PENGEMBANGAN APLIKASI MANAJEMEN INFORMASI SEDEKAH BERBAGI MAKANAN BERBASIS ANDROID DENGAN METODE REKAYASA PERANGKAT LUNAK BERORIENTASI PENGGUNAAN ULANG

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Annisa Fitriani Nur
NIM: 145150201111070

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PENGEMBANGAN APLIKASI MANAJEMEN INFORMASI SEDEKAH BERBAGI MAKANAN BERBASIS ANDROID DENGAN METODE REKAYASA PERANGKAT LUNAK BERORIENTASI PENGGUNAAN ULANG

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:
Annisa Fitriani Nur
NIM: 145150201111070

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Agi Putra Kharisma, S.T, M.T	Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIK: 201304860430 1 001	NIP: 19710518 200312 1 001

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 pustaka.

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

Malang, 2 Agustus 2018

Annisa Fitriani Nur

NIM: 145150201111070

KATA PENGANTAR

Alhamduilllah, Segala puji bagi Allah SWT karena atas limpahan rahmat dan hidayah-Nya serta kenikmatan yang luar biasa besarnya sehingga penulis dapat menyelesaikan tugas akhir skripsi yang berjudul "Pengembangan Aplikasi Manajemen Informasi Sedekah Berbagi Makanan Berbasis Android dengan Metode Rekayasa Perangkat Lunak Berorientasi Penggunaan Ulang". Ucapan terimakasih penulis sampaikan kepada seluruh pihak yang turut membimbing dan menemani penulis dalam menyelesaikan skripsi ini.

Penyusunan tugas akhir skripsi ini dilakukan untuk menyelesaikan persyaratan kuliah guna memperoleh gelar Sarjana Komputer (S.Kom) pada Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya Malang. Adapun tujuan dari penyusunan skripsi ini adalah sebagai wujud dan kajian pengetahuan dan/atau penerapan teknologi berdasarkan apa yang telah dipelajari semasa perkuliahan.

Pada kesempatan ini penulis juga menyampaikan rasa terima kasih kepada pihak-pihak yang telah membantu selama menyusun tugas akhir skripsi, diantaranya:

1. Allah swt. yang telah memberikan nikmat kesempatan, waktu, kesehatan dan nikmat-nikmat besar lainnya.

2. Bapak Wayan Firdaus Mahmudy, S.Si, M.T. Ph. D, selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.

3. Bapak Agus Wahyu Widodo, S.T, M. Cs, selaku Ketua Program Studi Teknik Informatika Universitas Brawijaya.

4. Bapak Tri Astoto Kurniawan S.T, M.T, Ph. D, selaku Ketua Jurusan Teknik Informatika sekaligus Dosen Pembimbing Skripsi yang telah membimbing, mengarahkan, memberi nasihat serta motivasi selama penyusunan skripsi.

5. Bapak Agi Putra Kharisma, S.T, M.T, selaku Dosen Pembimbing Skripsi yang telah banyak meluangkan waktu dalam membimbing, mengarahkan, memberi nasihat serta saran selama penyusunan skripsi.

6. Bapak Djamaluddin Nur dan Ibu Nursiah selaku Orang Tua penulis yang telah memberi nasihat, perhatian dan kesabaran selama menyelesaikan skripsi.

7. Ustadzah Kholishotu Tahlia yang selalu memberikan nasihat dan dan pesan moril selama menyelesaikan tugas akhir ini.

8. Teman-teman penulis yaitu Dyah Ayu Prabandari, Febriyani Riyanda, Andra Pargiyani, Dyah Ayu Ocky, Alqis Rausanfita, Indah Mutia Ayudita, Riski Puspa Dewi dan Diajeng Ninda Armiyanti yang telah membantu, menemani dan memberi dukungan dalam menyelesaikan tugas akhir skripsi ini.

9. Saudara Andri Suranta Ginting sebagai tentor yang telah mengajari dan memberi masukan dalam menyelesaikan skripsi.

10. Teman-teman mahasiswa teknik informatika angkatan 2014 yang telah membantu penulis dalam menyusun skripsi.

11. Semua pihak yang banyak membantu dan memberi dukungan dalam penyusunan tugas akhir skripsi ini.

Penulis menyadari masih terdapat banyak kekurangan dalam penyusunan tugas akhir skripsi ini. Demi kesempurnaan tugas akhir skripsi ini, penulis mengharapkan adanya kritik serta saran yang sifatnya membangun. Semoga karya tulis ini dapat bermanfaat bagi pihak manapun.

Malang, 2 Agustus 2018

Annisa Fitriani Nur

annisafitrianin@gmail.com

ABSTRAK

Annisa Fitriani Nur, Pengembangan Aplikasi Manajemen Informasi Sedekah Berbagi Makanan Berbasis Android Dengan Metode Rekayasa Perangkat Lunak Berorientasi Penggunaan Ulang

Pembimbing: Agi Putra Kharisma, S.T., M.T. dan Tri Astoto Kurniawan S.T., M.T., Ph.D.

Memberi sedekah adalah salah satu aktivitas yang sangat dianjurkan dalam agama Islam. Memberi sedekah dapat berupa uang, barang atau makanan. Salah satu kendala pemberian sedekah adalah tidak adanya informasi terkait waktu dan tempat kegiatan sedekah. Oleh karena itu, dibutuhkan sebuah aplikasi manajemen informasi sedekah yang diharapkan dapat mengatasi permasalahan tersebut. Sistem ini membutuhkan sebuah aplikasi penampil peta untuk menunjukan lokasi kegiatan, pencarian lokasi terdekat dan komponen-komponen penggunaan ulang lainnya. Sehingga aplikasi ini menggunakan metode rekayasa perangkat lunak berbasis penggunaan ulang komponen dengan *component retrieval* dalam pengembangannya. Metode penggunaan ulang komponen diantaranya adalah analisis kebutuhan, analisis komponen, modifikasi kebutuhan, perancangan dengan penggunaan ulang, kontruksi serta pengujian. Aplikasi ini menggunakan bahasa pemodelan UML *(Unified Modeling Languange)*, implementasinya menggunakan bahasa pemrograman Java dengan Android SDK (*Software Development Kit*). Pengujian dilakukan dengan pengujian unit, pengujian integrasi dan pengujian validasi. Pengujian unit dan integrasi menggunakan pendekatan kotak putih (*white-box*) serta pengujian validasi menggunakan pendekatan kotak hitam (*black-box*).

Kata kunci: aplikasi manajemen informasi sedekah, *component retrieval*, *library*, UML, android.

ABSTRACT

Annisa Fitriani Nur, Pengembangan Aplikasi Manajemen Informasi Sedekah Berbagi Makanan Berbasis Android Dengan Metode Rekayasa Perangkat Lunak Berorientasi Penggunaan Ulang

Supervisor: Agi Putra Kharisma, S.T., M.T. and Tri Astoto Kurniawan S.T., M.T., Ph.D.

Giving alms is one of the activities that is highly recommended in Islam. Giving alms could be done by giving money, goods, and food. One of the obstacles almsgiving is there's no adequate information about the detail of event will be held, such as time and place of the event. Therefore, these problem can be solved by an alms information management aplication. This application requires a maps viewer to show information about the location of the event, nearby location searching, and other components reuse. This application is using a software engineering method based on components reuse with component retrieval in its development. Method of components reuse consists of requirements analysis, components analysis, requirements modification, design with reuse, construction, and testing. This application is using UML (Unified Modeling Language) modeling language and Java programming language with Android SDK (Software Development Kit) in implementation. This application has been tested through the unit testing, integration testing, and validation testing. Unit testing and integration testing are using white-box approach then validation testing is using black-box approach.

Keywords: alms information management application, component retrieval, library, UML, android.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Sedekah adalah salah satu aktivitas pemberian sebagian harta berupa barang atau makanan secara sukarela kepada orang lain yang membutuhkan. Kata sedekah berasal dari bahasa Arab yaitu "*shadaqoh"* yang berarti pemberian tanpa ada batasan waktu dan jumlah tertentu. Sedekah dalam arti luas digambarkan adalah seluruh amal perbuatan. Bahkan dalam sebuah hadist dikatakan "memberikan senyuman kepada saudaramu adalah sedekah", yang berarti bahwa kata sedekah tidak terbatas hanya pada pemberian berupa barang atau makanan, namun pemberian secara keseluruhan termasuk perbuatan.

Berdasarkan survei yang dilakukan dengan jumlah responden 39 orang mengatakan bahwa semua orang berminat untuk melakukan sedekah dalam bentuk makanan. Namun beberapa responden mengalami kendala saat melakukan sedekah makanan. Berikut beberapa kendala yang dialami:

1. Kurang percaya jika memberikan bantuan dalam bentuk uang
2. Merasa kesulitan menentukan kepada siapa dan di mana harus memberikan sedekah
3. Tidak memiliki informasi yang jelas mengenai tempat dan waktu serta orang yang berhak menerima sedekah
4. Tidak memiliki kesempatan yang banyak untuk mencari orang-orang yang layak untuk menerima sedekah

Berdasarkan hasil survei dapat disimpulkan bahwa sebagian besar masyarakat masih mengalami hambatan ketika ingin bersedekah. Hambatan yang paling besar adalah tidak adanya informasi mengenai tempat dan waktu kegiatan sedekah serta kepada siapa saja yang berhak menerima sedekah.

Pada survei ini, informasi lain yang kami dapatkan yaitu 83,3% masyarakat dengan berbagai latar belakang pekerjaan yang berbeda memberikan keterangan bahwa mereka membutuhkan aplikasi yang menunjukkan informasi kegiatan sedekah baik berupa kegiatan, tempat, dan waktu. Hal ini juga diperkuat dengan pernyataan responden bahwa 38,5% orang termasuk pengguna *smartphone* yang aktif menggunakan *smartphone* lebih dari 5 jam sehari, dan 33,3% menggunakan lebih dari 10 jam perhari. Dari informasi survei dapat dikatakan bahwa sebagian besar masyarakat banyak menggunakan perangkat *smartphone* dalam aktivitas sehari-hari, baik untuk berkomunikasi maupun memanfaatkan aplikasi *smartphone*.

Salah satu perangkat *smartphone* yang banyak digunakan adalah sistem operasi berbasis android. Dari sisi pengembangan, banyak perusahaan-perusahaan penghasil telepon seluler yang menggunakan sistem operasi android karena beberapa keunggulannya. Salah satu keunggulan sistem operasi android adalah sebagai *software* yang memakai basis kode komputer yang distribusinya

secara terbuka *(open source)* sehingga pengguna dapat memakai aplikasi yang dengan mudah diunduh secara gratis melalui Google Play Store.

Sistem ini membutuhkan aplikasi penampil peta, pencarian lokasi-lokasi terdekat serta komponen-komponen *reuse* yang tersedia lainnya. Sehingga untuk mengurangi biaya pengembangan dan waktu pengembangan yang relatif singkat maka dibutuhkan sebuah metode pengembangan perangkat lunak yang menggunakan kembali komponen perangkat lunak yang dibutuhkan tanpa harus membuat komponen dari awal.

Salah satu metode pengembangan perangkat lunak adalah penggunaan ulang komponen atau *component reuse.* Metode penggunaan ulang adalah suatu metode pengembangan perangkat lunak yang menggunakan sistem perangkat lunak dari perangkat lunak yang sudah ada untuk membangun perangkat lunak yang baru (Somerville, 2011). Penggunaan ulang memiliki beberapa kelebihan diantaranya mengurangi biaya pengembangan perangkat lunak dan menghemat waktu pengembangan.

Oleh karena itu, pada penelitian ini, peneliti akan menerapkan metode pengembangan perangkat lunak berorientasi penggunaan ulang dalam membuat sebuah aplikasi penyedia informasi kegiatan sedekah. Aplikasi ini diharapkan dapat membantu masyarakat dalam melakukan kegiatan sedekah dengan menggunakan perangkat *smartphone* berbasis android.

1.2 Rumusan Masalah

1. Bagaimana hasil analisis dan spesifikasi kebutuhan sistem manajemen informasi sedekah berbagi makanan yang sesuai dengan kebutuhan pengguna?
2. Bagaimana rancangan sistem perangkat lunak yang sesuai dengan spesifikasi kebutuhan sistem manajemen informasi sedekah berbagi makanan berorientasi penggunaan ulang?
3. Bagaimana hasil implementasi sistem manajemen informasi sedekah berbagi makanan dengan menggunakan *platform* android yang sesuai dengan rancangan sistem tersebut?
4. Bagaimana hasil pengujian sistem perangkat lunak manajemen informasi sedekah berbagi makanan?

1.3 Tujuan

1. Mengetahui hasil analisis dan spesifikasi kebutuhan sistem manajemen informasi sedekah berbagi makanan yang sesuai dengan kebutuhan pengguna.
2. Mengetahui rancangan sistem perangkat lunak yang sesuai dengan spesifikasi kebutuhan sistem manajemen informasi sedekah berbagi makanan berorientasi penggunaan ulang.

3. Mengetahui hasil implementasi dari rancangan sistem manajemen informasi sedekah berbagi makanan dengan menggunakan *platform* android.
4. Mengetahui hasil uji sistem perangkat lunak manajemen informasi sedekah berbagai makanan.

1.4 Manfaat

Pada penelitian ini, manfaat yang didapatkan antara lain:

1. Bagi pemberi sedekah
   a. Dapat membuat kegiatan sedekah menjadi lebih mudah, cepat dan efisien menggunakan aplikasi *smartphone.*
   b. Dapat menentukan siapa saja yang berhak menerima sedekah makanan.
2. Bagi penerima sedekah
   a. Dapat dengan mudah mengetahui informasi lokasi dan waktu kegiatan pembagian sedekah makanan.
3. Bagi pengembang perangkat lunak
   a. Memberikan rujukan baru bagi pengembang aplikasi mengenai penerapan rekayasa perangkat lunak berorientasi penggunaan ulang.

1.5 Batasan Masalah

Agar permasalahan yang dirumuskan tidak keluar melebihi konteks permasalahan yang ada, maka penelitian ini dibatasi dalam hal :

1. Komponen penggunaan ulang yang digunakan terbatas pada komponen yang tersedia di *library* pada *platform* android.
2. Aplikasi manajemen informasi sedekah ini hanya digunakan pada perangkat dengan sistem operasi Android minimal API 16.

1.6 Sistematika Pembahasan

Sistematika pembahasan pada penelitian ini adalah sebagai berikut:

1. **BAB 1 PENDAHULUAN**

   Pada bab ini berisi latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika pembahasan.

2. **BAB 2 LANDASAN TEORI**

   Bab ini menjelaskan mengenai tinjauan pustaka dan dasar-dasar teori yang digunakan. Pada bab ini berisi hasil penelitian-penelitian sebelumnya sedangkan pada dasar teori berisi manajemen sistem informasi, sedekah, rekayasa perangkat lunak, rekayasa perangkat lunak berorientasi

penggunaan ulang, *component retrieval*, UML, android, dan Firebase Realtime Database.

3. **BAB 3 METODOLOGI**

   Bab ini menjelaskan kerangka kerja dari penelitian ini. Kerangka kerja berisi metode pengumpulan data yang terdiri dari studi literatur dan survei, analisis kebutuhan, analisis komponen, modifikasi kebutuhan, perancangan, implementasi dan pengujian serta kesimpulan dan saran.

4. **BAB 4 ANALISIS KEBUTUHAN**

   Pada bab ini menjelaskan spesifikasi kebutuhan di antaranya deskripsi umum sistem, karakter pengguna, kebutuhan fungsional dan nonfungsional. Adapun pemodelan kebutuhan di antaranya adalah *use case diagram*, *use case scenario*, *diagram activity*, dan *flow diagram*.

5. **BAB 5 ANALISIS KOMPONEN DAN PERANCANGAN**

   Pada bab ini menjelaskan mengenai analisis komponen yaitu *component retrieval*. Sedangkan perancangan sistem terdiri dari *sequence diagram*, diagram komponen, *class diagram,* perancangan algoritme dan perancangan basis data.

6. **BAB 6 IMPLEMENTASI DAN PENGUJIAN**

   Pada bab ini terdiri dari dua bagian yaitu implementasi dan pengujian. Tahap implementasi terdiri dari spesifikasi sistem, implementasi antarmuka, dan implementasi kode program. Sedangkan tahap pengujian terdiri dari pengujian unit, pengujian integrasi, dan pengujian validasi.

7. **BAB 7 PENUTUP**

   Pada bab ini berisi kesimpulan dari hasil penelitian dan saran.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Tinjauan Pustaka

Tinjauan pustaka pada penelitian ini akan membahas beberapa hasil penelitian sebelumnya yang menjadi rujukan. Pada penelitian ini dilakukan tinjauan pustaka dengan hasil penelitian-penelitian dan literatur-literatur terdahulu. Dalam kajian ini disertakan beberapa penelitian yang menggunakan metode pengembangan berbasis penggunaan ulang. Jurnal pertama yang menjadi rujukan yaitu berjudul *"Development of a Software Repository for the Precise Search and Exact Retrieval of the Components"* oleh Amandeep Bakshi, Seema Bawa (Bakshi & Bawa, 2013). Dalam jurnal tersebut dijelaskan bagaimana aplikasi penyimpanan dan pencarian dibangun berdasarkan komponen penggunaan ulang dengan *exact retrieval,* yaitu pengambilan komponen secara tepat. Jurnal ini mengarahkan mekanisme penggunaan ulang komponen android mulai dari analisis komponen hingga pengujian perangkat lunak. Dalam mengembangkan aplikasi manajemen informasi sedekah, metode yang digunakan sama namun lebih spesifik adalah penggunaan ulang komponen *library* pada platform android.

Jurnal kedua yang menjadi bahan rujukan berjudul "Sistem Informasi Manajemen Zakat, Infak, dan Sedekah pada Badan Amil Zakat Nasional" oleh Agung Pandu Dwipratma (Dwipratma, 2011). Dalam jurnal tersebut menghasilkan suatu artefak yaitu perangkat lunak sistem informasi manajemen zakat, infak dan sedekah berbasis web dengan model pengembangan *object oriented*. Ada beberapa hal yang menjadikan jurnal ini menjadi rujukan dalam melakukan penelitian yaitu objek penelitian dan model pengembangan *object oriented.* Dalam mengembangkan aplikasi manajemen informasi sedekah, objek kajian yang digunakan sama namun dalam implementasinya aplikasi yang digunakan berbasis android.

Tinjauan jurnal yang ketiga adalah jurnal yang ditulis oleh Edward (Edward 2014) yang berjudul "Perancangan Sistem Informasi Manajemen Zakat". Pada jurnal tersebut berisi tentang rancangan berbasis android mengenai manajemen zakat namun model pengembangan yang digunakan adalah struktural, sehingga dalam perancangannya menggunakan model DFD *(Data Flow Diagram)* dan diagram struktural lainnya. Ide yang peneliti ambil dari jurnal tersebut yaitu objek penelitian dan *platform* yang digunakan. Dalam mengembangkan aplikasi manajemen informasi sedekah, objek kajian yang digunakan sama namun pada penelitian ini menggunakan model OO (*Object oriented*) dalam perancangannya.

Tabel 2. 1 Tabel penelitian terdahulu

No.	Judul penelitian	Nama peneliti & tahun penelitian	Fokus penelitian
1.	*Development of a Software Repository for the Precise Search and*	Amandeep Bakshi & Seema Bawa (2013)	Fokus penelitian ini adalah mengembangkan perangkat lunak

	Exact Retrieval of the Components		menggunakan metode *exact retrieval* pada sistem penggunaan ulang komponen
2.	Sistem Informasi Manajemen Zakat, Infak, dan Sedekah pada Badan Amil Zakat Nasional	Agung Pandu Dwipratma (2011)	Fokus penelitian ini adalah pengembangan sistem informasi manajemen yang mengadaptasi prinsip-prinsip rekayasa perangkat lunak berbasis web
3.	Perancangan Sistem Informasi Manajemen Zakat	Edward (2014)	Fokus penelitian ini adalah merancang sebuah perangkat lunak berbasis android dengan menerapkan prinsip-prinsip rekayasa perangkat lunak

2.2 Sedekah

Sedekah berasal dari kata bahasa Arab yaitu *shadaqoh* yang menurut para ahli fiqh berarti suatu pemberian yang diberikan oleh seorang muslim kepada orang lain secara spontan dan sukarela tanpa dibatasi waktu dan jumlah tertentu (Fahrul, 2016). Tujuan manusia bersedekah tidak lain adalah semata-mata ibadah kepada pencipta dan bentuk rasa syukur atas kenikmatan harta yang dimiliki. Islam adalah agama yang sempurna, paripurna, dan menjadi solusi atas segala permasalah di muka bumi ini, termasuk memberikan solusi yang tepat guna meringankan permasalahan orang lain yang kurang mampu dengan cara bersedekah. Allah swt. Menegaskan bahwa dalam harta orang-orang yang kaya terdapat hak orang fakir miskin dan dhuafa.

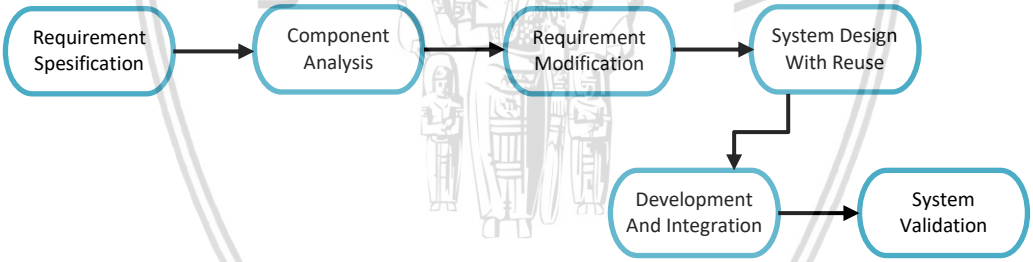
Pengertian *shadaqoh* lebih luas dari infak dan zakat, karena sedekah tidak hanya mengeluarkan berupa harta namun mencakup seluruh amal atau perbuatan yang baik, seperti dalam sebuah hadis digambarkan bahwa "memberikan senyum kepada saudaramu adalah sedekah". Makna hadis tersebut sangat jelas bahwa sedekah adalah segala bentuk kebaikan yang dilakukan seorang muslim semata-mata mengharap ridho Allah swt. (Maulan, 2008).

Adapun macam-macam sedekah salah satunya adalah membantu urusan orang lain, Abdillah bin Qais bin Salim Al-Madani, dari Nabi Muhammad saw. bahwa beliau bersabda, "Setiap muslim harus bersedekah" Salah seorang sahabat bertanya, "Bagaimana pendapatmu, wahai Rasulullah, jika ia tidak mendapatkan (harta yang dapat disedekahkan)?" Rasulullah saw. bersabda, "Bekerja dengan tangannya sendiri kemudian ia memanfaatkannya untuk dirinya dan bersedekah"

Salah seorang sahabat bertanya, “Bagaimana jika ia tidak mampu, wahai Rasulullah saw?” Beliau bersabda, “Menolong orang yang membutuhkan lagi teraniaya” Salah seorang sahabat bertanya, “Bagaimana jika ia tidak mampu, wahai Rasulullah saw.?” Beliau menjawab, “Mengajak pada yang ma’ruf atau kebaikan” Salah seorang sahabat bertanya, “Bagaimana jika ia tidak mampu, wahai Rasulullah saw.?” Beliau menjawab, “Menahan diri dari perbuatan buruk, itu merupakan sedekah” (HR. Muslim).

2.3 Rekayasa Perangkat Lunak Berorientasi Penggunaan Ulang

Salah satu model pengembangan perangkat lunak adalah penggunaan kembali komponen. Hal ini sering terjadi secara tidak langsung ketika pengembang mengerjakan proyek dengan mengetahui desain atau kode yang sudah ada dan dibutuhkan. Komponen yang dibutuhkan akan dicari kemudian dimodifikasi sesuai kebutuhan dan menggabungkannya kedalam sistem yang dibuat (Somerville 2011). Di abad 21 proses pengembangan perangkat lunak yang fokus pada penggunaan ulang telah banyak digunakan. Pendekatan berorientasi penggunaan ulang bergantung pada basis komponen perangkat lunak yang dapat digunakan kembali dan *framework* berintegrasi komposisi dengan komponen ini. Terkadang komponen ini adalah sistem dalam hak mereka sendiri (COTS atau *commercial off-the-shelf systems*) yang memberikan fungsional yang spesifik seperti pengolah kata atau *spreadsheet* (Somerville, 2011). Model proses umum untuk pengembangan berorientasi penggunaan ulang ditunjukkan pada Gambar 2.2.



Gambar 2. 1 Tahapan model rekayasa perangkat lunak berorientasi penggunaan ulang

Sumber: Somerville (2011)

Meski tahap awal adalah spesifikasi kebutuhan dan validasi sistem sebanding dengan proses perangkat lunak lainnya. Tahap peralihan dalam penggunaan ulang memiliki proses yang berbeda (Somerville, 2011). Adapun tahapan-tahapan itu adalah:

1. Spesifikasi Kebutuhan dilakukan untuk mendefinisikan kebutuhan fungsional dan nonfungsional yang diperoleh dari hasil pengumpulan data.

2. Analisis komponen dilakukan untuk mendapatkan komponen yang sesuai dengan kebutuhan. Hasil analisis komponen akan diaplikasikan pada kebutuhan yang telah didefinisikan. pada dasarnya, tidak semua fungsi yang

ada pada komponen digunakan seluruhnya, namun hanya beberapa fungsi yang diperlukan saja.

3. Modifikasi kebutuhan mungkin saja dilakukan jika tidak menemukan komponen yang sesuai dari analisis komponen. Kebutuhan dimodifikasikan berdasarkan komponen yang tersedia. Jika modifikasi tidak mungkin dilakukan, maka kegiatan analisis komponen bisa diulang untuk mencari solusi alternatif.

4. *framework* dapat dirancang atau digunakan kembali. Perancang sistem atau *designer* harus mempertimbangkan komponen yang digunakan kembali dan menyesuaikan dengan kerangka sistem. jika komponen penggunaan ulang tidak tersedia, maka memungkinkan akan dilakukan perancangan untuk perangkat lunak yang baru.

5. Tahap pengembangan dan integrasi adalah tahap mengontruksi data, mengimplementasikan fungsi-fungsi yang telah dirancang sebagai arsitektur perangkat lunak, dan menerjemahkan dalam bahasa pemrograman.

6. Tahap validasi sistem bertujuan untuk menguji kelayakan sistem dan untuk membuktikan apakah sistem berjalan sesuai dengan kebutuhan pengguna.

2.4 Component Retrieval

Komponen adalah entitas yang dapat digunakan oleh program yang berbeda. Komponen retrieval adalah komponen yang dapat digunakan kembali dimana komponen dapat disimpan dan diambil sesuai kebutuhan pada aktivitas komponen yang dibutuhkan (Dutta & Sengupta, 2015). Salah satu komponen penggunaan ulang adalah komponen pada *library platform* android. Android adalah salah satu *platform* yang bersifat *open source* dimana sistem pengembangannya bebas dan tersebar. Pengembang atau pengguna dapat menggunakan, mengubah, menambah dan memperbaiki komponen pada *library* yang tersedia. Ada milyaran *library* dan aplikasi yang tersedia untuk mendukung proses pengembangan. Khususnya pada pengembangan yang berorientasi pada komponen penggunaan ulang komponen-komponen yang ada. Sehingga proses pemilihan komponen harus menggunakan metode yang tepat. Pada penelitian ini beberapa komponen penggunaan ulang yang digunakan yaitu komponen aplikasi *map*, komponen *date picker*, komponen *time picker* dan komponen pencarian lokasi.

1. Komponen aplikasi *map*

Aplikasi *map* adalah sebuah panduan yang menentukan posisi dari suatu lokasi. Aplikasi *map* adalah sebuah alat untuk menunjukkan arah dan memulai pencarian suatu lokasi. Ada banyak aplikasi-aplikasi *map* khususnya pada *platform* android yang sedang bekembang dan menyempurnakan proses pengembangan.

Adapun *library* yang digunakan disesuaikan pada kebutuhan perangkat lunak yang dibuat.

2. Komponen *date picker* dan *time picker*

Time picker adalah *widget* antarmuka pengguna grafis yang memungkinkan pengguna memilih tanggal dari kalender atau waktu dari rentang waktu yang ditampilkan. Biasanya, standar tampilan *date picker* dan *time picker* menyediakan bidang kotak teks yang ketika diklik untuk memasukkan tanggal, maka kalender akan ditampilkan di samping atau di bawah bidang yang memungkinkan pengguna untuk mengisi bidang dengan tanggal yang sesuai, atau menyediakan kotak teks dengan ikon kalender sedemikian rupa sehingga ketika ikon diklik, maka kalender (atau bidang waktu) ditampilkan . Pada *library* android tersedia ratusan komponen *date picker* dan *time picker* yang dapat digunakan sesuai dengan kebutuhan perangkat lunak.

3. Komponen pencarian lokasi

Salah satu dari fitur *smartphone* dimana pengguna dapat mengetahui titik kordinat dari suatu lokasi adalah Global Partitioning System (GPS). GPS berperan dalam menampilkan kordinat lokasi yang digunakan untuk mengikatkan data spasial ke dalam sistem koordinat dan data global (Prahasta, 2009). Salah satu pemanfaatan dari GPS adalah pengembangan aplikasi untuk memperoleh informasi lokasi. Pada aplikasi ini dibutuhkan suatu komponen aplikasi yang menampilkan lokasi tertentu untuk melakukan kegiatan sedekah, agar kegiatan dapat dilakukan dengan efisien. Banyaknya aplikasi penyedia informasi lokasi serta fitur-fiturnya membuat pengembang harus memilih aplikasi yang sesuai dengan kebutuhan pengguna.

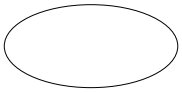
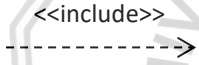
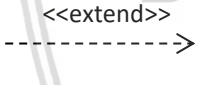
2.5 UML

UML adalah singkatan dari *Unified Modeling Languange,* yaitu sebuah bahasa pemodelan yang digunakan untuk memvisualisasi, merancang, dan mendokumentasikan perangkat lunak. UML adalah sebagai sebuah standar untuk merancang model sebuah sistem. Tujuan penggunaan bahasa UML adalah untuk mendefinisikan kebutuhan, melakukan analisis dan melalukan perancangan untuk menggambarkan arsitektur sistem pada pendekatan pemrograman berorientasi objek (Fowler, 2003). Adapun UML digambarkan pada diagram-diagram yang memiliki fungsi-fungsi yang berbeda (Fowler, 2003):

2.5.1 *Use Case Diagram*

Use case diagram adalah sebuah diagram yang menggambarkan fungsionalitas yang dibutuhkan oleh sistem. Pada diagram ini menekankan pada prilaku aktor sebagai entitas yang melakukan sebuah aktivitas. Notasi-notasi yang ada pada *use case diagram* dapat dilihat pada Tabel 2.2.

Tabel 2. 2 Notasi *use case diagram*

No.	Notasi	Penjelasan
1.		***Use case*** adalah representasi dari setiap fungsional seperti komponen atau *event* sebuah klas. *Use case* menggambarkan fungsionalitas dari sistem, nama *use case* biasanya dinyatakan dengan menggunakan kata kerja.
2.		**Aktor** adalah pelaku dari sebuah sistem yang berada di luar sistem berupa orang, proses atau sistem lain yang berinteraksi dengan sistem
3.		**Asosiasi** adalah sebuah interaksi antara aktor dan *use case*.
4.		**Generalisasi** adalah hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* di mana fungsi yang satu adalah fungsi yang lebih umum dari yang lainnya.
5.		***Include*** adalah sebuah relasi tambahan ke sebuah *use case* yang menyertakan *use case* lain untuk menjalankan fungsinya.
6.		***Extend*** adalah sebuah relasi tambahan. Dimana *use case* dapat berdiri sendiri tanpa tergantung dengan *use case* lain
7.		***Notes*** adalah komentar dalam diagram. *Notes* dapat berdiri sendiri atau dihubungkan dengan garis putus-putus pada elemen yang dikomentari.

