenis_gangguan',true);
        $deskripsi = $this->input->post('deskripsi',true);
        $lokasi_gangguan            =             $this->input-
>post('lokasi_gangguan',true);
        $status = "Belum Ditangani";
        $laporan = array(
            "nipp_pelapor"=> $nipp_pelapor,
            "kodegangguan" => $kodegangguan,
            "jenis_gangguan" => $jenis_gangguan,
            "deskripsi" => $deskripsi,
            "lokasi_gangguan"=> $lokasi_gangguan,
            "waktu" => $waktu,
            "status" => $status
         );

        $lokasi                =               $this->M_Lokasi-
>getLokasiPesan($lokasi_gangguan)->lokasi;
        $jenisgangguan         =         $this->M_JenisGangguan-
>getJenisGangguanPesan($jenis_gangguan)->nama_jenis_gangguan;
        $message = "Ada $jenisgangguan yang terjadi di $lokasi
dengan NIPP pelapor : $nipp_pelapor pada $waktu";
        $this->M_Gangguan-
>buatLaporanGangguan('t_gangguan',$laporan);
        $url = site_url('C_Petugas/viewLihatLaporanMasuk');

        $petugasAktif = $this->M_JadwalKerja->getPetugasAktif();
        $headings ="Laporan Gangguan Masuk";
        $img = base_url()."assets/img/logo_pdf.png";
```

Implementasi komponen fungsi buatLaporanGangguan()

```
        $content = array(
            "en" => "$message"
        );
        $headings = array(
            "en" => "$headings"
        );

        foreach($petugasAktif as $p){
            $id_user = $p['id_user'];
            $fields = array(
                'app_id'      =>      "a0c33864-c057-45c6-866e-
85930ee9e794",
                'filters' => array(array("field" => "tag", "key"
=> "user_id", "relation" => "=", "value" => "$id_user")),
                'url' => $url,
                'contents' => $content,
                'chrome_web_icon' => $img,
                'headings' => $headings
            );
            $fields = json_encode($fields);
            $ch = curl_init();
            curl_setopt($ch,                          CURLOPT_URL,
"https://onesignal.com/api/v1/notifications");
            curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-
Type: application/json; charset=utf-8',
                'Authorization:                              Basic
NDM0NDQ2NzktMWEzOC00YzNiLTg1ZmYtM2UxYjI1NTc2YTdi'));
            curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
            curl_setopt($ch, CURLOPT_HEADER, FALSE);
            curl_setopt($ch, CURLOPT_POST, TRUE);
            curl_setopt($ch, CURLOPT_POSTFIELDS, $fields);
            curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
            $response = curl_exec($ch);
            curl_close($ch);

            $my_apikey = "A7H2QHUS4IQOE9C0SKZJ";
            $notelp = $p['nomor_telepon'];
            $nol = "/^0/";
            $rpltxt = "62";
```

Implementasi komponen fungsi buatLaporanGangguan()

```
            $destination = preg_replace($nol, $rpltxt, $notelp);
            $api_url                                                    =
"http://panel.apiwha.com/send_message.php";
            $api_url .= "?apikey=". urlencode ($my_apikey);
            $api_url .= "&number=". urlencode ($destination);
            $api_url .= "&text=". urlencode ($message);
            $my_result_object                                           =
json_decode(file_get_contents($api_url, false));
        }


        $data['laporan']          =          $this->M_Gangguan-
>getDetailPelaporan($kodegangguan);
        $this->load->view('Auth/pelapor/laporanSukses',$data);
    }
```

5.2.2.3 Implementasi Komponen cekNipp()

Berikut ini merupakan hasil implementasi komponen untuk fungsi cekNipp() yang dapat dilihat pada Tabel 5.15. Dalam komponen ini dilakukan proses pengecekan antara NIPP yang di*input*kan oleh Pimpinan Unit IT dengan NIPP yang ada pada *database*.

Nama Kelas : C_Pimpinan

Nama Fungsi : cekNipp()

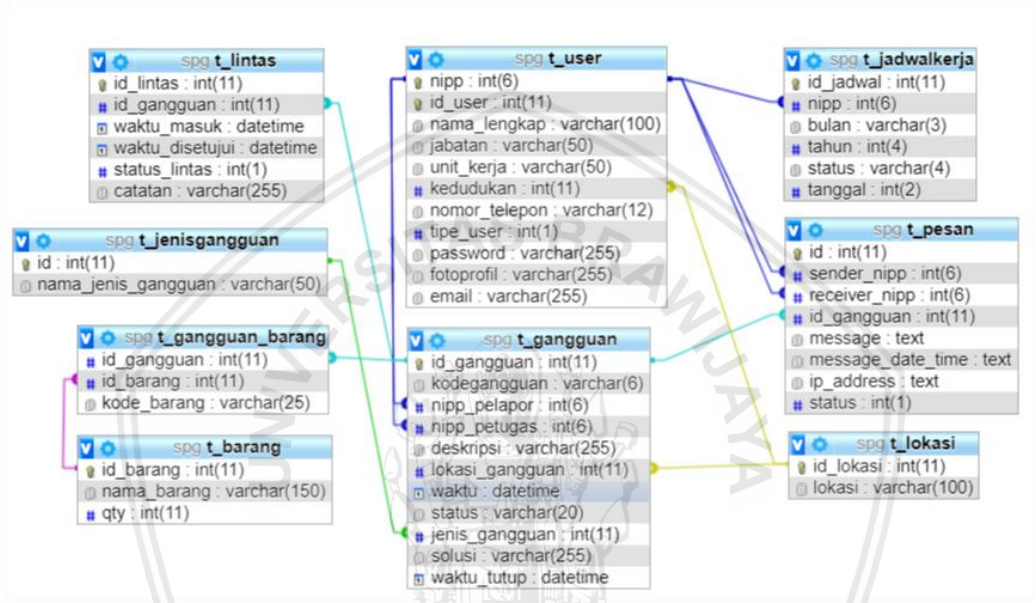
Tabel 5.15 Implementasi komponen fungsi cekNipp()

Implementasi komponen fungsi cekNipp()

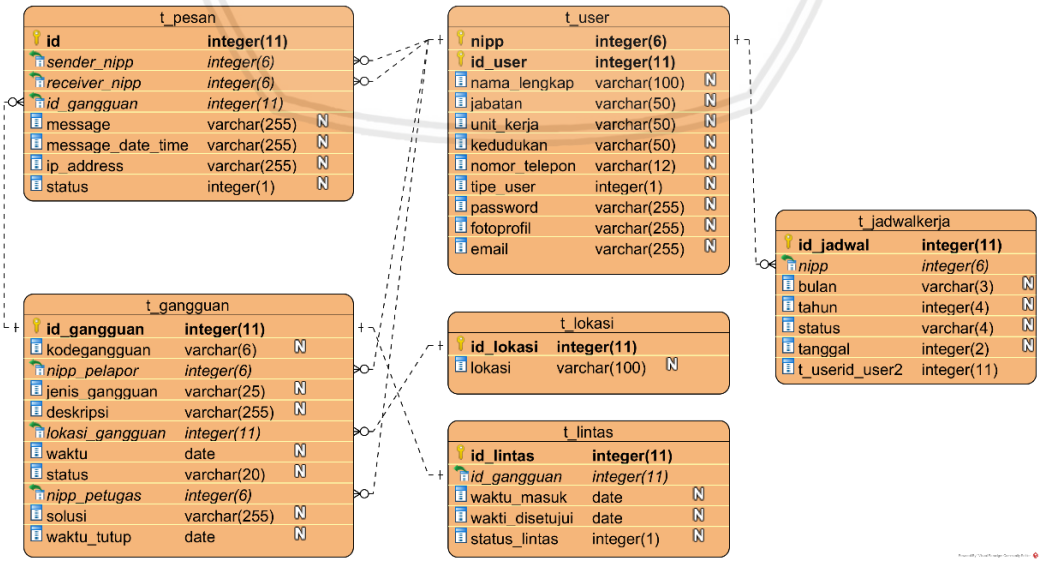
```
    public function cekNipp(){
        $nipp = $this->input->post('nipp');
        $exists = $this->M_User->cekNipp($nipp);
        $count = count($exists);
        $data['state'] = 0;
        if ($count > 0) {
            $data['state'] = 1;
        } else {
            $data['state'] = 0;
        }
        print(json_encode($data));
    }
```

5.2.3 Implementasi *Database*

Sistem pelaporan gangguan dibuat dengan tujuan untuk meningkatkan efisiensi dan efektivitas pada proses pelaporan gangguan pada unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon. Gambar 5.10 menggambarkan skema *database* sistem pelaporan gangguan. Dalam skema *database* tersebut terdapat 9 tabel yaitu t_lintas, t_user, t_gangguan, t_lokasi, t_jadwalkerja, t_barang, t_gangguan_barang, dan t_jenisgangguan dan t_pesan. Penulis juga menampilkan *Physical Data Model* yang dapat dilihat pada Gambar 5.11. Penjelasan atribut-atribut pada hasil implementasi *database* dijelaskan pada Tabel 5.16.



Gambar 5.10 Skema *database* sistem pelaporan gangguan



Gambar 5.11 *Physical data model* sistem pelaporan gangguan

Tabel 5.16 Penjelasan atribut hasil implementasi *database*

No	Nama Atribut	Nama Entitas	Keterangan
1.	id_jadwal	t_jadwalkerja	*Primary key* dari entitas t_jadwalkerja, yang diinisialisasi secara inkremen.
2.	nipp	t_jadwalkerja	Merupakan atribut yang menandakan pegawai yang memiliki jadwal tersebut.
3.	bulan	t_jadwalkerja	Merupakan atribut yang menandakan pada bulan apa jadwal tersebut.
4.	tahun	t_jadwalkerja	Merupakan atribut yang menandakan pada tahun apa jadwal tersebut.
5.	status	t_jadwalkerja	Merupakan atribut yang menandakan status kerja pada jadwal tersebut.
6.	tanggal	t_jadwalkerja	Merupakan atribut yang menandakan pada tanggal berapa jadwal tersebut.
7.	id_lokasi	t_lokasi	Merupakan *primary key* dari entitas t_lokasi sebagai penanda lokasi dan diinisialisasi secara inkremen.
8.	lokasi	t_lokasi	Merupakan atribut yang menjelaskan nama dari suatu lokasi.
9.	id_lintas	t_lintas	Merupakan *primary key* dari entitas t_lintas sebagai penanda *lintas* dan diinisialisasi secara inkremen.
10.	id_gangguan	t_lintas	Merupakan *foreign key* dari entitas t_gangguan yang menyatakan gangguan mana yang

No	Nama Atribut	Nama Entitas	Keterangan
			terhubung dengan *lintas* tersebut.
11.	waktu_masuk	t_lintas	Merupakan atribut yang menjelaskan kapan pengajuan *lintas* masuk.
12.	waktu_disetujui	t_lintas	Merupakan atribut yang menjelaskan kapan pengajuan *lintas* disetujui.
13.	status_lintas	t_lintas	Merupakan atribut yang menjelaskan status dari sebuah pengajuan *lintas.*
14.	catatan	t_lintas	Merupakan atribut yang menjelaskan catatan dari pimpinan unit IT untuk sebuah *lintas.*
15.	id	t_pesan	Merupakan *primary key* dari entitas t_pesan sebagai penanda pesan dan diinisialisasi secara inkremen.
16.	sender_nipp	t_pesan	Merupakan atribut yang merupakan *foreign key* dari entitas t_user yang menyatakan nipp dari pengirim pesan.
17.	receiver_nipp	t_pesan	Merupakan atribut yang merupakan *foreign key* dari entitas t_user yang menyatakan nipp dari penerima pesan..
18.	id_gangguan	t_pesan	Merupakan *foreign key* dari entitas t_gangguan yang menyatakan gangguan mana yang terhubung dengan pesan tersebut.
19.	message	t_pesan	Atribut yang menjelaskan isi dari suatu pesan.

No	Nama Atribut	Nama Entitas	Keterangan
20.	message_date_time	t_pesan	Atribut yang menjelaskan kapan suatu pesan dikirim.
21.	ip_address	t_pesan	Atribut yang menjelaskan *ip address* dari pengirim pesan.
22.	status	t_pesan	Atribut yang menjelaskan status dari sebuah pesan.
23.	id_user	t_user	Merupakan *primary key* dari entitas t_user sebagai penanda *user* dan diinisialisasi secara inkremen.
24.	nipp	t_user	Atribut yang juga merupakan *primary key* pada entitas t_user yang menampilkan nomor induk kepegawaian.
25.	nama_lengkap	t_user	Atribut yang berisi nama lengkap dari seorang pegawai.
26.	jabatan	t_user	Atribut yang berisi jabatan dari seorang pegawai.
27.	unit_kerja	t_user	Atribut yang berisi unit_kerja dari seorang pegawai.
28.	kedudukan	t_user	Atribut yang berisi kedudukan dari seorang pegawai.
29.	nomor_telepon	t_user	Atribut yang berisi nomor telepon dari seorang pegawai.
30.	tipe_user	t_user	Atribut yang berisi tipe *user* dari seorang pegawai.
31.	password	t_user	Atribut yang berisi *password* dari seorang pegawai.

No	Nama Atribut	Nama Entitas	Keterangan
32.	fotoprofil	t_user	Atribut yang berisi *path* dari foto profil dari seorang pegawai.
33.	email	t_user	Atribut yang berisi alamat *email* dari seorang pegawai.
34.	id_gangguan	t_gangguan	Merupakan *primary key* dari entitas t_gangguan sebagai penanda gangguan dan diinisialisasi secara inkremen.
35.	kodegangguan	t_gangguan	Atribut yang berisi kodegangguan dari sebuah laporan gangguan.
36.	nipp_pelapor	t_gangguan	Atribut yang berisi nomor induk pegawai pelapor.
37.	jenis_gangguan	t_gangguan	Atribut yang berisi jenis gangguan dari sebuah laporan gangguan.
38.	deskripsi	t_gangguan	Atribut yang berisi deskripsi gangguan dari sebuah laporan gangguan.
39.	lokasi_gangguan	t_gangguan	Merupakan *foreign key* dari entitas t_lokasi yang berisi lokasi terjadinya sebuah laporan gangguan.
40.	waktu	t_gangguan	Atribut yang berisi kapan sebuah laporan gangguan masuk.
41.	status	t_gangguan	Atribut yang berisi status penanganan sebuah laporan gangguan.
42.	nipp_petugas	t_gangguan	Atribut yang berisi nomor induk pegawai petugas yang mengatasi sebuah laporan gangguan.
43.	solusi	t_gangguan	Atribut yang berisi solusi dari sebuah laporan gangguan.

No	Nama Atribut	Nama Entitas	Keterangan
44.	waktu_tutup	t_gangguan	Atribut yang berisi kapan sebuah laporan gangguan ditutup.
45.	id_barang	t_barang	Atribut yang berisi *id* dari suatu barang.
46.	nama_barang	t_barang	Atribut yang berisi nama dari suatu barang.
47.	qty	t_barang	Atribut yang berisi jumlah tersedia dari suatu barang.
48.	id_barang	t_gangguan_barang	Atribut yang merupakan *foreign key* dari entitas t_barang yang menyimpan *id* dari barang.
49.	kode_barang	t_gangguan_barang	Atribut yang berisi kode barang dari sebuah barang.
50.	id_gangguan	t_gangguan_barang	Atribut yang merupakan *foreign key* dari entitas t_gangguan yang menyimpan *id* gangguan.
51.	id	t_jenisgangguan	Atribut yang berisi *id* dari setiap nama jenis gangguan.
52.	nama_jenis_gangguan	t_jenisgangguan	Atribut yang berisi nama jenis gangguan.

5.2.4 Implementasi Antarmuka

Implementasi antarmuka dibuat berdasarkan perancangan antarmuka yang telah dibuat sebelumnya. Pada Gambar 5.12 menggambarkan hasil implementasi antarmuka halaman jadwal kerja pada tampilan pimpinan unit IT, Gambar 5.13 menggambarkan hasil implementasi antarmuka halaman buat laporan gangguan, dan Gambar 5.14 menggambarkan hasil implementasi antarmuka halaman *detail* laporan gangguan pada tampilan pelapor.

5.2.4.1 Implementasi Antarmuka Halaman Jadwal Kerja Pada Tampilan Pimpinan Unit IT

Pada implementasi antarmuka halaman jadwal kerja pada tampilan pimpinan unit IT terdapat logo aplikasi, tombol menu, nama pengguna *login*, foto profil, foto profil *sidebar*, nama pengguna *login sidebar*, unit kerja dan jabatan, menu administrasi, menu jadwal kerja, menu permohonan *lintas*, menu

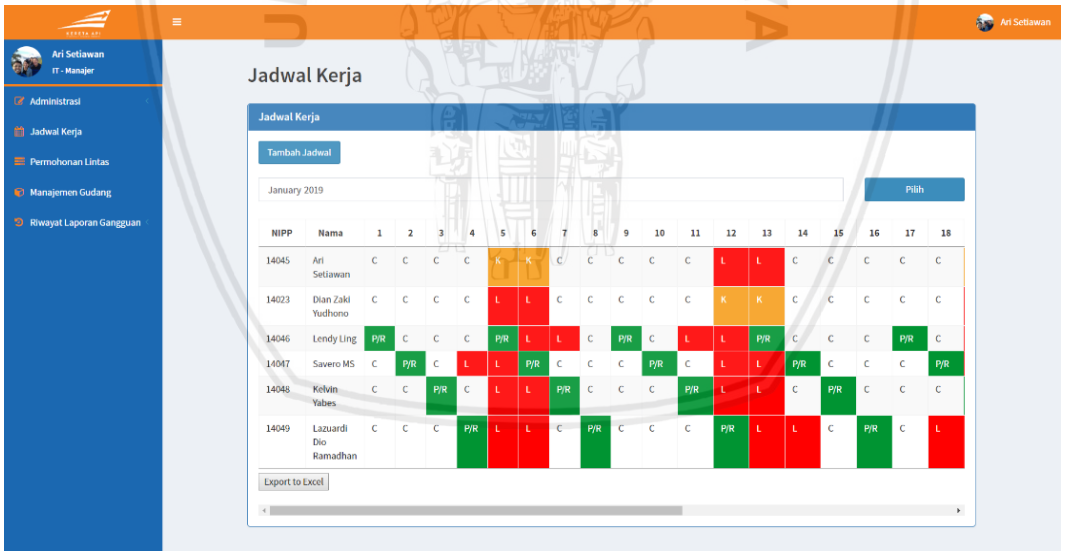
manajemen gudang, menu riwayat laporan gangguan, tombol *Export to Excel*, nama halaman, nama panel jadwal kerja, tombol tambah jadwal, *input* bulan dan tahun, tombol pilih, dan tabel jadwal.

5.2.4.2 Implementasi Antarmuka Halaman Buat Laporan Gangguan

Pada implementasi antarmuka halaman buat laporan gangguan terdapat logo aplikasi, tombol menu, nama pengguna *login*, foto profil, foto profil *sidebar*, nama pengguna *login sidebar*, unit kerja dan jabatan, menu pelaporan, nama halaman, nama panel, legenda *input* tipe gangguan, *input* tipe gangguan, legenda *input* deskripsi gangguan, *input* deskripsi gangguan, legenda *input* lokasi gangguan, *input* lokasi gangguan dan tombol laporkan gangguan.

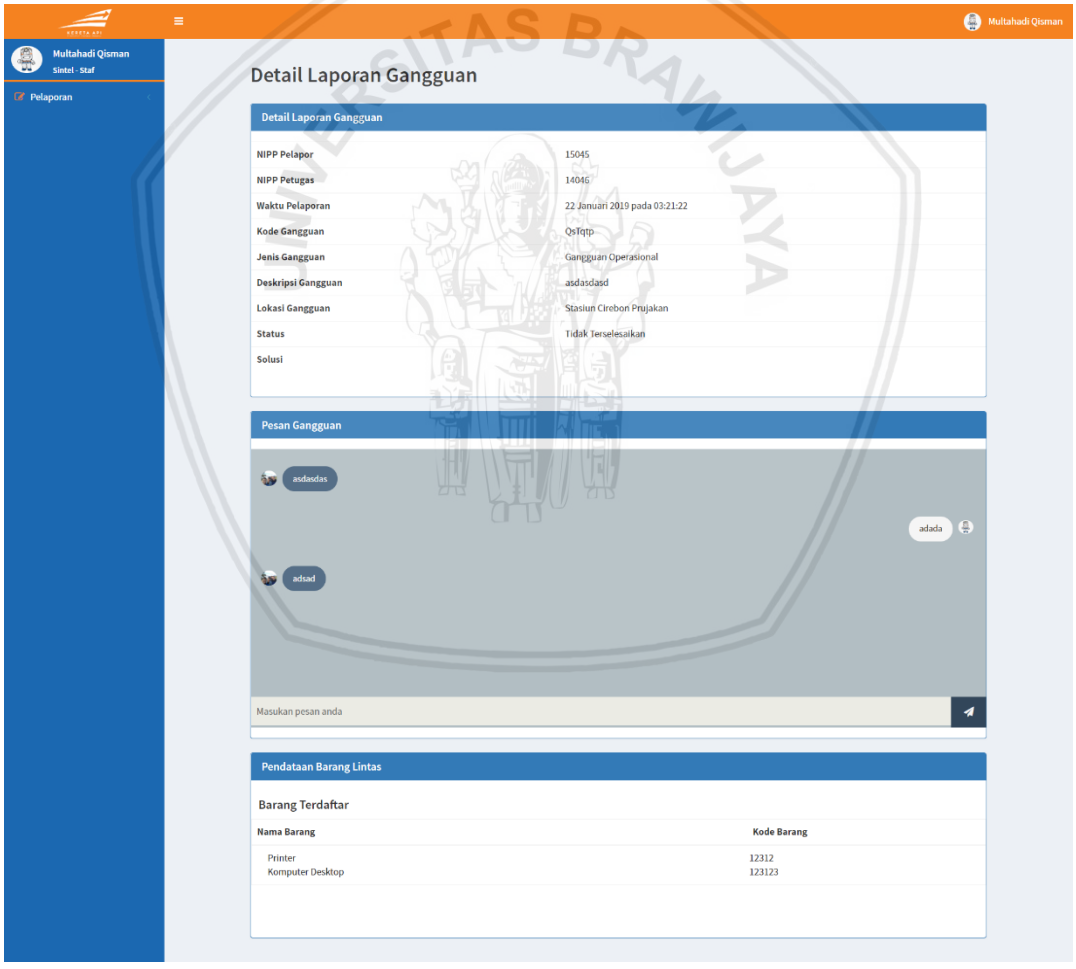
5.2.4.3 Implementasi Antarmuka Halaman *Detail* Laporan Gangguan Pada Tampilan Pelapor

Pada implementasi antarmuka halaman *detail* laporan gangguan pada tampilan pelapor terdapat logo aplikasi, tombol menu, nama pengguna *login*, foto profil, foto profil *sidebar*, nama pengguna *login sidebar*, unit kerja dan jabatan, menu pelaporan, nama halaman, nama panel *detail* laporan gangguan, tabel *detail* laporan gangguan, nama panel pesan gangguan, layar pesan isi pesan tombol kirim, nama panel pendataan barang *lintas*, nama tabel barang terdaftar dan tabel barang terdaftar.



