PENGEMBANGAN APLIKASI ANGKUT SAMPAH DENGAN MENGGUNAKAN METODE *PROTOTYPING*

Disusun Oleh:

N a m a : Tesar Firstyaji Pramudya

NIM : 18523244

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2023

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN APLIKASI ANGKUT SAMPAH DENGAN MENGGUNAKAN METODE *PROTOTYPING*

TUGAS AKHIR

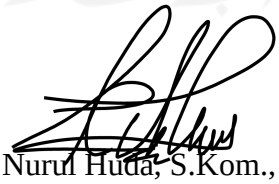
Disusun Oleh:

N a m a : Tesar Firstyaji Pramudya

NIM : 18523244

Yogyakarta, 15 Desember 2022

Pembimbing,



(Sheila Nurul Huda, S.Kom., M.Cs.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN APLIKASI ANGKUT SAMPAH DENGAN MENGGUNAKAN METODE *PROTOTYPING*

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 12 Januari 2023

Tim Penguji

Sheila Nurul Huda, S.Kom., M.Cs. _____

Anggota 1

Chanifah Indah Ratnasari, S.Kom., M.Kom. _____

Anggota 2

Fayruz Rahma, S.T., M.Eng. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

iv

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Tesar Firstyaji Pramudya

NIM : 18523244

Tugas akhir dengan judul:

PENGEMBANGAN APLIKASI ANGKUT SAMPAH DENGAN MENGGUNAKAN METODE PROTOTYPING

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 12 Desember 2022

(Tesar Firstyaji Pramudya)

HALAMAN PERSEMBAHAN

Pertama-tama saya panjatkan puji dan syukur atas nikmat dan kasih sayang Allah Swt yang telah memberikan kekuatan dan kesehatan serta pertolongan-Nya sehingga saya dapat menyelesaikan Tugas Akhir (TA) dengan lancar tanpa hambatan. Tak lupa selawat serta salam selalu tercurahkan kepada junjungan Nabi Muhammad saw dan semoga mendapatkan syafaatnya di hari akhir. Penulisan tugas akhir ini saya persembahkan untuk Bapak Soleh dan Ibu Sugiyaningsih selaku kedua orang tua serta kedua adik saya, yaitu Zahra Nur Fayza Pramesty dan Keisha Zharifah Alfitri yang selalu mendukung baik secara moral maupun materiel dan memberikan dorongan semangat setiap saat yang tiada henti-hentinya dalam menyelesaikan tugas akhir. Untuk itu, saya sangat berterima kasih atas segalanya yang telah orang tua berikan sehingga saya dapat menempuh pendidikan sampai di titik ini. Tidak lupa juga untuk tim saya, yaitu Scapper yang menjadi titik awal dalam membangun sebuah *start-up* dengan memiliki anggota yang semangatnya tinggi, saling mendukung, dan segala suka maupun duka dalam pengembangan tetap konsisten selalu bersama-sama. Untuk teman-teman saya, terima kasih juga atas dukungan, motivasi, dan segala hal yang membantu kelancaran dalam penyusunan laporan ini. Selaku dosen pembimbing saya, yaitu Ibu Sheila Nurul Huda, saya juga mengucapkan terima kasih atas bimbingannya dengan sabar dan selalu memberikan arahan serta masukkan selama penulisan tugas akhir.

HALAMAN MOTO

“Bekerjalah untuk duniamu, seakan-akan engkau hidup selamanya. Dan beramallah untuk akhiratmu, seakan-akan esok hari engkau meninggal dunia.”

(Ibnu Umar Radhiyallahu Anhuma)

“Pengetahuan tanpa tindakan adalah sia-sia, dan tindakan tanpa pengetahuan adalah kegilaan.”

(Abu Hamid Al Ghazali)

“Great things are not done by impulse, but a series of small things brought together.”

(Vincent van Gogh)

“Betapa ringan langkah kita jika diawali doa dan senyuman karena itu menggambarkan ketulusan hati yang kuat dalam menghadapi banyak hal.”

(Mario Teguh)

KATA PENGANTAR

Tiada kata yang paling pantas untuk diucapkan selain rasa syukur atas nikmat dan karunia-Nya yang telah memberikan kesehatan dan kekuatan serta rida-Nya dalam menyelesaikan laporan Tugas Akhir (TA). Penulisan laporan bertujuan untuk memenuhi sebagian persyaratan kelulusan dalam memperoleh gelar Sarjana S-1 (S1) Jurusan Informatika, Fakultas Teknologi Industri di Universitas Islam Indonesia. Adapun judul tugas akhir ini adalah **"Pengembangan Aplikasi Angkut Sampah Dengan Menggunakan Metode *Prototyping*.**

Selama melaksanakan perintisan bisnis sampai penyusunan tugas akhir ini, penulis tidak lepas dari banyaknya pihak yang dengan senang hati meluangkan waktu, pikiran, dan tenaga serta doa, motivasi, dukungan baik secara moral maupun materiel yang tak henti-hentinya kepada penulis yang juga menjadi semangat positif untuk menyelesaikan tugas akhir ini. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu dalam penyelesaian tugas akhir ini baik langsung maupun tidak langsung kepada:

1. Kedua orang tuaku yang telah memberikan dukungan yang luar biasa dan pengorbanan yang tiada hentinya serta doa setiap langkah penulis menuju aktivitas pendidikannya.
2. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
3. Bapak Prof. Dr. Ir. Hari Purnomo, M.T., selaku Dekan Fakultas Teknologi Industri.
4. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Jurusan Informatika Fakultas Teknologi Industri.
5. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Ketua Program Studi Informatika – Program Sarjana.
6. Ibu Sheila Nurul Huda, S.Kom., M.Cs., selaku dosen pembimbing yang telah memberikan bimbingan, dukungan, saran, dan perhatian dalam penelitian dan penyusunan tugas akhir.
7. Ibu Chanifah Indah Ratnasari, S.Kom., M.Kom., selaku dosen penguji yang telah memberikan saran dan bimbingannya dalam penulisan tugas akhir ini.
8. Ibu Fayruz Rahma, S.T., M.Eng., selaku dosen penguji yang telah memberikan saran dan bimbingannya dalam penulisan tugas akhir ini.
9. Teman-teman tim Scapper yang merupakan tim perintisan bisnis bagi penulis yang terdiri dua anggota lain yang telah berkenan untuk membangun sebuah bisnis

dengan semangatnya yang tinggi dan kerja samanya dalam penelitian dan penyusunan tugas akhir.

10. Segenap teman-teman keluarga besar Prodi Informatika - Program Sarjana Angkatan 2018 yang senantiasa memberikan bantuan dan dukungannya.
11. Semua pihak yang tidak dapat disebutkan satu per satu, yang telah dengan tulus ikhlas membantu penulis dalam menyelesaikan tugas akhir ini.

Sebagai penulis tentu menyadari bahwa tugas akhir ini masih jauh dari sempurna. Oleh karena itu, dalam penulisan laporan tugas akhir ini apabila terdapat kekurangan, penulis mengharapkan saran dan masukkan untuk menyempurnakan karya ini. Semoga tugas akhir penulis dapat bermanfaat dan berguna bagi orang lain.

Yogyakarta, 23 November 2022

(Tesar Firstyaji Pramudya)

SARI

Sampah merupakan sesuatu hal yang menjadi masalah sosial, di mana volume sampah secara terus-menerus meningkat dan diikuti dengan meningkatnya populasi penduduk serta keterbatasan lahan, sarana, dan prasarana yang terjadi di sejumlah wilayah Indonesia, seperti daerah Yogyakarta yang termasuk penyumbang sampah terbesar. Masyarakat sering kali membuang sampah di tempat umum, seperti sungai, aliran irigasi, dan danau. Transportasi sampah sebagai proses dalam pengangkutan sampah, masih terbatas dari sisi armada dinas pengelola sampah. Pengangkutan sampah tidak dapat hanya dibatasi pada armada dari pemerintah daerah, melainkan dapat memanfaatkan keberadaan armada dari pengepul sampah. Dalam pengangkutan sampah pada masyarakat sering terjadi keterlambatan pengambilan akibat jadwal pengambilan yang akan diangkut tergolong tidak teratur setiap minggunya dan adanya pengabaian dalam pengangkutan sampah pada hari pengangkutan. Penelitian ini dilakukan untuk memberikan solusi alternatif kepada masyarakat untuk mengurangi penumpukan dan pembuangan sampah dengan baik melalui aplikasi angkut sampah (Bersih Kotaku) sebagai platform digital yang menawarkan jasa angkut sampah yang dilakukan pengepul sampah berdasarkan titik lokasi pengangkutan dari masyarakat. Sistem ini dikembangkan menggunakan bahasa pemrograman Java dengan metode pengembangan *prototyping*. Penggunaan dari metode ini tidak hanya rancangan desain dari *prototype* saja, tetapi menghasilkan sebuah sistem yang sudah di-*develop* dengan luaran berupa aplikasi yang nyata dan siap untuk digunakan oleh pengguna. Aplikasi angkut sampah dikembangkan dalam dua aplikasi berdasarkan jenis penggunanya, yaitu masyarakat dan pengepul sampah. Aplikasi ini dibangun bertujuan dapat menjadi salah satu *start-up* yang mampu bertahan di tengah masyarakat. *Start-up* aplikasi angkut sampah akan memperoleh pendapatan melalui pengepul sampah yang melakukan sewa terhadap aplikasi admin. Aplikasi angkut sampah telah dilakukan pengujian melalui metode *blackbox* dengan seluruh skenario pengujian dan hasil pengujian pada aplikasi tersebut berjalan dengan baik tanpa ada kesalahan kode program pada sistem.

Kata kunci: pembuangan sampah, pengangkutan sampah, aplikasi, java, *prototyping*, *blackbox*.

GLOSARIUM

Debugger	tindakan untuk menelusuri dan mengidentifikasi adanya kesalahan kode program dalam sebuah aplikasi.
Gradle	sebuah cara untuk membangun alat otomatisasi yang digunakan untuk pengembangan perangkat lunak. Dengan ini, proses kompilasi, *packaging*, *testing*, *deploying*, dan *pubslishing* serta instalasi *library* yang dapat diatur dengan *gradle.*
IDE	sebuah *software* di dalam lingkungan pengembangan yang berfungsi untuk memfasilitasi berbagai *tools* pemrograman dalam satu aplikasi yang digunakan.
JSON	format untuk menyimpan, membaca, serta pertukaran data antara *server* dan pengguna. Format ini memiliki struktur kode sederhana, *file* yang lebih ringan, dan format berbentuk teks sehingga mudah dibaca manusia dan dipahami oleh komputer.
Platform	sebuah kombinasi perangkat lunak dan perangkat keras yang digunakan sebagai wadah untuk menjalankan aplikasi yang dikembangkan.
Prototyping	metode dalam pengembangan perangkat lunak.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Sampah menjadi suatu masalah sosial yang terjadi di wilayah Indonesia yang setiap tahunnya volume sampah secara terus-menerus meningkat sehingga menyebabkan tingginya pencemaran lingkungan. Berdasarkan data dari Indonesia National Plastic Action Partneship, setiap tahun sampah di Indonesia menghasilkan 67,2 juta ton sampah yang masih tertimbun di berbagai tempat pembuangan akhir (TPA), 620 ribu ton masih dibuang di tempat umum, seperti sungai, laut, aliran irigasi dan danau. (Jimmy Ramadhan Azhari, 2021). Pada tahun 2018 hingga sekarang pemicu tingginya lonjakan sampah telah disumbangkan oleh sampah rumah tangga khususnya jenis sampah dari sisa makanan. Sampah-sampah sisa makanan cukup berbahaya jika mendominasi terus-menerus untuk beberapa tahun ke depan karena jenis sampah tersebut menyebabkan aroma di lingkungan sekitar menjadi tidak sehat dibandingkan jenis sampah lainnya. Keberadaan masalah sampah tersebut sering ditemukan di daerah kota besar yang memiliki jumlah penduduk cukup tinggi, seperti Yogyakarta yang termasuk penyumbang sampah terbesar. Jumlah penduduk daerah Yogyakarta pada tahun 2021 sebanyak 3.970.220 jiwa dan mengalami peningkatan sebanyak 4.021.816 pada tahun 2022 berdasarkan data Badan Pusat Statistik (Badan Pusat Statistik, 2021). Peningkatan jumlah penduduk harus sebanding terhadap sarana dan prasarana yang dapat menjamin keberlangsungan hidup masyarakat Yogyakarta. Permasalahan sampah di Yogyakarta masih belum sepenuhnya terselesaikan dengan baik. Seperti diketahui, masih banyak masyarakat yang belum menyadari dampak jangka panjang dari pembuangan sampah secara liar dan sampah dibiarkan menumpuk begitu saja yang diikuti dengan peningkatan jumlah penduduk dan makin bervariasinya sampah yang disebabkan oleh makin beragam aktivitas masyarakat. Hal itu, menuntut keharusan menciptakan lingkungan ramah dan bersih untuk tetap memberikan kenyamanan masyarakat sehingga mengurangi dampak buruk, seperti kerusakan lingkungan yang baru, polusi sampah, dan menghambat proses air tanah. Dengan banyaknya sampah yang masih dibuang di tempat umum mengindikasi bahwa masyarakat masih kurang kesadarannya terhadap lingkungannya.

Transportasi sampah sebagai proses dalam pengangkutan sampah yang akan membawa sampah dari lokasi sumber sampah secara langsung menuju tempat pembuangan akhir (TPA) masih terbatas dengan adanya armada dari dinas pengelola sampah. Masyarakat tidak harus bergantung secara terus-menerus dengan armada dari pemerintah daerah, melainkan

memanfaatkan keberadaan armada dari pengepul sampah yang ikut serta membantu mengurangi penumpukan sampah. Dalam pengangkutan sampah pada masyarakat sering terjadi keterlambatan pengambilan akibat jadwal pengambilan yang akan diangkut tergolong tidak teratur setiap minggunya dan adanya pengabaian dalam pengangkutan sampah pada hari pengangkutan. Selain itu, proses pembayaran masih dilakukan secara konvensional. Dengan pembayaran masih konvensional, pengepul sampah sering mendapati masyarakat yang terlambat membayar jasa dan terdapat masyarakat yang tidak membayar jasa terhadap penyedia jasa angkut sampah.

Keberadaan sampah akan terus berkelanjutan karena sampah menjadi hasil aktivitas manusia yang berasal dari berbagai tempat yang ditinjau berdasarkan sumber sampah yang dihasilkan. Sampah berdasarkan sumbernya terdiri dari sampah permukiman dan sampah nonpermukiman. Sampah permukiman merupakan sampah yang telah tercampur dari berbagai sampah lainnya, seperti sampah basah, sampah kering, dan sisa makanan atau sayuran. Sampah tersebut biasanya dihasilkan oleh kalangan keluarga atau mahasiswa yang bertempat tinggal di daerah padat permukiman, seperti perumahan, kos-kosan, dan sebagainya (Tutuko, 2008). Sampah nonpermukiman merupakan sampah yang dihasilkan oleh berbagai tempat umum yang memiliki populasi masyarakat tinggi dengan banyaknya interaksi antar masyarakat, seperti kegiatan dalam perkantoran baik dalam perusahaan maupun restoran, kegiatan komersial pada pertokoan, dan tempat umum lainnya.

Dari permasalahan tersebut, dilakukan pengembangan sistem pengangkutan sampah di daerah Yogyakarta berupa aplikasi berbasis Android yang dapat diakses dengan perangkat *smartphone* sebagai solusi alternatif untuk mengurangi penumpukan sampah pada masyarakat, sebagai sarana optimalisasi pengangkutan sampah dengan memanfaatkan moda transportasi milik penggerak sampah, dan mengoptimalkan upaya kebersihan lingkungan. Sistem pengangkutan sampah dikembangkan berbasis Android dikarenakan ada beberapa fungsi yang tidak bisa digunakan pada platform web, seperti penggunaan kamera, notifikaasi, GPS, di mana pada sistem ini terdapat proses unggah bukti pembayaran melalui media perangkat maupun akses ke kamera dan penggunaan notifikasi, GPS yang akan dikembangkan lebih lanjut serta pengembangan aplikasi dapat memberikan respons untuk menampilkan data lebih cepat. Sistem ini akan dikembangkan dengan dua aplikasi berdasarkan jenis penggunanya. Pengguna dari sistem ini, yaitu masyarakat dan pengepul sampah sebagai admin. Aplikasi khusus bagi pengguna, yaitu masyarakat memiliki peran utama dalam proses pemesanan paket pengangkutan sampah berdasarkan titik lokasi masyarakat dan didukung dengan fitur-fitur

yang lain, seperti *login*, fitur notifikasi, riwayat pesanan dan sampah, pembayaran secara transfer serta profil akun masyarakat sehingga diharapkan dapat memberikan layanan jasa angkut sampah secara efisien. Selanjutnya, aplikasi bagi pengepul sampah yang memiliki peran utama dalam mengelola dan memproses jasa angkut sampah atas permintaan dari masyarakat yang didukung dengan fitur-fitur, seperti *login*, proses konfirmasi pesanan, data pesanan yang terkonfirmasi dan siap untuk dilakukan proses angkut sampah serta data status angkut sampah mengenai informasi hasil pengangkutan sampah yang telah dilakukan. Dengan begitu, sistem pengangkutan sampah akan menggunakan nama Bersih Kotaku. Aplikasi ini dibangun dengan tujuan dapat menjadi salah satu *start-up* yang mampu berkontribusi di tengah masyarakat dan dapat bertahan dengan mempertimbangkan aspek bisnis. Program Bersih Kotaku akan dibangun menggunakan perangkat lunak Android Studio dengan bahasa pemrograman Java untuk masyarakat dan admin sebagai pengepul sampah.

Proses pengembangan sistem pengangkutan sampah merupakan bagian dari jalur perintisan bisnis dengan menghasilkan sebuah *start-up* yang dapat membantu persoalan sampah di tengah masyarakat. Jalur perintisan bisnis terdapat tiga sumber daya manusia yang terlibat dalam membangun sistem pengangkutan sampah, yaitu Muhammad Yusuf Imaddudin sebagai *hustler* yang berperan dalam menganalisis permasalahan dan kebutuhan, aspek bisnis pada sistem yang dikembangkan, dan melakukan perancangan desain sistem. Selain itu, Muhammad Izzul Yaqien sebagai *hipster* yang berperan dalam membangun tampilan sistem berupa desain aplikasi yang terdiri dari rancangan sketsa (*wireframe*), rancangan *mockup* (*prototype*), dan melakukan pengujian hasil rancangan desain aplikasi terhadap pengguna. Yang terakhir sebagai *hacker* yang diperankan oleh Tesar Firstyaji Pramudya yang memiliki tugas untuk melakukan pengembangan sistem berbasis Android berdasarkan perencanaan yang telah dilakukan oleh Yusuf dan Izzul. Pengembangan sistem yang dibangun akan diimplementasikan dalam perangkat lunak Android Studio untuk dilakukan pemrograman yang nantinya akan menghasilkan sebuah aplikasi yang sesungguhnya dan dapat berinteraksi dengan pengguna.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka disusun rumusan masalah yang akan dibahas pada penelitian ini, berupa: Bagaimana mengembangkan sistem aplikasi angkut sampah berbasis Android dengan metode *prototyping* sebagai *start-up* penyedia jasa pengangkutan sampah?

1.3 Tujuan

Adapun tujuan yang hendak dicapai dalam pengerjaan Tugas Akhir ini adalah:

a. Mengembangkan sistem aplikasi angkut sampah berbasis Android sebagai *start-up* penyedia jasa pengangkutan sampah di tengah masyarakat.
b. Memudahkan pengepul sampah dalam mengelola jasa pengangkutan sampah.

1.4 Manfaat

Penyusunan Tugas Akhir ini diharapkan dapat memberikan manfaat kepada:

a. Masyarakat

   Memudahkan masyarakat untuk berlangganan jasa pengangkutan sampah secara efisien sebagai solusi untuk mengurangi terjadinya penumpukan dan pembuangan sampah sembarangan.
b. Pengepul Sampah

   Memudahkan dalam mengelola jasa pengangkutan sampah.
c. Penulis

   Penulis dapat mengembangkan dan menerapkan ilmu yang diperoleh selama kuliah khususnya metode dan teknologi dalam pengembangan perangkat lunak berbasis Android.

1.5 Batasan Masalah

Dalam penyusunan Tugas Akhir ini, penulis akan membatasi cakupan pembahasan agar penyusunan penelitian ini terfokus dengan tujuan adalah sebagai berikut:

a. Menggunakan standar metode *prototyping* dalam membangun aplikasi angkut sampah (Bersih Kotaku).
b. Target pengguna dari aplikasi terkhusus bagi masyarakat Yogyakarta.
c. Pembayaran hanya dapat dilakukan dengan metode transfer bank ke rekening pengepul sampah.
d. Aplikasi yang dikembangkan hanya dapat diakses menggunakan *smartphone* berbasis Android oleh pengguna.
e. Sistem operasi Android yang bisa dijalankan minimal versi 5.0 (Lollipop).

1.6 Metode Penelitian

Metode penelitian sebagai tahapan-tahapan yang dilakukan dalam penyusunan laporan tugas akhir dengan tahapan yang akan dilalui meliputi:

1.6.1 Analisis

Analisis kebutuhan merupakan tahap awal dalam pengembangan sistem dengan melakukan pengumpulan data terhadap spesifikasi kebutuhan fungsional apa saja yang dibutuhkan dalam sebuah sistem yang akan dibangun. Selain itu, menganalisis proses bisnis, seperti apa yang nantinya dilakukan pada proses pengembangan aplikasi agar aplikasi bisa bertahan di tengah masyarakat.

1.6.2 Membangun Desain Sistem dan *Prototype*

Tahapan ini melakukan pembuatan rancangan gambaran dari sistem yang akan dibangun agar memudahkan pengembang dalam pengembangan sistem yang didukung dengan rancangan, seperti hubungan antara pengguna dan sistem, perancangan diagram aktivitas, perancangan basis data, dan perancangan antarmuka aplikasi.

1.6.3 Pengujian dan Evaluasi *Prototype*

Tahapan ini dilakukan untuk pengujian desain aplikasi yang telah dirancang agar pengguna dari aplikasi dapat mengetahui gambaran desain apakah telah sesuai dengan kebutuhan pengguna. Dengan kata lain, jika evaluasi dilakukan apabila desain yang diujikan belum dapat diterima dengan baik oleh pengguna.

1.6.4 Pengembangan Sistem

Tahap pengembangan sistem melakukan pengodean bahasa pemrograman di dalam perangkat lunak sebagai bentuk representasi kode program yang nantinya menghasilkan sebuah sistem yang dapat dijalankan dan digunakan dengan nyata oleh pengguna.

1.6.5 Pengujian Sistem

Tahapan ini merupakan tahap terakhir di mana sistem yang telah menjadi *software* dilakukan pengujian untuk memastikan bahwa sistem yang dikembangkan dari segi fungsionalitas berjalan baik dan menghasilkan luaran yang sesuai dengan kebutuhan.

1.7 Sistematika Penulisan

Sistematika penulisan merupakan gambaran umum dalam penyusunan laporan tugas akhir adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini membahas latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, metode penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini berisi penjelasan teori-teori yang menjadi acuan dan berkaitan tentang kajian informasi sampah dan pengembangan aplikasi.

BAB III METODOLOGI

Bab ini membahas tahapan dalam melakukan pengembangan aplikasi yang memuat analisis, membangun desain sistem dan *prototype*, pengujian dan evaluasi *prototype*, pengembangan sistem, dan pengujian sistem.

BAB IV HASIL DAN PEMBAHASAN

Bab ini menguraikan rincian hasil yang didapatkan dari metode penelitian yang dilakukan. Hasil penelitian dapat diwujudkan berupa pembahasan dari implementasi antarmuka aplikasi, hasil pengujian aplikasi berdasarkan pengembangan aplikasi yang telah dilakukan.

BAB V PENUTUP

Bab ini berisi kesimpulan mengenai pernyataan singkat dan tepat yang diuraikan dari hasil penelitian dan pembahasan yang telah dilakukan. Selain itu, juga terdapat saran yang berisi usulan ataupun solusi berdasarkan pengamatan dari peneliti mengenai hasil dan pembahasan yang telah dilakukan.

BAB II
LANDASAN TEORI

2.1 Pengertian Sampah

Sampah merupakan sebuah barang yang tidak digunakan, tidak terpakai atau sesuatu yang dibuang oleh pemilik sampah, tetapi sampah yang dibuang jika sebagian masyarakat dapat mengelola dengan baik dapat mengurangi volume sampah yang dihasilkan sehingga menekan penumpukan sampah. Berdasarkan UU No. 18 Tahun 2008 tentang Pengelolaan Sampah, disebutkan bahwa sampah merupakan sisa aktivitas manusia atau proses alam yang berbentuk padat yang berupa zat organik (dapat teurai) dan anorganik (sulit terurai). (Dwi et al., n.d.). Sampah-sampah yang dihasilkan oleh manusia berasal dari berbagai tempat yang ditinjau berdasarkan sumbernya terdiri dari sampah permukiman dan nonpermukiman. (Goleman et al., 2019).

2.1.1 Sampah berdasarkan sumbernya

Sampah permukiman

Sampah permukiman merupakan sampah yang menghasilkan volume cukup besar yang diikuti mobilitas masyarakat yang tinggi. Sampah yang dihasilkan merupakan bentuk kegiatan harian dari suatu keluarga yang bertempat tinggal di permukiman desa, perumahan atau kos-kosan. Jenis sampah yang dihasilkan biasanya berupa sampah organik, seperti sampah sisa makanan atau sayuran baik yang telah dimasak atau belum, sampah kering, dan sampah basah.

Sampah nonpermukiman

Sampah nonpermukiman merupakan sampah yang dihasilkan dari tempat umum, seperti restoran, pertokoan, terminal bus, dan lainnya serta dari tempat perkantoran, seperti perkantoran pendidikan, perusahaan, departemen, dan lainnya. Potensi sampah yang dihasilkan nonpermukiman terbilang cukup besar dalam memproduksi sampah dengan banyaknya aktivitas orang yang berkumpul dan melakukan kegiatan. Jenis sampah yang dihasilkan biasanya berupa sampah botol atau kaleng, kertas, plastik, dan umumnya sampah bersifat anorganik dan mudah terbakar.

2.2 Angkutan Sampah

Angkutan merupakan sarana dan prasarana kendaraan yang digunakan untuk mengangkut sampah dari lokasi sumbernya untuk dibawa ke tempat tujuan, seperti tempat pembuangan sementara ataupun tempat pembuangan akhir. Ketersediaan armada angkutan harus selalu ada agar sewaktu-waktu ketika pengangkutan sampah dengan populasi sampah makin tinggi, penggerak pengelola sampah dapat ikut serta dalam mengurangi masalah sampah. (Rakhmad, 2002).