Sumber: Fowler (2003)

2.5.2 *Use Case Scenario*

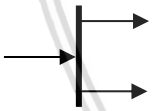
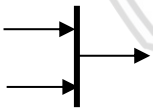
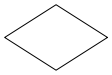
Use case scenario adalah sebuah kejadian yang mengilustrasikan beberapa interaksi yang diusulkan oleh sistem (Arms, 2001). *Use case scenario* menjabarkan secara detail setiap *use case* dari *use case diagram* yaitu nama *use case*, nomor fungsional *use case*, objektif *use case*, aktor, *pre-condition, main flow, alternative flow,* dan *post condition*. Nama *use case* menggunakan kata kerja, nomor fungsional menjelaskan nomor identitas fungsional *use case*, objektif yaitu penjelasan singkat mengenai tujuan *use case,* dan nilai yang ingin didapatkan oleh aktor. *Pre-condition* atau kondisi sebelumnya yaitu kondisi yang perlu ada sebelum *use case* melakukan fungsinya. *Post-condition* atau kondisi sesudah yaitu kondisi yang sudah dipenuhi ketika *use case* telah melakukan fungsinya. *Main flow* yaitu inti dari *use case scenario*. Pada *main flow* digambarkan aliran proses dari awal sampai akhir yang dilakukan oleh aktor dan sistem. Sedangkan *alternative flow*

adalah alur yang memberikan alternatif lain yang tidak termasuk kedalam *main flow*.

2.5.3 *Activity Diagram*

Activity diagram menggambarkan perilaku dinamis suatu sistem atau bagian dari suatu sistem melalui aliran kontrol antara tindakan yang dilakukan sistem. Activity diagram menunjukkan bagaimana perangkat lunak dan aktivitas manusia berinteraksi (Fowler, 2003). Komponen utama *activity diagram* adalah *action node* yang dinotasikan dengan gambar bulat persegi panjang, yang sesuai dengan tugas yang dilakukan oleh sistem perangkat lunak. Panah dari satu *action node* ke *action node* yang lainnya menunjukkan aliran kontrol. Simbol-simbol yang digunakan pada notasi *activity diagram* ditunjukkan pada Tabel 2.3.

Tabel 2. 3 Notasi *activity diagram*

No.	Notasi	Penjelasan
1.		***Activity*** adalah aktivitas yang dieksekusi selama masa transisi, biasanya diawali dengan kata kerja.
2.		***Start State*** adalah status awal sebuah aktivitas
3.		***End State*** adalah status akhir sebuah aktivitas
4.		***Fork*** percabangan dimana satu aliran masuk dan akan mengeluarkan lebih dari satu aliran
5.		***Join*** penggabungan dimana lebih dari satu aliran aktivitas digabungkan menjadi satu.
6.		***Decision*** adalah pilihan untuk mengambil keputusan jika ada pilihan aktivitas lebih dari satu.

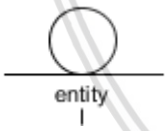
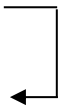
Sumber: Fowler (2003)

2.5.4 *Sequence Diagram*

Sequence diagram adalah diagram yang menggambarkan interaksi antar objek di dalam dan di sekitar termasuk aktor, sistem, dan sebagainya berupa pesan yang dikirimkan oleh aktor atau objek. *Sequence diagram* adalah scenario umum yang digunakan untuk mencari tahu apa yang terjadi dalam perangkat lunak (Fowler, 2003) *Sequence diagram* juga digunakan untuk menggambarkan rangkaian langkah-langkah yang dilakukan sebagai balasan dari sebuah *event* untuk

menghasilkan *output* tertentu. *Sequence diagram* menggambarkan aktivitas objek pada *use case* dengan mendeskripsikan *lifeline* objek dan *message* yang dikirim atau diterima objek. Tabel 2.4 adalah notasi yang ada dalam *sequence diagram.*

Tabel 2. 4 Notasi *sequence diagram*

No.	Notasi	Penjelasan
1.		***Lifeline*** merepresentasikan suatu objek sedang aktif dalam berinteraksi
		***Aktor*** adalah pelaku dari sebuah sistem yang berada di luar sistem berupa orang, proses atau sistem lain yang berinteraksi dengan sistem
2.		***Activation*** merepresentasikan periode pengaktifan dari objek atau komponen yang menyatakan objek sedang berinteraksi.
3.		***Boundary*** merepresentasikan interaksi sistem dengan lingkungan diluar sistem
4.		***Control*** merepresentasikan sebuah relasi antara *boundary* dan *entity*
5.		***Entity*** merepresentasikan entitas yang mewakili model
6.		***Message*** merepresentasikan komunikasi berupa pesan yang dikirimkan oleh objek dengan objek tujuannya.
7.		***Return Message*** merepresentasikan respon atau pesan yang dikirimkan oleh objek dengan objek tujuannya.
8.		***Self Message*** merepresentasikan sebuah operasi rekursif yang memanggil dirinya sendiri.

Sumber: Visual Paradigm (2016)

2.5.5 Diagram Komponen

Diagram komponen dalam UML adalah sebuah diagram yang menggambarkan komponen dan hubungan antar komponen dalam sistem. komponen diagram digunakan saat membagi sistem menjadi komponen-komponen yang menampilkan hubungan antar komponen melalui antarmuka atau penguraian komponen ke tingkat struktur yang lebih rendah (Fowler, 2003). Notasi diagram komponen dapat dilihat pada Tabel 2.5.

Tabel 2. 5 Notasi diagram komponen

No.	Notasi	Penjelasan
1.		**Komponen** adalah unit-unit yang dapat berdiri sendiri atau berhubungan dengan komponen lainnya
2.		***Port*** merupakan titik interaksi antara komponen serta lingkungan
3.		***Interface*** merepresentasikan sebuah deklarasi dari suatu operasi
4.		***Dependency*** adalah hubungan ketergantungan komponen dimana panah mengarah pada komponen yang digunakan
5.		***Link*** adalah hubungan atau relasi antar komponen

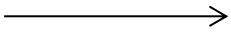
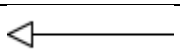
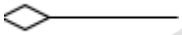
Sumber: Visual Paradigm (2016)

2.5.6 *Class Diagram*

Class diagram adalah salah satu komponen UML yang menggambarkan struktur sistem dengan mendefinisikan klas-klas yang terlibat dalam membangun sistem. *Class diagram* menggambarkan jenis objek dalam sistem dan berbagai jenis hubungan statis. *Class diagram* juga menunjukkan sifat dan operasi klas dan batasan yang berlaku objek terhubung (Fowler, 2003). Adapun simbol-simbol *class diagram* digambarkan pada Tabel 2.6.

Tabel 2. 6 Simbol pada *class diagram*

No.	Notasi	Penjelasan
1.	Nama_class (-) atribut (+) operasi	***Class*** merepresentasikan struktur sistem

		Asosiasi adalah hubungan antar klas secara umum
2.		**Dependensi** merepresentasikan hubungan ketergantungan antar klas
3.		**Generalisasi** merepresentasikan hubungan umum-khusus atau *parent class* dan *child class*
4.		**Komposisi** merepresentasikan hubungan antar klas dimana klas satu bergantung pada klas yang lain
5.		**Agregasi** merepresentasikan hubungan dimana klas satu adalah bagian dari klas lain namun tidak saling bergantung.

Sumber: Fowler (2003)

2.6 Android

Android adalah sistem operasi selular yang menawarkan suatu pendekatan terpadu untuk pengembangan aplikasi. Pengembang perlu mengembangkan aplikasi menggunakan android dan aplikasi pada android dapat berjalan pada berbagai perangkat yang berbeda, sebagai perangkat yang didukung android (Kadibagil, Mahesh and Guruprasad, 2014). Android adalah sistem operasi yang bersifat *open source*. Sistem operasi ini memiliki Lisensi Apache yang sifatnya terbuka dan bebas. Pada awal perintisannya, android dikembangkan oleh Android Inc., kemudian diakuisisi oleh Google sehingga menjadi salah satu bagian dari produk Google. Android dibangun di atas fondasi yang solid dan terbukti yaitu kernel Linux. Linux dibuat oleh Linus Torvalds pada tahun 1991, Linux dapat ditemukan hari ini dalam segala hal dari jam tangan ke superkomputer. Linux menyediakan abstraksi perangkat keras lapisan untuk Android yang memungkinkan android untuk diport ke berbagai macam platform di masa depan (Burnette, 2010).

Adapun kelebihan dan kekurangan android adalah:

1. **Kelebihan android**
   a. *Switching* dan *multitasking:* android sangat mendukung *multitasking* aplikasi, pengguna dapat menggunakan banyak aplikasi tanpa harus menutup atau menunggu aplikasi yang sedang berjalan.
   b. Kapabilitas: android memiliki kapasitas yang lebih baik untuk beragam *widget* yang akan semakin memudahkan dan memanjakan aktivitas pengguna.
   c. *Location service:* salah satu kelebihan sistem operasi android yaitu penyedia lokasi. perangkat android memiliki kemampuan mengakses ke

location service, seperti GPS yang dapat mendeteksi posisi dari perangkat atau *device*. Fitur ini dapat mengambil data pada peta dan menampilkan lokasi.

d. Notifikasi: android dapat menyediakan notifikasi. Perkembangan terbaru dimana pengguna dapat langsung membalas notifikasi yang masuk tanpa harus membuka nofitikasi, misalnya membalas *chatting*.
e. *Drag and drop* dan *multitouch:* android mengizinkan untuk memindahkan data dari satu tempat ke tempat yang lain hanya dengan gerakan *drag and drop* fitur ini sangat mempermudah dan mempercepat aktivitas pengguna.

2. Kekurangan android

a. Koneksi internet yang terus menerus mengakibatkan pengguna harus selalu menyediakan paket GPRS sesuai dengan kebutuhan. Selain itu karena GPRS yang selalu aktif maka baterai yang digunakan akan boros.
b. Aplikasi pada android dapat digunakan secara gratis dan mudah, sehingga konsekuensinya aplikasi android akan selalu diikuti dengan iklan yang banyak.

2.7 Firebase Database Realtime

Firebase Realtime Database adalah *database cloud-hosted* yang disimpan sebagai JSON dan disinkronisasikan secara *realtime* ke setiap *client* yang terhubung. *Client* yang terhubung akan otomatis menerima pembaruan dengan data terbaru. *firebase realtime database* memungkinkan pengguna membuat aplikasi kolaboratif yang mengizinkan akses aman ke basis data langsung dari kode sisi *client*. Data akan tetap ada secara lokal dan bahkan saat *offline*. *Realtime* akan terus berlanjut hingga memberi kesan pengguna responsif terhadap pengguna akhir. Saat perangkat mendapatkan koneksi kembali, *database realtime* akan melakukan sinkronisasi perubahan data lokal dengan pembaharuan jarak jauh bahkan saat *client* dalam kondisi *offline* (Google Firebase, n.d).

Database realtime menyediakan bahasa yang fleksibel, *expression-base rules* yang disebut aturan keamanan *firebase realtime database* untuk menentukan bagaimana data harus terstruktur dan kapan data dapat dibaca atau ditulis. Bila terintegrasi dengan *firebase authentication*, pengembang dapat menentukan siapa yang memiliki akses ke data apa dan bagaimana untuk dapat mengaksesnya.

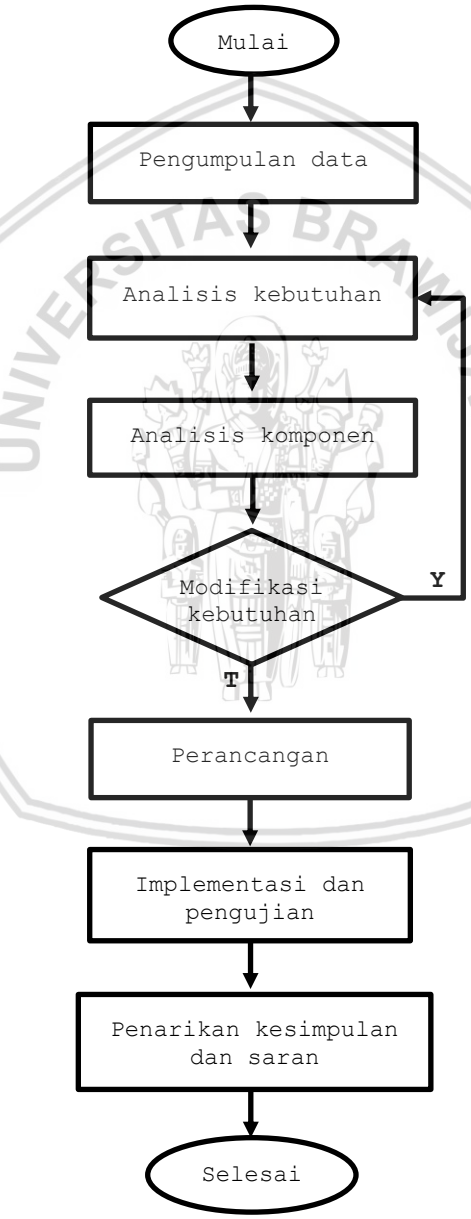
Database realtime adalah basis data NoSQL dan karena itu memiliki optimasi dan fungsionalitas yang berbeda dibandingkan dengan basis data relasional. API (*Application Programming Interface*) pada *database realtime* dirancang untuk memungkinkan operasi yang cepat. Memungkinkan untuk melayani jutaan pengguna tanpa mengorbankan responsif. Karena itu, penting untuk memikirkan bagaimana pengguna perlu mengakses data dan menyusunnya.

2.8 Pengujian

Pengujian perangkat lunak adalah serangkaian langkah di mana penguji dapat menempatkan teknik desain uji spesifik dan metode pengujian harus ditentukan untuk proses perangkat lunak (Pressman, 2008). Pengujian dilakukan untuk mendapatkan nilai kualitas dari perangkat lunak. Pada pengujian perangkat lunak ada beberapa level tingkatan pengujian sebagai tahapan pengujian diantaranya pengujian unit, pengujian integrasi, pengujian validasi dan pengujian sistem. Pengujian unit dan pengujian integrasi adalah model pengujian yang menggunakan metode *white box testing* dan *black box testing*. *White box testing* adalah model pengujian yang memperhatikan struktur program sehingga kesalahan akan sangat jelas terlihat. Pengujian validasi adalah pengujian kebutuhan. Setiap kebutuhan diuji dengan metode *black box testing*. *Black box testing* adalah pengujian dengan memperhatikan *input* pengguna dan *output* sistem. *Black box testing* tidak memperhatikan struktur kode. Pengujian harus memastikan bahwa setiap fungsi harus berjalan sesuai dengan rencana uji atau kasus uji yang telah didefinisikan.

BAB 3 METODOLOGI

Metode penelitian adalah tahapan-tahapan atau langkah-langkah yang digunakan dalam pengembangan Aplikasi Manajemen Informasi Sedekah Berbagi Makanan. Metode penelitian ini dilakukan hingga menghasilkan karya ilmiah dan produk perangkat lunak. Metode pengembangan yang digunakan pada aplikasi manajemen informasi sedekah ini adalah rekayasa perangkat lunak berorientasi penggunaan ulang. Adapun langkah-langkah dari metode pengembangan rekayasa perangkat lunak berorientasi penggunaan ulang ditunjukkan pada Gambar 3.1.



Gambar 3. 1 Kerangka kerja sistem manajemen informasi sedekah berbagi makanan

3.1 Pengumpulan Data

Proses pengumpulan data dilakukan untuk mendapatkan data-data yang akan digunakan untuk mendeskripsikan kebutuhan-kebutuhan fungsional dan nonfungsional oleh pengguna atau *stakeholder*. Pengumpulan data ini dapat dilakukan dengan cara:

3.1.1 Studi Literatur

Pada penelitian ini memerlukan studi literatur dari dasar teori yang ada. Landasan kepustakaan disusun berdasarkan beberapa referensi yang didapatkan berdasarkan jurnal, buku, maupun hasil bimbingan dengan pembimbing skripsi secara langsung. Berikut merupakan studi literatur yang dibutuhkan sebagai dasar dalam penelitian ini:

1. Tinjauan pustaka
2. Sedekah
3. Rekayasa perangkat lunak berorientasi penggunaan ulang
4. *Component Retrieval*
5. UML
    a. *Use case*
    b. *Use case Scenario*
    c. *Activity Diagram*
    d. *Sequence Diagram*
    e. *Class Diagram*
6. Android
7. Firebase Realtime Database
8. Pengujian

3.1.2 Survei

Salah satu metode pengumpulan data yang banyak digunakan adalah survei. Survei dilakukan untuk mendapatkan kebutuhan pengguna. Survei dilakukan dengan memberikan kuisioner kepada target responden. Survei yang digunakan untuk mengetahui seberapa besar keinginan masyarakat untuk melakukan sedekah berbagi makanan serta kendala-kendala yang pernah dialami dalam melakukan kegiatan sedekah. Aplikasi survei yang digunakan adalah aplikasi Google Form dari Google Development. Target responden survei adalah masyarakat secara umum dengan berbagai latar belakang pekerjaan. Survei ini dilakukan untuk mengetahui seberapa penting aplikasi yang menampilkan informasi kegiatan sedekah. Survei ini juga dilakukan untuk mengetahui intensitas penggunaan *smartphone* dengan sistem operasi android.

3.2 Analisis Kebutuhan

Pada tahap ini, dilakukan survei untuk mendapatkan kebutuhan sistem. data-data hasil survei akan dideskripsikan dalam spesifikasi kebutuhan dan digambarkan dalam pemodelan kebutuhan.

Spesifikasi kebutuhan menggambarkan gambaran umum dari komponen-komponen sistem yang terlibat. Tahap ini juga akan menjelaskan karakteristik pengguna sebagai aktor yang terlibat dan berinteraksi dalam sistem. poin utama pada tahap analisis kebutuhan adalah mendefinisikan kebutuhan fungsional dan kebutuhan nonfungsional. Kebutuhan fungsional dan nonfungsional didapatkan dari proses pengumpulan data yang dilakukan sebelumnya.

Tahap pemodelan adalah tahapan dimana kebutuhan yang telah didefinisikan sebelumnya dimodelkan untuk persiapan proses perancangan. Adapun tujuan lain tahap pemodelan adalah untuk memberikan gambaran secara umum kepada pengguna dan *stakeholder* tentang sistem yang akan dibuat. Proses pada tahap analisis kebutuhan ini antara lain:

1. Membuat gambaran umum sistem untuk mempermudah memahami proses berjalannya suatu perangkat lunak secara umum dan komponen sistem lain yang digunakan.
2. Menentukan karakteristik pengguna untuk dapat membedakan akses yang didapatkan dari setiap pengguna, setelah itu melakukan analisis kebutuhan pengguna dan mendefinisikan kebutuhan tersebut serta membuat spesifikasi kebutuhan yang diidentifikasi dengan pemberian nomor dari setiap kebutuhan.
3. Hasil dari analisis kebutuan dibedakan menjadi dua kategori, yaitu kebutuhan fungsional dan nonfungsional.
4. Melakukan pemodelan kebutuhan berdasarkan setiap kebutuhan yang didefinisikan. Diantaranya adalah *use case diagram*, *use case scenario*, *activity diagram* dan *user interface.*

3.3 Analisis Komponen

Tahap analisis komponen adalah tahapan yang dilakukan setelah mendapatkan spesifikasi kebutuhan dari proses analisis kebutuhan sebelumnya. Tahap analisis komponen dilakukan beberapa proses untuk mendapatkan komponen yang sesuai dengan kebutuhan. Penggunaan ulang yang berhasil, membutuhkan beragam komponen yang berkualitas tinggi, mekanisme klasifikasi, mekanisme *retrieval* yang tepat, dan dokumentasi komponen yang tepat, sarana untuk menggabungkan komponen, dan sarana untuk menyesuaikan komponen dan kebutuhan spesifik.

Analisis komponen dilakukan dengan mendefinisikan *component retrieval*. Pada *component retrieval* akan memberikan gambaran komponen yang akan digunakan kembali. Komponen yang akan digunakan kembali harus sesuai dengan kriteria komponen, sehingga dapat dilakukan proses pengambilan komponen yang tepat.

3.4 Modifikasi Kebutuhan

Modifikasi kebutuhan mungkin saja dilakukan jika hasil analisis komponen tidak ditemukan komponen yang sesuai atau komponen tidak tersedia. Selanjutnya kebutuhan akan dimodifikasikan berdasarkan komponen yang tersedia. Pada aplikasi manajemen informasi sedekah, komponen yang akan digunakan diantaranya adalah komponen aplikasi *map*, komponen *date picker* dan *time picker* dan komponen pencarian lokasi. Komponen-komponen penggunaan ulang tersebut akan dipilih dan ditentukan berdasarkan kriteria. Jika komponen-komponen tersebut telah ditemukan dan sesuai dengan kebutuhan, maka modifikasi kebutuhan tidak perlu dilakukan. Namun sebaliknya, jika dalam proses analisis komponen telah ditentukan dan dipilih komponen penggunaan ulang yang tidak sesuai dengan kebutuhan, maka harus dilakukan modifikasi kebutuhan. Komponen yang telah ditemukan untuk digunakan kembali selanjutnya akan dimasukkan dalam perancangan sistem.

3.5 Perancangan

Tahap perancangan adalah tahap yang dilakukan setelah melakukan analisis komponen dengan merancang *component retrieval* yang telah dipilih. Perancangan sistem dibuat berdasarkan analisis kebutuhan. Perancangan pada aplikasi manajemen informasi sedekah ini meliputi *sequence diagram,* diagram komponen, *class diagram,* perancangan algoritme dan perancangan basis data. *Sequence diagram* dibuat sebagai alur proses dalam sistem, pada perancangan *sequence diagram* ini hanya dibuat 3 sampel diagram. Pada diagram komponen dibuat berdasarkan hasil proses *component retrieval. Class diagram* dibuat untuk mendefinisikan klas-klas yang akan digunakan pada tahap implementasi, serta perancangan basis data yang dibuat sebagai acuan dalam implementasi basis data.

3.6 Implementasi dan Pengujian

Tahap implementasi dan pengujian adalah tahap yang dilakukan setelah tahap pembuatan perancangan. Perancangan tersebut sebagai acuan dalam proses implementasi. Pada aplikasi ini menggunakan arsitektur Android Studio dengan bahasa pemrograman XML dan Java dalam implementasinya. Tahap implementasi mencakup spesifikasi sistem perangkat keras dan perangkat lunak untuk mendefiniskan spesifikasi perangkat yang digunakan untuk membuat aplikasi. Tahap ini juga dilakukan implementasi antarmuka, implementasi kode program dan implementasi basis data.

Tahap pengujian dilakukan untuk mengetahui apakah perangkat lunak yang dibangun sudah memenuhi kebutuhan atau belum. Pengujian yang dilakukan berupa pengujian unit dan pengujian integrasi menggunakan *white-box testing* serta pengujian validasi menggunakan *black-box testing*. Pengujian unit dan pengujian integrasi dilakukan dengan menguji masing-masing *test case* jalur algoritme serta pengujian validasi pada kebutuhan fungsional dan nonfungsional akan dilakukan dengan membandingkan antara *expected result* dan hasil. Hasil

pengujian ini akan menjadi acuan apakah aplikasi yang dikembangkan sudah sesuai dengan kebutuhan atau tidak.

3.7 Penarikan Kesimpulan dan Saran

Tahapan terakhir dari penelitian ini adalah penarikan kesimpulan dan saran. Aktivitas ini sangat berguna dalam merangkum hasil akhir dari penelitian, yaitu berkaitan dengan nilai kualitas dan kelayakan aplikasi. Penelitian ini juga dapat digunakan sebagai acuan pada penelitian serupa selanjutnya.

BAB 4 ANALISIS KEBUTUHAN

Tahap analisis kebutuhan dimulai dengan melakukan survei kepada masyarakat umum yang memiliki latar belakang pekerjaan yang berbeda-beda. Survei dilakukan untuk mengetahui kendala-kendala apa saja yang dialami masyarakat ketika akan melakukan sedekah. Survei juga dilakukan untuk mengetahui seberapa penting adanya sebuah aplikasi penyedia informasi sedekah. Servei dilakukan dengan memberikan kuisioner melalui aplikasi google form kepada masyarakat umum dengan minimal responden 30 orang. Dari data-data survei ini akan menjadi bahan peneliti untuk mendefinisikan kebutuhan sistem yang akan dideskripsikan dalam spesifikasi kebutuhan. Tahap selanjutnya dari analisis kebutuhan adalah membuat spesifikasi kebutuhan dan pemodelan kebutuhan.

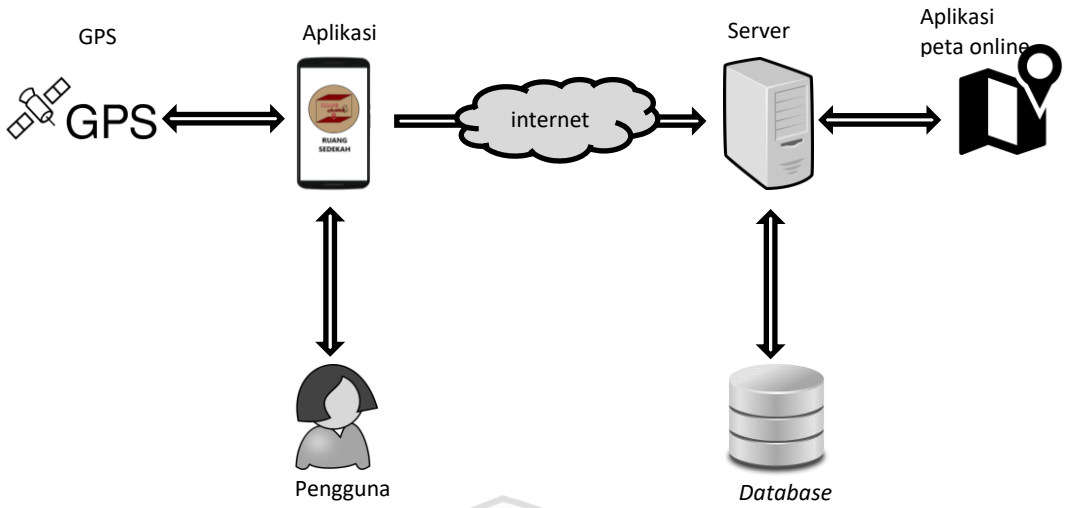
4.1 Spesifikasi Kebutuhan

Tahapan spesifikasi kebutuhan dilakukan untuk memenuhi kebutuhan yang diinginkan. Spesifikasi kebutuhan terdiri dari gambaran umum perangkat lunak, karakteristik pengguna serta kebutuhan fungsional dan nonfungsional.

4.1.1 Gambaran Umum Sistem

Aplikasi manajemen informasi sedekah berbagi makanan adalah aplikasi berbasis android yang diperuntukkan bagi seluruh kalangan masyarakat. Aplikasi ini dibuat dengan arsitektur android dan bahasa pemrograman java. Salah satu kelebihan dari arsitektur android adalah *library* yang banyak dan dapat digunakan ulang tanpa harus membuatnya dari awal, sehingga dapat mengurangi waktu pengerjaan dan biaya pengeluaran.

Komponen-komponen yang bersifat *reusable* didefinisikan kembali dan jika dibutuhkan membuat komponen yang baru untuk melengkapi komponen yang lama. Gambaran umum sistem manajemen informasi sedekah berbagi makanan dapat dilihat pada Gambar 4.1.



Gambar 4. 1 Gambaran umum sistem manajemen informasi sedekah berbagi makanan

Berikut penjelasan mengenai masing-masing entitas pada Gambar 4.1.

a. Pengguna

Pengguna adalah entitas yang langsung berinteraksi dengan aplikasi. Pengguna dapat langsung memanfaatkan fitur yang terdapat pada aplikasi manajemen informasi sedekah berbagi makanan ini. Pengguna dapat melakukan permintaan sesuai dengan fungsi yang ada dan menerima respon atau *output* dari permintaan secara langsung melalui aplikasi pada *smartphone.*

b. Aplikasi

Aplikasi dibagi menjadi dua, yaitu *front-end* (komponen *client*) dan *back-end* (komponen *server*). Komponen *client* menerima masukan dari pengguna dan kemudian menggunakan teknologi tertentu untuk sampai pada komponen *server* dengan implementasi kode dan penerjemahan bahasa komputer.

c. *Server*

Server adalah entitas yang berkomunikasi dengan aplikasi pada sisi *back-end* atau komponen *server*. Tugas utama *server* adalah melayani komputer *client*. *Server* juga dapat digunakan untuk menyimpan berbagai macam aplikasi yang diakses oleh *client* dan menyimpan data dengan cara berkomunikasi dengan basis data.

d. *Database*

Database adalah entitas yang digunakan untuk menyimpan data dan informasi secara sistematis dan dapat dimanipulasi.

e. Internet

Internet adalah seluruh jaringan komputer yang saling terhubung yang melayani milyaran pengguna di dunia. Internet menghubungkan semua komponen untuk bertukar data.

f. GPS

GPS Adalah sistem navigasi berbasis satelit yang dirancang untuk menyediakan pelacakan posisi dan informasi tempat di muka bumi serta pengaturan waktu (Pendleton, 2002). Penggunaan GPS pada aplikasi ini adalah menentukan posisi tempat untuk melakukan sedekah dengan bantuan aplikasi peta online.

g. Aplikasi peta *online*

aplikasi peta *online* memberikan layanan bagi pengguna, untuk menunjukkan arah dari sebuah lokasi.

4.1.2 Karakteristik Pengguna

Karakteristik pengguna bertujuan untuk mengidentifikasikan dan memberi gambaran pengguna yang terlibat dalam sistem. berikut adalah karakteristik pengguna yang digambarkan pada Tabel 4.1.

Tabel 4. 1 Karakteristik pengguna

No.	Identifikasi Pengguna	Karakteristik
1.	*Member*	*Member* merupakan pengguna yang telah mendaftarkan diri sebagai pengguna aktif yang akan melakukan input informasi kegiatan dan menerima beberapa fitur-fitur eksklusif lainnya
2.	Pengunjung	Merupakan pengguna yang hanya dapat melihat informasi kegiatan sedekah berbagi makanan

4.1.3 Kebutuhan Fungsional dan Kebutuhan Nonfungsional

Kebutuhan fungsional dan kebutuhan nonfungsional didapatkan dari proses pengumpulan data yang dilakukan untuk mengidentifikasi kebutuhan pengguna. Kebutuhan fungsional menggambarkan layanan yang disediakan sistem kepada pengguna terhadap suatu inputan tertentu. Sedangkan kebutuhan nonfungsional memberikan batasan-batasan layanan yang ditawarkan sistem sesuai dengan kebutuhan pengguna. Pada kebutuhan fungsional ini akan diberikan kode kebutuhan yaitu RS-F-XX dan untuk kebutuhan nonfungsional akan diberikan kode RS-NF-XX. RS adalah singkatan dari nama aplikasi yaitu Ruang Sedekah, F dan NF merupakan singkatan dari Fungsional dan Nonfungsional, sedangkan XX adalah nomor kebutuhan. Kebutuhan fungsional pada sistem ini dibuat berdasarkan kebutuhan masing-masing aktor. Definisi kebutuhan fungsional dan nonfungsional dapat dilihat pada Tabel 4.2 sampai dengan Tabel 4.4.

