PERANCANGAN APLIKASI JASA TRANSPORTASI *ONLINE* PADA KOTA MEDAN MENGGUNAKAN *FIREBASE* DAN ALGORITMA *DIJKSTRA* BERBASIS *MOBILE*

SKRIPSI

ABDUL ALFATTAH HIDAYAH

0702162039

PROGRAM STUDI SISTEM INFORMASI

FAKULTASI SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUMATERA UTARA

MEDAN

2021 M / 1443

PERANCANGAN APLIKASI JASA TRANSPORTASI *ONLINE* PADA KOTA MEDAN MENGGUNAKAN *FIREBASE* DAN ALGORITMA *DIJKSTRA* BERBASIS *MOBILE*

SKRIPSI

ABDUL ALFATTAH HIDAYAH

0702162039

PROGRAM STUDI SISTEM INFORMASI

FAKULTASI SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUMATERA UTARA

MEDAN

2021 M / 1443 H

PERSETUJUAN SKRIPSI

Hal : Surat Persetujuan Skripsi
Lamp : -

Kepada Yth.
Dekan Fakultas Sains dan Teknologi
Universitas Islam Negeri Medan Sumatera Utara Medan

Assalamualaikum Wr, Wb.
Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan, maka kami selaku pembimbing berpendapat bahwa skripsi saudara:

Nama	:	Abdul Alfattah Hidayah
Nomor Induk Mahasiswa	:	0702162039
Program Studi	:	Sistem Informasi
Judul	:	Perancangan Aplikasi Jasa Transportasi *Online* Pada Kota Medan Menggunakan *Firebase* Dan Algoritma *Dijkstra* Berbasis *Mobile*

Dapat disetujui untuk segera di*munaqasyahkan*. Atas perhatiannya kami ucapkan terimakasih.

Medan, 26 Agustus 2021 M
17 Muharram 1443 H

Komisi Pembimbing

Pembimbing I

Samsudin, S.T, M.Kom
NIP. 197612272011011002

Pembimbing II

Triase, S.T, M.Kom
NIB 1100000122

SURAT PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan dibawah ini:

Nama	:	Abdul Alfattah Hidayah
Nomor Induk Mahasiswa	:	0702162039
Program Studi	:	Sistem Informasi
Judul	:	Perancangan Aplikasi Jasa Transportasi *Online* Pada Kota Medan Menggunakan *Firebase* Dan Algoritma *Dijkstra* Berbasis *Mobile*

Menyatakan bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing disebutkan sumbernya. Apabila dikemudian hari ditemukan plagiat dalam skripsi ini saya bersedia menerima sanksi pencabutan gelar akademik yang saya peroleh dari sanksi lainnya sesuai dengan peraturan yang berlaku.

Medan, 18 Agustus 2021

Abdul Alfattah Hidayah

NIM 0702162039

KEMENTERIAN AGAMA REPUBLIK INDONESIA
UNIVERSITAS ISLAM NEGERI SUMATERA UTARA MEDAN
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Lapangan Golf No.120, Kp. Tengah, Kode Pos 20353
Telp. (061) 6615683-6622925, Fax. (061) 6615683
Url: http://saintek.uinsu.ac.id, E-mail: saintek@uinsu.ac.id

PENGESAHAN SKRIPSI
Nomor: B.155/ST/ST.V.2/PP.01.1/09/2021

Judul	:	Perancangan Aplikasi Jasa Transportasi *Online* Pada Kota Medan Menggunakan *Firebase* Dan Algoritma *Dijkstra* Berbasis *Mobile*
Nama	:	Abdul Alfattah Hidayah
Nomor Induk Mahasiswa	:	0702162039
Program Studi	:	Sistem Informasi
Fakultas	:	Sains dan Teknologi

Telah dipertahankan dihadapan Dewan Penguji Skripsi Program Studi Sistem Informasi Fakultas Sains dan Teknologi Universitas Islam Negeri Sumatera Utara Medan dan dinyatakan **LULUS**.

Pada hari/tanggal	:	Selasa, 07 September 2021
Tempat	:	Online

Tim Ujian Munaqasyah,
Ketua,

Samsudin, ST.M.Kom
NIP. 197612272011011002

Dewan Penguji,

Penguji I,

Suendri, M.Kom
NIP. 198712082015031003

Penguji II,

Muhammad Dedi Irawan, M.Kom
NIP. 199001312019031019

Penguji III,

Samsudin, ST, M.Kom
NIP. 197612272011011002

Penguji IV,

Triase, S.T, M.Kom
NIB. 1100000122

Mengesahkan,
Dekan Fakultas Sains dan Teknologi
UIN Sumatera Utara Medan,

Dr. Mhd. Syahnan, MA
NIP. 196609031991031002

MOTTO DAN PERSEMBAHAN

MOTTO

"..Allah tidak membebani seseorang itu melainkan sesuai dengan kesanggupannya.."

(QS. Al – Baqarah:286)

"..Dan sesungguhnya telah Kami muliakan anak-anak Adam, Kami angkat mereka di daratan dan di lautan, Kami beri mereka rezeki dari yang baik-baik dan Kami lebihkan dengan kelebihan yang sempurna atas kebanyakan makhluk yang telah Kami ciptakan.."

(QS. Al-Isra ayat 70)

PERSEMBAHAN

Dengan mengucapkan syukur Alhamdulillah, saya persembahkan karya tulis ini untuk orang-orang yang saya cintai:

1. Ibu, dan alm. Ayah saya tercinta yang telah mengisi dunia saya dengan begitu banyak kebahagiaan sehingga seumur hidup tidak cukup untuk menikmati semuanya. Terima kasih atas semua cinta yang telah ayah dan ibu berikan kepada saya. Saya bahkan tidak bisa menjelaskan betapa bersyukurnya saya memiliki kalian dalam hidup saya.
2. Keluarga saya tercinta yang telah memberikan kasih sayang, doa dan dukungannya kepada saya.
3. Sahabat dan teman-teman yang selalu ada disisi saya. Tanpa inspirasi, dorongan, dan dukungan yang telah kalian berikan kepada saya, saya mungkin bukan apa-apa saat ini.

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi. Wabarakatuh.

Alhamdulillahirabbil'alamiin. Segala puji bagi Allah atas segala rahmat dan karunia-Nya serta shalawat beriring salam disampaikan kepada baginda besar kita Nabi Muhammad SAW, sehingga penulis dapat menyelesaikan Skripsi ini dalam rangka memenuhi salah satu syarat untuk menyelesaikan program studi strata-1 pada Universitas Islam Negeri Sumatera Utara (UINSU) Fakultas Sains dan Teknologi.

Skripsi ini dibuat berdasarkan penelitian dan studi pustaka yang telah penulis lakukan. Selama penyusunan Skripsi ini Penulis mendapatkan doa restu dan bantuan dari berbagai pihak, terutama dari orang tua dan juga mendapatkan bimbingan dari berbagai pihak. Selain itu pada kesempatan ini Penulis ingin mengucapkan terima kasih kepada:

1. Bapak Prof. Dr. Syahrin Harahap, M.A selaku Rektor Universitas Islam Negeri Sumatera Utara
2. Bapak Dr. Mhd. Syahnan, M.A selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Sumatera Utara.
3. Bapak Samsudin, S.T, M.Kom selaku Ketua Program Studi Sistem Informasi Universitas Islam Negeri Sumatera serta dosen pembimbing I penulis, dan juga selaku dosen pembimbing akademik, yang telah membantu dalam memberikan arahan dan masukan kepada penulis selama penulisan Skripsi ini.
4. Bapak Suendri, M.Kom selaku Sekretaris Program Studi Sistem Informasi Fakultas Sains dan Teknologi pada Universitas Islam Negeri Sumatera Utara.
5. Ibu Triase, S.T, M.Kom Selaku dosen pembimbing II penulis, yang telah membantu dalam memberikan arahan dan masukan kepada penulis selama penulisan Skripsi ini.

6. Seluruh dosen Program Studi Sistem Informasi yang telah banyak memberikan arahan dan masukan kepada penulis
7. Para teman-teman seperjuangan yang banyak memberikan *support* kepada penulis, sehingga penulis tetap semangat dalam menyelesaikan laporan ini.

Semoga Allah SWT selalu memberikan hidayah serta kesehatan kepada kita semua, dan nantinya semoga skripsi ini nantinya dapat bermanfaat kepada Universitas Islam Negeri Sumatera Utara, Kota Medan, pembaca, serta bagi banyak masyarakat luas. Akhir kata penulis mengucapkan terima kasih.

Medan, 14 Januari 2021

Penulis,

Abdul Alfattah Hidayah

Nim. 0702162039

PERANCANGAN APLIKASI JASA TRANSPORTASI *ONLINE* PADA KOTA MEDAN MENGGUNAKAN *FIREBASE* DAN ALGORITMA *DIJKSTRA* BERBASIS *MOBILE*

ABSTRAK

Aplikasi transportasi *online* sudah menjadi daya tarik tersendiri untuk para pengusaha yang ingin memulai mencari peruntungan pada bidang teknologi dan transportasi, mengingat peluang yang cukup besar dimana jumlah populasi manusia yang semakin banyak, dan beberapa kesibukan manusia yang harus dipenuhi. Beberapa perusahaan yang ingin mencari peruntungan dalam bidang transportasi dan teknologi yang pada sistemnya masih terdapat beberapa masalah dan membutuhkan pengembangan. Untuk memecahkan permasalahan yang ada, maka dibutuhkan sebuah sistem dan juga algoritma yang dapat memecahkan masalah dan memudahkan pengembangan seperti *firebase* dan juga algoritma *Dijkstra*. *Firebase* merupakan sebuah layanan *cloud computing* dari *Google* yang bertujuan untuk memudahkan dalam proses pengembangan aplikasi agar lebih efisien dan dapat memangkas biaya operasional serta waktu pengerjaannya. Dalam melakukan integrasi antara aplikasi *android* dan *firebase* sendiri, seorang pengembang dapat memanfaatkan SDK yang sudah disediakan didalam *firebase*. Algoritma *dijkstra* merupakan sebuah algoritma yang digunakan untuk menyelesaikan persoalan pencarian rute terpendek ataupun lintasan terpendek dari satu *vertex* ke *vertex* yang lainnya pada suatu *graf* yang berbobot, jarak antara *vertex* adalah nilai bobot dari setiap *edge* pada *graph*. Suatu *graf* yang mempunyai bobot, harus mempunyai nilai yang positif (bobot >= 0). Algoritma *Dijkstra* sendiri menggunakan strategi *greedy* dalam pengerjaannya, dimana pada setiap langkah dipilih sisi dengan bobot terkecil yang menghubungkan sebuah simpul yang sudah terpilih dengan simpul lain yang belum terpilih. Tugas akhir ini bertujuan sebagai pengembangan sebuah aplikasi transportasi *online* dengan menggunakan *firebase* sebagai *database*-nya dan juga algoritma *Dijkstra* untuk menentukan rute terpendek pada sebuah proses orderan.

Kata Kunci: Transportasi *online*, *Firebase*, Algoritma *dijkstra*

DESIGN AND BUILD ONLINE TRANSPORTATION SERVICES APPLICATION FOR MEDAN CITY USING FIREBASE AND DIJKSTRA ALGORITHM MOBILE-BASED

ABSTRACT

Online transportation applications have become a special attraction for entrepreneurs who want to start looking for their fortune in the field of technology and transportation, considering the large opportunities where the number of human populations is increasing, and several human activities must be fulfilled. Some companies that want to make profits in the field of transportation and technology in their systems still have some problems and development needs. To solve existing problems, a system and algorithm are needed that can solve problems and facilitate development such as firebase and Dijkstra's algorithm. Firebase is a cloud computing service from Google that aims to make the application development process more efficient and reduce operational costs and processing time. In integrating the android application and firebase itself, a developer can take advantage of the SDK that has been provided in firebase. The dijkstra algorithm is an algorithm used to solve the problem of finding the shortest route or the shortest path from one vertex to another in a weighted graph, the distance between vertices is the weight value of each edge in the graph. A graph that has a weight, must have a positive value (weight >= 0). Dijkstra's algorithm itself uses a greedy strategy in its operation, where at each step the side with the smallest weight is selected that connects a node that has been selected with another node that has not been selected. This final project aims to develop an online transportation application using firebase as its database and also Dijkstra's algorithm to determine the shortest route in an order process.

Keywords: Online transportation, Firebase, dijkstra Algorithm

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Di era *modern* saat ini, manusia sudah banyak melakukan revolusi ilmiah, baik dalam bidang fisika, astronomi, kimia, biologi, teknologi, dan beberapa ilmu pengetahuan lain. Salah satu bidang ilmiah yang mempunyai perkembangan sangat pesat adalah pada bidang teknologi. Teknologi sudah menjadi kebutuhan primer pada manusia saat ini, salah satunya ada pemanfaatan teknologi pada bidang angkutan dan transportasi yaitu ojol (ojek *online*) yang dapat mengantarkan penumpang kemana saja dan kapan saja. Aplikasi ojol ini pun sudah menjadi daya tarik tersendiri untuk para pengusaha yang ingin memulai mencari peruntungan pada bidang teknologi dan transportasi, mengingat peluang yang cukup besar dimana jumlah populasi manusia yang semakin banyak, dan beberapa kesibukan manusia yang harus dipenuhi. Aplikasi ojol ini pun tentunya akan dapat digunakan dalam beberapa macam aktifitas manusia sehari-hari seperti mengantar ke kantor, sekolah, pengantaran paket. Bahkan menurut (Anindhita dkk., 2016) beberapa *Startup* yang besar sudah memiliki beberapa fitur keren lainnya seperti layanan pemesanan dan pengantaran makanan secara *online*, layanan belanja kehidupan sehari-hari, dan masih banyak lagi.

Kota Medan adalah ibu kota Provinsi Sumatera Utara yang merupakan kota terbesar keempat di Indonesia setelah DKI Jakarta, Surabaya, dan Bandung. Namun masih sedikit putra-putra Kota Medan yang dapat mengasah dan menyalurkan bakatnya dalam bidang IT. Pada Kota Medan sendiri masih sedikit pula putra-putra daerah yang berhasil membuat aplikasi jasa transportasi *online* yang sedang maraknya ada di Indonesia. Sehingga masih sedikit aplikasi jasa transportasi *online* yang berasal dari kota Medan yang dapat bersaing dengan aplikasi-aplikasi jasa transportasi *online* lainnya yang ada di Indonesia. Tidak sedikit pula perusahaan yang ingin bersaing dalam mencari peruntungan dalam bidang transportasi dan teknologi ini. Namun, ada beberapa perusahaan yang mempunyai masalah dalam mengembangkan aplikasi ini, dimana masih

banyaknya *bug* pada sistem yang mengakibatkan sulitnya terjadi transaksi antara pelanggan dan *driver* sendiri. Adapun *bug* yang terdapat pada beberapa perusahaan jasa transportasi *online* di Kota Medan adalah, tidak *realtime* nya jarak antara *customer* dengan *driver*, lokasi antara *customer* dan *driver* yang salah sehingga membuat para *customer* kesulitan dalam melakukan pemesanan, serta status keaktifan *driver*, dan juga perpindahan *icon driver* pada map disetiap detik dan jaraknya pada saat proses penjemputan dan pengantaran *customer*. Beberapa perusahaan yang ingin bersaing dalam mencari peruntungan dalam bidang jasa transportasi dan teknologi inipun menggunakan fitur berbayar yang disediakan oleh *google* untuk mencari rute terpendek dan menghitung jarak pada setiap orderan yang berlangsung, sehingga perusahaan harus melakukan pengeluaran dana untuk membayar fitur tersebut dan membuat turunnya kreativitas putra-putra daerah dalam mengembangkan fitur pencarian rute terpendek secara mandiri.