2.3 Android Studio

Android Studio adalah *Integrated Development Environmnet* milik Google yang memuat *tools* untuk melakukan edit bahasa pemrograman, desain, pengujian, dan *debug* serta dapat diakses secara gratis dan umumnya digunakan untuk pengembangan aplikasi berbasis Android dengan *development environment* Android baru yang berdasarkan Intellij IDEA. Kemampuan di dalam platform ini dengan menawarkan fitur berupa *gradle* untuk membangun dukungan atau *library*, terdapat *layout* editor yang memungkinkan untuk *drag and drop* UI komponen, dan *layout* pratinjau untuk konfigurasi layar**,** emulator yang cepat dan kaya fitur serta alat *lint* untuk menangkap kompatibilitas versi, kinerja, dan masalah lainnya. (Anwar, 2019).

2.4 *Android Software Development Kit*

Software Development Kit (SDK) merupakan kit dan *tools* API (*Aplication Programming Interface*) yang dapat digunakan para *developer* untuk mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Android merupakan *subset* perangkat lunak *smartphone* yang meliputi sistem operasi, *middleware*, dan aplikasi kunci yang dirintis oleh Google. Di dalam *SDK* terdapat beberapa *tools* yang dapat membantu kelancaran proses pengembangan aplikasi, seperti *software libraries*, emulator, *debugger*, *sample code*, dan dokumentasi. (Nagib, 2014).

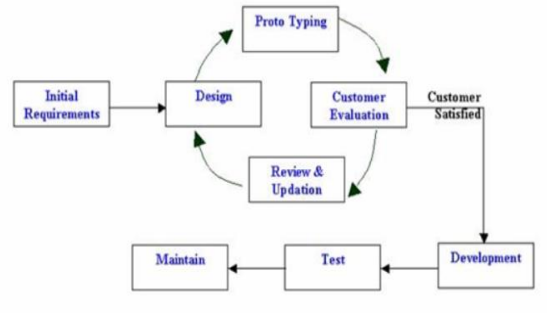
2.5 Java

Java merupakan bahasa pemrograman berorientasi objek yang dikembangkan oleh Sun Microsystem pada tahun 1995. Java salah satu bahasa yang sering digunakan dan populer karena perangkat lunak yang dikembangkan menggunakan bahasa ini sangat luas. Bahasa pemrograman Java bersifat *write once, run anywhere* dalam artian program yang ditulis satu kali dan dapat berjalan di berbagai jenis sistem operasi. Hal tersebut, bisa terjadi karena Java

memiliki kode pemrograman level tinggi, di mana ketika eksekusi suatu program maka akan di-*compile* dengan *java virtual machine* menjadi *kode numeric* platform. (Ika Purwanti, 2013).

2.6 *Prototyping*

Prototyping adalah metode pengembangan perangkat lunak dengan membuat model rancangan atau *prototype* secara cepat dengan tujuan pengujian konsep yang telah direncanakan. Dengan kata lain, *prototype* merupakan bentuk representasi dari perancangan aplikasi yang akan dibuat. Penggunaan metode ini, dapat menentukan rancangan produk menjadi rancangan yang final karena setiap rancangan yang dibuat akan dilakukan pengujian konsep desain dan pengguna dapat memberikan umpan balik mengenai rancangan yang ada. Dengan begitu, tim pengembang dapat mengetahui kesalahan atau kekurangan hasil rancangan serta apa saja yang menjadi prioritas dari kebutuhan pengguna. Perancangan aplikasi awalnya berbentuk sketsa (*wireframe*) setelah itu, dibangun *prototype* berupa gambar rancangan yang final sesuai dengan kebutuhan pengguna dan nantinya akan dijadikan produk akhir sebagai *output* dari pengembangan aplikasi. Metode *prototyping* memiliki siklus sebagaimana tampak pada Gambar 2.1. Metode tersebut, memiliki tahapan yang harus dilalui untuk membangun sistem di antaranya tahap analisis, membangun desain sistem dan *prototype*, pengujian dan evaluasi *prototype*, pengembangan sistem, dan pengujian produk pada pengguna. Keuntungan dalam menggunakan metode ini, pengguna dan tim pengembang dapat saling berkomunikasi khususnya dalam hal penyetaraan persepsi sewaktu proses pembuatan *prototype system* (Bsi university, 2022). Terkadang, di saat merancang sistem, pengguna hanya menginterpretasikan secara umum apa yang diminta tanpa menyebut prosesnya sehingga dapat menyebabkan perancangan tersebut kurang maksimal dan interaktif. Untuk itu, pengguna harus memahami produk agar dalam menentukan model sistem bersama tim pengembang dapat berjalan sesuai dengan apa yang diharapkan. Selain itu, menjadi pedoman bagi *developer* dengan lebih mudah mengimplementasikan hasil rancangan desain yang telah final ke dalam kode program yang akan dibuat sehingga tidak perlu lagi khawatir mengenai kesalahan pada tahapan yang telah dilalui. (Akif et al., 2015).



Gambar 2.1 Metode *prototyping*

2.7 *Use Case Diagram*

Use case diagram menggambarkan kegiatan apa saja yang dilakukan oleh sistem. Diagram *use case* dekat kaitannya dengan kejadian dari interaksi antara pengguna sistem dengan sistem yang akan dikembangkan. (Paramitha, 2020).

2.8 *Activity Diagram*

Activity diagram merupakan aktivitas yang menggambarkan *workflow* dari fungsionalitas pada sebuah sistem yang diawali dengan mulainya aksi, di mana berhentinya, aksi apa yang terjadi selama *workflow*, dan bagaimana urutan kejadian aksi tersebut. Tujuan melakukan pemodelan *activity diagram* sebagai tingkah laku dinamis dari sistem dengan cara menguraikan *workflow* pesan dari aktivitas aksi ke aktivitas lain. (Apriliah et al., 2021).

2.9 *Firebase Firestore Database*

Firebase Firestore Database merupakan layanan basis data NoSQL yang di-*hosting* di-*cloud* yang dapat digunakan untuk menyimpan, menyinkronkan, dan membuat kueri data pada perangkat lunak. Data akan tersimpan dalam dokumen, yang sangat mirip dengan JSON. *Cloud firestore* memiliki kemampuan dalam mengoperasikan kueri yang lebih cepat, kompleks, dan skala atau data yang lebih besar daripada *realtime database*. *Cloud firestore* mendukung adanya sinkronisasi data secara *offline,* di mana akan menyimpan *cache* data dari perangkat apa pun, seperti Android, iOS, dan web sehingga pengguna ketika mengakses data mereka pada aplikasi tertentu dan terjadi perubahan data pada *cache* maka saat perangkat kembali *online* basis data *firestore* melakukan sinkronisasi setiap perubahan lokal ke dalam *firestore*. Dengan hal itu, dapat diimplementasikan untuk mengembangkan aplikasi yang responsif, *powerfull*, dan mampu bekerja tanpa bergantung pada latensi jaringan. (James Tamplin, n.d.-b).

2.10 *Cloud Storage for Firebase*

Cloud storage for firebase merupakan layanan yang tersedia di-*firebase* yang digunakan sebagai media penyimpanan dan menampilkan konten buatan *user*, seperti foto, *file* dokumen, audio ataupun video. *Cloud storage* mampu melakukan proses unggah dan unduh dalam kondisi jaringan yang lemah dan stabil. Konten yang diunggah atau diunduh dapat dimulai kembali di saat proses unggah atau unduh berhenti dan tidak diulang dari awal sehingga dapat menghemat *bandwidth* dan waktu pengguna. *Cloud storage* juga memiliki keterikatan dengan *firebase authentication* dalam hal keamanan data yang kuat dan mudah sehingga *developer* dapat mengimplementasikan model keamanan yang deklaratif berdasarkan nama *file*, jenis konten, ukuran, dan metadata lainnya. (Prasetyo et al., n.d.).

2.11 *Firebase Authentication*

Firebase Authentication salah satu fitur *firebase* yang menyediakan layanan autentikasi dengan mudah dan menariknya *firebase* menyiapkan *library* UI siap pakai yang fleksibel sehingga proses implementasi autentikasi akan lebih efisien. Dengan *firebaseUI* menyediakan alternatif autentikasi *drop-in* yang menangani proses UI untuk *user* yang *login* dengan *email* dan kata sandi atau menggunakan integrasi identitas gabungan, seperti akun Google, Facebook, Twitter ataupun Github. Dengan memanfaatkan komponen *firebaseUI Auth* dapat mengoptimalkan konversi masuk dan *register* oleh pengguna pada aplikasi. *Firebase authentication* terhubung dengan fitur *firebase* lainnya dengan memanfaatkan jenis standar industri, seperti *OpenID Connect* dan *OAuth 2.0* yang memudahkan integrasi dengan *backend* kustom oleh *developer* itu sendiri. (James Tamplin, n.d.-a).

2.12 *BlackBox Testing*

BlackBox Testing merupakan salah satu metode yang digunakan untuk pengujian yang berfokus pada kebutuhan fungsionalitas terhadap perangkat lunak yang dirancang. Pengujian *blackbox* akan dilakukan pada proses input dan *output* yang dikembangkan. Penguji akan mencoba program dengan memasukkan salah satu data pada halaman tertentu, setelah itu akan diamati secara rinci apa yang terjadi pada luaran dari program tersebut. Metode ini tentu sangat diperlukan bagi *developer* karena hal ini akan menjadi kunci keberhasilan dalam mengembangkan suatu perangkat lunak. Maka dari itu, berbagai fungsi yang diujikan di dalam proses input harus menghasilkan *output* sesuai kondisi input yang diberikan sehingga

memperoleh luaran dengan apa yang diharapkan dan perangkat lunak tersebut *user friendly*. (Shadiq et al., 2021).

BAB III
METODOLOGI

Membangun sebuah perangkat lunak tentu memerlukan sebuah metodologi agar proses bisnis yang sudah direncanakan dapat berjalan dengan tepat dan membantu segala rangkaian pengembangan sistem sesuai dengan kebutuhan pengguna. Berkaitan dengan proses pengembangan sistem berbasis Android, metodologi yang digunakan pada penelitian ini, yaitu metode *prototyping*. Dipilihnya metode ini dikarenakan *prototyping* dapat dilakukan secara iteratif dalam artian setiap pekerjaan ketika berada ditahapan *prototype* dengan membuat model rancangan pertama kalinya akan berfokus pada pengujian konsep desain, apabila model rancangan belum sesuai yang diharapkan pengguna maka secara terus-menerus dilakukan pengujian sampai konsep desain secara keseluruhan dapat diterima dengan baik dan puas bagi pengguna. Dalam proses metode ini, pengguna dapat ikut serta dalam proses pengembangan produk dengan cara mengevaluasi *interface* dan memberikan *feedback* sehingga tim pengembang lebih mudah mengetahui kebutuhan produk yang diharapkan pengguna. Maka dari itu, umpan balik yang diberikan dapat dijadikan sebagai acuan pengembangan produk secara final. Proses pengembangan sistem terdapat tiga peran orang yang terlibat dalam merintis *start-up* aplikasi Bersih Kotaku di antaranya *hustler* yang menjadi tugas dari Muhammad Yusuf Imaduddin dengan melakukan *task* antara lain melakukan proses analisis kebutuhan, analisis proses bisnis, dan proses perancangan desain sistem. Selanjutnya, *hipster* yang menjadi tugas dari Muhammad Izzul Yaqien dengan melakukan *task* antara lain merancang *prototype* yang terdiri dari *interface low fidelity* dan *interface high fidelity* serta melakukan pengujian dan evaluasi *prototype* sampai rancangan yang dikembangkan dapat diterima baik oleh pengguna. Selain itu, *hacker* yang menjadi tugas dari Tesar Firstyaji Pramudya dengan melakukan *task* antara lain mengimplementasikan segala kebutuhan sistem dan hasil *prototype* yang final untuk dilanjutkan proses pengembangan sistem berupa pembuatan kode program dan pengujian pada fungsionalitas sistem. Berikut ini merupakan pemaparan dari tahapan metode penelitian dalam pengembangan sistem.

3.1 Tahap Analisis

3.1.1 Analisis Kebutuhan

Tahap analisis kebutuhan dilakukan dengan tujuan identifikasi masalah dan pengumpulan kebutuhan fungsional dalam sebuah sistem. Kebutuhan fungsional mengenai hal-

hal yang dibutuhkan sistem dengan mendefinisikan kebutuhan input, kebutuhan *process*, dan kebutuhan *output*. Penelitian yang dilakukan dalam pengembangan produk, yaitu berupa aplikasi pengangkutan sampah di daerah Yogyakarta yang merupakan layanan jasa angkut sampah yang menghubungkan antara masyarakat dengan pengepul sampah yang dapat diakses menggunakan *smartphone* berbasis Android. Aplikasi ini memiliki fitur utama yang berupa paket sampah bulanan yang dapat diproses masyarakat dengan melakukan permintaan jemput sampah terhadap pengepul sampah berdasarkan titik lokasi yang dimasukkan oleh masyarakat. Sistem aplikasi yang akan dibangun hendak disesuaikan dengan jenis penggunanya karena setiap pengguna mempunyai kebutuhan fungsional yang berbeda. Oleh sebab itu, aplikasi yang dibangun akan dipisahkan berdasarkan jenis penggunanya. Dengan dilakukannya analisis kebutuhan fungsional, dapat menggambarkan layanan-layanan yang bisa diberikan sistem kepada pengguna secara rinci. Kebutuhan fungsional akan diimplementasikan setiap pengguna dari aplikasi yang terdiri dari masyarakat dan pengepul sampah (admin). Masing-masing pengguna tersebut memiliki peran dalam aplikasi yang berbeda sebagaimana yang tampak pada Tabel 3.1.

Kebutuhan masukkan

Dalam mengembangkan aplikasi diperlukan beberapa masukkan berdasarkan penggunanya, yaitu sebagai berikut:

a. Bagi Masyarakat
    1. Data *email* yang terintegrasi dengan akun Google.
    2. Paket yang akan dipesan.
    3. Data alamat yang diisikan sebagai titik jemput sampah.
    4. Data bukti pembayaran ke sistem.
    5. Edit data identitas akun.
    6. Data unggah gambar profil.
b. Bagi Pengepul Sampah
    1. Data rekening sebagai informasi pembayaran.
    2. Menerima atau menolak data pesanan.
    3. Memberikan informasi status angkut sampah selama proses pengangkutan sampah berlangsung.

Kebutuhan proses

Kebutuhan proses yang dilakukan pengguna ketika melakukan aksi input, yaitu sebagai berikut:

a. Bagi Masyarakat
   1. Sistem memproses autentikasi berupa *register* dan *login*.
   2. Sistem dapat memproses pemesanan paket sampah.
   3. Sistem dapat memproses dan menyimpan data alamat.
   4. Sistem dapat memproses dan mengunggah bukti data pembayaran.
   5. Sistem dapat memproses dan menyimpan edit data identitas akun.
   6. Sistem dapat memproses dan mengunggah foto profil.

b. Bagi Pengepul Sampah
   1. Sistem memproses konfirmasi pesanan paket sampah dari masyarakat.
   2. Sistem memproses informasi status pengambilan sampah.

Kebutuhan keluaran

Kebutuhan keluaran yang terjadi setelah sistem melakukan aksi pada fungsi yang ada, yaitu sebagai berikut:

a. Bagi Masyarakat
   1. Sistem menampilkan data paket sampah yang berhasil dipesan dan aktif pada halaman beranda dan riwayat pesanan.
   2. Sistem menampilkan data status angkut sampah pada riwayat sampah.
   3. Sistem menampilkan data identitas akun.
   4. Sistem menampilkan informasi data pesan notifikasi mengenai hasil kegiatan masyarakat dalam sebuah sistem.

b. Bagi Pengepul Sampah
   1. Sistem menampilkan data permintaan berlangganan jasa angkut sampah.
   2. Sistem menampilkan informasi daftar masyarakat yang berlangganan jasa angkut sampah berupa paket sampah yang terkonfirmasi untuk dilakukan pengangkutan.
   3. Sistem menampilkan data status angkut sampah pada halaman status pesanan.

Tabel 3.1 Pengguna aplikasi Bersih Kotaku

Aktor	Deskripsi
Masyarakat	• Peran masyarakat dapat melakukan langganan jasa angkut sampah berdasarkan input titik lokasi. • Melihat data pesanan pada riwayat pesanan, dan status angkut sampah pada riwayat sampah. • Melakukan transaksi secara *online*.
Pengepul sampah	• Melakukan konfirmasi layanan pada masyarakat berupa data pesanan paket sampah. • Melihat daftar masyarakat yang telah berlangganan paket sampah. • Memberikan informasi status sampah dari hasil selama pengangkutan sampah berlangsung.

3.1.2 Analisis Proses Bisnis

Analisis proses bisnis dilakukan untuk memastikan bahwa *business plan* yang telah direncanakan dan diproyeksikan dalam rencana bisnis, yaitu aplikasi pengangkutan sampah yang akan dikembangkan dapat menjadi salah satu *start-up* dan bertahan di tengah masyarakat. Sebuah bisnis yang dikembangkan tentu melewati tahap ini karena bisnis yang matang akan dilakukan analisis dan perancangan sebagaimana bisnis itu dapat menghasilkan nilai (*value*) untuk mendapatkan luaran (*output*) berupa segala kebutuhan yang belum disediakan atau kebutuhan dengan kustomisasi yang baru serta bisa mengisi pasar tertentu.

Proses bisnis di dalam sistem yang akan dikembangkan merupakan bentuk dari permasalahan yang terjadi pada masyarakat dan pengepul sampah. Di sisi masyarakatnya sendiri masih sering mengalami keluhan apalagi yang berada di daerah padat permukiman ataupun nonpermukiman di mana sampah yang dibuang masih sering terjadi keterlambatan pengambilan akibat jadwal sampah yang akan diangkut tergolong tidak teratur atau menentu

setiap minggunya dan adanya pengabaian dalam pengangkutan sampah pada hari pengangkutan sehingga memicu penumpukan sampah yang diikuti aktivitas masyarakat makin tinggi. Di sisi pengepul sampah**,** masyarakat sering terjadi keterlambatan membayar dan terdapat masyarakat yang tidak membayar terhadap jasa pengambilan sampah. Masalah seperti ini sudah tidak asing lagi bagi setiap daerah dan harus diminimalisasi mungkin dengan sosialisasi ataupun dengan berkembangnya ilmu dan pengetahuan teknologi bisa menjadikan solusi di era sekarang dengan produk digital yang makin berkembang.

Produk digital dapat diartikan sebagai produk yang direalisasikan dalam bentuk aplikasi maupun web. Setiap produk yang berjalan di suatu platform khususnya berbasis aplikasi tentu memiliki sebuah konsep yang dirancang dengan tujuan agar konsep tersebut dapat dijadikan solusi bagi masalah yang ada di kehidupan sehari-hari. Untuk itu, mengenai produk digital yang merujuk pada masalah sampah sudah terealisasi di beberapa aplikasi maupun web, seperti Aplikasi Juru Sampah dan Web Resik *Plus*.

Munculnya beberapa aplikasi atau web mengenai sampah sekaligus sebagai kompetitor dalam melakukan pengembangan bisnis. Keberadaan kompetitor di sini bertujuan untuk memotivasi para pebisnis ketika ingin merintis sebuah produk harus berkualitas dan memberikan fitur-fitur yang kompleks (ada ciri khasnya sendiri tiap produk) dari kompetitor yang ada sehingga masyarakat lebih tertarik dengan bisnis yang akan dikembangkan. Setiap kompetitor di atas tentu memiliki konsep yang berbeda-beda di antaranya:

1. Aplikasi Juru Sampah menyediakan layanan untuk pengguna sebagai konsumen dan sebagai *driver* atau tukang angkut sampah. Aplikasi ini memiliki kelebihan adanya fitur penjualan sampah, di mana masyarakat akan menjual sampahnya pada aplikasi dan akan mendapatkan penghasilan dari penjualan sampah tersebut. Untuk kekurangan di aplikasi ini terlalu banyak fitur yang rumit dan di salah satu fitur, yaitu angkut sampah yang tidak terdapat harga angkut sampah yang tertera sehingga menyulitkan konsumen untuk menggunakan aplikasi juru sampah.
2. Resik *Plus* sebagai kompetitor yang merupakan layanan jasa angkut sampah di Yogyakarta berbasis web. Layanan angkut sampah yang ditawarkan hanya terdapat dua paket, yaitu paket sampah campur dan sampah terpilah. Paket pertama yang ditawarkan seharga 75.000,00 untuk pengambilan sampah tidak terpilah atau campur dengan 500 pembeli dan paket kedua yang ditawarkan seharga 50.000,00 untuk pengambilan sampah terpilah dengan 500 pembeli.

Dengan demikian, bisnis yang akan dikembangkan harus memiliki sisi perbedaan dari para kompetitor di atas baik dari konsep ataupun fitur yang ada agar bisnis tersebut dapat menarik perhatian masyarakat. Jasa yang ditawarkan berupa layanan pengangkutan sampah. Layanan ini bertujuan untuk mengurangi penumpukan sampah yang ada di lingkungan dan dapat memanfaatkan moda transportasi angkut sampah dari para pengepul sampah. Bisnis ini akan dikembangkan berbasis Android yang bernama aplikasi Bersih Kotaku dan dapat diakses melalui *smartphone.* Aplikasi Bersih Kotaku memiliki kelebihan dari para kompetitor, yaitu dari kompetitor Juru Sampah, aplikasi Bersih Kotaku memiliki daftar harga di setiap paket pesan jasa angkut sampah dan setiap item jenis paket sampah tertera jadwal pengambilan sampah serta paket dengan harga yang terjangkau. Selain itu, dapat memberikan informasi berupa hasil status angkut sampah selama proses pengangkutan telah dilakukan. Selanjutnya kompetitor kedua, yaitu Resik *Plus*, pada aplikasi Bersih Kotaku terdapat paket yang dapat dipesan siapa saja dan tidak memiliki batasan untuk pembeli atau orang yang berlangganan jasa angkut sampah. Memiliki item paket sampah yang beragam, seperti sampah yang dapat diambil setiap hari dan sampah yang dapat diambil untuk beberapa kali dalam seminggu. Pada aplikasi Bersih Kotaku dapat memudahkan pengguna dengan adanya sistem yang *simplify* dan *user friendly*.

Sistem Bersih Kotaku memiliki fitur-fitur pada aplikasi berdasarkan jenis penggunanya antara lain:

Aplikasi untuk masyarakat yang terdiri:

1. *Login.*
2. Pemesanan paket sampah.
3. Pembayaran secara *online.*
4. Terdapat notifikasi.
5. Terdapat data riwayat pesanan paket dan sampah.
6. Profil akun pengguna.

Aplikasi untuk pengepul sampah yang terdiri:

1. *Login.*
2. Konfirmasi pesanan.
3. Data pesanan paket sampah.
4. Data status angkut sampah.

Akan terdapat dua aktor yang menggunakan aplikasi tersebut, yaitu masyarakat dan pengepul sampah. Untuk *user* (masyarakat) secara langsung bisa *login* menggunakan akun *email* yang terintegrasi dengan Google. Setelah itu, masyarakat akan mendapatkan hak operasional sistem ini untuk memesan jasa pengangkutan sampah dengan berbagai jenis paket sampah bulanan yang ditawarkan oleh sistem. Apabila telah memilih paket bulanan sampah, masyarakat dapat melakukan transaksi agar jasa yang dipesan bisa di konfirmasi oleh pengepul sampah dan pengepul sampah dapat melakukan pengangkutan sampah. Selain itu, untuk *user* (pengepul sampah) dapat *login*, di mana proses *login* secara otomatis langsung masuk ke tampilan utama karena telah diatur menggunakan satu akun yang sama, yaitu sebagai admin. Setelah *login*, pengepul sampah dapat memproses konfirmasi pesanan atas permintaan dari masyarakat, melihat daftar pelanggan yang telah berlangganan paket sampah, dan yang terakhir adalah pengepul sampah berperan dalam memberikan informasi status pengambilan sampah. Dengan proses tersebut, setiap masyarakat yang menggunakan jasa ini akan mendapatkan tagihan pembayaran sesuai paket yang telah dipesan dan pihak pengepul sampah akan menerima pembayaran dari masyarakat. Sehubungan hal itu, pendapatan dari aplikasi akan menerima keuntungan dengan *subscribe* di mana pengepul sampah akan menyewa layanan aplikasi admin yang dikembangkan.

3.2 Tahap Membangun Desain Sistem dan *Prototype*

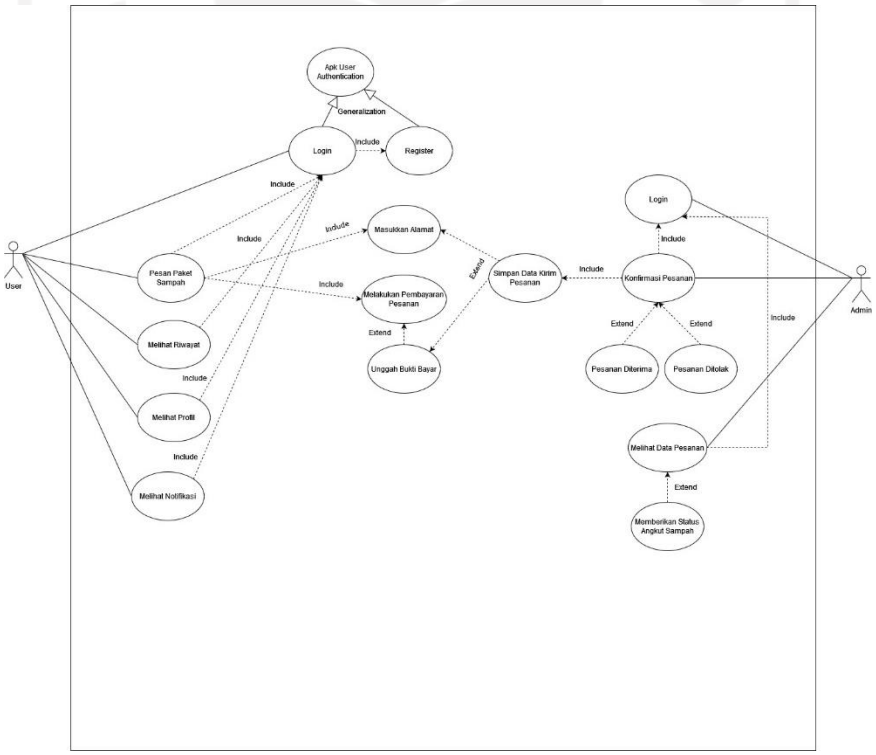
Tahap ini merupakan dua tahap yang saling bergantungan dengan tujuan menggambarkan bagaimana sistem yang dirancang dapat memenuhi kebutuhan pengguna. Bagian dari aktivitas dalam desain sistem dapat dipandang sebagai desain antarmuka (*interface*), alur proses sistem, dan struktur basis data yang menghasilkan spesifikasi fungsional yang sesuai dengan produk yang akan dikembangkan. Dalam desain sistem terdapat beberapa aktivitas yang akan dilakukan antara lain membuat *use case*, perancangan *activity diagram*, dan perancangan basis data. Proses perancangan desain sistem dilakukan oleh *hustler* dengan berdiskusi bersama *hipster* maupun *hacker* untuk memberikan masukkan agar perancangan desain sistem dapat sesuai dengan rencana dari pengembangan aplikasi Bersih Kotaku. Di samping itu, perancangan *prototype* dikerjakan oleh *hipster* yang menjadi tugas dalam pembuatan desain tampilan aplikasi.