4.1.3.1 Kebutuhan Fungsional

a. *Member*

Tabel 4. 2 Kebutuhan fungsional *member*

No. Kebutuhan	Nama Kebutuhan	Kebutuhan fungsional	*Use case*
RS-F-1.1	Membuat informasi kegiatan sedekah	*Member* dapat membuat informasi kegiatan sedekah berbagi makanan	Membuat informasi sedekah
RS-F-1.2	Menghapus informasi kegitan sedekah	*Member* dapat menghapus informasi kegiatan sedekah berbagi makanan	Menghapus informasi sedekah
RS-F-1.3	Mengedit informasi kegiatan sedekah	*Member* dapat mengedit informasi kegiatan sedekah berbagi makanan	Mengedit informasi sedekah
RS-F-1.4	*Logout* sistem	*Member* dapat melakukan *logout* atau keluar dari sistem	*Logout*

b. Pengunjung

Tabel 4. 3 Kebutuhan fungsional pengunjung

No. Kebutuhan	Nama Kebutuhan	Kebutuhan fungsional	*Use case*
RS-F-2.1	Melihat informasi kegiatan sedekah	Pengunjung dapat melihat informasi kegiatan sedekah berbagi makanan dalam bentuk detail informasi pada *marker* atau berdasarkan daftar kegiatan pada *timeline*	Melihat informasi sedekah
RS-F-2.2	Memfilter informasi kegiatan sedekah	Pengunjung dapat memfilter informasi berdasarkan lokasi kurang dari 10 km, dan waktu pelaksanaan kegiatan kurang dari 30 hari	Memfilter informasi sedekah
RS-F-2.3	Registrasi	Pengunjung dapat melakukan registrasi untuk menjadi *member*	Registrasi
RS-F-2.4	Login sistem	Pengunjung dapat melakukan *login* pada sistem	*Login*

4.1.3.2 Kebutuhan Nonfungsional

Tabel 4. 4 Kebutuhan nonfungsional

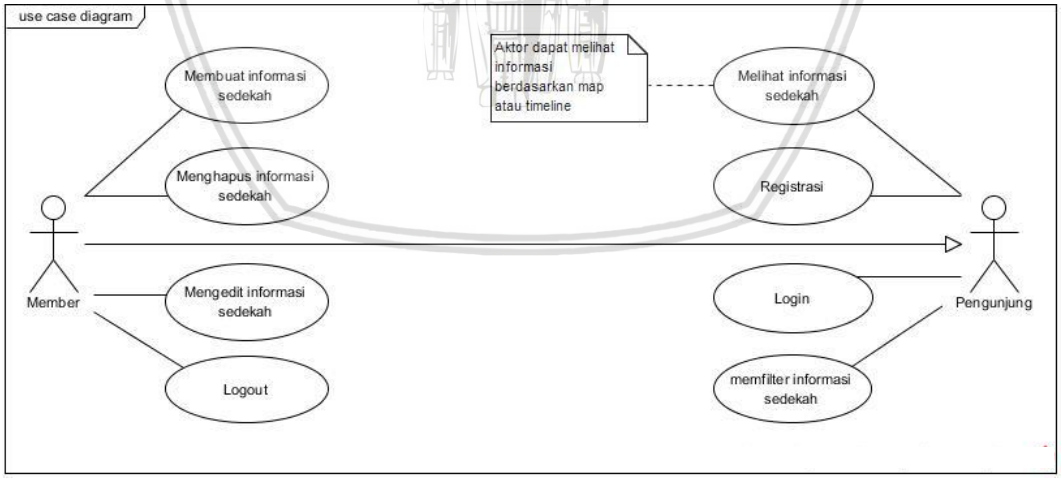
No. Kebutuhan	Kebutuhan nonfungsional	Paramater
RS-NF-1.1	Aplikasi dapat menghasilkan antarmuka yang mudah digunakan dengan target kelayakan > 60%	*Usability*
RS-NF-1.2	Aplikasi dapat berjalan pada level SDK 16-26	*Compatibility*

4.2 Pemodelan Kebutuhan

Tahap pemodelan kebutuhan diperlukan untuk meningkatkan pemahaman terhadap kebutuhan yang sedang dianalisis. Pada penelitian ini pemodelan kebutuhan yang digunakan adalah berbasis *object oriented*. Pemodelan OO (*Object Oriented*) digunakan karena memiliki konsep *reusability*. Adapun Pemodelan kebutuhan *object oriented* pada aplikasi manajemen informasi sedekah meliputi *use case diagram*, *use case scenario* dan *activity diagram*.

4.2.1 *Use Case Diagram*

Use case diagram digunakan untuk memodelkan aspek perilaku sistem berdasarkan kebutuhan fungsional dari aplikasi manajemen informasi sedekah ini. *Use case diagram* ini menggambarkan aktivitas apa yang dapat dilakukan oleh aktor serta hubungannya dengan *use case*. Adapun *use case diagram* dari aplikasi manajemen informasi sedekah berbagi makanan dapat dilihat pada Gambar 4.2.



Gambar 4. 2 *Use case diagram* sistem manajemen informasi sedekah

4.2.2 *Use Case Scenario*

Use case scenario dari aplikasi manajemen informasi sedekah berbagi makanan bertujuan untuk menjabarkan aliran utama (*main flow*) dan tanggapan sistem ketika dikenal satu *case* yang diberikan oleh aktor. Detail *use case scenario* dapat dilihat pada Tabel 4.5 sampai dengan Tabel 4.12.

4.2.2.1 *Use Case Scenario* Membuat Informasi Kegiatan Sedekah

Tabel 4. 5 *Use case scenario* membuat informasi kegiatan sedekah

Nama Use Case	Membuat informasi sedekah
Kode Use Case	RS-UC-1.1
Objektif	Aktor dapat membuat informasi kegiatan sedekah berbagi makanan
Aktor	*Member*
Pre-Condition	Aktor sudah masuk ke halaman utama dengan *privilege* sebagai *member*
Main Flow	1. Aktor mengklik tombol (+) pada halaman activity_buat_kegiatan 2. Sistem menampilkan halaman membuat informasi 3. Aktor memilih tanggal kegiatan dengan mengklik tombol "Pick Date" 4. Sistem menampilkan tanggal yg dipilih 5. Aktor mengisi nama kegiatan pada *field* nama kegiatan 6. Aktor mengatur waktu kegiatan berupa jam dan menit dengan klik tombol jam 7. Sistem menampilkan waktu berupa jam dan menit 8. Aktor menentukan lokasi kegiatan dengan klik tombol lokasi 9. Sistem menampilkan lokasi kegiatan 10. Aktor mengklik tombol "Select" 11. Aktor menulis deskripsi kegiatan pada *field* deskripsi 12. Aktor menekan tombol "Submit" 13. Sistem mengecek data kegiatan 14. Sistem menyimpan informasi kegiatan ke dalam *database* dan menampilkan pesan "Kegiatan baru ditambahkan" 15. Sistem menampilkan *marker* 16. Aktor mengklik *marker*
Alternative Flow	1. Jika tidak mengisi *field* nama kegiatan maka akan menampilkan pesan "Masukkan Nama Kegiatan" 2. Jika tidak mengisi *field* tanggal maka akan menampilkan pesan "Pilih Tanggal Kegiatan" 3. Jika tidak mengisi *field* waktu maka akan menampilkan pesan "Pilih Waktu Kegiatan"

	4. Jika tidak mengisi *field* Lokasi Kegiatan maka akan menampilkan pesan "Pilih Lokasi Kegiatan"
Post Condition	Menampilkan detail informasi pada *marker*

4.2.2.2 *Use Case Scenario* Menghapus Informasi Kegiatan Sedekah

Tabel 4. 6 *Use case scenario* menghapus informasi kegiatan sedekah

Nama Use Case	Menghapus informasi kegiatan sedekah
Kode Use Case	RS-UC-1.2
Objektif	Aktor dapat menghapus informasi kegiatan sedekah berbagi makanan
Aktor	*Member*
Pre-Condition	Aktor sudah masuk ke halaman profil
Main Flow	1. Aktor memilih informasi yang akan dihapus 2. Aktor menekan tombol "Hapus" 3. Sistem menampilkan dialog konfirmasi penghapusan informasi 4. Aktor mengklik tombol "ya" 5. Sistem menghapus informasi kegiatan dari *database* 6. Sistem menampilkan pesan "informasi kegiatan dihapus"
Alternative Flow	1. Aktor mengklik tombol "tidak" sistem tidak menghapus kegiatan
Post Condition	Berhasil menghapus *marker* dan daftar kegiatan pada *timeline*

4.2.2.3 *Use Case Scenario* Mengedit Informasi Kegiatan Sedekah

Tabel 4. 7 *Use case scenario* mengedit informasi kegiatan sedekah

Nama Use Case	Mengedit informasi kegiatan sedekah
Kode Use Case	RS-UC-1.3
Objektif	Aktor dapat melakukan edit informasi kegiatan sedekah berbagi makanan
Aktor	*Member*
Pre-Condition	Aktor sudah masuk ke halaman profil
Main flow	1. Aktor memilih informasi yang akan diedit 2. Aktor menekan tombol "edit" 3. Sistem menampilkan halaman membuat informasi

	4. Aktor mengedit informasi 5. Aktor menekan tombol “Submit” 6. Sistem mengecek data kegiatan 7. Sistem menyimpan data baru ke *database* dan menampilkan data “informasi berhasil diedit” 8. Sistem menampilkan *marker* 9. Aktor mengklik *marker*
Alternative Flow	1. Jika tidak mengisi *field* nama kegiatan maka akan menampilkan pesan “Masukkan Nama Kegiatan” 2. Jika tidak mengisi *field* tanggal maka akan menampilkan pesan “Pilih Tanggal Kegiatan” 3. Jika tidak mengisi *field* waktu maka akan menampilkan pesan “Pilih Waktu Kegiatan” 4. Jika tidak mengisi *field* Lokasi Kegiatan maka akan menampilkan pesan “Pilih Lokasi Kegiatan”
Post Condition	Menampilkan detail informasi baru pada *marker*

4.2.2.4 *Use Case Scenario Logout* Sistem

Tabel 4. 8 *Use case scenario logout* sistem

Nama Use Case	*Logout* sistem
Kode Use Case	RS-UC-1.4
Objektif	Aktor dapat melakukan *logout* atau keluar dari sistem
Aktor	*Member*
Pre-Condition	Aktor sudah masuk ke halaman utama
Main flow	1. Aktor mengklik menu “Logout” 2. Sistem menghapus *member session* dan menampilkan halaman awal 3. Menampilkan pesan “Logout”
Alternative Flow	-
Post Condition	Kembali ke halaman awal dan menghapus *session member*

4.2.2.5 *Use Case Scenario* Melihat Informasi Kegiatan Sedekah

Tabel 4. 9 *Use case scenario* melihat informasi kegiatan sedekah

Nama Use Case	Melihat informasi kegiatan sedekah
Kode Use Case	RS-UC-2.1
Objektif	Aktor dapat melihat informasi kegiatan sedekah berbagi makanan dalam bentuk detail informasi pada *marker* atau berdasarkan daftar kegiatan pada *timeline*

Aktor	Pengunjung
Pre-Condition	Aktor berada pada halaman awal
Main flow	1. Aktor mengklik *link* "Lewati ke Halaman Utama" 2. Sistem menampilkan halaman utama berupa halaman *map* 3. Aktor mengklik *marker*
Alternative Flow	1. Aktor mengklik halaman *timeline* 2. Sistem menampilkan halaman *timeline*
Post Condition	Menampilkan detail informasi pada *marker* dan menampilkan daftar informasi kegiatan pada *timeline*

4.2.2.6 *Use Case Scenario* Memfilter Informasi Kegiatan Sedekah

Tabel 4. 10 *Use case scenario* memfilter informasi kegiatan sedekah

Nama Use Case	Memfilter informasi kegiatan sedekah
Kode Use Case	RS-UC-2.2
Objektif	Aktor dapat memfilter informasi sedekah berdasarkan lokasi kurang dari 10 km dan waktu pelaksanaan kegiatan kurang dari 30 hari
Aktor	*Member*
Pre-Condition	Aktor sudah masuk ke halaman *timeline* dengan menampilkan semua informasi kegiatan
Main flow	1. Aktor mengklik tombol filter 2. Aktor memilih filter berdasarkan lokasi
Alternative Flow	1. Aktor memilih filter berdasarkan waktu
Post Condition	Menampilkan hasil filter informasi

4.2.2.7 *Use Case Scenario* Registrasi

Tabel 4. 11 *Use case scenario* registrasi

Nama Use Case	Registrasi
Kode Use Case	RS-UC-2.3
Objektif	Aktor dapat malakukan registrasi untuk menjadi *member*
Aktor	Pengunjung
Pre-Condition	Aktor masuk ke halaman awal

Main flow	1. Aktor mengklik tombol "Sign Up" 2. Sistem menampilkan halaman registrasi 3. Aktor mengisi *field* yang tersedia 4. Aktor menekan tombol "Sign Up" 5. Sistem mengecek data registrasi 6. Sistem menyimpan data *member* ke dalam *database* dan menampilkan pesan "Pengguna berhasil ditambahkan"
Alternative Flow	1. Jika aktor tidak mengisi salah satu *field* maka akan menampilkan pesan "Masukkan (*field* yang kosong)" 2. Jika *password* kurang dari 6 karakter maka akan menampilkan pesan *"Password* terlalu pendek, masukan minimal 6 karakter" 3. Jika aktor tidak mengisi semua *field* maka akan menampilkan pesan "registrasi gagal"
Post Condition	Menampilkan halaman utama dan masuk sebagai *member*

4.2.2.8 *Use Case Scenario Login* Sistem

Tabel 4. 12 *Use case scenario login* sistem

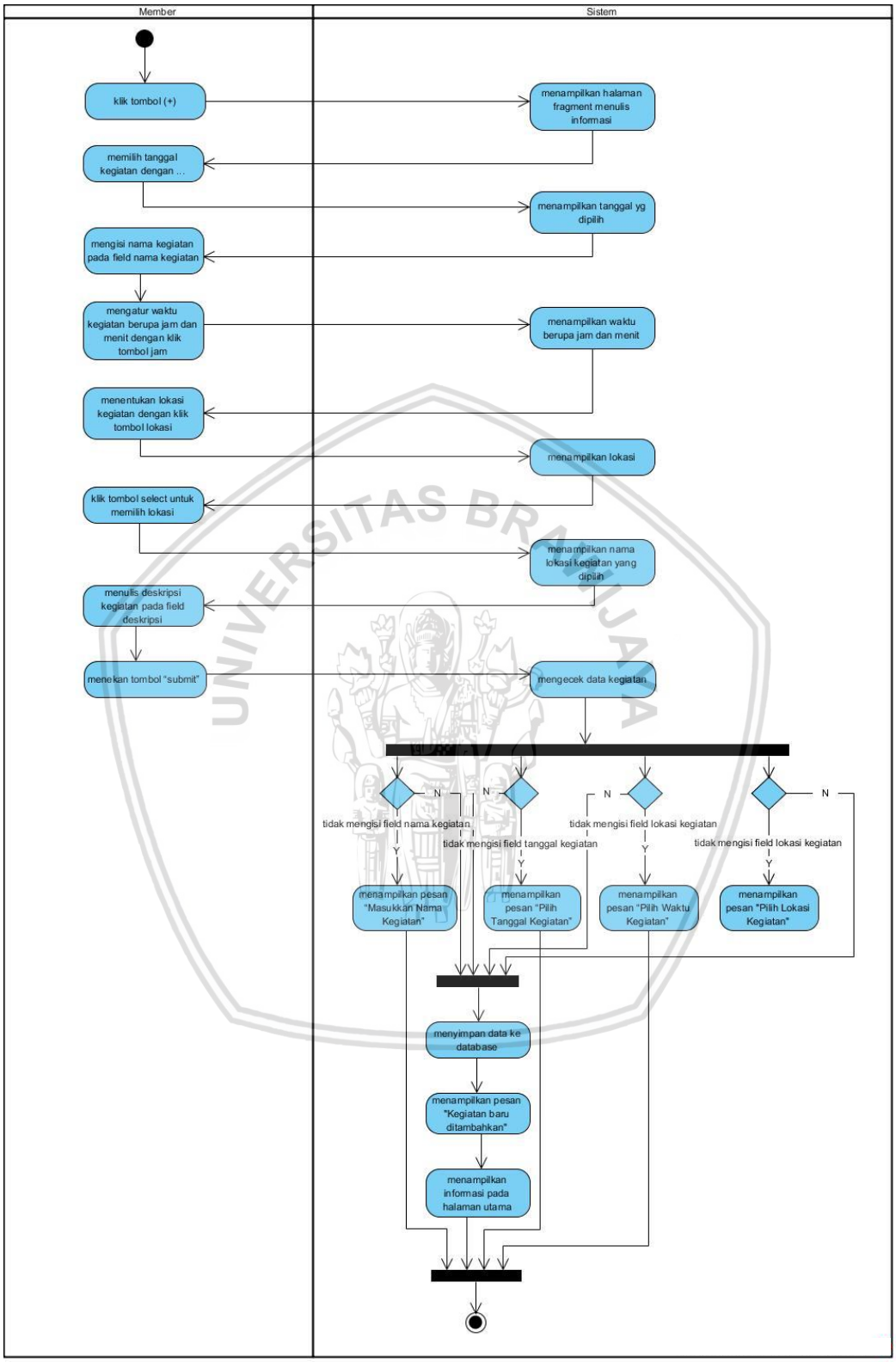
Nama Use Case	*Login* sistem
Kode Use Case	RS-UC-2.4
Objektif	Aktor dapat melakukan *login* atau masuk ke dalam sistem
Aktor	Pengunjung
Pre-Condition	Aktor berada di halaman awal
Main flow	1. Aktor mengklik tombol "Sign In" 2. Sistem menampilkan halaman *login* 3. Aktor mengisi *username* dan *password* 4. Aktor mengklik tombol "Sign in" 5. Sistem mengecek data *member* pada *database* 6. Sistem menampilkan halaman utama dan menampilkan pesan "Login Berhasil (email)"
Alternative Flow	1. Jika email kosong maka akan menampilkan pesan "Masukkan email" 2. Jika *password* kosong maka akan menampilkan pesan "Masukkan *password*" 3. Jika email atau *password* salah maka akan menampilkan "Email atau *Password* Salah, Periksa Lagi"

Post Condition	Menampilkan halaman utama dan masuk sebagai *member*

4.2.3 *Activity Diagram*

Activity diagram dari aplikasi manajemen informasi sedekah berbagi makanan bertujuan untuk menggambarkan aktivitas yang terjadi pada sistem maupun aktivitas oleh aktor. *Activity diagram* menggambarkan aktivitas aktor dan sistem secara garis besar dengan menggambarkan alur kontrol. *Activity diagram* aplikasi manajemen informasi sedekah dapat dilihat pada Gambar 4.3 sampai dengan Gambar 4.10.

4.2.3.1 *Activity Diagram* Membuat Informasi Kegiatan Sedekah

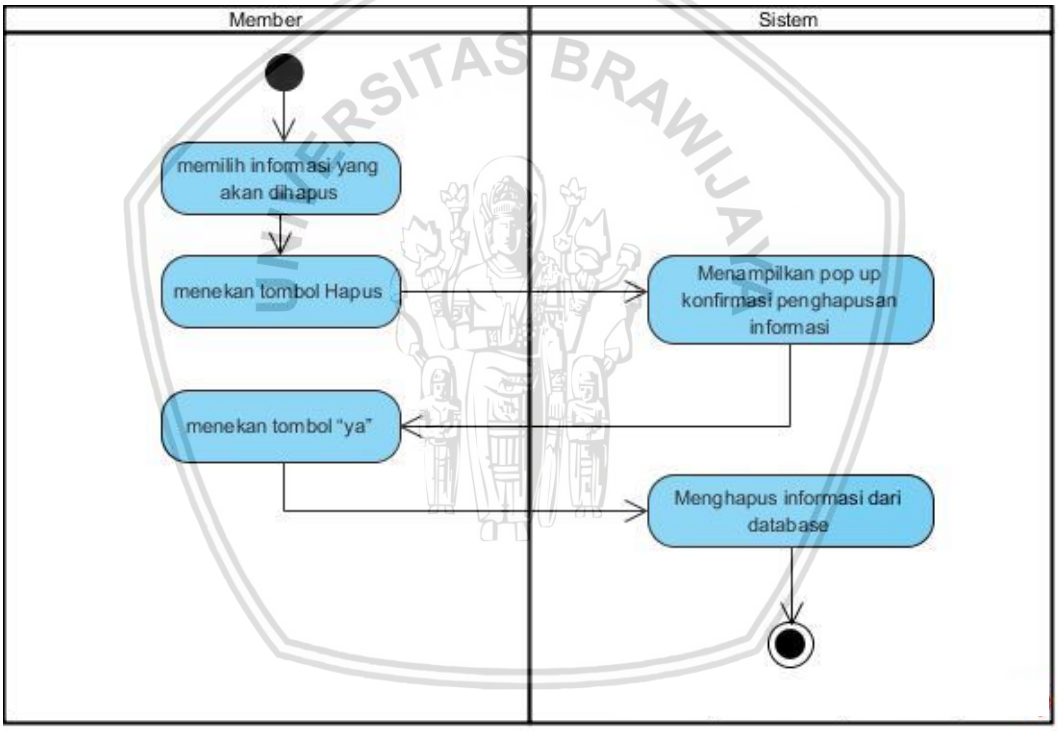


Gambar 4. 3 Gambar *activity diagram* membuat informasi kegiatan sedekah

Gambar 4.3 adalah diagram aktivitas dari fungsional membuat informasi. Awalnya aktor *member* akan mengklik tombol (+) pada halaman utama dan

menampilkan halaman menulis informasi. Setelah halaman Menulis Informasi ditampilkan, aktor mengisi kotak isi yang tersedia yaitu tanggal, nama kegiatan, waktu dan menentukan lokasi kegiatan. Pada lokasi kegiatan aktor dapat menentukan lokasi tertentu. Aktor menekan tombol "Submit" maka sistem mengecek data kegiatan dengan beberapa kondisi. Pertama jika aktor tidak mengisi *field* nama kegiatan maka akan menampilkan pesan "Masukkan Nama Kegiatan". Jika aktor tidak mengisi *field* tanggal maka akan menampilkan pesan "Pilih Tanggal Kegiatan". Jika aktor tidak mengisi *field* waktu maka sistem akan menampilkan pesan "Pilih Waktu Kegiatan". Jika aktor tidak mengisi *field* lokasi maka sistem akan menampilkan pesan "Pilih Lokasi Kegiatan" Jika aktor memenuhi semua syarat maka sistem akan menyimpan data ke basis data dan menampilkan pesan "Kegiatan Baru ditambahkan". Sistem akan menampilkan informasi pada halaman utama.

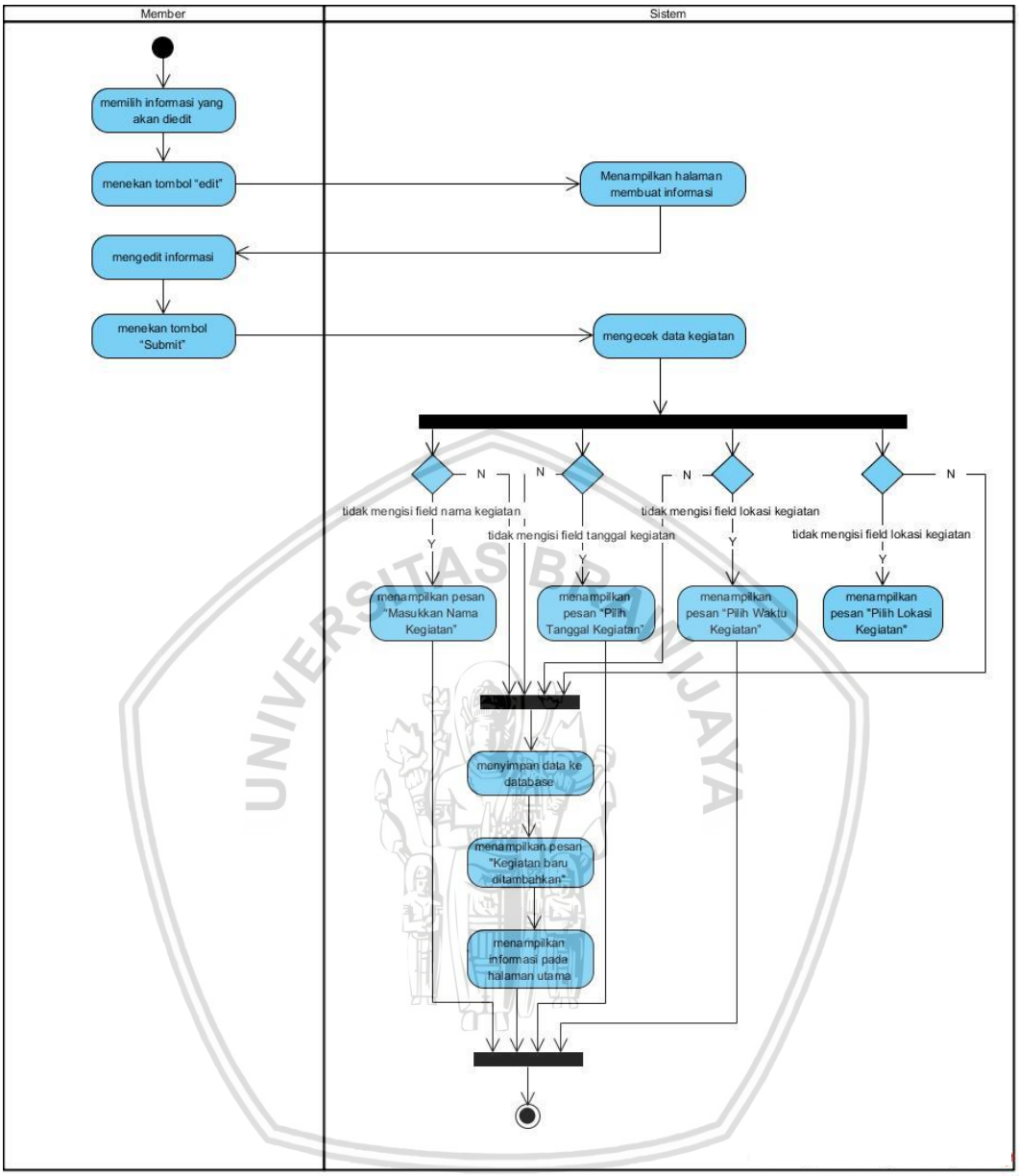
4.2.3.2 *Activity Diagram* Menghapus Informasi Kegiatan Sedekah



Gambar 4. 4 Gambar *activity diagram* menghapus informasi kegiatan sedekah

Activity diagram pada Gambar 4.4 menggambarkan aktivitas dari fungsionalitas menghapus informasi yang telah dibuat. Awalnya *member* memilih informasi yang akan dihapus pada Halaman Timeline, kemudian menekan tombol "Hapus". Sistem akan memproses permintaan dengan menampilkan dialog konfirmasi penghapusan. *Member* menekan tombol "Ya", maka sistem dapat menghapus informasi kegiatan dari *database*.

4.2.3.3 *Activity Diagram* Mengedit Informasi Kegiatan Sedekah

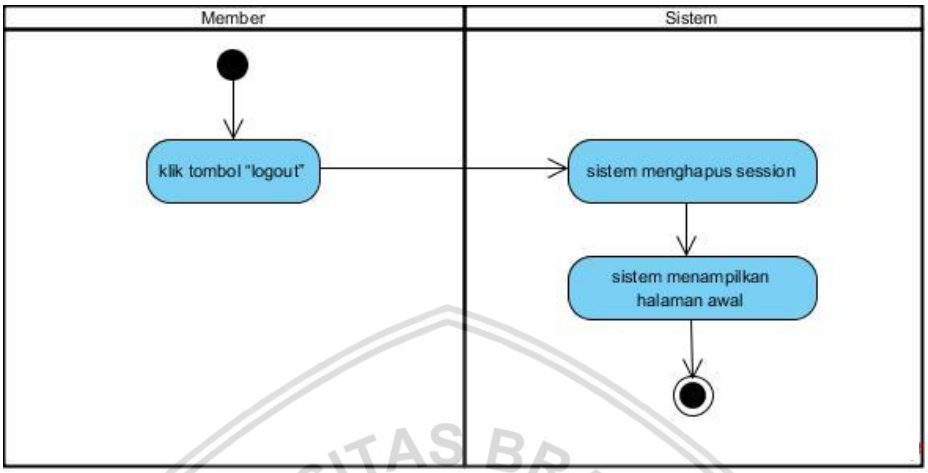


Gambar 4. 5 Gambar *activity diagram* mengedit informasi kegiatan sedekah

Activity diagram pada Gambar 4.5 adalah *activity diagram* mengedit informasi. Awalnya aktor *member* akan masuk ke halaman Profil dan memilih informasi yang akan diedit. Selanjutnya aktor menekan tombol "Edit" dan sistem menampilkan halaman membuat informasi serta mengambil nilai sebelumnya. Aktor selanjutnya mengedit informasi dan menekan tombol "Submit". Aktor menekan tombol "Submit" maka sistem mengecek data kegiatan dengan beberapa kondisi. Pertama jika aktor tidak mengisi *field* nama kegiatan maka akan menampilkan pesan "Masukkan Nama Kegiatan". Jika aktor tidak mengisi *field* tanggal maka akan menampilkan pesan "Pilih Tanggal Kegiatan". Jika aktor tidak mengisi *field* waktu maka sistem akan menampilkan pesan "Pilih Waktu Kegiatan". Jika aktor tidak mengisi *field* lokasi maka sistem akan menampilkan pesan "Pilih

Lokasi Kegiatan" Jika aktor memenuhi semua syarat maka sistem akan menyimpan data ke basis data dan menampilkan pesan "Informasi Berhasil diedit". Sistem akan menampilkan informasi pada halaman utama.

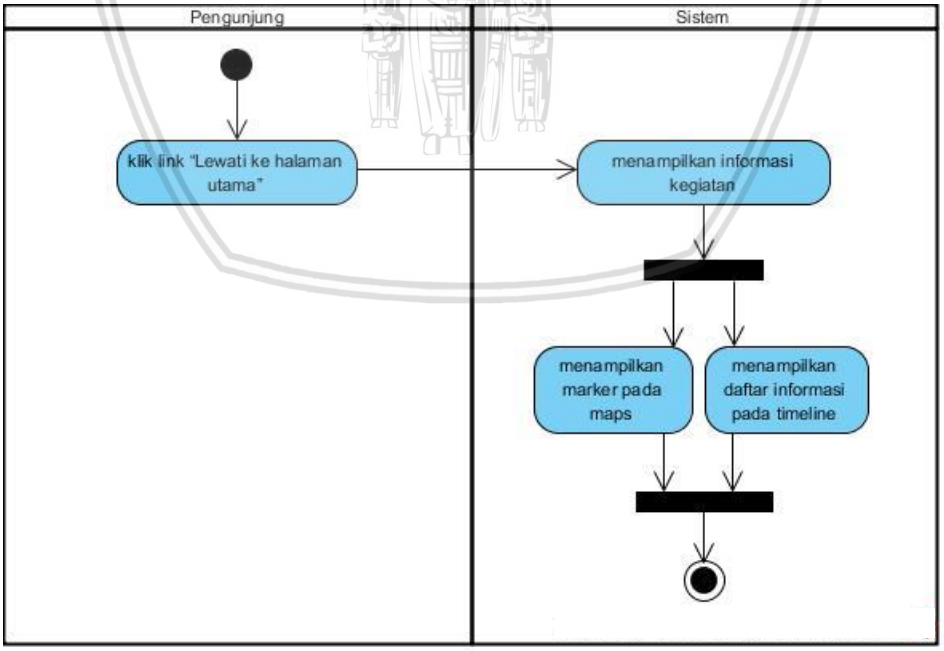
4.2.3.4 *Activity Diagram Logout* Sistem



Gambar 4. 6 Gambar *activity diagram logout* sistem

Activity diagram yang ditunjukkan pada Gambar 4.6 adalah aktivitas dari fungsi *logout*. Awalnya *member* akan menekan tombol "Logout" dan sistem akan memproses perintah *logout* dengan menghapus *member session* kemudian sistem menampilkan halaman awal.