Aplikasi yang akan dibangun nanti akan dibangun dengan *Android Studio* yang akan digunakan untuk membangun aplikasi *driver* dan *customer*, *Firebase* sebagai sinkronisasi data secara *realtime*, dan juga Algoritma *Dijkstra* untuk mencari rute terpendek pada sebuah orderan. *Android Studio* adalah sebuah pengembangan dari *Eclipse*, dan juga merupakan sebuah *IDE (Integrated Development Environtment)* atau merupakan program komputer yang didalamnya menyediakan berbagai utilitas yang diperlukan dalam membentuk sebuah perangkat lunak. *Android Studio* juga merupakan sebuah aplikasi yang banyak digunakan dalam pengembangan aplikasi *android* (Marisa & Wijaya, 2016). *Firebase* merupakan sebuah *platform* dan layanan dari Google dalam mempermudah mengembangkan sebuah aplikasi yang dapat mengakses *database* secara *realtime* (Sanad dkk., 2018). Algoritma *Dijkstra* merupakan salah satu algoritma yang banyak digunakan dan sangat popular dalam menyelesaikan sebuah permasalahan dalam pencarian jalur terpendek (Triase & Aprilia, 2020). Adapun aplikasi yang akan dibangun nantinya mempunyai target minimum *SDK* untuk versi *android* minimum yang dapat menjalankan aplikasi yang akan dibuat penulis adalah *Android* dengan versi *Android* 4.1 *(Jelly Bean) API* 16. Menurut (Kusniyati & Sitanggang, 2016) *Android* versi *JellyBean* merupakan sebuah nama

kode yang diberikan kepada versi kesepuluh dari *Android* sistem operasi selular yang dikembangkan oleh *Google*, yang mencakup tiga poin utama (versi 4.1 hingga 4.3.1)

Pada penelitian terdahulu yang menjadi rujukan penulis yaitu, menurut Iwan Pahendra dengan judul "PEMBUATAN APLIKASI OJEK ONLINE UNTUK MASYARAKAT SEPUTAR KAMPUS UNSRI INDRALAYA". Pada penelitian ini penulis mencoba membangun dan mengimplementasikan aplikasi jasa transportasi *online* atau biasa disebut ojek *online*, dimana aplikasi ini dapat menyelesaikan permasalahan transportasi kampus dan juga akan dimanfaatkan di seputaran kampus UNSRI Indralaya yang sangat memerlukan jasa transportasi *online* tersebut, dikarenakan masih banyak layanan ojol yang belum dapat menjangkau daerah-daerah tertentu dikarenakan dibatasi oleh regulasi (Pahendra dkk., 2019)

Berdasarkan penelitian yang sudah ada sebelumnya maka terdapat beberapa hal yang dapat peneliti lakukan untuk mengembangkan aplikasi atau sistem sebelumnya, sehingga dapat mengembangkan aplikasi yang lebih efektif dan efisien dalam penerapan untuk masyarakat luas. Dimana pengembangan sistem atau aplikasi yang akan penulis lakukan akan dirancang menggunakan *Android Studio*, *Firebase*, dan juga algoritma *Dijkstra*. Maka dari itu berdasarkan latar belakang dan rumusan masalah diatas, penulis tertarik untuk mengambil judul yang nantinya dapat membantu mengembangkan sebuah aplikasi *android* pada Kota Medan, dan dapat dimanfaatkan masyarakat luas dengan judul: **"PERANCANGAN APLIKASI JASA TRANSPORTASI ONLINE PADA KOTA MEDAN MENGGUNAKAN FIREBASE DAN ALGORITMA DIJKSTRA BERBASIS MOBILE"**

1.2 Rumusan Masalah

Berikut merupakan rumusan masalah yang akan ditemukan solusinya pada penelitian ini:

1. Bagaimana cara mengembangkan transportasi *online* dengan menggunakan *realtime firebase* yang nantinya dapat dimanfaat masyarakat luas?
2. Bagaimanakah cara menentukan rute terpendek perjalanan dengan menggunakan algoritma *Dijkstra* pada aplikasi jasa transportasi *online* yang akan dibangun?

1.3 Batasan Masalah

Dalam membangun aplikasi jasa transportasi ini, penulis akan membatasi masalah yang akan dibahas, yakni:

1. Penelitian ini dibangun menggunakan *Android Studio* versi 4.0.1 sebagai aplikasi untuk membangun aplikasi *android* yang akan dibangun, *Firebase* sebagai pengelola database secara *realtime*.
2. Penelitian nantinya hanya berfokus pada proses pengembangan aplikasi *android* pada *customer* dan *driver* jasa transportasi *online* tersebut
3. Penelitian nantinya dibatasi sampai dengan penerapan fitur pemesanan *bike* pada aplikasi jasa transportasi *online* yang akan dibuat
4. Penelitian nantinya akan menggunakan algoritma *dijkstra* yang akan diterapkan pada aplikasi *driver* yang bertujuan untuk mencari rute terdekat

1.4 Tujuan Penelitian

Terdapat tujuan yang dilakukan dalam penelitian ini yang nantinya akan mencapai hasil yang diinginkan, yakni:

1. Merancang suatu aplikasi yang nantinya dapat menjadi refernsi perusahaan-perusahaan pada Kota Medan dalam membangun sebuah aplikasi jasa transportasi *online* dengan baik dengan menggunakan *Android Studio* versi 4.0.1, dan *Firebase*
2. Merancang sebuah sistem yang dapat membantu *driver* mengetahui ataupun menemukan lokasi penjemputan *customer* maupun lokasi

pengantaran *customer* dan menentukan rute perjalanan dengan menggunakan algoritma *Dijkstra*

1.5 Manfaat Penelitian

1. Bagi Universitas
    a. Penelitian ini nantinya diharapkan dapat digunakan untuk menambah referensi sebagai bahan penelitian lanjutan yang lebih mendalam pada masa yang akan datang.
    b. Dapat mengetahui bagaimana kemampuan mahasiswa dalam menguasai teori dan praktik yang diperoleh selama kuliah
2. Bagi Penulis
    a. Penelitian ini diharapkan dapat memberi kontribusi bagi peneliti dalam menyelesaikan kurikulum tingkat akhir Program Studi Sistem Informasi, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sumatera Utara.
    b. Dapat menambahkan wawasan dan pengalaman penulis dalam bidang programing.
    c. Dapat meningkatkan pengetahuan dan pemahaman penulis mengenai fungsi dari *firebase* dan juga mengenai algoritma *dijkstra* yang akan diterapkan dalam bahasa pemrograman *kotlin* menggunakan Android Studio
3. Bagi Kota Medan
    a. Penelitian ini diharapkan dapat memberikan kontribusi yang lebih baik dalam memulai ide bisnis di bidang transportasi dan jasa

BAB II

TINJAUAN PUSTAKA

2.1 Sistem Informasi

2.1.1 Definisi Sistem

Menurut (Fatoni & Dwi, 2016) Sistem merupakan sekelompok elemen yang saling berintegrasi ataupun bekerja sama untuk mencapai suatu sasaran dan tujuan tertentu. Elemen-elemen yang terdapat didalam suatu sistem tidak dapat berdiri sendiri dikarenakan semua komponen ini saling berhubungan dan saling membutuhkan antara satu elemen dengan elemen yang lainnya untuk mencapai tujuan.

2.1.2 Karakteristik Sistem

Menurut (Turang, 2015) Sistem memiliki karakteristik ataupun sifat-sifat tertentu yakni:

1. Komponen sistem *(component)*

   Komponen sistem adalah segala sesuatu yang mempunyai peran dalam tersusunnya sebuah sistem (Fajarianto dkk., 2017). Seberapapun kecilnya suatu sistem, didalamnya selalu mengandung komponen-komponen ataupun sub sistem
2. Mempunyai batas *(boundary)*

   Batasan sistem berguna sebagai pembeda antara satu sistem dengan sistem yang lainnya. Jika tidak ada batasan sistem maka akan sulit untuk menjelaskan suatu sistem
3. Mempunyai lingkungan *(environments)*

   Terdapat 2 resiko dalam sebuah lingkungan sistem yaitu dapat menguntungkan atupun merugikan. Apabila sebuah lingkungan sistem dapat memberikan keuntungkan biasanya akan selalu dipertahankan untuk menjaga keawetan keberlangsungan sistem. Namun jika lingkungan tersebut memberikan kerugian akan selalu diupayakan agar lingkungan

tersebut mempunyai pengaruh yang seminimal mungkin, bahkan bila perlu akan ditiadakan, jika tidak dilakukan tindakan maka akan dapat berupaya mengganggu kelangsungan hidup dari sistem (Abdullah, 2015)

4. Mempunyai penghubung atau antar muka *(interface)*

   Interface berfungsi sebagai jembatan antara pengguna maupun teknologi itu sendiri. Teknologi informasi antara satu dengan yang lainnya memiliki *interface* yang berbeda-beda sesuai dengan fungsi dan kebutuhan penggunanya. Pembuatan *interface* bertujuan untuk membuat teknologi informasi menjadi lebih mudah digunakan oleh penggunanya ataupun biasa disebut *user friendly* (Yasin & Yumarlin, 2016)

5. Mempunyai masukan *(input)*

   Segala sesuatu yang dimasukkan kedalam sistem sebagai bahan yang akan diproses lebih lanjut untuk menghasilkan *output* (keluaran) (Fajarianto dkk., 2017)

6. Mempunyai pengolahan *(processing)*

   Processing merupakan aktivitas untuk mentransformasikan *input* menjadi *output* yang berguna bagi para pemakainya

7. Mempunyai keluaran *(output)*

   Output merupakan hasil operasi dari energi yang diolah ataupun masukan *(input)* yang diklasifikasikan menjadi *output* sehingga menjadi tujuan sasaran ataupun target pengoperasian suatu sistem yang berguna dari sisa pembuangan

8. Mempunyai sasaran *(objectives)*

   Semua komponen yang terdapat pada sistem harus selalu dirawat dan dijaga agar saling bekerjasama agar dapat mencapai tujuan dan sasaran sistem (Abdullah, 2015)

9. Mempunyai umpan balik *(feedback)*

   Berfungsi untuk mengecek bagaimana terjadinya penyimpanan pada proses oleh suatu sistem dan akan mengembalikannya pada keadaan normal (Fajarianto dkk., 2017)

2.1.3 Definisi Informasi

Informasi adalah data yang sudah diolah sehingga menjadi bentuk yang lebih berguna dan lebih berarti bagi para penerimanya. Data merupakan bentuk jamak dari bentuk tunggal dari sebuah data item. Data adalah kenyataan yang menggambarkan dan menerangkan suatu kejadian dan kesatuan nyata yang sudah terjadi. Kegunaan informasi adalah mengurangi segala jenis ketidakpastian didalam sebuah proses pengambilan kepustusan dalam sebuah kejadian dan keadaan. Informasi dikatakan bernilai ketika mempunyai manfaat yang lebih efektif bila dibandingkan dengan biaya ketika akan mendapatkan atupun memperoleh suatu informasi tersebut. Kualistas dari suatu informasi sangat ditentukan dandi pengaruhi oleh beberapa hal, yakni relevan, efisien, tepat waktu, konsisten, dapat dipercaya, ekonomis, akurat, dan juga ketersediaannya (Jonathan & Lestari, 2015)

2.1.4 Definisi Sistem Informasi

Sistem informasi adalah suatu sistem yang terdapat di dalam suatu organisasi dimana sistem tersebut bersifat manajerial, dapat mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan (Jonathan & Lestari, 2015)

Pengertian sistem informasi adalah sebuah sistem yang menyediakan informasi dengan cara sedemikian rupa sehingga bermanfaat bagi penerima. Secara lebih detil, sistem informasi dapat didefinisikan sebagai sekumpulan perangkat entitas yang terdiri dari perangkat keras (*hardware*), perangkat lunak (*software*), maupun juga *brainware* yang saling berhubungan dan bekerjasama untuk menyediakan data yang diolah sehingga berguna dan bermanfaat bagi penerima data tersebut (Herliana & Rasyid, 2016)

2.2 Perancangan

Perancangan merupakan sebuah langkah terdepan untuk membangun sebuah sistem yang bertujuan untuk menyelesaikan permasalahan-permasalahan yang terdapat pada sebuah sistem yang sedang berjalan, yang dimulai dengan melakukan tahap-tahap analisis terlebih dahulu sehingga sistem yang akan dibuat nantinya akan mengahasilkan sebuah produk yang sangat memuaskan pengguna nantinya (Arifin dkk., 2016)

2.3 Aplikasi

Aplikasi merupakan suatu program komputer yang dibuat melalui bahasa pemrograman tertentu yang digunakan untuk mengerjakan dan melaksanakan suatu tugas ataupun pekerjaan khusus dari para penggunanya. Aplikasi juga merupakan beberapa rangkaian perintah ataupun kegiatan yang dieksekusi oleh komputer. Aplikasi sendiri merupakan suatu program dimana merupakan sebuah *instruction set* yang akan dijalankan oleh *software* yang berperan sebagai sebuah pemroses. Program sendiri berisi beberapa konstruksi logika yang dibuat oleh manusia yang sudah diterjemahkan ke dalam bahasa mesin atau bahasa pemrograman yang sesuai dengan format yang ada pada *instruction set*, sehingga dapat menjadi sebuah program aplikasi yang siap pakai (Al Faruq, 2015)

Aplikasi merupakan sebuah subkelas pada perangkat lunak komputer yang memanfaatkan maupun mengintegrasikan berbagai kemampuan komputer untuk melakukan suatu tugas yang diinginkan pengguna dan dapat menguntungkan pengguna. Aplikasi juga merupakan sebuah *software* yang mempunyai fungsi untuk membantu dan melakukan berbagai bentuk pekerjaan atau tugas-tugas tertentu dari pengguna seperti melakukan penerapan, penambahan data, maupun pnggunaan (Hartati dkk., 2017)

2.4 Jasa

Jasa adalah seluruh aktivitas ekonomi yang dapat menghasilkan sebuah hasil/*output* dalam pengertian fisik, dikonsumsi dan diproduksi pada saat yang

bersamaan, memberikan nilai tambah dan secara prinsip tidak berwujud (*intangible*) bagi pembeli pertamanya. Jasa adalah sebuah aktivitas ekonomi yang tidak menghasilkan sebuah hasil/*output* dalam bentuk produk (Ningratri, 2018)

2.4.1 Kualitas Jasa

Menurut (Azkiyah dkk., 2018), Terdapat lima dimensi kualitas jasa, yakni:

1. *Tangible* (Dimensi tampilan fisik), dimana perusahaan memberikan beberapa fasilitas kepada konsumen yakni sarana komunikasi, fasilitas fisik, perlengkapan pegawai.
2. *Reliability* (Dimensi Keandalan), kemampuan layanan yang dijanjikan perusahaan kepada konsumen meliputi pelayanan yang memuaskan, kecepatan pelayanan, dan keakuratan pelayanan
3. *Responsiveness* (Dimensi Daya Tanggap), layanan dari perusahaan berupa para staf perusahaan dalam membantu konsumen dan memberikan pelayanan secara tanggap
4. *Assurance* (Dimensi Jaminan), perusahaan memberikan layanan kepada konsumen yang mencakup kesopanan, kemampuan, serta sifat yang dapat dipercaya yang dimiliki para staff yakni bebas dari resiko, bebas dari bahaya, serta keraguan-keraguan
5. *Emphaty* (Dimensi Empati), meliputi sebuah kemudahan dalam melakukan sebuah hubungan, perhatian pribadi, komunikasi yang baik, serta mampu memahami kebutuhan para konsumen

2.5 Transportasi

Transportasi merupakan suatu kegiatan dengan memanfaatkan ataupun menggunakan alat pengangkutan yang digerakan oleh tenaga manusia, hewan (kerbau, kuda, sapi) ataupun dengan menggunakan mesin untuk berpindah dari suatu tempat ke tempat lain. Konsep transportasi didasarkan pada adanya asal, tujuan (*destination*), dan perjalanan (*trip*) (Manueke dkk., 2018)

2.5.1 Manfaat, fungsi, dan Jenis Transportasi

Menurut (Setiani, 2015), transportasi memiliki manfaat, fungsi, serta jenisnya yakni:

1. Manfaat:
    a. Manfaat Ekonomi, dapat menciptakan sebuah manfaat ekonomi dikarenakan dapat memenuhi banyak kebutuhan manusia.
    b. Manfaat Sosial, transportasi dapat menciptakan berbagai kemudahaan, yakni pelayanan untuk perorangan maupun kelompok, mendekatkan suatu jarak, dapat memencarkan penduduk, dapat dimanfaatkan sebagai media pertukaran dan penyampaian informasi, dan juga dapat dimanfaatakan sebagai perjalanan untuk bersantai.
    c. Manfaat Politik, transportasi dapat dimanfaatkan sebahagai pelayanan yang lebih luas, dapat menciptakan suatu persatuan, dapat dimanfaatakan sebagai keamanan sebuah negara, dapat dimanfaatakan sebagai mengatasi suatu musibah dan bencana, dan sebagainya
    d. Manfaat Kewilayahan, memenuhi kebutuhan penduduk di desan maupun di pedalaman
2. Fungsi
    a. Dapat melancarkan arus perpindahan barang dan manusia
    b. dapat dimanfaatkan sebagai penunjang pembangunan (*the promoting sector*)
3. Jenis
    a. Transportasi Darat
    b. Transportasi Laut
    c. Transportasi Udara

2.6 *Online*

Pada dasarnya *online* adalah terhubung dengan internet, terkoneksi dengan internet, serta aktif sehingga siap untuk dioperasikan dan dapat menjalin komunikasi dengan atau dikontrol oleh komputer. *Online* juga dapat didefinisikan

sebagai suatu keadaan ataupun situasi dimana sebuah *device* yaitu komputer dapat terhubung dengan komputer ataupun *device* yang lain dan biasanya melalui perangkat modern. Pengertian *online* juga dapat dijelaskan sebagai suatu keadaan dimana pengguna sedang menggunakan jaringan, serta dapat terhubung dalam jaringan, sehingga dapat menghubungkan satu perangkat dengan perangkat lainnya sehingga bisa terjalinnya suatu komunikasi antara satu perangkat dengan perangkat lainnya (Almuttaqin, 2016)

2.7 Transportasi *Online*

Transportasi *online* atau biasa juga disebut ojek *online* adalah sebuah jasa yang dapat mengangkut barang sekalipun manusia yang sistem order dan pemesanannya berbasis *online* dengan menggunakan smartphone.

(Zakinah, 2019), Ojek *online* merupakan bagian dari kemajuan teknologi yang diciptakan untuk mempermudah berbagai aktivitas manusia sehari-hari seperti bepergian kemanapun. Ojek *online* sudah menjadi alternatif di kalangan masyarakat karena mempunyai beberapa keunggulannya yang mencakup:

1. Kepraktisan, dimana layanan tersebut dapat di order cukup menggunakan ponsel pintar ataupun smartphone yang didalamnya hanya membutuhkan koneksi internet dan aplikasi jasa transportasi *online* yang sudah di download.
2. Keterpercayaan dan keamanan, dimana setiap pengemudi ataupun *driver* jasa transportasi *online* sudah terdaftar identitas lengkapmya di perusahaan jasa transportasi *online* yang berkaitan, berupa NIK, No. STNK, dll. Sehingga dapat meminimalisirkan beberapa resiko tindak kejahatan pada kendaraan umum.
3. Transparansi, dimana setiap konsumen ataupun pengguna dapat mengetahui dengan pasti setiap informasi tentang driver, waktu tempuh perrjalanan, harga yang sudah ditetapkan, serta posisi dari kendaraan
4. Lahan kerja sampingan yang baru bagi beberapa orang karena mempunyai waktu kerja yang sangat fleksibel dan cara daftar yang mudah. (Zakinah, 2019)

2.8 *Firebase*

Firebase merupakan sebuah *platform* untuk mengembangkan aplikasi berbasis *mobile* maupun *web* secara *realtime Firebase* sendiri dilengkapi dengan *library* yang lengkap untuk sebagian besar *platform mobile* maupun *web* dan dapat digabungkan dengan berbagai *framework* seperti *node, java, Java Script, AngularJS,* dan masih banyak lagi. *Application Programming Interface* (API) untuk menyimpan dan sinkronisasi data akan disimpan sebagai bit-bit dalam bentuk *JSON* pada *cloud* dan akan disinkronisasi secaca *realtime. Firebase* sendiri mampu menerima data dari 1 juta perangkat ataupun lebih secara bersamaan (Susanti dkk., 2016)

Menurut (Sanad dkk., 2018) *firebase* mempunyai beberapa fitur yang sudah disediakan, seperti:

1. *Analytic,* sebuah fitur dimana *firebase* akan memunculkan sebuah *dashboard* dimana bertujuan untuk mengamati tingkah laku pengguna dalam penggunaan sebuah aplikasi
2. *Develop,* dalam fitur ini dibagi lagi dengan beberapa fitur didalamnya seperti *hosting, testlab, cloud messaging, authentication, storage, realtime database,* maupun *crash reporting*
3. *Grow,* fitur ini berfungsi untuk mempublikasikan sebuah produk aplikasi

Gambar 2. 1 Fitur *Firebase*

Firebase juga mempunyai 3 layanan, yaitu:

1. *Realtime*, apabila terdapat beberapa perubahan pada *database*, maka seluruh *client* dengan otomatis akan mendapatkan perubahan yang cepat.
2. *Offline,* aplikasi yang menggunakan fitur ini akan tetap responsif walaupun dalam keadaan tidak terhubung dengan jaringan internet. Hal ini dikarenakan *Firebase SDK* dapat mempertahankan data
3. *Accessible From Client Devices,* yang merupakan sebuah layanan yang menawarkan kemudahan untuk mengakses *Firebase Realtime Database* secara langsung melalui sebuah perangkat *mobile* ataupun sebuah peramban *web* tanpa membutuhkan *server application.*

2.9 Aplikasi *Mobile*

Aplikasi *mobile* atau biasa disebut seluler merupakan sebuah aplikasi yang dibuat ataupun dirancang khusus untuk sebuah platform seluler seperti *IOS, Android* ataupun *Windows Mobile*. Aplikasi *mobile* juga menyediakan mekanisme interaksi *interface* yang unik, menyediakan kemampuan penyimpanan permanen di sebuah *platform*. Melalui penggunaan aplikasi *mobile*, pengguna dapat dengan mudah melakukan dan mendapatkan berbagai kegiatan dan aktivitas seperti media pembelajaran, hiburan, navigasi, dan sebagainya (Samsudin, Irawan, dkk., 2019)

2.10 *Android*

Android merupakan sistem operasi yang menggunakan *Linux* yang dirancang untuk telepon pintar (*smartphone*). Pada awalnya *Android* dikembangkan oleh *Android, Inc.* yang didukung dengan finansial dari *Google* dan kemudian membelinya pada tahun 2005. *Android* merupakan sebuah sistem operasi yang didukung dengan *open source,* dan *Google* merilis kodenya di bawah lisensi *apache*. *Android* memungkinkan penggunanya untuk memodifikasi perangkat lunaknya secara bebas dan didistribusikan oleh pembuat perangkat, operator nirkabel dan pengembang aplikasi (Angela & Gani, 2016)

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis sistem operasi *Linux* yang dapat mencakup sistem operasi, *middleware* dan aplikasi. *Android* menyediakan *platform* terbuka ataupun *opensource* bagi para pengembang, diamana para pengguna ataupun pengembangnya dapat dengan bebas menciptakan bahkan memodifikasi aplikasi mereka sesuai keingingan dan kebutuhan (Pratama & Hermawan, 2016)

Adapun aplikasi yang akan dibuat nantinya mempunyai target minimum *SDK* untuk versi *android* minimum yang dapat menjalankan aplikasi yang akan dibuat penulis adalah *Android* dengan versi *Android* 4.1 *(Jelly Bean) API* 16. Menurut (Kusniyati & Sitanggang, 2016) *Android* versi *JellyBean* merupakan sebuah nama kode yang diberikan kepada versi kesepuluh dari *Android* sistem operasi selular yang dikembangkan oleh *Google*, yang mencakup tiga poin utama (versi 4.1 hingga 4.3.1)

2.10.1 Arsitektur *Android*

Menurut (Anwar, 2015), Arsitektur *Android* dapat dijelaskan dan digambarkan sebagai berikut:

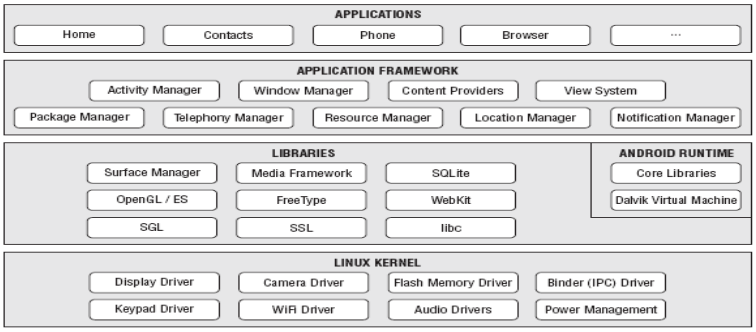
1. *Applications* dan *Widgets,* yang merupakan sebuah lapisan atau layer di mana pengguna hanya dapat berhubungan dengan aplikasi saja, seperti *browser* bawaan, *maps,* kontak, SMS, serta aplikasi-aplikasi lainnya yang dapat di download yang dibuat menggunakan bahasa pemrograman *Java*.
2. *Applications Framework,* dimana para pembuat dan pengembang aplikasi dapat dengan bebas melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi *Android*. Adapun komponen - komponen yang termasuk di dalam *Applications Frameworks* adalah sebagai berikut:
    a. *Views*
    b. *Content Provider*
    c. *Resource Manager*
    d. *Notification Manager*

e. *Activity Manager*

3. *Libraries* adalah *layer* di mana fitur - fitur *Android* berada, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya. Berjalan di atas kernel, Layer ini meliputi berbagai library C/C++ inti seperti Libc dan SSL, serta:
   a. *libraries* media untuk pemutaran media audio dan video.
   b. *libraries* untuk manajemen tampilan.
   c. *libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D.
   d. *libraries SQLite* untuk dukungan database.
   e. *libraries* SSL dan *WebKit* terintegrasi dengan *web browser* dan *security*.
   f. *libraries LiveWebcore* mencakup modern *web browser* dengan *engine embeded web view*.
   g. *libraries* 3D yang mencakup implementasi OpenGL ES 1.o API's
4. *Android RunTime Layer* yang membuat aplikasi *Android* dapat dijalankan di mana dalam prosesnya menggunakan Implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam *Android Run Time* dibagi menjadi dua bagian yaitu:
   a. *Core Libraries*: Aplikasi *Android* dibangun dalam bahasa java, sementara Dalvik sebagai virtual mesinnya bukan *Virtual Machine Java*, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa java/c yang ditangani oleh *Core Libraries*.
   b. *Dalvik Virtual Machine*: *Virtual* mesin berbasis *register* yang dioptimalkan untuk menjalankan fungsi - fungsi secara efisien, di mana merupakan pengembangan yang mampu membuat linux kernel untuk melakukan *threading* dan manajemen tingkat rendah.

5. *Linux Kernel* adalah layer di mana inti dari *operating system* dari *Android* itu berada. Berisi file - file system yang mengatur sistem p*rocessing, memory, resource, drivers*, dan sistem - sistem operasi *android* lainnya.

Linux kernel yang digunakan *android* adalah *linux kernel* release 2.6



Gambar 2. 2 Arsitektur *Android*

2.10.2 *Android SDK (Software Development Kit)*

Menurut (Kusniyati & Sitanggang, 2016) *Android SDK* merupakan *tools* API (*Application Programming Interface*) ataupun *kit* yang diperlukan untuk memulai mengembangkan aplikasi pada *platform Android* menggunakan bahasa pemrograman *Java* oleh *developer*. Beberapa fitur *Android* yang penting adalah sebagai berikut:

a. *Framework* aplikasi yang mendukung penggantian komponen
b. *DVM* ataupun *Dalvik Virtual Machine* yang ioptimalkan untuk perangkat mobile.
c. *Integrated browser* berdasarkan *engine open source WebKit*.
d. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarkan spesifikasi *OpenGL ES 1.0.*
e. *SQLite* untuk penyimpanan data.
f. Dukungan untuk audio, video dan gambar.
g. Bluetooth, Edge, 3G, Wifi.
h. Kamera, *GPS*, kompas dan *accelerometer*. Lingkungan development yang lengkap dan kaya termasuk perangkat emulator, *tools* untuk *debugging*, profil dan kinerja memori serta *plugins* untuk *IDE Eclipse.*

2.10.3 Versi *Android*

Dari awal kemunculannya hingga sampai saat ini, *android* selalu melakukan pembaruan pada sistem operasinya.

Tabel 2. 1 Versi *Android* (Firly, 2018)

Tanggal Rilis	**Versi**	**Nama**
23 September 2008	1.0	*Alpha*
09 Februari 2009	1.1	*Beta*
27 April 2009	1.5	*Cupcake*
15 September 2009	1.6	*Donut*
26 Oktober 2009	2.0	*Eclair*
03 Desember 2009	2.0.1	*Eclair*
12 Januari 2009	2.1	*Eclair*
20 Mei 2010	2.2-2.2.3	*Gingerbread*
06 Desember 2010	2.3-2.3.2	*Gingerbread*
09 Februari 2011	2.3.3-2.3.7	*Gingerbread*
22 Februari 2011	3.0	*Honeycomb*
10 Mei 2011	3.1	*Honeycomb*
15 Juli 2011	3.2	*Honeycomb*
19 Oktober 2011	4.0-4.0.2	*Ice Cream Sandwich*
16 Desember 2011	4.0.3-4.0.4	*Ice Cream Sandwich*
27 Juni 2012	4.1	*Jelly Bean*
29 Oktober 2012	4.2	*Jelly Bean*

16 Desember 2013	4.3	*Jelly Bean*
31 Oktober 2013	4.4	*KitKat*
12 November 2014	5.0	*Lollipop*
05 Oktober 2015	6.0	*MarshMallow*
09 Maret 2016	7.0	*Nougat*
19 Oktober 2016	7.1	*Nougat*
21 Maret 2017	8.0	*Oreo*

Terdapat pula generasi Android 9.0 yang dinamakan Android *Pie* yang diperkenalkan pada Agustus 2018. Android 10 diluncurkan pada September 2019 dan Android 11 yang diluncurkan pada tahun 2020.