Gambar 5.12 Implementasi antarmuka halaman jadwal kerja pada tampilan pimpinan unit it

Gambar 5.13 Implementasi antarmuka halaman buat laporan gangguan



Gambar 5.14 Implementasi antarmuka halaman *detail* laporan gangguan pada tampilan pelapor

BAB 6 PENGUJIAN DAN ANALISIS

Pengujian merupakan tahap yang dilakukan setelah tahap implementasi selesai. Tahap pengujian ini memiliki tujuan untuk memastikan kesesuaian antara hasil implementasi dengan kebutuhan pengguna dan hasil perancangan sistem yang sudah dibuat sebelumnya. Pada penelitian ini akan dilakukan *verification testing* dan *validation testing*. Untuk *verification testing* akan dilakukan pengujian unit dan pengujian integrasi dengan menggunakan *white-box testing* jenis *basis path testing*. Pengujian integrasi akan menggunakan pendekatan *top-down*. Validation *testing* terdiri dari *validation testing* kebutuhan fungsional dan *validation testing* kebutuhan non-fungsional yang akan dilakukan dengan menggunakan *black-box testing*. C*ompatibility testing* yang merupakan bagian dari *validation testing* kebutuhan non-fungsional akan dilakukan menggunakan kakas bantu *SortSite* 5.

6.1 Pengujian Unit

Pengujian unit atau *unit testing* merupakan pengujian yang digunakan untuk menguji unit atau *individual components* untuk menyakini bahwa unit atau komponen tersebut beroperasi secara benar. Setiap komponen akan diujikan secara terpisah tanpa komponen sistem yang lainnya.

6.1.1 Pengujian Unit Komponen Fungsi buatAkunImport()

Pada Tabel 6.1 dapat dilihat *pseudocode* dari fungsi buatAkunImport() dan pada Gambar 6.1 dapat dilihat *flowgraph* dari fungsi buatAkunImport(). Pada Tabel 6.2 dapat dilihat kasus uji dari fungsi buatAkunImport().

1. *Pseudocode* algoritme fungsi buatAkunImport()

Nama kelas : C_Pimpinan

Nama fungsi : buatAkunImport()

Tabel 6.1 *Pseudocode* fungsi buatAkunImport()

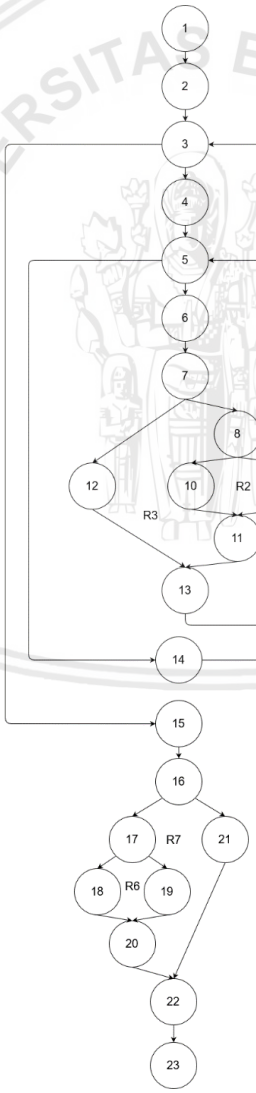
Pseudocode fungsi buatAkunImport()	
1	Mulai
2	Inisialisasi variabel path = path dari file excel Inisialisasi variabel object = hasil pemanggilan fungsi load(path) Inisialisasi variabel data dengan tipe array Inisialisasi nipp_terdaftar dengan tipe array
3	FOREACH setiap workshet yang ada di file excel
4	Inisialisasi highestRow = hasil pemanggilan fungsi getHighestRow()
5	FOR nilai awal row = 2 hingga row <= highestRow, row++

Pseudocode fungsi buatAkunImport()	
6	Inisialisasi variabel nipp = hasil dari pemanggilan fungsi getCellByColumnAndRow(0, row)->getValue() Inisialisasi variabel nama_lengkap = hasil dari pemanggilan fungsi getCellByColumnAndRow(1, row)->getValue() Inisialisasi variabel unit_kerja = hasil dari pemanggilan fungsi getCellByColumnAndRow(2, row)->getValue() Inisialisasi variabel jabatan = hasil dari pemanggilan fungsi getCellByColumnAndRow(3, row)->getValue() Inisialisasi variabel kedudukan = hasil dari pemanggilan fungsi getCellByColumnAndRow(4, row)->getValue() Inisialisasi variabel nomor_telepon = hasil dari pemanggilan fungsi getCellByColumnAndRow(5, row)->getValue() Inisialisasi variabel tipe_user = hasil dari pemanggilan fungsi getCellByColumnAndRow(6, row)->getValue() Inisialisasi variabel email = hasil dari pemanggilan fungsi getCellByColumnAndRow(7, row)->getValue() Inisialisasi variabel fotoprofil = assets/img/fotoprofil/defaultprofile.jpg Inisialisasi variabel password = $2y$10$KKxo3PjBIqeBykuVAeqmM.DRK5xyPVGXVUjZVh6amZUpO9Prbymci Inisialisasi variabel count dengan nilai sama dengan nilai kembalian dari pemanggilan fungsi cekNippExcel(nipp)
7	IF nipp != null
8	IF count == 0
9	Inisialisasi variabel data dengan nilai array yang berisikan nipp, nama_lengkap, unit_kerja, jabatan, kedudukan, nomor_telepon, tipe_user, email, fotoprofil, dan password
10	ELSE Pemanggilan method array_push(nipp_terdaftar,nipp)
11	END IF
12	ELSE Continue
13	END IF
14	END FOR
15	END FOREACH
16	IF nipp_data != null
17	IF nipp_terdaftar != null
18	Menampilkan pesan impor akun gagal karena ada nipp yang telah terdaftar
19	ELSE

Pseudocode fungsi buatAkunImport()	
	Pemanggilan fungsi untuk menyimpan data akun ke dalam database Menampilkan pesan akun berhasil diimpor
20	END IF
21	ELSE Menampilkan pesan tidak ada data di dalam file excel
22	END IF
23	Selesai

2. *Basis Path Testing*

   a. *Flow Graph*



Gambar 6.1 *Flow graph* fungsi buatAkunImport()

b. *Cyclomatic Complexity*

V(G) = R, dimana R merupakan jumlah region pada *flow graph*

V(G) = 7, yaitu R1, R2, R3, R4, R5, R6 dan R7

V(G) = 28 *edges* – 23 *nodes* + 2

= 7

V(G) = 6 *predicate nodes* + 1

= 7

c. *Independent Path*

Jalur 1 = 1-2-3-15-16-21-22-23

Jalur 2 = 1-2-3-**4-5-14**-3-15-16-21-22-23

Jalur 3 = 1-2-3-4-5-**6-7-8-10-11-13**-5-14-3-15-16-21-22-23

Jalur 4 = 1-2-3-4-5-6-7-8-10-11-13-5-14-3-15-16-**17-18-20**-22-23

Jalur 5 = 1-2-3-4-5-6-7-8-**9**-11-13-5-14-3-15-16-17-18-20-22-23

Jalur 6 = 1-2-3-4-5-6-7-8-9-11-13-5-14-3-15-16-17-**19**-20-22-23

Jalur 7 = 1-2-3-4-5-6-7-**12**-13-5-14-3-15-16-21-22-23

Tabel 6.2 Kasus uji fungsi buatAkunImport()

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1.	Penguji memasukkan *file excel* yang berstatus *protected* pada *input*an *file excel*. Kemudian penguji menekan tombol tambahkan akun. Kondisi awal tersebut menyebabkan tidak melewati *statement foreach* dan karena variabel data = null maka akan menjalanakan *statement else.*	Menampilkan pesan "Tidak ada data di dalam *file excel*."	Menampilkan pesan "Tidak ada data di dalam *file excel*."	Valid
2.	Penguji memasukkan *file excel* pada *input*an *file excel* dengan tidak ada data setelah *header*. Kondisi awal tersebut menyebabkan proses eksekusi akan melewati	Menampilkan pesan "Tidak ada data di dalam *file excel*."	Menampilkan pesan "Tidak ada data di dalam *file excel*."	Valid

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
	statement foreach lalu menginisialisasi variabel highestRow = 1 yang akan menyebabkan proses eksekusi tidak melewati *statement for*. Kemudian karena variabel data bernilai null maka akan menjalankan *statement else.*			
3.	Penguji memasukkan *file excel* dengan satu data yang berisi nipp yang sudah terdaftar pada sistem. Kondisi awal tersebut akan menyebabkan proses eksekusi akan melewati *statement foreach* lalu menginisialisasi variabel highestRow = 2 yang akan menyebabkan proses eksekusi akan melewati *statement for* yang kemudian akan menginisialisasi variabel nipp, nama_lengkap, unit_kerja, jabatan, kedudukan, nomor_telepon, tipe_user, *email*, fotoprofil, *password* dan *count*. Karena nilai variabel nipp tidak sama dengan null maka akan masuk ke *statement if* lalu akan mengecek variabel *count*. Karena sudah ada nipp yang sama yang terdaftar dalam sistem maka variabel *count* bernilai 1 yang membuat sistem menjalankan	Menampilkan pesan "Tidak ada data di dalam *file excel*."	Menampilkan pesan "Tidak ada data di dalam *file excel*."	Valid

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
	statement else yang akan menjalankan fungsi array_push() yang berfungsi untuk menambahkan nilai ke variabel nipp_terdaftar. Setelah itu akan keluar dari *statement for* dan *foreach* dan akan melakukan pengecekan pada statement *if.* Karena nilai dari variabel data=null maka akan menjalankan *statement else.*			
4.	Penguji memasukkan *file excel* dengan dua data yang berisi satu data yang mengandung nipp yang sudah terdaftar pada sistem (nipp=14045) dan nipp yang belum terdaftar untuk data yang lainnya. Kondisi awal tersebut akan menyebabkan proses eksekusi akan melewati *statement foreach* lalu menginisialisasi variabel highestRow = 3 yang akan menyebabkan proses eksekusi akan melewati *statement for*. Pada perulangan *for* yang pertama akan menginisialisasi variabel nipp, nama_lengkap, unit_kerja, jabatan, kedudukan, nomor_telepon, tipe_user, *email*, fotoprofil, *password* dan *count*. Karena nilai variabel nipp tidak sama dengan null maka akan	Menampilkan pesan "Impor akun gagal dilakukan, NIPP 14045 sudah terdaftar dalam sistem".	Menampilkan pesan "Impor akun gagal dilakukan, NIPP 14045 sudah terdaftar dalam sistem".	Valid

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
	masuk ke *statement if* lalu akan mengecek variabel *count*. Karena sudah ada nipp yang sama yang terdaftar dalam sistem maka variabel *count* bernilai 1 yang membuat sistem menjalankan *statement else* yang akan menjalankan fungsi array_push() yang berfungsi untuk menambahkan nilai ke variabel nipp_terdaftar. Pada perulangan *for* yang kedua akan menginisialisasi variabel nipp, nama_lengkap, unit_kerja, jabatan, kedudukan, nomor_telepon, tipe_user, *email*, fotoprofil, *password* dan *count*. Karena nilai variabel nipp tidak sama dengan null maka akan masuk ke *statement if* lalu akan mengecek variabel *count*. Karena belum ada nipp yang sama pada *database* maka variabel *count* akan bernilai 0 yang membuat sistem akan menjalankan *statement if* yang menginisialisasi variabel data. Setelah itu akan keluar dari *statement for* dan *foreach* dan akan melakukan pengecekan pada statement *if.* Karena nilai dari variabel data != null maka akan menjalankan *statement if*. Pada *statement if* akan			

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
	melakukan pengecekan apakah variabel nipp_terdaftar != null. Karena pada perulangan *for* yang pertama telah dilakukan penambahan nilai pada variabel nipp_terdaftar maka akan menjalankan *statement if* ini.			
5.	Penguji memasukkan *file excel* dengan dua data yang berisi satu data yang nippnya belum terdaftar pada sistem dan nipp yang sudah terdaftar pada sistem (nipp=14045) untuk data yang lainnya. Kondisi awal tersebut akan menyebabkan proses eksekusi akan melewati *statement foreach* lalu menginisialisasi variabel highestRow = 3 yang akan menyebabkan proses eksekusi akan melewati *statement for*. Pada perulangan *for* yang pertama akan menginisialisasi variabel nipp, nama_lengkap, unit_kerja, jabatan, kedudukan, nomor_telepon, tipe_user, *email*, fotoprofil, *password* dan *count*. Karena nilai variabel nipp tidak sama dengan null maka akan masuk ke *statement if* lalu akan mengecek variabel *count*. Karena belum ada nipp yang sama pada	Menampilkan pesan "Impor akun gagal dilakukan, NIPP 14045 sudah terdaftar dalam sistem".	Menampilkan pesan "Impor akun gagal dilakukan, NIPP 14045 sudah terdaftar dalam sistem".	Valid

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
	database maka variabel *count* akan bernilai 0 yang membuat sistem akan menjalankan *statement if* yang menginisialisasi variabel data. Pada perulangan *for* yang kedua akan menginisialisasi variabel nipp, nama_lengkap, unit_kerja, jabatan, kedudukan, nomor_telepon, tipe_user, *email*, fotoprofil, *password* dan *count*. Karena nilai variabel nipp tidak sama dengan null maka akan masuk ke *statement if* lalu akan mengecek variabel *count*. Karena sudah ada nipp yang sama yang terdaftar dalam sistem maka variabel *count* bernilai 1 yang membuat sistem menjalankan *statement else* yang akan menjalankan fungsi array_push() yang berfungsi untuk menambahkan nilai ke variabel nipp_terdaftar. Setelah itu akan keluar dari *statement for* dan *foreach* dan akan melakukan pengecekan pada statement *if.* Karena nilai dari variabel data != null maka akan menjalankan *statement if*. Pada *statement if* akan melakukan pengecekan apakah variabel nipp_terdaftar != null. Karena pada perulangan			

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
	for yang kedua telah dilakukan penambahan nilai pada variabel nipp_terdaftar maka akan menjalankan *statement if* ini.			
6.	Penguji memasukkan *file excel* dengan satu data yang berisi nipp belum terdaftar pada sistem. Kondisi awal tersebut akan menyebabkan proses eksekusi akan melewati *statement foreach* lalu menginisialisasi variabel highestRow = 2 yang akan menyebabkan proses eksekusi akan melewati *statement for*. Pada perulangan *for* yang pertama akan menginisialisasi variabel nipp, nama_lengkap, unit_kerja, jabatan, kedudukan, nomor_telepon, tipe_user, *email*, fotoprofil, *password* dan *count*. Karena nilai variabel nipp tidak sama dengan null maka akan masuk ke *statement if* lalu akan mengecek variabel *count*. Karena belum ada nipp yang sama pada *database* maka variabel *count* akan bernilai 0 yang membuat sistem akan menjalankan *statement if* yang menginisialisasi variabel data. Setelah itu akan keluar dari *statement for* dan *foreach* dan akan	Menyimpan data akun dalam *database* dan menampilkan pesan "Akun berhasil diimpor".	Menyimpan data akun dalam *database* dan menampilkan pesan "Akun berhasil diimpor".	Valid

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
	melakukan pengecekan pada *statement if.* Karena nilai dari variabel data != null maka akan menjalankan *statement if*. Pada *statement if* akan melakukan pengecekan apakah variabel nipp_terdaftar != null. Karena variabel nipp_terdaftar belum pernah diberikan nilai maka nilanya = null yang membuat sistem menjalankan *statement else*.			
7.	Penguji memasukkan *file excel* dengan satu data yang pada kolom nippnya dikosongkan, sedangkan pada data yang lain diisi. Kondisi awal seperti ini akan membuat sistem akan mengeksekusi *statement foreach* yang setelah itu akan menginisialisasi variabel highestRow = 2. Hal ini menyebabkan *statement for* akan dijalankan. Lalu karena kolom nipp dikosongkan maka akan menginisialisasi variabel nipp=null dan akan menginisialisasi nama_lengkap, unit_kerja, jabatan, kedudukan, nomor_telepon, tipe_user, *email*, fotoprofil, *password* dan *count* dengan nilai yang sudah ditentukan. Lalu akan dilakukan	Menampilkan pesan "Tidak ada data di dalam *file excel*."	Menampilkan pesan "Tidak ada data di dalam *file excel*."	Valid

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
	pengecekan apakah nilai dari variabel nipp != null. Karena variabel nipp sebelumnya telah diinisialisasi = null maka akan menjalankan *statement else* yang memberikan *statement continue* yang berarti melanjutkan perulangan. Karena highestRow bernilai sama dengan 2 yang berarti perulangan hanya dilakukan sekali maka keluar dari perulangan *for* dan keluar dari perulangan *foreach*. Selanjutnya dilakukan pengecekan terhadap nilai dari variabel data apakah nilainya != null. Karena pada proses eksekusi belum terjadi pemberian nilai terhadap variabel data maka variabel data akan bernilai null yang akan membuat sistem menjalankan *statement else.*			

6.1.2 Pengujian Unit Komponen Fungsi buatLaporanGangguan()

Pada Tabel 6.3 dapat dilihat *pseudocode* dari fungsi buatLaporanGangguan() dan pada Gambar 6.2 dapat dilihat *flowgraph* dari fungsi buatLaporanGangguan(). Pada Tabel 6.4 dapat dilihat kasus uji dari fungsi buatLaporanGangguan().