Setelah desain sistem terpenuhi, proses selanjutnya akan dilakukan pembangunan *prototype* pada sistem sesuai dengan kebutuhan. Tujuan membangun *prototype*, yaitu untuk membuat tampilan interaksi antara pengguna dengan sistem yang seefisien mungkin dan

menarik yang berfokus pada penyajian kepada pengguna dan memberikan arus balik yang tepat kepada pengguna. Aktivitas dari proses *prototype* meliputi beberapa hal, seperti perancangan desain *interface low fidelity* berupa *wireframe* dan desain *interface high fidelity* berupa *mockup.* Perancangan desain antarmuka berupa sketsa yang sederhana (*wireframe*) dilakukan untuk memberikan gambaran singkat tentang sistem yang ingin dibuat. Setelah sketsa berhasil dibangun, akan dilakukan pembangunan *prototype* berupa *mockup* secara mendetail.

3.2.1 *Use case Diagram*

Diagram *use case* merepresentasikan hubungan antara aktor dengan sistem yang dirancang dengan tujuan untuk memudahkan *user* dalam membaca informasi yang diberikan. *Use case* terdapat fungsi apa saja yang ada di dalam sistem dan siapa saja yang berhak menggunakan fungsi tersebut. Aplikasi atau sistem yang akan dikembangkan merepresentasikan hubungan antara aktor sistem, yaitu masyarakat sebagai *end-user* dan admin sebagai pengepul sampah yang di dalamnya terdapat fungsi-fungsi yang akan dilakukan.



Gambar 3.1 *Use case diagram*

Gambar 3.1 menunjukkan hasil *use case diagram* yang telah dirancang. Terdapat enam fungsionalitas untuk *user* sebagai masyarakat, yaitu *login* menggunakan akun *email* Google, *register*, melakukan pemesanan jasa angkut sampah berupa paket sampah, melihat riwayat, melihat profil, dan notifikasi. Selain itu, untuk pihak admin sebagai pengepul sampah memiliki tiga fungsionalitas, yaitu *login* yang menggunakan satu akun yang telah diatur sehingga otomatis langsung masuk ke halaman utama, melihat data pesanan *user*, dan melakukan konfirmasi pesanan atas permintaan jasa dari *user*. Dengan begitu, terdapat sembilan fungsionalitas pada sistem yang akan dikembangkan.

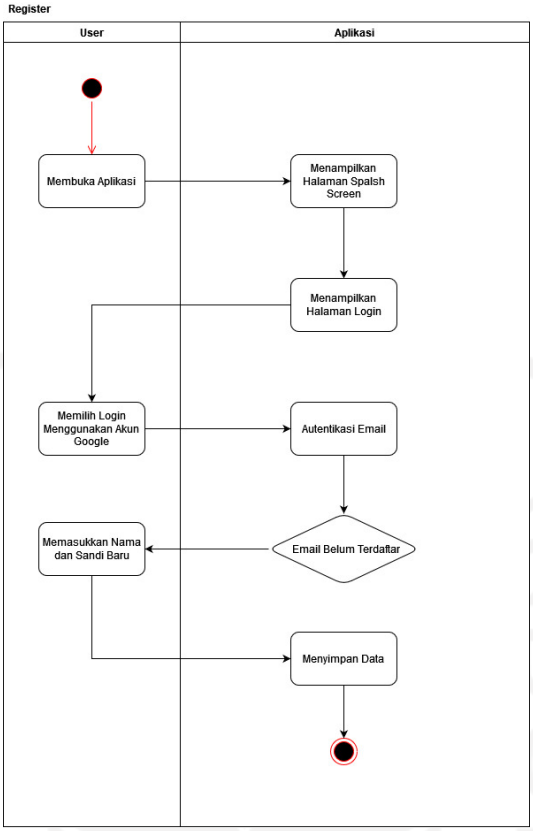
3.2.2 *Activity Diagram*

Activity diagram merepresentasikan *workflow* (aliran kerja) dari fungsionalitas sebuah sistem atau yang terdapat pada *use case* yang dibangun dengan tujuan untuk menjelaskan urutan aktivitas dalam sebuah fungsionalitas. Berikut diagram aktivitas yang digunakan dalam pengembangan aplikasi ini:

Activity Diagram Untuk *End-User*

a. Activity Diagram Register

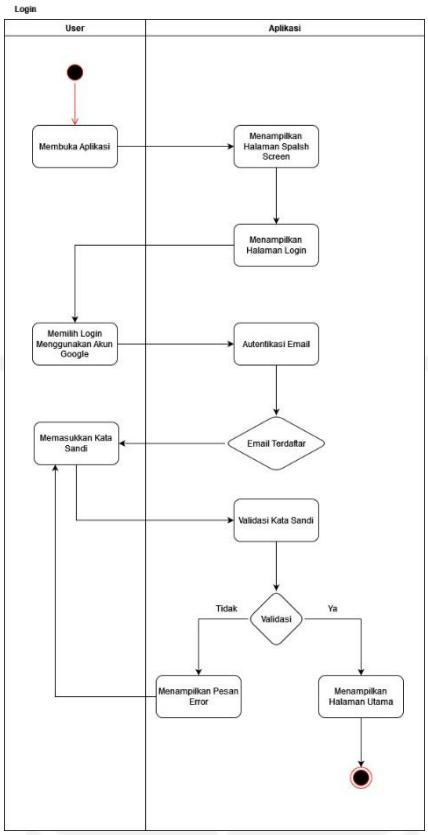
Untuk melakukan akses ke sistem yang dikembangkan terdapat aktivitas yang harus dilakukan, yaitu *register*. Aktivitas ini sebagai proses autentikasi data yang dapat dilakukan ketika masyarakat melakukan proses masuk dengan akun *email* yang terhubung dengan Google. Jika masyarakat telah memilih *email* yang akan digunakan untuk masuk pada aplikasi, sistem akan mengautentikasi *email* tersebut untuk mengetahui apakah *email* telah terdaftar atau belum. Jika hasil proses autentikasi *email* belum terdaftar, sistem mengarahkan ke bagian *register* dari tampilan *default* bawaan fitur autentikasi dan masyarakat diminta untuk memasukkan nama dan kata sandi baru kemudian sistem akan menyimpan data tersebut. Setelah itu, sistem mengarahkan ke bagian *login* dari tampilan *default* bawaan fitur autentikasi saat masyarakat memilih *button* simpan pada proses daftar. Gambar 3.2 menunjukkan aliran kerja dari aktivitas pada proses *register* untuk masyarakat.



Gambar 3.2 *Activity diagram register*

b. Activity Diagram Login

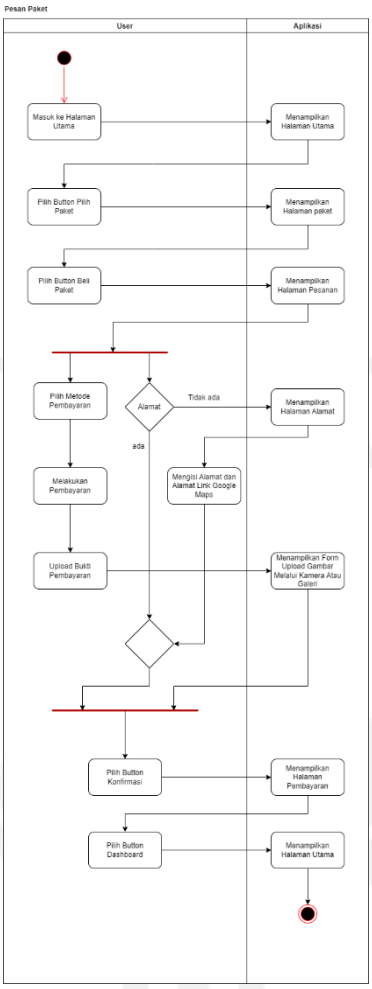
Dalam melakukan akses ke sistem yang dikembangkan terdapat aktivitas yang harus dilakukan, yaitu *login.* Ketika masyarakat membuka aplikasi akan ditampilkan halaman *splash screen* dan seketika itu langsung masuk ke halaman *login.* Untuk melakukan *login,* masyarakat dapat menggunakan akun *email* yang terhubung dengan Google, kemudian dilakukan autentikasi *email*. Jika saat autentikasi *email*, *email* masyarakat telah terdaftar di sistem maka masyarakat diminta untuk memasukkan data berupa kata sandi. Selanjutnya, sistem akan memvalidasi kata sandi, apabila kata sandi yang dimasukkan benar langsung diarahkan menuju tampilan halaman utama aplikasi. Jika kata sandi yang dimasukkan salah, akan menampilkan pesan eror dan masyarakat diarahkan untuk mengisi kata sandi lagi dengan kata sandi yang sesuai saat proses *register*. Gambar 3.3 menunjukkan aliran kerja dari aktivitas pada proses *login* untuk masyarakat.



Gambar 3.3 *Activity diagram login*

c. *Activity Diagram* Pesan Paket Sampah

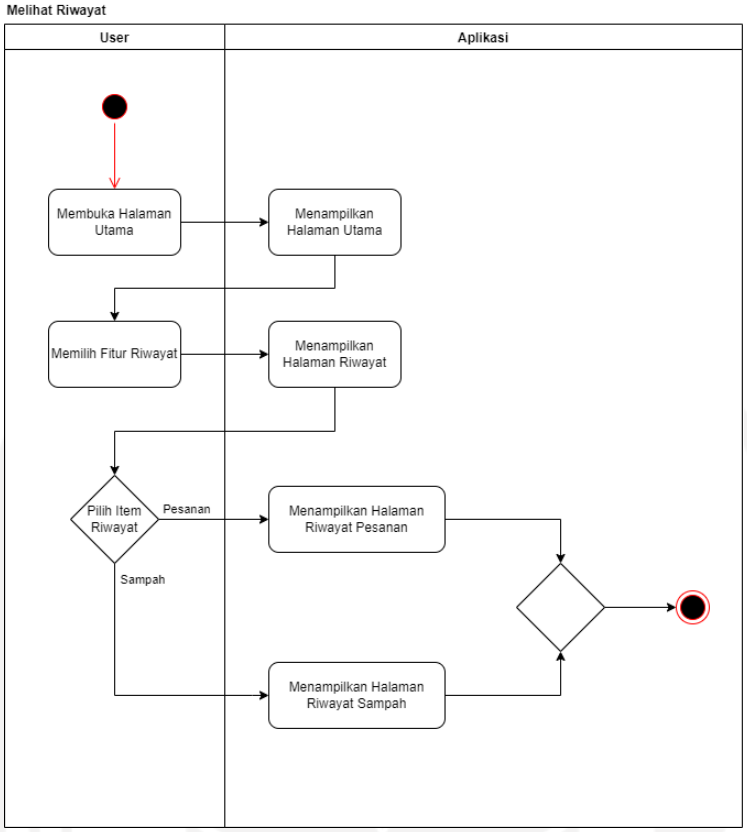
Pada halaman utama aplikasi terdapat item pilih paket sampah yang digunakan untuk proses pembelian paket sampah pada halaman paket. Di dalam halaman paket tersedia varian paket sampah dan masyarakat dapat beli paket sesuai dengan keinginan. Setelah pilih beli paket, masyarakat akan diarahkan ke halaman pesanan yang menampilkan rincian paket yang dipesan, data alamat sebagai titik pengangkutan sampah, dan data bukti pembayaran. Pada data alamat, apabila alamatnya belum terisi masyarakat dapat menambahkan alamat dengan memilih *icon* tambah sehingga akan ditampilkan halaman tambah alamat dengan melakukan isi data alamat berupa alamat dan alamat *link* Google Maps. Di sisi lain, jika masyarakat telah melakukan pembayaran dengan segera untuk meng-*upload* bukti pembayaran agar segala data pesanan dapat terisi semua sehingga masyarakat dapat melakukan kirim pesanan dan menunggu konfirmasi pesanan dari pengepul sampah. Gambar 3.4 menunjukkan aliran kerja dari aktivitas pada proses pesan paket sampah.



Gambar 3.4 *Activity diagram* pesan paket

c. *Activity Diagram* Melihat Riwayat

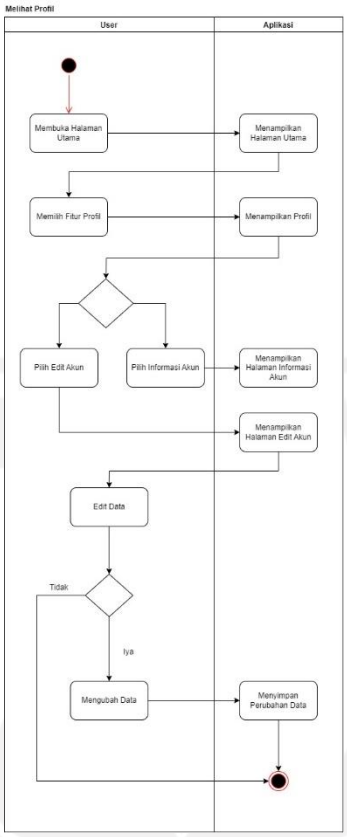
Untuk melihat segala riwayat, masyarakat dapat memilih fitur riwayat yang terdapat pada halaman utama. Saat memilih navigasi riwayat, sistem akan menampilkan halaman riwayat. Di dalam antarmuka riwayat terdapat dua item informasi riwayat berupa pesanan dan sampah. Jika masyarakat memilih pesanan, masyarakat akan diarahkan ke tampilan *list* riwayat pesanan yang berisi informasi paket yang telah dipesan. Sementara itu, jika masyarakat memilih sampah diarahkan ke tampilan *list* riwayat sampah yang berisi informasi mengenai aktivitas angkut sampah. Gambar 3.5 menunjukkan aliran kerja dari aktivitas pada proses melihat riwayat.



Gambar 3.5 *Activity diagram* melihat riwayat

d. *Activity Diagram* Melihat Profil

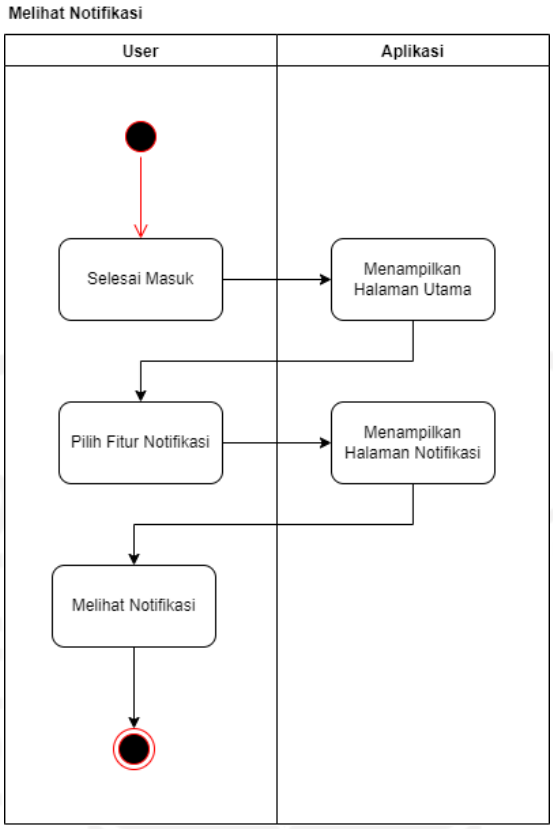
Untuk melihat profil, masyarakat dapat memilih fitur profil yang terdapat pada halaman utama dengan memilih *bottom navigation* akun sehingga sistem akan menampilkan halaman profil. Jika ingin mengetahui data diri, masyarakat dapat memilih item informasi akun yang akan menuju ke halaman informasi akun. Selain itu, masyarakat dapat memilih item edit akun sehingga sistem menampilkan halaman edit akun, di mana item ini sebagai proses masyarakat jika ingin menambahkan atau mengubah data. Dalam halaman edit akun, masyarakat dapat menambah atau memperbaharui data diri pada *form* yang tersedia untuk data yang ingin ditambah atau diubah. Setelah data diisi, sistem akan melakukan simpan data ke basis data. Gambar 3.6 menunjukkan aliran kerja dari aktivitas pada proses melihat profil.



Gambar 3.6 *Activity diagram* melihat profil

e. *Activity Diagram* Melihat Notifikasi

Masyarakat dapat melihat notifikasi dengan memilih fitur notifikasi pada *icon* notifikasi yang terletak di *header* pada halaman utama aplikasi. Setelah memilih *icon* notifikasi masyarakat akan diarahkan pada halaman notifikasi. Di halaman ini masyarakat dapat mengetahui informasi pesan mengenai aktivitas yang dilakukan masyarakat selama menggunakan aplikasi. Gambar 3.7 menunjukkan aliran kerja dari aktivitas pada proses melihat notifikasi.

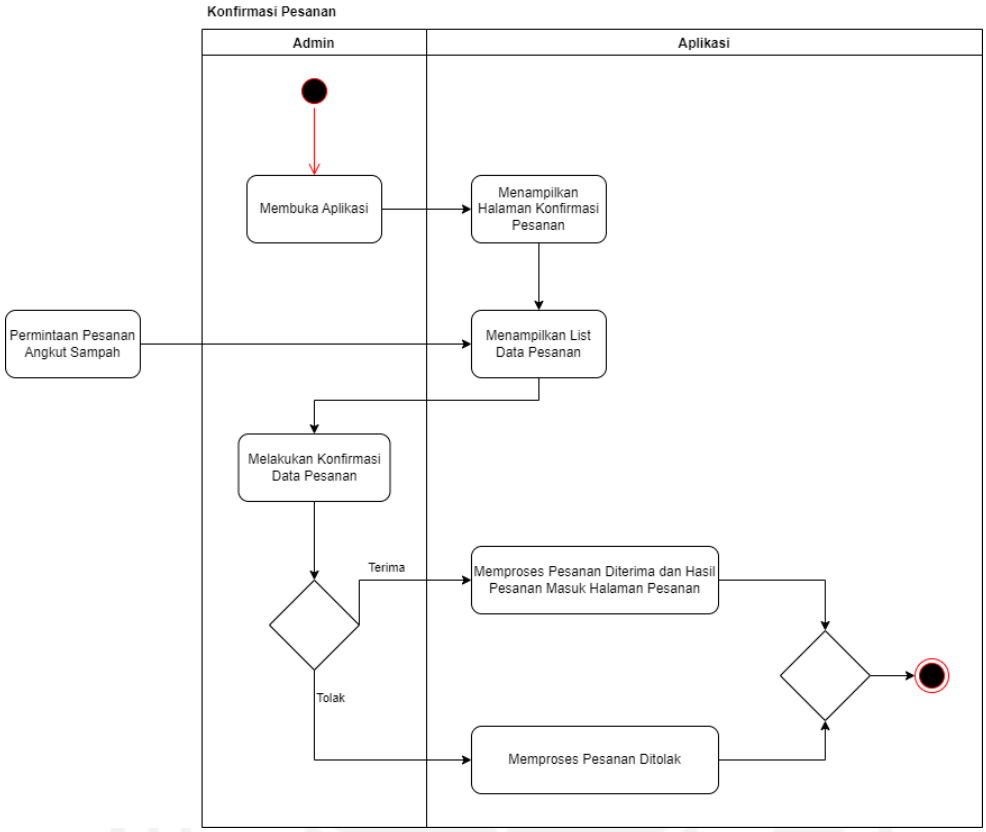


Gambar 3.7 *Activity diagram* melihat notifikasi

Activity Diagram Untuk Pengepul Sampah

a. *Activity Diagram* Konfirmasi Pesanan

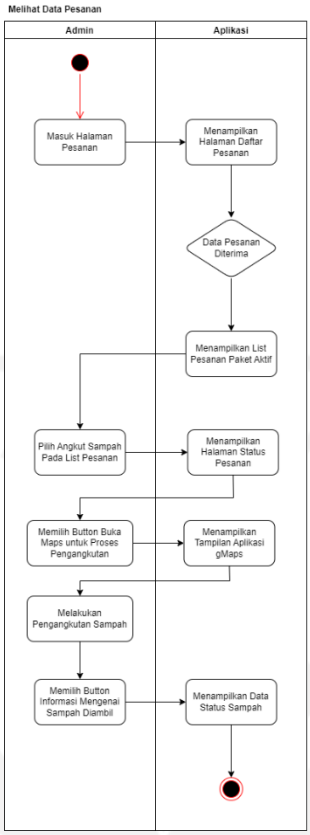
Pada saat pengepul sampah membuka aplikasi, pengepul sampah otomatis telah *login* dengan akun yang telah diatur oleh sistem dan akan langsung diarahkan menuju halaman konfirmasi pesanan. Halaman konfirmasi pesanan pengepul sampah akan menampilkan *list* data pesanan atas permintaan pesanan angkut sampah. Daftar atau *list* data pesanan yang tersedia dari masyarakat akan dilakukan konfirmasi. Jika data permintaan valid, pengepul sampah menyetujui bahwa data pesanan angkut sampah nantinya akan dilakukan. Jika data permintaan tidak valid, pengepul sampah menolak untuk dilakukannya proses angkut sampah. Gambar 3.8 menunjukkan aliran kerja dari aktivitas pada proses konfirmasi pesanan.



Gambar 3.8 *Activity diagram* konfirmasi pesanan

b. *Activity Diagram* Melihat Data Pesanan

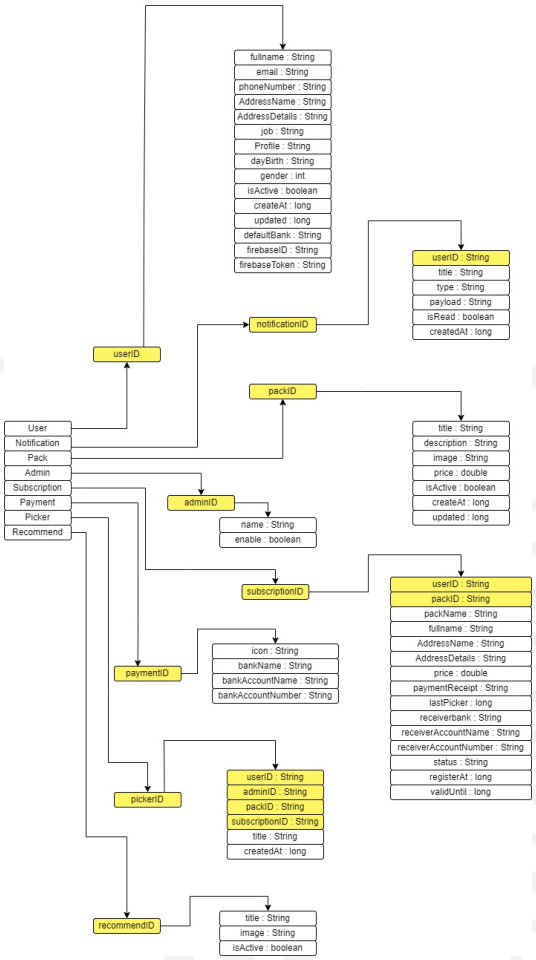
Untuk melihat data pesanan, pengepul sampah dapat masuk ke halaman pesanan lalu akan diarahkan ke tampilan halaman daftar pesanan. Di halaman daftar pesanan, pengepul sampah akan melihat informasi data pesanan dari masyarakat, di mana data pesanan yang ada pada halaman ini merupakan pesanan yang terkonfirmasi dengan status diterima dan menampilkan *list* pesanan paket sampah aktif yang siap untuk dilakukan angkut sampah. Setelah itu, pengepul sampah dapat pilih angkut sampah pada *list* pesanan lalu akan diarahkan ke tampilan halaman status pesanan. Selanjutnya, pengepul sampah dapat memilih *button* buka *maps* untuk mengakses dan melihat rute menuju alamat masyarakat sehingga akan dilakukan pengangkutan sampah. Setelah melakukan pengangkutan sampah, pengepul sampah dapat memberikan informasi mengenai aktivitas angkut sampah dengan memilih *button* mengenai sampah telah diambil. Setelah itu, sistem akan menampilkan informasi berupa data sampah yang telah berhasil dilakukan pengangkutan oleh pengepul sampah dan masyarakat akan mengetahui informasi tersebut. Gambar 3.9 menunjukkan aliran kerja dari aktivitas pada proses melihat data pesanan.



Gambar 3.9 *Activity diagram* melihat data pesanan

3.2.3 Perancangan Basis Data

Basis data sebagai suatu model untuk mendefinisikan hubungan antar data dalam basis data berdasarkan objek-objek data yang mempunyai hubungan antar relasi. Basis data dalam aplikasi layanan jasa pengangkutan sampah menggunakan *Firebase Firestore Database* dengan hubungan antar relasi model data berskema JSON. Dalam perancangan basis data pada aplikasi terdapat delapan objek kelas, yaitu *User*, *Notification*, *Pack*, Admin, *Subscription*, *Payment*, *Picker*, dan *Recommend.* Perancangan basis data dengan skema JSON dapat dilihat pada Gambar 3.10.



Gambar 3.10 Basis data skema JSON

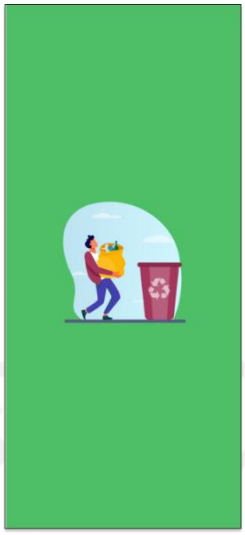
3.2.4 Perancangan *Prototype*

Perancangan *prototype* dilakukan untuk memberikan gambaran desain aplikasi, fitur-fitur apa saja yang akan dikembangkan, dan bagaimana setiap fungsionalitas yang ada dapat bekerja sesuai yang diharapkan. *Prototype* ini diperuntukkan bagi pengguna aplikasi yang terdiri dari masyarakat dan pengepul sampah. Berikut merupakan perancangan *prototype* pada aplikasi yang dikembangkan.

Perancangan Aplikasi Untuk Masyarakat

a. Antarmuka *Splash Screen*

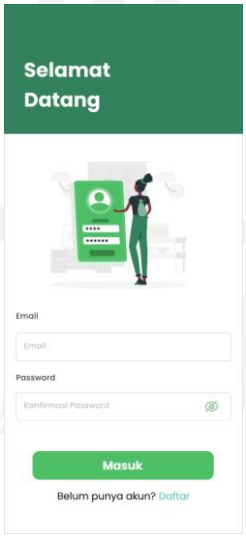
Antarmuka ini akan menampilkan *splash screen* sebagai tampilan awal saat membuka aplikasi. Dalam tampilan ini terdapat item gambar yang mengindikasikan masyarakat melakukan pembuangan sampah dan nantinya akan dilakukan pengangkutan oleh pengepul. Antarmuka ini akan tampak sebagaimana pada Gambar 3.11.