2.11 *Kotlin*

Kotlin awalnya dikembangkan oleh *JetBrains*, yang merupakan sebuah perusahaan dibalik *Intellij IDEA*. Setelah melalui banyak perkembangan, *JetBrains* merilis *kotlin* secara *open source* dan sampai saat ini perkembangannya semakin maju. *Google* mendukung penuh kotlin sebagai pengembang aplikasi *android*. *Kotlin* merupakan sebuah bahasa pemrograman berbasis *JVM (Java Virtual Machine)*. Bahasa pemrograman *kotlin* sangat praktis untuk android apabila digunakan untuk mengkombinasikan *object oriented (OO*) dengan bahasa fungsional. Bahasa pemrograman *kotlin* juga mempunyai sifat *interoperabilitas* berartikan sebuah aplikasi dapat berinteraksi dengan aplikasi lainnya dengan melewati sebuah protokol yang sama. Sifat *interoperabilitas* pada *kotlin* sendiri memungkinkan bahasa ini dapat digabungkan dalam satu *project* yang sama dengan menggunakan bahasa pemrograman *java*. Selain digunakan untuk membuat dan mengembangkan aplikasi berbasis *android* bahasa pemrograman

kotlin sendiri juga dapat digunakan untuk membuat ataupun mengembangkan aplikasi berbasis *desktop*, *web* dan juga *backend*. (Febriandirza, 2020)

Menurut (Sibarani dkk., 2018) Bahasa pemrograman *kotlin* mempunyai kelebihan ataupun performa yang lebih baik daripada bahasa pemrograman bahasa *java*, dimana aplikasi yang dibuat ataupun dikembangkan dengan bahasa pemrograman *kotlin* lebih minim dalam penggunaan *CPU Usage* daripada menggunakan bahasa pemrograman *java* walaupun mempunyai perbedaan yang tidak signifikan yaitu 0.65%. Bahasa pemrograman *kotlin* juga lebih kecil dalam penggunaan *memory usage* dengan perhitungan lebih dari 2 kali lipat lebih hemat dalam penggunaan memori dari pada menggunakan bahasa pemrograman *java*. Selain itu dalam mengeksekusi sebuah program *(execution time program)* bahasa pemrograman *kotlin* menghabiskan waktu yang lebih cepat dibandingkan menggunakan bahasa pemrograman *java*. Berdasarkan hasil keseluruhan pengukuran dapat disimpulkan bahwa bahasa pemrograman Kotlin memiliki performa yang lebih baikdibandingkan bahasa pemrograman Java untuk aplikasi berbasis android

2.12 *Android Studio*

Android Studio merupakan sebuah IDE (Integrated development Environment) atau merupakan program komputer yang didalamnya menyediakan berbagai utilitas yang diperlukan dalam membentuk sebuah perangkat lunak dan bersifat *open source* atau gratis. *Android Studio* menyediakan berbagai fitur dan peralatan yang sangat dibutuhkan oleh para developer (pengembang) dengan pemograman *java*, sehingga dapat memberikan kemudahan dalam pembuatan aplikasi *android* dan juga meningkatkan produktivitas dalam membuat aplikasi *android***.** *Android Studio* di perkenalkan oleh google secara resmi pada tahun 2013 (Tangkudung dkk., 2018)

Menurut (Juansyah, 2015), *Android studio* memiliki fitur:

1. Projek berbasis pada *Gradle Build*
2. *Refactory* dan pembenahan *bug* yang cepat

3. *Tools* baru yang bernama "*Lint*" dikalim dapat memonitor kecepatan, kegunaan, serta kompetibelitas aplikasi dengan cepat.
4. Mendukung *Proguard And App-signing* sebagai keamanan.
5. Memiliki *GUI* aplikasi *android* lebih mudah
6. Didukung oleh *Google Cloud Platfrom* untuk setiap aplikasi yang dikembangkan.

2.13 Algoritma *Dijkstra*

Seorang ilmuwan komputer asal Belanda yang bernama Edsger Dijkstra berhasil menemukan sebuah algoritma teori *graf* yang diberikan nama algoritma *Dijkstra*. Algoritma *dijkstra* merupakan sebuah algoritma yang digunakan untuk menyelesaikan persoalan pencarian rute terpendek ataupun lintasan terpendek dari satu *vertex* ke *vertex* yang lainnya pada suatu *graf* yang berbobot, jarak antara *vertex* adalah nilai bobot dari setiap *edge* pada *graph*. Suatu *graf* yang mempunyai bobot, harus mempunyai nilai yang positif (bobot >= 0) Algoritma *Dijkstra* sendiri menggunakan strategi *greedy* dalam pengerjaannya, di mana pada setiap langkah dipilih sisi dengan bobot terkecil yang menghubungkan sebuah simpul yang sudah terpilih dengan simpul lain yang belum terpilih (Harahap & Khairina, 2017). Sedangkan menurut (Triase & Aprilia, 2020) algoritma *Dijkstra* merupakan salah satu algoritma yang banyak digunakan dan sangat popular dalam menyelesaikan sebuah permasalahan dalam pencarian jalur terpendek. Dalam penyelesaiannya algoritma ini memiliki teknik sumber tunggal untuk menentukan jalur terpendek. Algoritma ini mempunya sebuah teori dasar yaitu, dalam setiap *vertex* tertentu mempunyai peran sebagai titik jalur terpendek untuk ke semua titik lain yang ingin dituju. Algoritma *Dijkstra* ini tidak hanya terfokus dalam mencari jalur terpendek dari setiap *vertex,* namun dapat mencari jalur ke semua *vertex* yang ada

2.14 *GPS*

Global Positioning System (*GPS*) merupakan sistem navigasi berbasis

satelit yang dapat difungsikan 24 jam sehari dalam segala kondisi cuaca, tanpa biaya operasional pada bagian dunia mana pun. Terdapat setidaknya 24 satelit buatan yang mengorbit bumi pada ketinggian disekitar 20.000 km yang mengirimkan sinyal gelombang mikro ke bumi. Sinyal gelombang terssebutlah yang diterima dan dimanfaatkan untuk menentukan arah dan posisi. *GPS* didesain untuk memperoleh data posisi, infprmasi waktu serta kecepatan suatu perpindahan dalam bentuk tiga dimensi, secara terus menerus di seluruh dunia, dan tidak bergantung pada waktu dan cuaca. Posisi suatu titik dinyatakan dengan koordinat. *GPS* mempunyai 3 segmen utama, yaitu segmen pemakai, segmen angkasa, dan segmen sistem kontrol. *GPS* memerlukan alat penerima sinyal *GPS* (*GPS Receiver*) agar dapat menangkap sinyal data dan memproses sinyal yang diperoleh satelit *GPS* untuk digunakan pada penentuan posisi lokasi, kecepatan, waktu maupun parameter turunan lainnya (Susanti dkk., 2016)

2.14.1 *Latitude*

Latitude atau garis lintang merupakan garis yang menentukan lokasi berada di sebelah utara atau selatan ekuator. Garis lintang diukur mulai dari titik 0 derajat dari khatulistiwa sampai 90 derajat di kutub (Pamungkas, 2019)

2.14.2 *Longitude*

Longitude atau garis bujur merupakan digunakan untuk menentukan lokasi di wilayah barat atau timur dari garis utara selatan yang sering disebut juga garis meridian. Garis bujur diukur dari 0 derajat di wilayah Greenwich sampai 180 derajat di International Date Line (Pamungkas, 2019)

2.15 *UML*

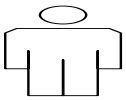
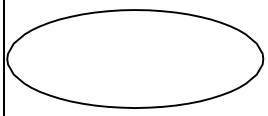
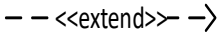
UML yang merupakan *Unified Modelling Language* ialah sebuah bahasa ataupun sekumpulan alat yang digunakan untuk merancang pemodelan dari sebuah sistem dan juga telah menjadi sebuah standar untuk visualisasi sehingga

dapat mempermudah dalam mengembangkan dan merancang sebuah aplikasi (Irawan, 2017)

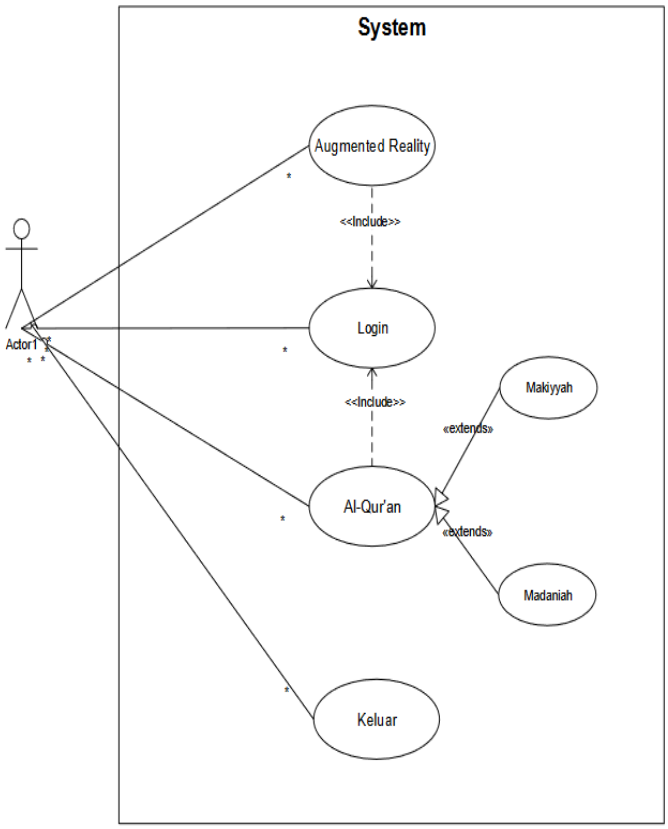
2.15.1 *Use Case Diagram*

Use Case Diagram berfungsi untuk menggambarkan jalannya suatu proses sistem yang terkait, dan juga kegunaan *use case* sendiri adalah menjelaskan fitur dan hubungan antara admin, aktor, dan sistem itu sendiri (Suendri dkk., 2020)

Tabel 2. 2 Simbol *Use Case Diagram*

No	Simbol	Keterangan
1.		*Actor* Simbol ini merupakan himpunan peran dari pengguna ketika melakukan hubungan atau interaksi dengan *use case*.
2.		*Use case* Gambaran sebuah interaksi atau hubungan antara aktor dengan sistem
3.		*Association* Penghubung antara objek dengan objek lainnya
4.		*Ekstensi* Menjelaskan jika suatu use case mempunyai kondisi yang terpenuhi maka *use case* lainnya merupakan sebuah tambahan fungsional

5.	<<include>>	*Include* Menjelaskan jika suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya
6.		Generalisasi/Generalization Menggambarkan spesialisasi aktor untuk dapat berpartisipasi dengan *use case*



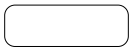
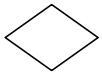
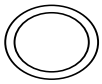
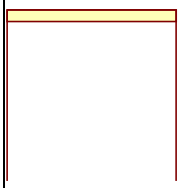
Gambar 2. 3 Contoh *Use Case Diagram (Samsudin, Zufria, dkk., 2019)*

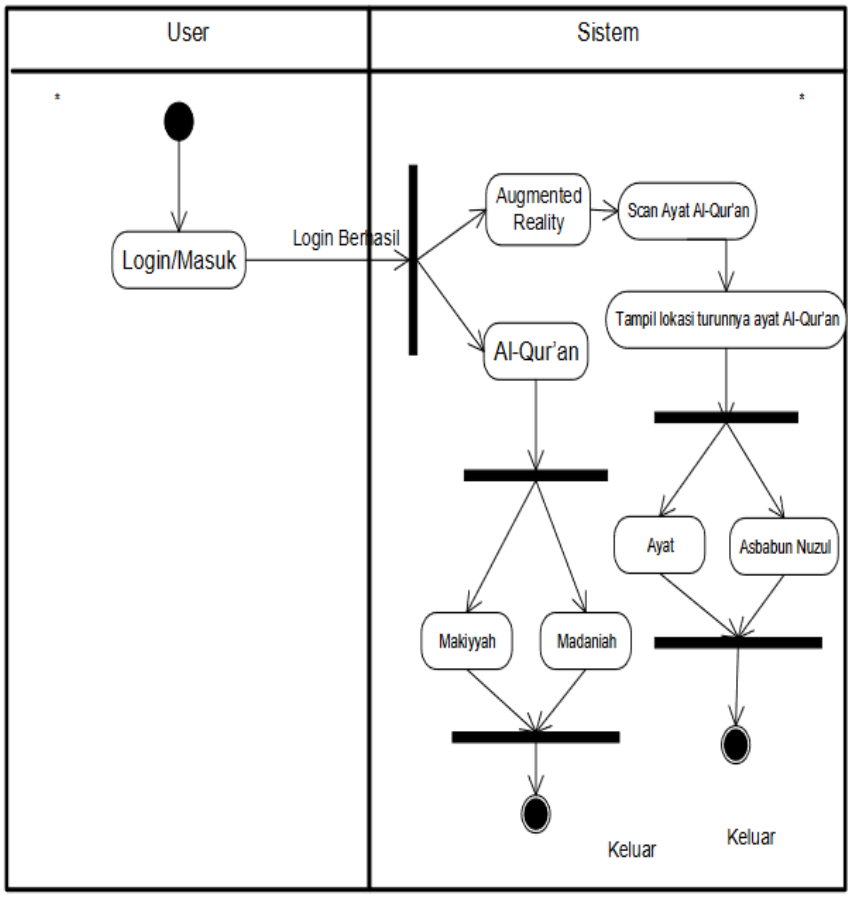
2.15.2 *Activity Diagram*

Activity diagram merupakan penggambaran dari aliran kerja *(workflow)* dan juga aktivitas dari sebuah sistem atau proses bisnis yang ada pada perangkat

lunak. *Activity diagram* sendiri menggambarkan sebuah aktivitas sistem itu sendiri dan bukan apa yang dilakukan aktor, namun sebuah aktivitas yang dapat dilakukan oleh sistem (Samsudin, Zufria, dkk., 2019)

Tabel 2. 3 Simbol *Activity Diagram*

1.		*Initial Node* Status awal dari sebuah objek pada sistem
2.		*Activity* Aktivitas dari satu sistem agar dapat saling beriteraksi satu sama lain
3.		Percabangan / *Decision* Sebuah percabangan yang digunakan apabila ada sebuah tindakan atapun keputusan lebih dari satu yang akan diambil
4.		Penggabungan/ *Join* Menggabungkan lebih dari satu aktivitas
5.		*Final* Status akhir dari sebuah objek pada sistem
6.		*Swimline* Menunjukkan organisasi penanggung jawab terhadap aktivitas yang sudah terjadi.