1. *Pseudocode* algoritme fungsi buatLaporanGangguan()

Nama kelas : C_Pelapor

Nama fungsi : buatLaporanGangguan()

Tabel 6.3 *Pseudocode* fungsi buatLaporanGangguan()

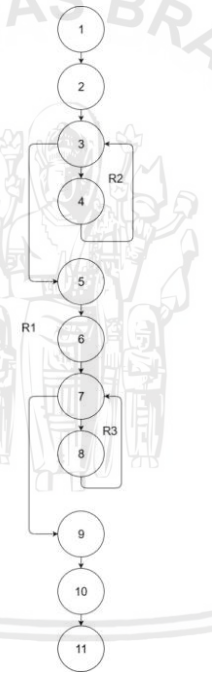
***Pseudocode* fungsi buatLaporanGangguan()**	
1	Mulai

***Pseudocode* fungsi buatLaporanGangguan()**	
2	Set waktu lokal menjadi waktu Indonesia Inisialisasi nipp_pelapor = variabel nipp dari session user yang sedang login Inisialisasi waktu = hasil kembalian dari pemanggilan fungsi date('Y-m-d H:i:s') Inisialisasi variabel kodegangguan = hasil kembalian dari pemanggilan fungsi random_string('alnum',6) Inisialiasasi variabel result = hasil kembalian dari pemanggilan fungsi cekKodeGangguan(kodegangguan)
3	WHILE result !=null
4	Inisialisasi variabel kodegangguan = hasil kembalian dari pemanggilan fungsi random_string('alnum',6) Inisialiasasi variabel result = hasil kembalian dari pemanggilan fungsi cekKodeGangguan(kodegangguan)
5	END WHILE
6	Inisialisasi variabel jenis_gangguan = nilai dari inputan form dengan nilai variabel name = jenis_gangguan Inisialisasi variabel deskripsi = nilai dari inputan form dengan nilai variabel name = deskripsi Inisialisasi variabel lokasi_gangguan = nilai dari inputan form dengan nilai variabel name = lokasi gangguan_gangguan Inisialisasi variabel status dengan nilai = "Belum Ditangani" Inisialisasi variabel array laporan = array("nipp_pelapor"=> nipp_pelapor, "kodegangguan" => kodegangguan, "jenis_gangguan" => jenis_gangguan, "deskripsi" => deskripsi, "lokasi_gangguan"=> lokasi_gangguan, "waktu" => waktu, "status" => status); Inisialisasi variabel lokasi dengan nilai sama dengan nilai hasil kembalian dari pemanggilan fungsi getLokasiPesan() Inisialisasi variabel jenisgangguan dengan nilai sama dengan nilai hasil kembalian dari pemanggilan fungsi getJenisGangguanPesan() Inisialisasi variabel message dengan nilai "Ada jenisgangguan terjadi di lokasi dengan NIPP pelapor: nipp pelapor pada waktu Pemanggilan metod buatLaporanGangguan() dengan parameter variabel laporan

***Pseudocode* fungsi buatLaporanGangguan()**	
	Inisialisasi variabel url dengan nilai sama dengan nilai kembalian dari fungsi site_url() Inisialisasi variabel petugasAktif sama dengan hasil kembalian dari pemanggilan fungsi getPetugasAktif()
7	FOREACH variabel petugasAktif
8	Mengirim notifikasi kepada petugas aktif
9	END FOREACH
10	Inisialisasi data['laporan'] = hasil pemanggilan dari method getDetailPelaporan(kodegangguan) Menampilkan halaman laporanSukses
11	Selesai

2. *Basis Path Testing*

   a. *Flow Graph*



Gambar 6.2 *Flow graph* fungsi buatLaporanGangguan()

   b. *Cyclomatic Complexity*

   V(G) = R, dimana R merupakan jumlah *region* pada graf

   V(G) = 3, ada 3 *region* yaitu R1, R2 dan R3

   V(G) = 12 *edge* – 11 *nodes* + 2

   = 3

   V(G) = 2 *predicate nodes* + 1

   = 3

c. *Independent Path*

Jalur 1 = 1-2-3-5-6-7-9-10-11

Jalur 2 = 1-2-3-**4**-3-5-6-7-9-10-11

Jalur 3 = 1-2-3-5-6-7-**8**-9-10-11

Tabel 6.4 Kasus uji fungsi buatLaporanGangguan()

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1.	*Set* waktu lokal menjadi indonesia. Inisialisasi variabel nipp_pelapor=nipp user yang sedang *login*. Inisialisasi variabel waktu = date('Y-m-d H:i:s'). Inisialisasi variabel kodegangguan = random_string() yang belum ada di *database*. Inisialisasi variabel result = cekKodeGangguan (kodegangguan) yang akan menghasilkan null karena kodegangguan tidak ada dalam *database*. Melakukan pengecekan pada *statement while*. Karena nilai dari variabel result = null maka tidak menjalankan *statement* didalam while. Inisialisasi variabel jenis gangguan, deskripsi, dan lokasi_gangguan dari *input*an penguji. Inisialisasi variabel status = "Belum ditangani". Inisialisasi variabel laporan = array(nipp_pelapor, kodegangguan, jenis_gangguan, deskripsi, lokasi_gangguan). Inisialisasi variabel lokasi =	Menyimpan laporan gangguan pada *database*, menampilkan halaman laporan sukses dan tidak memberi notifikasi.	Menyimpan laporan gangguan pada *database*, menampilkan halaman laporan sukses dan tidak memberi notifikasi.	Valid

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
	getLokasiPesan(lokasi_gangguan). Inisialisasi variabel jenisgangguan = getJenisGangguan(jenis_gangguan). Pemanggilan fungsi buatLaporanGangguan(). Inisialisasi variabel url = site_url () dengan parameter C_Petugas / viewLihatLaporanMasuk. Inisialisasi variabel petugasAktif = getPetugasAktif() dengan nilai kembalian array(null) atau sedang tidak ada petugas yang aktif. Proses eksekusi tidak melewati *statement foreach* karena nilai dari variabel petugasAktif = null dan menginisialisasi data['laporan'] = getDetailPelaporan(kodegangguan)			
2.	*Set* waktu lokal menjadi indonesia. Inisialisasi variabel nipp_pelapor=nipp user yang sedang *login*. Inisialisasi variabel waktu = date('Y-m-d H:i:s'). Inisialisasi variabel kodegangguan = random_string() yang sudah ada di *database*. Inisialisasi variabel result = cekKodeGangguan (kodegangguan) yang akan menghasilkan array()	Menyimpan laporan gangguan pada *database*, menampilkan halaman laporan sukses dan tidak memberi notifikasi.	Menyimpan laporan gangguan pada *database*, menampilkan halaman laporan sukses dan tidak memberi notifikasi.	Valid

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
	karena kodegangguan ada dalam *database*. Melakukan pengecekan pada *statement while*. Karena nilai dari variabel result != null maka akan menjalankan *statement* didalam while yaitu Inisialisasi variabel kodegangguan = hasil *generate* acak oleh fungsi random_string(). Inisialisasi variabel result = cekKodeGangguan(kodegangguan). Ini akan selalu dilakukan hingga kodegangguan yang di *generate* tidak terdaftar di *database*. Inisialisasi variabel jenis gangguan, deskripsi, dan lokasi_gangguan dari *input*an penguji. Inisialisasi variabel status = "Belum ditangani". Inisialisasi variabel laporan = array(nipp_pelapor, kodegangguan, jenis_gangguan, deskripsi, lokasi_gangguan). Inisialisasi variabel lokasi = getLokasiPesan(lokasi_gangguan). Inisialisasi variabel jenisgangguan = getJenisGangguan(jenis_gangguan). Pemanggilan fungsi buatLaporanGangguan(). Inisialisasi variabel url = site_url () dengan parameter C_Petugas /			

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
	viewLihatLaporanMasuk. Inisialisasi variabel petugasAktif = getPetugasAktif() dengan nilai kembalian array(null) atau sedang tidak ada petugas yang aktif. Proses eksekusi tidak melewati *statement foreach* karena nilai dari variabel petugasAktif = null dan menginisialisasi data['laporan'] = getDetailPelaporan(kodegangguan)			
3.	*Set* waktu lokal menjadi indonesia. Inisialisasi variabel nipp_pelapor=nipp user yang sedang *login*. Inisialisasi variabel waktu = date('Y-m-d H:i:s'). Inisialisasi variabel kodegangguan = random_string() yang belum ada di *database*. Inisialisasi variabel result = cekKodeGangguan(kodegangguan) yang akan menghasilkan null karena kodegangguan tidak ada dalam *database*. Melakukan pengecekan pada *statement while*. Karena nilai dari variabel result = null maka tidak menjalankan *statement* didalam while. Inisialisasi variabel jenis gangguan, deskripsi, dan lokasi_gangguan dari *input*an penguji. Inisialisasi	Menyimpan laporan gangguan pada *database*, menampilkan halaman laporan sukses dan memberi notifikasi kepada petugas yang sedang aktif.	Menyimpan laporan gangguan pada *database*, menampilkan halaman laporan sukses dan memberi notifikasi kepada petugas yang sedang aktif.	Valid

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
	variabel status = “Belum ditangani”. Inisialisasi variabel laporan = array(nipp_pelapor, kodegangguan, jenis_gangguan, deskripsi, lokasi_gangguan). Inisialisasi variabel lokasi = getLokasiPesan(lokasi_gangguan). Inisialisasi variabel jenisgangguan = getJenisGangguan(jenis_gangguan). Pemanggilan gungsi buatLaporanGangguan(). Inisialisasi variabel url = site_url () dengan parameter C_Petugas / viewLihatLaporanMasuk. Inisialisasi variabel petugasAktif = getPetugasAktif() dengan nilai kembalian array() yang berisi daftar petugas yang sedang aktif. Proses eksekusi akan melewati *statement foreach* karena nilai dari variabel petugasAktif != array(null) yang akan mengirimkan pesan kepada petugas yang sedang aktif dan menginisialisasi data[‘laporan’] = getDetailPelaporan(kodegangguan)			

6.1.3 Pengujian Unit Komponen Fungsi cekNipp()

Pada Tabel 6.5 dapat dilihat *pseudocode* dari fungsi cekNipp() dan pada Gambar 6.3 dapat dilihat *flowgraph* dari fungsi cekNipp(). Pada Tabel 6.6 dapat dilihat kasus uji dari fungsi cekNipp().

3. *Pseudocode* algoritme fungsi cekNipp()

Nama kelas : C_Pimpinan

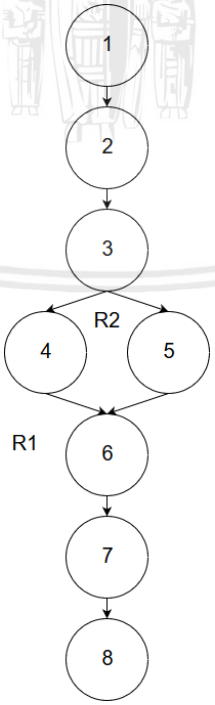
Nama fungsi : cekNipp()

Tabel 6.5 *Pseudocode* fungsi cekNipp()

***Pseudocode* fungsi cekNipp()**	
1	Mulai
2	Inisialisasi nipp = nilai inputan Pimpinan Inisialisasi exist = kembalian dari pemanggilan fungsi cekNipp(nipp) Inisialisasi count = hasil pemanggilan fungsi count(exists) Inisialisasi data['state'] = 0
3	IF count > 0
4	Inisialisasi data['state'] = 1
5	ELSE Inisialisasi data['state'] = 0
6	END IF
7	Cetak hasil pemanggilan fungsi json_encode(data)
8	Selesai

4. *Basis Path Testing*

a. *Flow Graph*



Gambar 6.3 *Flow graph* fungsi cekNipp()

b. *Cyclomatic Complexity*

V(G) = R, dimana R merupakan jumlah *region* pada graf

V(G) = 2, ada 2 *region* yaitu R1 dan R2

V(G) = 8 *edge* – 8 *nodes* + 2

= 2

V(G) = 1 *predicate nodes* + 1

= 2

c. *Independent Path*

Jalur 1 = 1-2-3-4-6-7-8

Jalur 2 = 1-2-3-**5**-6-7-8

Tabel 6.6 Kasus uji fungsi cekNipp()

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1.	Inisialisasi variabel nipp dengan nilai yang sudah ada pada *database*. Inisialisasi variabel exist = cekNipp(nipp) yang karena nilai nipp sudah ada *database* akan mengembalikan array data gangguan yang memiliki nipp sama dengan nilai variabel nipp. Inisialisasi variabel count = count(exist). Karena nilai variabel exist mengandung data gangguan yang memiliki nipp sama dengan variabel nipp, maka variabel count bernilai 1. Inisialisasi variabel data['state'] = 0. Karena nilai variabel count bernilai satu maka memenuhi *statement if* yang berarti akan menjalankan *statement* inisialisasi data['state'] = 1. Lalu akan mencetak hasil	Mencetak json {"state":1}	Mencetak json {"state":1}	Valid

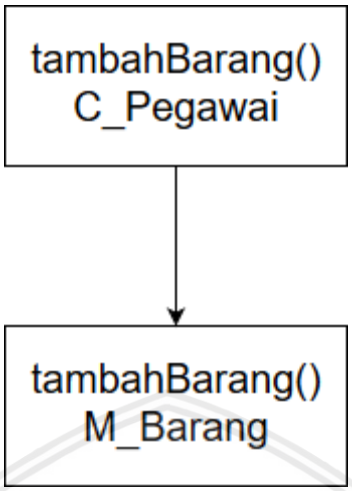
No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
	pemanggilan fungsi json_encode(data);			
2.	Inisialisasi variabel nipp dengan nilai yang belum ada pada *database*. Inisialisasi variabel exist = cekNipp(nipp) yang karena nilai nipp belum ada *database* akan mengembalikan array(null) karena tidak ada nipp yang sama. Inisialisasi variabel count = count(exist). Karena nilai variabel exist tidak mengandung data gangguan yang memiliki nipp sama dengan variabel nipp, maka variabel count bernilai 0. Inisialisasi variabel data['state'] = 0. Karena nilai variabel count bernilai 0 maka tidak memenuhi *statement if* yang berarti akan menjalankan *statement else* yang melakukan inisialisasi data['state'] = 0. Lalu akan mencetak hasil pemanggilan fungsi json_encode(data);	Mencetak json {"state":0}	Mencetak json {"state":0}	Valid

6.2 Pengujian Integrasi

Pengujian integrasi merupakan pengujian yang dilakukan untuk menguji komponen atau unit yang saling berinteraksi dalam sistem untuk membangun sebuah fungsionalitas sistem. Pada penelitian ini, pengujian integrasi akan dilakukan dengan pendekatan *top-down*.

Pengujian integrasi akan dilakukan dengan menggunakan sampel fungsional menambah barang di daftar barang gudang penyimpanan. Dalam menjalankan fungsional ini fungsi tambahBarang() pada kelas C_Pegawai terintegrasi dengan fungsi tambahBarang() pada kelas M_Barang seperti yang digambarkan pada

Gambar 6.4. Pada pengujian ini akan dibuat sebuah *stub* yang digunakan untuk menggantikan modul yang sedang diuji.



Gambar 6.4 Diagram hierarki pengujian integrasi

Tabel 6.7 Langkah Uji Pengujian Integrasi

No.	Langkah Uji	Keterangan
1.	Menjalankan fungsi tambahBarang() pada kelas C_Pegawai → Menjalankan fungsi stubTambahBarang() pada kelas M_Barang	Fungsi tambahBarang() pada kelas C_Pegawai dijalankan untuk menguji fungsi tambahBarang() dengan menggunakan stubTambahBarang() yang diberikan nilai *input* nama_barang = LCD, qty = 5 dan nama_barang dan qty dikosongkan

Tabel 6.8 *Pseudocode* stubTambahBarang()

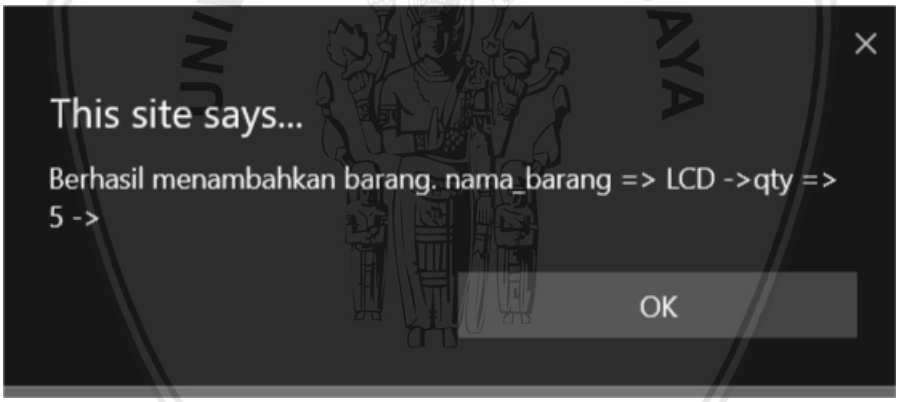
***Pseudocode* stubTambahBarang()**
MULAI Menerima parameter data Return variabel data SELESAI

Tabel 6.9 *Pseudocode* pengujian fungsi stubTambahBarang() dengan nilai variabel nama_barang = LCD dan nilai variabel qty = 5

***Pseudocode* pengujian fungsi stubTambahBarang() dengan nilai variabel nama_barang = LCD dan nilai variabel qty = 5**
MULAI Inisialisasi variabel nama_barang = 'LCD' Inisialisasi variabel qty = 5 IF(nama_barang!= null && qty != null) Inisialisasi variabel data = array(

```
                'nama_barang' => $nama_barang,
                'qty' => $qty)
Inisialisasi variabel result = hasil kembalian dari pemanggilan
fungsi stubTambahBarang(data)
Menampilkan alert nilai dari variabel data
ELSE
Menampilkan alert gagal menambahkan barang
END IF
SELESAI
```

Tabel 6.8 merupakan *peseudocode* dari stubTambahBarang() yang telah dibuat untuk pengujian integrasi yang dilakukan. Tabel 6.9 merupakan *peseudocode* yang digunakan untuk pengujian integrasi fungsi tambahBarang() dengan fungsi stubTambahBarang(). Hasil pengujian dari penggunaan variabel nama_barang dengan nilai 'LCD' dan variabel qty dengan nilai 5 adalah fungsi tambahBarang() sukses memanggil fungsi stubTambahBarang() dengan nilai yang dikembalikan oleh fungsi stubTambahBarang() sama dengan nilai yang telah dilewatkan menjadi parameter saat pemanggilan fungsi stubTambahBarang() itu sendiri sebagaimana dapat dilihat pada Gambar 6.5.

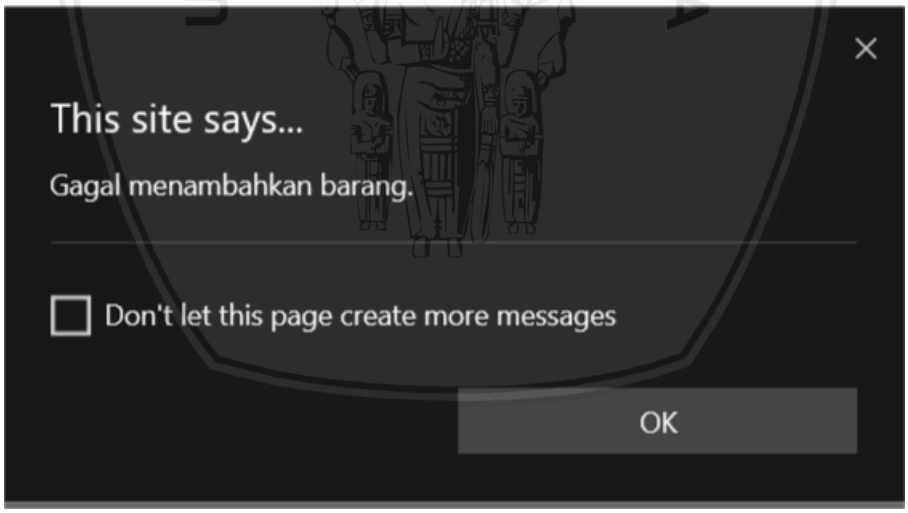


Gambar 6.5 Hasil pengujian fungsi stubTambahBarang() dengan nilai variabel nama_barang = LCD dan nilai variabel qty = 5

Setelah itu nilai dari variabel nama_barang dan qty kemudian dirubah menjadi kosong atau *empty* untuk menguji apakah fungsi tambahBarang() berhasil tidak memanggil fungsi stubTambahBarang(). Tabel 6.10 merupakan *pseudocode* dari pengujian integrasi fungsi tambahBarang() dengan fungsi stubTambahBarang() dengan nilai variabel nama_barang = '' dan nilai qty ='. Hasil dari penggunaan variabel nama_barang = '' dan variabel qty = '' adalah berhasil masuk ke *statement else* yang tidak memanggil stubTambahBarang(). Dapat dilihat bahwa bukti bahwa proses eksekusi program masuk ke dalam *statement else* yaitu sebagaimana yang ditampilkan Gambar 6.6 bahwa sistem menampilkan pesan "Gagal menambahkan barang".

Tabel 6.10 *Pseudocode* pengujian fungsi stubTambahBarang() dengan nilai variabel nama_barang = '' dan nilai variabel qty = ''

***Pseudocode* pengujian fungsi stubTambahBarang() dengan nilai variabel nama_barang = '' dan nilai variabel qty = ''**
MULAI
Inisialisasi variabel nama_barang = ''
Inisialisasi variabel qty = ''; IF(nama_barang!= null && qty != null) Inisialisasi variabel data = array('nama_barang' => $nama_barang, 'qty' => $qty)
Inisialisasi variabel result = hasil kembalian dari pemanggilan fungsi stubTambahBarang(data)
Menampilkan alert nilai dari variabel data
ELSE
Menampilkan alert gagal menambahkan barang
END IF
SELESAI



Gambar 6.6 Hasil pengujian fungsi stubTambahBarang() dengan nilai variabel nama_barang = '' dan nilai variabel qty = ''

Selanjutnya langkah yang harus dilakukan adalah mengintegrasikan fungsi tambahBarang() pada kelas C_Pegawai dengan fungsi tambahBarang() pada kelas M_Barang() untuk menguji apakah proses integrasi yang terjadi akan sama dengan pengujian yang telah dilakukan sebelumnya.