4.2.3.5 *Activity Diagram* Melihat Informasi Kegiatan Sedekah

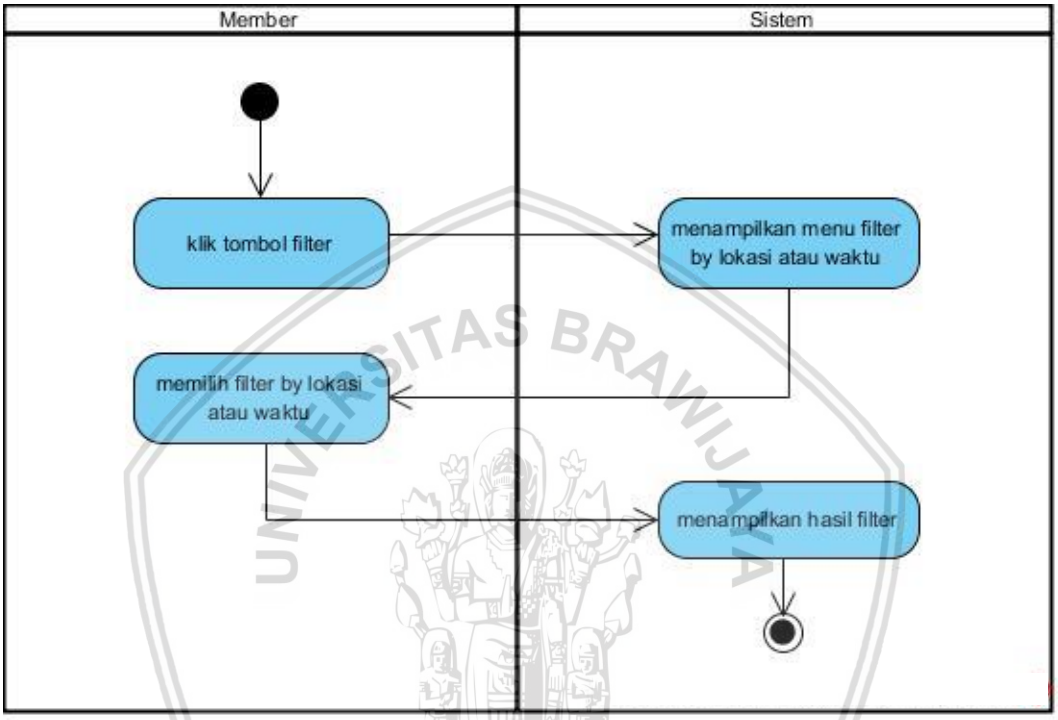


Gambar 4. 7 Gambar a*ctivity diagram* melihat informasi kegiatan sedekah

Gambar 4.7 adalah *activity diagram* untuk fungsi melihat kegiatan sedekah. Awalnya *member* akan menekan *link* "Lewati ke halaman utama" pada halaman

awal sistem. Selanjutnya sistem akan menampilkan informasi kegiatan. Pada proses ini ada dua aktivitas yang akan dijalankan yang pertama yaitu sistem menampilkan *marker* pada *map* berupa detail informasi kegiatan dan yang kedua adalah sistem menampilkan informasi kegiatan berupa daftar kegiatan pada *timeline*. *Diagram activity* ini berakhir ketika telah menampilkan dua halaman tersebut.

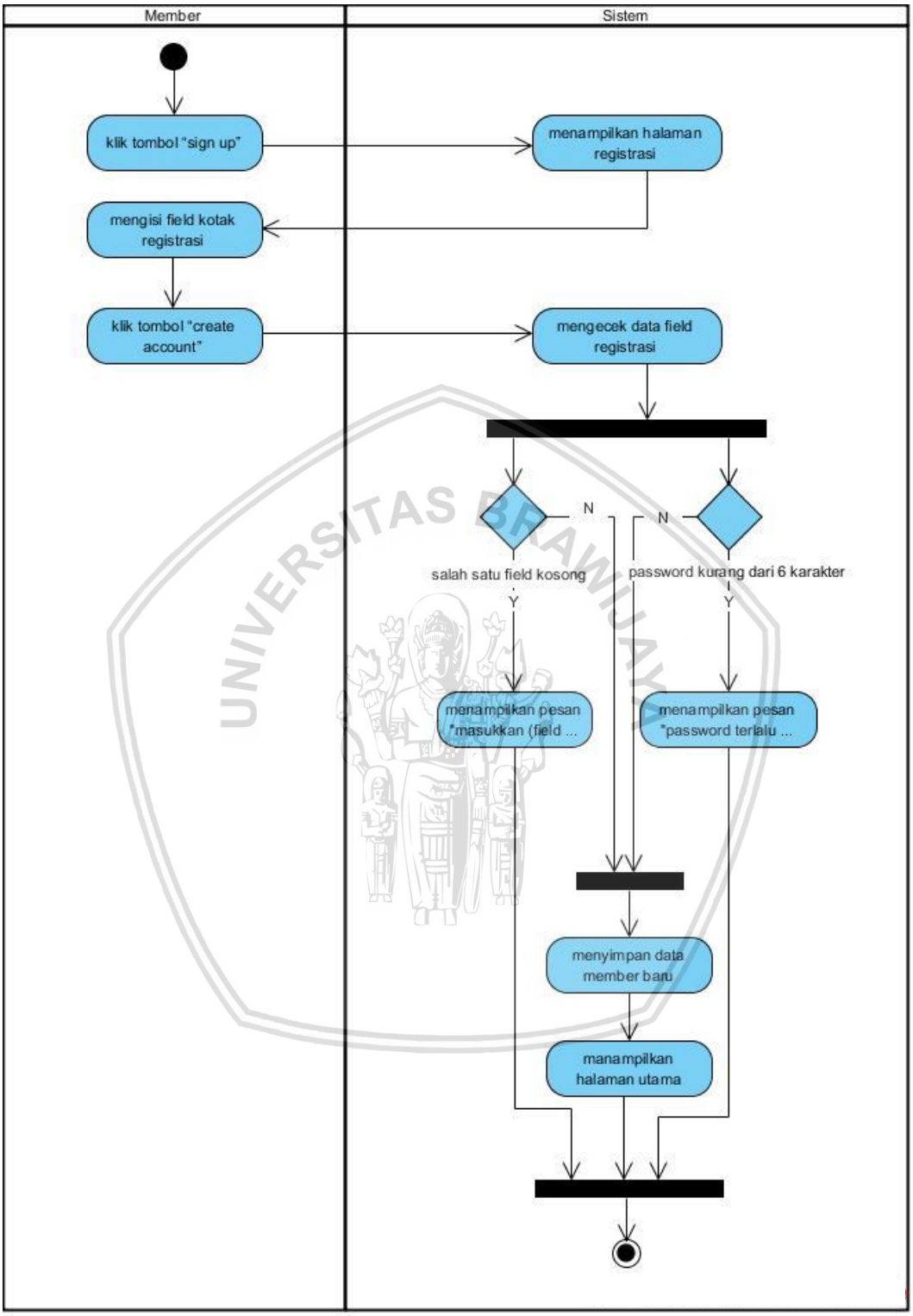
4.2.3.6 *Activity Diagram* Memfilter Informasi Kegiatan Sedekah



Gambar 4. 8 Gambar *activity diagram* memfilter informasi kegiatan sedekah

Gambar 4.8 adalah *activity diagram* memfilter informasi kegiatan sedekah. Awalnya aktor sudah masuk ke halaman Timeline. Untuk memfilter informasi, aktor mengklik tombol filter untuk menentukan filter berdasarkan waktu atau lokasi maka sistem akan menampilkan hasil filter.

4.2.3.7 *Activity Diagram* Registrasi

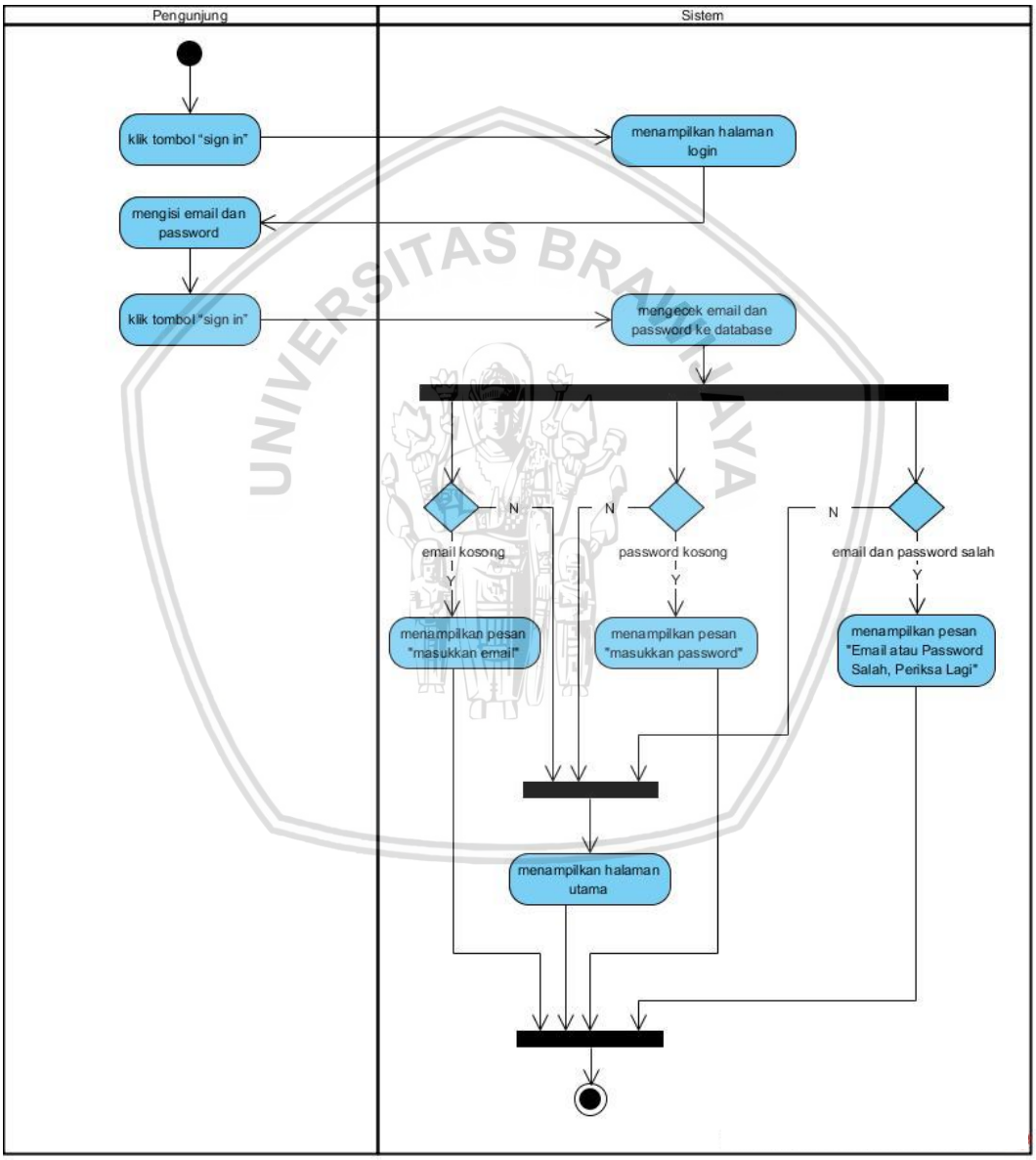


Gambar 4. 9 Gambar *activity diagram* registrasi

Activity diagram yang ditunjukan pada Gambar 4.9 adalah *activity diagram* dari fungsi registrasi. Awalnya *member* akan menekan tombol "*Sign up*" pada halaman awal sistem. Kemudian sistem akan merespon permintaan dari *member*

dengan menampilkan halaman registrasi. Pada halaman registrasi, *member* akan mengisi *field* kotak registrasi yang akan dan menekan tombol "Sign up". Sistem akan mengecek data *field* registrasi. Jika salah satu *field* kosong, maka sistem akan menampilkan pesan "Masukkan (*field* yang kosong)". Jika aktor mengisi *password* kurang dari 6 karakter maka sistem akan menampilkan pesan "*Password* terlalu pendek, masukkan minimal 6 karakter". Jika persyaratan terpenuhi maka sistem akan menyimpan data *member* baru dan menampilkan halaman utama. Aktor mandapatkan *privilege* sebagai *member*.

4.2.3.8 *Activity Diagram Login* Sistem



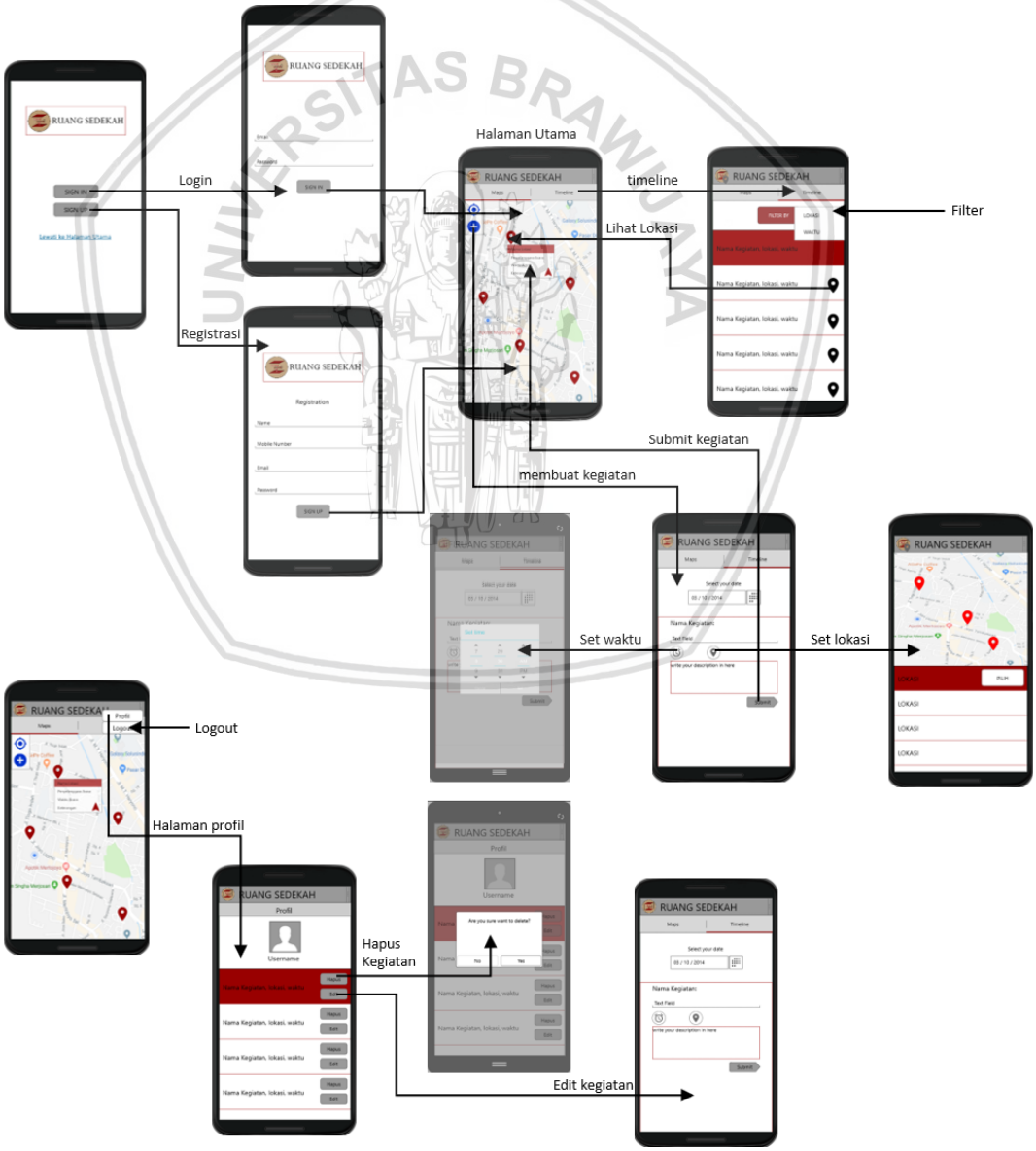
Gambar 4. 10 Gambar *activity diagram login* sistem

Activity diagram yang ditunjukkan pada Gambar 4.10 akan menjelaskan diagram alur dari fungsi *login*. Awalnya *member* akan menekan tombol "Sign in". Sistem akan menampilkan halaman halaman *login* dan *member* dapat mengisi

field email dan *password* pada *text field* yang ada. Selanjutnya *member* menekan tombol "Sign in". Sistem akan mengecek email dan *password* yang diinputkan terlebih dahulu. Jika aktor mengklik tombol "Sign in" dengan keadaan email kosong maka sistem akan menampilkan pesan "Masukkan email". Jika kondisi *password* kosong maka sistem akan menampilkan pesan "Masukkan *password*". Dan jika email dan *password* salah maka sistem akan menampilkan pesan "email dan *password* salah, periksa lagi". Jika syarat telah terpenuhi maka sistem akan menampilkan halaman utama.

4.2.4 *Screen Flow*

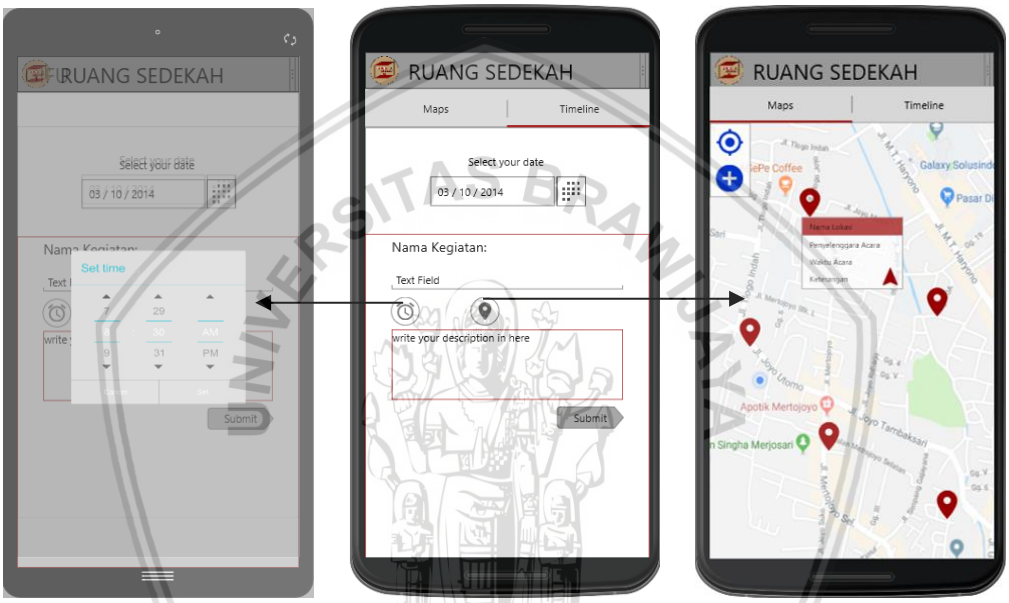
Screen flow menggambarkan alur aktivitas dari tampilan UI (*User Interface*)dari aplikasi sistem manajemen informasi sedekah berbagi makanan dalam bentuk *mockup* yang menunjukkan hubungan antar halaman aplikasi. *Screen flow* tersebut dapat ditunjukkan pada Gambar 4.11.



Gambar 4. 11 *Screen flow* halaman sistem

4.2.4.1 *User Interface* Halaman Membuat Informasi Kegiatan Sedekah

Gambar di bawah ini adalah tampilan halaman membuat informasi sedekah. Pada halaman ini *member* akan melakukan penambahan informasi kegiatan sedekah berbagi makanan. Pada halaman ini *member* akan menentukan tanggal kegiatan pada tombol memilih tanggal dan akan menampilkan kalender. Halaman ini juga menampilkan tombol untuk memilih waktu kegiatan dan akan menampilkan waktu dengan tampilan analog, tombol lokasi untuk menentukan lokasi, text input untuk menginputkan nama kegiatan dan deskripsi kegiatan. Jika menulis informasi kegiatan telah selesai, disediakan tombol "Submit" untuk memasukkan data kedalam *database*. *User interfac*e halaman membuat informasi kegiatan sedekah dapat dilihat pada Gambar 4.12.



Gambar 4. 12 *User interface* halaman membuat informasi kegiatan sedekah

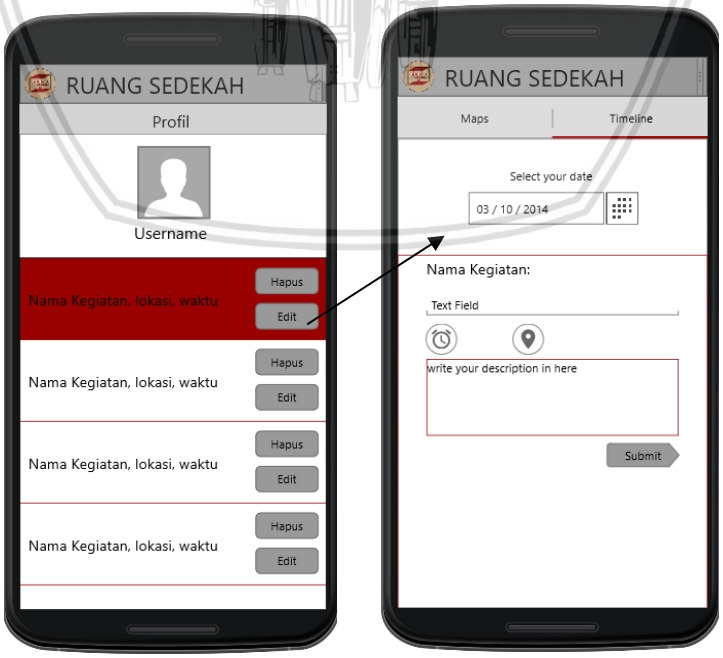
4.2.4.2 *User Interface* Halaman Menghapus Informasi Kegiatan Sedekah

Gambar di bawah ini adalah *user interface* dari halaman menghapus informasi sedekah. Pada halaman profil ditampilkan data profil berupa *username* dan email serta kegiatan yang dibuat oleh *member*. Pada halaman ini juga disediakan dua tombol untuk mengelola data yaitu tombol edit dan tombol hapus. *Member* akan mengklik tombol hapus dan akan menampilkan dialog konfirmasi penghapusan data. *User interface* Halaman menghapus informasi kegiatan sedekah dapat dilihat pada Gambar 4.13.

Gambar 4. 13 *User interface* halaman menghapus informasi kegiatan sedekah

4.2.4.3 *User Interface* Halaman Mengedit Informasi Kegiatan Sedekah

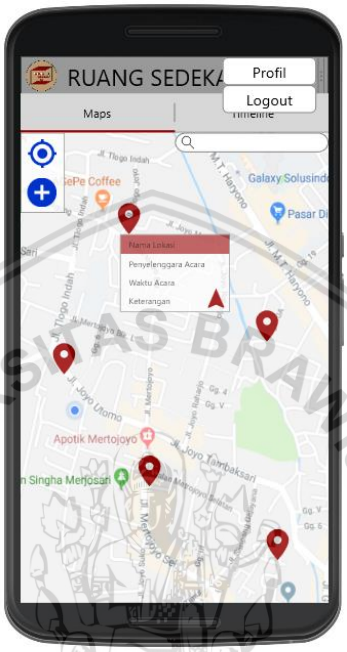
Gambar di bawah ini adalah tampilan halaman mengedit informasi sedekah. Pada dasarnya, *fragment* yang digunakan untuk menulis informasi kegiatan dan melakukan edit informasi adalah sama. Sehingga *member* akan membuka halaman membuat informasi kegiatan dengan *precondition* adalah data yang sudah ada sebelumnya. *User interface* Halaman mengedit informasi kegiatan sedekah dapat dilihat pada Gambar 4.14.



Gambar 4. 14 *User interface* halaman mengedit informasi kegiatan sedekah

4.2.4.4 *User Interface* Halaman *Logout* Sistem

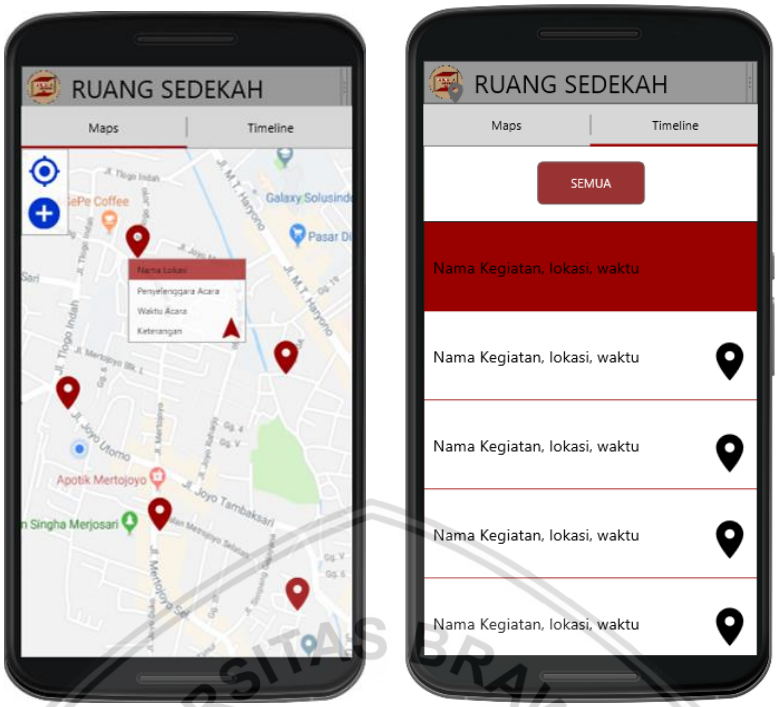
Gambar di bawah ini adalah tampilan dari halaman *logout*. Tombol *logout* terdapat pada halaman utama aplikasi. Tombol *logout* terletak pada bagian atas kanan layar bersama dengan tombol profil dalam bentuk menu. Untuk menjalankan fungsi ini, *member* harus menekan tombol *logout* maka *member* akan keluar dari sistem. *User interface* Halaman Logout dapat dilihat pada Gambar 4.15.



Gambar 4. 15 *User interface* halaman *logout* sistem

4.2.4.5 *User Interface* Halaman Melihat Informasi Kegiatan Sedekah

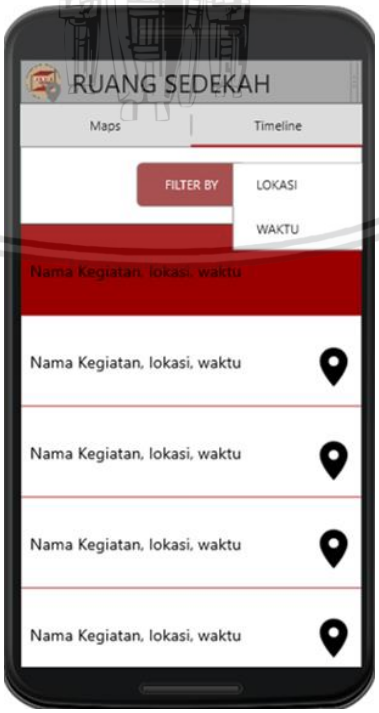
Gambar di bawah ini adalah tampilan dari halaman melihat informasi kegiatan sedekah. Pengunjung atau *member* dapat melakukan fungsi ini sehingga tidak perlu melakukan *sign in* atau *sign up*. Cukup menekan *link* "lewati ke halaman utama", maka sistem akan menampilkan halaman informasi kegiatan dengan berupa *map*. Detail informasi kegiatan dapat dilihat dengan menekan *marker.* Detail *marker* yang ditampilkan adalah nama kegiatan, tanggal kegiatan, waktu kegiatan dan nama lokasi kegiatan. Selain itu pengunjung juga dapat melihat informasi kegiatan berupa daftar kegiatan pada *timeline*. Pada halaman Timeline akan menampilkan daftar informasi kegiatan sedekah dan pilihan filter berupa tombol *spinner*. *User interface* Halaman melihat informasi kegiatan sedekah dapat dilihat pada Gambar 4.16.



Gambar 4. 16 *User interface* halaman melihat informasi kegiatan sedekah

4.2.4.6 *User Interface* Halaman Memfilter Informasi Kegiatan Sedekah

Gambar di bawah ini adalah *mockup* dari halaman memfilter informasi kegiatan. Pengunjung atau *member* dapat melakukan filter berdasarkan lokasi dan waktu yang ditampilkan berupa *spinner. User interface* halaman filter dapat dilihat pada Gambar 4.17.



Gambar 4. 17 *User interface* halaman memfilter informasi kegiatan sedekah

4.2.4.7 *User Interface* Halaman Registrasi

Gambar di bawah ini adalah *mockup* dari halaman registrasi. Pengunjung akan menekan "Sign up" pada halaman awal ketika aplikasi dijalankan. Setelah itu, halaman registrasi dibuka dan menampilkan *fragment* yang berisi *field-field* yang harus diisi. Halaman registrasi menampilkan form registrasi nama, nomor telepon, email dan password serta tombol "Sign Up" untuk mendaftarkan member. *User interface* Halaman Registrasi dapat dilihat pada Gambar 4.18.

Gambar 4. 18 *User interface* halaman registrasi

4.2.4.8 User Interface Halaman *Login* Sistem

Gambar di bawah ini adalah *mockup* halaman *login*. Dimana pengunjung ketika akan masuk ke dalam sistem sebagai *member*, maka pengunjung akan melakukan *login* dengan akun yang sudah terdaftar sebelumnya. Awalnya sistem akan menampilkan halaman awal berupa pilihan *sign in*, *sign up* atau menuju ke halaman utama. Pengunjung menekan tombol *sign in* maka halaman *login* akan ditampilkan. Pada halaman login, disediakan *text input* email dan *password* serta tombol "Sign in". *User interface* Halaman Login dapat dilihat pada Gambar 4.19.

Gambar 4. 19 *User interface* halaman *login* sistem

BAB 5 ANALISIS KOMPONEN DAN PERANCANGAN

5.1 Analisis Komponen

Analisis komponen dilakukan dengan mendefinisikan *component retrieval*. *Component retrieval* adalah proses pencarian informasi kondisi dan komponen yang ada pada *library*. Untuk memulai pencarian komponen maka yang harus dilakukan adalah mendefinisikan kebutuhan dari komponen penggunaan ulang yang akan dicari dengan membuat tabel kriteria komponen pada masing-masing komponen penggunaan ulang. Selanjutnya dari komponen yang tersedia maka akan dipilih salah satu komponen yang sesuai dengan kebutuhan. Untuk melihat kriteria komponen dan *component retrieval* dari masing-masing komponen penggunaan ulang dapat dilihat pada Tabel 5.1 sampai dengan Tabel 5.8.

a. Sistem Dapat Menampilkan Aplikasi *Map*

Tabel 5. 1 Tabel kriteria komponen aplikasi *map*

No.	Kode kriteria	Kriteria komponen	No. Kebutuhan
1.	A	Dapat menampilkan peta	RS-F-2.1
2.	B	Dapat membuat *marker*	RS-F-1.1
3.	C	Dapat menampilkan marker	RS-F-2.1
4.	D	Dapat melakukan pencarian lokasi terdekat	RS-F-1.1
5.	E	Dapat melakukan sugesti pencarian lokasi	RS-F-1.1

Tabel 5. 2 Tabel *component retrieval* aplikasi *map*

No.	Komponen yang tersedia	*Package*	Kode komponen	Komponen yang dipilih
1.	OsmAnd	net.osmand.plus	A, B, C, E	Google Maps
2.	Osmdroid	org.osmdroid.gpkg	A, B, C, E	
3.	Google Maps	com.google.android.gms:play-services-maps:12.0.1	A, B, C, D, E	
4.	*MAPS*.ME	com.mapswithme.maps.pro	A, B, C, E	

Komponen aplikasi *map* yang dipilih berdasarkan kriteria komponen diatas adalah Google Maps. Google Maps memenuhi kebutuhan dari aplikasi.

b. Sistem Dapat Menampilkan *Date Picker*

Tabel 5. 3 Tabel kriteria komponen *date picker*

No.	Kode kriteria	Kriteria komponen	No. Kebutuhan
1.	A	Menampilkan kalender bulan	RS-F-1.1
2.	B	Dapat mengatur hari, bulan dan tanggal	RS-F-1.1
3.	C	Dapat memilih tanggal dengan mudah	RS-NF-1.1

Tabel 5. 4 Tabel *component retrieval date picker*

No.	Komponen yang tersedia	*Package*	Kode komponen	Komponen yang dipilih
1.	Horizontal Calendar	com.github.jhonnyx2012:horizontal-picker:1.0.6	B	Material Date Picker
2.	Material Date Picker	android.app.DatePickerDialog;	A, B, C	
3.	Spinner Datepicker	com.github.drawers:SpinnerDatePicker:1.0.6	B	
4.	RWeekCalendar	com.github.rameshvoltella:RWeekCalendar:0.1.0	B	
5.	Single Date and Time Picker	com.github.florent37:singledateandtimepicker:(last version)	B	

Komponen *date picker* yang dipilih berdasarkan kriteria komponen diatas adalah Material Date Picker. Material Date Picker memenuhi kebutuhan dari aplikasi.

c. Sistem Dapat Menampilkan *Time Picker*

Tabel 5. 5 Tabel kriteria komponen *time picker*

No.	Kode kriteria	Kriteria komponen	No. Kebutuhan
1.	A	Dapat mengatur jam dan menit	RS-F-1.1
2.	B	Dapat menampilkan jam analog	RS-NF-1.1
3.	C	Dapat memilih waktu dengan mudah	RS-NF-1.1

Tabel 5. 6 Tabel *component retrieval time picker*

No.	Komponen yang tersedia	*Package*	Kode komponen	Komponen yang dipilih
1.	Hms Picker	com.github.DeweyReed:HmsPicker:1.0.0	A	Material Time Picker Dialog
2.	Material Time Picker	android.app.TimePickerDialog;	A, B, C	
3.	Single Date and Time Picker	com.github.florent37:singledateandtimepicker:(last version)	A, C	
4.	Dial Time Picker	com.github.ugurtekbas:dialTimePicker:8d263fc3a1	A	

Komponen *time picker* yang dipilih berdasarkan kriteria komponen diatas adalah Material Time Picker Dialog. Material Time Picker Dialog memenuhi kebutuhan dari aplikasi.

d. Sistem Dapat Menampilkan Lokasi Pencarian

Tabel 5. 7 Tabel kriteria komponen pencarian lokasi

No.	Kode kriteria	Kriteria komponen	No. Kebutuhan
1.	A	dapat menampilkan lokasi-lokasi terdekat	RS-F-1.1
2.	B	Dapat menuliskan nama lokasi dan tempat pada *field* pencarian	RS-F-1.1
3.	C	Dapat menampilkan sugesti tempat dari hasil pencarian	RS-F-1.1

Tabel 5. 8 Tabel *component retrieval* pencarian lokasi

No.	Komponen yang tersedia	*Package*	Spesifikasi komponen	Komponen yang dipilih
1.	Google Place Picker	com.google.android.gms.location.places.Place;	A, B, C	Google Place Picker
2.	Google Place Autocomplete	com.google.maps.places.Autocomplete	B, C	

Komponen pencarian lokasi yang dipilih berdasarkan kriteria komponen diatas adalah Google Place Picker. Google Place Picker memenuhi kebutuhan dari aplikasi.