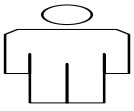


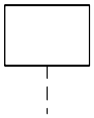
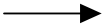
Gambar 2. 4 Contoh *Activity Diagram (Samsudin, Zufria, dkk., 2019)*

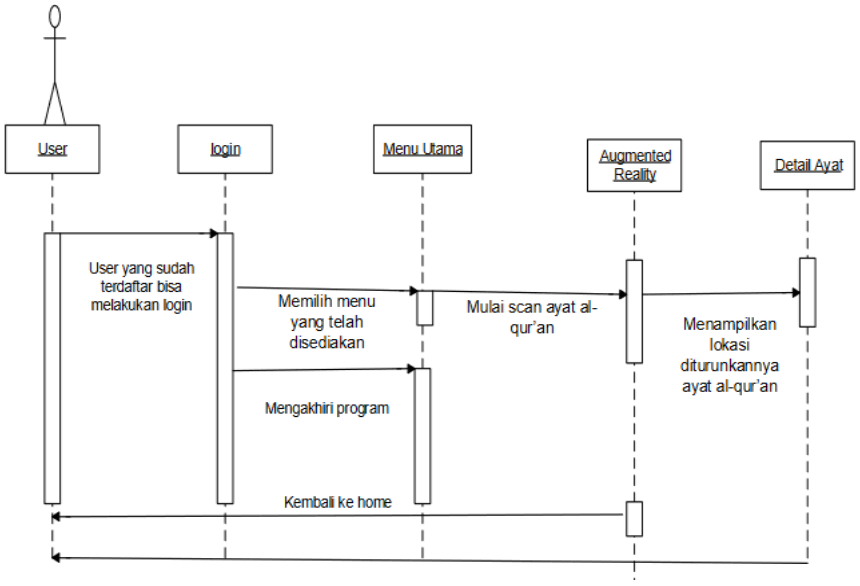
2.15.3 *Sequence Diagram*

Sequence Diagram berfungsi untuk menggambarkan aktivitas antara suatu objek dengan objek lainnya melalui pesan diantara objek dan *sequence* nya yang melakukan suatu proses tertentu (Suendri dkk., 2020)

Tabel 2. 4 Simbol *Sequence Diagram*

No	Simbol	Keterangan
1.		*Actor* *User* yang berinteraksi dengan sistem

2.		*LifeLine* Sebuah antarmuka mulai dan berakhirnya suatu pesan
3.		*Message* Sebuah pesan dari satu objek ke objek lain
4.		*Message to self* Sebuah aktivitas atau pesan yang memuat pada aktivitas itu sendiri



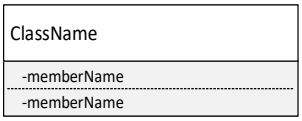
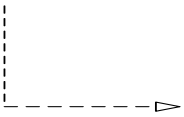
Gambar 2. 5 Contoh *Sequence Diagram (Samsudin, Zufria, dkk., 2019)*

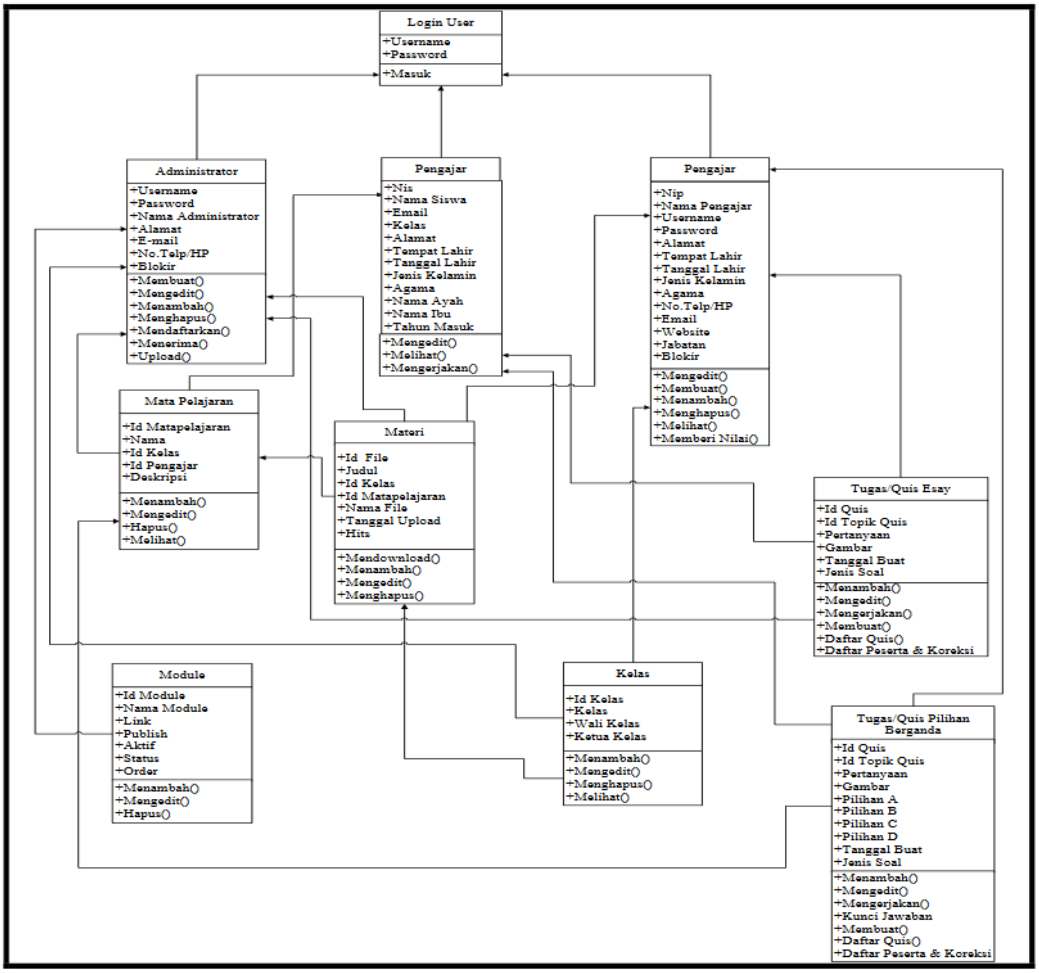
2.15.4 *Classs Diagram*

Class diagram digunakan untuk menggambarkan hubungan antar *class*, perbedaan yang cukup mendasar antar satu *class* dengan *class* yang lainnya, dan juga menggambarkan dimana letak *sub-sistem class* tersebut. Pada *class diagram*

terdapat komponen-komponen, yaitu nama dari sebuah *class*, *attributes* dari sebuah *class*, *operation*, serta *association* (hubungan antar *class*) (Ikhwan, 2020)

Tabel 2. 5 Simbol *Class Diagram*

1.		*Generalization* Merupakan sebuah relasi atau hubungan antar *class* yang mempunyai makna umum
2.		*Class* Sebuah class pada struktur sistem
4.		*Realization* Sebuah class harus wajib mengikuti aturan yang ditetapkan *class* lain dikarenakan sudah mempunyai hubungan
5.		*Dependency* Hubungan antar *class* yang mempunyai ketergantungan dengan *class* lain karena tidak mempunyai suatu elemen yang mandiri
6.		*Association* Hubungan antara *class* satu dengan *class* lainnya



Gambar 2. 6 Contoh Class Diagram (Sihotang, 2017)

BAB III

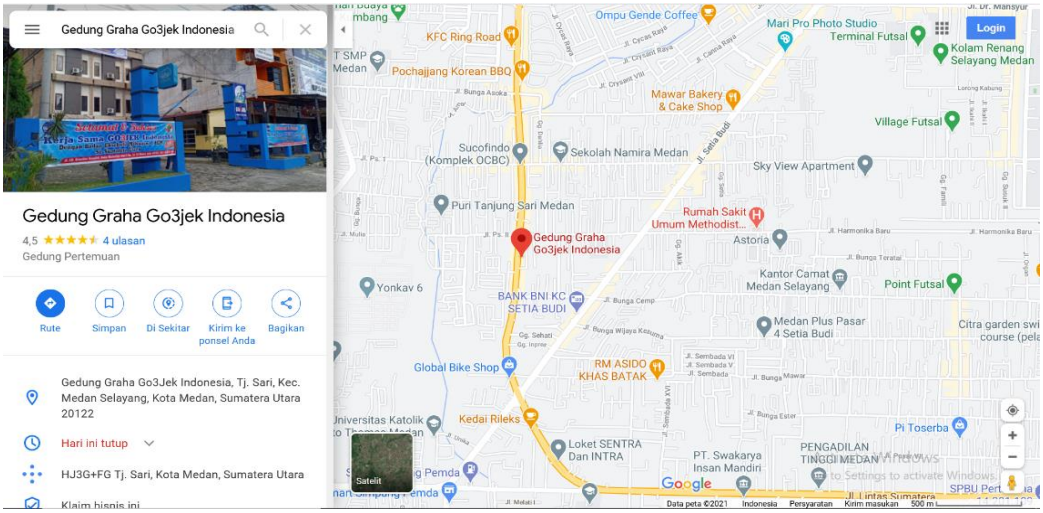
METODE PENELITIAN

3.1 Tempat dan Waktu Penelitian

Dalam penulisan skripsi ini penulis melakukan penelitian di sebuah perusahaan rintisan ataupun *Startup* yang masih berada pada level *Cockroach* yang akan membangun sebuah sistem ojek *online* atau bisa disebut transportasi *online* asal Medan yang mempunyai nama GO3JEK. Penelitian ini dilakukan pada perusahaan GO3JEK dikarenakan, perusahaan tersebut ingin membuat aplikasi jasa transportasi *online* yang baru untuk menggantikan peran aplikasi GO3JEK sebelumnya yang sudah beroperasi di Kota Medan, namun terdapat beberapa masalah pada aplikasi sebelumnya tersebut.

3.1.1 Tempat Penelitian

Dalam penulisan skripsi dengan judul "PERANCANGAN APLIKASI JASA TRANSPORTASI ONLINE PADA KOTA MEDAN MENGGUNAKAN *FIREBASE* DAN ALGORITMA *DIJKSTRA* BERBASIS *MOBILE*" penulis melakukan penelitian di GO3JEK Medan yang beralamatkan di Gedung Graha GO3JEK Indonesia, Tj. Sari, Kec. Medan Selayang, Kota Medan, Sumatera Utara



Gambar 3. 1 Map Lokasi GO3JEK (Sumber Google Map)

Kode Pos 20122

3.1.2 Waktu & Jadwal Pelaksanaan Penelitian

Penelitian dilakukan pada antara bulan September tahun 2020 sampai dengan bulan Februari tahun 2021 dengan deskripsi sebagai berikut:

Tabel 3. 1 Waktu & Jadwal Penelitian

Waktu Kegiatan	Tahun 2020-2021									
	November	Desember	Januari	Februari	Maret	April	Mei	Juni	Juli	Agustus
Identifikasi Masalah										
Pengajuan dan Pengerjaan Proposal										
Seminar Proposal										
Pengumpulan Data										
Analisis Sistem										
Perancangan Sistem										
Pembuatan Coding										
Uji Coba										

Adapun jadwal penelitian yang dibutuhkan adalah sebagai berikut:

1. Identifikasi Masalah

   Dalam tahap ini penulis melakukan observasi terlebih dahulu terhadap permasalahan yang dimiliki oleh perusahaan yang terkait yang bisa diangkat menjadi tema penelitian yang bisa diselesaikan dengan kemajuan teknologi.

2. Pengajuan dan Pengerjaan Proposal Skripsi

Setelah menemukan masalah yang ingin diangkat, penulis melakukan pengajuan judul sebagai syarat untuk mengajukan proposal skripsi, dan melakukan pengerjaan proposal skripsi sebagai persiapan seminar proposal.

3. Seminar Proposal Skripsi

   Seminar proposal diadakan agar melihat kesesuaian penelitian yang diangkat dengan melakukan presentasi judul terkait. Seminar proposal ini akan menguji kelayakan lxatar belakang, rumusan masalah, batasan masalah, tujuan penelitian, dan materi-materi pendukung lainnya yang sudah di persiapkan penulis dan sudah melakukan proses bimbingan kepada dosen pembimbing I, dan II sebelumnya.

4. Pengumpulan Data

   Setelah seminar proposal maka dilakukan pengumpulan data ketempat peneletian yang menjadi sasaran penelitian, melakukan wawancara, serta studi pustaka terkait penelitian.

5. Analisis Sistem

   Setelah data terkumpul maka penulis melakukan analisa terhadap data yang didapatkan untuk membuat sebuah usulan sistem yang lebih baik dari sistem yang berjalan sebelumnya.

6. Perancangan Sistem

   Pada tahap ini penulis mulai melakukan perancangan dengan membuat alur sistem melalui diagram model *UML*, perancangan database, dan juga perancangan *interface*.

7. Pembuatan *Coding*

   Tahap ini merupakan tahap dimana penulis melakukan pembuatan kode program sehingga menjadi suatu aplikasi

8. Uji Coba

   Setelah *coding* sudah diselesaikan, maka penulis akan melakukan uji coba terhadap sistem, sehingga dapat diketahui apakah sistem berjalan sesuai dengan yang diharapkan atau tidak

3.2 Kebutuhan Sistem

Kebutuhan sistem harus dilengkapi untuk menyelesaikan penelitian skripsi, adapun kebutuhan sistem dibagi menjadi perangkat keras dan perangkat lunak

3.2.1 Perangkat Keras

Adapun kebutuhan perangkat keras ataupun *hardware* yang digunakan untuk proses pembuatan sistem tersebut. Spesifikasi *hardware* yang digunakan pada perancangan sistem, yakni:

1. Tipe ACER Aspire E 14
2. Processor Intel(R) Core (TM) i5-8250U CPU @1.60Hz 1.80GHz
3. RAM DDR4 12 GB
4. HDD 1000 GB
5. FULL HD 1080 Graphics

3.2.2 Perangkat Lunak

Adapun kebutuhan perangkat lunak ataupun *software* untuk kebutuhan proses pembuatan sistem tersebut. Spesifikasi *software* yang digunakan pada perancangan sistem, yakni:

1. *Android Studio4.0.1*
2. *Firebase*
3. *MYSQL*

3.3 Cara Kerja

Cara kerja pada penelitian ini menggunakan metode pengumpulan data yang dilakukan secara kualitatif yaitu berupa observasi, wawancara, dan studi pustaka. Untuk metode pengembangan sistem menggunakan model *Prototype*.

3.3.1 Metode Pengumpulan Data

Metode pengumpulan data dilakakukan pada penelitian ini yaitu dengan cara observasi, wawancara, studi pustaka seperti jurnal dan buku-buku yang berkaitan dengan penelitian ini. Penjelasan sumber data-data tersebut ialah sebagaiberikut:

1. Observasi

   Observasi bisa disebut sebagai pengamatan, observasi ataupun pengamatan ini dilakukan secara sistematis yang dilakukan melalui pengamatan secara langsung terhadap tempat maupun objek penelitian. Dalam hal ini penulis melakukan observasi langsung kepada perusahaan GO3JEK
2. Wawancara

   Wawancara merupakan tahapan yang dilakukan dalam pengumpulan data melalui tanya jawab yang dilakukan peneliti dan narasumber yang mengetahui informasi terkait penelitian yang diangkat untuk skripsi. Dalam hal ini penulis melakukan wawancara dengan Bapak Direktur Utama GO3JEK terkait beberapa kekurangan sistem yang pernah dibangun sebelumnya.