Tabel 6.11 *Pseudocode* fungsi tambahBarang() kelas C_Pegawai

***Pseudocode* fungsi tambahBarang() kelas C_Pegawai**

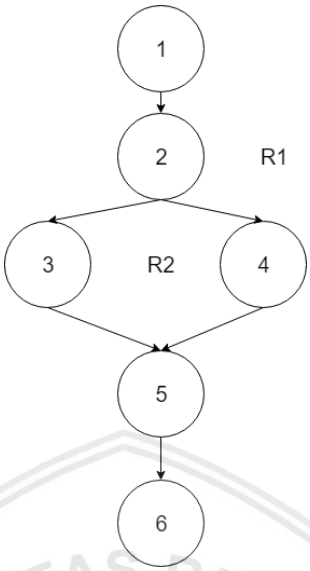
```
MULAI
Inisialisasi variabel nama_barang = ‘’
Inisialisasi variabel qty = ‘’;
IF(nama_barang!= null && qty != null)
Inisialisasi variabel data = array(
                'nama_barang' => $nama_barang,
                'qty' => $qty)
Pemanggilan fungsi tambahBarang (data)
Menampilkan alert berhasil menambahkan barang
Redirect ke halaman manajemen gudang penyimpanan
ELSE
Menampilkan alert gagal menambahkan barang
Redirect ke halaman manajemen gudang penyimpanan
END IF
SELESAI
```

Tabel 6.12 *Pseudocode* fungsi tambahBarang() kelas M_Barang

***Pseudocode* fungsi tambahBarang() kelas M_Barang**
MULAI Menerima parameter data Pemanggilan fungsi insert(‘t_barang’,data) SELESAI

Tabel 6.11 merupakan *pseudocode* fungsi tambahBarang() pada kelas C_Pegawai dan Tabel 6.12 merupakan *pseudocode* fungsi tambahBarang() pada kelas M_Barang yang akan dilakukan pengujian integrasi. Pada pengujian integrasi ini akan dilakukan dengan teknik *white-box testing* dengan menggunakan *basis path testing*.

1. *Flow Graph*



Gambar 6.7 *Flow graph* integrasi fungsi tambahBarang() kelas C_Pegawai dan funngsi tambahBarang() kelas M_Model

2. *Cyclomatic Complexity*

   V(G) = jumlah *region*

   = 2, yaitu R1 dan R2

   V(G) = 6 *edges* – 5 *node* + 2

   = 2

   V(G) = 1 *predicate node* + 1

   = 2

3. *Independent Path*

   Jalur 1 = 1-2-**3**-5-6

   Jalur 2 = 1-2-**4**-5-6

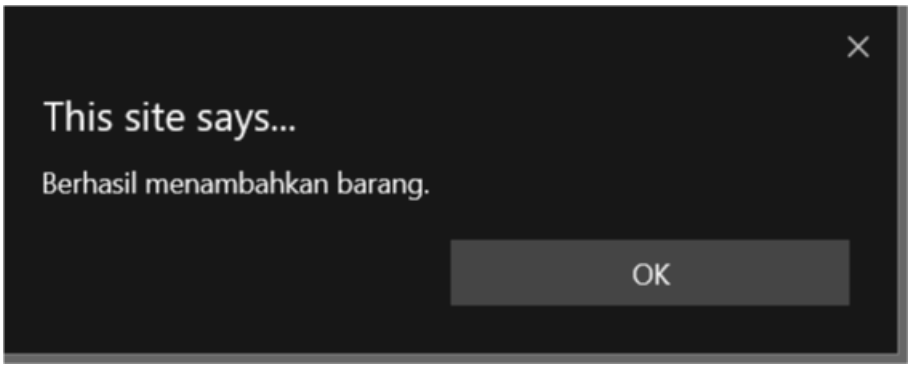
Tabel 6.13 Kasus uji pengujian integrasi fungsi tambahBarang() kelas C_Pegawai dan fungsi tambahBarang() kelas M_Barang

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1.	Nilai dari variabel nama_barang = 'LCD' Nilai dari variabel qty = 5 sehingga akan memanggil fungsi tambahBarang() pada kelas M_Barang	Fungsi tambahBarang() pada kelas C_Pegawai akan memanggil fungsi tambahBarang() pada kelas	Fungsi tambahBarang() pada kelas C_Pegawai akan memanggil fungsi tambahBarang() pada kelas	Valid

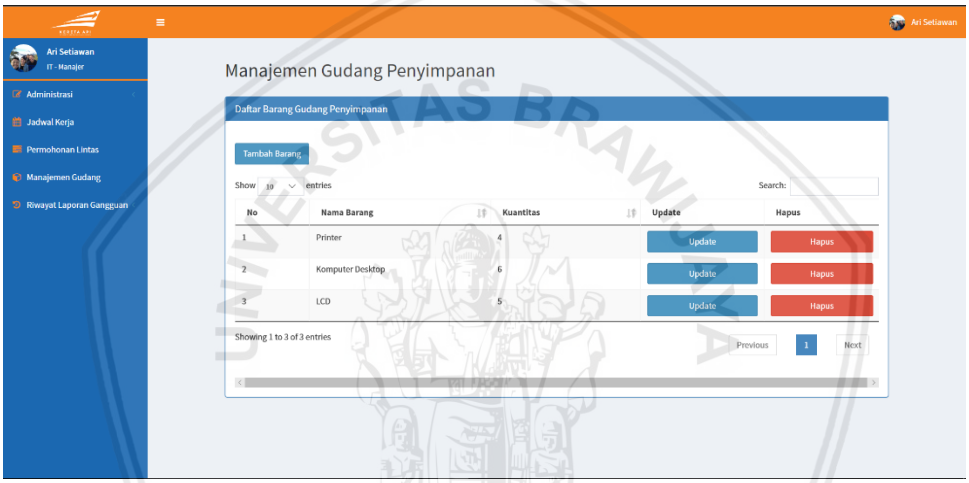
No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
		M_Barang untuk melakukan penyimpanan data ke dalam *database*, menampilkan pesan "Berhasil menambahkan barang" dan menampilkan halaman manajemen gudang penyimpanan.	M_Barang untuk melakukan penyimpanan data ke dalam *database*, menampilkan pesan "Berhasil menambahkan barang" dan menampilkan halaman manajemen gudang penyimpanan.	
2.	Nilai dari variabel nama_barang = '' atau kosong Nilai dari variabel qty = '' atau kosong sehingga tidak akan memanggil fungsi tambahBarang() pada kelas M_Barang	Fungsi tambahBarang() pada kelas C_Pegawai tidak memanggil fungsi tambahBarang() pada kelas M_Barang, menampilkan pesan "Gagal menambahkan barang" dan menampilkan halaman manajemen gudang penyimpanan.	Fungsi tambahBarang() pada kelas C_Pegawai tidak memanggil fungsi tambahBarang() pada kelas M_Barang, menampilkan pesan "Gagal menambahkan barang" dan menampilkan halaman manajemen gudang penyimpanan.	Valid

Langkah berikutnya adalah membandingkan hasil yang telah didapatkan pada saat menggunakan fungsi stubTambahBarang() dengan hasil yang telah didapatkan dengan pada saat menggunakan fungsi tambahBarang() pada kelas M_Barang. Pada Gambar 6.8 dan Gambar 6.9 ditunjukkan hasil dari pengujian integrasi antara fungsi tambahBarang() di kelas C_Pegawai dan fungsi tambahBarang() di kelas M_Barang pada kasus uji 1. Karena hasil yang didapatkan

sama dengan hasil yang diharapkan maka status pengujian pada jalur pertama dikatakan valid.

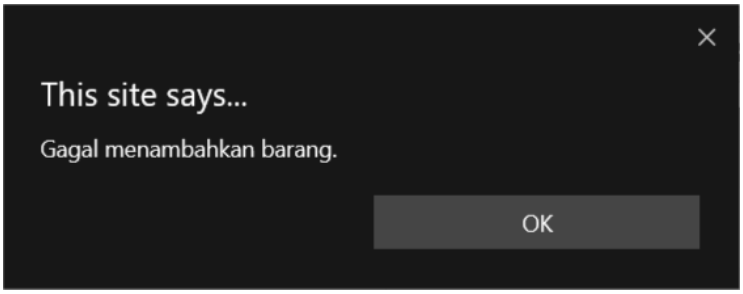


Gambar 6.8 Hasil uji jalur 1 berhasil menampilkan pesan

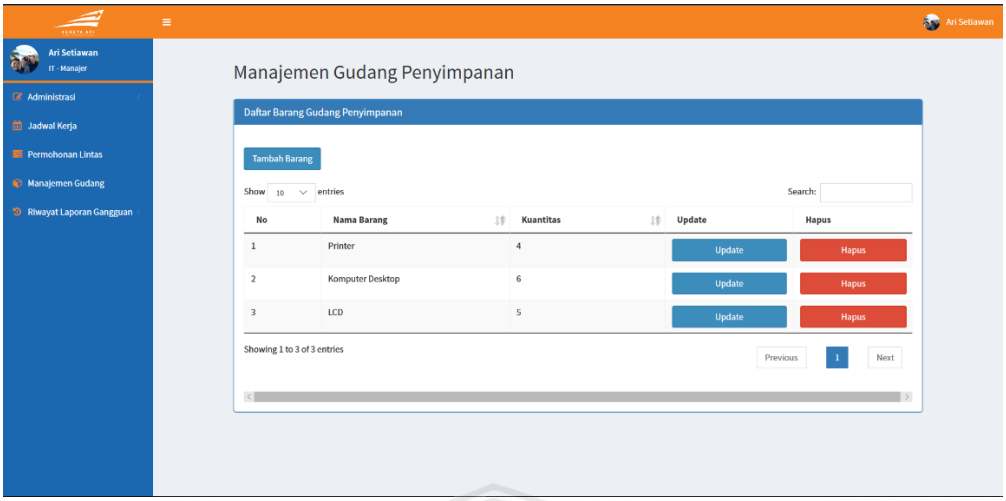


Gambar 6.9 Hasil uji jalur 1 berhasil menampilkan halaman manajemen gudang penyimpanan

Pada Gambar 6.10 dan Gambar 6.11 ditunjukkan hasil dari pengujian integrasi antara fungsi tambahBarang() di kelas C_Pegawai dan fungsi tambahBarang() di kelas M_Barang pada kasus uji 2. Karena hasil yang didapatkan sama dengan hasil yang diharapkan maka status pengujian pada jalur pertama dikatakan valid.



Gambar 6.10 Hasil uji jalur 2 berhasil menampilkan pesan



Gambar 6.11 Hasil uji jalur 2 berhasil menampilkan halaman manajemen gudang penyimpanan

6.3 Pengujian Validasi Kebutuhan Fungsional

Pengujian validasi merupakan pengujian yang digunakan untuk mengetahui apakah sistem yang sudah dibangun sesuai dengan kebutuhan yang telah didefinisikan pada proses analisis kebutuhan atau tidak. Pengujian validasi ini dilakukan dengan melakukan pemeriksaan sistem untuk memastikan sistem berjalan dengan baik dan tidak ada kesalahan yang terjadi. Pengujian validasi ini akan dilakukan dengan menggunakan *black-box testing*. Tabel 6.14 sampai Tabel 6.110 merupakan hasil dari pengujian vadilasi kebutuhan fungsional.

6.3.1 Pengujian Validasi *Login*

6.3.1.1 Kasus Uji *Login* Berhasil

Tabel 6.14 Kasus uji *login* berhasil

Nama Kasus Uji	Kasus uji *login* berhasil.
Prosedur	1. Penguji masuk ke halaman *login.* 2. Penguji memasukkan NIPP dan *password.* 3. Penguji menekan tombol *login.*
Hasil yang diharapkan	Sistem menampilkan halaman utama.
Hasil	Sistem menampilkan halaman utama.
Status	Valid.

6.3.1.2 Kasus Uji *Login* dengan NIPP Tidak Terisi

Tabel 6.15 Kasus uji *login* dengan NIPP tidak terisi

Nama Kasus Uji	Kasus uji *login* dengan NIPP tidak terisi.

Prosedur	1. Penguji masuk ke halaman *login.* 2. Penguji mengosongkan NIPP dan memasukkan *password.* 3. Penguji menekan tombol *login.*
Hasil yang diharapkan	Menampilkan pesan "NIPP wajib diisi terdiri dari angka".
Hasil	Menampilkan pesan "NIPP wajib diisi terdiri dari angka".
Status	Valid.

6.3.1.3 Kasus Uji *Login* dengan *Password* Tidak Terisi

Tabel 6.16 Kasus uji *login* dengan *password* tidak terisi

Nama Kasus Uji	Kasus uji *login* dengan *password* tidak terisi.
Prosedur	1. Penguji masuk ke halaman *login.* 2. Penguji mengosongkan *password* dan memasukkan NIPP. 3. Penguji menekan tombol *login.*
Hasil yang diharapkan	Menampilkan pesan "*Password* wajib diisi".
Hasil	Menampilkan pesan "*Password* wajib diisi".
Status	Valid.

6.3.1.4 Kasus Uji *Login* dengan NIPP atau *Password* Tidak Sesuai dengan *Database*

Tabel 6.17 Kasus uji login dengan NIPP atau *password* tidak sesuai dengan *database*

Nama Kasus Uji	Kasus uji login dengan NIPP atau *password* tidak sesuai dengan *database.*
Prosedur	1. Penguji masuk ke halaman *login.* 2. Penguji memasukkan NIPP = 1 dan *password* = pass. 3. Penguji menekan tombol *login.*
Hasil yang diharapkan	Menampilkan pesan "NIPP atau *password* yang anda masukkan salah, silahkan coba lagi".
Hasil	Menampilkan pesan "NIPP atau *password* yang anda masukkan salah, silahkan coba lagi".
Status	Valid.

6.3.2 Kasus Uji *Logout*

6.3.2.1 Kasus Uji *Logout*

Tabel 6.18 Kasus uji *logout*

Nama Kasus Uji	Kasus uji *logout.*
Prosedur	1. Penguji berada dalam kondisi *login.* 2. Penguji menekan foto profil di pojok kanan atas. 3. Penguji menekan tombol *logout.*
Hasil yang diharapkan	Sistem menampilkan halaman *login.*
Hasil	Sistem menampilkan halaman *login.*
Status	Valid.

6.3.3 Kasus Uji Melihat Daftar Akun

6.3.3.1 Kasus Uji Melihat Daftar Akun

Tabel 6.19 Kasus uji melihat daftar akun

Nama Kasus Uji	Kasus uji melihat daftar akun.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu administrasi. 3. Penguji memilih menu daftar akun.
Hasil yang diharapkan	Sistem akan menampilkan halaman daftar akun yang akan menampilkan tabel yang didalamnya terdapat data nomor, NIPP, nama lengkap, jabatan, unit kerja, kedudukan, nomor telepon, tipe *user*, dan *email* dari setiap akun serta tombol untuk menghapus akun tersebut.
Hasil	Sistem akan menampilkan halaman daftar akun yang akan menampilkan tabel yang didalamnya terdapat data nomor, NIPP, nama lengkap, jabatan, unit kerja, kedudukan, nomor telepon, tipe *user*, dan *email* dari setiap akun serta tombol untuk menghapus akun tersebut.
Status	Valid.

6.3.4 Kasus Uji Menambahkan Akun

6.3.4.1 Kasus Uji Menambahkan Akun Berhasil

Tabel 6.20 Kasus uji menambahkan akun berhasil

Nama Kasus Uji	Kasus uji menambahkan akun berhasil.
Prosedur	1. Penguji berada pada halaman tambah akun. 2. Penguji mengisi NIPP yang belum ada pada *database*, nama lengkap, unit kerja, jabatan, kedudukan, nomor telepon, *email*, dan tipe akun. 3. Penguji menekan tombol tambahkan akun.
Hasil yang diharapkan	Sistem menyimpan data akun dan menampilkan pesan "Berhasil membuat akun baru".
Hasil	Sistem menyimpan data akun dan menampilkan pesan "Berhasil membuat akun baru".
Status	Valid.

6.3.4.2 Kasus Uji Menambahkan Akun dengan NIPP yang Telah Terdaftar

Tabel 6.21 Kasus uji menambahkan akun dengan nipp yang telah terdaftar

Nama Kasus Uji	Kasus uji menambahkan akun dengan nipp yang telah terdaftar.
Prosedur	1. Penguji berada pada halaman tambah akun. 2. Penguji mengisi NIPP yang sudah ada pada *database* pada *field* NIPP.
Hasil yang diharapkan	Menampilkan pesan "NIPP sudah terdaftar".
Hasil	Menampilkan pesan "NIPP sudah terdaftar".
Status	Valid.

6.3.4.3 Kasus Uji Menambahkan Akun dengan Mengosongkan Salah Satu *Input*an

Tabel 6.22 Kasus uji menambahkan akun dengan mengosongkan salah satu *input*an

Nama Kasus Uji	Kasus uji menambahkan akun dengan mengosongkan salah satu *input*an.
Prosedur	1. Penguji berada pada halaman tambah akun.

	2. Penguji mengisi *form* tambah akun dengan tidak lengkap dan dengan NIPP yang sudah terdaftar pada *database.*
Hasil yang diharapkan	Sistem akan *disable* tombol "Tambahkan Akun".
Hasil	Sistem akan *disable* tombol "Tambahkan Akun"..
Status	Valid.

6.3.5 Kasus Uji Menambahkan Akun dengan Impor *File*

6.3.5.1 Kasus Uji Menambahkan Akun dengan Impor *File* Berhasil

Tabel 6.23 Kasus uji menambahkan akun dengan impor *file* berhasil

Nama Kasus Uji	Kasus uji menambahkan akun dengan impor *file* berhasil.
Prosedur	1. Penguji berada pada halaman tambah akun dengan impor. 2. Penguji mengunduh format yang disediakan. 3. Penguji mengisi data setiap baris dengan lengkap. 4. Penguji mengisi *form* tambah akun dengan impor *file* dengan lengkap. 5. Penguji menekan tombol tambahkan akun.
Hasil yang diharapkan	Sistem menyimpan data akun dan menampilkan pesan "Akun berhasil diimpor".
Hasil	Sistem menyimpan data akun dan menampilkan pesan "Akun berhasil diimpor".
Status	Valid.

6.3.5.2 Kasus Uji Menambahkan Akun dengan Impor *File* dengan *Form* Tidak Diisi Lengkap

Tabel 6.24 Kasus uji menambahkan akun dengan impor *file* dengan *form* tidak diisi lengkap

Nama Kasus Uji	Kasus uji menambahkan akun dengan impor *file* dengan *form* tidak diisi lengkap.
Prosedur	1. Penguji berada pada halaman tambah akun dengan impor. 2. Penguji mengunduh format yang disediakan.

	3. Penguji mengisi data setiap baris dengan lengkap. 4. Penguji tidak mengisi *form* tambah akun dengan impor *file* dengan lengkap. 5. Penguji menekan tombol tambahkan akun.
Hasil yang diharapkan	Sistem menampilkan pesan "*Please select a file*".
Hasil	Sistem menampilkan pesan "*Please select a file*".
Status	Valid.

6.3.5.3 Kasus Uji Menambahkan Akun dengan Impor *File* dengan NIPP yang Telah Tedaftar pada *Database*

Tabel 6.25 Kasus uji menambahkan akun dengan impor *file* dengan NIPP yang telah tedaftar pada *database*

Nama Kasus Uji	Kasus uji menambahkan akun dengan impor *file* dengan NIPP yang telah tedaftar pada *database.*
Prosedur	1. Penguji berada pada halaman tambah akun dengan impor. 2. Penguji mengunduh format yang disediakan. 3. Penguji mengisi data setiap baris dengan lengkap dan dengan NIPP yang telah terdaftar pada *database.* 4. Penguji mengisi *form* tambah akun dengan impor *file* dengan lengkap. 5. Penguji menekan tombol tambahkan akun.
Hasil yang diharapkan	Sistem menampilkan pesan "Impor akun gagal dilakukan, NIPP (NIPP yang terdaftar) sudah terdaftar dalam sistem."
Hasil	Sistem menampilkan pesan "Impor akun gagal dilakukan, NIPP (NIPP yang terdaftar) sudah terdaftar dalam sistem."
Status	Valid.

6.3.5.4 Kasus Uji Menambahkan Akun dengan Impor *File* dengan Tidak Mengisi *File* Format

Tabel 6.26 Kasus uji menambahkan akun dengan impor *file* dengan tidak mengisi *file* format

Nama Kasus Uji	Kasus uji menambahkan akun dengan impor *file* dengan tidak mengisi *file* format.

Prosedur	1. Penguji berada pada halaman tambah akun dengan impor. 2. Penguji mengunduh format yang disediakan. 3. Penguji tidak mengisi data pada format yang disediakan. 4. Penguji mengisi *form* tambah akun dengan impor *file* dengan lengkap. 5. Penguji menekan tombol tambahkan akun.
Hasil yang diharapkan	Sistem menampilkan pesan "Tidak ada data di dalam *file excel*."
Hasil	Sistem menampilkan pesan "Tidak ada data di dalam *file excel*."
Status	Valid.

6.3.6 Kasus Uji Menghapus Akun

6.3.6.1 Kasus Uji Menghapus Akun Berhasil

Tabel 6.27 Kasus uji menghapus akun berhasil

Nama Kasus Uji	Kasus uji menghapus akun berhasil.
Prosedur	1. Penguji berada pada halaman daftar akun. 2. Penguji menekan tombol hapus pada satu baris akun yang ingin dihapus. 3. Penguji menekan tombol Ya pada modal konfirmasi hapus akun.
Hasil yang diharapkan	Sistem menghapus data akun dan menampilkan halaman daftar akun.
Hasil	Sistem menghapus data akun dan menampilkan halaman daftar akun.
Status	Valid.

6.3.6.2 Kasus Uji Menghapus Akun dengan Menekan Tombol Tidak Pada Modal Hapus Akun

Tabel 6.28 Kasus uji menghapus akun dengan menekan tombol tidak pada modal hapus akun

Nama Kasus Uji	Kasus uji menghapus akun dengan menekan tombol tidak pada modal hapus akun.
Prosedur	1. Penguji berada pada halaman daftar akun.