Gambar 3.11 Rancangan antarmuka *splash screen*

b. Antarmuka *Login*

Antarmuka *login* akan tampil setelah proses *splash screen* selesai. Halaman ini terdapat item *email* dan *password* yang digunakan untuk *login* ke dalam antarmuka utama dalam aplikasi. Proses *login* ini hanya berlaku untuk pengguna dari aplikasi, yaitu masyarakat. Masyarakat dapat masuk ke antarmuka utama apabila telah melakukan *register* pada aplikasi. Gambar 3.12 menunjukkan rancangan antarmuka *login*.

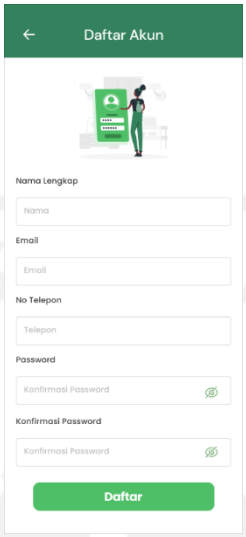


Gambar 3.12 Rancangan antarmuka *login*

c. Antarmuka *Register*

Antarmuka *register* terdapat data yang harus dilengkapi oleh masyarakat ketika belum memiliki akun untuk mengakses sumber daya operasional dari aplikasi Bersih Kotaku. Data

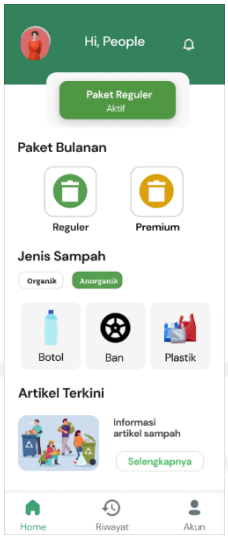
yang diperlukan untuk diisi di antaranya nama lengkap, *email*, nomor telepon, *password* beserta konfirmasi *password* agar kata sandi yang ditambahkan relevan dan valid. Perancangan antarmuka ini sebagaimana tampak pada Gambar 3.13.



Gambar 3.13 Rancangan antarmuka *register*

d. Antarmuka Beranda

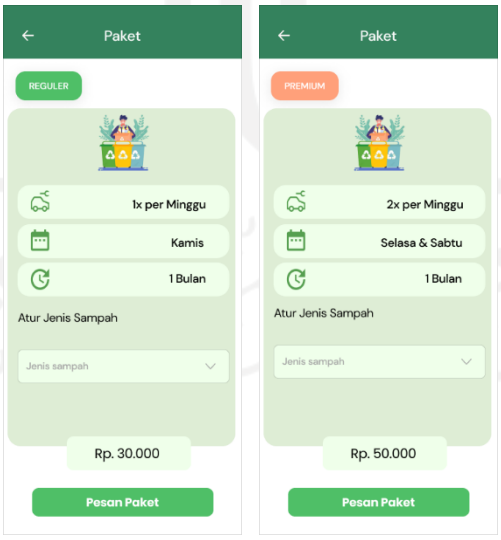
Antarmuka beranda akan menampilkan informasi yang akan diterima masyarakat. Informasi tersebut terdiri dari gambar profil dan nama yang sesuai dengan data saat *login*, notifikasi, dan status paket apabila pengguna sudah berlangganan. Selain itu, ada fitur utama yang berisi paket bulanan sampah meliputi reguler dan premium dan terdapat menu jenis sampah berdasarkan sifatnya serta menu artikel terkini mengenai artikel sampah sebagai informasi edukasi terhadap masyarakat. Selanjutnya, juga ada *bottom navigation* sebagai item yang dapat memudahkan masyarakat dalam mencari informasi secara efisien. Antarmuka beranda akan tampak sebagaimana pada Gambar 3.14.



Gambar 3.14 Rancangan antarmuka beranda

e. Antarmuka Paket

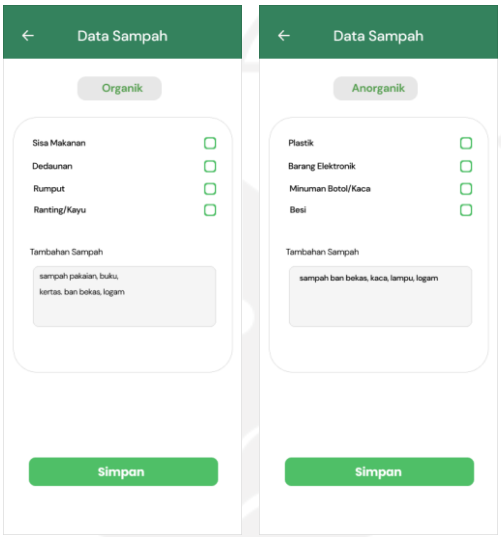
Antarmuka paket akan menampilkan informasi berupa dua item paket bulanan yang bisa dipilih ketika ada pada halaman beranda. Paket ini tersedia di antaranya reguler dan premium. Masing-masing paket memiliki informasi mengenai jadwal pengambilan sampah, periode sampah, dan pengguna dapat mengatur jenis sampah apa yang akan dibuang, seperti sampah organik dan anorganik serta setiap paket mempunyai harga yang berbeda. Gambar 3.15 menunjukkan rancangan antarmuka paket pada masyarakat.



Gambar 3.15 Rancangan antarmuka paket

f. Antarmuka Data Sampah

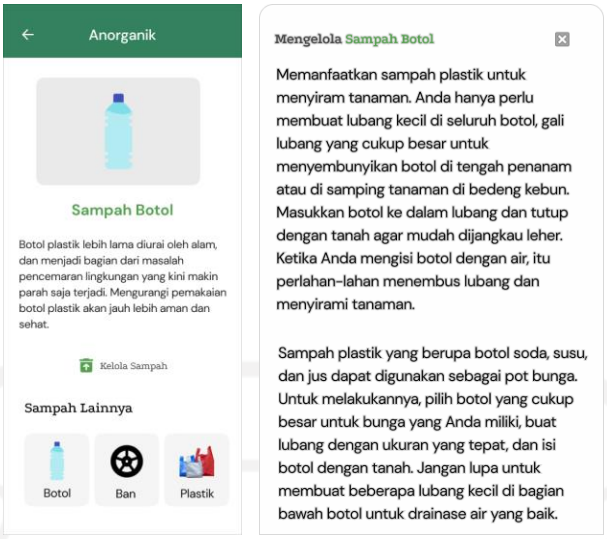
Antarmuka data sampah memiliki dua data yang berbeda dan akan menampilkan data sesuai dengan pemilahan jenis sampah yang diatur antara organik dan anorganik saat berada di antarmuka paket. Antarmuka ini berisi informasi pemilihan sampah apa saja yang akan dibuang, apabila pemilihan sampah yang tersedia kurang lengkap, masyarakat dapat menambahkan sampah yang akan dibuang. Antarmuka data sampah akan tampak sebagaimana pada Gambar 3.16.



Gambar 3.16 Rancangan antarmuka data sampah

g. Antarmuka Jenis Sampah

Antarmuka jenis sampah meliputi dua jenis, yaitu organik dan anorganik. Setiap jenis sampah yang dipilih akan menampilkan informasi berupa varian sampah. Antarmuka ini merupakan jenis sampah anorganik, di mana terdapat informasi sampah, seperti sampah botol beserta penjelasannya. Masyarakat dapat mengetahui informasi sampah lainnya tanpa perlu melakukan *back* ke halaman beranda dan setiap item yang ada dapat di-*scroll* ke kiri sehingga lebih mudah dalam mengakses item lain. Setelah itu, masyarakat dapat mengetahui cara mengelola sampah botol dengan memilih *button* kelola sampah maka akan muncul *pop-up* yang berisi informasi kelola sampah. Antarmuka jenis sampah sebagaimana tampak pada Gambar 3.17.



Gambar 3.17 Rancangan antarmuka jenis sampah

h. Antarmuka Artikel

Antarmuka artikel akan tampil ketika masyarakat memilih menu artikel terkini yang ada di antarmuka beranda. Antarmuka ini berisi informasi kumpulan artikel mengenai sampah. Kumpulan artikel akan tampak sebagaimana pada Gambar 3.18.

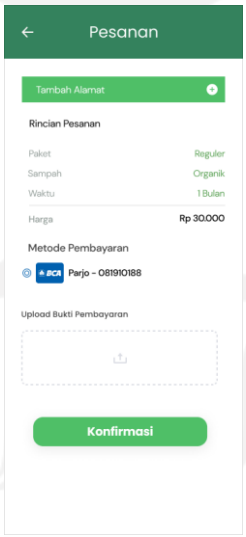
Gambar 3.18 Rancangan antarmuka kumpulan artikel

Selanjutnya, setiap artikel yang akan dipilih akan menampilkan informasi ringkasan sebagaimana tampak pada Gambar 3.19 sesuai dengan judul artikel yang ada. Adapun informasi yang lebih detail masyarakat harus memilih *link* yang tersedia karena *link* tersebut akan mengarahkan ke sebuah situs artikel yang lebih kompleks.

Gambar 3.19 Rancangan antarmuka informasi artikel

i. Antarmuka Pesanan

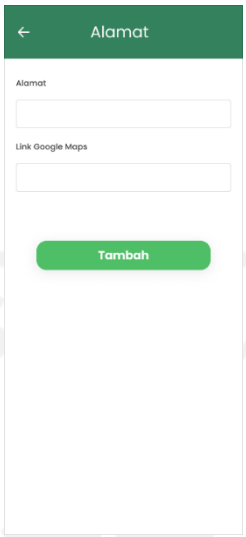
Antarmuka ini sebagai informasi data paket sampah yang telah dipilih. Tampilan ini terdapat item tambah alamat, rincian pesanan, metode pembayaran, dan bukti pembayaran. Tampilan antarmuka pesanan sebelum menambahkan alamat akan tampak sebagaimana pada Gambar 3.20.



Gambar 3.20 Rancangan antarmuka pesanan tanpa alamat

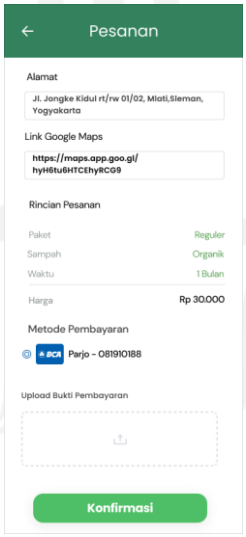
Sehubungan antarmuka pesanan, masyarakat wajib menambahkan alamat sebagai titik penjemputan sampah. Tambah alamat terdapat item alamat dan *link* Google Maps, di mana Google Maps untuk masyarakat dapat menambahkan hasil *link* alamat dari Google Maps

sebagai titik akurat akses penjemputan sebagaimana rancangan antarmuka yang bisa dilihat pada Gambar 3.21.



Gambar 3.21 Rancangan antarmuka tambah alamat

Setelah menambahkan alamat, masyarakat akan menuju ke tampilan pesanan. Pada tampilan pesanan akan muncul *layout text bar* pada item alamat dan *link* Google Maps dengan menampilkan hasil alamat yang telah dimasukkan oleh masyarakat. Tampilan ini akan tampak sebagaimana pada Gambar 3.22.

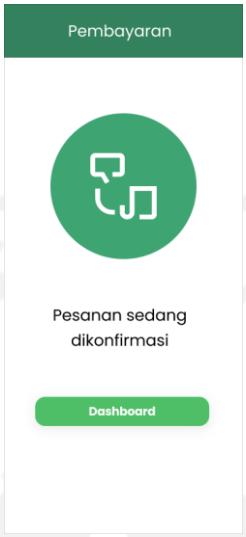


Gambar 3.22 Rancangan antarmukan pesanan setelah tambah alamat

j. Antarmuka Pembayaran

Antarmuka pembayaran akan muncul ketika masyarakat telah menyelesaikan pemesanan paket sampah. Antarmuka ini hanya memberikan informasi pada masyarakat bahwa pesanan

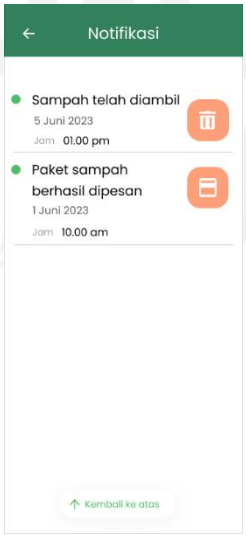
paket akan segera dilakukan konfirmasi oleh pihak pengepul sampah. Untuk itu, masyarakat dapat menunggu mengenai hasil pesanan jasa angkut sampah. Antarmuka pembayaran akan tampak sebagaimana pada Gambar 3.23.



Gambar 3.23 Rancangan antarmuka pembayaran

k. Antarmuka Notifikasi

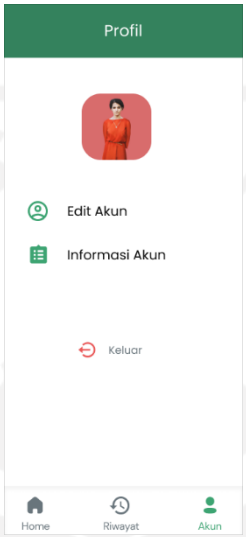
Antarmuka notifikasi akan muncul ketika *icon* notif dipilih pada antarmuka beranda yang berada di *header* sebelah kanan nama pengguna. Tampilan ini akan memberikan informasi terbaru mengenai aktivitas masyarakat. Bagian *bottom* terdapat *button* "kembali ke atas" yang berfungsi untuk melakukan aksi *scroll* ke paling atas secara cepat tanpa perlu menggeser secara perlahan. Antarmuka notifikasi akan tampak sebagaimana pada Gambar 3.24.



Gambar 3.24 Rancangan antarmuka notifikasi

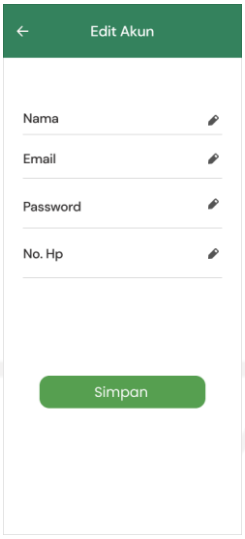
l. Antarmuka Profil

Antarmuka profil akan menampilkan informasi yang berhubungan dengan identitas masyarakat. Informasi yang diberikan terdapat item edit akun dan informasi akun. Selain itu, ada item *log-out* sebagai akses keluar dari aplikasi. Antarmuka profil akan tampak sebagaimana pada Gambar 3.25.



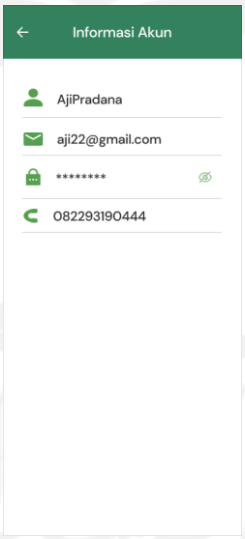
Gambar 3.25 Rancangan antarmuka profil

Di samping itu, masyarakat dapat memilih salah satu dari dua informasi yang tersedia pada antarmuka profil. Ketika *user* memilih item edit akun, akan ditampilkan *form* edit akun yang berisi nama, *email*, *password*, dan nomor telepon. Data yang diperbarui di antarmuka ini nantinya akan muncul pada antarmuka informasi akun dengan data yang berbeda. Perancangan antarmuka ini sebagaimana tampak pada Gambar 3.26.



Gambar 3.26 Rancangan antarmuka edit akun

Jika masyarakat memilih item informasi akun, akan ditampilkan antarmuka informasi akun. Informasi akun akan berisi nama, *email*, *password*, dan nomor telepon masyarakat. Data yang muncul terhubung dengan data *register* pada saat masyarakat mendaftarkan identitas diri pada aplikasi. Antarmuka informasi akun sebagaimana akan tampak pada Gambar 3.27.

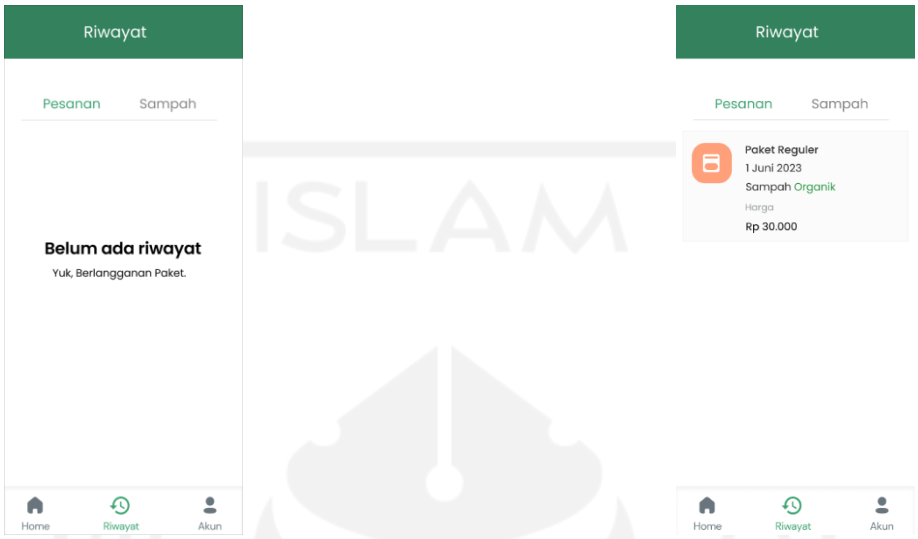


Gambar 3.27 Rancangan antarmuka informasi akun

m. Antarmuka Riwayat

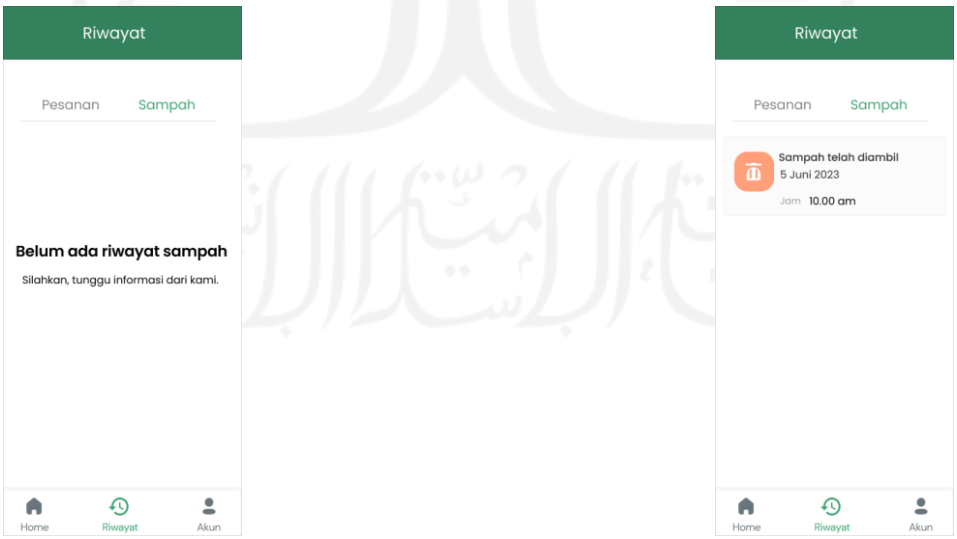
Antarmuka riwayat akan muncul ketika masyarakat memilih *bottom navigation* riwayat. Tampilan ini terdapat item berupa riwayat pesanan dan riwayat sampah. Dalam antarmuka ini, masyarakat akan langsung diarahkan pada item pesanan. Item pesanan memiliki informasi yang dapat dilihat oleh masyarakat, seperti nama paket sampah bulanan, tanggal mulai

berlangganan, dan harga paket yang dilanggan apabila telah melakukan pemesanan paket sampah. Tampilan ini nantinya dapat dilakukan *scrolling view* apabila data pesanan yang tampil melebihi batas *layout* antarmuka. Antarmuka riwayat akan tampak sebagaimana pada Gambar 3.28.



Gambar 3.28 Rancangan antarmuka riwayat pesanan

Selain itu, masyarakat dapat memilih item riwayat sampah yang akan menampilkan informasi berupa status sampah, tanggal pengangkutan, dan jam. Apabila data sampah belum ada, akan ditampilkan informasi mengenai riwayat sampah kosong. Antarmuka riwayat sampah akan tampak sebagaimana pada Gambar 3.29.

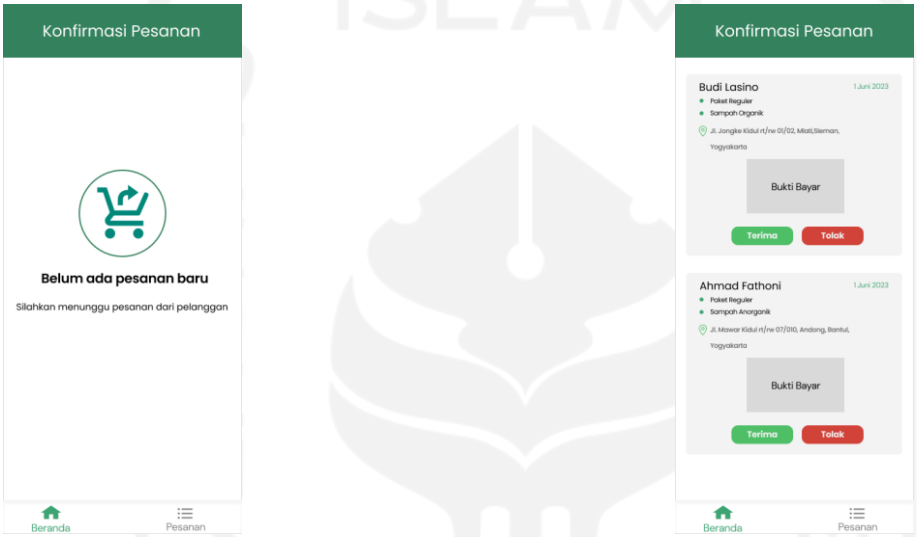


Gambar 3.29 Rancangan antarmuka riwayat sampah

Perancangan Aplikasi Untuk Pengepul Sampah

a. Antarmuka Konfirmasi Pesanan

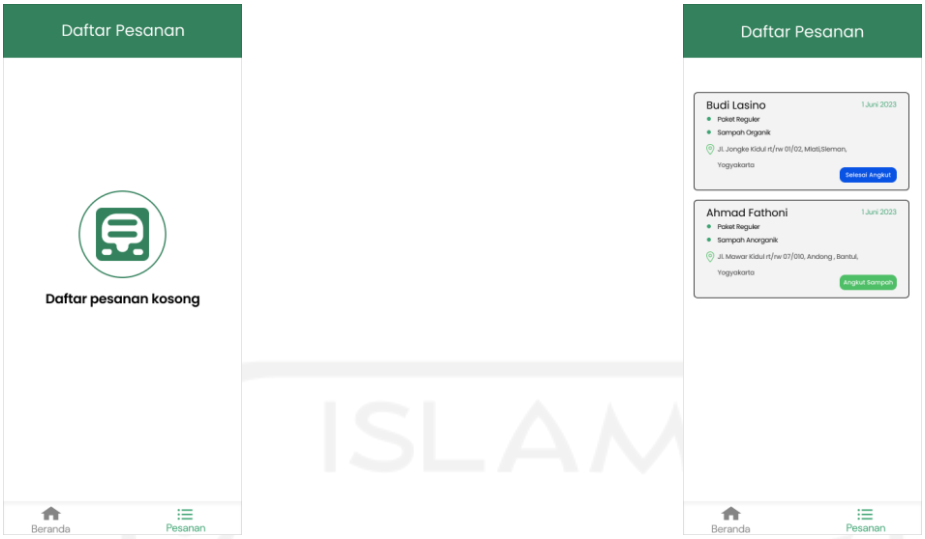
Antarmuka konfirmasi pesanan akan menampilkan *form* pesanan yang akan muncul ketika masyarakat telah melakukan pemesanan paket sampah. Setiap *form* data berisi informasi nama, paket yang dipesan, tanggal, alamat, dan bukti pembayaran yang telah dikirim oleh masyarakat. Selain itu, jika tidak ada *form* pesanan yang muncul atas permintaan jasa dari masyarakat, akan disajikan tampilan yang berbeda dengan memberikan informasi singkat berupa belum ada pesanan baru. Antarmuka konfirmasi pesanan akan tampak sebagaimana pada Gambar 3.30.



Gambar 3.30 Rancangan antarmuka konfirmasi pesanan

b. Antarmuka Daftar Pesanan

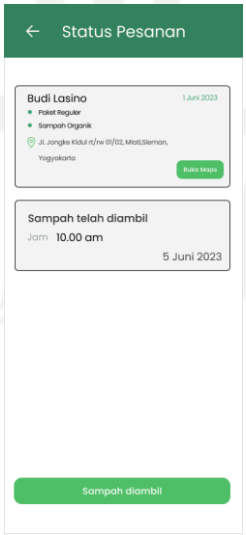
Antarmuka daftar pesanan akan menampilkan informasi mengenai daftar orang yang berlangganan paket sampah yang aktif selama satu bulan. Setiap data yang muncul berisi informasi nama pelanggan jasa angkut sampah, paket sampah yang dipesan, alamat pengangkutan, dan tanggal mulainya berlangganan. Selanjutnya, terdapat *button* angkut sampah yang berwarna hijau akan menuju ke antarmuka status pesanan sedangkan *button* selesai angkut berwarna biru menandakan bahwa sampah telah dilakukan pengangkutan. Selain itu, jika tidak terdapat daftar pesanan milik masyarakat yang aktif, akan disajikan tampilan yang berbeda dengan memberikan informasi singkat berupa daftar pesanan kosong. Antarmuka daftar pesanan akan tampak sebagaimana pada Gambar 3.31.



Gambar 3.31 Rancangan antarmuka daftar pesanan

c. Antarmuka Status Pesanan

Antarmuka status pesanan akan menampilkan data status pengangkutan. Setiap data pesanan pada antarmuka data pesanan yang akan dipilih akan menuju ke tampilan ini dengan meng-*generate* identitas data pesanan. Terdapat informasi yang akan menampilkan data baru yang berisi status sampah, jam, dan tanggal saat pengangkutan sampah telah dilaksanakan. Selain itu, terdapat *button* buka *maps* yang akan mengarahkan ke tampilan luar dari aplikasi dan menuju ke Google Maps. Antarmuka status pesanan akan tampak sebagaimana pada Gambar 3.32.