5.2 Modifikasi Kebutuhan

Dari hasil analisis komponen, dapat dilihat bahwa komponen-komponen penggunaan ulang yang dipilih dapat memenuhi semua kriteria kebutuhan yang didefinisikan. Selain itu komponen yang dicari telah sesuai atau telah memenuhi kebutuhan sistem, sehingga tahap modifikasi kebutuhan tidak perlu untuk dilakukan.

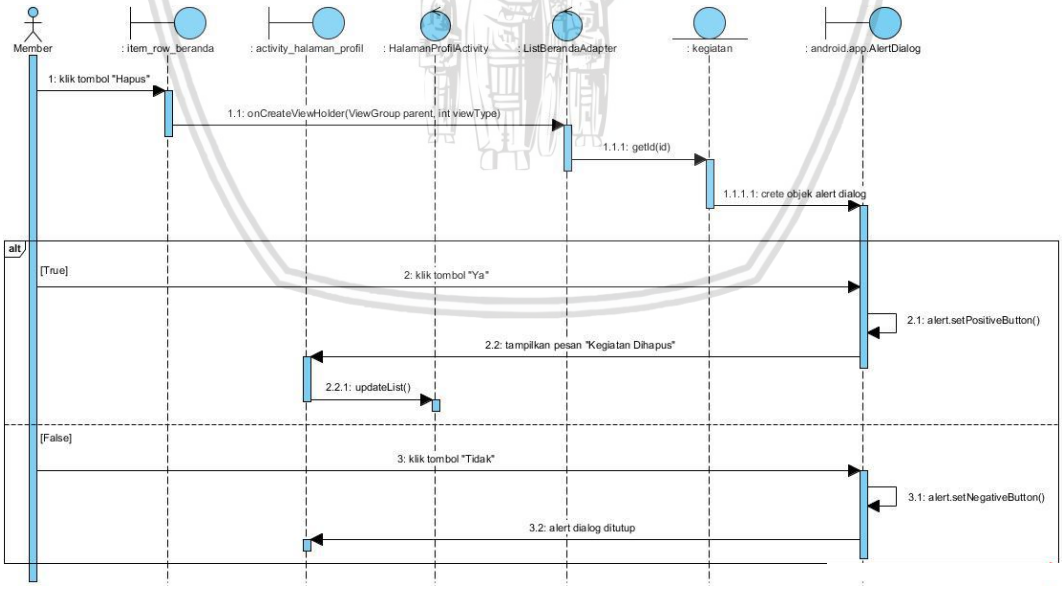
5.3 Perancangan Sistem

Tahap perancangan adalah tahap perangkat lunak dirancangan sesuai dengan kebutuhan yang telah didefinisikan sebelumnya. Dalam perancangan sistem ini terdiri dari *sequence diagram*, perancangan antarmuka, perancangan basis data, diagram komponen dan *class diagram*.

5.3.1 *Sequence Diagram*

Sequence diagram menggambarkan interaksi alur interaksi antar objek. Sequence diagram ini didapatkan dari *use case scenario* pada tahapan analisis kebutuhan. Dalam makalah ini hanya dituliskan 3 sampel *sequence diagram* diantaranya *sequence diagram* membuat informasi kegiatan, *sequence diagram* menghapus informasi kegiatan dan *sequence diagram* memfilter informasi kegiatan. *Sequence diagram* pada aplikasi manajemen informasi sedekah dapat dilihat pada Gambar 5.1 hingga Gambar 5.3.

5.3.1.1 Sequence Diagram Menghapus Informasi

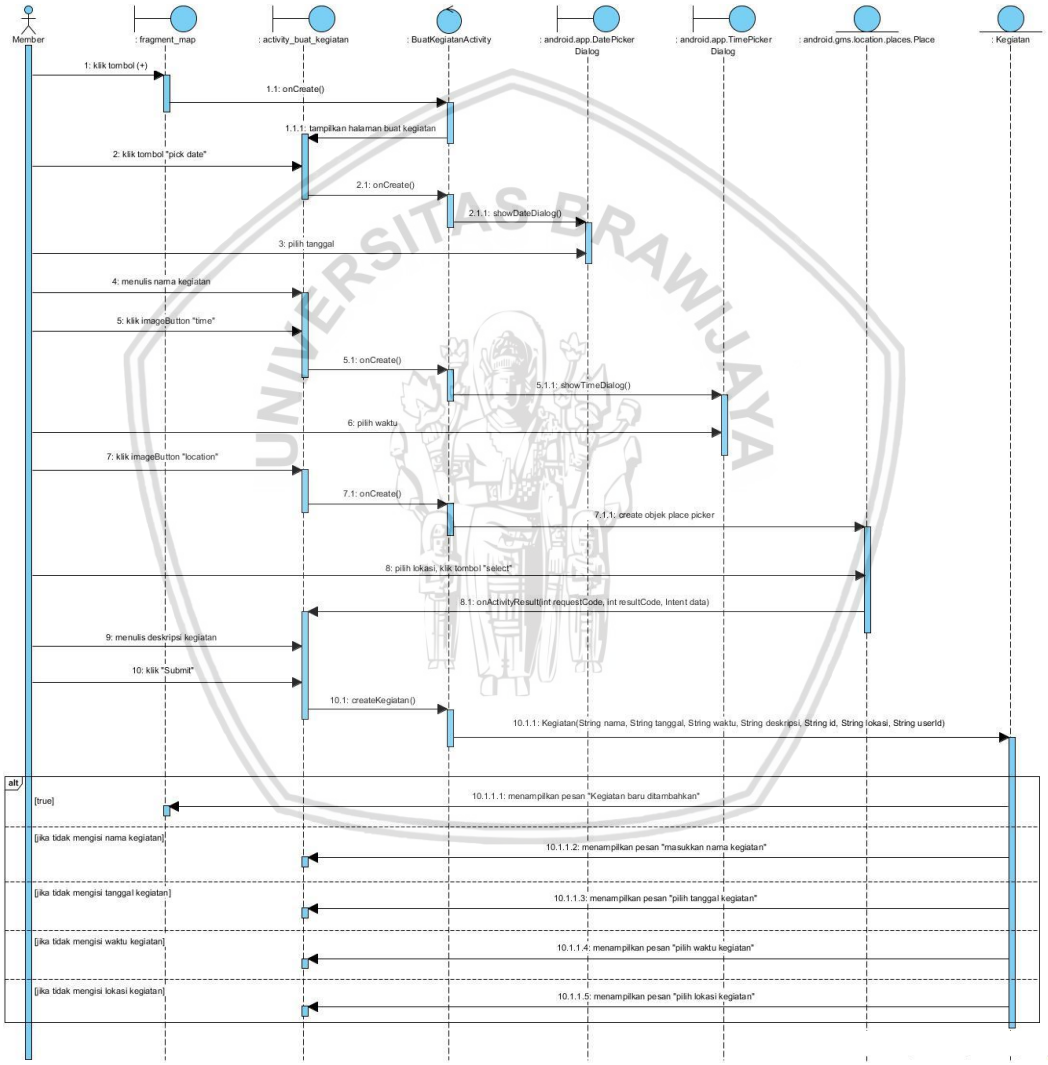


Gambar 5. 1*Sequence diagram* menghapus informasi

Sequence diagram di atas menggambarkan alur dari fungsi menghapus informasi. Aktor mengklik tombol “Hapus” pada item *view* item_row_timeline. *view* item_row_timeline adalah tampilan dari *item recycler view* pada halaman profil. Sistem menjalankan *method* onCreateViewHolder() pada *class*

ListBerandaAdapter. Untuk mendapatkan id kegiatan, maka method getId dipanggil dari *class* kegiatan. Sistem akan membuat objek android.app.AlertDialog. Jika aktor mengklik tombol "Ya", maka method alert.setPositiveButton() akan dijalankan dan menampilkan pesan "Kegiatan dihapus", untuk menampilkan *list* baru, maka sistem akan memanggil *method* updateList() pada *class* HalamanProfilActivity. Dan sebaliknya jika aktor mengklik tombol "Tidak", maka method alert.setNegativeButton() akan dipanggil dan menutup objek android.app.AlertDialog. *Sequence diagram* menghapus informasi sedekah dapat dilihat pada Gambar 5.1.

5.3.1.2 *Sequence Diagram* Membuat Informasi Sedekah



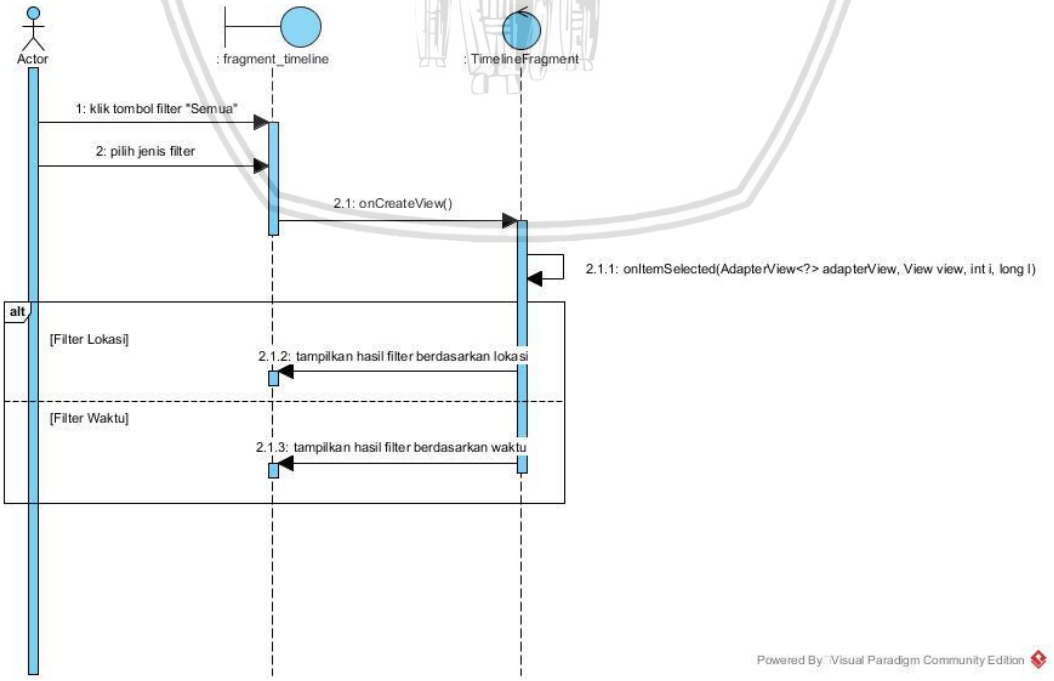
Gambar 5. 2 *Sequence diagram* membuat informasi sedekah

Sequence diagram di atas menggambarkan alur dari fungsi membuat informasi sedekah. Awalnya aktor akan mengklik tombol (+) pada *boundary* fragment_map, sistem menjalankan *method* onCreate() pada *class* BuatKegiatanActivity dan menampilkan *layout* acitivity_buat_kegiatan. Aktor selanjutnya mengklik tombol "Pick Date" dan memanggil *method* onCreate(). Selanjutnya sistem memanggil

method showDataDialog() untuk memanggil memanggil objek *date picker* yaitu android.app.TimePickerDialog. ketika objek telah dipanggil maka aktor dapat memilih tanggal kegiatan. Selanjutnya adalah menentukan waktu kegiatan dengan mengklik *image button* "Time", sistem akan memanggil *method* onCreate() dan memanggil *method* showTimeDialog() untuk menginstansiasi objek *time picker* yaitu android.app, aktor dapat memilih waktu kegiatan. selanjutnya adalah memilih lokasi kegiatan, aktor mengklik *image button* "Location" maka sistem akan memanggil *method* onCreate() dan menginstansiasi objek android.gms.location.places.Place dengan PlacePicker.IntentBuilder(), selanjutnya aktor dapat memilih lokasi tertentu yang disediakan dengan klik tombol "Select".

Untuk menampilkan nama lokasi yang dipilih makan akan memanggil *method* onActivityResult() dari *class* BuatKegiatanActivity. Selanjutnya menulis deskripsi kegiatan dan mengklik tombol "Submit" maka sistem akan menjalankan *method* createKegiatan() pada *class* BuatKegiatanActivity dan memanggil konstruktor Kegiatan(nama, date, time, desc, id, loc) pada *class* Model Kegiatan untuk menyimpan informasi baru. Jika penyimpanan berhasil maka akan menampilkan pesan "informasi berhasil diedit". Jika tidak mengisi nama kegiatan maka sistem akan menampilkan pesan "masukkan nama kegiatan". Jika tidak mengisi tanggal kegiatan maka sistem akan menampilkan pesan "Pilih Tanggal Kegiatan". Jika tidak mengisi waktu kegiatan maka sistem akan menampilkan pesan "Pilih Waktu Kegiatan". Jika tidak mengisi lokasi kegiatan maka sistem akan menampilkan pesan "Pilih Lokasi Kegiatan". *Sequence diagram* membuat informasi sedekah dapat dilihat pada Gambar 5.2.

5.3.1.3 Sequence Diagram Memfilter Informasi Kegiatan

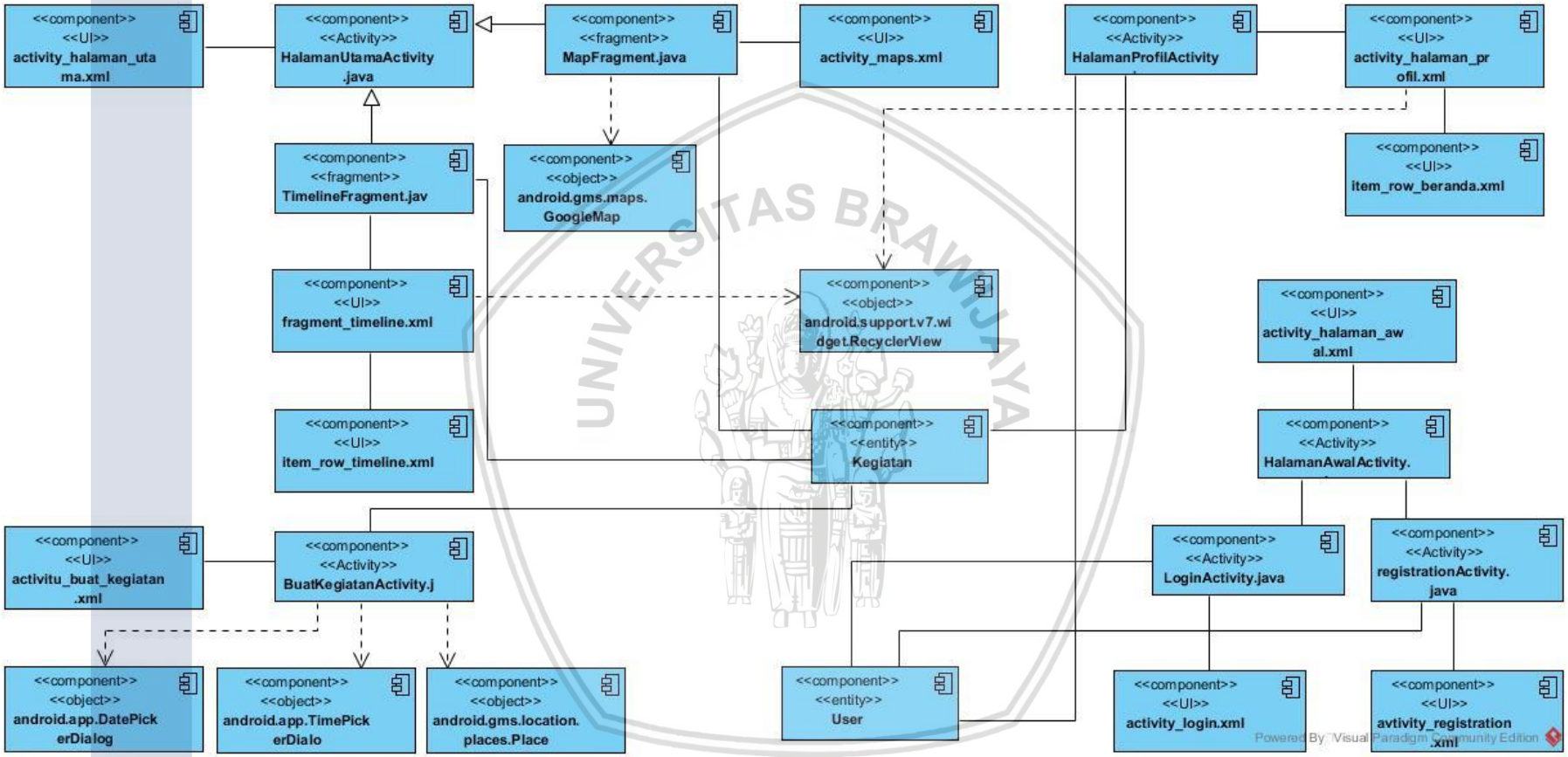


Gambar 5. 3 *Sequence diagram* memfilter informasi kegiatan

Sequence diagram di atas menggambarkan alur dari fungsi memfilter informasi. Aktor masuk ke halaman TimelineFragment dan memilih tombol *Spinner* filter. Sistem akan menampilkan filter berdasarkan lokasi tertentu atau waktu tertentu. Sistem akan menjalankan *method* onCreateView() Sebagai *action* dari pilihan filter pengguna. Selanjutnya sistem akan menjalankan *method* onItemSelected() untuk masing-masing filter. Pada tahap ini akan dibuat percabangan *switch case*. Jika *case* bernilai 0 maka akan menampilkan semua informasi, jika *case* bernilai 1 maka akan menampilkan hasil filter berdasarkan lokasi, jika *case* bernilai 2 maka akan menampilkan hasil filter berdasarkan waktu. *Sequence diagram* memfilter informasi sedekah dapat dilihat pada Gambar 5.3.

5.3.2 Diagram Komponen

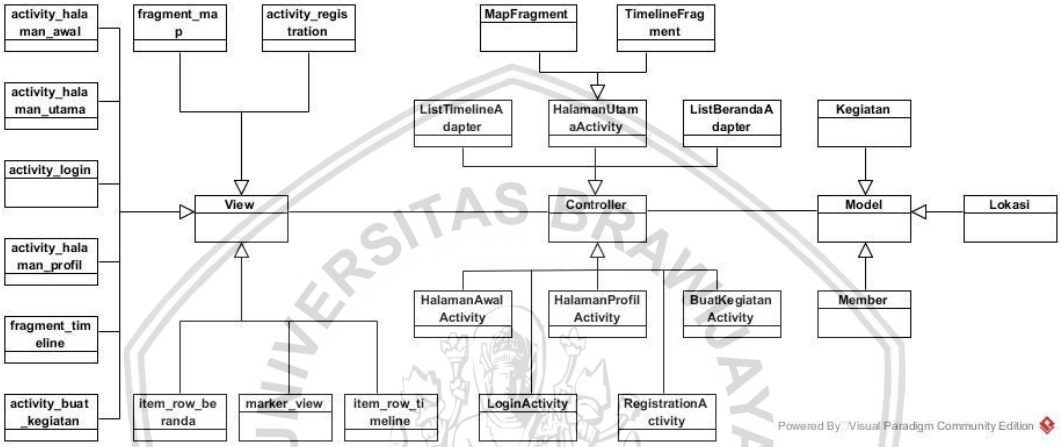
Diagram komponen di bawah ini menjelaskan hubungan antar komponen yang saling ketergantungan *(dependency)*. Komponen-komponen ini terdiri dari komponen klas, *user interface,* dan objek berupa *library* pada komponen android. Diagram komponen aplikasi manajemen informasi sedekah ini memberikan gambaran objek penggunaan ulang yang digunakan dan hubungannya dengan komponen pembangunnya. Diagram komponen aplikasi manajemen informasi sedekah berbagi makanan dapat dilihat pada Gambar 5.4



Gambar 5. 4 Diagram komponen sistem manajeman informasi sedekah berbagi makanan

5.3.3 *Class Diagram*

Class diagram menggambarkan struktur sistem aplikasi manajemen informasi sedekah dari beberapa klas sistem yang akan dibuat. Pada *class diagram* ini terdiri atas 3 bagian utama yaitu nama klas, atribut, dan operasi. Setiap klas memungkinkan akan saling berkomunikasi. Pada framework MVC (Model, View dan Controller), klas-klas dikelompokan dan digeneralisasi berdasarkan jenis model, *view* atau *controller* tersebut. Hubungan antar class model, *view* dan *controller* adalah asosiasi. Gambaran umum *class diagram* sistem manajemen informasi sedekah dapat dilihat pada Gambar 5.5.



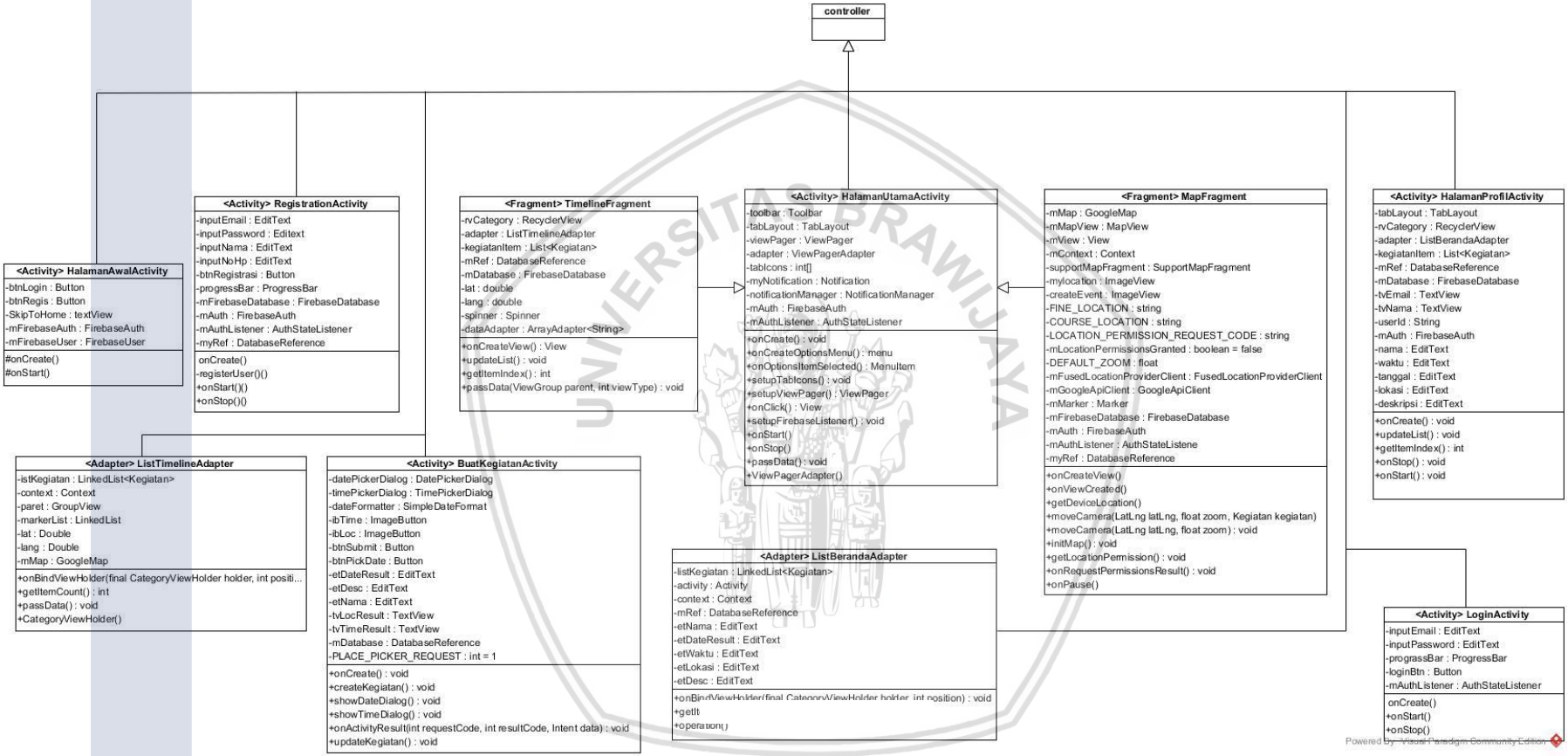
Gambar 5. 5 Gambaran umum c*lass diagram* sistem manajeman informasi sedekah berbagi makanan

5.3.3.1 Class Diagram Controller

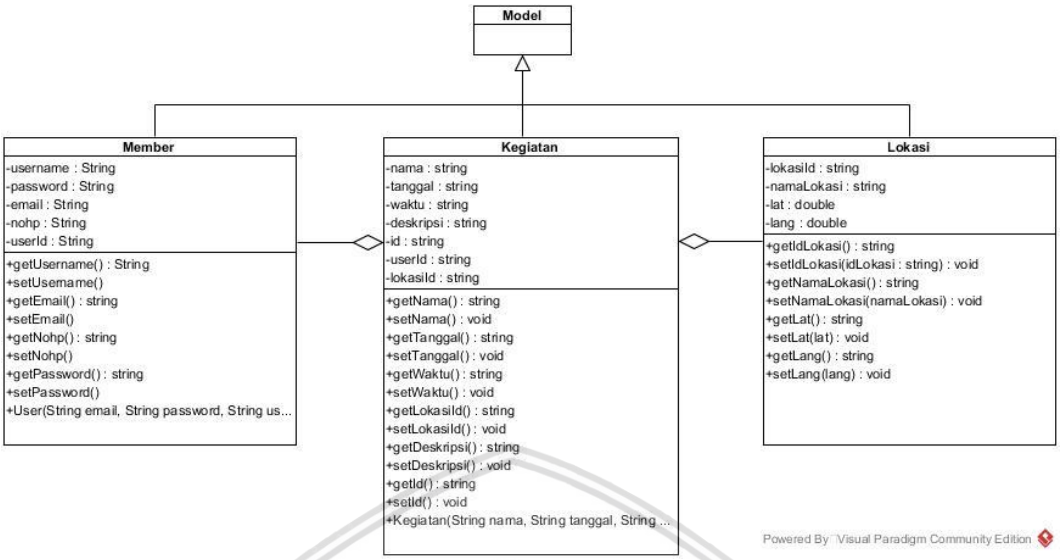
Class diagram controller pada sistem manajemen informasi sedekah menggambarkan hubungan antar *class controller.* Klas-klas yang termasuk *class controller* diantaranya HalamanAwalActivity, RegistrationActivity, ListBerandaAdapter, Timelinefragment, BuatKegiatanActivity, HalamanUtamaActivity, ListBerandaAdapter, MapFragment, HalamanProfilActivity dan LoginActivity. *Class diagram controller* dapat dilihat pada Gambar 5.6.

5.3.3.2 Class Diagram Entity

Class diagram entity pada perancangan ini menggambarkan hubungan antar klas-klas *entity*. Klas *entity* diantaranya adalah klas Kegiatan, klas Member dan klas Lokasi. *Class diagram entity* dapat dilihat pada Gambar 5.7.



Gambar 5. 6 *Class diagram controller*



Gambar 5. 7 *Class diagram* entity

5.3.4 Perancangan Algoritme

Perancangan algoritme menggambarkan urutan dan langkah logis dalam sebuah fungsional sebagai acuan dalam implementasi kode. Dalam perancangan algoritme sistem manajemen informasi sedekah ini dilampirkan 3 buah komponen klas yaitu klas ListBerandaAdapter dengan *method* btnHapus.setOnClickListener(), klas BuatKegiatanActivity dengan *method* createKegiatan() dan klas TimelineFragment dengan *method* onCreateView().

5.3.5 Perancangan Komponen Klas ListBerandaAdapter

Nama *Method* : btnHapus.setOnClickListener()

Algoritme :

Tabel 5. 9 Perancangan algoritme klas ListBerandaAdapter *method* btnHapus.setOnClickListener()

```
1.  AlertDialog alert
2.      Set Title "delete data"
3.      Set Message "Anda yakin menghapus kegiatan?"
4.      Set PositiveButton ("Ya")
5.           removeValue
6.           print "kegiatan berhasil dihapus"
7.      Set NegativeButton("Tidak")
8.           Tutup alert dialog
9.  end
```

5.3.6 Perancangan Komponen Klas BuatKegiatanActivity

Nama *Method* : createKegiatan()

Algoritme :

Tabel 5. 10 Perancangan algoritme klas BuatKegiatanActivity *method* createKegiatan()

```
1.  String date = etDateResult
2.  String nama = etNama
3.  String desc = etDesc
4.  String time = tvTimeResult
5.  String loc = tvLocResult
6.      if (nama = null)
7.          print "Masukkan Nama Kegiatan"
8.      else if (date = null)
9.          print "Pilih Tanggal Kegiatan"
10.     else if (time = null)
11.         print "Pilih Waktu Kegiatan"
12.     else if loc = null
13.         Print "Pilih Lokasi Kegiatan”
14.     Else
15.         Print “Kegiatan ditambahkan”
16.         Tampilkan HalamanUtama
17.         StartActivity
18. End if
```

5.3.7 Perancangan Komponen Klas TimelineFragment

Nama *Method* : onCreateView()

Algoritme :

Tabel 5. 11 Perancangan algoritme klas TimelineFragment *method* onCreateView()

```
1.  Firebase database mDatabase
2.  Database reference mRef
3.  LinkedList kegiatanItem
4.  linkedList lokasi
5.  RecyclerView rvCategory
6.  ListTimelineAdapter adapter
7.  Membuat event pada objek spinner
8.       Switch(i)
9.            Case 0
10.                UpdateList()
11.           Case 1
12.                If distance < 0.050
13.                     Tampilkan kegiatan
14.           Case 2
15.                If days >=0 dan <= 30
16.                     Tampilkan kegiatan
17.      End case
```

5.3.8 Perancangan Basis Data

Dalam aplikasi manajemen informasi berbagi makanan ini diperlukan adanya basis data untuk menyimpan data pengguna dan kegiatan yang telah dibuat. Perancangan basis data dibuat untuk mempermudah dalam pendefinisian data yang akan diimplementasikan. Perancangan basis data dapat dilihat pada keterangan berikut:

1. Tabel *Member*

Tabel *member* digunakan untuk menyimpan data dari akun *member* yang telah melakukan registrasi. Pada Tabel *member* ada beberapa *field* yang harus didefinisikan, yaitu ID pada Tabel *member* berfungsi sebagai *primary key* dan menjadi *foreign key* pada Tabel Kegiatan. Perancangan basis data Tabel *member* dapat dilihat pada Tabel 5.1

Tabel 5. 12 Perancangan basis data Tabel *member*

NO.	Nama *Field*	Tipe Data	Keterangan
1.	UserId	String	Berisi nilai ID dari Tabel *member* dan merupakan *primary key*
2.	Username	String	Berisi *username* yang akan dipakai untuk nama profil
3.	Email	String	Berisi email yang sudah terdaftar untuk *login* dan mengirim kode registrasi
4.	Nohp	String	Berisi nomor telepon yang digunakan untuk registrasi
5.	Password	String	Berisi *password* untuk *login*

2. Tabel Kegiatan

Tabel kegiatan digunakan untuk menyimpan data dari informasi kegiatan yang dibuat. Tabel kegiatan menyimpan seluruh kegiatan yang akan dibuat secara detail pada Tabel di bawah. Pada kolom Id berfungsi sebagai *primary key* dan menjadi *foreign key* pada Tabel Member. Tabel 5.12 menggambarkan perancangan basis data Tabel Kegiatan.