3. Studi Pustaka

   Studi Pustaka dilakukan dengan mempelajari banyak penelitian terdahulu, baik berupa jurnal, skripsi dan juga dengan mempelajari buku-buku terkait permasalahan penelitian ini. Adapun penelitian terdahulu yang dimaksud seperti yang dibuat oleh Iwan Pahendra dengan judul "*Pembuatan Aplikasi Ojek Online Untuk Masyarakat Seputar Kampus Unsri Indralaya*", dan juga jurnal-jurnal atupun penelitian terdahulu lainnya.

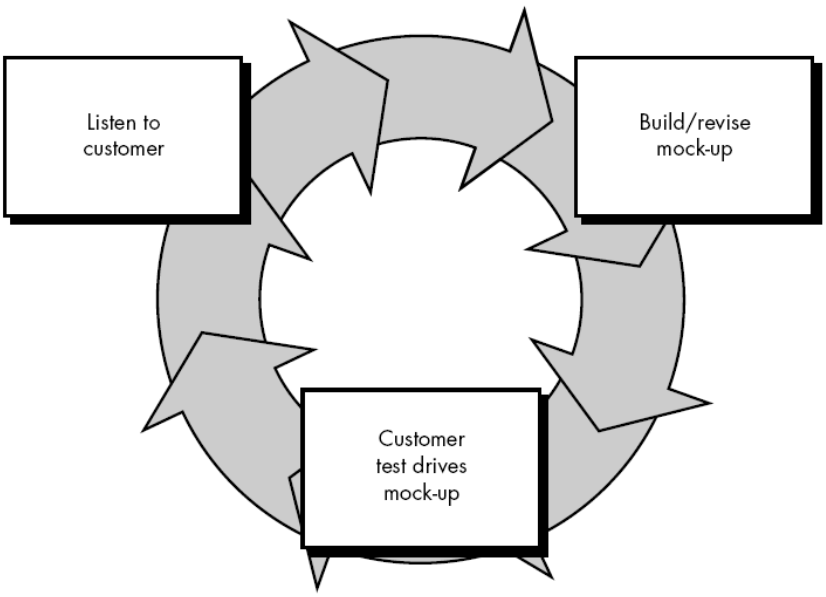
3.3.2 Jenis Data

Adapun data yang didapatkan dan dikumpulkan penulis dibagi menjadi dua jenis data yaitu:

1. Data Primer, yang merupakan data yang dikumpulkan melalui perorangan ataupun melalui instansi pada lokasi penelitian dengan cara melakukan wawancara atau observasi atau pengamatan secara langsung. Pada penelitian ini penulis melakukan observasi dan wawancara kepada Bapak Direktur Utama GO3JEK. Adapun data yang didapatkan berupa kekurangan-kekurangan sistem aplikasi jasa transportasi *online* GO3JEK yang pernah dibangun sebelumnya seperti tidak *realtime*-nya jarak antara *customer* dengan *driver*, lokasi antara *customer* dan *driver* yang salah, serta status keaktifan *driver*, dan juga perpindahan *icon driver* pada map disetiap detik dan jaraknya pada saat proses penjemputan dan pengantaran *customer*.
2. Data sekunder merupakan data yang dikumpukan melalui penelitian-penelitian terdahulu ataupun buku terkait dengan tema penelitian yang menjadi landasan ataupun acuan dalam membuat beberapa pertanyaan saat wawancara ataupun objek apa saja yang akan diamati pada saat melakukan observasi. Adapun penelitian terdahulu yang dimaksud seperti yang dibuat oleh Iwan Pahendra dengan judul "*Pembuatan Aplikasi Ojek Online Untuk Masyarakat Seputar Kampus Unsri Indralaya*", dan juga jurnal-jurnal atupun penelitian terdahulu lainnya.

3.3.3 Metode Pengembangan Sistem

Pengembangan sistem yang akan digunakan penulis pada pengembangan aplikasi ini yaitu dengan menggunakan Metode *Prototype.* Menurut (Widiyanto, 2018), metode *Prototype* merupakan sebuah proses dalam pembuatan model secara sederhana terhadap sebuah *software* yang memungkinkan pengguna memiliki gambaran mengenai sistem yang akan dibuat dan memungkinkan melakukan pengujian terhadap sistem. Metode *Prototype* sendiri memungkinkan untuk seorang pengembang dan pemakai dapat melakukan interaksi dalam proses pembuatan sistem, sehingga memudahkan pengembang dalam melakukan pemodelan terhadap sistem yang akan dibuat.



Gambar 3. 2 Metode *Prototype*

Adapun tahapan-tahapan dalam penggunaan metode *Prototype* yaitu:

1. *Listen to Customer*, dalam tahap ini adalah penulis melakukan sebuah komunikasi untuk mengidentifikasi permasalahan-permasalahan yang terdapat pada perusahaan yang berkaitan. Penulis akan melakukan identifikasi sistem, yakni:
   a. Identifikasi Sistem Berjalan

      Adapun sistem berjalan saat ini adalah masih banyaknya *bug* pada sistem tersebut, sehingga tidak memudahkan pelanggan ataupun *customer* dalam melakukan pemesanan ojek *online* ataupun transportasi *online.* Adapun *bug* yang dimaksud adalah, tidak *realtime*-nya jarak antara *customer* dengan *driver*, lokasi antara *customer* dan *driver* yang salah sehingga membuat para *customer* kesulitan dalam melakukan pemesanan, serta status keaktifan *driver*, dan juga perpindahan *icon driver* pada map disetiap detik dan jaraknya pada saat proses penjemputan dan pengantaran *customer*.
   b. Identifikasi Sistem Usulan

      Kemudian sistem usulan yang ingin diangkat adalah dengan membangun sebuah sistem transportasi *online* atupun ojek *online*

dengan menggunakan dengan *firebase* dan juga mengimplementasikan algoritma *Dijkstra*, sehingga jarak dan lokasi antara *customer* dan *driver* nantinya akan bersifat *realtime*, perpindahan *icon* yang benar, serta tampilan yang *user friendly* agar dapat memudahkan penggunanya.

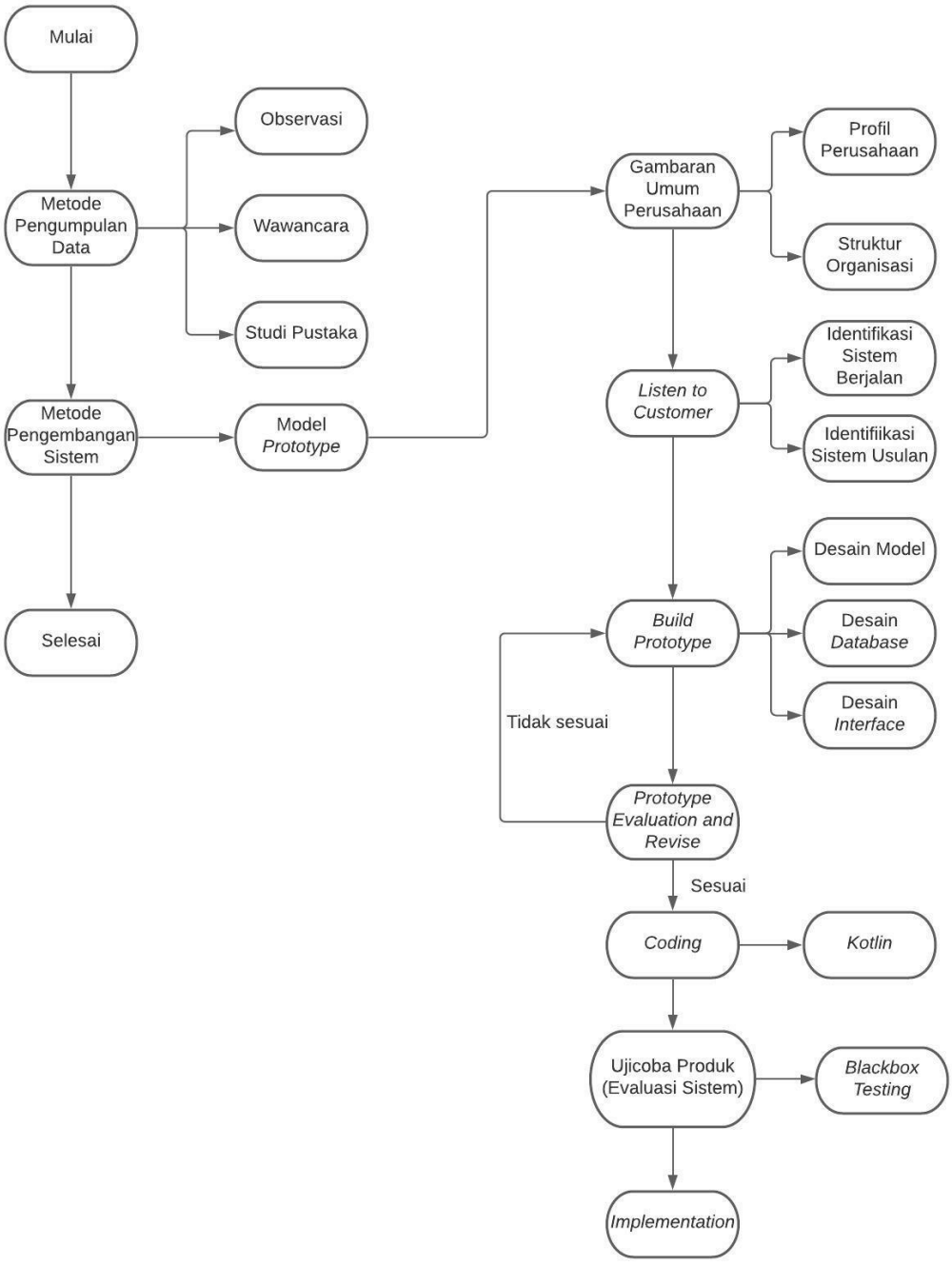
2. *Build Prototype*, pada tahap ini bertujuan untuk memberikan gambaran tampilan yang akan dikerjakan beserta gambaran tentang tahap-tahap yang akan dikerjakan. Dalam melakukan tahap ini dapat sangat membantu dalam mendefinisikan arsitektur sistem yang akan dibuat secara keseluruhan. Dalam penelitian ini penulis membagi tahapan desain menjadi bagian sebagai berikut:
   a. *Process Design*, Pada tahap desain proses ini, penulis melakukan identifikasi aktor-aktor yang terlibat dalam sistem yang mana data tersebut didapatkan dari dari tahap sebelumnya. Pada tahap desain proses ini penulis menggunakan *UML* (*Unified Model Langague*) dengan diagram model yang digunakan ialah:
      1) *Use Case Diagram*
      2) *Class Diagram*
      3) *Activity Diagram.*
      4) *Sequence Diagram*
   b. *Database Design*, yang dilandaskan dari *use case diagram* yang memiliki beberapa objek penting seperti, admin, *driver*, *customer*, dan lainnya. Maka penulis dapat merancang *database* yang memiliki tabel-tabel seperti table jenis kendaraan, tabel jenis pekerjaan, table data *driver*, dan lainnya.
   c. *Interface Design*, pada tahap ini perancangan *interface* berfungsi untuk menggambarkan tampilan pada sistem yang akan berjalan.
3. *Revise* (*Prototype Evaluation*), pada tahap ini pelanggan akan melakukan evaluasi kepada *prototype* yang sudah di desain, apabila sudah sesuai maka akan masuk ke proses selanjutnya yaitu *coding*. Apabila *prototype*

belum sesuai keinginan maka penulis akan melakukan revisi terhadap *prototype* dengan mengulang langkah-langkah sebelumnya.

4. *Coding*, merupakan penerjemahan desain dalam bahasa pemrograman yang bisa dikenali oleh komputer. Dalam penelitian ini, penulis akan melakukan *coding* menggunakan bahasa pemrograman Java melalui *Android Studio.*
5. *Testing*, pada tahap ini penulis akan menguji coba sistem yang sudah dibuat agar benar-benar sesuai dengan kebutuhan. Tahapan yang dilakukan untuk testing adalah pengetesan program untuk menemukan kesalahan-kesalahan ataupun *bug* yang mungkin terjadi.
6. *Implementation and Maintenance*, pada tahap ini sistem yang sudah siap untuk digunakan akan di implementasikan pada pengguna dan nantinya melakukan pengembangan lebih lanjut terhadap sistem tersebut.

3.3.4 Kerangka Berpikir

Berdasarkan metode pengumpulan data dan pengembangan sistem dalam penelitian ini, maka akan dirangkum dalam diagram alur kerangka berfikir sebagai berikut:



Gambar 3. 3 Kerangka Berpikir

BAB IV

HASIL DAN PEMBAHASAN

4.1 Gambaran Umum Perusahaan

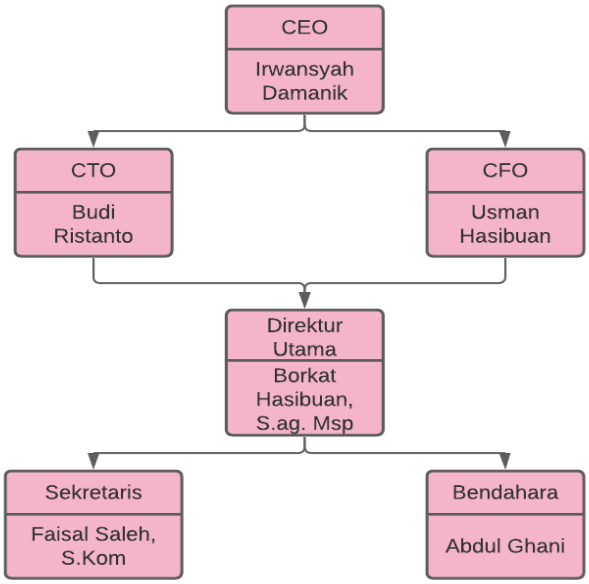
4.1.1 Profil GO3JEK

Aplikasi ojek *online* GO3JEK merupakan sebuah aplikasi yang berfokus pada penyediaan jasa transportasi berbasis *online* yang dikembangkan oleh pengusaha daerah di kota Medan. Kota Medan sendiri mempunyai berbagai potensi dan peluang yang besar untuk mengembangkan berbagai jenis usaha termasuk usaha ojek *online*. Potensi dan peluang tersebut banyak dimanfaatkan dengan baik oleh putra daerah setempat. Salah satu pengusaha di kota Medan kini mengembangkan usaha ojek *online* dengan nama GO3JEK. GO3JEK hadir di kota Medan dengan menghadirkan layanan ojek *online*, seperti kendaraan roda dua maupun roda empat, jasa pengantaran makanan, dan beragam jasa lainnya.

GO3JEK adalah perusahaan di Medan Sumatra Utara yang menggunakan teknologi berbasis aplikasi untuk memberikan kemudahan akses bagi penyedia jasa dan pengguna jasa dalam memesan transportasi *online* roda dua maupun roda empat, pengiriman barang, jasa pemesanan makanan, serta beragam jasa lainnya. GO3JEK bermitra dengan para pengendara ojek guna menyediakan layanan ojek *online* terpercaya dengan tarif terjangkau. Dengan membuka usaha ojek *online*, GO3JEK ingin memberikan kemudahan bagi masyarakat kota medan serta membuka lapangan kerja untuk meningkatkan perekonomian kota Medan.