Gambar 3.32 Rancangan antarmuka status pesanan

3.3 Tahap Pengujian dan Evaluasi *Prototype*

Tahap ini dilakukan oleh *hipster* untuk proses pengujian terhadap *prototype* sistem serta mengevaluasi apakah *prototype* yang telah dibuat sudah sesuai dengan apa yang diharapkan oleh pengguna. Pengujian *prototype* nantinya diujikan pada pengguna dari sistem tersebut. Teknik yang dilakukan dalam proses pengujian *prototype* menggunakan *software* Figma dengan menjalankan fitur *prototyping* yang ada di dalamnya. Pengujian tersebut dapat dilakukan dengan berinteraksi langsung dengan pengguna yang didampingi *hipster*. Jika dalam proses pengujian terdapat *feedback* dari pengguna, akan dilakukan proses evaluasi atau *re-design*. Proses evaluasi akan terus berlanjut atau berulang-ulang apabila desain yang diujikan masih kurang memuaskan bagi pengguna.

3.3.1 Pengujian *Prototype*

Proses pengujian dilakukan pada seluruh antarmuka dari aplikasi yang dibangun dengan melakukan interaksi langsung kepada pengguna sebagai pengujian dengan iterasi pertama. Dalam pengujian, pengguna cukup menjalankan dan mengamati *prototyping* yang sudah disediakan dengan dibantu arahan dari *hipster* agar pengujian dapat berjalan dengan lancar dan tepat.

3.3.2 Evaluasi *Prototype*

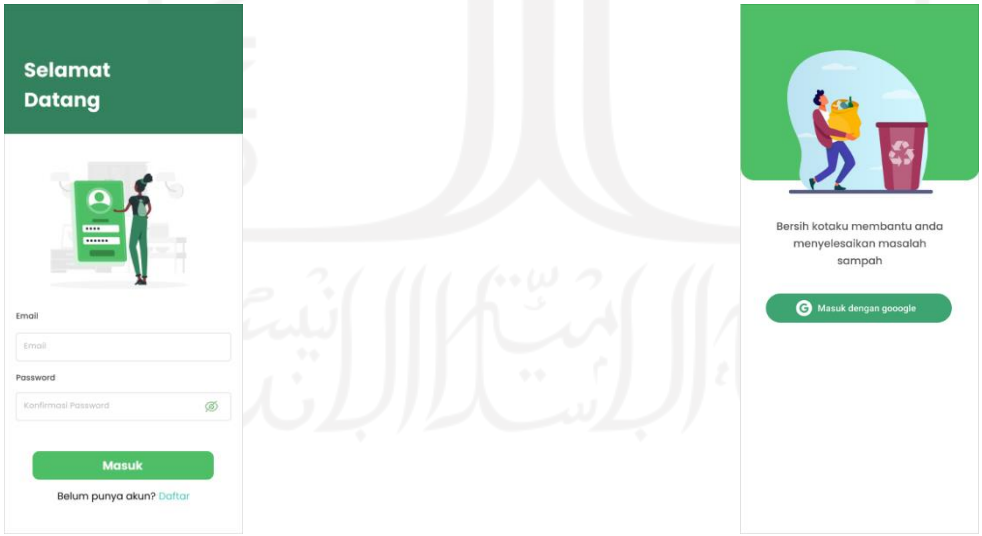
Sehubungan proses pengujian telah dilakukan pada pengguna, terdapat hal yang menjadi perhatian karena dalam pengujian terdapat *feedback* oleh pengguna pada beberapa antarmuka aplikasi. Hasil dari pengujian *prototype* yang dilakukan pada masyarakat sebanyak sembilan orang, dari sembilan orang tersebut memberikan masukkan terhadap rancangan pertama kali disajikan. Berdasarkan kesembilan orang tersebut, yang memberikan masukkan relatif hampir sama yang kemudian akan dilakukan evaluasi atau *re-design* sesuai dengan masukkan masyarakat sampai desain yang dirancang dapat diterima dengan puas dan baik oleh masyarakat. Selanjutnya, hasil pengujian desain yang dilakukan pada pengepul sampah sebanyak lima orang, di antara lima orang tersebut ketika mengetahui desain aplikasi untuk pengepul sampah tidak memberikan masukkan atau menerima rancangan desain yang sudah dibuat dengan pertama kalinya. Hasil pengujian *prototype* didapatkan dari *hipster* yang merupakan tugas mengenai hal ini. Berikut beberapa desain yang hilang di antaranya antarmuka *register*, data sampah, jenis sampah, dan artikel karena terdapat masukkan dari

masyarakat. Bagian *register* dideskripsikan pada antarmuka *login*, antarmuka data sampah dideskripsikan pada antarmuka paket, dan antarmuka jenis sampah serta artikel dideskripsikan pada antarmuka beranda. Pengujian dengan iterasi pertama terdapat masukkan dari pengguna sehingga dilakukan evaluasi dan perbaikan tampilan terhadap beberapa antarmuka aplikasi antara lain:

Evaluasi *Prototype* Untuk Masyarakat

a. Antarmuka *Login*

Bagian antarmuka *login* yang ditunjukkan pada Gambar 3.33 dilakukan proses desain ulang dengan menyesuaikan *feedback* dari masyarakat. Pada saat pengujian, masyarakat memberikan masukkan terhadap proses masuk dalam aplikasi yang dapat dilakukan dengan efisien, seperti menggunakan akun Google sehingga membuat masyarakat tidak perlu *register* secara detail pada aplikasi serta masyarakat saat ini memiliki akun *email* yang sehari-harinya sering digunakan dengan adanya kegiatan masyarakat yang makin tinggi dalam menggunakan produk digital. Dengan begitu, tim pengembang melakukan evaluasi terhadap proses *register* yang secara otomatis dalam aplikasi tidak lagi dibutuhkan dan proses *login* dilakukan perubahan tampilan dengan membuat proses masuk dalam antarmuka beranda menggunakan akun *email* yang terhubung dengan Google.

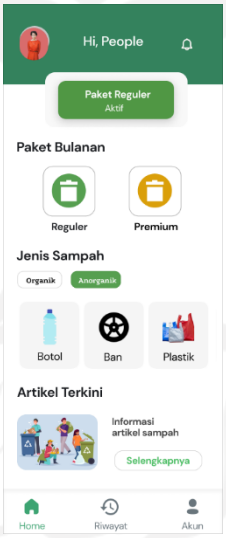


Desain lama Desain baru

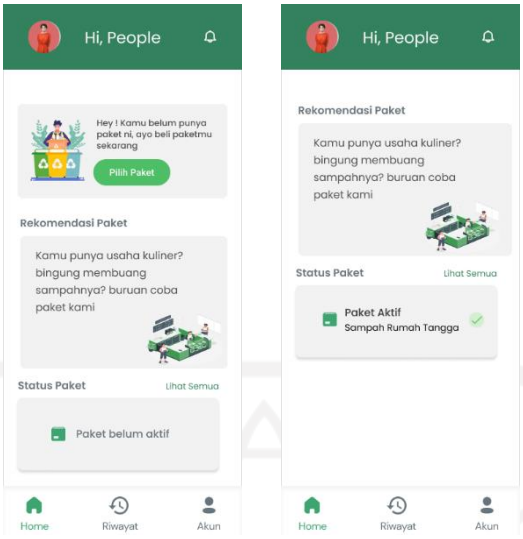
Gambar 3.33 Evaluasi antarmuka *login*

b. Antarmuka Beranda

Bagian antarmuka beranda yang ditunjukkan pada Gambar 3.34 telah dilakukan desain ulang dengan menyesuaikan *feedback* dari masyarakat. Proses perubahan desain dilakukan yang pertama pada menu jenis sampah karena informasi yang disampaikan dengan daya tarik mengenai informasi tersebut kurang terhadap masyarakat. Yang kedua pada menu artikel terkini, informasi berita ini mengenai artikel sampah akan tetapi, bagi masyarakat mengenai artikel yang tersedia ini kurang menarik, tidak memberikan rasa edukasi, dan tampilan kurang nyaman. Hal itu, membuat masyarakat dan tim pengembang melakukan evaluasi dan sepakat untuk menghapus informasi mengenai sampah. Perubahan desain yang baru memiliki informasi yang berisi fitur beli paket untuk jasa angkut sampah, item rekomendasi paket sebagai informasi awal paket pada masyarakat, dan terakhir fitur status paket mengenai keadaan paket aktif dan tidaknya. Selanjutnya, ketika masyarakat telah berlangganan paket sampah, akan ditampilkan perubahan mengenai item keadaan status paket yang semula belum aktif menjadi paket aktif serta pembelian paket sampah lanjutan belum dapat dilakukan apabila paket sampah saat ini masih aktif. Hal tersebut menjadikan aplikasi lebih fleksibel digunakan, *simplify*, dan spesifik.



Desain lama

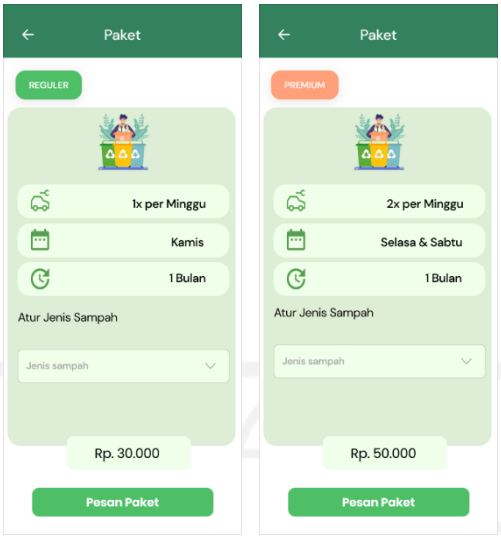


Desain baru

Gambar 3.34 Evaluasi antarmuka beranda

c. Antarmuka Paket

Pada antarmuka paket yang ditunjukkan pada Gambar 3.35 dilakukan proses desain ulang karena masyarakat merasa kurang tertarik dengan adanya pengaturan jenis sampah yang akan dibuang. Kebiasaan masyarakat dalam membuang sampah dengan notabene yang langsung membuang tanpa ingin susah melakukan pemilahan sampah. Hal tersebut, membuat proses isi data sampah organik atau anorganik tidak dapat dilakukan karena tidak adanya proses pemilahan sampah yang akan dibuang. Masyarakat saat ini dengan adanya pilah sampah masih cukup sulit untuk ditekankan apabila tidak bersamaan dengan penerjunan atau sosialisasi langsung pada masyarakat. Sehubungan hal itu, tim pengembang yang melibatkan pengguna melakukan proses evaluasi di mana paket sampah yang akan ditawarkan berbeda tidak ada pemilahan sampah yang akan dibuang melainkan paket-paket sampah berdasarkan sumbernya, seperti paket sampah rumah tangga, sampah kos, sampah restoran, dan sampah kantor maupun jenis paket sampah lainnya sesuai paket yang ditawarkan oleh pengepul sampah dengan informasi berupa jadwal pengangkutan sampah dan harga paket yang bervariasi.



Desain lama

Desain baru

Gambar 3.35 Evaluasi antarmuka paket

d. Antarmuka Pesanan

Pada antarmuka pesanan yang ditunjukkan pada Gambar 3.36 dilakukan proses perubahan isi data yang dilakukan tim pengembang menyesuaikan dengan ketersediaan paket sampah pada antarmuka paket yang telah mengalami proses evaluasi sehingga bagian rincian pesanan hanya terdapat item paket yang telah dipilih oleh masyarakat.

Desain lama Desain baru

Gambar 3.36 Evaluasi antarmuka pesanan

e. Antarmuka Informasi Akun

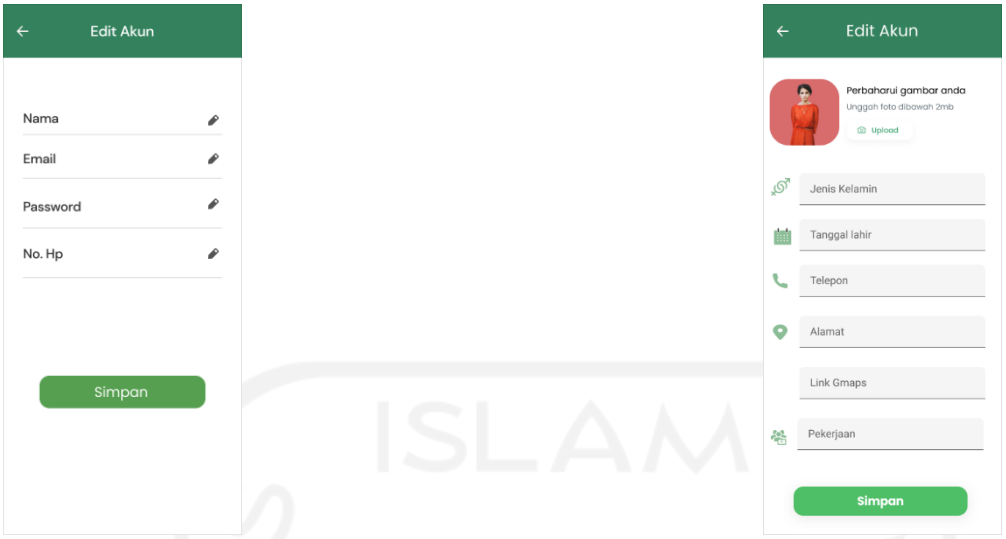
Pada antarmuka informasi akun yang ditunjukkan pada Gambar 3.37 dilakukan proses desain ulang menyesuaikan dari sisi pengguna dan pengembang. Informasi akun di dalam antarmuka ini berdasarkan proses *login* ketika menggunakan identitas *email* yang terikat dengan Google. Selain itu, *feedback* yang diberikan masyarakat di antaranya terlalu banyak ruang kosong dengan informasi data akun yang sedikit sehingga terkesan kurang nyaman untuk dilihat. Untuk melengkapi kekosongan pada *layout*, pengguna memberikan saran untuk menambahkan data identitas akun berupa jenis kelamin, tanggal lahir, nomor telepon, alamat, dan pekerjaan. Keberadaan data jenis kelamin, tanggal lahir, dan pekerjaan untuk menghindari pemalsuan identitas dari pihak masyarakat dan membantu pengepul sampah dalam melakukan penjemputan sampah dengan memastikan data yang valid dari masyarakat sehingga meminimalisir terjadinya pemesanan palsu dalam proses jasa angkut sampah. Data identitas tambahan tersebut nantinya dapat diperbarui isi datanya melalui edit akun. Dengan begitu, tim pengembang melakukan evaluasi menyesuaikan dengan antarmuka *login* karena proses *login* hanya menggunakan *email* yang terhubung Google sehingga data yang muncul di antarmuka ini, seperti *email* sesuai dengan data yang terdaftar di Google, nama pengguna, dan data tersebut tidak dapat diperbarui serta terdapat identitas tambahan berdasarkan umpan balik dari pengguna.

Desain lama Desain baru

Gambar 3.37 Evaluasi antarmuka informasi akun

f. Antarmuka Edit Akun

Pada antarmuka edit akun yang ditunjukkan pada Gambar 3.38 dilakukan proses desain ulang menyesuaikan dari sisi masyarakat dan pengembang. Ketika pengujian telah selesai, masyarakat menginginkan di dalam antarmuka edit akun terdapat foto profil akun serta foto profil tersebut dapat dilakukan perubahan setiap kali dibutuhkan masyarakat. Selain itu, sisi warna pada *button* simpan kurang menarik. Selain itu, tim pengembang melakukan evaluasi terhadap tampilan pada *form* edit data karena data yang dapat ditambah atau diedit menyesuikan pada tampilan informasi akun. Dengan begitu, data yang dapat di-*update* maupun ditambahkan oleh masyarakat di antaranya jenis kelamin, tanggal lahir, nomor telepon, alamat, dan pekerjaan. Data yang telah ditambahkan maupun di-*update* akan muncul pada antarmuka informasi akun.



Desain lama Desain baru

Gambar 3.38 Evaluasi antarmuka edit akun

Setelah proses iterasi pertama dilakukan, *hipster* dapat melakukan kembali pengujian dengan iterasi yang kedua dari tahapan *prototyping* terhadap *prototype* yang telah dievaluasi untuk mengetahui desain apakah sudah sesuai dengan kebutuhan pengguna atau tidak. Sehubungan pengujian telah dilakukan *hipster* terhadap sembilan orang pengguna khususnya masyarakat, terdapat hasil pengujian mengenai perancangan desain yang baru dengan hasil pengujian tidak ditemukan lagi masukkan dari pengguna aplikasi Bersih Kotaku.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi sistem

Pengembangan sistem telah berhasil dilakukan dengan mengimplementasikan perencanaan yang telah dibuat pada tahapan metode *prototyping*. Sistem yang berhasil dikembangkan secara langsung melibatkan pengembang *(programmer)* untuk melakukan pembangunan aplikasi lebih lanjut. Hal ini tentu memudahkan *programmer* dalam pengembangan karena tidak perlu memikirkan rancangan seperti apa yang harus diimplementasikan dalam sistem. Proses pengembangan sistem terdiri dari dua bagian yang harus dikerjakan**,** yaitu *frontend* dan *backend*. *Frontend* (*client-side*) merupakan proses pengkodean yang berfokus pada pembuatan tampilan aplikasi yang bisa dirasakan pengguna, sedangkan *backend* merupakan proses pengkodean yang berfokus pada sistem di balik aplikasi yang mengolah *server* dan *database.* Proses pengembangan sistem menggunakan *software* Android Studio yang merupakan *Integrated Development Environment* (IDE) yang mempermudah dalam mengembangkan aplikasi untuk Android. Untuk bahasa pemrograman menggunakan bahasa Java, dan bahasa ini memerlukan akses ke *Android Software Development Kit* (SDK) agar aplikasi yang dijalankan dapat berjalan mulus di perangkat. Selain proses *frontend*, sebuah aplikasi tentu memerlukan basis data untuk proses penyimpanan data. Basis data dalam aplikasi ini menggunakan *firebase* yang disediakan oleh Google secara gratis untuk mempercepat pekerjaan para *developer* karena *firebase* terdapat fitur-fitur yang menangani urusan *backend* secara langsung sehingga memperkecil kode yang akan dilakukan dalam pengembangan aplikasi.

Sistem yang dihasilkan berupa sistem pengangkutan sampah berbasis Android yang diberi nama aplikasi Bersih Kotaku. Sesuai dengan tahap analisis, sistem yang dibangun terdiri dari dua aplikasi dengan dua pengguna, yaitu masyarakat sebagai pengguna utama dari aplikasi Bersih Kotaku dan pengepul sampah sebagai admin untuk mengelola jasa pengangkutan sampah. Dalam aplikasi Bersih Kotaku untuk masyarakat terdapat antarmuka masuk untuk melakukan hak akses ke dalam antarmuka utama sebuah aplikasi. Jika masyarakat berhasil masuk, sistem akan menampilkan antarmuka beranda berisi informasi pembelian paket sampah. Pembelian paket sampah memiliki berbagai varian yang dapat dipilih. Setelah masuk dalam pembelian paket, masyarakat dapat memilih salah satu dari paket sampah yang ada, masyarakat dapat lanjut ke antarmuka berikutnya untuk melakukan pembayaran paket sampah

melalui transfer ke nomor rekening yang sudah tertera serta menambahkan alamat untuk titik lokasi pengangkutan sampah. Setelah melakukan pembayaran, data pesanan dapat dikirim dan masyarakat menunggu pesanan bahwa telah terkonfirmasi oleh pengepul sampah. Data pesanan akan disimpan di dalam *database firestore firebase* dan pihak pengepul sampah dapat melakukan proses konfirmasi pesanan di dalam aplikasi pengepul sampah (admin).

4.1.1 Antarmuka Aplikasi Untuk Masyarakat

a. Antarmuka *Splash Screen*

Tampilan antarmuka *splash screen* sebagai tampilan awal ketika masyarakat membuka aplikasi dengan adanya waktu *splash screen* selama dua detik. Masyarakat tidak perlu melakukan aksi *klik* tampilan ataupun hal lain akan tetapi, menunggu sebentar proses berjalan dua detik. Setelah dua detik berjalan, tampilan ini akan menuju ke antarmuka *login*. Tampilan antarmuka tampak seperti pada Gambar 4.1.

Gambar 4.1 Antarmuka *splash screen*

Potongan kode:

Kode yang terlihat pada Gambar 4.2 melakukan pembuatan objek *thread* atau utas, di mana metode ini dijalankan ketika masyarakat mulai masuk pada utas tersebut. Dalam objek *thread*, dapat mengatur durasi waktu pada utas *splash screen* menggunakan *class* TimeUnit dengan memberikan durasi dua detik pada parameter *timeout* yang artinya tampilan *splash screen* akan bertahan aktif di layar selama dua detik. Penggunaan metode *runOnUiThread* sebagai tindakan yang menjalankan utas saat UI *splash screen* kembali diakses. Setelah itu, mengambil objek FirebaseAuth dan memanggil fungsi *getCurrentUser* untuk mengetahui

pengguna yang masuk saat ini. Jika *CurrentUser* kondisi *null*, akan bernavigasi ke tampilan *login* yang dideklarasikan dari *id login_nav* pada *layout login.* Jika *CurrentUser* kondisi bukan *null*, yang artinya pengguna saat ini sudah masuk dan tidak melakukan *logout*, akan bernavigasi ke tampilan beranda yang dideklarasikan dari *id nav_graph. Id* tersebut merupakan *id* yang mewakili seluruh tindakan navigasi antar tampilan dalam aplikasi.

```
new Thread(() -> {
    try {
        TimeUnit.SECONDS.sleep(2);
        getActivity().runOnUiThread(() -> {

            if (FirebaseAuth.getInstance().getCurrentUser() == null) {

NavHostFragment.findNavController(this).navigate(R.id.login_nav,null,
                        new NavOptions.Builder()
                                .setLaunchSingleTop(true)
                                .setPopUpTo(R.id.login_nav, true)
                                .build());
            } else {

NavHostFragment.findNavController(this).navigate(R.id.nav_graph, null,
                        new NavOptions.Builder()
                                .setLaunchSingleTop(true)
                                .setPopUpTo(R.id.nav_graph, true)
                                .build());
            }
        });

    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}).start();
```

Gambar 4.2 Kode program proses layar *splash screen*

b. Antarmuka *Login*

Tampilan antarmuka *login* akan muncul setelah proses *splash screen* selesai dilakukan. *Login* pada aplikasi ini berbeda dengan *login* pada umumnya karena masyarakat dapat langsung *login* menggunakan akun *email* yang terintegrasi dengan identitas Google tanpa perlu mendaftarkan identitas diri secara detail di dalam aplikasi sehingga membuat masyarakat lebih efisien dalam menggunakan aplikasi. Tampilan antarmuka tampak seperti pada Gambar 4.3.

Gambar 4.3 Antarmuka *login*

Potongan kode:

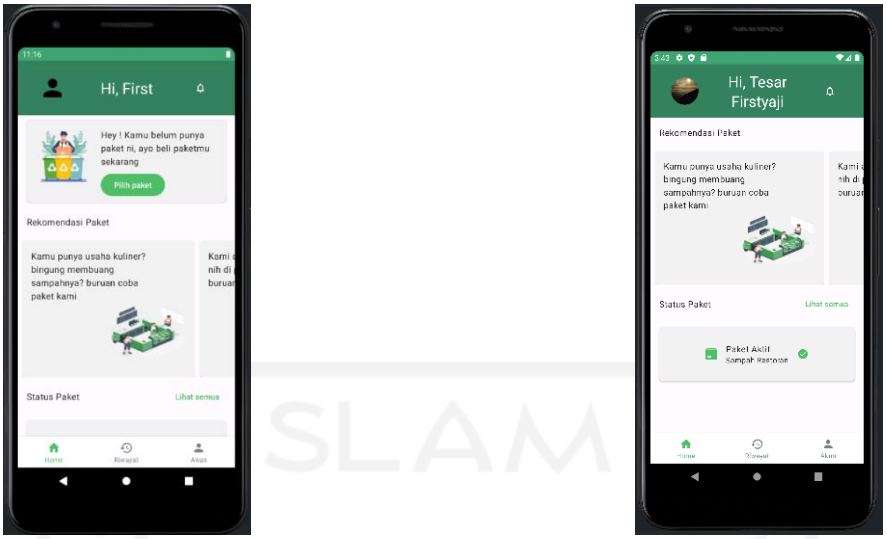
Kode yang terlihat pada Gambar 4.4 untuk memulai atau peluncuran aktivitas yang melakukan tindakan ditampilan lain dengan mendefinisikan *ActivityResultLauncher* dengan mendaftarkan permintaan untuk memulai aktivitas untuk hasil dari objek FirebaseAuthUIActivityResultContract. Objek tersebut memproses mengambil hasil dari proses *login* menggunakan identitas gabungan yang menerapkan *Firebase Authentication*. Penggunaan *authentication* memberikan efektivitas dalam *login*, karena FirebaseAuth menyediakan antarmuka *default* untuk autentikasi sehingga mempercepat proses masuk. Tindakan ini dideklarasikan pada fungsi *requestLogin* melalui objek AuthUI.IdpConfig yang menentukan konfigurasi proses *login* menggunakan identitas gabungan yang diinginkan, salah satunya Google serta memanggil fungsi *signInLauncher* dengan metode *launch* yang berisi variabel *signInIntent* untuk menampilkan antarmuka *default*. Selanjutnya, fungsi *onSignInResult* untuk menerima hasil proses *login* yang didapat dari konfigurasi masuk dengan Google dengan memanggil fungsi *loginRegister* untuk menambahkan data pengguna ke basis data bagi yang belum pernah *login* sama sekali ataupun membaca data pengguna yang pernah *login* pertama kali dan terautentikasi pada aplikasi.

```
private final ActivityResultLauncher<Intent> signInLauncher =
registerForActivityResult(new FirebaseAuthUIActivityResultContract(),
this::onSignInResult);

binding.loginButton.setOnClickListener(view1 -> {
    requestLogin();
});


private void requestLogin() {
    List<AuthUI.IdpConfig> providers = Collections.singletonList(
          // new AuthUI.IdpConfig.EmailBuilder().build()
            new AuthUI.IdpConfig.GoogleBuilder().build()
    );

    Intent signInIntent = AuthUI.getInstance()
            .createSignInIntentBuilder()
            .setAvailableProviders(providers)
            .setTheme(R.style.Theme_BersihKotaku)
            .build();
    signInLauncher.launch(signInIntent);
}

private void onSignInResult(FirebaseAuthUIAuthenticationResult result) {
    IdpResponse response = result.getIdpResponse();
    if (result.getResultCode() == RESULT_OK &&
FirebaseAuth.getInstance().getCurrentUser() != null) {
        mViewModel.loginRegister(FirebaseAuth.getInstance().getCurrentUser());
    } else {
        if (response != null && response.getError() != null) {
            showErrorMessage(String.format(Locale.ENGLISH, "No login",
response.getError().getErrorCode()));
        }
    }
}
```

Gambar 4.4 Kode program proses *login* untuk masyarakat pada aplikasi

c. Antarmuka Beranda

Tampilan antarmuka beranda menampilkan beberapa informasi untuk masyarakat di antaranya fitur utama berisi pembelian paket sampah, terdapat item rekomendasi paket sebagai informasi masyarakat untuk melakukan pembelian paket, dan item status paket yang nantinya menampilkan paket sampah yang telah aktif yang telah dipesan oleh masyarakat. Selain itu, tampilan ini terdapat menu notifikasi, menu navigasi yang terdiri *home*, riwayat, dan akun yang berfungsi untuk mengakses antarmuka lain untuk mempermudah masyarakat dalam menggunakan aplikasi serta tampilan ini dapat dilakukan secara *scrolling view*. Tampilan antarmuka tampak seperti pada Gambar 4.5.