	2. Penguji menekan tombol hapus pada satu baris akun yang ingin dihapus. 3. Penguji menekan tombol Tidak pada modal hapus akun.
Hasil yang diharapkan	Sistem menutup modal konfirmasi hapus akun.
Hasil	Sistem menutup modal konfirmasi hapus akun.
Status	Valid.

6.3.7 Kasus Uji Melihat Jadwal Kerja

6.3.7.1 Kasus Uji Melihat Jadwal Kerja Berhasil

Tabel 6.29 Kasus uji melihat jadwal kerja berhasil

Nama Kasus Uji	Kasus uji melihat jadwal kerja berhasil.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu Jadwal Kerja.
Hasil yang diharapkan	Sistem menampilkan jadwal kerja yaitu NIPP, nama lengkap, semua tanggal pada bulan tersebut, dan jadwal masing masing pegawai di setiap tanggalnya.
Hasil	Sistem menampilkan jadwal kerja yaitu NIPP, nama lengkap, semua tanggal pada bulan tersebut, dan jadwal masing masing pegawai di setiap tanggalnya.
Status	Valid.

6.3.7.2 Kasus Uji Melihat Jadwal Kerja Pada Bulan yang Belum Dibuat

Tabel 6.30 Kasus uji melihat jadwal kerja pada bulan yang belum dibuat

Nama Kasus Uji	Kasus uji melihat jadwal kerja pada bulan yang belum dibuat.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu Jadwal Kerja. 3. Penguji memilih bulan yang belum dibuat jadwal kerjanya.
Hasil yang diharapkan	Sistem menampilkan "*No data available in table*" pada tabel.
Hasil	Sistem menampilkan "*No data available in table*" pada tabel.
Status	Valid.

6.3.8 Kasus Uji Menambahkan Jadwal Kerja

6.3.8.1 Kasus Uji Menambahkan Jadwal Kerja Berhasil

Tabel 6.31 Kasus uji menambahkan jadwal kerja berhasil

Nama Kasus Uji	Kasus uji menambahkan jadwal kerja berhasil.
Prosedur	1. Penguji berada pada halaman jadwal kerja. 2. Penguji menekan tombol tambah jadwal.
Hasil yang diharapkan	Sistem menyimpan jadwal kerja, menampilkan pesan "Berhasil membuat jadwal kerja" dan menampilkan halaman jadwal kerja.
Hasil	Sistem menyimpan jadwal kerja, menampilkan pesan "Berhasil membuat jadwal kerja" dan menampilkan halaman jadwal kerja.
Status	Valid.

6.3.9 Kasus Uji Melihat Daftar Permohonan *Lintas*

6.3.9.1 Kasus Uji Melihat Daftar Permohonan *Lintas* Ketika Terdapat Permohonan *Lintas*

Tabel 6.32 Kasus uji melihat daftar permohonan *lintas* ketika terdapat permohonan *lintas*

Nama Kasus Uji	Kasus uji melihat daftar permohonan *lintas* ketika terdapat permohonan *lintas.*
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji menekan menu Permohonan *Lintas.*
Hasil yang diharapkan	Sistem akan menampilkan halaman daftar permohonan *lintas* yang terdapat tabel daftar permohonan *lintas* yang memiliki kolom kode gangguan, NIPP petugas, lokasi gangguan, jenis gangguan dan tombol izinkan.
Hasil	Sistem akan menampilkan halaman daftar permohonan *lintas* yang terdapat tabel daftar permohonan *lintas* yang memiliki kolom kode gangguan, NIPP petugas, lokasi gangguan, jenis gangguan dan tombol izinkan.
Status	Valid.

6.3.9.2 Kasus Melihat Daftar Permohonan *Lintas* Ketika Tidak Terdapat Permohonan *Lintas*

Tabel 6.33 Kasus uji melihat daftar permohonan *lintas* ketika tidak terdapat permohonan *lintas*

Nama Kasus Uji	Kasus uji melihat daftar permohonan *lintas* ketika tidak terdapat permohonan *lintas.*
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji menekan menu Permohonan *Lintas.*
Hasil yang diharapkan	Sistem akan maka akan menampilkan pesan "*No data available in table"* pada tabel.
Hasil	Sistem akan maka akan menampilkan pesan "*No data available in table"* pada tabel.
Status	Valid.

6.3.10 Kasus Uji Memberi Izin Permohonan *Lintas*

6.3.10.1 Kasus Uji Memberi Izin Permohonan *Lintas* Berhasil

Tabel 6.34 Kasus uji memberi izin permohonan *lintas* berhasil

Nama Kasus Uji	Kasus uji memberi izin permohonan *lintas* berhasil.
Prosedur	1. Penguji berada pada halaman daftar permohonan *lintas.* 2. Penguji menekan tombol izinkan pada *lintas* yang dipilih. 3. Penguji mengisi catatan pada modal izinkan *lintas.* 4. Penguji menekan tombol izinkan.
Hasil yang diharapkan	Sistem mengupdate status *lintas* dan status gangguan dan menampilkan halaman daftar permohonan *lintas.*
Hasil	Sistem mengupdate status *lintas* dan status gangguan dan menampilkan halaman daftar permohonan *lintas.*
Status	Valid.

6.3.10.2 Kasus Uji Memberi Izin Permohonan *Lintas* dengan Tombol Tidak Ditekan

Tabel 6.35 Kasus uji memberi izin permohonan *lintas* dengan tombol tidak ditekan

Nama Kasus Uji	Kasus uji memberi izin permohonan *lintas* dengan tombol tidak ditekan.
Prosedur	1. Penguji berada pada halaman daftar permohonan *lintas.* 2. Penguji menekan tombol izinkan pada *lintas* yang dipilih. 3. Penguji mengisi catatan pada modal izinkan *lintas.* 4. Penguji menekan tombol batal.
Hasil yang diharapkan	Sistem menutup modal izinkan *lintas.*
Hasil	Sistem menutup modal izinkan *lintas.*
Status	Valid.

6.3.11 Kasus Uji Melihat Daftar Petugas Aktif

6.3.11.1 Kasus Uji Melihat Daftar Petugas Aktif Berhasil

Tabel 6.36 Kasus uji melihat daftar petugas aktif berhasil

Nama Kasus Uji	Kasus uji melihat daftar petugas aktif berhasil.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu pelaporan. 3. Penguji memilih menu daftar petugas aktif.
Hasil yang diharapkan	Sistem akan menampilkan halaman lihat petugas aktif yang berisi data pegawai yang sedang aktif dengan data nama lengkap, nipp, *email* dan nomor telepon.
Hasil	Sistem akan menampilkan halaman lihat petugas aktif yang berisi data pegawai yang sedang aktif dengan data nama lengkap, nipp, *email* dan nomor telepon.
Status	Valid.

6.3.12 Kasus Uji Membuat Laporan Gangguan

6.3.12.1 Kasus Uji Membuat Laporan Gangguan Berhasil

Tabel 6.37 Kasus uji membuat laporan gangguan berhasil

Nama Kasus Uji	Kasus uji membuat laporan gangguan berhasil.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu pelaporan. 3. Penguji memilih menu lapor gangguan. 4. Penguji mengisi *form* buat laporan gangguan dengan lengkap. 5. Aktor menekan tombol laporkan gangguan.
Hasil yang diharapkan	Sistem menyimpan laporan gangguan pada *database,* menampilkan *detail* laporan gangguan dan memberikan notifikasi kepada petugas aktif dan menampilkan halaman laporan gangguan berhasil dilaporkan.
Hasil	Sistem menyimpan laporan gangguan pada *database,* menampilkan *detail* laporan gangguan dan memberikan notifikasi kepada petugas aktif dan menampilkan halaman laporan gangguan berhasil dilaporkan.
Status	Valid.

6.3.12.2 Kasus Uji Membuat Laporan Gangguan dengan Tidak Mengisi Lengkap *Form*

Tabel 6.38 Kasus uji membuat laporan gangguan dengan tidak mengisi lengkap *form*

Nama Kasus Uji	Kasus uji membuat laporan gangguan dengan tidak mengisi lengkap *form.*
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu pelaporan. 3. Penguji memilih menu lapor gangguan. 4. Penguji tidak mengisi *form* buat laporan gangguan dengan lengkap. 5. Aktor menekan tombol laporkan gangguan.
Hasil yang diharapkan	Sistem akan menampilkan pesan "*Please fill out this field".*

Hasil	Sistem akan menampilkan pesan "*Please fill out this field".*
Status	Valid.

6.3.13 Kasus Uji Mengunduh *E-Ticket* Pelaporan Gangguan

6.3.13.1 Kasus Uji Mengunduh *E-Ticket* Pelaporan Gangguan

Tabel 6.39 Kasus uji mengunduh *e-ticket* pelaporan gangguan

Nama Kasus Uji	Kasus uji mengunduh *e-ticket* pelaporan gangguan.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu pelaporan. 3. Penguji memilih menu daftar laporan gangguan. 4. Penguji memilih laporan gangguan yang ingin diunduh *e-ticketnya* dengan menekan tombol "Unduh *E-Ticket*".
Hasil yang diharapkan	*E-Ticket* akan terunduh ke perangkat penguji.
Hasil	*E-Ticket* akan terunduh ke perangkat penguji.
Status	Valid.

6.3.14 Kasus Uji Melihat Daftar Laporan Gangguan Yang Telah Terlaporkan

6.3.14.1 Kasus Uji Melihat Daftar Laporan Gangguan yang Telah Terlaporkan Berhasil

Tabel 6.40 Kasus uji melihat daftar laporan gangguan yang telah terlaporkan berhasil

Nama Kasus Uji	Kasus uji melihat daftar laporan gangguan yang telah terlaporkan berhasil.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu pelaporan. 3. Penguji memilih menu daftar laporan gangguan.
Hasil yang diharapkan	Sistem akan menampilkan halaman daftar laporan gangguan terlaporkan yang terdapat tabel yang berisi nomor, kode gangguan, NIPP petugas, jenis gangguan, lokasi gangguan, waktu lapor, status, tombol unduh *e-ticket*, tombol lihat *detail*, dan tombol batalkan dari setiap laporan gangguan.

Hasil	Sistem akan menampilkan halaman daftar laporan gangguan terlaporkan yang terdapat tabel yang berisi nomor, kode gangguan, NIPP petugas, jenis gangguan, lokasi gangguan, waktu lapor, status, tombol unduh *e-ticket*, tombol lihat *detail*, dan tombol batalkan dari setiap laporan gangguan.
Status	Valid.

6.3.14.2 Kasus Uji Melihat Daftar Laporan Gangguan yang Telah Terlaporkan Dengan Tidak Ada Data Yang Ditampilkan

Tabel 6.41 Kasus uji melihat daftar laporan gangguan yang telah terlaporkan dengan tidak ada data yang ditampilkan

Nama Kasus Uji	Kasus uji melihat daftar laporan gangguan yang telah terlaporkan dengan tidak ada data yang ditampilkan.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu pelaporan. 3. Penguji memilih menu daftar laporan gangguan.
Hasil yang diharapkan	Sistem akan menampilkan halaman lihat daftar laporan gangguan yang terlaporkan dengan tabel yang berisi "*No data available in table*".
Hasil	Sistem akan menampilkan halaman lihat daftar laporan gangguan yang terlaporkan dengan tabel yang berisi "*No data available in table*".
Status	Valid.

6.3.15 Kasus Uji Melihat *Detail* Laporan Gangguan yang Telah Terlaporkan

6.3.15.1 Kasus Uji Melihat *Detail* Laporan Gangguan yang Telah Terlaporkan

Tabel 6.42 Kasus uji melihat *detail* laporan gangguan yang telah terlaporkan

Nama Kasus Uji	Kasus uji melihat *detail* laporan gangguan yang telah terlaporkan.
Prosedur	1. Penguji berada pada halaman daftar laporan gangguan terlaporkan. 2. Penguji memilih laporan gangguan yang ingin dilihat *detail*nya.
Hasil yang diharapkan	Sistem akan menampilkan halaman *detail* laporan gangguan yang terdapat bagian *detail* laporan

	gangguan yang menampilkan NIPP pelapor, NIPP Petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, solusi, bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan dan bagian pendataan barang *lintas* yang memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan.
Hasil	Sistem akan menampilkan halaman *detail* laporan gangguan yang terdapat bagian *detail* laporan gangguan yang menampilkan NIPP pelapor, NIPP Petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, solusi, bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan dan bagian pendataan barang *lintas* yang memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan.
Status	Valid.

6.3.16 Kasus Uji Membatalkan Laporan Gangguan

6.3.16.1 Kasus Uji Membatalkan Laporan Gangguan Berhasil

Tabel 6.43 Kasus uji membatalkan laporan gangguan berhasil

Nama Kasus Uji	Kasus uji membatalkan laporan gangguan berhasil.
Prosedur	1. Penguji berada pada halaman daftar laporan gangguan terlaporkan. 2. Penguji memilih laporan gangguan yang ingin dibatalkan dengan menekan tombol batalkan. 3. Penguji menekan tombol Ya.
Hasil yang diharapkan	Sistem akan meng*update* status laporan gangguan menjadi dibatalkan dan akan menampilkan halaman daftar laporan gangguan terlaporkan.
Hasil	Sistem akan meng*update* status laporan gangguan menjadi dibatalkan dan akan menampilkan halaman daftar laporan gangguan terlaporkan.
Status	Valid.

6.3.16.2 Kasus Uji Membatalkan Laporan Gangguan Dibatalkan

Tabel 6.44 Kasus uji membatalkan laporan gangguan dibatalkan

Nama Kasus Uji	Kasus uji membatalkan laporan gangguan dibatalkan.
Prosedur	1. Penguji berada pada halaman daftar laporan gangguan terlaporkan. 2. Penguji memilih laporan gangguan yang ingin dibatalkan dengan menekan tombol batalkan. 3. Penguji menekan tombol Tidak.
Hasil yang diharapkan	Sistem akan menutup modal batalkan laporan gangguan.
Hasil	Sistem akan menutup modal batalkan laporan gangguan.
Status	Valid.

6.3.17 Kasus Uji Mengirim Pesan

6.3.17.1 Kasus Uji Mengirim Pesan

Tabel 6.45 Kasus uji mengirim pesan

Nama Kasus Uji	Kasus uji mengirim pesan.
Prosedur	1. Penguji berada pada halaman *detail* laporan gangguan. 2. Penguji memasukkan isi pesan. 3. Penguji menekan tombol kirim.
Hasil yang diharapkan	Sistem akan menyimpan pesan dalam *database*, menampilkannya dalam bagian pesan dan memberikan notifikasi kepada penerima.
Hasil	Sistem akan menyimpan pesan dalam *database*, menampilkannya dalam bagian pesan dan memberikan notifikasi kepada penerima.
Status	Valid.

6.3.18 Kasus Uji Melihat Daftar Laporan Gangguan Masuk

6.3.18.1 Kasus Uji Melihat Daftar Laporan Gangguan Masuk Ketika Ada Laporan Gangguan Masuk

Tabel 6.46 Kasus uji melihat daftar laporan gangguan masuk ketika ada laporan gangguan masuk

Nama Kasus Uji	Kasus uji melihat daftar laporan gangguan masuk ketika ada laporan gangguan masuk.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu penanganan. 3. Penguji memilih submenu laporan gangguan masuk.
Hasil yang diharapkan	Sistem akan menampilkan halaman daftar laporan gangguan masuk yang menampilkan menampilkan nomor, kode gangguan, NIPP pelapor, jenis gangguan, deskripsi, lokasi gangguan, waktu lapor, dan tombol "tangani" dari setiap laporan gangguan.
Hasil	Sistem akan menampilkan halaman daftar laporan gangguan masuk yang menampilkan menampilkan nomor, kode gangguan, NIPP pelapor, jenis gangguan, deskripsi, lokasi gangguan, waktu lapor, dan tombol "tangani" dari setiap laporan gangguan.
Status	Valid.

6.3.18.2 Kasus Uji Melihat Daftar Laporan Gangguan Masuk Ketika Tidak Ada Laporan Gangguan Masuk

Tabel 6.47 Kasus uji melihat daftar laporan gangguan masuk ketika tidak ada laporan gangguan masuk

Nama Kasus Uji	Kasus uji melihat daftar laporan gangguan masuk ketika tidak ada laporan gangguan masuk.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu penanganan. 3. Penguji memilih submenu laporan gangguan masuk.
Hasil yang diharapkan	Jika tidak ada laporan gangguan yang masuk atau tidak ada data yang dapat ditampilkan maka sistem akan menampilkan "*No data available in tabel*" pada tabel.

Hasil	Jika tidak ada laporan gangguan yang masuk atau tidak ada data yang dapat ditampilkan maka sistem akan menampilkan "*No data available in tabel*" pada tabel.
Status	Valid.

6.3.19 Kasus Uji Menangani Laporan Gangguan

6.3.19.1 Kasus Uji Menangani Laporan Gangguan Berhasil

Tabel 6.48 Kasus uji menangani laporan gangguan berhasil

Nama Kasus Uji	Kasus uji menangani laporan gangguan berhasil.
Prosedur	1. Penguji berada pada halaman daftar laporan gangguan masuk. 2. Penguji memilih laporan yang ingin ditangani dengan menekan tombol tangani. 3. Penguji memilih tombol Ya pada modal tangani laporan gangguan.
Hasil yang diharapkan	Sistem akan meng*update* status laporan gangguan menjadi sedang ditangani, memberikan notifikasi kepada pelapor dan akan menampilkan halaman daftar laporan gangguan ditangani.
Hasil	Sistem akan meng*update* status laporan gangguan menjadi sedang ditangani, memberikan notifikasi kepada pelapor dan akan menampilkan halaman daftar laporan gangguan ditangani.
Status	Valid.

6.3.19.2 Kasus Uji Menangani Laporan Gangguan Ketika Tombol Tidak Ditekan

Tabel 6.49 Kasus uji menangani laporan gangguan ketika tombol tidak ditekan

Nama Kasus Uji	Kasus uji menangani laporan gangguan ketika tombol tidak ditekan.
Prosedur	1. Penguji berada pada halaman daftar laporan gangguan masuk. 2. Penguji memilih laporan yang ingin ditangani dengan menekan tombol tangani. 3. Penguji memilih tombol Tidak pada modal tangani laporan gangguan.
Hasil yang diharapkan	Sistem akan menutup modal tangani laporan gangguan.

Hasil	Sistem akan menutup modal tangani laporan gangguan.
Status	Valid.

6.3.20 Kasus Uji Melihat Daftar Laporan Gangguan yang Ditangani

6.3.20.1 Kasus Uji Melihat Daftar Laporan Gangguan yang Ditangani Ketika Ada Laporan Gangguan yang Ditangani

Tabel 6.50 Kasus uji melihat daftar laporan gangguan yang ditangani ketika ada laporan gangguan yang ditangani

Nama Kasus Uji	Kasus uji melihat daftar laporan gangguan yang ditangani ketika ada laporan gangguan yang ditangani.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu penanganan. 3. Penguji memilih submenu laporan gangguan ditangani.
Hasil yang diharapkan	Sistem akan menampilkan halaman daftar laporan gangguan ditangani yang menampilkan tabel dengan kolom nomor, kode gangguan, NIPP pelapor, jenis gangguan, lokasi gangguan, waktu lapor, status dan tombol "*detail*" dari setiap laporan gangguan.
Hasil	Sistem akan menampilkan halaman daftar laporan gangguan ditangani yang menampilkan tabel dengan kolom nomor, kode gangguan, NIPP pelapor, jenis gangguan, lokasi gangguan, waktu lapor, status dan tombol "*detail*" dari setiap laporan gangguan.
Status	Valid.

6.3.20.2 Kasus Uji Melihat Daftar Laporan Gangguan Yang Ditangani Ketika Tidak Ada Laporan Gangguan Yang Ditangani

Tabel 6.51 Kasus uji melihat daftar laporan gangguan yang ditangani ketika tidak ada laporan gangguan yang ditangani

Nama Kasus Uji	Kasus uji melihat daftar laporan gangguan yang ditangani ketika tidak ada laporan gangguan yang ditangani.
Prosedur	1. Penguji berada pada halaman utama.

	2. Penguji memilih menu penanganan. 3. Penguji memilih submenu laporan gangguan ditangani.
Hasil yang diharapkan	Sistem akan menampilkan "*No data available in tabel*" pada tabel.
Hasil	Sistem akan menampilkan "*No data available in tabel*" pada tabel.
Status	Valid.

6.3.21 Kasus Uji Melihat *Detail* Laporan Gangguan yang Ditangani

6.3.21.1 Kasus Uji Melihat *Detail* Laporan Gangguan yang Ditangani

Tabel 6.52 Kasus uji melihat *detail* laporan gangguan yang ditangani

Nama Kasus Uji	Kasus uji melihat *detail* laporan gangguan yang ditangani.
Prosedur	1. Penguji berada pada halaman daftar laporan gangguan ditangani. 2. Penguji memilih laporan gangguan yang akan dilihat *detail*nya dengan menekan tombol *detail* pada data riwayat yang ingin dilihat *detail*nya.
Hasil yang diharapkan	Sistem akan menampilkan halaman *detail* laporan gangguan yang terdapat bagian *detail* laporan gangguan yang menampilkan NIPP pelapor, NIPP Petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, solusi, tombol "Ajukan Lintas" dan tombol "Tutup Laporan", bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan dan bagian pendataan barang *lintas* yang memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan.