Tabel 5. 13 Perancangan basis data Tabel kegiatan

NO.	Nama *Field*	Tipe Data	Keterangan
1.	Id	String	Berisi nilai id dari Tabel Kegiatan dan merupakan *primary key*
2.	Nama	String	Berisi nama kegiatan sedekah
3.	Tanggal	String	Berisi tanggal kegiatan dilaksanakan
4.	Waktu	String	Berisi waktu jam dan menit kegiatan dilaksanakan
5.	Deskripsi	String	Berisi deskripsi lengkap kegiatan
6.	userId	String	Berisi nilai id dari Tabel Member dan merupakan *foreign key*

7.	lokasiId	String	Berisi nilai id dari Tabel Lokasi dan merupakan *foreign key*

3. Tabel Lokasi

Tabel Lokasi digunakan untuk menyimpan data dari lokasi kegiatan. Tabel lokasi menyimpan data lokasi berupa nama lokasi, latitude, longitude dan id lokasi dan id kegiatan. Pada kolom lokasiId berfungsi sebagai *primary key* dan menjadi *foreign key* pada Tabel Lokasi. Tabel 5.13 menggambarkan perancangan basis data Tabel Lokasi.

Tabel 5. 14 Perancangan basis data Tabel Lokasi

NO.	Nama *Field*	Tipe Data	Keterangan
1.	lokasiId	String	Berisi nilai id lokasi kegiatan yang merupakan *primary key*
3.	Lat	Double	Berisi nilai latitude
4.	Lang	Double	Berisi nilai longitude
5.	namaTempat	String	Berisi nama lokasi kegiatan

BAB 6 IMPLEMENTASI DAN PENGUJIAN

6.1 Implementasi

Pada bagian ini akan menjelaskan implementasi dari hasil perancangan sebelumnya. Hasil implementasi ini akan membuktikan kesesuaian dengan analisis kebutuhan dan perancangan. Bagian implementasi terdiri dari spesifikasi sistem yaitu spesifikasi perangkat keras, spesifikasi perangkat *smartphone*, spesifikasi perangkat lunak komputer dan spesifikasi perangkat lunak *smartphone,* implementasi antarmuka, implementasi kode program, dan implementasi basis data.

6.1.1 Spesifikasi Sistem

Spesifikasi sistem diperlukan untuk mendefinisikan standar dari perangkat lunak dan perangkat keras digunakan untuk aplikasi ini. Spesifikasi sistem akan memberi kriteria dan standar dari perangkat lunak dan perangkat keras yang digunakan selama tahap implementasi.

6.1.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras diantaranya untuk mendefinisikan perangkat komputer dan *device* yang digunakan pada aplikasi sistem manajemen informasi sedekah. Perangkat komputer digunakan dalam proses pembuatan kode program aplikasi, sementara perangkat lain dalam hal ini adalah smartphone digunakan untuk instalasi aplikasi dan sebagai *running emulator.* Spesifikasi perangkat keras ini dapat dilihat pada Tabel 6.1.

Tabel 6. 1 Spesifikasi perangkat keras komputer

No.	Nama Komponen	Spesifikasi
1.	Model	Hawlett-Packard 14 Notebook PC
2.	Prosesor	Intel(R) Celeron(R) CPU N2840 @ 2.16GHz
3.	Memory	8GB RAM, 131GB HDD
4.	Display	Intel(R) HD Gaphics 2GB

Adapun spesifikasi perangkat *smartphone* dapat dilihat pada Tabel 6.2.

Tabel 6. 2 Spesifikasi perangkat *smartphone*

No.	Nama Komponen	Spesifikasi
1.	Model	ASUS_Z00VD (WW_Phone)
2.	Prosesor	ARM Cortex-A7 2.0 GHz
3.	Memory	2GB RAM
4.	Display	4 inch, resolusi HD IPS 720 x 1280 pixels (292 dpi pixel density)

6.1.1.2 Spesifikasi Perangkat Lunak Komputer

Spesifikasi perangkat lunak komputer akan menggambarkan perangkat lunak yang digunakan dalam implementasi pengembangan sistem. Spesifikasi perangkat lunak dapat dilihat pada Tabel 6.3.

Tabel 6. 3 Spesifikasi perangkat lunak komputer

No.	Nama Komponen	Spesifikasi
1.	Sistem Komputer	Microsoft Windows 10 Enterprise 64-bit
2.	Bahasa Pemrograman	Java
3.	Editor	Android Studio
4.	*Database Server*	Firebase 2.0

Spesifikasi perangkat lunak *smartphone* dapat dilihat pada Tabel 6.4.

Tabel 6. 4 Tabel spesifikasi perangkat lunak *smartphone*

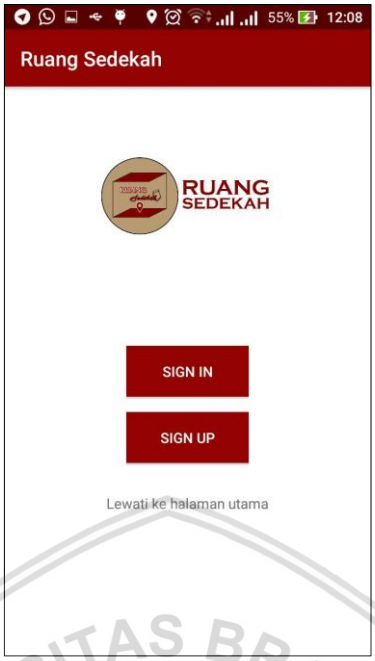
No.	Nama Komponen	Spesifikasi
1.	*Platform*	Android 5.1, SDK Version 22, Version WW-12.1.1.5-20151021

6.1.2 Implementasi Antarmuka

Implementasi antarmuka dilakukan dengan mengimplementasikan hasil rancangan *user interface* yang telah dibuat. Implementasi antaramuka pada aplikasi ini dibuat pada *layout* android studio. Berikut adalah implementasi aplikasi manajemen informasi sedekah berbagi makanan.

6.1.2.1 Implementasi Antarmuka Halaman Awal

Antarmuka halaman awal adalah antarmuka pertama ketika aplikasi dijalankan. Pada halaman awal ini pengguna dapat memilih untuk *login,* register atau menuju ke halaman utama. Antarmuka ini disesuaikan dengan perancangan *user interface* yang sudah digambarkan sebelumnya. Implementasi antarmuka halaman awal dapat dilihat pada Gambar 6.1.



Gambar 6. 1 Implementasi antarmuka halaman awal

6.1.2.2 Implementasi Antarmuka Halaman *Login*

Antarmuka di bawah ini adalah antarmuka halaman *login*, di mana pengguna dapat melakukan *login* dengan mengisi email dan *password*. Antarmuka halaman *login* ini sesuai dengan perancangan antarmuka halaman *login* yang telah dibuat sebelumnya. Implementasi antarmuka halaman *login* dapat dilihat pada Gambar 6.2.

Gambar 6. 2 Implementasi antarmuka halaman *login*

6.1.2.3 Implementasi Antarmuka Halaman Registrasi

Implementasi antarmuka halaman registrasi adalah antarmuka yang dibuat sesuai dengan hasil perancangan *user interface* dan kebutuhan pengguna pada halaman ini pengguna dapat melakukan registrasi dengan mengisi nama, nomor hp, email dan *password*. Halaman ini adalah halaman menambah akun dan *privilege* sebagai *member*. Implementasi antarmuka halaman registrasi dapat dilihat pada Gambar 6.3.

Gambar 6. 3 Implementasi antarmuka halaman registrasi

6.1.2.4 Implementasi Antarmuka Halaman *Map*

Implementasi antarmuka halaman *map* adalah antarmuka hasil dari perancangan sebelumnya. Pada halaman ini pengguna dapat melihat informasi kegiatan sedekah dalam bentuk *marker* dan detail informasi. Implementasi antarmuka halaman *map* dapat dilihat pada Gambar 6.4.

Gambar 6. 4 Implementasi antarmuka halaman *map*

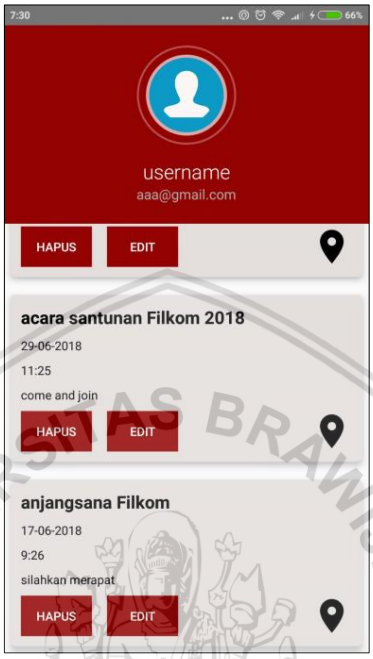
6.1.2.5 Implementasi Antarmuka Halaman *Timeline*

Implementasi antarmuka halaman Timeline adalah antarmuka hasil dari perancangan *user interface* yang telah dibuat. Pada halaman ini pengguna dapat melihat informasi kegiatan sedekah dalam bentuk daftar. Implementasi antarmuka halaman *timeline* dapat dilihat pada Gambar 6.5.

Gambar 6. 5 Implementasi antarmuka halaman *timeline*

6.1.2.6 Implementasi Antarmuka Halaman Profil

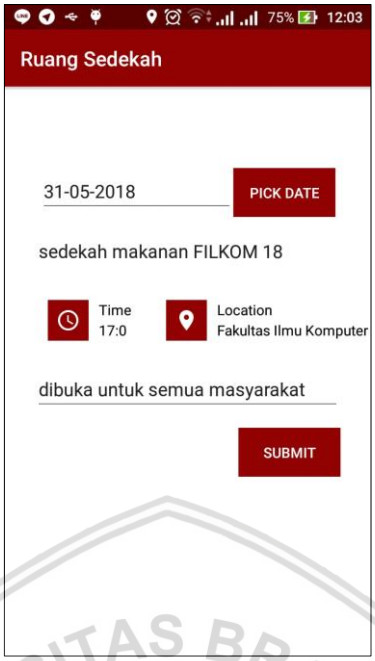
Implementasi antarmuka halaman profil adalah sebagai antarmuka yang akan menampilkan informasi profil dan informasi yang buat oleh *member*. Antarmuka ini dibuat berdasarkan hasil perancangan pada bab sebelumnya. Implementasi antarmuka halaman profil dapat dilihat pada Gambar 6.6.



Gambar 6. 6 Implementasi antarmuka halaman profil

6.1.2.7 Implementasi Antarmuka Halaman Membuat Informasi Kegiatan

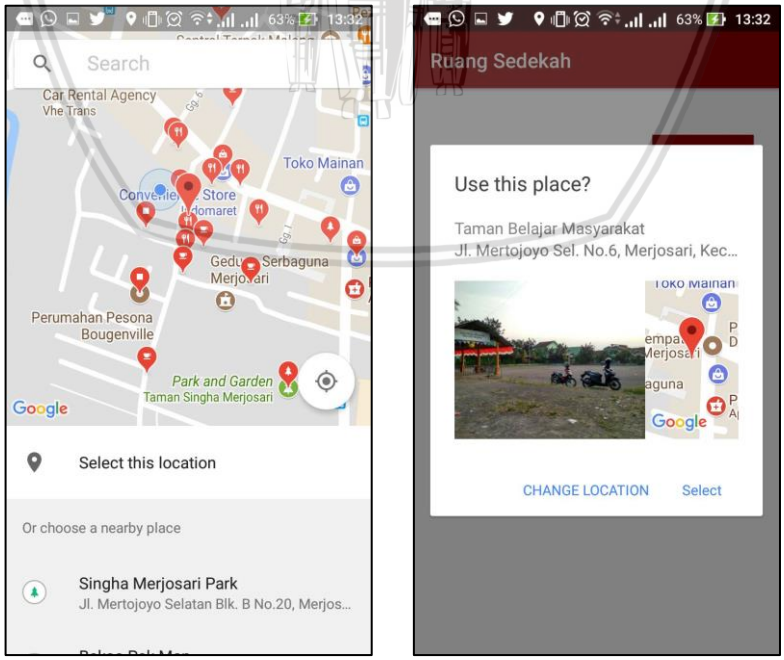
Antarmuka halaman membuat informasi kegiatan adalah antarmuka yang dibuat sesuai dengan hasil perancangan pada bab sebelumnya. Halaman membuat informasi kegiatan menampilkan *field* yang akan diisi sesuai dengan kebutuhan informasi. Implementasi antarmuka halaman profil dapat dilihat pada Gambar 6.7.



Gambar 6. 7 Implementasi antarmuka membuat informasi kegiatan

6.1.2.8 Implementasi Antarmuka Halaman Memilih Lokasi Kegiatan

Implementasi antarmuka halaman memilih lokasi kegiatan adalah halaman antarmuka yang disesuaikan dengan kebutuhan pengguna. Pada halaman ini ditampilkan lokasi-lokasi tertentu sebagai rujukan *member* untuk memilih tempat dilakukan kegiatan sedekah. Implementasi antarmuka halaman memilih lokasi dapat dilihat pada Gambar 6.8.



Gambar 6. 8 Implementasi antarmuka halaman memilih lokasi kegiatan

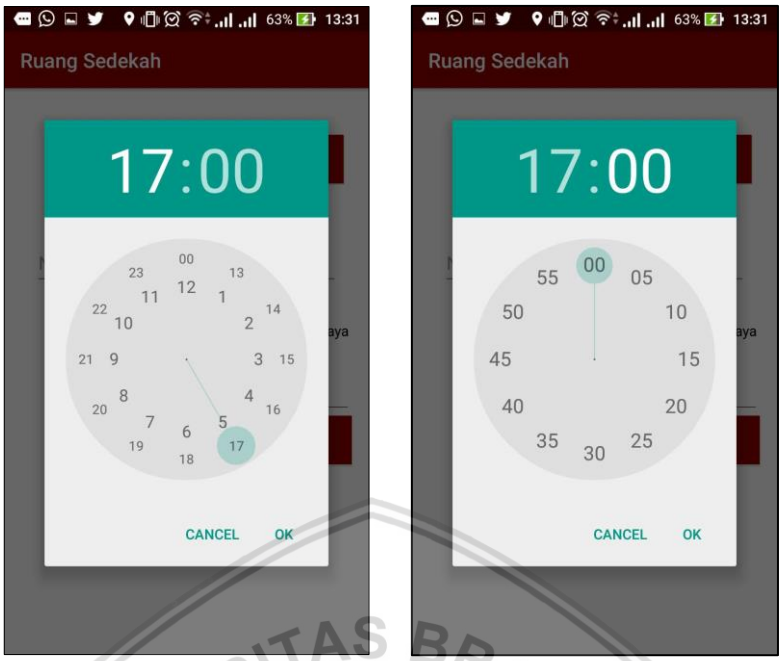
6.1.2.9 Implementasi Antarmuka Halaman Memilih Tanggal Kegiatan

Implementasi antarmuka halaman memilih tanggal kegiatan adalah antarmuka yang sesuai dengan kebutuhan pengguna. Pada halaman ini *member* dapat memilih tanggal kegiatan sedekah berbagi makanan. Implementasi antarmuka halaman memilih tanggal kegiatan dapat dilihat pada Gambar 6.9.

Gambar 6. 9 Implementasi antarmuka halaman memilih tanggal kegiatan

6.1.2.10 Implementasi Antarmuka Halaman Memilih Waktu Kegiatan

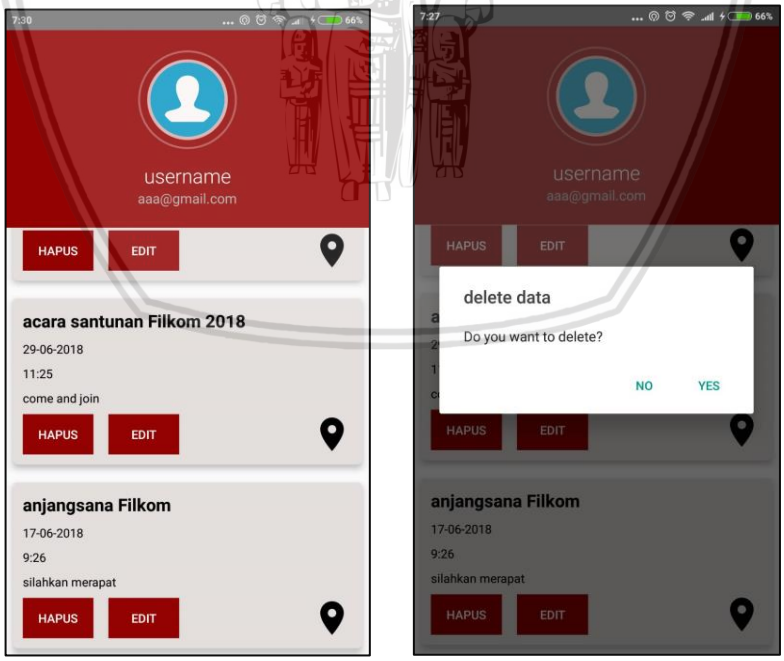
Implementasi antarmuka halaman memilih waktu kegiatan adalah hasil dari *component retrieval*. Pada halaman ini *member* dapat memilih waktu pelaksanaan kegiatan berbagi sedekah. Implementasi antarmuka halaman memilih waktu kegiatan dapat dilihat pada Gambar 6.10.



Gambar 6. 10 Implementasi antarmuka halaman memilih waktu kegiatan

6.1.2.11 Implementasi Antarmuka Halaman Menghapus Informasi Kegiatan

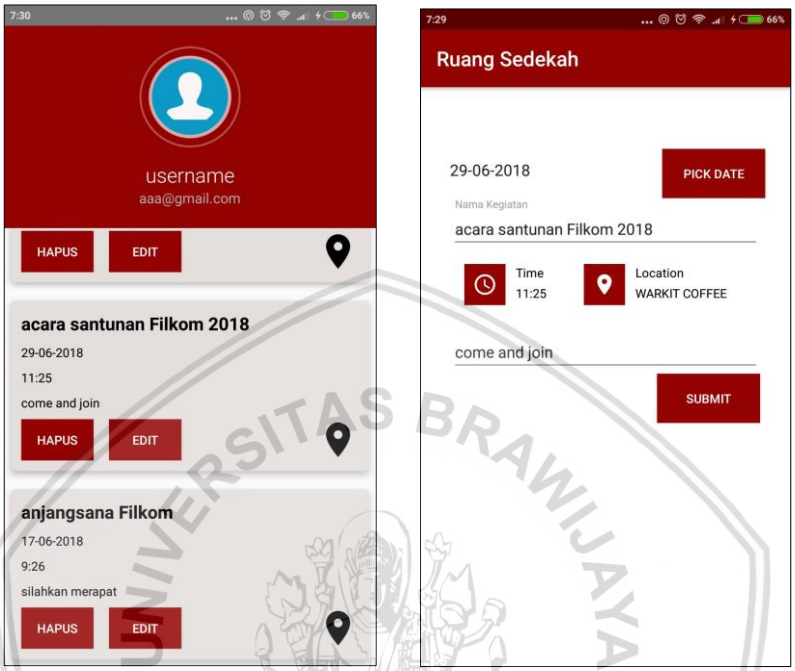
Implementasi antarmuka halaman menghapus informasi kegiatan dibuat sesuai dengan kebutuhan yang telah dirancang sebelumnya. Implementasi antarmuka halaman mengahapus informasi dapat ditunjukkan pada Gambar 6.11.



Gambar 6. 11 Implementasi antarmuka halaman menghapus informasi kegiatan

6.1.2.12 Implementasi Antarmuka Halaman Mengedit Informasi Kegiatan

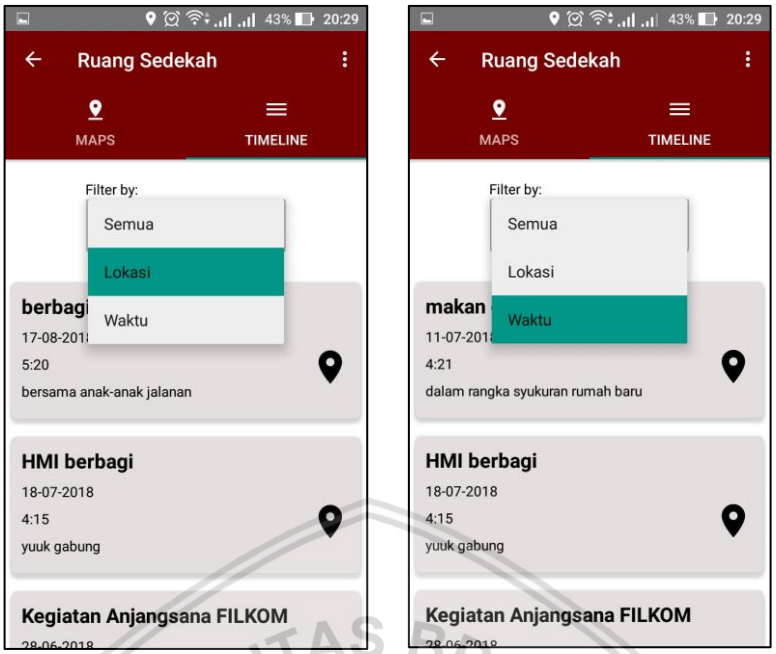
Implementasi antarmuka halaman mengedit informasi kegiatan berisi halaman membuat informasi dengan mengambil data sebelumnya. Antarmuka ini dibuat sesuai dengan perancangan. Implementasi antarmuka halaman mengedit informasi dapat ditunjukkan pada Gambar 6.12.



Gambar 6. 12 Implementasi antarmuka halaman mengedit informasi kegiatan

6.1.2.13 Implementasi Antarmuka Memfilter Informasi

Implementasi antarmuka memfilter informasi dibuat berdasarkan perancangan sebelumnya. Pengguna dapat melihat informasi berdasarkan lokasi atau waktu. Implementasi antarmuka memfilter informasi dapat dilihat pada Gambar 6.13.



Gambar 6. 13 Implementasi antarmuka memfilter informasi

6.1.2.14 Implementasi Antarmuka Halaman *Logout*

Implementasi antarmuka halaman *logout* adalah antarmuka yang dibuat menyesuaikan pada kebutuhan pengguna. Pada halaman ini *member* dapat keluar dari sistem. Implementasi antarmuka halaman *logout* dapat dilihat pada Gambar 6.14.

Gambar 6. 14 Implementasi antarmuka halaman *logout*

6.1.3 Implementasi Kode Program

6.1.3.1 Implementasi Kode Program Membuat Informasi Kegiatan Sedekah

Tabel 6.5 adalah implementasi kode program membuat informasi kegiatan sedekah pada *method* createKegiatan() pada *class* BuatKegiatanActivity.java

Tabel 6. 5 Tabel implementasi kode program membuat informasi kegiatan sedekah

BuatKegiatanActivity.java

```
private void createKegiatan() {

        String date = etDateResult.getText().toString().trim();
        String nama = etNama.getText().toString().trim();
        String desc = etDesc.getText().toString().trim();
        String time = tvTimeResult.getText().toString().trim();
        String loc = tvLocResult.getText().toString().trim();
        String userId = getIntent().getStringExtra("userId");
        double lat = this.lat;
        double lang = this.lang;
        Lokasi lokasi = new Lokasi();
        lokasi.setLat(lat);
        lokasi.setLang(lang);
        lokasi.setNamaTempat(loc);

        if (TextUtils.isEmpty(nama)) {
            Toast.makeText(this, "Masukkan Nama Kegiatan",
Toast.LENGTH_LONG).show();
            return;
        } else if (TextUtils.isEmpty(date)) {
            Toast.makeText(this, "Pilih Tanggal Kegiatan",
Toast.LENGTH_LONG).show();
            return;
        }else if (TextUtils.isEmpty(time)) {
            Toast.makeText(this, "Pilih Waktu Kegiatan",
Toast.LENGTH_LONG).show();
            return;
        }else if (TextUtils.isEmpty(loc)) {
            Toast.makeText(this, "Pilih Lokasi Kegiatan",
Toast.LENGTH_LONG).show();
            return;
        }else{
            String id = mDatabase.push().getKey();
            String lokasiId =
FirebaseDatabase.getInstance().getReference("Lokasi").push().getKey();
            lokasi.setKegiatanId(id);
            lokasi.setLokasiId(lokasiId);

            Kegiatan kegiatan = new Kegiatan(nama, date, time, desc,
id, lokasi, userId);

            mDatabase.child(id).setValue(kegiatan);
FirebaseDatabase.getInstance().getReference("Lokasi").child(lokasiId).s
etValue(lokasi);

            //displaying a success toast
            Toast.makeText(this, "Kegiatan Baru ditambahkan",
Toast.LENGTH_LONG).show();
            Intent intent = new Intent(BuatKegiatanActivity.this,
HalamanUtamaActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
```

```
            startActivity(intent);
        }
    }
```

6.1.3.2 Implementasi Kode Program Menghapus Informasi Kegiatan Sedekah

Tabel 6.6 adalah implementasi Kode Program Menghapus Informasi Kegiatan Sedekah pada *method* btnHapus.setOnClickListener() pada *class* ListBerandaAdapter.java

Tabel 6. 6 Tabel implementasi kode program menghapus informasi kegiatan sedekah

ListBerandaAdapter.java

```
holder.btnHapus.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
   AlertDialog.Builder alert = new AlertDialog.Builder(context);
   alert.setTitle("delete data");
   alert.setMessage("Apakah anda yakin akan menghapus informasi?");
   alert.setPositiveButton("Ya", new DialogInterface.OnClickListener()
{
      @Override
      public void onClick(DialogInterface dialogInterface, int i) {
          mRef.child(kegiatan.getId()).removeValue();
          Toast.makeText(context, "Informasi Kegiatan Dihapus",
Toast.LENGTH_SHORT).show();
          }
      });
          alert.setNegativeButton("Tidak", new
DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface dialogInterface, int i) {
          Toast.makeText(context, "", Toast.LENGTH_SHORT).show();
          }
      });
      alert.create().show();
      }
});
```

6.1.3.3 Implementasi Kode Program Mengedit Informasi Kegiatan

Tabel 6.7 adalah implementasi kode program mengedit informasi kegiatan sedekah pada *method* updateKegiatan() pada *class* BuatKegiatanActivity.java

Tabel 6. 7 Tabel implementasi kode program mengedit informasi kegiatan

BuatKegiatanActivity.java

```
private  void updateKegiatan()
    {
        string date = etDateResult.getText().toString().trim();
        String nama = etNama.getText().toString().trim();
        String desc = etDesc.getText().toString().trim();
        String time = tvTimeResult.getText().toString().trim();
        String loc = tvLocResult.getText().toString().trim();
        String userId = kegiatan.getUserId();
        Double lat = this.lat;
        Double lang = this.lang;
```

```
        Lokasi lokasi = new Lokasi();
        lokasi.setLokasiId(kegiatan.getLokasi().getLokasiId());
        lokasi.setLang(lang);
        lokasi.setLat(lat);
        lokasi.setKegiatanId(kegiatan.getId());
        lokasi.setNamaTempat(loc);


        if (TextUtils.isEmpty(nama)) {
            Toast.makeText(this, "Masukkan Nama Kegiatan",
Toast.LENGTH_LONG).show();
        } else if (TextUtils.isEmpty(date)) {
            Toast.makeText(this, "Pilih Tanggal Kegiatan",
Toast.LENGTH_LONG).show();
        }else if (TextUtils.isEmpty(time)) {
            Toast.makeText(this, "Pilih Waktu Kegiatan",
Toast.LENGTH_LONG).show();
        }else if (TextUtils.isEmpty(loc)) {
            Toast.makeText(this, "Pilih Lokasi Kegiatan",
Toast.LENGTH_LONG).show();
        }else{

            Kegiatan kegiatan = new Kegiatan(nama, date, time, desc,
id, lokasi, userId);

            mDatabase.child(id).setValue(kegiatan);
FirebaseDatabase.getInstance().getReference("Lokasi").child(lokasi.getL
okasiId()).setValue(lokasi);

            Toast.makeText(this, "Informasi berhasil diedit",
Toast.LENGTH_LONG).show();
            Intent intent = new Intent(BuatKegiatanActivity.this,
HalamanUtamaActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
            startActivity(intent);
        }
    }
```

6.1.3.4 Implementasi Kode Program *Logout*

Tabel 6.8 adalah implementasi kode program *logout* pada *method* onOptionSelected() pada *class* HalamanUtamaActivity.java

Tabel 6. 8 Tabel implementasi kode program *logout*

HalamanUtamaActivity.java

```
Public 74oolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.menu_profil) {
        Intent intent = new Intent(this, HalamanProfil.class);
        startActivity(intent);
        finish();
    } else if (id == R.id.menu_logout) {
        Toast.makeText(this, "Logout", Toast.LENGTH_SHORT).show();
        Log.d(TAG, "onClick: attempting to sign out the user.");
        FirebaseAuth.getInstance().signOut();
        Intent intent = new Intent(this, LoginActivity.class);
        startActivity(intent);
        finish();
    } else if (id == R.id.home) {
```

```
        onBackPressed();
    }
    return true;
}
```

6.1.3.5 Implementasi Kode Program Melihat Informasi Kegiatan

Tabel 6.9 adalah implementasi kode program melihat informasi kegiatan pada *method* fetchKegiatanDataFromFirebase(), *method* add*Marker*ToMap() pada *class* MapFragment.java dan *method* onBindViewHolder() pada *class* ListTimerAdapter.java.