Gambar 4.5 Antarmuka beranda

Potongan kode:

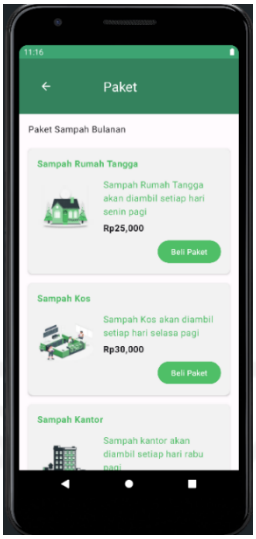
Pada antarmuka beranda, dibutuhkan layanan untuk menampilkan data pesanan yang telah di konfirmasi muncul pada item status paket sehingga masyarakat mengetahui bahwa paket yang dipesan telah berhasil dan aktif. Layanan ini membutuhkan sebuah *query* yang bisa membaca dan mengambil data dari basis data. Fungsi yang dilakukan dengan membuat variabel *listenCurrentActiveSubs* dengan membangun kueri yang membaca data dari koleksi *subscription* dengan memfilter *query whereEqualTo* untuk mengambil atau memberi beberapa data dokumen dari kolom *userID* dan status dengan keadaan *Accepted* sedangkan *query whereGreaterThanOrEqualTo* untuk mengambil nilai *validUntil*, di mana *validUntil* memfilter rentang selama satu bulan yang lebih besar sama dengan *currentTime* serta *query addSnapshotListener* untuk mengambil data dokumen secara *realtime* dari basis data sebagaimana yang terlihat pada Gambar 4.6. Penggunaan *addsnapshotListener* dapat memudahkan dalam memantau setiap data di dalam dokumen pada koleksi, di mana ketika ada proses penambahan data, perubahan data pada basis data maka blok kode *addsnapshotListener* akan segera dieksekusi.

```
public Observable<Optional<Subscription>> listenCurrentActiveSubs(String userID) {
    long currentTime = Instant.now().toEpochMilli();
    return Observable.create(emitter -> {
        ListenerRegistration listener = this.firestore.collection("subscriptions")
                .whereEqualTo("userID", userID)
                .whereGreaterThanOrEqualTo("validUntil", currentTime)
                .whereEqualTo("status", "Accepted").limit(1)
                .addSnapshotListener((value, error) -> {
                    if (error != null) {
                        emitter.onError(error);
                        return;
                    }

                    if (value != null && !value.isEmpty()) {
                        try {
                            for (DocumentSnapshot snapshot : value.getDocuments()) {
                                if (snapshot.exists()) {
                                    Subscription subscription =
Objects.requireNonNull(snapshot.toObject(Subscription.class))
                                            .withId(snapshot.getId());
                                    emitter.onNext(Optional.of(subscription));
                                }
                            }
                        } catch (Exception e) {
                            emitter.onError(e);
                        }
                    } else {
                        emitter.onNext(Optional.empty());
                    }
                });
        emitter.setCancellable(listener::remove);
    });
}
```

Gambar 4.6 Kode program membaca data pesanan paket dari basis data

d. Antarmuka Paket

Tampilan antarmuka paket menampilkan berbagai varian paket sampah yang akan dipesan oleh masyarakat. Varian paket sampah yang ada pada tampilan ini meliputi sampah rumah tangga, sampah kos, sampah kantor, dan sampah restoran. Setiap paket sampah memiliki informasi berupa waktu pengangkutan sampah dan harga paket yang berbeda. Paket sampah akan berlaku selama satu bulan, apabila masyarakat ingin menambahkan paket sampah lagi harus menunggu paket sampah yang awal telah melewati masa satu bulannya. Tampilan antarmuka tampak seperti pada Gambar 4.7.



Gambar 4.7 Antarmuka paket

Potongan kode:

Kode yang terlihat pada Gambar 4.8 melakukan pembuatan objek LiveData dengan mengimplementasikan status data berupa *List* pada *class* ViewModel. Objek ini bagian dari *class observable* yang bertindak sebagai pemegang data dan komponen data yang akan ditetapkan dapat diambil dengan berinteraksi pada *class service* untuk mendapatkan akses data yang dibutuhkan kemudian menyimpannya, seperti akses ke *class packService* pada data kueri *listenPack.* Status data paket yang ada dilakukan pengurutan data berdasarkan harga menggunakan *method* comparingDouble dari *interface Comparator* dengan mengumpulkan data harga ke dalam *list* melalui operator *collect* yang diatur pada operator *map*. Penggunaan operator *map* digunakan untuk transformasi item dari setiap model data yang akan dilakukan perubahan dengan melakukan *stream* data pada variabel *packs*. Selanjutnya, penggunaan operator *subscribeOn* untuk menentukan *Scheduler* yang berbeda saat *observable* pada objek *list Pack* harus menjalankan operasinya dan memancarkan data dari kueri *listenPack*. Objek LiveData berbasis *lifecycle* proses yang berarti objek ini akan memberi informasi berupa status data baik data sama atau adanya perubahan data ke pengontrol UI, seperti aktivitas atau *fragment.*

```
public LiveData<List<Pack>> getPack() {
    FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
    if (firebaseUser == null) {
        return null;
    }
    // access instance packservice
    Observable<List<Pack>> recommends = this.packService.listenPack()
            .map(packs -> packs.stream().sorted(Comparator.comparingDouble(pack
-> pack.price)).collect(Collectors.toList()))
            .subscribeOn(Schedulers.io());
    return
LiveDataReactiveStreams.fromPublisher(recommends.toFlowable(BackpressureStrateg
y.LATEST));
}
```

Gambar 4.8 Kode program menyimpan dan memegang data dari layanan basis data

Status data yang telah disimpan oleh objek LiveData pada *class* ViewModel dapat diamati atau *observer* oleh komponen UI yang pada dasarnya, yaitu *class fragment*. *Class* tersebut dapat menginisialisasi objek ViewModel menggunakan *ViewModelProvider* untuk memanggil *class PackViewModel*. Setiap pengguna saat mengakses tampilan paket, peristiwa UI akan memanggil variabel mViewModel yang berisi *method getPack* dan *method observer* yang mengambil objek *ViewLifecycleOwner* dan *packController*. Hal tersebut mendaftarkan objek packController ke objek LiveData. Penggunaan *method observe* melakukan pengamat pada fungsi *getPack* dari sumber data yang di-*observable* dan setiap kali data yang tersimpan dalam objek LiveData berubah maka proses *update* men-*trigger observer* dan objek *packController* akan mendapatkan informasi terkait data lama atau baru untuk ditampilkan status data tersebut dalam model data *List* sehingga pengontrol UI otomatis menerima data dan melakukan *update* apabila ada perubahan data sebagaimana proses ini terlihat pada Gambar 4.9.

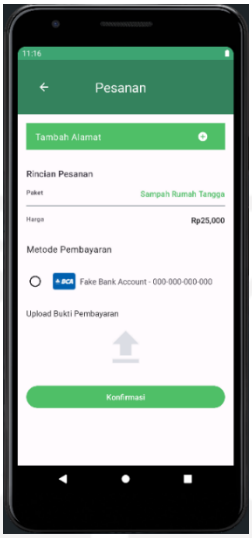
```
mViewModel = new ViewModelProvider(this).get(PackViewModel.class);

mViewModel.getPack().observe(getViewLifecycleOwner(),
packController::setPackList);
```

Gambar 4.9 Kode program menampilkan data paket pada UI

e. Antarmuka Pesanan

Tampilan antarmuka pesanan menampilkan data pesanan paket sampah yang telah dipilih oleh masyarakat pada antarmuka paket. Tampilan ini terdapat informasi berupa menu tambah alamat, rincian pesanan terdiri dari paket sampah yang dipesan, harga, dan metode pembayaran

sebagai akses bayar untuk masyarakat serta ada item *upload* bukti pembayaran sebagai bentuk data keseriusan dalam permintaan jasa pengangkutan sampah. Tampilan antarmuka tampak seperti pada Gambar 4.10.



Gambar 4.10 Antarmuka pesanan sebelum tambah alamat

Potongan kode:

Kode yang terlihat pada Gambar 4.11 untuk memulai atau peluncuruan aktivitas yang melakukan tindakan di tampilan lain saat masyarakat akan memproses unggah bukti pembayaran pada objek ActivityResultLauncher yang mengimplementasikan *Intent* dengan mendeklarasikan variabel *startImagePicker*. Penggunaan *registerForActivityResult* untuk mendaftarkan permintaan untuk memulai aktivitas untuk hasil dengan mengambil objek ActivityResultContracts. Objek tersebut mendefinisikan tindakan *intent* yang pada dasarnya proses ini mengambil gambar dari kamera atau *file* dokumen. Setiap aktivitas untuk hasil dari objek ActivityResultContracts, tindakan *Intent* dapat memproses untuk mengambil data. Jika hasil data tidak kosong dengan menetapkan data yang diperoleh dari tindakan *Intent* berupa data *Uri* yang kemudian memanggil fungsi *setCurrentReceipt* dengan parameter variabel *fileUrl*. Tindakan masyarakat untuk mengunggah bukti pembayaran dengan memanggil fungsi *setRequestReceiptOnClick* dengan parameter memanggil fungsi *requestImagePicker*. Fungsi *requestImagePicker* akan mulai proses menampilkan tampilan lain dan memproses hasil aktivitas dari variabel *startImagePicker* menggunakan *method launch*.

```
private final ActivityResultLauncher<Intent> startImagePicker =
registerForActivityResult(new
ActivityResultContracts.StartActivityForResult(), result -> {
    Intent data = result.getData();

    if (result.getResultCode() == Activity.RESULT_OK && data != null) {
        Uri fileUrl = data.getData();
        mViewModel.setCurrentReceipt(fileUrl);
    }
});

paymentController.setRequestReceiptOnClick(this::requestImagePicker);



private void requestImagePicker() {
    ImagePicker.with(this)
            .compress(1024)
            .createIntent(intent -> {
                startImagePicker.launch(intent);
                return null;
            });
}
```

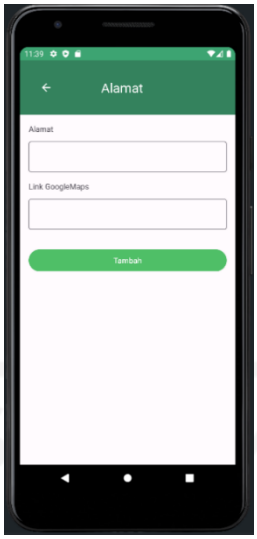
Gambar 4.11 Kode program untuk proses unggah *file* dari interaksi tampilan lain

Setelah memproses aktivitas untuk hasil dari objek ActivityResultConstracts, di mana masyarakat mengirimkan data unggah pada aplikasi dan hasil data diambil saat tindakan *Intent* dengan menetapkan data tersebut berupa *Uri* dengan memanggil fungsi *setCurrentReceipt* pada variabel *mViewModel*. Fungsi *setCurrentReceipt* melakukan operasi dengan membuat referensi jalur ke *file* dari objek FirebaseStorage agar sistem memperoleh akses ke *cloud storage* yang digunakan untuk mengunggah, mengunduh, mendapatkan, memperbaharui, dan menghapus data dengan membuat *method getReference* ke lokasi yang lebih dalam atau memiliki banyak *sub file* melalui *method child* dengan membuat referensi format *String*. Hasil data yang didapat dari proses kode di atas dapat diunggah di *cloud storage* melalui *method putFile* sebagaimana proses ini terlihat pada Gambar 4.12.

```
public void setCurrentReceipt(Uri fileURI) {
    FirebaseUser firebaseUser =
FirebaseAuth.getInstance().getCurrentUser();
    if (firebaseUser == null) {
        return;
    }
    FirebaseStorage.getInstance()

.getReference().child(String.format("users/%s/receipt/%s.jpeg",
firebaseUser.getUid(), UUID.randomUUID().toString()))
            .putFile(fileURI).addOnCompleteListener(task -> { // proses
upload file ke storage

                if (task.isSuccessful()) {

task.getResult().getStorage().getDownloadUrl().addOnCompleteListener(down
loadTask -> {
                        if (downloadTask.isSuccessful()) {

currentReceipt.accept(downloadTask.getResult().toString());
                        }
                    });
                } else {
                    currentReceipt.accept("");
                }
            });
}
```

Gambar 4.12 Kode program untuk mengirim hasil unggah *file* ke *cloud storage*

Setelah itu, masyarakat perlu menambahkan alamat rumah sebagai titik penjemputan sampah yang akan dilakukan oleh pengepul. Alamat yang ditambahkan berisi informasi alamat yang mencakup nama perumahan atau desa, kelurahan, kecamatan, sedangkan *link* Google Maps sebagai alamat dengan titik keakuratan rumah masyarakat dengan menambahkan *link* yang di dapat dari aplikasi Google Maps. Tampilan antarmuka akan tampak seperti pada Gambar 4.13.



Gambar 4.13 Antarmuka tambah alamat

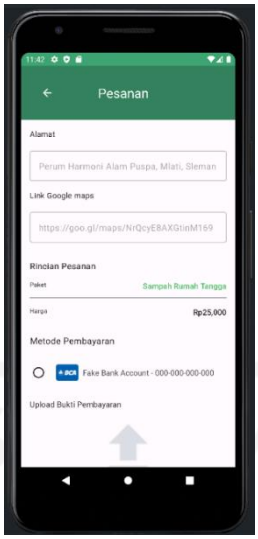
Potongan kode:

Proses isi data alamat terdapat dua jenis alamat, yaitu alamat rumah dan alamat akurat dari Google Maps yang dideklarasikan pada variabel *addressName* dan *addressDetails* sebagaimana yang terlihat pada Gambar 4.14. Setelah kedua variabel diisi datanya, akan dipanggil fungsi *updateAddress* dari *class ViewModel* untuk memperbarui atau menambah data alamat milik masyarakat di basis data serta di pengontrol UI. Pada fungsi *updateAddress* terdapat variabel *disposable* dari *class CompositeDisposable* untuk menambah permintaan tindakan lain dengan metode *add* yang berisi parameter *getUsers* untuk membaca data pengguna berupa *email* yang diteruskan dengan memanggil fungsi *concatMap*. Fungsi tersebut untuk memancarkan data baru *observable* (yang dapat diamati). Jika *userData* di-*set* fungsi *isPresent* untuk memeriksa apakah ada nilai atau tidak pada data pengguna khususnya masyarakat mengenai data *AddressName* dan *AddressDetails*. Setelah nilai dari data ada, dapat memproses tindakan tersebut untuk mengembalikan dengan memanggil fungsi *updateUsers* untuk memperbarui data alamat yang baru.

```
viewBinding.addressFragmentSave.setOnClickListener( button -> {
    String addressName =
Objects.requireNonNull(viewBinding.addressFragmentNameContent.getEditText()).getT
ext().toString();
    String addressDetails =
Objects.requireNonNull(viewBinding.addressFragmentDetailsContent.getEditText()).g
etText().toString();
    mViewModel.updateAddress(addressName, addressDetails);
    Toast.makeText(requireContext(), R.string.address_updated,
Toast.LENGTH_SHORT).show();
    NavHostFragment.findNavController(this).navigateUp();
});

void updateAddress(String name, String details) {
    FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
    if (firebaseUser == null) return;


disposable.add(userService.getUsers(firebaseUser.getEmail()).concatMap(userData -
> {
        if (userData.isPresent()) {
            User user = userData.get();
            user.AddressName = name;
            user.AddressDetails = details;
            return userService.updateUsers(user);
        } else {
            return Observable.just(Optional.empty());
        }
    }).onErrorComplete().subscribe());
}
```

Gambar 4.14 Kode program proses menambah data alamat

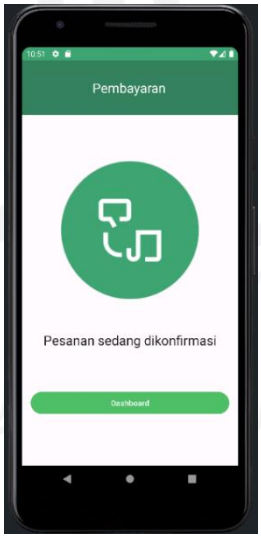
Jika telah menambahkan alamat, masyarakat akan kembali ke tampilan antarmuka pesanan dengan *layout* pada item alamat yang berbeda, di mana data alamat yang telah ditambahkan masyarakat akan muncul pada *layout text bar* alamat dan *link* Google Maps. Antarmuka ini dapat dilakukan *scrolling view*. Selanjutnya, masyarakat dapat mengirimkan pesanan jika semua data telah diisi pada tampilan ini. Tampilan antarmuka akan tampak seperti pada Gambar 4.15.



Gambar 4.15 Antarmuka pesanan setelah tambah alamat

f. Antarmuka Pembayaran

Tampilan antarmuka pembayaran menampilkan informasi berupa deskripsi pesanan sedang dilakukan konfirmasi, setelah masyarakat melakukan pengiriman data pesanan paket sampah pada pengepul sampah. Selama proses konfirmasi, masyarakat dapat kembali ke antarmuka utama untuk menunggu hasil konfirmasi pesanan paket sampah. Tampilan antarmuka akan tampak seperti pada Gambar 4.16.

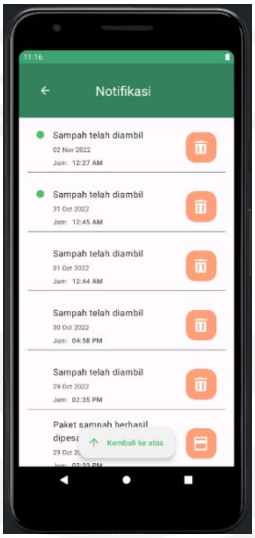


Gambar 4.16 Antarmuka pembayaran

g. Antarmuka Notifikasi

Tampilan antarmuka notifikasi menampilkan beberapa informasi terbaru mengenai aktivitas masyarakat di antaranya notifikasi ketika pengepul sampah telah menerima

konfirmasi pesanan atas jasa paket sampah dari masyarakat dan notifikasi ketika pengepul sampah telah melakukan pengangkutan sampah selama satu bulan. Selain itu, terdapat *button* "kembali ke atas" yang digunakan untuk menampilkan data notifikasi terbaru secara cepat tanpa melakukan *scroll* secara perlahan. Kondisi tersebut akan berlaku jika data yang muncul sudah memenuhi untuk dilakukan *scroll* ke bawah. Tampilan antarmuka akan tampak seperti pada Gambar 4.17.



Gambar 4.17 Antarmuka notifikasi

Potongan kode:

Kode yang terlihat pada Gambar 4.18 mengenai kueri membaca data notifikasi dari basis data dengan mendeklarasikan variabel *listenNotification* pada objek Observable yang mengimplementasikan data berupa *List* pada *class Notification* dengan melihat data dari subkoleksi *notification* yang terletak pada dokumen dari setiap masing-masing *userID* pada *container* koleksi *users* dengan memfilter kueri *orderBy* untuk mengurutkan berdasarkan data *createdAt* yang mengonversi berupa tanggal pengiriman notifikasi, di mana data tersebut diurutkan dari data yang diterima paling akhir. Data notifikasi yang masuk akan membatasi jumlah datanya dengan 50 data terbaru melalui kueri *limit.* Selanjutnya, dalam mengambil data secara *realtime,* dapat menambahkan pemroses *addSnapshotListener* untuk memperoleh atau mendengarkan hasil data saat ini atau setiap data terdapat perubahan pada dokumen. Hal tersebut, dapat membuat parameter *value* dan *error* pada *addSnapshotListener*. Jika *value* tidak kosong, dapat mendeklarasikan variabel *snapshot* pada objek DocumentSnapshot yang digunakan untuk mengetahui isi data berupa *value* yang dibaca dari dokumen dan mengecek apakah *value* yang dibaca dalam dokumen terdapat *value* yang berubah atau tidak dan

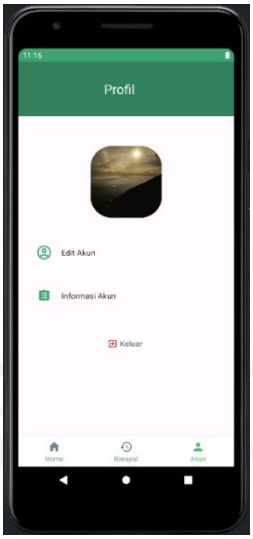
mengembalikan *value* dokumen tersebut yang ditangani *snapshot* ke objek Notification serta mengambil data dalam dokumen berdasarkan *id*. Data berupa *value* pada *notification* baik data saat ini atau baru akan diproses dengan memanggil *method add* pada variabel *notificationList* untuk menambahkan data *notification* ke objek ArrayList sebagai daftar data pada *Notification*. Penggunaan *snapshotListener* dapat menerima *snapshot* kueri baru setiap kali hasil kueri berubah.

```
public Observable<List<Notification>> listenNotification(String userID) {
    return Observable.create(emitter -> {
        ListenerRegistration listener = this.firestore.collection("users")
                .document(userID).collection("notification")
                .orderBy("createdAt", Query.Direction.DESCENDING)
                .limit(50)
                .addSnapshotListener((value, error) -> {
                    if (error != null) {
                        emitter.onError(error);
                        return;
                    }

                    if (value != null && !value.isEmpty()) {
                        try {

                            List<Notification> notificationsList = new
ArrayList<>();
                            for (DocumentSnapshot snapshot : value.getDocuments()) {
                                if (snapshot.exists()) {
                                    Notification notification =
Objects.requireNonNull(snapshot.toObject(Notification.class))
                                            .withId(snapshot.getId());
                                    notificationsList.add(notification);
                                }
                            }
                            emitter.onNext(notificationsList);
                        } catch (Exception e) {
                            emitter.onError(e);
                        }
                    }
                });
        emitter.setCancellable(listener::remove);
    });
}
```

Gambar 4.18 Kode program membaca data notifikasi dari basis data

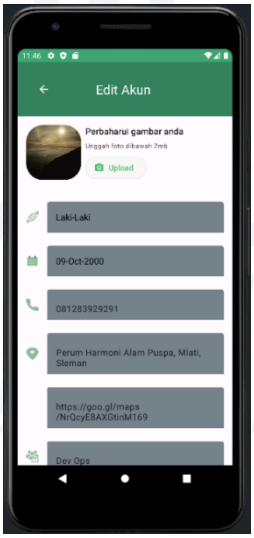
h. Antarmuka Profil

Tampilan antarmuka profil menampilkan informasi yang berkaitan dengan identitas masyarakat. Informasi dalam tampilan ini berisi edit akun, informasi akun, dan menu keluar. Sementara itu, menu keluar yang akan dilakukan masyarakat nantinya akan menuju ke antarmuka *login*. Tampilan antarmuka akan tampak seperti pada Gambar 4.19.



Gambar 4.19 Antarmuka profil

Selanjutnya, jika masyarakat memilih item edit akun, sistem akan menampilkan *form* edit akun, di mana *form* tersebut dapat dilakukan *scrolling view* dan masyarakat dapat menambah atau mengubah data sesuai dengan kebutuhan. Dalam *form* edit berisi data tanggal lahir, jenis kelamin, nomor telepon, alamat, dan pekerjaan serta masyarakat bisa perbarui gambar profil dengan memilih *button upload* dengan kriteria foto yang menyesuaikan keterangan yang ada sebagaimana tampak seperti pada Gambar 4.20.



Gambar 4.20 Antarmuka edit akun

Potongan kode:

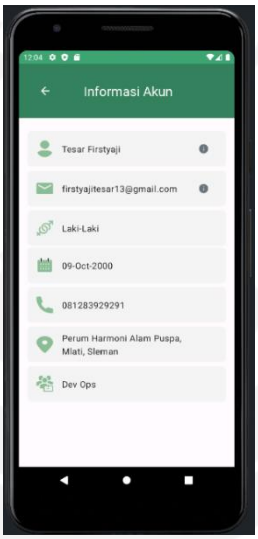
Kode yang terlihat pada Gambar 4.21 sebagai tindakan untuk memperbarui atau menambah identitas data masyarakat dengan memanggil fungsi *updateData* yang diterapkan

pada *method setOnClickListener* yang merujuk dari *id editProfileSave* yang berkaitan dengan *widget Button.* Fungsi tersebut terdiri enam variabel data yang bisa dilakukan *editText* untuk menginput data sesuai kebutuhan dan memanggil *method updateProfile* dari variabel mViewModel untuk meneruskan data yang akan diperbarui untuk mengisi data terbaru pada basis data maupun data pada antarmuka informasi akun.

```
viewBinding.editProfileSave.setOnClickListener(v -> updateData());

void updateData() {
    String date =
viewBinding.editProfileDate.editProfileComponentTextInput.getText().toString();
    String gender =
viewBinding.editProfileGender.editProfileComponentTextInput.getText().toString();
    String phone =
Objects.requireNonNull(viewBinding.editProfilePhone.editProfileComponentTextInput.
getEditText()).getText().toString();
    String address =
Objects.requireNonNull(viewBinding.editProfileAddress.editProfileComponentTextInpu
t.getEditText()).getText().toString();
    String addressMaps =
Objects.requireNonNull(viewBinding.editProfileAddressMaps.editProfileComponentText
Input.getEditText().getText().toString());
    String job =
Objects.requireNonNull(viewBinding.editProfileJob.editProfileComponentTextInput.ge
tEditText()).getText().toString();

    mViewModel.updateProfile(date, gender, phone, address, addressMaps, job);
    Toast.makeText(requireContext(), R.string.profile_updated,
Toast.LENGTH_SHORT).show();
}

public void updateProfile(String date, String gender, String phone, String
address, String addressMaps, String job) {
    FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
    if (firebaseUser == null) {
        return;
    }
    disposable.add(
            this.userService.getUsers(firebaseUser.getEmail()).concatMap(user -> {
                if (user.isPresent()) {
                    User data = user.get();
                    data.dayBirth = date;
                    data.gender = parseGender(gender);
                    data.phoneNumber = phone;
                    data.AddressName = address;
                    data.AddressDetails = addressMaps;
                    data.job = job;
                    return this.userService.updateUsers(data);
                } else {
                    return Observable.error(new Throwable("Gagal update data"));
                }
            }).observeOn(Schedulers.io()).onErrorComplete().subscribe()
    );
}
```

Gambar 4.21 Kode program meng-*update* data identitas masyarakat

Jika masyarakat memilih item informasi akun, sistem akan menuju ke antarmuka informasi akun. Informasi akun berisi identitas masyarakat, dan sebagian ada data yang tidak dapat di-*update* di sistem aplikasi di antaranya data nama dan *email*. Untuk data lainnya, seperti tanggal lahir, jenis kelamin, nomor telepon, alamat, dan pekerjaan awalnya data ini masih kosong ketika masyarakat masuk ke aplikasi. Masyarakat tentu saja dapat melakukan isi data terhadap masing-masing data sesuai dengan kebutuhan masyarakat. Tampilan antarmuka akan tampak pada Gambar 4.22.