Hasil	Sistem akan menampilkan halaman *detail* laporan gangguan yang terdapat bagian *detail* laporan gangguan yang menampilkan NIPP pelapor, NIPP Petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, solusi, tombol "Ajukan Lintas" dan tombol "Tutup Laporan", bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan

	petugas yang menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan dan bagian pendataan barang *lintas* yang memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan.
Status	Valid.

6.3.22 Kasus Uji Mengajukan *Lintas*

6.3.22.1 Kasus Uji Mengajukan *Lintas* Berhasil

Tabel 6.53 Kasus uji mengajukan *lintas* berhasil

Nama Kasus Uji	Kasus uji mengajukan *lintas* berhasil.
Prosedur	1. Penguji berada pada halaman *detail* dari sebuah laporan gangguan. 2. Penguji menekan tombol ajukan *lintas.* 3. Penguji menekan tombol Ya.
Hasil yang diharapkan	Sistem akan menyimpan permohonan *lintas*, meng*update* status dari laporan gangguan menjadi mengajukan *lintas*, mengirimkan notifikasi kepada pimpinan Unit IT dan menampilkan kembali halaman *detail* dari laporan tersebut.
Hasil	Sistem akan menyimpan permohonan *lintas*, meng*update* status dari laporan gangguan menjadi mengajukan *lintas*, mengirimkan notifikasi kepada pimpinan Unit IT dan menampilkan kembali halaman *detail* dari laporan tersebut.
Status	Valid.

6.3.22.2 Kasus Uji Mengajukan *Lintas* Ketika Tombol Tidak Ditekan

Tabel 6.54 Kasus uji mengajukan *lintas* ketika tombol tidak ditekan

Nama Kasus Uji	Kasus uji mengajukan *lintas* ketika tombol tidak ditekan.
Prosedur	1. Penguji berada pada halaman *detail* dari sebuah laporan gangguan. 2. Penguji menekan tombol ajukan *lintas.* 3. Penguji menekan tombol Tidak.
Hasil yang diharapkan	Sistem akan menutup modal ajukan *lintas.*
Hasil	Sistem akan menutup modal ajukan *lintas.*

Status	Valid.

6.3.23 Kasus Uji Menutup Laporan

6.3.23.1 Kasus Uji Menutup Laporan dengan Status Terselesaikan

Tabel 6.55 Kasus uji menutup laporan dengan status terselesaikan

Nama Kasus Uji	Kasus uji menutup laporan dengan status terselesaikan.
Prosedur	1. Penguji berada pada halaman *detail* dari sebuah laporan gangguan. 2. Penguji menekan tombol tutup laporan. 3. Penguji memilih pilihan terselesaikan. 4. Penguji mengisi *input*an solusi. 5. Penguji menekan tombol simpan.
Hasil yang diharapkan	Sistem menyimpan solusi, meng*update* status gangguan menjadi terselesaikan dan menampilkan halaman lihat daftar laporan gangguan ditangani.
Hasil	Sistem menyimpan solusi, meng*update* status gangguan menjadi terselesaikan dan menampilkan halaman lihat daftar laporan gangguan ditangani.
Status	Valid.

6.3.23.2 Kasus Uji Menutup Laporan dengan Status Tidak Terselesaikan

Tabel 6.56 Kasus uji menutup laporan dengan status tidak terselesaikan

Nama Kasus Uji	Kasus uji menutup laporan dengan status tidak terselesaikan.
Prosedur	1. Penguji berada pada halaman *detail* dari sebuah laporan gangguan. 2. Penguji menekan tombol tutup laporan. 3. Penguji memilih pilihan tidak terselesaikan. 4. Penguji menekan tombol simpan.
Hasil yang diharapkan	Sistem meng*update* status gangguan menjadi tidak terselesaikan dan menampilkan halaman lihat daftar laporan gangguan ditangani.
Hasil	Sistem meng*update* status gangguan menjadi tidak terselesaikan dan menampilkan halaman lihat daftar laporan gangguan ditangani.

Status	Valid.

6.3.23.3 Kasus Uji Menutup Laporan dengan Tidak Memilih Status Laporan Gangguan

Tabel 6.57 Kasus uji menutup laporan dengan tidak memilih status laporan gangguan

Nama Kasus Uji	Kasus uji menutup laporan dengan tidak memilih status laporan gangguan.
Prosedur	1. Penguji berada pada halaman *detail* dari sebuah laporan gangguan. 2. Penguji menekan tombol tutup laporan. 3. Penguji tidak memilih pilihan status laporan gangguan. 4. Penguji menekan tombol simpan.
Hasil yang diharapkan	Sistem akan menampilkan pesan *"Please select an item in the list".*
Hasil	Sistem akan menampilkan pesan *"Please select an item in the list".*
Status	Valid.

6.3.24 Kasus Uji Melihat Daftar *Lintas*

6.3.24.1 Kasus Uji Melihat Daftar *Lintas* Ketika Terdapat *Lintas*

Tabel 6.58 Kasus uji melihat daftar *lintas* ketika terdapat *lintas*

Nama Kasus Uji	Kasus uji melihat daftar *lintas* ketika terdapat *lintas.*
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu penanganan. 3. Penguji memilih submenu daftar *lintas.*
Hasil yang diharapkan	Sistem menampilkan halaman daftar *lintas* yang menampilkan tabel dengan kolom nomor, kode gangguan, NIPP pelapor, lokasi gangguan, jenis gangguan, status *lintas* dan catatan.
Hasil	Sistem menampilkan halaman daftar *lintas* yang menampilkan tabel dengan kolom nomor, kode gangguan, NIPP pelapor, lokasi gangguan, jenis gangguan, status *lintas* dan catatan.
Status	Valid.

6.3.24.2 Kasus Uji Melihat Daftar *Lintas* Ketika Tidak Terdapat *Lintas*

Tabel 6.59 Kasus uji melihat daftar *lintas* ketika tidak terdapat *lintas*

Nama Kasus Uji	Kasus uji melihat daftar *lintas* ketika tidak terdapat *lintas.*
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu penanganan. 3. Penguji memilih submenu daftar *lintas.*
Hasil yang diharapkan	Sistem menampilkan "*No data available in tabel*" pada tabel.
Hasil	Sistem menampilkan "*No data available in tabel*" pada tabel.
Status	Valid.

6.3.25 Kasus Uji Menambahkan Barang yang Telah Digunakan

6.3.25.1 Kasus Uji Menambahkan Barang yang Telah Digunakan Berhasil

Tabel 6.60 Kasus uji menambahkan barang yang telah digunakan berhasil

Nama Kasus Uji	Kasus uji menambahkan barang yang telah digunakan berhasil.
Prosedur	1. Penguji berada pada halaman *detail* dari sebuah laporan gangguan. 2. Penguji menekan tombol tambah barang. 3. Penguji memilih barang-barang yang ingin dipilih. 4. Penguji mengisi dengan lengkap *form* tambah barang *lintas*. 5. Penguju menekan tombol Simpan.
Hasil yang diharapkan	Sistem menyimpan daftar barang ke dalam *database* dan menampilkan kembali halaman *detail* dari laporan gangguan tersebut.
Hasil	Sistem menyimpan daftar barang ke dalam *database* dan menampilkan kembali halaman *detail* dari laporan gangguan tersebut.
Status	Valid.

6.3.25.2 Kasus Uji Menambahkan Barang yang Telah Digunakan yang Sudah Ada *Form* Tambah Barang *Lintas*

Tabel 6.61 Kasus uji menambahkan barang yang telah digunakan yang sudah ada *form* tambah barang *lintas*

Nama Kasus Uji	Kasus uji menambahkan barang yang telah digunakan yang sudah ada *form* tambah barang *lintas.*
Prosedur	1. Penguji berada pada halaman *detail* dari sebuah laporan gangguan. 2. Penguji menekan tombol tambah barang. 3. Penguji memilih barang-barang yang sudah terdaftar pada *form* tambah barang.
Hasil yang diharapkan	Sistem akan menampilkan pesan "Perhatian! Nama barang sudah ada di dalam daftar."
Hasil	Sistem akan menampilkan pesan "Perhatian! Nama barang sudah ada di dalam daftar."
Status	Valid.

6.3.25.3 Kasus Uji Menambahkan Barang yang Telah Digunakan dengan Mengisi Jumlah Barang Lebih Dari Jumlah Tersedia

Tabel 6.62 Kasus uji menambahkan barang yang telah digunakan dengan mengisi jumlah barang lebih dari jumlah tersedia

Nama Kasus Uji	Kasus uji menambahkan barang yang telah digunakan dengan mengisi jumlah barang lebih dari jumlah tersedia.
Prosedur	1. Penguji berada pada halaman *detail* dari sebuah laporan gangguan. 2. Penguji menekan tombol tambah barang. 3. Penguji memilih barang-barang yang sudah terdaftar pada *form* tambah barang. 4. Penguji mengisi *form* tambah barang *lintas* dengan mengisi jumlah barang lebih dari jumlah tersedia.
Hasil yang diharapkan	Sistem akan menampilkan "Perhatian! Anda tidak bisa melakukan pengambilan barang dengan jumlah LEBIH BESAR dari ketersediaan barang. Jumlah barang tersedia adalah (jumlah tersedia)".

Hasil	Sistem akan menampilkan "Perhatian! Anda tidak bisa melakukan pengambilan barang dengan jumlah LEBIH BESAR dari ketersediaan barang. Jumlah barang tersedia adalah (jumlah tersedia)".
Status	Valid.

6.3.25.4 Kasus Uji Menambahkan Barang yang Telah Digunakan Dengan Tidak Mengisi Lengkap *Form* Tambah Barang *Lintas*

Tabel 6.63 Kasus uji menambahkan barang yang telah digunakan dengan tidak mengisi lengkap *form* tambah barang *lintas*

Nama Kasus Uji	Kasus uji menambahkan barang yang telah digunakan dengan tidak mengisi lengkap *form* tambah barang *lintas.*
Prosedur	1. Penguji berada pada halaman *detail* dari sebuah laporan gangguan. 2. Penguji menekan tombol tambah barang. 3. Penguji memilih barang-barang yang sudah terdaftar pada *form* tambah barang. 4. Penguji mengisi *form* tambah barang *lintas* dengan tidak lengkap. 5. Penguji menekan tombol simpan.
Hasil yang diharapkan	Sistem akan menampilkan pesan *"Please fill out this field".*
Hasil	Sistem akan menampilkan pesan *"Please fill out this field".*
Status	Valid.

6.3.26 Kasus Uji Melihat Daftar Barang di Gudang Penyimpanan

6.3.26.1 Kasus Uji Melihat Daftar Barang di Gudang Penyimpanan Ketika Ada Barang di Gudang Penyimpan

Tabel 6.64 Kasus uji melihat daftar barang di gudang penyimpanan ketika ada barang di gudang penyimpanan

Nama Kasus Uji	Kasus uji melihat daftar barang di gudang penyimpanan ketika ada barang di gudang penyimpanan.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu Manajemen Gudang.

Hasil yang diharapkan	Sistem menampilkan halaman manajemen gudang penyimpanan yang didalamnya terdapat tabel yang memiliki kolom nomor nama barang, kuantitas, tombol *update* dan tombol hapus.
Hasil	Sistem menampilkan halaman manajemen gudang penyimpanan yang didalamnya terdapat tabel yang memiliki kolom nomor nama barang, kuantitas, tombol *update* dan tombol hapus.
Status	Valid.

6.3.26.2 Kasus Uji Melihat Daftar Barang di Gudang Penyimpanan Ketika Tidak Ada Barang Di Gudang Penyimpanan

Tabel 6.65 Kasus uji melihat daftar barang di gudang penyimpanan ketika tidak ada barang di gudang penyimpanan

Nama Kasus Uji	Kasus uji melihat daftar barang di gudang penyimpanan ketika tidak ada barang di gudang penyimpanan.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu Manajemen Gudang.
Hasil yang diharapkan	Sistem menampilkan menampilkan "*No data available in tabel*" pada tabel.
Hasil	Sistem menampilkan menampilkan "*No data available in tabel*" pada tabel.
Status	Valid.

6.3.27 Kasus Uji Menambah Barang di Daftar Barang Gudang Penyimpanan

6.3.27.1 Kasus Uji Menambah Barang di Daftar Barang Gudang Penyimpanan Berhasil

Tabel 6.66 Kasus uji menambah barang di daftar barang gudang penyimpanan berhasil

Nama Kasus Uji	Kasus uji menambah barang di daftar barang gudang penyimpanan berhasil.
Prosedur	1. Penguji berada pada halaman manajemen gudang penyimpanan. 2. Penguji menekan tombol tambah barang. 3. Penguji mengisi *form* tambah barang dengan lengkap.

	4. Penguji menekan tombol tambah.
Hasil yang diharapkan	Sistem akan menyimpan barang pada *database*, menampilkan pesan "Berhasil menambahkan barang" dan menampilkan halaman daftar barang gudang penyimpanan.
Hasil	Sistem akan menyimpan barang pada *database*, menampilkan pesan "Berhasil menambahkan barang" dan menampilkan halaman daftar barang gudang penyimpanan.
Status	Valid.

6.3.27.2 Kasus Uji Menambah Barang di Daftar Barang Gudang Penyimpanan Dengan Tidak Mengisi *Form* Tambah Barang Dengan Lengkap

Tabel 6.67 Kasus uji menambah barang di daftar barang gudang penyimpanan dengan tidak mengisi *form* tambah barang dengan lengkap

Nama Kasus Uji	Kasus uji menambah barang di daftar barang gudang penyimpanan dengan tidak mengisi *form* tambah barang dengan lengkap.
Prosedur	1. Penguji berada pada halaman manajemen gudang penyimpanan. 2. Penguji menekan tombol tambah barang. 3. Penguji mengisi *form* tambah barang dengan tidak lengkap. 4. Penguji menekan tombol tambah.
Hasil yang diharapkan	Sistem akan menampilkan pesan "*Please fill out this field*".
Hasil	Sistem akan menampilkan pesan "*Please fill out this field*".
Status	Valid.

6.3.27.3 Kasus Uji Menambah Barang di Daftar Barang Gudang Penyimpanan Ketika Tombol Tidak Ditekan

Tabel 6.68 Kasus uji menambah barang di daftar barang gudang penyimpanan ketika tombol tidak ditekan

Nama Kasus Uji	Kasus uji menambah barang di daftar barang gudang penyimpanan ketika tombol tidak ditekan.
Prosedur	1. Penguji berada pada halaman manajemen gudang penyimpanan.

	2. Penguji menekan tombol tambah barang. 3. Penguji mengisi *form* tambah barang dengan lengkap. 4. Penguji menekan tombol Tidak.
Hasil yang diharapkan	Sistem akan menutup modal tambah barang.
Hasil	Sistem akan menutup modal tambah barang.
Status	Valid.

6.3.28 Kasus Uji Meng*update* Barang di Daftar Barang Gudang Penyimpanan

6.3.28.1 Kasus Uji Meng*update* Barang di Daftar Barang Gudang Penyimpanan Berhasil

Tabel 6.69 Kasus uji meng*update* barang di daftar barang gudang penyimpanan berhasil

Nama Kasus Uji	Kasus uji meng*update* barang di daftar barang gudang penyimpanan berhasil.
Prosedur	1. Penguji berada pada halaman manajemen gudang penyimpanan. 2. Penguji memilih barang yang akan di*update* dengan menekan tombol edit pada baris data tersebut. 3. Penguji mengisikan *field* yang akan diedit. 4. Penguji menekan *update.*
Hasil yang diharapkan	Sistem akan meng*update* data barang pada *database*, menampilkan pesan "Berhasil meng*update* barang" dan menampilkan halaman manajemen gudang penyimpanan.
Hasil	Sistem akan meng*update* data barang pada *database*, menampilkan pesan "Berhasil meng*update* barang" dan menampilkan halaman manajemen gudang penyimpanan.
Status	Valid.

6.3.28.2 Kasus Uji Meng*update* Barang di Daftar Barang Gudang Penyimpanan Dengan Mengosongkan *Input*an

Tabel 6.70 Kasus uji meng*update* barang di daftar barang gudang penyimpanan dengan mengosongkan *input*an

Nama Kasus Uji	Kasus uji meng*update* barang di daftar barang gudang penyimpanan dengan mengosongkan *input*an.
Prosedur	1. Penguji berada pada halaman manajemen gudang penyimpanan. 2. Penguji memilih barang yang akan di*update* dengan menekan tombol edit pada baris data tersebut. 3. Penguji mengosongkan *input*an. 4. Aktor menekan *update.*
Hasil yang diharapkan	Sistem akan menampilkan pesan "*Please fill out this field*".
Hasil	Sistem akan menampilkan pesan "*Please fill out this field*".
Status	Valid.

6.3.28.3 Kasus Uji Meng*update* Barang di Daftar Barang Gudang Penyimpanan Ketika Tombol Batal Ditekan

Tabel 6.71 Kasus uji meng*update* barang di daftar barang gudang penyimpanan ketika tombol batal ditekan

Nama Kasus Uji	Kasus uji meng*update* barang di daftar barang gudang penyimpanan ketika tombol batal ditekan.
Prosedur	1. Penguji berada pada halaman manajemen gudang penyimpanan. 2. Penguji memilih barang yang akan di*update* dengan menekan tombol edit pada baris data tersebut. 3. Penguji mengisikan *field* yang akan diedit. 4. Penguji menekan tombol Tidak.
Hasil yang diharapkan	Sistem akan menutup modal edit barang.
Hasil	Sistem akan menutup modal edit barang.
Status	Valid.

6.3.29 Kasus Uji Menghapus Barang di Daftar Barang Gudang Penyimpanan

6.3.29.1 Kasus Uji Menghapus Barang di Daftar Barang Gudang Penyimpanan Berhasil

Tabel 6.72 Kasus uji menghapus barang di daftar barang gudang penyimpanan berhasil

Nama Kasus Uji	Kasus uji menghapus barang di daftar barang gudang penyimpanan berhasil.
Prosedur	1. Penguji berada pada halaman manajemen gudang penyimpanan. 2. Penguji memilih barang yang akan dihapus dengan menekan tombol hapus pada baris data tersebut. 3. Penguji menekan tombol hapus.
Hasil yang diharapkan	Sistem akan menghapus data barang dari *database*, menampilkan pesan "Berhasil menghapus barang" dan menampilkan halaman manajemen gudang penyimpanan.
Hasil	Sistem akan menghapus data barang dari *database*, menampilkan pesan "Berhasil menghapus barang" dan menampilkan halaman manajemen gudang penyimpanan.
Status	Valid.

6.3.29.2 Kasus Uji Menghapus Barang Di Daftar Barang Gudang Penyimpanan Ketika Menekan Tombol Tidak

Tabel 6.73 Kasus uji menghapus barang di daftar barang gudang penyimpanan berhasil

Nama Kasus Uji	Kasus uji menghapus barang di daftar barang gudang penyimpanan berhasil.
Prosedur	1. Penguji berada pada halaman manajemen gudang penyimpanan. 2. Penguji memilih barang yang akan dihapus dengan menekan tombol hapus pada baris data tersebut. 3. Penguji menekan tombol Tidak.
Hasil yang diharapkan	Sistem akan menutup modal hapus barang.

Hasil	Sistem akan menutup modal hapus barang.
Status	Valid.

6.3.30 Kasus Uji Melihat Daftar Riwayat Laporan Gangguan

6.3.30.1 Kasus Uji Melihat Daftar Riwayat Laporan Gangguan Ketika Terdapat Riwayat Laporan Gangguan

Tabel 6.74 Kasus uji melihat daftar riwayat laporan gangguan ketika terdapat riwayat laporan gangguan

Nama Kasus Uji	Kasus uji melihat daftar riwayat laporan gangguan ketika terdapat riwayat laporan gangguan.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu riwayat laporan gangguan. 3. Penguji memilih menu daftar riwayat.
Hasil yang diharapkan	Sistem akan menampilkan halaman daftar riwayat laporan gangguan yang terdiri dari *form filter* yang terdiri dari *input*an tanggal awal dan tanggal akhir serta tabel yang berisi kolom nomor, kode gangguan, NIPP pelapor, NIPP petugas, jenis gangguan, lokasi, status dan tombol *detail.*
Hasil	Sistem akan menampilkan halaman daftar riwayat laporan gangguan yang terdiri dari *form filter* yang terdiri dari *input*an tanggal awal dan tanggal akhir serta tabel yang berisi kolom nomor, kode gangguan, NIPP pelapor, NIPP petugas, jenis gangguan, lokasi, status dan tombol *detail.*
Status	Valid.

6.3.30.2 Kasus Uji Melihat Daftar Riwayat Laporan Gangguan Ketika Tidak Terdapat Riwayat Laporan Gangguan

Tabel 6.75 Kasus uji melihat daftar riwayat laporan gangguan ketika terdapat tidak riwayat laporan gangguan

Nama Kasus Uji	Kasus uji melihat daftar riwayat laporan gangguan ketika terdapat tidak riwayat laporan gangguan.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu riwayat laporan gangguan. 3. Penguji memilih menu daftar riwayat.

Hasil yang diharapkan	Sistem akan menampilkan "*No data available in tabel*" pada tabel.