4.1.2 Struktur Organisasi

Sebuah organisasi ataupun perusahaan tentunya harus mempunyai struktur organisasi dalam sebuah pembagian tugas yang nantinya akan dikelompokkan dan dikoordinasikan secara formal. Adapun struktur organisasi pada perusahaan GO3JEK yaitu:



Gambar 4. 1 Struktur Organisai GO3JEK

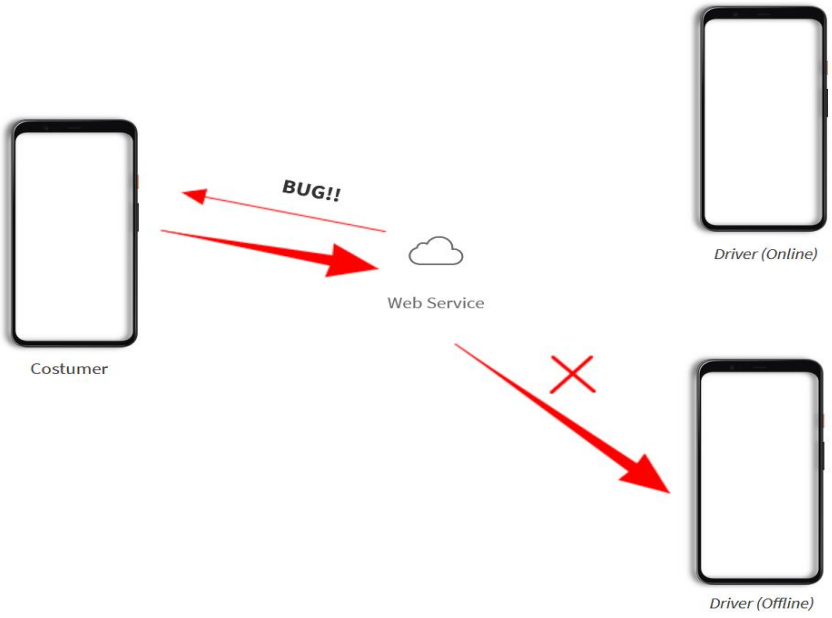
4.2 *Listen to Customer*

Pada tahap ini, penulis melakukan interaksi komunikasi kepada pimpinan GO3JEK untuk mengindentifikasi kebutuhan dari sistem yang akan dibangun nantinya dengan mengumpulkan data-data yang terkait. Pada tahap *listen to customer* ini akan menjelaskan tentang analisis sistem yang sedang berjalan dan juga analisis sistem usulan.

4.2.1 Analisis Sistem Berjalan

Sebagai tolak ukur untuk pengembangan sistem selanjutnya, maka diperlukan analisis terhadap sistem yang sedang berjalan. Adapun sistem yang sedang berjalan pada perusahaan GO3JEK yang beralamatkan di Gedung Graha GO3JEK Indonesia, Tj. Sari, Kec. Medan Selayang, Kota Medan, Sumatera Utara Kode Pos 20122, penulis mengenalisis bahwa aplikasi GO3JEK masih menggunakan *web service* tersendiri sebagai pihak ketiga dalam menjembatani komunikasi antara costumer dan driver. *Web service* yang digunakan oleh GO3JEK tersebutlah yang menyebabkan terjadinya *bug* seperti status keaktifan

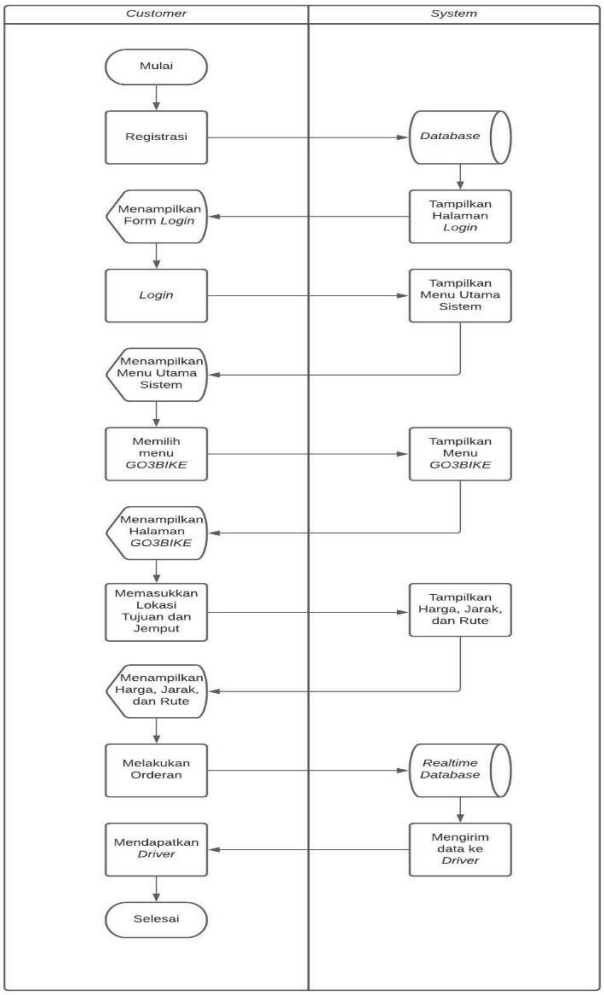
driver yang tidak ataupun belum *real time*, posisi *driver* yang tidak sesuai ataupun tidak *real time* sehingga menyebabkan kesulitan yang dialami oleh *customer* dalam melakukan pemesanan jasa transportasi *online*. Dalam pemetaannya, hingga penentuan jarak tempuh pada aplikasi transportasi *online* GO3JEK perusahaan GOJEK menggunakan fitur yang disediakan layanan *google* sehingga perusahaan tersebut mempunyai pengeluaran dana yang tidak sedikit setiap bulannya yang harus rutin dibayar kepada pihak *google.*



Gambar 4. 2 Analisis Sistem Sedang Berjalan

4.2.2 Analisis Sistem Usulan

Kelemahan dari sistem berjalan sehingga menyebabkan *bug* seperti status keaktifan *driver* yang tidak ataupun belum *real time*, posisi *driver* yang tidak sesuai ataupun tidak *real time* sehingga menyebabkan kesulitan yang dialami oleh *customer* dalam melakukan pemesanan jasa transportasi *online*, maka dibuatlah sebuah sistem usulan yaitu pembuatan aplikasi jasa transportasi *online* menggunakan *firebase* sebagai penyimpanan *real time database* dan juga sebagai pihak ketiga dalam menjembatani komunikasi antara *customer* dan *driver*. Berikut *flowmap* pada sistem usulan yang akan dibangun:

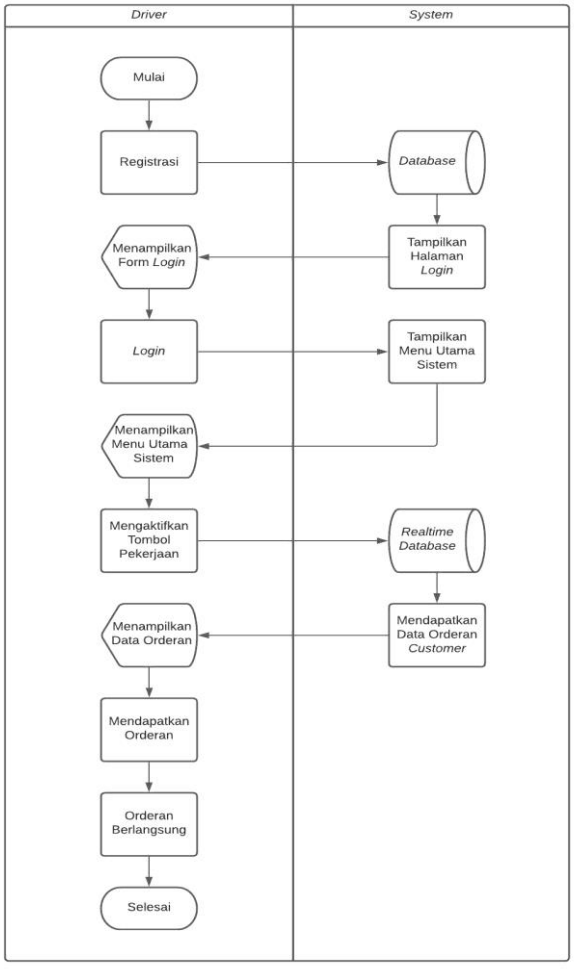


Gambar 4. 3 *Flowmap* Sistem Usulan Dari Sisi *Customer*

Pada sistem usulan gambar 4.3 merupakan sebuah *flow* ataupun alur dari sistem yang akan dibuat dari sisi *customer*. Adapun penjelasannya adalah sebagai berikut:

1. *Customer* membuka aplikasi
2. *Customer* melakukan *registrasi* ataupun pendaftaran akun kedalam sistem
3. Kemudian *customer* melakukan proses *login* kedalam sistem dengan menggunakan akun yang sudah didaftarkan melalui tahap *registrasi*
4. Setelah *customer* melakukan *registrasi*, *customer* akan menuju halaman menu utama pada aplikasi

5. Setelah berada pada halaman menu utama, *customer* memilih menu *GO3BIKE*
6. *Customer* memasukkan lokasi antar dan penjemputan
7. *Customer* melakukan pemesanan
8. Sistem merekam dan mengolah data pesanan yang dibuat *customer* kedalam *firebase*, kemudian meneruskan data tersebut ke *driver* yang terdeteksi sedang *online*
9. *Customer* mendapatkan *driver*, kemudian proses orderan pun berlangsung
10. Proses selesai

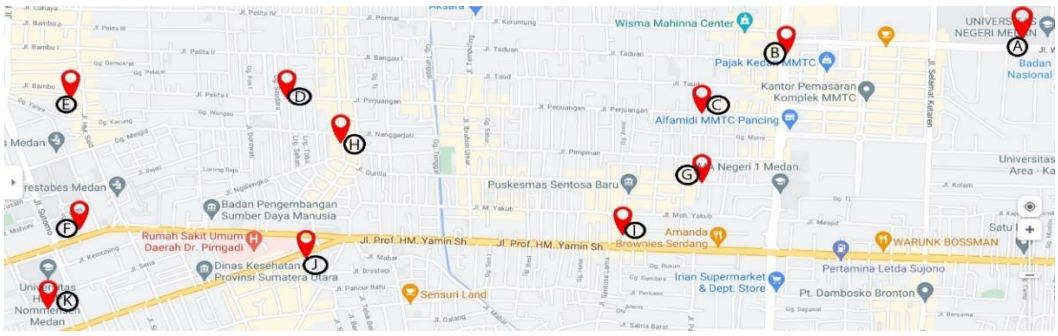


Gambar 4. 4 *Flowmap* Sistem Usulan Dari Sisi *Driver*

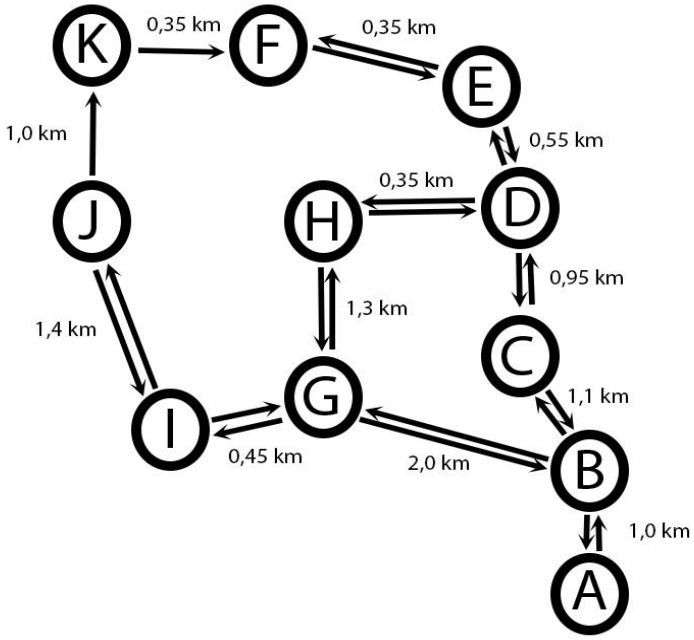
Selanjutnya, pada gambar 4.4 merupakan alur ataupun *flow* dari sistem usulan dari sisi *driver*. Adapun penjelasannya sebagai berikut:

1. *Driver* membuka aplikasi
2. *Driver* melakukan *registrasi* atupun pendaftaran akun kedalam sistem
3. Kemudian *driver* melakukan proses *login* kedalam sistem dengan menggunakan akun yang sudah didaftarkan melalui tahap *registrasi*
4. Setelah *driver* melakukan *registrasi*, *driver* akan menuju halaman utama pada aplikasi
5. *Driver* melakukan pengaktifan *GPS* dan juga tombol pekerjaan pada halaman utama
6. *Sistem* akan merekam data status keaktifan *driver* kedalam *firebase*
7. Sistem akan meneruskan data orderan yang masuk dari *customer* kepada *driver*.
8. Setelah mendapatkan orderan masuk, sistem akan menampilkan data-data orderan yang dikirim *customer* kepada *driver*, kemudian proses orderan pun berlangsung
9. Proses selesai

Pada penelitian ini, aplikasi jasa transportasi *online* yang akan dibangun nantinya menggunakan algoritma *dijkstra* sebagai penentuan rute terpendek dan juga jarak yang akan ditempuh. Adapun contoh kasus yang diangkat penulis mengenai pencarian rute terpendek seperti tertera pada gambar berikut ini:



Gambar 4. 5 Contoh Kasus Algoritma *Dijkstra*



Gambar 4. 6 *Graf* Contoh Kasus Algoritma *Dijkstra*

Seperti yang tertera pada gambar 4.6 diatas yang merupakan sebuah *graf* berarah dan berbobot yang merupakan hasil penerjemahan dari gambar 4.5, dapat dilihat pada kasus yang diangkat penulis menetapkan *node* ataupun *vertex* berjumlah sebanyak 11 titik *node*. Adapun *node-node* tersebut yakni:

Tabel 4. 1 Daftar Lokasi Rute Terpendek

Node	**Daerah/Jalan**
A	UNIMED
B	Jalan Willem Iskandar
C	Jalan Perjuangan
D	Jalan Pelita I
E	Jalan H. M Said
F	Kampus Pasca Sarjana UINSU
G	Jalan Gurilla
H	Jalan. Rakyat
I	Jalan Sentosa Baru

J	Jalan Prof. M. Yamin
K	Universitas HKBP Nommensen

Dari tabel 4.1, terdapat jarak antar *vertex* ataupun *node* yang sudah ditentukan penulis berdasarkan jarak yang tertera pada *Google Maps* menggunakan satuan jarak kilometer (km) yakni:

Tabel 4. 2 Tabel Jarak Antara *Node*

Node Awal	Node Tujuan	Jarak (Kilometer)
A	B	1,0 km
B	A	1,0 km
B	C	1,1 km
B	G	2,0 km
C	B	1,1 km
C	D	0,95 km
D	C	0,95 km
D	E	0,55 km
D	H	0,35 km
E	D	0,55 km
E	F	0,35 km
F	E	0,35 km
G	B	2,0 km
G	H	1,3 km
G	I	0,45 km
H	D	0,35 km
H	G	1,3 km
I	G	0,45 km
I	J	1,4 km
J	I	1,4 km
J	K	1,0 km
K	F	0,35 km