Gambar 4.22 Antarmuka informasi akun

Potongan kode:

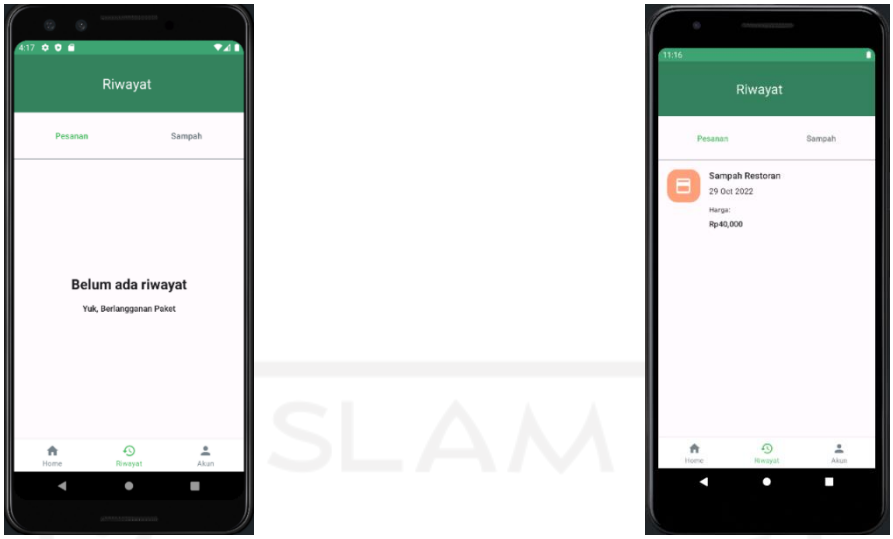
Pada antarmuka informasi akun, dibutuhkan layanan untuk menampilkan data pengguna khususnya masyarakat. Layanan ini membutuhkan sebuah *query* untuk membaca data dari basis data yang dideklarasikan pada variabel *listenUser* dengan parameter *email.* Dalam variabel tersebut dapat dibangun kueri yang membaca data dari koleksi *users* dengan memfilter *query whereEqualTo* untuk mengambil data dokumen dari kolom *email* yang mewakili keseluruhan data dengan membatasi hanya mengambil satu identitas *email* milik masyarakat melalui *query limit* yang diatur pada koleksi sebagaimana yang terlihat pada Gambar 4.23. Selain itu, operasi *addSnapshotListener* sebagai proses untuk mendengarkan atau memperoleh data secara *realtime* dari hasil kueri saat ini atau setiap perubahan kueri yang membuat adanya penambahan, perubahan data di dalam dokumen pada sebuah koleksi.

```
public Observable<User> listenUser(String email) {
    return Observable.create(emitter -> {
        ListenerRegistration listener = this.firestore.collection("users")
                .whereEqualTo("email", email).limit(1)
                .addSnapshotListener((value, error) -> {
                    if (error != null) {
                        emitter.onError(error);
                        return;
                    }

                    if (value != null && !value.isEmpty()) {
                        try {
                            for (DocumentSnapshot snapshot :
value.getDocuments()) {
                                if (snapshot.exists()) {
                                    User user =
Objects.requireNonNull(snapshot.toObject(User.class))
                                            .withId(snapshot.getId());
                                    emitter.onNext(user);
                                }
                            }
                        } catch (Exception e) {
                            emitter.onError(e);
                        }
                    }
                });
        emitter.setCancellable(listener::remove);
    });
}
```

Gambar 4.23 Kode program membaca data identitas masyarakat dari basis data

i. Antarmuka Riwayat

Tampilan antarmuka riwayat dapat muncul dengan memilih *bottom navigation* riwayat pada aplikasi. Tampilan ini akan menampilkan informasi mengenai aktivitas oleh masyarakat setelah melakukan langganan jasa jemput sampah. Informasi yang diberikan akan masuk ke item riwayat pesanan dan sampah. Pada tampilan riwayat, masyarakat akan langsung diarahkan ke item pesanan. Item pesanan akan menampilkan informasi berkaitan dengan data pesanan yang terdiri nama paket sampah, tanggal mulai berlangganan, dan harga paket yang dilanggan serta tampilan ini dapat dilakukan secara *scrolling view*. Masyarakat wajib untuk memesan paket sampah terlebih dahulu agar data pesanan di konfirmasi pengepul sampah sehingga data pesanan akan tersedia di antarmuka ini. Jika tidak ada data pesanan yang muncul, antarmuka tersebut memberikan informasi untuk masyarakat agar melakukan pembelian paket sampah. Tampilan antarmuka akan tampak seperti pada Gambar 4.24.



Gambar 4.24 Antarmuka riwayat pesanan

Potongan kode:

Kode yang terlihat pada Gambar 4.25 menunjukkan proses membaca data paket sampah yang berhasil dipesan dengan berinteraksi dari *method collection* dengan data koleksi, yaitu *subscriptions* dengan memfilter kueri *whereEqualTo* untuk memperoleh *userID* masyarakat, kueri *whereEqualTo* untuk memperoleh data *subscriptions* dengan kolom status paket sampah bernilai *Accepted* yang kemudian diurutkan secara menurun atau data yang paling akhir masuk berdasarkan data *registerAt* dan secara keseluruhan dibatasi hanya menampilkan data paket sampah sebanyak 50 data yang di-*set* kueri *limit.*

```
public Observable<List<Subscription>> listenActiveSubs(String userID) {
    return Observable.create(emitter -> {
        ListenerRegistration listener =
this.firestore.collection("subscriptions")
                .whereEqualTo("userID", userID)
                .whereEqualTo("status", "Accepted")
                .orderBy("registerAt", Query.Direction.DESCENDING)
                .limit(50).addSnapshotListener((value, error) -> {
                    if (error != null) {
                        emitter.onError(error);
                        return;
                    }

                    if (value != null && !value.isEmpty()) {
                        try {
                            List<Subscription> subscriptionList = new
ArrayList<>();
                            for (DocumentSnapshot snapshot :
value.getDocuments()) {
                                if (snapshot.exists()) {
                                    Subscription subscription =
Objects.requireNonNull(snapshot.toObject(Subscription.class))
                                            .withId(snapshot.getId());
                                    subscriptionList.add(subscription);
                                }
                            }
                            emitter.onNext(subscriptionList);
                        } catch (Exception e) {
                            emitter.onError(e);
                        }
                    }
                });
        emitter.setCancellable(listener::remove);
    });
}
```

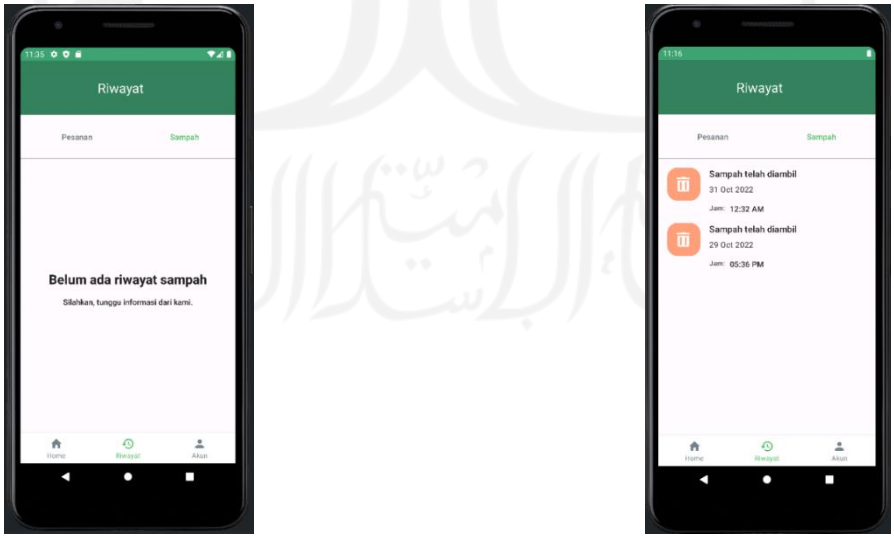
Gambar 4.25 Kode program membaca paket sampah yang berhasil dipesan

Selanjutnya, membuat objek LiveData dengan mengimplementasikan *wrapper* data *subscriptions* berupa *List* sebagaimana yang terlihat pada Gambar 4.26. Objek ini disimpan pada *class ViewModel* sebagai tempat untuk menyimpan, mengelola data yang akan ditampilkan di UI. Dalam objek LiveData bertindak sebagai pemegang data dari *Observable* karena LiveData merupakan *class Observable* yang dapat memancarkan data. Data yang akan dipancarkan dapat berinteraksi pada *class packService* yang memanggil fungsi dari Gambar 4.25 untuk mendapatkan akses data yang dibutuhkan. Setiap objek LiveData yang menangani data dapat diakses bagi masyarakat yang terautentikasi dalam aplikasi dengan mengambil informasi akun melalui objek FirebaseUser. Objek LiveData pada *class ViewModel* nantinya akan dipanggil pengontrol UI, yaitu *fragment* untuk dideklarasikan pada *fragment* itu sendiri sehingga data dapat diterima oleh masyarakat saat mengakses antarmuka riwayat pesanan.

```
public LiveData<List<Subscription>> listSubs() {
    FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
    if (firebaseUser == null) {
        return null;
    }
    Observable<List<Subscription>> subsObservable =
this.packService.listenActiveSubs(firebaseUser.getUid())
                    .observeOn(Schedulers.io());
    return
LiveDataReactiveStreams.fromPublisher(subsObservable.toFlowable(BackpressureStra
tegy.LATEST));
}
```

Gambar 4.26 Kode program menyimpan, mengelola data untuk ditampilkan di UI

Setelah itu, jika masyarakat memilih item riwayat sampah akan menampilkan informasi berupa status sampah, tanggal pengangkutan, dan jam pengangkutan yang diberikan pengepul sampah mengenai status pengangkutan sampah selama satu bulan serta tampilan ini dapat dilakukan secara *scrolling view*. Hal ini sebagai bentuk pelayanan pengepul usai melakukan pengangkutan sehingga memberikan informasi status sampah selama masyarakat masih aktif berlangganan jasa angkut sampah. Tampilan antarmuka akan tampak seperti pada Gambar 4.27.



Gambar 4.27 Antarmuka riwayat sampah

Potongan kode:

Kode yang terlihat pada Gambar 4.28 menunjukkan proses membaca data status hasil pengangkutan sampah yang telah dilakukan pengepul dari *collection picker* dengan memfilter kueri *whereEqualTo* untuk memperoleh *userID* masyarakat karena setiap koleksi pada data *picker* yang telah diberi informasi hasil pengangkutan sampah, masyarakat dapat mengetahui mengenai informasi tersebut. Selanjutnya, data *picker* diurutkan secara menurun atau data yang paling akhir masuk berdasarkan data *createdAt* atau data yang menunjukkan tanggal di saat memberikan informasi hasil status pengangkutan dan secara keseluruhan dibatasi hanya menampilkan data status hasil pengangkutan sebanyak 100 data terbaru yang di-*set* kueri *limit.*

```
public Observable<List<Picker>> listenPicker(String userID) {
    return Observable.create(emitter -> {
        ListenerRegistration listener = this.firestore.collection("picker")
                .whereEqualTo("userID", userID)
                .orderBy("createdAt", Query.Direction.DESCENDING)
.limit(100).addSnapshotListener(((value, error) -> {
                    if (error != null) {
                        emitter.onError(error);
                        return;
                    }

                    if (value != null && !value.isEmpty()) {
                        try {
                            List<Picker> pickerList = new ArrayList<>();
                            for (DocumentSnapshot snapshot :
value.getDocuments()) {
                                if (snapshot.exists()) {
                                    Picker pick =
Objects.requireNonNull(snapshot.toObject(Picker.class))
                                            .withId(snapshot.getId());
                                   pickerList.add(pick);
                                }
                            }
                            emitter.onNext(pickerList);

                        } catch (Exception e) {
                            emitter.onError(e);
                        }
                    }
        }));
        emitter.setCancellable(listener::remove);
    });
}
```

Gambar 4.28 Kode program membaca data status hasil pengangkutan sampah

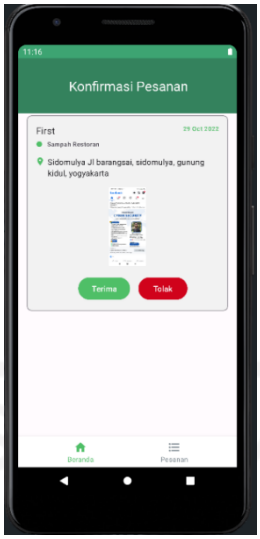
4.1.2 Antarmuka Aplikasi Untuk Pengepul Sampah

a. Antarmuka Konfirmasi Pesanan

Tampilan antarmuka konfirmasi pesanan dapat muncul dengan memilih *bottom navigation* beranda pada aplikasi. Tampilan ini akan menampilkan informasi mengenai permintaan jasa oleh masyarakat terhadap penyedia layanan pengangkutan sampah. Tampilan antarmuka konfirmasi pesanan akan tampak pada Gambar 4.29.

Gambar 4.29 Antarmuka awal konfirmasi pesanan

Selanjutnya, jika masyarakat telah melakukan pemesanan paket sampah, di halaman konfirmasi pesanan akan tertera *form* hasil pesanan milik masyarakat. Setiap data yang akan di konfirmasi terdapat item di dalamnya antara lain nama masyarakat, paket yang dipesan, tanggal, alamat, dan bukti pembayaran yang dilakukan masyarakat serta tampilan ini dapat dilakukan secara *scrolling view*. Data pesanan yang ada di antarmuka ini nantinya dapat diterima maupun ditolak oleh pihak pengepul sampah, kemudian data yang telah diterima akan masuk ke antarmuka daftar pesanan sebagai *list* pesanan yang telah terkonfirmasi dan aktif selama satu bulan. Tampilan antarmuka akan tampak seperti pada Gambar 4.30.



Gambar 4.30 Antarmuka melakukan konfirmasi pesanan

Potongan kode:

Proses konfirmasi pesanan dapat dilakukan dengan membuat fungsi terima pesanan saat memilih *button* terima yang dideklarasikan pada variabel *accept* dengan parameter *subscription* dari *class Subscription* sebagaimana yang terlihat pada Gambar 4.31. Variabel *disposable* dengan fungsi *add* untuk *request* tindakan tambahan lain dengan memanggil fungsi *acceptSubs* dari variabel *subscriptionService* mengenai komunikasi kueri ke basis data sehingga data pesanan yang diterima akan berubah status dari *Pending* ke *Accepted* dan data paket akan masuk ke antarmuka daftar pesanan. Di samping itu, diteruskan dengan memanggil fungsi *concatMap* untuk memancarkan data baru atau proses dari *klik button* terima ke *class Notification* dengan membuat objeknya yang memiliki enam model data dengan *value* masing-masing yang telah ditentukan yang kemudian mengembalikan *value notification* tersebut ke subkoleksi *notification* dalam basis data melalui fungsi *insertNotification.* Pembuatan fungsi terima pesanan juga berlaku untuk proses tolak pesanan. Selanjutnya, fungsi *accept* dan *reject* dipanggil dan ditetapkan pada *method setOnClickListener* dari *class Controller* yang memanggil objek SubPendingModel. Pada SubPendingModel telah di-*set* untuk *id* dari masing-masing *button* dapat dilakukan proses *onClick* sehingga semua *button* dapat dieksekusi yang menghasilkan proses terima atau tolak pesanan.

```
public void accept(Subscription subscription) {
    FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
    Objects.requireNonNull(firebaseUser);

    disposable.add(

subscriptionService.acceptSubs(subscription).observeOn(Schedulers.io())
                    .concatMap(v -> {
                        if (v) {
                            Notification notification = new Notification();
                            notification.createdAt =
Instant.now().toEpochMilli();
                            notification.isRead = false;
                            notification.userID = subscription.userID;
                            notification.payload = subscription.id;
                            notification.type = "SubsAccept";
                            notification.title = "Paket sampah berhasil
dipesan";
                            return
subscriptionService.insertNotification(notification);
                        } else {
                            return Observable.error(new Throwable("gagal
menerima paket"));
                        }
                    }).onErrorComplete().subscribe()
    );
}

orderPendingController.setOnClickListener(new SubPendingModel.OnClickListener()
{
    @Override
    public void onClick(Subscription subscription) {
        mViewModel.accept(subscription);
    }

    @Override
    public void onReject(Subscription subscription) {
        mViewModel.reject(subscription);
    }

});
```

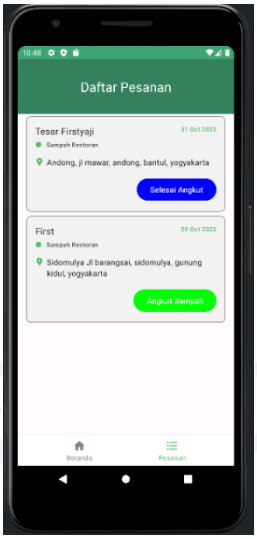
Gambar 4.31 Kode program terima pesanan pada proses konfirmasi pesanan

b. Antarmuka Daftar Pesanan

Tampilan antarmuka daftar pesanan dapat muncul dengan memilih *bottom navigation* pesanan pada aplikasi. Tampilan awal akan menampilkan informasi singkat mengenai daftar pesanan kosong apabila pengepul sampah belum melakukan konfirmasi data pesanan milik masyarakat. Tampilan antarmuka daftar pesanan akan tampak pada Gambar 4.32.

Gambar 4.32 Antarmuka awal daftar pesanan

Selanjutnya, jika pengepul sampah telah melakukan konfirmasi pesanan, pada tampilan daftar pesanan yang semula berisi daftar pesanan kosong akan berubah dengan menampilkan *list* data masyarakat yang telah terkonfirmasi pengepul sampah dengan status data pesanan diterima sehingga masyarakat berhasil berlangganan paket sampah. Setiap data yang masuk terdiri dari beberapa item di antaranya nama masyarakat, paket sampah, alamat, dan tanggal mulainya paket dipesan serta tampilan ini dapat dilakukan secara *scrolling view*. Selain itu, pengepul sampah dapat memberikan informasi mengenai status pengangkutan sampah melalui *button* angkut sampah yang tertera pada tampilan ini. Pada *button* angkut sampah warna hijau menandakan bahwa sampah dari masyarakat segera untuk dilakukan pengangkutan, sedangkan *button* selesai angkut warna biru menandakan bahwa sampah baru saja dilakukan pengangkutan. Tampilan antarmuka akan tampak seperti pada Gambar 4.33.



Gambar 4.33 Antarmuka daftar pesanan

Potongan kode:

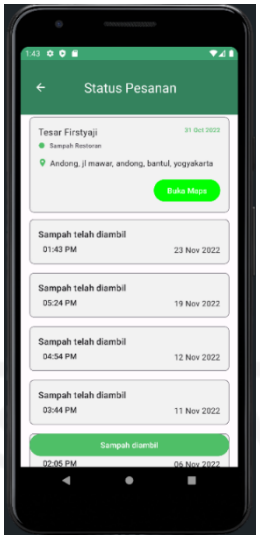
Kode yang terlihat pada Gambar 4.34 sebagai proses membaca paket sampah aktif saat ini dengan mengimplementasikan *wrapper* data *Subscription* berupa *List* yang dideklarasikan pada variabel *listenCurrentActiveSubs*. Dalam variabel tersebut, terdapat variabel *currentTime* untuk mengambil waktu saat ini dari *class Instant.now* yang memanggil fungsi *toEpochMili.* Selanjutnya, membuat *Observable* untuk memancarkan data dengan konsep *stream* data secara asinkronus yang dikelola *emitter* sebagai pengirim data. Data paket sampah aktif diambil dari *collection subscription* dengan memfilter *query whereGreaterThanOrEqualTo* untuk mengambil nilai *validUntil*, di mana *validUntil* memfilter rentang selama satu bulan yang lebih besar sama dengan *currentTime* serta data diurutkan secara menurun atau data yang masuk pada antarmuka daftar pesanan paling terakhir berdasarkan data *validUntil*. Di samping itu, penggunaan *addSnapshotListener* sebagai proses untuk mendengarkan atau memperoleh data secara *realtime* dari hasil kueri saat ini atau setiap perubahan kueri yang membuat adanya penambahan, perubahan data di dalam dokumen pada sebuah koleksi.

```
public Observable<List<Subscription>> listenCurrentActiveSubs() {
    long currentTime = Instant.now().toEpochMilli();
    return Observable.create(emitter -> {
        ListenerRegistration listener =
this.firestore.collection("subscriptions")
                .whereGreaterThanOrEqualTo("validUntil", currentTime)
                .orderBy("validUntil", Query.Direction.DESCENDING)
                .addSnapshotListener((value, error) -> {
                    if (error != null) {
                        emitter.onError(error);
                        return;
                    }

                    if (value != null && !value.isEmpty()) {
                        try {
                            List<Subscription> subscriptionList = new
ArrayList<>();
                            for (DocumentSnapshot snapshot :
value.getDocuments()) {
                                if (snapshot.exists()) {
                                    Subscription subscription =
Objects.requireNonNull(snapshot.toObject(Subscription.class))
                                            .withId(snapshot.getId());


                                    if
(subscription.status.equals("Accepted")) {
                                        subscriptionList.add(subscription);
                                    }
                                }
                            }
                            emitter.onNext(subscriptionList);
                        } catch (Exception e) {
                            emitter.onError(e);
                        }
                    }
        emitter.setCancellable(listener::remove);
    });
}
```

Gambar 4.34 Kode program membaca paket sampah aktif saat ini pada aplikasi pengepul

c. Antarmuka Status Pesanan

Tampilan antarmuka status pesanan menampilkan setiap data pesanan milik masyarakat yang akan diberikan informasi status pengangkutan sampah selama satu bulan dengan *button* yang tersedia pada tampilan ini. Dengan begitu, akan menampilkan data baru yang berisi status sampah, jam, dan tanggal saat pengangkutan sampah telah dilakukan. Data baru yang muncul mengenai status angkut sampah akan masuk pada pesan notifikasi dan item riwayat sampah pada aplikasi masyarakat. Dalam *form* juga terdapat *button* buka *maps* yang membantu pengepul sampah dalam mengetahui titik koordinat alamat masyarakat sehingga dapat melihat rute perjalanan ke alamat masyarakat melalui Google Maps. Tampilan ini dapat dilakukan secara *scrolling view*. Tampilan antarmuka akan tampak seperti pada Gambar 4.35.



Gambar 4.35 Antarmuka status pesanan

Potongan kode:

Kode yang terlihat pada Gambar 4.36 menunjukkan proses menampilkan *maps* melalui tindakan *Intent* ke tampilan yang akan menangani proses *intent* tersebut dengan data berupa *Uri* untuk menetapkan data yang berisi *link* alamat *maps default* dengan menyertakan data *addressDetails* dari *class subscription*, di mana *addressDetails* memiliki *value* dari basis data yang ditambahkan masyarakat berupa *link maps* berdasarkan titik *latitude atau longitude.*

```
pickerController.setSublistener(subscription -> {
    Uri map = Uri.parse("http://maps.google.co.in/maps?q=" +
subscription.addressDetails);
    if (subscription.addressDetails.contains("https")) {
        map = Uri.parse(subscription.addressDetails);
    }

    Intent i = new Intent(Intent.ACTION_VIEW, map); // interaction to maps
    startActivity(i);
});
```

Gambar 4.36 Kode program menampilkan *maps*

4.2 Pengujian Sistem

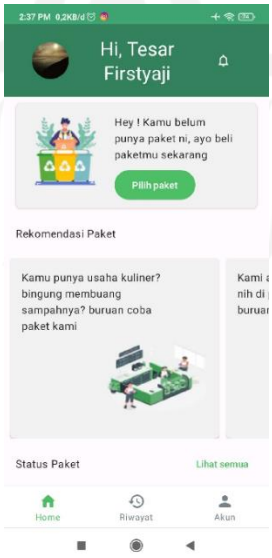
Tahap ini merupakan proses pengujian yang akan dilakukan pada sistem ketika sistem yang dikembangkan telah selesai. Pengujian sistem dilakukan oleh penulis sebagai *hacker* untuk memastikan setiap fungsionalitas pada antarmuka aplikasi dapat berjalan dengan baik. Teknik pengujian sistem akan menggunakan *black box testing*. Proses pengujian ini berfokus

pada segi fungsionalitas sistem yang terdiri dari masukkan, proses, dan keluaran yang dibangun tanpa melakukan pengujian terhadap desain aplikasi maupun kode program. Dengan adanya metode uji *black box*, dapat diketahui hasil luaran fungsi di dalam aplikasi dari pertama kalinya fungsi dijalankan hingga selesai menyelesaikan tugas (*task*). Jika fungsi yang dijalankan menghasilkan luaran yang sesuai dengan data masukkan, fungsi dapat dikatakan berhasil dan bisa digunakan pengguna dengan nyaman. Pengujian sistem pada aplikasi yang dikembangkan terdiri dua aplikasi dengan dua pengguna. Aplikasi yang pertama untuk masyarakat dan aplikasi yang kedua untuk pengepul sampah. Kedua aplikasi tersebut datanya saling berhubungan karena setiap aktivitas yang dilakukan oleh masyarakat akan dikelola oleh pengepul sampah.

Pengujian sistem diujikan pada perangkat *smartphone* berbasis Android dengan tiga *device*, yaitu Xiomi Redmi 8, Xiomi Redmi 6A, dan Asus Zenfone Max yang memiliki spesifikasi antara lain:

1. Xiomi Redmi 8

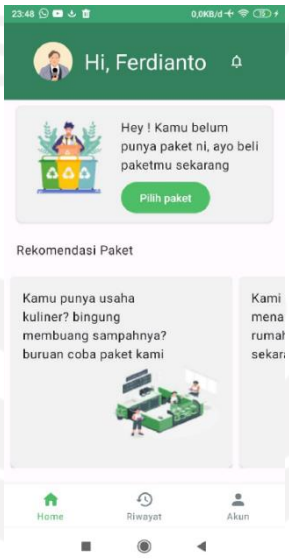
CPU	: Octa-core (4x1.95 GHz Cortex-A53 & 4x1.45 GHz Cortex A53).
GPU	: Adreno 505.
Chipset	: Qualcomm SDM439 Snapdragon 439 (12 nm).
Memory	: 4GB.
Internal Storage	: 64GB.
Operating System	: Android 9.0 (Pie).