Tabel 6. 9 Tabel implementasi kode program melihat informasi kegiatan

MapFragment.java

```
private void fetchKegiatanDataFromFirebase(final GoogleMap googleMap) {
    DatabaseReference reference =
FirebaseDatabase.getInstance().getReference().child("Kegiatan");
    reference.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            for (DataSnapshot ds : dataSnapshot.getChildren()) {
                mKegiatan = ds.getValue(Kegiatan.class);
                if (mKegiatan != null) {
                    Log.d("fetching",
String.valueOf(mKegiatan.getLat()));
                    Log.d("fetching_lat",
String.valueOf(mKegiatan.lat));
                    mListKegiatan.add(mKegiatan);
                }
                addMarkerToMap(mListKegiatan,googleMap);
            }
        }
        @Override
        public void onCancelled(DatabaseError databaseError) {
        }
    });
}
```

MapFragment.java

```
private void addMarkerToMap(ArrayList<Kegiatan> mListKegiatan,
GoogleMap googleMap) {
    for (int i = 0; i < mListKegiatan.size(); i++) {
        final Kegiatan kgtn = mListKegiatan.get(i);
        LatLng position = new
LatLng(mListKegiatan.get(i).getLokasi().getLat(),
mListKegiatan.get(i).getLokasi().getLang());
        Marker mMark = googleMap.addMarker(new
MarkerOptions().position(position)
                .title(kgtn.getNama())
                .snippet(kgtn.toString()));

        googleMap.setInfoWindowAdapter(new
GoogleMap.InfoWindowAdapter() {

            private View view =
getLayoutInflater().inflate(R.layout.marker_view, null);
```

```
            @Override
            public View getInfoWindow(Marker marker) {
                final TextView titleUi = ((TextView)
view.findViewById(R.id.worldmap_infowindow_username));
                String title = marker.getTitle();

                if (title != null) titleUi.setText(title);
                else titleUi.setText("-");
                String[] strings = marker.getSnippet().split("[|]");

                final TextView snippetUi = ((TextView)
view.findViewById(R.id.worldmap_infowindow_name));
                if (strings[0] != null) snippetUi.setText(strings[0]);
                else snippetUi.setText("-");

                final TextView decUi = ((TextView)
view.findViewById(R.id.worldmap_infowindow_details));
                if (strings[1] != null) decUi.setText(strings[1]);
                else snippetUi.setText("-");

                markerList.add(marker);

                return view;
            }

            @Override
            public View getInfoContents(Marker marker) {
                if (marker != null && marker.isInfoWindowShown()) {
                    marker.hideInfoWindow();
                    marker.showInfoWindow();
                }
                return null;
            }
        });
```

ListTimerAdapter.java

```
public void onBindViewHolder(final CategoryViewHolder holder, int
position) {

    final Kegiatan kegiatan = listKegiatan.get(position);

    holder.tvNama.setText(kegiatan.getNama());
    holder.tvTanggal.setText(kegiatan.getTanggal());
    holder.tvWaktu.setText(kegiatan.getWaktu());
    holder.tvDesc.setText(kegiatan.getDeskripsi());

    holder.tvNama.setText(kegiatan.getNama());

    holder.locLokasi.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            for (int i = 0; i < markerList.size(); i++)
            {
                Log.d("LISTTIMELINEADAPTER",
markerList.get(i).getTitle());
                if
(kegiatan.getNama().equals(markerList.get(i).getTitle()))
                {
                    markerList.get(i).showInfoWindow();
                    break;
                }
            }

            LatLng location = new LatLng(kegiatan.getLokasi().getLat(),
```

```
kegiatan.getLokasi().getLang());
            CameraPosition INIT = new
CameraPosition.Builder().target(location).zoom(16.5F).bearing(300F) //
orientation
                    .build();

mMap.moveCamera(CameraUpdateFactory.newCameraPosition(INIT));
            mCallback = (ChangeViewPagerItemListener)
parent.getContext();
            mCallback.item(0);
        }
    });
```

6.1.3.6 Implementasi Kode Program Memfilter Informasi

Tabel 6.10 adalah implementasi kode program memfilter informasi pada *method* spinner.setOnItemSelectedListener() pada *class* Timelinefragment.java.

Tabel 6. 10 Tabel implementasi kode program memfilter informasi

Timelinefragment.java

```
spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
     @Override
     public void onItemSelected(AdapterView<?> adapterView, View view,
int i, long l) {
        switch (i){
           case 0:
           kegiatanItem.clear();
           updateList();
           break;

           case 1:
           kegiatanItem.clear();
           mRef.addChildEventListener(new ChildEventListener() {
              @Override
              public void onChildAdded(DataSnapshot dataSnapshot,
String s) {

              double distance = Math.sqrt(
Math.pow(dataSnapshot.getValue(Kegiatan.class).lang - lang, 2) +
Math.pow(dataSnapshot.getValue(Kegiatan.class).lat - lat, 2)
              );
              if (distance < 0,025) {
kegiatanItem.addFirst(dataSnapshot.getValue(Kegiatan.class));
              adapter.notifyDataSetChanged();
              }
      }
              @Override
              public void onChildChanged(DataSnapshot dataSnapshot,
String s) {
              Kegiatan kegiatan =
dataSnapshot.getValue(Kegiatan.class);
              int index = getItemIndex(kegiatan);
              kegiatanItem.set(index, kegiatan);
              adapter.notifyItemChanged(index);
              }
              @Override
```

```
                public void onChildRemoved(DataSnapshot dataSnapshot) {
                Kegiatan kegiatan =
dataSnapshot.getValue(Kegiatan.class);
                int index = getItemIndex(kegiatan);
                kegiatanItem.remove(index);
                adapter.notifyItemRemoved(index);
                }
                @Override
                public void onChildMoved(DataSnapshot dataSnapshot,
String s) {
                }

                @Override
                public void onCancelled(DatabaseError databaseError) {
                }
      });
                break;

                case 2:
                kegiatanItem.clear();
                mRef.addChildEventListener(new ChildEventListener() {
                @Override
                public void onChildAdded(DataSnapshot dataSnapshot,
String s) {

                Calendar currentDate = Calendar.getInstance();
                SimpleDateFormat dateFormatter = new
SimpleDateFormat("dd-MM-yyyy", Locale.US);
                String dateString =
dateFormatter.format(currentDate.getTime());
                try {
                Date current = dateFormatter.parse(dateString);
                Date eventDate =
dateFormatter.parse(dataSnapshot.child("tanggal").getValue().toString()
);
      long diff = eventDate.getTime() - current.getTime();
      long days = TimeUnit.DAYS.convert(diff, TimeUnit.MILLISECONDS);
      if (days <= 30) {
kegiatanItem.addFirst(dataSnapshot.getValue(Kegiatan.class));
                adapter.notifyDataSetChanged();
                }

      } catch (ParseException e) {
        e.printStackTrace();
      }
}
      @Override
      public void onChildChanged(DataSnapshot dataSnapshot, String s) {
      Kegiatan kegiatan = dataSnapshot.getValue(Kegiatan.class);
      int index = getItemIndex(kegiatan);
      kegiatanItem.set(index, kegiatan);
      adapter.notifyItemChanged(index);
      }
      @Override
      public void onChildRemoved(DataSnapshot dataSnapshot) {
      Kegiatan kegiatan = dataSnapshot.getValue(Kegiatan.class);
      int index = getItemIndex(kegiatan);
```

```
        kegiatanItem.remove(index);
        adapter.notifyItemRemoved(index);
        }
        @Override
        public void onChildMoved(DataSnapshot dataSnapshot, String s) {
}
        @Override
        public void onCancelled(DatabaseError databaseError) {
        }
});
        break;
        }
}
        @Override
        public void onNothingSelected(AdapterView<?> adapterView) {
        }
});
```

6.1.3.7 Implementasi Kode Program Registrasi

Tabel 6.11 adalah implementasi kode program registrasi pada *method* registerUser() pada *class* RegistrationActivity.java.

Tabel 6. 11 Tabel implementasi kode program registrasi

RegistrationActivity.java

```
private void registerUser() {

        final String email = inputEmail.getText().toString().trim();
        final String password =
inputPassword.getText().toString().trim();
        final String nama = inputNama.getText().toString().trim();
        final String no_hp = inputNoHp.getText().toString().trim();

        if (TextUtils.isEmpty(email)) {
            Toast.makeText(getApplicationContext(), "Masukkan Email",
Toast.LENGTH_SHORT).show();
            return;
        }
        if (TextUtils.isEmpty(nama)) {
            Toast.makeText(getApplicationContext(), "Masukkan
Username", Toast.LENGTH_SHORT).show();
            return;
        }
        if (TextUtils.isEmpty(no_hp)) {
            Toast.makeText(getApplicationContext(), "Masukkan Nomor
Hp", Toast.LENGTH_SHORT).show();
            return;
        }

        if (TextUtils.isEmpty(password)) {
            Toast.makeText(getApplicationContext(), "Masukkan
Password", Toast.LENGTH_SHORT).show();
            return;
        }

        if (password.length() < 6) {
            Toast.makeText(getApplicationContext(), "Password terlalu
pendek, masukkan minimal 6 karakter", Toast.LENGTH_SHORT).show();
            return;
        }
```

```
        progressBar.setVisibility(View.VISIBLE);

        mAuth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult>
task) {

                        if (!email.equals("") && !password.equals("")
&& !nama.equals("") && !no_hp.equals("")) {
                            Toast.makeText(RegistrationActivity.this,
"Pengguna Behasil ditambahkan", Toast.LENGTH_LONG).show();
                            inputEmail.setText("");
                            inputPassword.setText("");
                            inputNama.setText("");
                            inputNoHp.setText("");

                            if (task.isSuccessful()) {
                                Member user = new Member(email,
password, nama, no_hp, mAuth.getCurrentUser().getUid());

myRef.child("Member").child(mAuth.getCurrentUser().getUid()).setValue(u
ser);
                            }
                            startActivity(new
Intent(RegistrationActivity.this, HalamanUtamaActivity.class));
                        } else {

                            Toast.makeText(RegistrationActivity.this,
"registrasi gagal", Toast.LENGTH_LONG).show();
                        }

                    }
                });
}
```

6.1.3.8 Implementasi Kode Program *Login*

Tabel 6.12 adalah implementasi kode program *login* pada *method* *login*Btn.setOnClickListener() pada *class* LoginActivity.java.

Tabel 6. 12 Tabel implementasi kode program *login*

LoginActivity.java

```
loginBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String email = inputEmail.getText().toString();
                final String password =
inputPassword.getText().toString();

                if (TextUtils.isEmpty(email)) {
                    Toast.makeText(getApplicationContext(), "Masukkan
Email", Toast.LENGTH_SHORT).show();
                    return;
                }

                if (TextUtils.isEmpty(password)) {
                    Toast.makeText(getApplicationContext(), "Masukkan
Password", Toast.LENGTH_SHORT).show();
                    return;
```

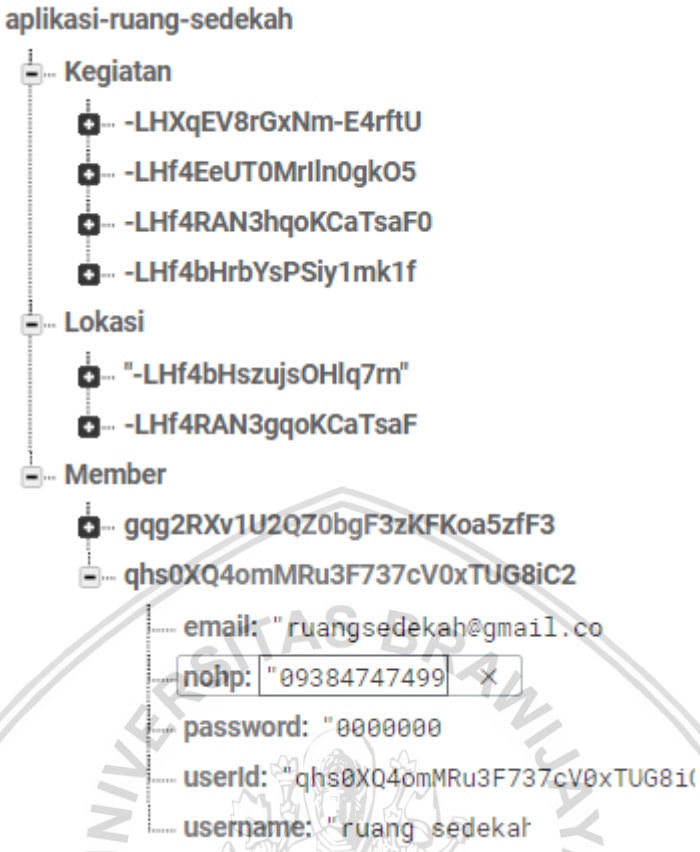
```
                }

                progressBar.setVisibility(View.VISIBLE);

                auth.signInWithEmailAndPassword(email, password)
                        .addOnCompleteListener(LoginActivity.this, new
OnCompleteListener<AuthResult>() {
                            @Override
                            public void onComplete(@NonNull
Task<AuthResult> task) {
                                progressBar.setVisibility(View.GONE);
                                if (!task.isSuccessful()) {
                                    if (password.length() < 6) {

inputPassword.setError(getString(R.string.minimum_password));
                                    } else {

Toast.makeText(LoginActivity.this, getString(R.string.auth_failed),
Toast.LENGTH_LONG).show();
                                    }
                                } else {

                                    Intent intent = new
Intent(LoginActivity.this, HalamanUtama.class);
                                    startActivity(intent);
                                    finish();
                                }
                            }
                        });
            }
        });
}
```

6.1.4 Implementasi Basis Data

Hasil perancangan basis data pada aplikasi manajemen informasi sedekah ini diimplementasikan dengan aplikasi Firebase Realtime Database. Beberapa data hasil implementasi yaitu data Member, data Kegiatan dan data Lokasi.

6.1.4.1 Implementasi Data Member

Pada implementasi data Member berisi atribut pengguna yaitu email, nohp, password, userId dan username. Data Member diimplementasikan langsung dari Android Studio dan *console* Firebase Realtime Database. Karena sifatnya *realtime* maka setiap perubahan data yang dilakukan oleh pengguna akan langsung terhubung pada firebase database. Implementasi data Member dapat dilihat pada Gambar 6.15.



Gambar 6. 15 Implementasi data Member

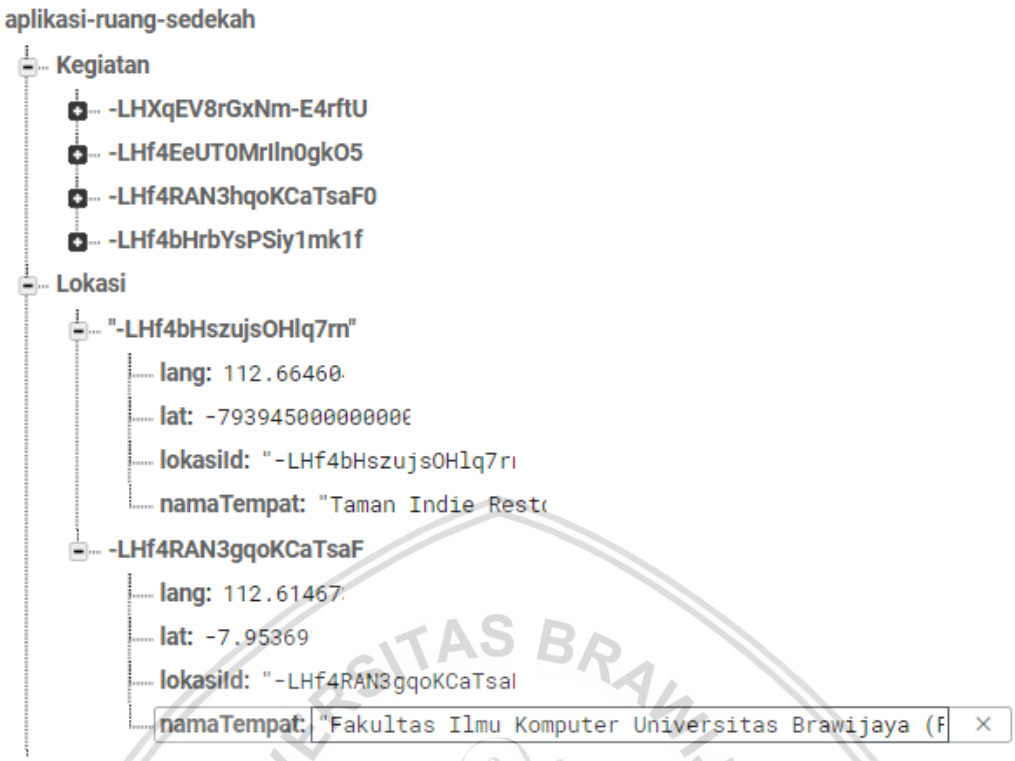
6.1.4.2 Implementasi Data Kegiatan

Pada implementasi data Kegiatan berisi atribut kegiatan yaitu deskripsi, id, lokasiId, nama, tanggal , userId dan waktu. Data Kegiatan diimplementasikan langsung dari Android Studio dan *console* Firebase Realtime Database. Karena sifatnya *realtime* maka setiap perubahan data yang dilakukan akan langsung terhubung pada firebase database. Implementasi data Kegiatan dapat dilihat pada Gambar 6.16.

Gambar 6. 16 Implementasi data Kegiatan

6.1.4.3 Implementasi Data Lokasi

Pada implementasi data Lokasi berisi atribut lokasi yaitu lang, lat, lokasiId dan namaTempat. Data Lokasi diimplementasikan langsung dari Android Studio dan *console* Firebase Realtime Database. Karena sifatnya *realtime* maka setiap perubahan data yang dilakukan akan langsung terhubung pada firebase database. Implementasi data Lokasi dapat dilihat pada Gambar 6.17.



Gambar 6. 17 Implementasi data Lokasi

6.2 Pengujian

Pengujian pada aplikasi Menejemen Informasi Sedekah Berbagi Makanan terdiri dari pengujian unit, pengujian integrasi dan pengujian validasi. Pada pengujian unit dan pengujian integrasi menggunakan teknik pengujian *white box testing* dengan *basis path testing*. Pengujian ini dimulai dengan membuat *flowgraph* dari masing-masing fungsi atau kebutuhan, membuat *cyclomatic complexity* untuk mendapatkan jumlah *path* dan membuat jalur *independent path.* Tahap selanjutnya pada pengujian *white box* ini adalah membuat *test case* dari *independent path* yang telah dibuat sebelumnya. Pada pengujian validasi dilakukan dengan menggunakan *black box testing*. Pengujian validasi dilakukan pada seluruh kebutuhan fungsional dan nonfungsional.

6.2.1 Pengujian Unit

Pengujian dilakukan untuk menguji unit-unit yang memiliki prioritas tinggi dalam sistem untuk memperoleh hasil sesuai dengan yang diharapkan. Pada pengujian unit ini menggunakan model *basis path testing.* Pada pengujian unit ini hanya dilakukan pada 3 sampel *method* yaitu *method* spinner.setOnItemSelectedListener() pada *class* Timelinefragment, *method* onOptionItemSelected() pada *class* HalamanUtamaActivity dan *method* loginBtn.setOnClickListener() pada *class* LoginActivity.

1. Pengujian algoritme *method* onCreateView() pada *class* Timelinefragment

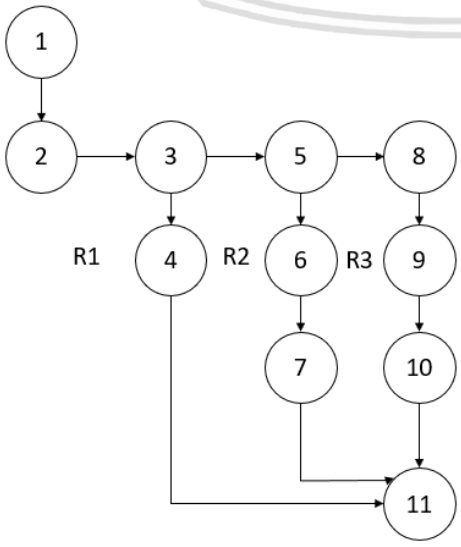
Tabel 6.13 merupakan algoritme *method* onCreateView() pada *class* Timelinefragment.

Tabel 6. 13 Algoritme *method* onCreateView() pada *class* Timelinefragment

Firebase database mDatabase	
Database reference mRef	
LinkedList kegiatanItem	
linkedList lokasi	*1*
RecyclerView rvCategory	
ListTimelineAdapter adapter	
Membuat event pada objek spinner	
Switch(i)	*2*
Case 0	*3*
Update List	*4*
Case 1	*5*
If distance < 0.050	*6*
Tampilkan kegiatan	*7*
Case 2	*8*
If days <= 30	*9*
Tampilkan kegiatan	*10*
End case	*11*

Untuk menentukan *Basis Path Testing,* maka dibuat *Flowgraph, Cyclomatic Complexity,* dan *Independent Path.* Berikut cara menentukan *basis path testing* untuk *method* onCreateView().

1. Flowgraph



2. *Cyclomatic Complexity*

   V(G) = Jumlah Region = 3

   V(G) = E – N + 2 = 9 – 8 + 2 = 3

   N(G) = P + 1 = 2 + 1 = 3

3. *Independent Path*

   - Jalur 1: 1-5-6-4
   - Jalur 2: 1-5-7-8-4
   - Jalur 3: 1-5-7-2-3-4

Tabel 6.14 merupakan hasil pengujian dari algoritme *method* onCreateView() pada *class* Timelinefragment berdasarkan *independent path* di atas.

Tabel 6. 14 *Test case* algoritme *method* onCreateView() pada *class* Timelinefragment

No. Jalur	Prosedur Uji	Expected Result	Result	Status
1.	Menjalankan class TimelineFragment melalui class driver HalamanUtamaActivity dan memanggil *method* onCreateView() dengan nilai variabel i = 1	Informasi kegiatan dengan jarak lokasi kurang dari 10 km dimasukkan ke dalam list kegiatanItem	Informasi kegiatan dengan jarak lokasi kurang dari 10 km dimasukkan ke dalam list kegiatanItem	Valid
2.	Menjalankan class TimelineFragment melalui class driver HalamanUtamaActivity dan memanggil *method* onCreateView() dengan nilai variabel i = 2	Informasi kegiatan dengan waktu kurang dari 30 hari dimasukkan ke dalam list kegiatanItem	Informasi kegiatan dengan waktu kurang dari 30 hari dimasukkan ke dalam list kegiatanItem	Valid
3.	Menjalankan class TimelineFragment melalui class driver HalamanUtamaActivity dan memanggil method onCreateView() dengan nilai variabel i = 0	Semua informasi kegiatan dimasukan ke dalam list kegitanItem	Semua informasi kegiatan dimasukan ke dalam list kegitanItem	Valid

2. Pengujian algoritme *method* onOptionItemSelected() pada *class* HalamanUtamaActivity

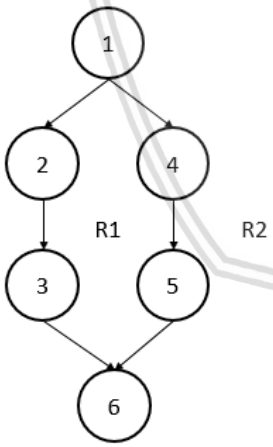
Tabel 6.15 merupakan algoritme *method* onOptionItemSelected() pada *class* HalamanUtamaActivity.

Tabel 6. 15 Algoritme *method* onOptionItemSelected() pada *class* HalamanUtamaActivity

int id	*1*
if (id == menu_profil)	*2*
tampilkan HalamanProfil	*3*
startActivity	
else if (id == menu_logout)	*4*
print "Logout"	*5*
FirebaseAuth.getInstance = signOut	
tampilkan LoginActivity	
startActivity	
end if	*6*

Untuk menentukan *basis path testing,* maka dibuat *Flowgraph, Cyclomatic Complexity,* dan *Independent Path.* Berikut cara menentukan *basis path testing* untuk *method* onOptionItemSelected().

1. *Flowgraph*



2. *Cyclomatic Complexity*

   V(G) = Jumlah Region = 2

   V(G) = E – N + 2 = 6 – 6 + 2 = 3

   N(G) = P + 1 = 1 + 1 = 2

3. *Independent Path*

   - Jalur 1: 1-2-3-6

- Jalur 2: 1-4-5-6

Tabel 6.16 merupakan hasil pengujian dari algoritme *method* onOptionItemSelected() pada *class* HalamanUtamaActivity berdasarkan *independent path* di atas.

Tabel 6. 16 *Test case* algoritme *method* onOptionItemSelected() pada *class* HalamanUtamaActivity

No. Jalur	Prosedur Uji	Expected Result	Result	Status
1.	Menjalankan *class* HalamanUtamaActivity sebagai class driver dan memanggil *method* onOptionItemSelected() dengan variabel id = R.id.menu_profil	Berhasil menampilkan halaman profil	Berhasil menampilkan halaman profil	Valid
2.	Menjalankan *class* HalamanUtamaActivity sebagai class driver dan memanggil *method* onOptionItemSelected() dengan variabel id = R.id_menu_logout	Sistem menghapus *session member* dan menampilkan halaman awal	Sistem menghapus *session member* dan menampilkan halaman awal	Valid

3. Pengujian algoritme *method* loginBtn.setOnClickListener() pada *class* LoginActivity

Tabel 6.17 adalah algoritme *method* loginBtn.setOnClickListener() pada *class* LoginActivity.java

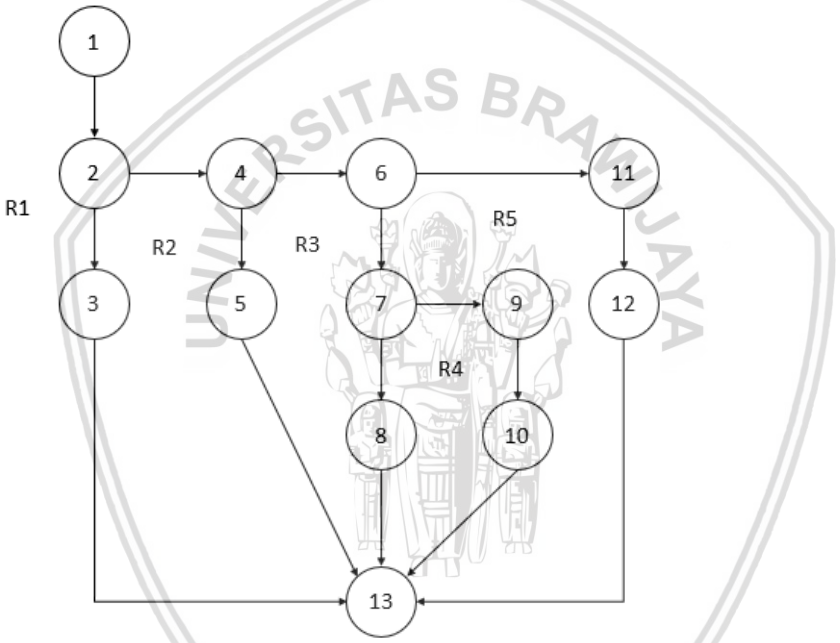
Tabel 6. 17 Algoritme *method* loginBtn.setOnClickListener() pada *class* LoginActivity

```
String email = inputEmail                                              1
String password = inputPassword
    if (email = null)  ------------------------------------------------> 2
        Print "Masukkan Email"                                          3
        Return
    if (password = null)  ---------------------------------------------> 4
        Print "Masukkan Password"                                       5
        Return
        auth.signInWithEmailAndPassword(email, password)                        6
        if (! Successfull login)
            if (password.length < 6) {  ------------------------------------------> 7
```

Print "password terlalu singkat, minimal 6 karakter"	*8*
else	*9*
Print "Login eror"	*10*
else	*11*
Tampilkan HalamanUtama *StartActivity*	*12*
end if	*13*

Untuk menentukan *Basis Path Testing,* maka dibuat *Flowgraph, Cyclomatic Complexity,* dan *Independent Path.* Berikut cara menentukan *Basis Path Testing* untuk *method* loginBtn.setOnClickListener().

1. *Flowgraph*



2. *Cyclomatic Complexity*

   V(G) = Jumlah Region = 5

   V(G) = E – N + 2 = 16 – 3 + 2 = 5

   N(G) = P + 1 = 4 + 1 = 5

3. *Independent Path*

   - Jalur 1: 1-2-3-13
   - Jalur 2: 1-2-4-5-13
   - Jalur 3: 1-2-4-6-7-8-13
   - Jalur 4: 1-2-4-6-7-9-10-13
   - Jalur 5: 1-2-4-6-11-12-13

Tabel 6.18 merupakan hasil pengujian dari algoritme *method* loginBtn.setOnClickListener() pada *class* LoginActivity berdasarkan *independent path* di atas.

Tabel 6. 18 *Test case* algoritme *method* loginBtn.setOnClickListener() pada *class* LoginActivity

No. Jalur	Prosedur Uji	Expected Result	Result	Status
1	Menjalankan *class* LoginActivity melalui class driver HalamanAwalActivity dan memanggil *method* loginBtn.setOnClickListener() dengan variabel email = null	Sistem menampilkan pesan "Masukkan Email"	Sistem menampilkan pesan "Masukkan Email"	Valid
2.	Menjalankan *class* LoginActivity melalui class driver HalamanAwalActivity dan memanggil *method* loginBtn.setOnClickListener() dengan variabel password = null	Sistem menampilkan pesan "Masukkan Password"	Sistem menampilkan pesan "Masukkan Password"	Valid
3.	Menjalankan *class* LoginActivity melalui class driver HalamanAwalActivity dan memanggil *method* loginBtn.setOnClickListener() dengan variabel password < 6 karakter	Sistem menampilkan pesan "*password* terlalu pendek, masukan minimal 6 karakter"	Sistem menampilkan pesan "*password* terlalu pendek, masukan minimal 6 karakter"	Valid
4.	Menjalankan *class* LoginActivity melalui class driver HalamanAwalActivity dan memanggil *method* loginBtn.setOnClickListener() dengan variabel email "ruangsedekah@gmail.com" dan password "1234567"	Sistem menampilkan pesan "Email atau *Password* Salah, Periksa Lagi"	Sistem menampilkan pesan "Email atau *Password* Salah, Periksa Lagi"	Valid
5.	Menjalankan *class* LoginActivity melalui class driver HalamanAwalActivity dan memanggil *method* loginBtn.setOnClickListener()	Berhasil login dan menampilkan halaman utama	Berhasil login dan menampilkan halaman utama	Valid

	dengan variabel email "ruangsedekah@gmail.com" dan password "0000000"			

6.2.2 Pengujian Integrasi

Pengujian integrasi adalah pengujian yang bertujuan untuk menguji interaksi klas dengan klas lain dengan menggunakan *basis path testing.* Pada pengujian integrasi ini dilakukan 3 sampel pengujian. Pengujian yang pertama pada createKegiatan() pada *class* BuatKegiatanActivity yang memanggil *method* konstruktor kegiatan (String nama, String tanggal, String waktu, String deskripsi, String id, Lokasi lokasi, String userId) pada *class* model kegiatan. Pengujian kedua pada *method* btnHapus.setOnClickListener() pada *class* ListBerandaAdapter yang memanggil *method* getId() di *class* model Kegiatan. Pengujian ketiga pada *method* registerUser() pada *class* RegistrationActivity yang memanggil konstruktor Member (String email, String password, String username, String nohp, String userId) dari *class* model *member*.

1. **Pengujian algoritme *method* createKegiatan() pada *class* BuatKegiatanActivity**

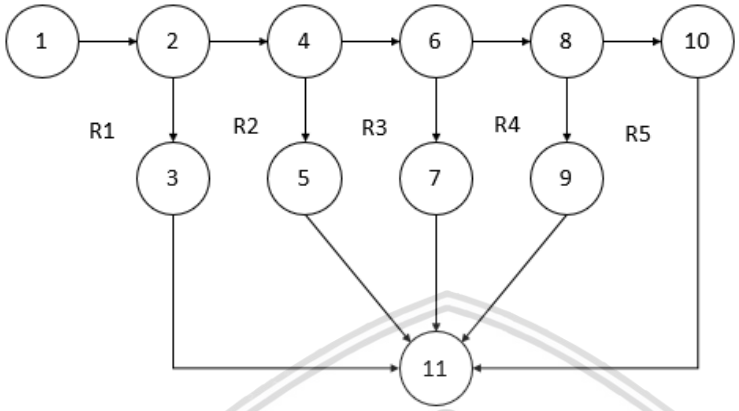
Tabel 6.19 merupakan algoritme createKegiatan() pada *class* BuatKegiatanActivity.

Tabel 6. 19 Algoritme *method* createKegiatan() pada *class* BuatKegiatanActivity

String date = etDateResult. *String nama = etNama* *String desc = etDesc* *String time = tvTimeResult* *String loc = tvLocResult*	*1*
if (nama = null)	*2*
print "Masukkan Nama Kegiatan"	*3*
else if (date = null)	*4*
print "Pilih Tanggal Kegiatan"	*5*
else if (time = null)	*6*
print "Pilih Waktu Kegiatan"	*7*
else if loc = null	*8*
Print "Pilih Lokasi Kegiatan"	*9*
Else *Print "Kegiatan ditambahkan"* *Tampilkan HalamanUtama* *StartActivity*	*10*
End if	*11*

Untuk menentukan *basis path testing,* maka dibuat *Flowgraph, Cyclomatic Complexity,* dan *Independent Path.* Berikut cara menentukan *basis path testing* untuk *method* createKegiatan().