Hasil	Sistem akan menampilkan "*No data available in tabel*" pada tabel.
Status	Valid.

6.3.31 Kasus Uji Mengekspor Daftar Riwayat Laporan Gangguan

6.3.31.1 Kasus Uji Mengekspor Daftar Riwayat Laporan Gangguan

Tabel 6.76 Kasus uji melihat daftar riwayat laporan gangguan ketika terdapat tidak riwayat laporan gangguan

Nama Kasus Uji	Kasus uji melihat daftar riwayat laporan gangguan ketika terdapat tidak riwayat laporan gangguan.
Prosedur	1. Penguji berada pada halaman daftar riwayat laporan gangguan. 2. Penguji menekan tombol *Export to Excel.*
Hasil yang diharapkan	Komputer akan mengunduh *file* hasil ekspor.
Hasil	Komputer akan mengunduh *file* hasil ekspor.
Status	Valid.

6.3.32 Kasus Uji Melihat *Detail* Riwayat Laporan Gangguan

6.3.32.1 Kasus Uji Melihat *Detail* Riwayat Laporan Gangguan

Tabel 6.77 Kasus uji melihat *detail* riwayat laporan gangguan

Nama Kasus Uji	Kasus uji melihat *detail* riwayat laporan gangguan.
Prosedur	1. Penguji berada pada halaman daftar riwayat laporan gangguan. 2. Penguji memilih riwayat laporan gangguan yang ingin dilihat *detail*nya dengan menekan tombol *detail.*
Hasil yang diharapkan	Sistem akan menampilkan halaman *detail* laporan gangguan yang terdapat bagian *detail* laporan gangguan yang menampilkan NIPP pelapor, NIPP Petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, dan solusi, bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan dan bagian pendataan barang *lintas* yang

	memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan.
Hasil	Sistem akan menampilkan halaman *detail* laporan gangguan yang terdapat bagian *detail* laporan gangguan yang menampilkan NIPP pelapor, NIPP Petugas, waktu pelaporan, kode gangguan, jenis gangguan, deskripsi gangguan, lokasi gangguan, status, dan solusi, bagian pesan gangguan yang digunakan untuk saling berkirim pesan dengan petugas yang menangani laporan gangguan yang terdiri dari *field* isi pesan dan tombol kirim pesan dan bagian pendataan barang *lintas* yang memperlihatkan barang atau perangkat apa saja yang sudah digunakan untuk menangani gangguan.
Status	Valid.

6.3.33 Kasus Uji Melihat Profil

6.3.33.1 Kasus Uji Melihat Profil

Tabel 6.78 Kasus uji melihat profil

Nama Kasus Uji	Kasus uji melihat profil.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji menekan foto profil yang ada pada pojok kanan atas. 3. Penguji memilih profil.
Hasil yang diharapkan	Sistem menampilkan halaman profil yang menampilkan nama lengkap, NIPP, unit kerja, jabatan, kedudukan, *email*, nomor telepon aktor, tombol edit profil, tombol edit foto profil, dan tombol edit *password*.
Hasil	Sistem menampilkan halaman profil yang menampilkan nama lengkap, NIPP, unit kerja, jabatan, kedudukan, *email*, nomor telepon aktor, tombol edit profil, tombol edit foto profil, dan tombol edit *password*.
Status	Valid.

6.3.34 Kasus Uji Mengedit Profil

6.3.34.1 Kasus Uji Mengedit Profil Berhasil

Tabel 6.79 Kasus uji mengedit profil berhasil

Nama Kasus Uji	Kasus uji mengedit profil berhasil.
Prosedur	1. Penguji berada pada halaman profil. 2. Penguji menekan tombol edit profil. 3. Penguji mengedit data profil yang ingin diedit. 4. Penguji menekan tombol simpan.
Hasil yang diharapkan	Sistem meng*update* profil pengguna pada *database* dan menampilkan halaman profil.
Hasil	Sistem meng*update* profil pengguna pada *database* dan menampilkan halaman profil.
Status	Valid.

6.3.34.2 Kasus Uji Mengedit Profil Ketika Mengosongkan *Input*

Tabel 6.80 Kasus uji mengedit profil berhasil ketika mengosongkan *input*

Nama Kasus Uji	Kasus uji mengedit profil berhasil ketika mengosongkan *input.*
Prosedur	1. Penguji berada pada halaman profil. 2. Penguji menekan tombol edit profil. 3. Penguji mengosongkan *input* data profil. 4. Penguji menekan tombol simpan.
Hasil yang diharapkan	Sistem akan menampilkan pesan "*Please fill out this field*".
Hasil	Sistem akan menampilkan pesan "*Please fill out this field*".
Status	Valid.

6.3.34.3 Kasus Uji Mengedit Profil Ketika Menekan Tombol Batal

Tabel 6.81 Kasus uji mengedit profil berhasil ketika menekan tombol batal

Nama Kasus Uji	Kasus uji mengedit profil berhasil ketika menekan tombol batal.
Prosedur	1. Penguji berada pada halaman profil. 2. Penguji menekan tombol edit profil.

	3. Penguji mengosongkan *input* data profil. 4. Penguji menekan tombol batal.
Hasil yang diharapkan	Sistem akan menampilkan *section* profil dan menutup *section* edit profil.
Hasil	Sistem akan menampilkan *section* profil dan menutup *section* edit profil..
Status	Valid.

6.3.35 Kasus Uji Mengedit *Password*

6.3.35.1 Kasus Uji Mengedit *Password* Ketika *Password* Lama Tidak Sesuai Dengan *Database*

Tabel 6.82 Kasus uji mengedit *password* ketika *password* lama tidak sesuai dengan *database*

Nama Kasus Uji	Kasus uji mengedit *password* ketika *password* lama tidak sesuai dengan *database.*
Prosedur	1. Penguji berada pada halaman profil. 2. Penguji menekan tombol edit *password.* 3. Penguji mengisi *field password* lama tidak sesuai dengan *password* yang ada pada *database.*
Hasil yang diharapkan	Sistem akan menampilkan pesan "*Password* tidak sesuai".
Hasil	Sistem akan menampilkan pesan "*Password* tidak sesuai".
Status	Valid.

6.3.35.2 Kasus Uji Mengededit *Password* Ketika *Password* Baru Kurang Dari 6 Karakter

Tabel 6.83 Kasus uji mengedit *password* ketika *password* baru kurang dari 6 karakter

Nama Kasus Uji	Kasus uji mengedit *password* ketika *password* baru kurang dari 6 karakter.
Prosedur	1. Penguji berada pada halaman profil. 2. Penguji menekan tombol edit *password.* 3. Penguji mengisi *field password* lama sesuai dengan *password* yang ada pada *database.*

	4. Penguji mengisi *field password* baru kurang dari 6 karakter.
Hasil yang diharapkan	Sistem akan menampilkan pesan "*Password* harus memiliki minimal 6 karakter".
Hasil	Sistem akan menampilkan pesan "*Password* harus memiliki minimal 6 karakter".
Status	Valid.

6.3.35.3 Kasus Uji Mengedit *Password* Ketika Ulangi *Password* Tidak Sesuai Dengan *Password* Baru

Tabel 6.84 Kasus uji mengedit *password* ketika ulangi *password* tidak sesuai dengan *password* baru

Nama Kasus Uji	Kasus uji mengedit *password* ketika ulangi *password* tidak sesuai dengan *password* baru.
Prosedur	1. Penguji berada pada halaman profil. 2. Penguji menekan tombol edit *password.* 3. Penguji mengisi *field password* lama sesuai dengan *password* yang ada pada *database.* 4. Penguji mengisi *field password* baru sama dengan 6 karakter. 5. Penguji mengisi *field* ulangi *password* tidak sesuai dengan *password* baru.
Hasil yang diharapkan	Sistem akan menampilkan pesan "*Password* baru dan ulangi *password* tidak sesuai".
Hasil	Sistem akan menampilkan pesan "*Password* baru dan ulangi *password* tidak sesuai".
Status	Valid.

6.3.35.4 Kasus Uji Mengedit *Password* Ketika Tombol Batal Ditekan

Tabel 6.85 Kasus uji mengedit *password* ketika tombol batal ditekan

Nama Kasus Uji	Kasus uji mengedit *password* ketika tombol batal ditekan.
Prosedur	1. Penguji berada pada halaman profil. 2. Penguji menekan tombol edit *password.* 3. Penguji mengisi *field password* lama sesuai dengan *password* yang ada pada *database.*

	4. Penguji mengisi *field password* baru sama dengan 6 karakter. 5. Penguji mengisi *field* ulangi *password* sesuai dengan *password* baru. 6. Penguji menekan tombol batal.
Hasil yang diharapkan	Sistem akan menutup modal edit *password.*
Hasil	Sistem akan menutup modal edit *password.*
Status	Valid.

6.3.35.5 Kasus Uji Mengedit *Password* Berhasil

Tabel 6.86 Kasus uji mengedit *password* berhasil

Nama Kasus Uji	Kasus uji mengedit *password* berhasil.
Prosedur	1. Penguji berada pada halaman profil. 2. Penguji menekan tombol edit *password.* 3. Penguji mengisi *field password* lama sesuai dengan *password* yang ada pada *database.* 4. Penguji mengisi *field password* baru sama dengan 6 karakter. 5. Penguji mengisi *field* ulangi *password* sesuai dengan *password* baru. 6. Penguji menekan tombol simpan.
Hasil yang diharapkan	Sistem meng*update password* aktor pada *database* dan menampilkan halaman profil.
Hasil	Sistem meng*update password* aktor pada *database* dan menampilkan halaman profil.
Status	Valid.

6.3.36 Kasus Uji Mengedit Foto Profil

6.3.36.1 Kasus Uji Mengedit Foto Profil Berhasil

Tabel 6.87 Kasus uji mengedit foto profil berhasil

Nama Kasus Uji	Kasus uji mengedit foto profil berhasil.
Prosedur	1. Penguji berada pada halaman profil. 2. Penguji menekan tombol edit foto profil. 3. Penguji memilih foto profil baru. 4. Penguji menekan tombol simpan.

Hasil yang diharapkan	Sistem meng*update* foto profil aktor pada *database* dan menampilkan halaman profil.
Hasil	Sistem meng*update* foto profil aktor pada *database* dan menampilkan halaman profil.
Status	Valid.

6.3.36.2 Kasus Uji Mengedit Foto Profil Ketika Tidak Mengisi *Form* Edit Foto Profil Dengan Tidak Lengkap

Tabel 6.88 Kasus uji mengedit foto profil ketika tidak mengisi *form* edit foto profil dengan tidak lengkap

Nama Kasus Uji	Kasus uji mengedit foto profil ketika tidak mengisi *form* edit foto profil dengan tidak lengkap.
Prosedur	1. Penguji berada pada halaman profil. 2. Penguji menekan tombol edit foto profil. 3. Penguji tidak memilih foto profil baru. 4. Penguji menekan tombol simpan.
Hasil yang diharapkan	Sistem akan menampilkan pesan "*Please select a file*".
Hasil	Sistem akan menampilkan pesan "*Please select a file*".
Status	Valid.

6.3.36.3 Kasus Uji Mengedit Foto Profil Ketika Tombol Batal Ditekan

Tabel 6.89 Kasus uji mengedit foto profil ketika tombol batal ditekan

Nama Kasus Uji	Kasus uji mengedit foto profil ketika tombol batal ditekan.
Prosedur	1. Penguji berada pada halaman profil. 2. Penguji menekan tombol edit foto profil. 3. Penguji memilih foto profil baru. 4. Penguji menekan tombol simpan.
Hasil yang diharapkan	Sistem akan menutup modal edit foto profil.
Hasil	Sistem akan menutup modal edit foto profil.
Status	Valid.

6.3.37 Kasus Uji Melihat Grafik Riwayat Laporan Gangguan Bulanan

6.3.37.1 Kasus Uji Melihat Grafik Riwayat Laporan Gangguan Bulanan

Tabel 6.90 Kasus uji melihat grafik riwayat laporan gangguan bulanan

Nama Kasus Uji	Kasus uji melihat grafik riwayat laporan gangguan bulanan.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu riwayat laporan gangguan. 3. Penguji memilih grafik riwayat. 4. Penguji memilih grafik perbulan.
Hasil yang diharapkan	Sistem akan menampilkan halaman grafik riwayat laporan gangguan perbulan yang terdari dari grafik riwayat laporan gangguan bulanan per stasiun dan grafik riwayat jenis gangguan pada laporan gangguan bulanan pada bulan saat itu.
Hasil	Sistem akan menampilkan halaman grafik riwayat laporan gangguan perbulan yang terdari dari grafik riwayat laporan gangguan bulanan per stasiun dan grafik riwayat jenis gangguan pada laporan gangguan bulanan pada bulan saat itu.
Status	Valid.

6.3.38 Kasus Uji Mencari Akun

6.3.38.1 Kasus Uji Mencari Akun Ditemukan

Tabel 6.91 Kasus uji mencari akun ditemukan

Nama Kasus Uji	Kasus uji mencari akun ditemukan.
Prosedur	1. Penguji berada pada halaman daftar akun. 2. Penguji memasukkan *keyword* di dalam *input search.*
Hasil yang diharapkan	Sistem menampilkan data akun pada tabel daftar akun.
Hasil	Sistem menampilkan data akun pada tabel daftar akun.
Status	Valid.

6.3.38.2 Kasus Uji Mencari Akun Tidak Ditemukan

Tabel 6.92 Kasus uji mencari akun tidak ditemukan

Nama Kasus Uji	Kasus uji mencari akun tidak ditemukan.
Prosedur	1. Penguji berada pada halaman daftar akun. 2. Penguji memasukkan *keyword* di dalam *input search.*
Hasil yang diharapkan	Sistem menampilkan "*No data available in tabel*" pada tabel.
Hasil	Sistem menampilkan "*No data available in tabel*" pada tabel.
Status	Valid.

6.3.39 Kasus Uji Mengekspor Daftar Akun

6.3.39.1 Kasus Uji Mengekspor Daftar Akun

Tabel 6.93 Kasus uji mengekspor daftar akun

Nama Kasus Uji	Kasus uji mengekspor daftar akun.
Prosedur	1. Penguji berada pada halaman daftar akun. 2. Penguji menekan tombol *Export to Excel.*
Hasil yang diharapkan	Sistem akan mengekspor data akun menjadi *file excel* dan perangkat penguji akan mengunduh *file* tersebut.
Hasil	Sistem akan mengekspor data akun menjadi *file excel* dan perangkat penguji akan mengunduh *file* tersebut.
Status	Valid.

6.3.40 Kasus Uji Mencari Permohonan *Lintas*

6.3.40.1 Kasus Uji Mencari Permohonan *Lintas* Ditemukan

Tabel 6.94 Kasus uji mencari permohonan *lintas* ditemukan

Nama Kasus Uji	Kasus uji mencari permohonan *lintas* ditemukan.
Prosedur	1. Penguji berada pada halaman daftar permohonan *lintas.* 2. Penguji memasukkan *keyword* di dalam *input search.*

Hasil yang diharapkan	Sistem menampilkan data permohonan *lintas* pada tabel daftar permohonan *lintas.*
Hasil	Sistem menampilkan data permohonan *lintas* pada tabel daftar permohonan *lintas.*
Status	Valid.

6.3.40.2 Kasus Uji Mencari Permohonan *Lintas* Tidak Ditemukan

Tabel 6.95 Kasus uji mencari permohonan *lintas* tidak ditemukan

Nama Kasus Uji	Kasus uji mencari permohonan *lintas* tidak ditemukan.
Prosedur	1. Penguji berada pada halaman daftar permohonan *lintas.* 2. Penguji memasukkan *keyword* di dalam *input search.*
Hasil yang diharapkan	Sistem menampilkan "*No data available in tabel*" pada tabel.
Hasil	Sistem menampilkan "*No data available in tabel*" pada tabel.
Status	Valid.

6.3.41 Kasus Uji Mencari Laporan Gangguan yang Telah Terlaporkan

6.3.41.1 Kasus Uji Mencari Laporan Gangguan yang Telah Terlaporkan Ditemukan

Tabel 6.96 Kasus uji mencari laporan gangguan yang telah terlaporkan ditemukan

Nama Kasus Uji	Kasus uji mencari laporan gangguan yang telah terlaporkan ditemukan.
Prosedur	1. Penguji berada pada halaman daftar laporan gangguan yang terlaporkan. 2. Penguji memasukkan *keyword* di dalam *input search.*
Hasil yang diharapkan	Sistem menampilkan data daftar laporan gangguan yang terlaporkan pada tabel daftar laporan gangguan yang terlaporkan.
Hasil	Sistem menampilkan data daftar laporan gangguan yang terlaporkan pada tabel daftar laporan gangguan yang terlaporkan.

Status	Valid.

6.3.41.2 Kasus Uji Mencari Laporan Gangguan yang Telah Terlaporkan Tidak Ditemukan

Tabel 6.97 Kasus uji mencari laporan gangguan yang telah terlaporkan tidak ditemukan

Nama Kasus Uji	Kasus uji mencari laporan gangguan yang telah terlaporkan tidak ditemukan.
Prosedur	1. Penguji berada pada halaman daftar laporan gangguan yang terlaporkan. 2. Penguji memasukkan *keyword* di dalam *input search.*
Hasil yang diharapkan	Sistem menampilkan "*No data available in tabel*" pada tabel.
Hasil	Sistem menampilkan "*No data available in tabel*" pada tabel.
Status	Valid.

6.3.42 Mencari Laporan Gangguan yang Ditangani

6.3.42.1 Mencari Laporan Gangguan yang Ditangani Ditemukan

Tabel 6.98 Kasus uji mencari laporan gangguan yang ditangani ditemukan

Nama Kasus Uji	Kasus uji mencari laporan gangguan yang ditangani ditemukan.
Prosedur	1. Penguji berada pada halaman daftar laporan gangguan yang ditangani. 2. Penguji memasukkan *keyword* di dalam *input search.*
Hasil yang diharapkan	Sistem menampilkan data daftar laporan gangguan yang ditangani pada tabel daftar laporan gangguan yang ditangani.
Hasil	Sistem menampilkan data daftar laporan gangguan yang ditangani pada tabel daftar laporan gangguan yang ditangani.
Status	Valid.

6.3.42.2 Mencari Laporan Gangguan yang Ditangani Tidak Ditemukan

Tabel 6.99 Kasus uji mencari laporan gangguan yang ditangani tidak ditemukan

Nama Kasus Uji	Kasus uji mencari laporan gangguan yang ditangani tidak ditemukan.
Prosedur	1. Penguji berada pada halaman daftar laporan gangguan yang ditangani. 2. Penguji memasukkan *keyword* di dalam *input search.*
Hasil yang diharapkan	Sistem menampilkan "*No data available in tabel*" pada tabel.
Hasil	Sistem menampilkan "*No data available in tabel*" pada tabel..
Status	Valid.

6.3.43 Kasus Uji Mencari *Lintas*

6.3.43.1 Kasus Uji Mencari *Lintas* Ditemukan

Tabel 6.100 Kasus uji mencari *lintas* ditemukan

Nama Kasus Uji	Kasus uji mencari *lintas* ditemukan.
Prosedur	1. Penguji berada pada halaman daftar *lintas.* 2. Penguji memasukkan *keyword* di dalam *input search.*
Hasil yang diharapkan	Sistem menampilkan data lintas pada tabel daftar *lintas.*
Hasil	Sistem menampilkan data lintas pada tabel daftar *lintas.*
Status	Valid.

6.3.43.2 Kasus Uji Mencari *Lintas* Tidak Ditemukan

Tabel 6.101 Kasus uji mencari *lintas* tidak ditemukan

Nama Kasus Uji	Kasus uji mencari *lintas* tidak ditemukan.
Prosedur	1. Penguji berada pada halaman daftar *lintas.* 2. Penguji memasukkan *keyword* di dalam *input search.*
Hasil yang diharapkan	Sistem akan menampilkan "*No data available in tabel*" pada tabel.

Hasil	Sistem akan menampilkan "*No data available in tabel*" pada tabel.
Status	Valid.

6.3.44 Kasus Uji Mencari Barang

6.3.44.1 Kasus Uji Mencari Barang Ditemukan

Tabel 6.102 Kasus uji mencari barang ditemukan

Nama Kasus Uji	Kasus uji mencari barang ditemukan.
Prosedur	1. Penguji berada pada halaman daftar barang gudang penyimpanan. 2. Penguji memasukkan *keyword* di dalam *input search.*
Hasil yang diharapkan	Sistem menampilkan data barang pada tabel daftar barang gudang penyimpanan.
Hasil	Sistem menampilkan data barang pada tabel daftar barang gudang penyimpanan.
Status	Valid.

6.3.44.2 Kasus Uji Mencari Barang Tidak Ditemukan

Tabel 6.103 Kasus uji mencari barang tidak ditemukan

Nama Kasus Uji	Kasus uji mencari barang tidak ditemukan..
Prosedur	1. Penguji berada pada halaman daftar barang gudang penyimpanan. 2. Penguji memasukkan *keyword* di dalam *input search.*
Hasil yang diharapkan	Sistem akan menampilkan "*No data available in tabel*" pada tabel.
Hasil	Sistem akan menampilkan "*No data available in tabel*" pada tabel.
Status	Valid.