Gambar 4.37 Pengujian sistem melalui *smartphone* Xiomi Redmi 8

2. Xiomi Redmi 6A

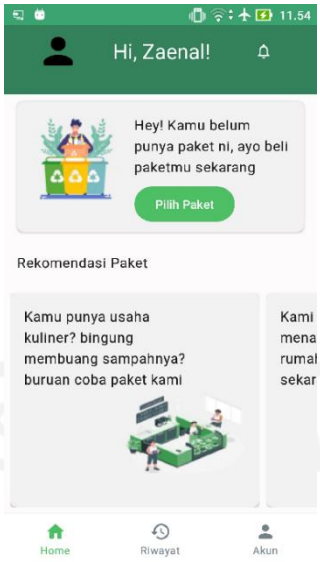
CPU	: Quad-core 2.0 GHz Cortex-A53.
GPU	: PowerVR GE8320.
Chipset	: Mediatek MT6761 Helio A22 (12 nm).
Memory	: 3GB.
Internal Storage	: 32GB.
Operating System	: Android 8.1 (Oreo).



Gambar 4.38 Pengujian sistem melalui *smartphone* Xiomi Redmi 6A

3. Asus Zenfone Max

CPU	: Quad-core 1.2 GHz Cortex-A53.
GPU	: Adreno 306.
Chipset	: Qualcomm MSM8916 Snapdragon 410 (28 nm).
Memory	: 2GB.
Internal Storage	: 16GB.
Operating System	: Android 6.0 (Marshmallow).



Gambar 4.39 Pengujian sistem melalui *smartphone* Asus

4.2.1 Hasil Pengujian Aplikasi Untuk Masyarakat

Hasil pengujian terhadap aplikasi bagi masyarakat telah dilakukan dengan menjalankan fungsi fitur di dalam aplikasi dengan pertama kali diujikan dan memberikan respons luaran yang sesuai dengan spesifikasi kebutuhan sehingga masyarakat nantinya dengan mudah menggunakan aplikasi tersebut dengan lancar tanpa *bug*. Dalam pengujian aplikasi masyarakat terdapat sebelas antarmuka yang berhubungan dengan fitur aplikasi yang telah diujikan pada pengujian ini. Berikut hasil pengujian aplikasi masyarakat dengan metode *blackbox* antara lain:

a. Pengujian Antarmuka *Splash Screen*

Tabel 4.1 Pengujian antarmuka *splash screen*

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	Masyarakat membuka aplikasi	Fungsi akan menampilkan antarmuka awal atau *splash screen* aplikasi selama dua detik	Menampilkan antarmuka *splash screen* selama dua detik	Berhasil

b. Pengujian Antarmuka *Login*

Tabel 4.2 Pengujian antarmuka *login*

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	*Email* yang terhubung dengan akun Google	Fungsi akan melakukan autentikasi *email*. Jika autentikasi dengan *email* belum terdaftar akan menampilkan *default* tampilan *register*. Jika	Dapat melakukan autentikasi *email*. Jika autentikasi dengan *email* belum terdaftar akan menampilkan *default*	Berhasil

		autentikasi dengan *email* terdaftar akan menampilkan *default* tampilan *login* yang kemudian akan menampilkan antarmuka utama aplikasi (antarmuka *home* atau beranda)	tampilan *register*. Jika autentikasi dengan *email* terdaftar akan menampilkan *default* tampilan *login* yang kemudian akan menampilkan antarmuka utama aplikasi (antarmuka *home* atau beranda)	

c. Pengujian Antarmuka Beranda

Tabel 4.3 Pengujian antarmuka beranda

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	*Button* pilih paket sampah	Fungsi akan menampilkan antarmuka informasi jenis paket sampah bulanan beserta jadwal pengangkutan sampah dan harga paket	Menampilkan antarmuka informasi jenis paket sampah bulanan beserta jadwal pengambilan dan harga paket	Berhasil
2	*Button icon* notifikasi yang dapat dipilih	Fungsi akan menampilkan antarmuka notifikasi yang berisi daftar pemberitahuan mengenai aktivitas masyarakat	Dapat menampilkan antarmuka notifikasi yang berisi daftar pemberitahuan mengenai aktivitas masyarakat	Berhasil
3	Menu navigasi *home*	Fungsi akan menampilkan daftar informasi awal akan pemesanan paket sampah dan keadaan paket sampah saat ini.	Dapat menampilkan daftar informasi awal akan pemesanan paket sampah dan keadaan paket sampah saat ini.	Berhasil
4	*Button text view* lihat semua	Fungsi akan menampilkan antarmuka paket yang berisi informasi *list* paket sampah yang tersedia.	Dapat menampilkan antarmuka paket yang berisi informasi *list* paket sampah yang tersedia	Berhasil

d. Pengujian Antarmuka Paket

Tabel 4.4 Pengujian antarmuka paket

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	*Button* beli paket	Fungsi akan menampilkan antarmuka pesanan yang dilengkapi dengan item rincian data pesanan, alamat, metode pembayaran, dan *form* bukti pembayaran	Dapat menampilkan antarmuka pesanan yang dilengkapi dengan item rincian data pesanan, alamat, metode pembayaran,	Berhasil

			dan *form* bukti pembayaran	
2	*Button icon* kembali dipilih	Fungsi akan menampilkan antarmuka *home* atau beranda	Dapat menampilkan antarmuka *home* atau beranda	Berhasil

e. Pengujian Antarmuka Pesanan

Tabel 4.5 Pengujian antarmuka pesanan

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	*Button* tambah alamat	Fungsi akan menampilkan antarmuka alamat dengan item di dalamnya yang berupa nama alamat dan alamat *link* Google Maps	Dapat menampilkan antarmuka alamat dengan item di dalamnya yang berupa nama alamat dan alamat *link* Google Maps	Berhasil
2	*Radio button* metode pembayaran	Fungsi akan menampilkan *radio button* terpilih yang telah dilakukan masyarakat	Menampilkan *radio button* terpilih yang telah dilakukan masyarakat	Berhasil
3	*Form upload* bukti pembayaran	Fungsi akan menampilkan hasil unggah oleh masyarakat berupa gambar bukti pembayaran	Dapat menampilkan hasil *upload* oleh masyarakat berupa gambar bukti pembayaran	Berhasil
4	*Button* konfirmasi	Fungsi akan menampilkan antarmuka pembayaran yang berisi informasi pesanan sedang dilakukan konfirmasi oleh pengepul sampah	Dapat menampilkan antarmuka pembayaran yang berisi informasi pesanan sedang dilakukan konfirmasi oleh pengepul sampah	Berhasil
5	*Button icon* kembali dipilih	Fungsi akan menampilkan antarmuka paket	Dapat menampilkan antarmuka paket	Berhasil

f. Pengujian Antarmuka Alamat

Tabel 4.6 Pengujian antarmuka alamat

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	*Edit text* informasi alamat	Fungsi akan melakukan proses isi data alamat pada *text field* yang tersedia	Dapat melakukan proses isi data alamat pada *text field* yang tersedia	Berhasil
2	*Edit text* informasi *link* Google Maps	Fungsi akan melakukan proses isi data alamat dengan menambahkan *link* alamat dari Google Maps pada *text field* yang tersedia	Dapat melakukan proses isi data alamat dengan menambahkan *link* alamat dari Google	Berhasil

			Maps pada *text field* yang tersedia	
3	*Button* tambah pada menu alamat	Fungsi akan menyimpan data alamat dengan informasi singkat *toast* alamat berhasil ditambahkan dan akan menampilkan antarmuka pesanan kembali setelah mengisi data alamat dengan menampilkan hasil alamat pada *layout text bar*	Dapat menyimpan data alamat dengan informasi singkat *toast* alamat berhasil ditambahkan dan akan menampilkan antarmuka pesanan kembali setelah mengisi data alamat dengan menampilkan hasil alamat pada *layout text bar*	Berhasil
4	*Button icon k*embali dipilih	Fungsi akan menampilkan antarmuka pesanan dengan keadaan data alamat kosong	Dapat menampilkan antarmuka pesanan dengan keadaan data alamat kosong	Berhasil

g. Pengujian Antarmuka Pembayaran

Tabel 4.7 Pengujian antarmuka pembayaran

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	*Button* dashboard	Fungsi akan menampilkan antarmuka utama atau beranda aplikasi setelah melakukan pemesanan paket sampah	Dapat menampilkan antarmuka *home* atau beranda setelah melakukan pemesanan paket sampah	Berhasil

h. Pengujian Antarmuka Notifikasi

Tabel 4.8 Pengujian antarmuka notifikasi

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	*Button k*embali ke atas	Fungsi akan melakukan pemanggilan terhadap data paling atas atau terbaru dengan cepat tanpa penggeseran setiap data secara perlahan	Dapat melakukan pemanggilan terhadap data paling atas atau terbaru dengan cepat tanpa penggeseran setiap data secara perlahan	Berhasil
2	*Button icon k*embali dipilih	Fungsi akan menampilkan antarmuka *home* atau beranda	Dapat menampilkan antarmuka *home* atau beranda	Berhasil

i. Pengujian Antarmuka Profil

Tabel 4.9 Pengujian antarmuka profil

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	Menu navigasi akun	Fungsi akan menampilkan informasi mengenai akun masyarakat berupa edit akun dan informasi akun serta foto profil	Dapat menampilkan informasi mengenai akun masyarakat berupa edit akun dan informasi akun serta foto profil	Berhasil
2	*Button* edit akun dipilih	Fungsi akan menyajikan informasi dengan item yang tersedia di antaranya pembaruan foto profil, nomor telepon, alamat, pekerjaan, tanggal lahir, dan jenis kelamin untuk dilakukan proses edit akun atau penambahan data yang masih kosong	Menampilkan informasi dengan item yang tersedia di antaranya pembaruan foto profil, nomor telepon, alamat, pekerjaan, tanggal lahir, dan jenis kelamin untuk dilakukan proses edit akun atau penambahan data yang masih kosong	Berhasil
3	*Button* informasi akun dipilih	Fungsi akan menampilkan data identitas masyarakat sesuai dengan akun saat *login* dan ada beberapa data yang dapat diperbarui	Dapat menampilkan data identitas masyarakat sesuai dengan akun saat *login* dan ada beberapa data yang dapat diperbarui	Berhasil
4	*Button* keluar dari aplikasi dipilih	Fungsi akan menampilkan antarmuka *login* sebagai tampilan awal setelah masyarakat keluar dari aplikasi	Menampilkan antarmuka *login* sebagai tampilan awal setelah masyarakat keluar dari aplikasi	Berhasil

j. Pengujian Antarmuka Edit Akun

Tabel 4.10 Pengujian antarmuka edit akun

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	*Button upload* foto profil dipilih	Fungsi akan menampilkan *pop-up* pemilihan ganti profil melalui kamera ataupun galeri serta akan melakukan pembaruan foto profil	Dapat menampilkan *pop-up* pemilihan ganti profil melalui kamera ataupun galeri serta melakukan pembaruan foto profil	Berhasil
2	*Edit text* informasi nomor telepon	Fungsi akan melakukan proses isi data nomor	Dapat melakukan proses isi data nomor	Berhasil

		telepon pada *text field* yang tersedia	telepon pada *text field* yang tersedia	
3	*Edit text* informasi alamat masyarakat	Fungsi akan melakukan proses isi data alamat pada *text field* yang tersedia	Dapat melakukan proses isi data alamat pada *text field* yang tersedia	Berhasil
4	*Edit text* informasi pekerjaan	Fungsi akan melakukan proses isi data pekerjaan pada *text field* yang tersedia	Dapat melakukan proses isi data pekerjaan pada *text field* yang tersedia	Berhasil
5	*Edit text* informasi tanggal lahir	Fungsi akan melakukan proses isi data tanggal lahir pada *text field* yang tersedia dengan menggunakan *date picker*	Dapat melakukan proses isi data tanggal lahir pada *text field* yang tersedia dengan menggunakan *date picker*	Berhasil
6	*Edit text* informasi jenis kelamin	Fungsi akan melakukan proses isi data jenis kelamin pada *text field* yang tersedia dengan menggunakan dialog	Dapat melakukan proses isi data jenis kelamin pada *text field* yang tersedia dengan menggunakan dialog	Berhasil
7	*Button* simpan dipilih	Fungsi akan melakukan penyimpanan data yang telah diisi dan akan menampilkan data pada informasi akun serta akan menampilkan *toast* data berhasil diedit	Melakukan penyimpanan data yang telah diisi dan menampilkan data pada informasi akun serta menampilkan *toast* data berhasil diedit	Berhasil
8	*Button icon k*embali dipilih	Fungsi akan menampilkan kembali antarmuka profil	Dapat menampilkan kembali antarmuka profil	Berhasil

k. Pengujian Antarmuka Riwayat

Tabel 4.11 Pengujian antarmuka riwayat

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	Menu navigasi riwayat	Fungsi akan menampilkan antarmuka riwayat disertai dengan item riwayat pesanan dan riwayat sampah	Dapat menampilkan antarmuka riwayat disertai dengan item riwayat pesanan dan riwayat sampah	Berhasil
2	*Button* item riwayat pesanan dipilih	Fungsi akan menyajikan informasi mengenai data pesanan yang sudah dipesan oleh masyarakat dan di konfirmasi oleh pengepul sampah	Dapat menyajikan informasi mengenai data pesanan yang sudah dipesan oleh masyarakat dan di konfirmasi oleh pengepul sampah	Berhasil

3	*Button* item riwayat sampah dipilih	Fungsi akan menyajikan informasi mengenai data status pengangkutan sampah	Dapat menyajikan informasi mengenai data status pengangkutan sampah	Berhasil

Pengujian dilakukan di setiap proses masukkan dan keluaran dalam antarmuka yang berkaitan dengan fitur yang dikembangkan pada aplikasi untuk menentukan apakah sistem tersebut sesuai dengan spesifikasi yang dibutuhkan. Sebuah fungsi dikatakan berhasil apabila data luaran yang dihasilkan sesuai dengan fungsi yang diharapkan berdasarkan data masukkan yang dilakukan, seperti masyarakat dengan fungsi masukkan *button* beli paket maka fungsi yang diharapkan akan menampilkan antarmuka pesanan yang dilengkapi dengan item-item pada antarmuka tersebut sehingga data luaran pada fungsi tersebut harus menampilkan data fungsi yang diharapkan. Jika hal tersebut sesuai, dapat disimpulkan fungsi berhasil dan tidak ada *bug* atau kesalahan kode program.

4.2.2 Hasil Pengujian Aplikasi Untuk Pengepul Sampah

Hasil pengujian terhadap aplikasi bagi pengepul sampah telah dilakukan dengan menjalankan fungsi fitur di dalam aplikasi dengan pertama kali diujikan dan memberikan respons luaran yang sesuai dengan spesifikasi kebutuhan. Dalam pengujian aplikasi pengepul sampah terdapat tiga antarmuka terkait fitur aplikasi yang telah diujikan pada pengujian ini. Berikut hasil pengujian aplikasi pengepul sampah dengan metode *blackbox* antara lain:

a. Pengujian Antarmuka Daftar Pesanan

Tabel 4.12 Pengujian antarmuka daftar pesanan

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	Menu navigasi pesanan	Fungsi akan menampilkan daftar data pesanan yang aktif setelah terkonfirmasi oleh pengepul sampah	Dapat menampilkan daftar data pesanan yang aktif setelah terkonfirmasi oleh pengepul sampah	Berhasil
2	*Button* angkut sampah dipilih	Fungsi akan menyajikan data pesanan yang dipilih untuk dilakukan angkut sampah pada antarmuka status pesanan	Dapat menyajikan data pesanan yang dipilih untuk dilakukan angkut sampah pada antarmuka status pesanan	Berhasil
3	*View button* selesai angkut	Fungsi akan mengetahui data pesanan yang telah	Dapat mengetahui data pesanan yang telah dilakukan	Berhasil

		dilakukan pengangkutan sampah pada saat itu juga	pengangkutan sampah pada saat itu juga	

b. Pengujian Antarmuka Konfirmasi Pesanan

Tabel 4.13 Pengujian antarmuka konfirmasi pesanan

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	Menu navigasi beranda	Fungsi akan menampilkan permintaan daftar pesanan disertai dengan bukti pembayaran yang telah dilakukan masyarakat	Menampilkan permintaan daftar pesanan disertai dengan bukti pembayaran yang telah dilakukan masyarakat	Berhasil
2	*Button* terima dipilih	Fungsi akan melakukan terima pesanan milik masyarakat sebagai persetujuan untuk dilakukan pengangkutan sampah	Dapat melakukan terima pesanan milik masyarakat sebagai persetujuan untuk dilakukan pengangkutan sampah	Berhasil
3	*Button* tolak dipilih	Fungsi akan melakukan tolak pesanan milik masyarakat sebagai penolakan untuk dilakukan pengangkutan sampah apabila data yang dipesan tidak sesuai	Dapat melakukan tolak pesanan milik masyarakat sebagai penolakan untuk dilakukan pengangkutan sampah apabila data yang dipesan tidak benar	Berhasil

n. Pengujian Antarmuka Status Pesanan

Tabel 4.14 Pengujian antarmuka status pesanan

No	Data Masukkan	Fungsi Yang Diharapkan	Data Keluaran	Kesimpulan
1	*Button* sampah diambil pada menu status pesanan	Fungsi akan menampilkan informasi mengenai data status sampah, jika sampah telah diangkut oleh pengepul sampah pada saat hari pengangkutan	Dapat menampilkan informasi mengenai data status sampah, jika sampah telah diangkut oleh pengepul sampah pada saat hari pengangkutan	Berhasil
2	*Button* buka *maps* dipilih	Fungsi akan menampilkan titik koordinat alamat milik masyarakat di Google Maps	Dapat menampilkan titik koordinat alamat milik masyarakat di Google Maps	Berhasil

3	*Button icon* kembali dipilih	Fungsi akan menampilkan antarmuka daftar pesanan	Dapat menampilkan antarmuka daftar pesanan	Berhasil

Hasil pengujian dilakukan pada setiap proses masukkan dan keluaran antarmuka yang terkait dengan fitur yang dikembangkan dalam aplikasi untuk menentukan apakah sistem tersebut memenuhi spesifikasi yang dibutuhkan. Suatu fungsi dikatakan berhasil jika data keluaran yang dihasilkan sesuai dengan fungsi yang diharapkan berdasarkan data masukkan yang dilakukan, seperti pengepul sampah dengan fungsi masukkan *button* terima maka fungsi yang diharapkan akan melakukan terima data pesanan dari masyarakat sebagai persetujuan untuk dilakukan pengangkutan sampah sehingga data keluaran fungsi tersebut harus menampilkan data fungsi yang diharapkan, jika fungsi sesuai dapat disimpulkan bahwa fungsi berhasil tanpa ada program eror yang menyebabkan kesalahan dalam eksekusi sistem perangkat lunak.

BAB V
PENUTUP

5.1 Kesimpulan

Dari tahapan dalam pengembangan sistem mulai dari analisis hingga pengujian sistem yang telah dilakukan untuk mengatasi masalah pada sampah maka dikembangkan sebuah sistem pengangkutan sampah. Berdasarkan tahapan yang telah dibahas dapat diambil kesimpulan sebagai berikut:

1. Proses pengembangan sistem pengangkutan sampah berhasil dikembangkan melalui metode *prototyping* yang terdiri dari lima tahap.
2. Tahap pertama, melakukan analisis kebutuhan untuk memenuhi kebutuhan fungsionalitas yang meliputi kebutuhan input, proses, dan *output* untuk sistem yang dikembangkan dan menganalisis peran dari setiap pengguna serta menentukan fitur-fitur pada sistem di antaranya aplikasi masyarakat yang terdiri fitur *login*, pemesanan paket sampah, notifikasi, pembayaran secara transfer, riwayat pesanan dan sampah serta akun profil masyarakat. Selanjutnya, aplikasi pengepul sampah yang terdiri fitur *login,* konfirmasi pesanan, melihat daftar pesanan, dan fitur status pesanan mengenai hasil pengangkutan sampah yang telah dilakukan selama periode satu bulan. Selain itu, analisis proses bisnis untuk menganalisis kelebihan sistem terhadap kompetitor dan memproyeksikan keuntungan sistem yang akan diterima dari pengguna.
3. Tahap kedua dan ketiga, membangun desain sistem dan *prototype* untuk membuat perancangan desain sistem dan tampilan aplikasi untuk masyarakat dan pengepul sampah. Selanjutnya, pengujian dan evaluasi *prototype* dilakukan dua kali iterasi, iterasi yang pertama pada pengujian *prototype* masih ditemukan *feedback* dari pengguna dan tim pengembang melakukan evaluasi. Setelah dievaluasi, dilakukan iterasi yang kedua pada pengujian *prototype* dengan hasil pengujian tidak terdapat masukkan dari pengguna sehingga iterasi selesai dilakukan dan rancangan desain berhasil diterima oleh pengguna.
4. Tahap keempat, pengembangan sistem. Sistem pengangkutan sampah merupakan sebuah aplikasi berbasis Android yang berhasil dikembangkan menggunakan bahasa pemrograman Java melalui *software* Android Studio. Sistem pengangkutan sampah terdiri dua aplikasi di antaranya aplikasi untuk masyarakat sebagai *end-user* dan aplikasi untuk admin sebagai pengepul sampah.

5. Tahap kelima, pengujian sistem telah dilakukan terhadap fitur-fitur dari aplikasi masyarakat dan aplikasi pengepul sampah menggunakan metode *blackbox* dengan hasil setiap fitur dapat berjalan dengan baik.

5.2 Saran

Aplikasi yang telah dikembangkan masih banyak kekurangan dan jauh dari kesempurnaan maka perlu ditingkatkan dan dilakukan pengembangan lebih lanjut dengan kemampuan sumber daya yang lebih besar dan luas dalam membangun sebuah sistem. Dalam aplikasi Bersih Kotaku diperlukan beberapa fitur yang bisa dikembangkan agar lebih memudahkan pengguna secara efisien di antara lain:

1. Mengembangkan fitur berbasis teknologi GPS dan *maps* sebagai proses tambah alamat dengan titik koordinat yang akurat.
2. Mengembangkan fitur komunikasi antara masyarakat dengan pengepul di dalam aplikasi Bersih Kotaku.
3. Dapat menambahkan integrasi pembayaran digital yang lebih luas dan mengembangkan aplikasi Bersih Kotaku yang dapat diakses oleh perangkat lain, seperti platform web atau iOS sehingga pengguna pada aplikasi lebih luas.

DAFTAR PUSTAKA

Akif, M., Prasetyo, Y. A., Ambarsari, N., Telekomunikasi, J., & Buah, T. (2015). *PENGEMBANGAN APLIKASI E-CRM BOJANA*. *2*(1), 1057–1070.

Anwar, R. S. (2019). Rancang Bangun Aplikasi File Materi Perkuliahan Di Akademi Telkom Jakarta Berbasis Android Menggunakan Android Studio. *EJurbal "Mahasiswa" Informatika Dan Telekomunikasi*, *1*(1), 1–5.

Apriliah, W., Subekti, N., & Haryati, T. (2021). Penerapan Model Waterfall Dalam Perancangan Aplikasi Sistem Informasi Simpan Pinjam Pada Koperasi Pt. Chiyoda Integre Indonesia Karawang. *Jurnal Interkom: Jurnal Publikasi Ilmiah Bidang Teknologi Informasi Dan Komunikasi*, *14*(2), 34–42. https://doi.org/10.35969/interkom.v14i2.69

Badan Pusat Statistik. (2021). *Jumlah penduduk daerah Yogyakarta*. Yogyakarta.Bps.Go.Id. https://yogyakarta.bps.go.id/indicator/12/133/2/proyeksi-jumlah-penduduk-menurut-kabupaten-kota-di-d-i-yogyakarta-.html

Bsi university. (2022). *Mengenal Metode Prototype*. Bsi.Today. https://bsi.today/metode-prototype/

Dwi, D., Adam, H., Nisa, H., Hasan, R. A., Nurazzahra, R. A., Azmi, U., Wardani, S., & Pamulang, U. (n.d.). *Mewujudkan Indonesia Bersih Dan Bebas Sampah Melalui*. *2*(1), 25–35.

Goleman et al. (2019). Hubungan Tingkat Pengetahuan Dan Sikap Dengan Perilaku Ibu PKK Dalam Pengelolaan Sampah Di Dusun Mengwitani Kecamatanh Mengwitani Kabupaten Badung. *Journal of Chemical Information and Modeling*, *53*(9), 1689–1699.

Ika Purwanti. (2013). PERANCANGAN APLIKASI PEMBELAJARAN HURUF HIJAIYAH BERPLATFORM ANDROID UNTUK MADRASAH BACA TULIS AL QURAN AL-FATTAH DESA WIDODAREN KABUPATEN NGAWI Ika Purwanti. *Seminar Riset Unggulan Nasional Informatika Dan Komputer FTI UNSA 2013*, *2*(1), 123–130.

James Tamplin, A. L. (n.d.-a). *Firebase Authentication*. Retrieved May 25, 2022, from https://firebase.google.com/docs/auth?hl=id

James Tamplin, A. L. (n.d.-b). *Firebase Firestore Database*. Retrieved May 24, 2022, from https://firebase.google.com/docs/firestore?hl=id

Jimmy Ramadhan Azhari. (2021). *Masalah sampah di Indonesia*. Kompas.Com. https://www.kompas.com/sains/read/2021/10/29/130000623/masalah-sampah-indonesia-ancam-target-nol-emisi-kok-bisa-?page=all

Nagib, C. (2014). Pengembangan Android. *Pengembangan Android*, 6–34.

Paramitha, A. (2020). *Diagram Use Case*. 1–12. https://repository.unikom.ac.id/63829/1/Materi 3 - Usecase Diagram_.pdf

Prasetyo, R., Marliana, I., Brajannoto, D. P., Pd, M., & Raharto, T. B. (n.d.). *Aplikasi Cloud Storage Menggunakan Firebase*. 1–6.

Rakhmad, A. N. (2002). *tentang Tata Cara Pengelolaan Teknik Sampah Perkotaan*. https://d1wqtxts1xzle7.cloudfront.net/31007687/diktatsampah-2010-bag-1-3-with-cover-page.pdf?Expires=1621838546&Signature=IpHveiL-ddb8ozeC9NE9DbzEa-zs1B6ZituRwnwzUCWWyRZnRmuiFD8lyzTPsQv8TgRuTf7o~VEmXDXR66yRzI7Yvj3hjlt2Mc6FGRLCpgF72QaO9XOGlKZwU7ZfWA-R1YaIi

Shadiq, J., Safei, A., & Loly, R. W. R. (2021). Pengujian Aplikasi Peminjaman Kendaraan Operasional Kantor Menggunakan BlackBox Testing. *INFORMATION MANAGEMENT FOR EDUCATORS AND PROFESSIONALS : Journal of Information Management*, *5*(2), 97. https://doi.org/10.51211/imbi.v5i2.1561

Tutuko, P. (2008). *Mengelola Sampah*. *2*(18), 1–14. https://doi.org/10.13140/RG.2.1.3996.3043

LAMPIRAN