1. *Flowgraph*



2. *Cyclomatic Complexity*

   V(G) = Jumlah Region = 5

   V(G) = E – N + 2 = 14 – 11 + 2 = 5

   N(G) = P + 1 = 4 + 1 = 5

3. *Independent Path*

   - Jalur 1: 1-2-3-11
   - Jalur 2: 1-2-4-5-11
   - Jalur 3: 1-2-4-6-7-11
   - Jalur 4: 1-2-4-6-8-9-11
   - Jalur 5: 1-2-4-6-8-10-11

Tabel 6.20 merupakan hasil pengujian dari algoritme *method* createKegiatan() pada *class* BuatKegiatanActivity berdasarkan *independent path* di atas.

Tabel 6. 20 *Test case* algoritme *method* createKegiatan() pada *class* BuatKegiatanActivity

No. Jalur	Prosedur Uji	Expected Result	Result	Status
1.	Menjalankan class BuatKegiatanActivity melalui class driver HalamanUtamaActivity dan memanggil *method* createKegiatan(), mengisi form halaman	Menampilkan pesan "Masukkan Nama Kegiatan"	Menampilkan pesan "Masukkan Nama Kegiatan"	Valid

	membuat informasi kegiatan dengan variabel nama = null			
2.	Menjalankan class BuatKegiatanActivity melalui class driver HalamanUtamaActivity dan memanggil *method* createKegiatan(), mengisi form halaman membuat informasi kegiatan dengan variabel date = null	Menampilkan pesan "Masukkan Tanggal Kegiatan"	Menampilkan pesan "Masukkan Tanggal Kegiatan"	Valid
3.	Menjalankan class BuatKegiatanActivity melalui class driver HalamanUtamaActivity dan memanggil *method* createKegiatan(), mengisi form halaman membuat informasi kegiatan dengan variabel time = null	Menampilkan pesan "Masukkan Waktu Kegiatan"	Menampilkan pesan "Masukkan Waktu Kegiatan"	Valid
4.	Menjalankan class BuatKegiatanActivity melalui class driver HalamanUtamaActivity dan memanggil *method* createKegiatan(), mengisi form halaman membuat informasi kegiatan dengan variabel loc = null	Menampilkan pesan "Masukkan Lokasi Kegiatan"	Menampilkan pesan "Masukkan Lokasi Kegiatan"	Valid
5.	Menjalankan class BuatKegiatanActivity melalui class driver HalamanUtamaActivity dan memanggil *method* createKegiatan(), mengisi form halaman membuat informasi	Sistem memasukkan informasi kegiatan baru ke dalam database dan menampilkan pesan "Kegiatan	Sistem memasukkan informasi kegiatan baru ke dalam database dan menampilkan pesan "Kegiatan	Valid

	kegiatan dengan lengkap	Baru ditambahkan”	Baru ditambahkan”	

2. Pengujian algoritme *method* btnHapus.setOnClickListener() pada *class* ListBerandaAdapter

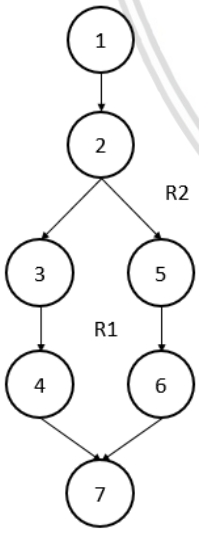
Tabel 6.21 merupakan algoritme *method* btnHapus.setOnClickListener() pada *class* ListBerandaAdapter.

Tabel 6. 21 Algoritme *method* btnHapus.setOnClickListener() pada *class* ListBerandaAdapter

AlertDialog alert	*1*
Set Title "delete data"	*2*
Set Message "Anda yakin menghapus kegiatan?"	
Set PositiveButton ("Ya")	*3*
removeValue	*4*
print “kegiatan berhasil dihapus”	
Set NegativeButton("Tidak")	*5*
Tutup alert dialog	*6*
end	*7*

Untuk menentukan *basis path testing,* maka dibuat *Flowgraph, Cyclomatic Complexity,* dan *Independent Path.* Berikut cara menentukan *basis path testing* untuk *method* btnHapus.setOnClickListener().

1. *Flowgraph*



2. *Cyclomatic Complexity*

V(G) = Jumlah Region = 2

V(G) = E – N + 2 = 7 – 7 + 2 = 2

N(G) = P + 1 = 1 + 1 = 2

3. *Independent Path*

- Jalur 1: 1-2-3-4-7
- Jalur 2: 1-2-5-6-7

Tabel 6.22 merupakan hasil pengujian dari algoritme *method* btnHapus.setOnClickListener() pada *class* ListBerandaAdapter berdasarkan *independent path* di atas.

Tabel 6. 22 *Test case* algoritme *method* btnHapus.setOnClickListener() pada *class* ListBerandaAdapter

No. Jalur	Prosedur Uji	Expected Result	Result	Status
1	Menjalankan class ListBerandaAdapter melalui class driver HalamanProfilActivity dan memanggil *method* btnHapus.setOnClickListener() dengan *setPositiveButton("Yes")*	Berhasil menghapus kegiatan dan menampilkan pesan "Informasi dihapus"	Berhasil menghapus kegiatan dan menampilkan pesan "Informasi dihapus"	Valid
2	Menjalankan class ListBerandaAdapter melalui class driver HalamanProfilActivity dan memanggil *method* btnHapus.setOnClickListener() dengan *setNegativeButton("No")*	Gagal menghapus kegiatan	Gagal menghapus kegiatan	Valid

3. Pengujian algoritme *method* registerUser() pada *class* RegistrationActivity

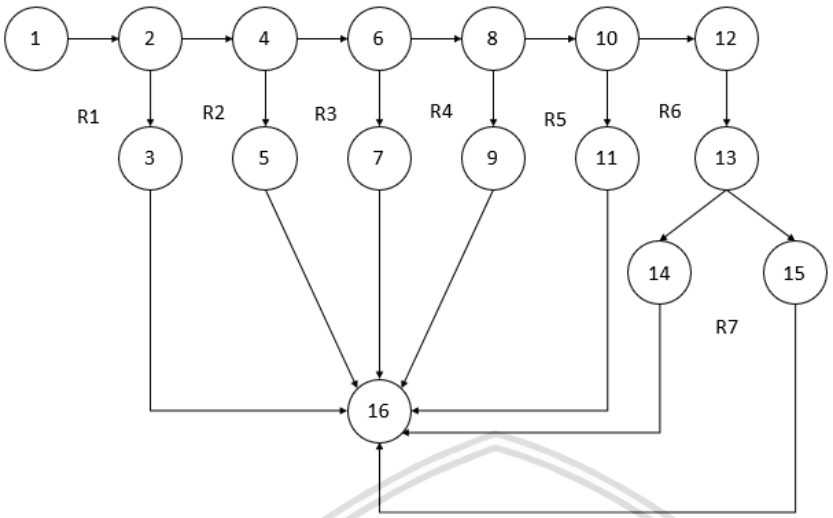
Tabel 6.23 merupakan algoritme *method* registerUser() pada *class* RegistrationActivity.

Tabel 6. 23 Algoritme *method* registerUser() pada *class* RegistrationActivity

```
String email = inputEmail                                                    1
String password = inputPassword
String nama = inputNama
String no_hp = inputNoHp
    if (email = null)                                                        2
      Print "Masukkan Email"                                                3
      return
    if (nama = null)                                                         4
      Print "Masukkan Username"                                             5
      return
    if (no_hp = null)                                                        6
      Print "Masukkan Nomor Hp"                                             7
      return
    if (password = null)                                                     8
      Print "Masukkan Password"                                             9
      return
    if (password.length < 6)                                                 10
      Print "Password terlalu pendek, masukkan minimal 6 karakter"          11
      return
    mAuth.createUserWithEmailAndPassword(email, password)                    12
      if (email && password && nama && no_hp != null) {
        Print "Pengguna Behasil ditambahkan"
      if (successfull register)                                              13
        SetValue(user);                                                      14
        Tampilkan HalamanUtamaActivity
      else                                                                   15
        Print "registrasi gagal"
end                                                                          16
```

Untuk menentukan *Basis Path Testing,* maka dibuat *Flowgraph, Cyclomatic Complexity,* dan *Independent Path.* Berikut cara menentukan *Basis Path Testing* untuk *method* registerUser().

1. *Flowgraph*



2. *Cyclomatic Complexity*

   V(G) = Jumlah Region = 7

   V(G) = E – N + 2 = 21 – 16 + 2 = 7

   N(G) = P + 1 = 6 + 1 = 7

3. *Independent Path*

   - Jalur 1: 1-2-3-16
   - Jalur 2: 1-2-4-5-16
   - Jalur 3: 1-2-4-6-7-16
   - Jalur 4: 1-2-4-6-8-9-16
   - Jalur 5: 1-2-4-6-8-10-11-16
   - Jalur 6: 1-2-4-6-8-10-12-13-14-16
   - Jalur 7: 1-2-4-6-8-10-12-13-15-16

Tabel 6.24 merupakan hasil pengujian dari algoritme *method* registerUser() pada *class* RegistrationActivity berdasarkan *independent path* di atas.

Tabel 6. 24 *Test case* algoritme *method* registerUser() pada *class* RegistrationActivity

No. Jalur	Prosedur Uji	Expected Result	Result	Status
1	Memanggil *method* registerUser() dengan variabel email = null	Menampilkan pesan kesalahan "Masukkan Email"	Menampilkan pesan kesalahan "Masukkan Email"	Valid

2	Memanggil *method* registerUser() dengan variabel nama = null	Menampilkan pesan kesalahan "Masukkan Username"	Menampilkan pesan kesalahan "Masukkan Username"	Valid
3	Memanggil *method* registerUser() dengan variabel no_hp = null	Menampilkan pesan kesalahan "Masukkan Nomor Hp"	Menampilkan pesan kesalahan "Masukkan Nomor Hp"	Valid
4	Memanggil *method* registerUser() dengan variabel password = null	Menampilkan pesan kesalahan "Masukkan Password"	Menampilkan pesan kesalahan "Masukkan Password"	Valid
5	Memanggil *method* registerUser() dengan variabel password < 6 karakter	Menampilkan pesan kesalahan "Password terlalu pendek, masukkan minimal 6 karakter"	Menampilkan pesan kesalahan "Password terlalu pendek, masukkan minimal 6 karakter"	Valid
6.	Memanggil *method* registerUser() dengan mengisi form registrasi dengan lengkap	Sistem memasukkan data member baru pada database dan menampilkan pesan "Pengguna berhasil ditambahkan"	Sistem memasukkan data member baru pada database dan menampilkan pesan "Pengguna berhasil ditambahkan"	Valid
7.	Memanggil *method* registerUser() dengan mengosongkan semua *field*	Menampilkan pesan kesalahan "Registrasi Gagal"	Menampilkan pesan kesalahan "Registrasi Gagal"	Valid

6.2.3 Pengujian Validasi

Pengujian validasi adalah model pengujian *black box testing*. Pengujian validasi dilakukan untuk menguji semua kebutuhan fungsional dan nonfungsional apakah

berjalan dengan baik atau tidak. Pengujian validasi kebutuhan fungsional dilakukan dengan membuat kasus uji untuk setiap kebutuhan fungsional, sedangkan kebutuhan nonfungsional yaitu *usability* dan *compatibility*

6.2.3.1 Pengujian Fungsional

Kasus uji dibutuhkan untuk setiap kebutuhan fungsional dari sistem. kasus uji ini akan merepresentasikan secara detail prosedur uji dan hasil yang diharapkan. Kasus uji dibuat sebagai prosedur untuk menentukan kesesuaian kebutuhan yang telah didefinisikan dengan hasil implementasi. Berikut kasus uji pengujian validasi fungsional.

1. Kasus Uji Membuat Informasi Kegiatan

Tabel 6.25 adalah kasus uji kebutuhan fungsional mengedit informasi yang menjelaskan prosedur pengujian mengedit informasi beserta hasil yang diharapkan dari hasil pengujian.

Tabel 6. 25 Kasus uji fungsi membuat informasi kegiatan

Nomor Kasus Uji	RS-TEST-F-1.1
Nama Kasus Uji	Membuat informasi kegiatan
Objek Uji	Kebutuhan fungsional mengedit informasi kegiatan (RS-F-1.1)
Tujuan pengujian	Pengujian dilakukan untuk membuktikan bahwa aplikasi dapat memenuhi salah satu kebutuhan fungsional yaitu fungsi membuat informasi kegiatan sedekah
Prosedur Pengujian	1. Menjalankan aplikasi 2. mengklik tombol “Sign In” 3. Mengisi *form login* dengan benar 4. Mengklik tombol “Sign In” 5. Mengklik tombol (+) pada *map* 6. Mengisi form membuat informasi kegiatan dengan benar 7. Mengklik tombol “Submit”
Hasil yang Diharapkan	Aplikasi dapat melakukan validasi terhadap perintah dengan menambah data baru ke dalam *database* serta menampilkan informasi kegiatan pada detail *marker* dan daftar kegiatan pada *timeline*
Hasil yang didapatkan	Aplikasi berhasil menambah data pada *database* serta menampilkan informasi kegiatan pada detail *marker* dan daftar kegiatan pada *timeline*
Status validasi	Valid

2. Kasus Uji Menghapus Informasi Kegiatan

Tabel 6.26 adalah kasus uji kebutuhan fungsional menghapus informasi yang menjelaskan prosedur pengujian menghapus informasi beserta hasil yang diharapkan dari hasil pengujian.

Tabel 6. 26 Kasus uji fungsi menghapus informasi kegiatan

Nomor Kasus Uji	RS-TEST-F-1.2
Nama Kasus Uji	Menghapus informasi kegiatan
Objek Uji	Kebutuhan Fungsional menghapus informasi kegiatan (RS-F-1.2)
Tujuan pengujian	Pengujian dilakukan untuk membuktikan bahwa aplikasi dapat memenuhi salah satu kebutuhan fungsional yaitu fungsi menghapus informasi dari informasi yang telah dibuat
Prosedur Pengujian	1. Menjalankan aplikasi 2. mengklik tombol "Sign In" 3. Mengisi *form login* dengan benar 4. Mengklik tombol "Sign In" 5. Mengklik menu profil 6. Memilih informasi kegiatan yang ingin dihapus 7. Mengklik menu hapus 8. Mengklik "Ya"
Hasil yang Diharapkan	Aplikasi dapat melakukan validasi terhadap perintah dengan menghapus data dari *database* dan informasi yang ditampilkan
Hasil yang didapatkan	Aplikasi berhasil menghapus data dari *database* dan informasi yang ditampilkan
Status validasi	Valid

3. Kasus Uji Mengedit Informasi Kegiatan

Tabel 6.27 adalah kasus uji kebutuhan fungsional mengedit informasi yang menjelaskan prosedur pengujian mengedit informasi beserta hasil yang diharapkan dari hasil pengujian.

Tabel 6. 27 Kasus uji fungsi mengedit informasi kegiatan

Nomor Kasus Uji	RS-TEST-F-1.3
Nama Kasus Uji	Mengedit informasi kegiatan
Objek Uji	Kebutuhan fungsional mengedit informasi kegiatan (RS-F-1.3)

Tujuan pengujian	Pengujian dilakukan untuk membuktikan bahwa aplikasi dapat memenuhi salah satu kebutuhan fungsional yaitu fungsi mengedit informasi dari informasi yang telah dibuat
Prosedur Pengujian	1. Menjalankan aplikasi 2. mengklik tombol "Sign In" 3. Mengisi *form login* dengan benar 4. Mengklik tombol "Sign In" 5. Mengklik menu profil 6. Memilih informasi kegiatan yang ingin diedit 7. Klik tahan informasi 8. Mengklik menu edit 9. Mengedit *form* membuat informasi dengan data yang ada 10. Mengklik tombol "submit"
Hasil yang Diharapkan	Aplikasi dapat melakukan validasi terhadap perintah dengan mengubah data pada *database* dan informasi yang ditampilkan.
Hasil yang didapatkan	Aplikasi berhasil mengubah data pada *database* dan informasi yang ditampilkan
Status validasi	Valid

4. Kasus Uji *Logout*

Tabel 6.28 adalah kasus uji kebutuhan fungsional *logout* yang menjelaskan prosedur pengujian *logout* beserta hasil yang diharapkan dari hasil pengujian.

Tabel 6. 28 Kasus uji fungsi *logout*

Nomor Kasus Uji	RS-TEST-F-1.4
Nama Kasus Uji	*Logout*
Objek Uji	Kebutuhan fungsional *logout* (RS-F-1.4)
Tujuan pengujian	Pengujian dilakukan untuk membuktikan bahwa aplikasi dapat memenuhi salah satu kebutuhan fungsional yaitu *logout*. Dimana *member* dapat keluar dari sistem.
Prosedur Pengujian	1. Menjalankan aplikasi 2. mengklik tombol "Sign In" 3. Mengisi *form login* dengan benar 4. Mengklik tombol "Sign In" 5. Mengklik menu *logout*

Hasil yang Diharapkan	Aplikasi dapat melakukan validasi terhadap perintah dengan menghapus *member session* dan menampilkan halaman awal
Hasil yang didapatkan	Aplikasi berhasil melakukan fungsi *logout* dengan menghapus *session member* dan menampilkan halaman awal
Status validasi	Valid

5. Kasus Uji Melihat Informasi Kegiatan

Tabel 6.29 adalah kasus uji kebutuhan fungsional melihat informasi kegiatan yang menjelaskan prosedur pengujian melihat informasi kegiatan beserta hasil yang diharapkan dari hasil pengujian.

Tabel 6. 29 Kasus uji fungsi melihat informasi kegiatan

Nomor Kasus Uji	RS-TEST-F-1.5
Nama Kasus Uji	Melihat informasi kegiatan
Objek Uji	Kebutuhan Fungsional melihat informasi (RS-F-2.1)
Tujuan pengujian	Pengujian dilakukan untuk membuktikan bahwa aplikasi dapat memenuhi salah satu kebutuhan fungsional yaitu fungsi melihat informasi yang dapat dilakukan oleh semua pengguna
Prosedur Pengujian	1. Menjalankan aplikasi 2. mengklik *text link* "lanjut ke halaman utama"
Hasil yang Diharapkan	Aplikasi dapat melakukan validasi terhadap perintah dan menampilkan informasi kegiatan berupa detail informasi pada *marker map* dan daftar informasi pada timeline.
Hasil yang didapatkan	Aplikasi berhasil menampilkan informasi kegiatan berupa detail informasi pada *marker map* dan daftar informasi pada *timeline*
Status validasi	Valid

6. Kasus Uji Memfilter Informasi Kegiatan

Tabel 6.30 adalah kasus uji kebutuhan fungsional memfilter informasi kegiatan yang menjelaskan prosedur pengujian memfilter informasi kegiatan beserta hasil yang diharapkan dari hasil pengujian.

Tabel 6. 30 Kasus uji fungsi memfilter informasi kegiatan

Nomor Kasus Uji	RS-TEST-F-1.6
Nama Kasus Uji	Memfilter informasi kegiatan

Objek Uji	Kebutuhan Fungsional memfilter informasi kegiatan (RS-F-2.2)
Tujuan pengujian	Pengujian dilakukan untuk membuktikan bahwa aplikasi dapat memenuhi salah satu kebutuhan fungsional yaitu memfilter informasi kegiatan berdasarkan lokasi tertentu atau berdasarkan waktu tertentu
Prosedur Pengujian	1. Menjalankan aplikasi 2. Login sistem dengan menginputkan email dan *password* yang benar 3. Mengklik tombol "Sign In" 4. Mengklik tab "Timeline" 5. Mengklik tombol filter 6. Memilih filter berdasarkan lokasi atau waktu
Hasil yang Diharapkan	Aplikasi dapat melakukan validasi pada perintah dan menampilkan hasil filter informasi kegiatan berdasarkan lokasi tertentu atau waktu tertentu
Hasil yang didapatkan	Aplikasi berhasil menampilkan hasil filter informasi kegiatan berdasarkan lokasi tertentu dan waktu tertentu
Status validasi	Valid

7. Kasus Uji Registrasi

Tabel 6.31 adalah kasus uji kebutuhan fungsional registrasi yang menjelaskan prosedur pengujian registrasi beserta hasil yang diharapkan dari hasil pengujian.

Tabel 6. 31 Kasus uji fungsi registrasi

Nomor Kasus Uji	RS-TEST-F-1.7
Nama Kasus Uji	Registrasi
Objek Uji	Kebutuhan Fungsional Registrasi (RS-F-2.3)
Tujuan pengujian	Pengujian dilakukan untuk membuktikan bahwa aplikasi dapat memenuhi salah satu kebutuhan fungsional yaitu fungsi registrasi dimana pengguna dapat mendaftarkan diri sebagai *member*
Prosedur Pengujian	1. Menjalankan aplikasi 2. mengklik tombol "Sign Up" 3. Mengisi form sign up dengan benar 4. Mengklik tombol "Sign Up"

Hasil yang Diharapkan	Aplikasi dapat melakukan validasi terhadap data yang diinputkan dengan menambahkan data *member* baru pada *database* dan menampilkan halaman utama
Hasil yang didapatkan	Aplikasi berhasil menjalankan fungsi registrasi dengan menambahkan data *member* baru pada *database* dan menampilkan halaman utama
Status validasi	Valid

8. Kasus Uji *Login*

Tabel 6.32 adalah kasus uji kebutuhan fungsional *login* yang menjelaskan prosedur pengujian *login* beserta hasil yang diharapkan dari hasil pengujian.

Tabel 6. 32 Kasus uji fungsi *login*

Nomor Kasus Uji	RS-TEST-F-1.8
Nama Kasus Uji	*Login*
Objek Uji	Kebutuhan Fungsional *Login* (RS-F-2.4)
Tujuan pengujian	Pengujian dilakukan untuk membuktikan bahwa aplikasi dapat memenuhi salah satu kebutuhan fungsional yaitu fungsi *login*
Prosedur Pengujian	1. Menjalankan aplikasi 2. mengklik tombol “Sign In” 3. Mengisi form sign in dengan benar 4. Mengklik tombol “Sign In”
Hasil yang Diharapkan	Aplikasi dapat melakukan validasi terhadap data yang diinputkan dan menampilkan halaman utama
Hasil yang didapatkan	Aplikasi berhasil menjalankan fungsi *login* dengan menampilkan halaman utama
Status validasi	Valid

6.2.3.2 Pengujian Nonfungsional

Pengujian validasi nonfungsional dilakukan untuk menguji usabilitas dan compatibilitas aplikasi. Kasus uji dibuat sebagai prosedur untuk menentukan kesesuaian kebutuhan yang telah didefinisikan dengan hasil implementasi. Berikut kasus uji pengujian validasi nonfungsional.

1. Kasus Uji *usability*

Pengujian *usability* dilakukan dengan menggunakan USE questionnaire dengan jumlah 30 pertanyaan yang mencakup seluruh subkarakteristik *usability*. Pengujian *usability* dilakukan oleh 10 orang penguji. Untuk dapat merepresentasikan nilai hasil pengujian digunakan perhitungan Skala Likert dengan interpretasi skor dengan interval 20 poin. Interpretasi skor skala likert dapat dilihat pada Tabel 6.33.

Tabel 6. 33 Tabel interpretasi skor skala likert

Skor likert	Interpretasi skor	Pilihan	Keterangan
5	80%-100%	Sangat tinggi	Interval 20% didapatkan dari nilai 100% dibagi jumlah skor likert
4	60%-79,99%	Tinggi	
3	40%-59,99%	Cukup tinggi	
2	20%-39,99%	Rendah	
1	0%-19,99%	Rendah sekali	

Hasil pengujian *usability* berupa total skor yang didapatkan dari jumlah pengujian dikalikan dengan skor penilaian. index persentase diperoleh dari total skor dibagi dengan skor maksimum yang diberikan. Sedangkan untuk menghitung skor persentase kelayakan menggunakan rumus (Lund, 2001):

$$kelayakan\ (\%) = \frac{total\ skor}{jumlah\ likert\ tertinggi} \times 100\%$$

Daftar pertanyaan USE Questionnaire dan Hasil pengujian *usability* dapat dilihat pada Lampiran A.1. Kasus uji fungsi *usability* dapat dilihat pada Tabel 6.34.

Tabel 6. 34 Kasus uji fungsi *usability*

Nomor Kasus Uji	RS-TEST-NF-1.9
Nama Kasus Uji	*Usability*
Objek Uji	Kebutuhan Nonfungsional *usability* (RS-NF-1.1)
Tujuan pengujian	Pengujian dilakukan untuk membuktikan bahwa aplikasi layak digunakan
Prosedur Pengujian	1. Penguji menginstal aplikasi pada device masing-masing 2. Penguji mencoba aplikasi selama 10 menit 3. Penguji mengisi kuisioner yang diberikan
Hasil yang Diharapkan	Aplikasi dapat melakukan validasi terhadap usabilitas aplikasi dengan hasil aplikasi layak digunakan
Hasil yang didapatkan	Dari hasil perhitungan persamaan pengujian *usability*, didapatkan skor nilai 83,2% yang menunjukkan aplikasi layak digunakan
Status validasi	Valid

2. Kasus Uji *compatibility*

Tabel 6.35 adalah kasus uji kebutuhan non fungsional *compatibility.* Pengujian *compatibility* dilakukan dengan menguji sistem android sdk 16, sdk 21 dan sdk 26. Hasil pengujian *compatilibity* dapat dilihat pada Lampiran A.2.

Tabel 6. 35 Kasus uji fungsi *compatibility*

Nomor Kasus Uji	RS-TEST-NF-1.10
Nama Kasus Uji	*Compatibility*
Objek Uji	Kebutuhan Nonfungsional *compatibility* (RS-NF-1.2)
Tujuan pengujian	Pengujian dilakukan untuk membuktikan bahwa aplikasi dapat berjalan pada level SDK 16-26
Prosedur Pengujian	1. Penguji menginstal *virtual machine* pada sistem operasi android dengan sdk 16, sdk 21 dan sdk 26 pada emulator aplikasi Genymotion 2. Penguji menjalankan programkan aplikasi manajemen informasi sedekah pada sdk 16, sdk 21 dan sdk 26
Hasil yang Diharapkan	Aplikasi dapat melakukan validasi terhadap kompatibilitas aplikasi dimana program dapat berjalan pada sistem android sdk 16, sdk 21 dan sdk 26
Hasil yang didapatkan	Aplikasi berhasil dijalankan pada sistem android sdk 16, sdk 21 dan sdk 26
Status validasi	Valid

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil analisis kebutuhan, analisis komponen dan perancangan, serta implementasi dan pengujian sistem yang dilakukan, maka dapat diambil kesimpulan sebagai berikut:

1. Berdasarkan hasil analisis kebutuhan, didapatkan 8 kebutuhan fungsional dan 2 kebutuhan nonfungsional. Kebutuan fungsional sistem terdapat kebutuhan fungsional utama yaitu membuat informasi kegiatan dan melihat informasi kegiatan sedekah berbagi makanan.
2. Berdasarkan hasil analisis komponen dan perancangan yang dilakukan, didapatkan hasil analisis komponen, *sequence diagram,* diagram komponen, *class diagram,* perancangan algoritme, perancangan basis data sebagai acuan dalam melakukan proses implementasi selanjutnya. Pada analisis komponen didapatkan 4 komponen penggunaan ulang yang tersedia yang memenuhi kebutuhan sistem.
3. Berdasarkan hasil implementasi didapatkan hasil implementasi berupa implementasi antarmuka, implementasi kode program dan implementasi basis data.
4. Berdasarkan hasil pengujian unit, pengujian integrasi dengan menggunakan teknik *white box testing* dan *black box testing* dan pengujian validasi dengan teknik *black box testing* diperoleh hasil bahwa sistem lulus uji dan dinyatakan valid tanpa adanya eror. Pada pengujian nonfungsional *usability* didapatkan nilai 83,2 % dan pengujian *compatibility* dinyatakan aplikasi layak digunakan.

7.2 Saran

Saran yang dapat diberikan untuk pengembangan Aplikasi Manajemen Informasi Sedekah Berbagi Makanan selanjutnya antara lain:

1. Aplikasi Manajemen Informasi Sedekah Berbagi Makanan dapat dikembangkan lagi dengan menambah fitur *reminder* sedekah.
2. Setelah ditambahkan fitur *reminder* sedekah, sistem dapat menambahkan fitur lainnya yaitu notifikasi kegiatan sedekah berdasarkan tempat tertentu dan waktu tertentu.

DAFTAR PUSTAKA

Pressman, Roger., 2009. Software Engineering: A Practitioner's Approach. 7th ed. New York: McGraw-Hill Book Company.

Somerville, Ian., 2011. Software Engineering. 9 th ed. Boston: Addison-Wesley.

Fowler, M., 2003. UML Distilled: A Brief Guide of the Standard Object Modeling Languange (3rd edition). Boston: Addison Wesley.

Bakshi, Amandeep. & Bawa, Seema., 2013. *Thesis Project: Development of a Software Repository for the Precise Search and Exact Retrieval of the Components.* S2. Thapar University. Tersedia di < https://pdfs.semanticscholar.org/50b9/12102efb0f3726508737ebaef1fc5db9d176.pdf> [Diakses 25 Juni 2018].

Dwipratma, Agung Pandu., 2011. *Sistem Informasi Manajemen Zakat, Infak, dan Sedekah pada Badan Amil Zakat Nasional*. S1. Universitas Islam Negeri Syarif Hidayatullah. Tersedia di < http://repository.uinjkt.ac.id/dspace/bitstream/123456789/210/1/101215-AGUNG%20PANDU%20DWIPRATAMA-FST.PDF > [Diakses 25 Juni 2018].

Edward., 2014. *Jurnal: Perancangan Sistem Informasi Manajemen Zakat.* S1. STMIK STIKOM Denpasar. Tersedia di < https://media.neliti.com/media/publications/131143-ID-perancangan-sistem-informasi-manajemen-z.pdf > [Diakses 25 Juni 2018].

Arms, William Y., 2001. *Computing and Information Science CS 5150 Software Engineering Scenarios.* New york. Tersedia di < https://www.coursehero.com/file/12177187/D2-use-cases/> [Diakses 20 Juli 2018].

Fahrul, Mu'is., 2016. *Dikejar Rezeki Dari Sedekah.* Solo: Taqiyah Publishing.

Maulan, Rizka., 2008. Syarah Hadist. *Makna Shadaqah.* [online] Tersedia di < https://www.dakwatuna.com/2008/04/30/573/makna-shadaqah/#axzz5JPpQfPm2> [Diakses 25 Juni 2018].

Dutta, Stobak., Sengupta, Sabnam., 2015. Retrieval of software component version from a software version database: A graph based approach. [online] Tersedia di https://ieeexplore.ieee.org/document/7164706/ [Diakses 9 Juli 2018].

Prahasta, E., 2009. *Sistem Informasi Geografis.* Bandung: Penerbit Informatika.

Kadibagil, Mahesh, Guruprasad, S., 2014*. Position Detection and Tracking System.* Bangalore: BMS College of Engineering.

Pendleton, Greg., 2002. *The Fudamental of GPS*. [online] Tersedia di http://www.directionsmag.com/articles/the-fundamentals-of-gps/124028 [Diakses 9 Julis 2018].

Google Developers, 2018. *Firebase Services.* [online] Tersedia di: <https://developers.google.com/actions/tools/assistant-firebase-services> [Diakses 25 Junis 2018].

Android Arsenal, 2014. *Android developer portal with tools, libraries, and apps.* [online] Tersedia di: https://android-arsenal.com/ [Diakses 25 Juni 2018].

Google Developers, 2018. *Place Picker.* [online] Tersedia di: https://developers.google.com/places/android-sdk/placepicker [Diakses 25 Juni 2018].

Github, Inc., 2018. *Learn Git and GitHub without any code.* [online] Tersedia di: https://github.com/ [Diakses 25 Juni 2018].

Ed, Burnette., 2010. *Hello Android Introducing Google's Mobile Development Platform*. 3nd Ed. Texas: The Pragmatic Programmers.

Lund, A, M., 2001. *Measuring usability with the USE questionnaire.* [online] Tersedia di <*www.stcsig.org/usability/newsletter/index.html*> [Diakses 12 Juli 2018].