6.3.45 Kasus Uji Mencari Riwayat Laporan Gangguan

6.3.45.1 Kasus Uji Mencari Riwayat Laporan Gangguan Ditemukan

Tabel 6.104 Kasus uji mencari riwayat laporan gangguan ditemukan

Nama Kasus Uji	Kasus uji mencari riwayat laporan gangguan ditemukan.

Prosedur	1. Penguji berada pada halaman daftar riwayat laporan gangguan. 2. Penguji memasukkan *keyword* di dalam *input search.*
Hasil yang diharapkan	Sistem menampilkan data riwayat laporan gangguan pada tabel daftar riwayat laporan gangguan.
Hasil	Sistem menampilkan data riwayat laporan gangguan pada tabel daftar riwayat laporan gangguan.
Status	Valid.

6.3.45.2 Kasus Uji Mencari Riwayat Laporan Gangguan Tidak Ditemukan

Tabel 6.105 Kasus uji mencari riwayat laporan gangguan tidak ditemukan

Nama Kasus Uji	Kasus uji mencari riwayat laporan gangguan tidak ditemukan.
Prosedur	1. Penguji berada pada halaman daftar riwayat laporan gangguan. 2. Penguji memasukkan *keyword* di dalam *input search.*
Hasil yang diharapkan	Sistem akan menampilkan "*No data available in tabel*" pada tabel.
Hasil	Sistem akan menampilkan "*No data available in tabel*" pada tabel.
Status	Valid.

6.3.46 Kasus Uji Melihat Grafik Riwayat Laporan Gangguan Tahunan

6.3.46.1 Kasus Uji Melihat Grafik Riwayat Laporan Gangguan Tahunan

Tabel 6.106 Kasus uji melihat grafik riwayat laporan gangguan tahunan

Nama Kasus Uji	Kasus uji melihat grafik riwayat laporan gangguan tahunan.
Prosedur	1. Penguji berada pada halaman utama. 2. Penguji memilih menu riwayat laporan gangguan. 3. Penguji memilih grafik riwayat. 4. Penguji memilih grafik pertahun.

Hasil yang diharapkan	Sistem akan menampilkan halaman grafik riwayat laporan gangguan tahunan berdasarkan jenis gangguan pertahunnya.
Hasil	Sistem akan menampilkan halaman grafik riwayat laporan gangguan tahunan berdasarkan jenis gangguan pertahunnya.
Status	Valid.

6.3.47 Kasus Uji *Filter* Daftar Riwayat Laporan Gangguan Berdasarkan Tanggal

6.3.47.1 Kasus Uji *Filter* Daftar Riwayat Laporan Gangguan Berdasarkan Tanggal Ketika Terdapat Data Pada *Range* yang Diberikan

Tabel 6.107 Kasus uji *filter* daftar riwayat laporan gangguan berdasarkan tanggal ketika terdapat data pada *range* yang diberikan

Nama Kasus Uji	Kasus uji *filter* daftar riwayat laporan gangguan berdasarkan tanggal ketika terdapat data pada *range* yang diberikan.
Prosedur	1. Penguji berada pada halaman daftar riwayat laporan gangguan. 2. Penguji mengisi *field input* tanggal awal. 3. Penguji mengisi *field input* tanggal akhir. 4. Penguji menekan tombol *filter.*
Hasil yang diharapkan	Sistem menampilkan data riwayat laporan gangguan yang berada diantara tanggal awal dan tanggal akhir pada tabel daftar riwayat laporan gangguan.
Hasil	Sistem menampilkan data riwayat laporan gangguan yang berada diantara tanggal awal dan tanggal akhir pada tabel daftar riwayat laporan gangguan.
Status	Valid.

6.3.47.2 Kasus Uji *Filter* Daftar Riwayat Laporan Gangguan Berdasarkan Tanggal Ketika Tidak Terdapat Data Pada *Range* yang Diberikan

Tabel 6.108 Kasus uji *filter* daftar riwayat laporan gangguan berdasarkan tanggal ketika tidak terdapat data pada *range* yang diberikan

Nama Kasus Uji	Kasus uji *filter* daftar riwayat laporan gangguan berdasarkan tanggal ketika tidak terdapat data pada *range* yang diberikan.
Prosedur	1. Penguji berada pada halaman daftar riwayat laporan gangguan. 2. Penguji mengisi *field input* tanggal awal. 3. Penguji mengisi *field input* tanggal akhir. 4. Penguji menekan tombol *filter.*
Hasil yang diharapkan	Sistem akan menampilkan "*No data available in tabel*" pada tabel.
Hasil	Sistem akan menampilkan "*No data available in tabel*" pada tabel.
Status	Valid.

6.3.48 Kasus Uji Menampilkan Grafik Riwayat Laporan Bulanan Berdasarkan Bulan

6.3.48.1 Kasus Uji Menampilkan Grafik Riwayat Laporan Bulanan Berdasarkan Bulan Ketika Pada Bulan Tersebut Terdapat Data Riwayat Laporan Gangguan

Tabel 6.109 Kasus uji menampilkan grafik riwayat laporan bulanan berdasarkan bulan ketika pada bulan tersebut terdapat data riwayat laporan gangguan

Nama Kasus Uji	Kasus uji menampilkan grafik riwayat laporan bulanan berdasarkan bulan ketika pada bulan tersebut terdapat data riwayat laporan gangguan.
Prosedur	1. Penguji berada pada halaman grafik riwayat laporan gangguan perbulan. 2. Penguji mengisi *field input* bulan tahun. 3. Penguji menekan tombol pilih.
Hasil yang diharapkan	Sistem menampilkan halaman grafik riwayat laporan gangguan perbulan berdasarkan *input*an bulan tahun yang dimasukan.

Hasil	Sistem menampilkan halaman grafik riwayat laporan gangguan perbulan berdasarkan *input*an bulan tahun yang dimasukan.
Status	Valid.

6.3.48.2 Kasus Uji Menampilkan Grafik Riwayat Laporan Bulanan Berdasarkan Bulan Ketika Pada Bulan Tersebut Tidak Terdapat Data Riwayat Laporan Gangguan

Tabel 6.110 Kasus uji menampilkan grafik riwayat laporan bulanan berdasarkan bulan ketika pada bulan tersebut tidak terdapat data riwayat laporan gangguan

Nama Kasus Uji	Kasus uji menampilkan grafik riwayat laporan bulanan berdasarkan bulan ketika pada bulan tersebut tidak terdapat data riwayat laporan gangguan.
Prosedur	1. Penguji berada pada halaman grafik riwayat laporan gangguan perbulan. 2. Penguji mengisi *field input* bulan tahun. 3. Penguji menekan tombol pilih.
Hasil yang diharapkan	Sistem akan menampilkan grafik kosong.
Hasil	Sistem akan menampilkan grafik kosong.
Status	Valid.

6.4 Pengujian Validasi Kebutuhan Non Fungsional

6.4.1 Pengujian Validasi *Compatibility*

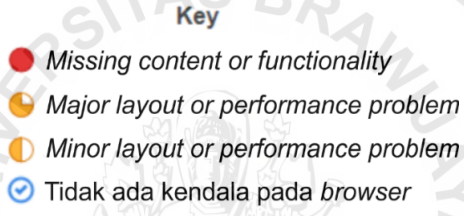
Pengujian *compatibility* atau *compatibility testing* dilakukan dengan tujuan untuk mengetahui apakah sistem yang dibangun ini dapat berjalan dengan semestinya pada setiap *browser* dan *device*. Kebutuhan non fungsional *compatibility* ini memiliki kode kebutuhan SPG_NF_01. *Compatibility testing* ini akan dilakukan menggunakan alat bantu *SortSite* versi 5.31.830.0. Alat ini digunakan untuk membantu untuk menemukan apakah terdapat masalah pada jika sistem dijalankan pada *browser* atau *device* tertentu. Pada Tabel 6.111 disebutkan *browser* yang digunakan menjadi parameter pengujian *Compatibility*.

Tabel 6.111 Browser yang digunakan pada *compatibility testing*

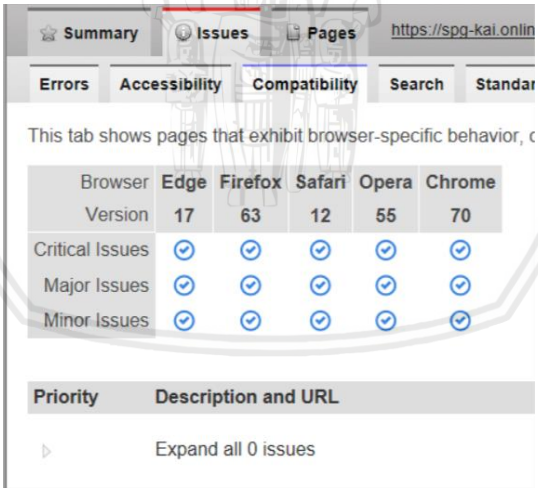
No	Nama Browser	Versi Browser
1	*Edge*	17
2	*Firefox*	63
3	*Safari*	12

4	*Opera*	55
5	*Chrome*	70

Pengujian ini dilakukan dengan memasukkan *url website* pada *address bar* yang tersedia di *SortSite*. Kemudian secara otomatis *SortSite* akan melakukan pengecekan terhadap *website* yang *url*nya dimasukkan untuk mengecek apakah terdapat masalah dalam *browser* tertentu dalam menjalankan *website* tersebut. Ada tiga kategori yang menjadi indikator masalah dalam proses pengujian *compatibility*, yaitu *missing content or functionality* yang berarti adanya konten atau fungsionalitas yang tidak muncul atau tidak berfungsi sehingga menjadikan *browser* tidak kompatibel untuk mengakses sistem, *major layout or performance problems* dan *minor layout or performance problems*. *Major layout or performance problems* dan *layout or performance problems* mengindikasikan bahwa ditemukan beberapa masalah yang muncul pada tampilan dan performa namun tidak mempengaruhi *browser* dalam menjalanankan sistem. Tiga jenis kategori ini dapat dilihat pada Gambar 6.12.



Gambar 6.12 Indikator masalah *compatibility testing*



Gambar 6.13 Hasil pengujian *compatibility*

Tabel 6.112 Hasil pengujian *compatibility*

Parameter	**Tag HTML dan atribut yang kompatibel pada *browser***	**Aturan CSS yang didukung oleh *browser***	**Tidak ada *element* yang hilang**	**Tidak ada properti *JavaScript* dan *DOM* dan fungsi yang memicu**

Browser				***bug* pada *browser***
Edge	Valid	Valid	Valid	Valid
Firefox	Valid	Valid	Valid	Valid
Safari	Valid	Valid	Valid	Valid
Opera	Valid	Valid	Valid	Valid
Chrome	Valid	Valid	Valid	Valid

Pada Gambar 6.13 dapat dilihat hasil pengujian *compatibility* pada sistem yang dibangun secara keseluruhan. Pada Tabel 6.112 dapat dilihat hasil pengujian *compatibility* berdasarkan parameter yang sudah ditentukan. Dengan menguji pada 5 jenis *browser* dalam versi yang berbeda-beda, didapatkan bahwa sistem yang dibuat tidak mengandung indikator masalah yang berarti sistem ini dapat dijalankan dengan baik di berbagai *browser*.

6.5 Pembahasan Hasil Pengujian

Ada empat pengujian yang dilakukan untuk menguji sistem pelaporan gangguan, yaitu pengujian verifikasi yang terdiri dari pengujian unit dan pengujian integrasi dan pengujian validasi yang terdiri dari pengujian validasi fungsional dan pengujian validasi non fungsional. Pengujian verifikasi dilakukan dengan menggunakan metode *white-box testing* sedangkan pengujian validasi menggunakan metode *black-box testing.* Pada pengujian unit dari 3 sampel uji yang terdapat total 12 *test case* menunjukkan bahwa 100% valid dengan rata-rata nilai *complexity number* yaitu 4. Pada pengujian integrasi dari 1 sampel uji yang terdapat 2 *test case* menunjukkan bahwa 100% valid. Pada pengujian validasi fungsional dari 96 kasus uji menunjukkan bahwa 100% valid. Dan yang terakhir pada pengujian non fungsional yaitu pengujian *compatibility* menunjukkan pada 5 *browser* dengan versi yang berbeda menunjukkan bahwa sistem telah kompatibel dengan semua *browser* yang diuji.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil analisis yang dilakukan dalam pembangunan sistem pelaporan gangguan ditentukan 48 kebutuhan fungsional sistem dan 1 kebutuhan non fungsional sistem. Selain itu berdasarkan hasil analisis didapatkan empat jenis aktor yang dapat mengakses sistem pelaporan gangguan ini, yaitu pengguna, pimpinan unit IT, petugas unit IT dan pelapor. Pengguna merupakan aktor yang dikenali sistem sebagai aktor yang belum valid, sedangkan pimpinan unit IT, petugas unit IT dan pelapor merupakan aktor yang dikenali sistem sebagai aktor yang valid, yang artinya dapat mengakses fungsionalitas utama sistem. Hasil analisis ini didapatkan dari kegiatan studi literatur, melakukan observasi, wawancara dan validasi kebutuhan sistem yang dibuat dengan asisten manajer unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon.

Pada perancangan yang dilakukan pada perancangan sistem pelaporan gangguan didapatkan bahwa sistem akan dibuat dengan berbasis web yang mengaplikasikan teknologi *progressive web application* yang menggunakan *application shell architecture*. Untuk mengaplikasikan teknologi *progressive web application* ini dibutuhkan 3 syarat utama yaitu *web app manifest* yang berfungsi untuk memberitahukan *browser* bagaimana aplikasi web yang dijalankan seharusnya berperilaku ketika dilakukan instalasi pada perangkat selular atau *desktop*, *service workers* yang bertugas untuk melakukan *caching* pada *user interface* dan infrastruktur aplikasi yang nantinya akan disimpan di *cache storage*, dan harus dijalankan diatas HTTPS. Pada sistem pelaporan gangguan gangguan bukan hanya *user interface* dan infrastruktur aplikasi saja yang disimpan namun juga menyimpan data dinamis yang disimpan pada *local storage*. Hal ini dilakukan agar sistem pelaporan gangguan dapat diakses walaupun dalam kondisi jaringan yang buruk ataupun tidak ada jaringan sama sekali atau *offline*. Untuk memberikan notifikasi sistem pelaporan gangguan menggunakan dua cara yaitu dengan *web push notification* dan pesan aplikasi *WhatsApp*. Untuk *web push notification* sistem pelaporan gangguan menggunakan bantuan API dari *OneSignal* karena API ini dapat membantu mengirimkan pesan hanya untuk pengguna yang diinginkan saja. Untuk pesan aplikasi *WhatsApp* sistem pelaporan gangguan menggunakan bantuan API dari APIWHA. Selain didapatkan perancangan arsitektur sistem pada proses perancangan sistem pelaporan gangguan juga didapatkan perancangan *sequence diagram*, perancangan *class diagram*, perancangan komponen, perancangan *database* dan perancangan antarmuka.

Hasil implementasi yang dilakukan adalah sistem pelaporan gangguan yang menggunakan bahasa pemrograman *PHP* untuk bagian *back-end* aplikasi dan bahasa *HTML*, *CSS*, dan *JavaScript* untuk bagian *front*-end aplikasi. Aplikasi ini menggunakan pola perancangan *Model-View-Controller* dengan menggunakan *framework CodeIgniter*. Sistem pelaporan gangguan ini berdiri pada *platform web* dengan menggunakan teknologi *progressive web application*. Untuk memberikan

notifikasi sistem ini menggunakan API dari *OneSignal* dan APIWHA. API *OneSignal* digunakan untuk memberikan notifikasi dengan menggunakan *web push notification*. Sedangkan API APIWHA digunakan untuk memberikan notifikasi menggunakan pesan aplikasi *WhatsApp*.

Hasil pengujian yang dilakukan dengan menggunakan pengujian unit pada tiga sampel uji memiliki rata-rata *complexity number* 4. Hasil pengujian ini menunjukkan bahwa sistem yang dibangun memiliki struktur dan penulisan yang baik serta *high testablity*, serta 100% hasil kasus ujinya bernilai valid. Hasil pengujian integrasi dari satu sampel uji bernilai 100% valid untuk 2 kasus uji. Hasil pengujian validasi fungsional yang digunakan dengan *black-box testing* menghasilkan nilai validitas sebesar 100%. Dan yang terakhir pengujian *compatibility* yang dilakukan dengan alat bantu *SortSite* 5 menunjukkan bahwa sistem dapat berjalan di berbagai *browser* dan *device*.

7.2 Saran

Saran yang dapat diberikan untuk pengembangan lebih lanjut sistem pelaporan gangguan, yaitu:

1. Menambahkan kebutuhan fungsional dimana petugas dan pelapor dapat melakukan pengecekan gangguan dengan menggunakan *QR code scanner* dari QR *code* yang ada pada *e-ticket* pelaporan gangguan sehingga dapat meningkatkan efektivitas dan efisiensi dalam proses pengecekan gangguan.
2. Menambahkan penggunaan algoritma pendukung seperti algoritma genetika dalam fitur penjadwalan sehingga dapat mendapatkan hasil penjadwalan yang optimal.

DAFTAR PUSTAKA

Arifianto, A., 2018. *Prosedur Pelaporan Gangguan Unit IT PT. Kereta Api Indonesia Daerah Operasi III Cirebon.* Diwawancarai oleh Ari Setiawan. [Wawancara] Kantor Unit IT PT. KAI DAOP III Cirebon, 25 Juli 2018.

Basir, N., 2005. *Desain dan Implementasi Aplikasi Pelaporan Gangguan Sistem SCADA (LGS) Pada PT. PLN APD Bandung*. S1. Universitas Telkom.

Biørn-Hansen, A., Majchrzak, T. A., dan Grønli, T. M., 2017. *Progressive Web Apps: The Possible Web-native Unifier for Mobile Development*. Proceedings of the 13th International Conference on Web Information Systems and Technologies. Porto, Portugal, 25-27 April 2017.

Caytiles, R. D., dan Lee, S., 2014. *A Review of an MVC Framework based Software Development*. International Journal of Software Engineering and Its Applications, 8(10), hal. 213-220.

Chou, C., Hsu, Y. H., dan Goo, Y. J., 2009. *Service Failures and Recovery Strategies from the Service Provider Perspective*. *Asia Pasific Management Review*, 14(2), hal. 237-249.

Gaunt, M., 2018. *Service Workers: an Introduction*. [online] Google Developers. Tersedia di:< https://developers.google.com/web/fundamentals/primers/service-workers/ > [Diakses 15 Januari 2019].

Gaunt, M., dan Kinlan, P., 2018. *The Wep App Manifest*. [online] Google Developers. Tersedia di:< https://developers.google.com/web/fundamentals/web-app-manifest/> [Diakses 15 Januari 2019].

Guru99, 2019a. *Mccabe's Cyclomatic Complexity: Calculate with Flow Graph (Example)*. [online] Guru99. Tersedia di:< https://www.guru99.com/cyclomatic-complexity.html > [Diakses 15 Januari 2019].

Guru99, 2019b. *What is BLACK Box Testing? Techniques, Example & Types*. [online] Guru99. Tersedia di:< https://www.guru99.com/black-box-testing.html > [Diakses 15 Januari 2019].

Khurshid, R., Naeem, H., Ejaz, S., Muktar, F., dan Batool, T., 2012. *Service Quality And Customer Satisfaction In Public Transport Sector Of Pakistan: An Empirical Study*. International Journal of Economics and Management Sciences, 1(9), hal. 24-30.

Lepage, P., 2018. *Your First Progressive Web App*. [online] Google Developers. Tersedia di:< https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/ > [Diakses 15 Januari 2019].

Malavolta, I., 2016. *Beyond Native Apps: Web Technologies to the Rescue!*. Proceedings of the 1st International Workshop on Mobile Development. Amsterdam, Belanda.

Permatasari, R. A., Priyambadha, B., dan Arwan, A., 2018. *Pengembangan Sistem Aplikasi Pelaporan Masyarakat Berbasis Web di Kabupaten Pekalongan*. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2(11) hal. 5604-5610.

Pressman, R. S., 2010. *Software engineering: A Practitioner's Approach*. 7th ed. New York: McGraw-Hill Irwin.

PT. Kereta Api Indonesia (Persero), 2017. *Profil Perusahaan*. [online] PT. Kereta Api Indonesia (Persero). Tersedia di:< https://www.kai.id/corporate/page/11 > [Diakses 21 Agustus 2018].

Rescorla, E., 2000. *HTTP Over TLS*. RFC Editor.

Ridho, M. R., Pinandito, A., dan Dewi, R. K., 2018. *Perbandingan Performa Progressive Web Apps dan Mobile Web Terkait Waktu Respon, Penggunaan Memori dan Penggunaan Media Penyimpanan.* Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2(10), hal. 3483-3491.

Riyandwyana, A., dan Mukhlason, E. S., 2012 *Pengembangan Sistem Rekomendasi Peminjaman Buku Berbasis Web Menggunakan Metode Self Organizing Map Clustering Pada Badan Perpustakaan dan Kearsipan (BAPERSIP) Provinsi Jawa Timur*. Jurnal Teknik ITS, 1(1), hal. 374-378.

Whitten, J. L., dan Bentley, L. D., 2007. *System Analysis & Design Methods*. 7th ed. New York: McGraw-Hill Irwin.