4.2.2.1 Perhitungan Manual Algoritma *Dijkstra*

Adapun perhitungan manual algoritma *dijkstra* yang telah dihitung penulis menurut contoh kasus yang diangkat penulis seperti yang tertera pada gambar 4.5 sebagai titik-titik lokasi yang tertera pada *google maps*, gambar 4.6 yang merupakan hasil *garph* dari terjemahan pada gambar 4.5, dan juga tabel 4.1 yang

didalamnya terdapat daftar lokasi dan juga jarak antara satu *vertex* dengan *vertex* yang lainnya adalah sebagai berikut:

1. *Node* A

Tabel 4. 3 Hasil Perhitungan Manual *Dijkstra Node* A

c	A	B	C	D	E	F	G	H	I	J	K
	0_A	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
A	0_A	$1{,}0_A$	∞	∞	∞	∞	∞	∞	∞	∞	∞
B	0_A	$1{,}0_A$	$2{,}1_B$	∞	∞	∞	$3{,}0_B$	∞	∞	∞	∞
C	0_A	$1{,}0_A$	$2{,}1_B$	$3{,}05_C$	∞	∞	$3{,}0_B$	∞	∞	∞	∞
G	0_A	$1{,}0_A$	$2{,}1_B$	$3{,}05_C$	∞	∞	$3{,}0_B$	$4{,}3_G$	$3{,}45_G$	∞	∞
D	0_A	$1{,}0_A$	$2{,}1_B$	$3{,}05_C$	$3{,}6_D$	∞	$3{,}0_B$	$3{,}4_D$	$3{,}45_G$	∞	∞
H	0_A	$1{,}0_A$	$2{,}1_B$	$3{,}05_C$	$3{,}6_D$	∞	$3{,}0_B$	$3{,}4_D$	$3{,}45_G$	∞	∞
I	0_A	$1{,}0_A$	$2{,}1_B$	$3{,}05_C$	$3{,}6_D$	∞	$3{,}0_B$	$3{,}4_D$	$3{,}45_G$	$4{,}85_I$	∞
E	0_A	$1{,}0_A$	$2{,}1_B$	$3{,}05_C$	$3{,}6_D$	$3{,}95_E$	$3{,}0_B$	$3{,}4_D$	$3{,}45_G$	$4{,}85_I$	∞
F	0_A	$1{,}0_A$	$2{,}1_B$	$3{,}05_C$	$3{,}6_D$	$3{,}95_E$	$3{,}0_B$	$3{,}4_D$	$3{,}45_G$	$4{,}85_I$	∞
J	0_A	$1{,}0_A$	$2{,}1_B$	$3{,}05_C$	$3{,}6_D$	$3{,}95_E$	$3{,}0_B$	$3{,}4_D$	$3{,}45_G$	$4{,}85_I$	$5{,}85_J$

Keterangan:

c = current

∞ = tak hingga (tidak terhubung)

d = distance

Tabel 4. 4 Rumus Perhitungan Manual *Dijkstra Node* A

Node	**Rumus (km)**	***Node***	**Rumus (km)**
A	= *c*+*d* = 0 + 0 = 0 km	G	= *c*+*d* = B + 2,0 = 1,0 + 2,0 = 3,0 km
B	= *c* + *d* = A + 1,0 = 0 + 1,0 = 1,0 km	H	= *c*+*d* = D + 0,35 = 3,05 + 0,35 = 3,4 km
C	= *c*+*d* = B + 1,1	I	= *c*+*d* = G + 0,45

	= 1,0 + 1,1 = 2,1 km		= 3,0 + 0,45 = 3,45 km
D	= *c*+*d* = C + 0,95 = 2,1 + 0,95 = 3,05 km	J	= *c*+*d* = I + 1,4 = 3,45 + 1,4 = 4,85 km
E	= *c*+*d* = D + 0,55 = 3,05 + 0.55 = 3,6 km	K	= *c*+*d* = J + 1,0 = 4,85 + 1,0 = 5,85 km
F	= *c*+*d* = E + 0,35 = 3,6 + 0,35 = 3,95 km		

Berdasarkan perhitungan *node* A, dapat disimpulakan *node* A mempunyai panjang lintasan masing-masing, yakni:

a. A → B = dengan panjang lintasan 1,0km
b. A → C = A-B-C, dengan panjang lintasan 2,1km
c. A → D = A-B-C-D, dengan panjang lintasan 3,05km
d. A → E = A-B-C-D-E, dengan panjang lintasan 3,6km
e. A → F = A-B-C-D-E-F, dengan panjang lintasan 3,95km
f. A → G = A-B-G, dengan panjang lintasan 3,0km
g. A → H = A-B-C-D-H, dengan panjang lintasan 3,4km
h. A → I = A-B-G-I, dengan panjang lintasan 3,45km
i. A → J = A-B-G-I-J, dengan panjang lintasan 4,85km
j. A → K = A-B-G-I-J-K, dengan panjang lintasan 5,85km

2. *Node* B

Tabel 4. 5 Hasil Perhitungan Manual *Dijkstra Node* B

c	A	B	C	D	E	F	G	H	I	J	K
	∞	0_B	∞	∞	∞	∞	∞	∞	∞	∞	∞
B	$1,0_B$	0_B	$1,1_B$	∞	∞	∞	$2,0_B$	∞	∞	∞	∞
A	$1,0_B$	0_B	$1,1_B$	∞	∞	∞	$2,0_B$	∞	∞	∞	∞
C	$1,0_B$	0_B	$1,1_B$	$2,05_C$	∞	∞	$2,0_B$	∞	∞	∞	∞

G	$1,0_B$	0_B	$1,1_B$	$2,05_C$	∞	∞	$2,0_B$	$3,3_G$	$2,45_G$	∞	∞
D	$1,0_B$	0_B	$1,1_B$	$2,05_C$	$2,6_D$	∞	$2,0_B$	$2,4_D$	$2,45_G$	∞	∞
H	$1,0_B$	0_B	$1,1_B$	$2,05_C$	$2,6_D$	∞	$2,0_B$	$2,4_D$	$2,45_G$	∞	∞
I	$1,0_B$	0_B	$1,1_B$	$2,05_C$	$2,6_D$	∞	$2,0_B$	$2,4_D$	$2,45_G$	$3,85_I$	∞
E	$1,0_B$	0_B	$1,1_B$	$2,05_C$	$2,6_D$	$2,95_E$	$2,0_B$	$2,4_D$	$2,45_G$	$3,85_I$	∞
F	$1,0_B$	0_B	$1,1_B$	$2,05_C$	$2,6_D$	$2,95_E$	$2,0_B$	$2,4_D$	$2,45_G$	$3,85_I$	∞
J	$1,0_B$	0_B	$1,1_B$	$2,05_C$	$2,6_D$	$2,95_E$	$2,0_B$	$2,4_D$	$2,45_G$	$3,85_I$	$4,85_J$

Keterangan:

c = current

∞ = tak hingga (tidak terhubung)

d = distance

Tabel 4. 6 Rumus Perhitungan Manual *Dijkstra Node* B

Node	**Rumus (km)**	***Node***	**Rumus (km)**
A	= *c*+*d* = B + 1,0 = 0 + 1,0 = 1,0 km	G	= *c*+*d* = B + 2,0 = 0 + 2,0 = 2,0 km
B	= *c* + *d* = 0 + 0 = 0 km	H	= *c*+*d* = D + 0,35 = 2,05 + 0,35 = 2,4 km
C	= *c*+*d* = B + 1,1 = 0 + 1,1 = 1,1 km	I	= *c*+*d* = G + 0,45 = 2,0 + 0,45 = 2,45 km
D	= *c*+*d* = C + 0,95 = 1,1 + 0,95 = 2,05 km	J	= *c*+*d* = I + 1,4 = 2,45 + 1,4 = 3,85 km
E	= *c*+*d* = D + 0,55 = 2,05 + 0.55 = 2,6 km	K	= *c*+*d* = J + 1,0 = 3,85 + 1,0 = 4,85 km
F	= *c*+*d* = E + 0,35 = 2,6 + 0,35 = 2,95 km		

Berdasarkan perhitungan *node* B, dapat disimpulakan *node* B mempunyai panjang lintasan masing-masing, yakni:

a. B → A = dengan panjang lintasan 1,0km
b. B → C = dengan panjang lintasan 1,1km
c. B → D = B-C-D, dengan panjang lintasan 2,05km
d. B → E = B-C-D-E, dengan panjang lintasan 2,6km
e. B → F = B-C-D-E-F, dengan panjang lintasan 2,95km
f. B → G = dengan panjang lintasan 2,0km
g. B → H = B-C-D-H, dengan panjang lintasan 2,4km
h. B → I = B-G-I, dengan panjang lintasan 2,45km
i. B → J = B-G-I-J, dengan panjang lintasan 3,85km
j. B → K = B-G-I-J-K, dengan panjang lintasan 4,85km

3. *Node* C

Tabel 4. 7 Hasil Perhitungan Manual *Dijkstra Node* C

c	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**
	∞	∞	0_C	∞	∞	∞	∞	∞	∞	∞	∞
C	∞	$1{,}1_C$	0_C	$0{,}95_C$	∞	∞	∞	∞	∞	∞	∞
D	∞	$1{,}1_C$	0_C	$0{,}95_C$	$1{,}5_D$	∞	∞	$1{,}3_D$	∞	∞	∞
B	$2{,}1_B$	$1{,}1_C$	0_C	$0{,}95_C$	$1{,}5_D$	∞	$3{,}1_B$	$1{,}3_D$	∞	∞	∞
H	$2{,}1_B$	$1{,}1_C$	0_C	$0{,}95_C$	$1{,}5_D$	∞	$2{,}6_H$	$1{,}3_D$	∞	∞	∞
E	$2{,}1_B$	$1{,}1_C$	0_C	$0{,}95_C$	$1{,}5_D$	$1{,}85_E$	$2{,}6_H$	$1{,}3_D$	∞	∞	∞
F	$2{,}1_B$	$1{,}1_C$	0_C	$0{,}95_C$	$1{,}5_D$	$1{,}85_E$	$2{,}6_H$	$1{,}3_D$	∞	∞	∞
A	$2{,}1_B$	$1{,}1_C$	0_C	$0{,}95_C$	$1{,}5_D$	$1{,}85_E$	$2{,}6_H$	$1{,}3_D$	∞	∞	∞
G	$2{,}1_B$	$1{,}1_C$	0_C	$0{,}95_C$	$1{,}5_D$	$1{,}85_E$	$2{,}6_H$	$1{,}3_D$	$3{,}05_G$	∞	∞
I	$2{,}1_B$	$1{,}1_C$	0_C	$0{,}95_C$	$1{,}5_D$	$1{,}85_E$	$2{,}6_H$	$1{,}3_D$	$3{,}05_G$	$4{,}45_I$	∞
J	$2{,}1_B$	$1{,}1_C$	0_C	$0{,}95_C$	$1{,}5_D$	$1{,}85_E$	$2{,}6_H$	$1{,}3_D$	$3{,}05_G$	$4{,}45_I$	$5{,}45_J$

Keterangan:

c = *current*

∞ = tak hingga (tidak terhubung)

d = distance

Tabel 4. 8 Rumus Perhitungan Manual *Dijkstra Node* C

Node	**Rumus (km)**	***Node***	**Rumus (km)**
A	= *c*+*d* = B + 1,0 = 1,1 + 1,0 = 2,1 km	G	= *c*+*d* = H + 1,3 = 1,3 + 1,3 = 2,6 km
B	= *c* + *d* = C + 1,1 = 0 + 1,1 = 1,1 km	H	= *c*+*d* = D + 0,35 = 0,95 + 0,35 = 1,3 km
C	= *c*+*d* = 0 + 0 = 0 km	I	= *c*+*d* = G + 0,45 = 2,6 + 0,45 = 3,05 km
D	= *c*+*d* = C + 0,95 = 0 + 0,95 = 0,95 km	J	= *c*+*d* = I + 1,4 = 3,05 + 1,4 = 4,45 km
E	= *c*+*d* = D + 0,55 = 0,95 + 0.55 = 1,5 km	K	= *c*+*d* = J + 1,0 = 4,45 + 1,0 = 5,45 km
F	= *c*+*d* = E + 0,35 = 1,5 + 0,35 = 1,85 km		

Berdasarkan perhitungan *node* C, dapat disimpulakan *node* C mempunyai panjang lintasan masing-masing, yakni:

a. C → A = C-B-Adengan panjang lintasan 2,1km
b. C → B = dengan panjang lintasan 1,1km
c. C → D = dengan panjang lintasan 0,95km
d. C → E = C-D-E, dengan panjang lintasan 1,5km
e. C → F = C-D-E-F, dengan panjang lintasan 1,85km
f. C → G = C-D-H-G, dengan panjang lintasan 2,6km

g. C → H = C-D-H, dengan panjang lintasan 1,3km

h. C → I = C-D-H-G-I, dengan panjang lintasan 3,05km

i. C → J = C-D-H-G-I-J, dengan panjang lintasan 4,45km

j. C → K = C-D-H-G-I-J-K, dengan panjang lintasan 5,45km

4. *Node* D

Tabel 4. 9 Hasil Perhitungan Manual *Dijkstra Node* D

c	A	B	C	D	E	F	G	H	I	J	K
	∞	∞	∞	0_D	∞	∞	∞	∞	∞	∞	∞
D	∞	∞	$0{,}95_D$	0_D	$0{,}55_D$	∞	∞	$0{,}35_D$	∞	∞	∞
H	∞	∞	$0{,}95_D$	0_D	$0{,}55_D$	∞	$1{,}65_H$	$0{,}35_D$	∞	∞	∞
E	∞	∞	$0{,}95_D$	0_D	$0{,}55_D$	$0{,}9_E$	$1{,}65_H$	$0{,}35_D$	∞	∞	∞
F	∞	∞	$0{,}95_D$	0_D	$0{,}55$	$0{,}9_E$	$1{,}65_H$	$0{,}35_D$	∞	∞	∞
C	∞	$2{,}05_C$	$0{,}95_D$	0_D	$0{,}55_D$	$0{,}9_E$	$1{,}65_H$	$0{,}35_D$	∞	∞	∞
G	∞	$2{,}05_C$	$0{,}95_D$	0_D	$0{,}55_D$	$0{,}9_E$	$1{,}65_H$	$0{,}35_D$	$2{,}1_G$	∞	∞
B	$3{,}05_B$	$2{,}05_C$	$0{,}95_D$	0_D	$0{,}55_D$	$0{,}9_E$	$1{,}65_H$	$0{,}35_D$	$2{,}1_G$	∞	∞
I	$3{,}05_B$	$2{,}05_C$	$0{,}95_D$	0_D	$0{,}55_D$	$0{,}9_E$	$1{,}65_H$	$0{,}35_D$	$2{,}1_G$	$3{,}5_I$	∞
A	$3{,}05_B$	$2{,}05_C$	$0{,}95_D$	0_D	$0{,}55_D$	$0{,}9_E$	$1{,}65_H$	$0{,}35_D$	$2{,}1_G$	$3{,}5_I$	∞
J	$3{,}05_B$	$2{,}05_C$	$0{,}95_D$	0_D	$0{,}55_D$	$0{,}9_E$	$1{,}65_H$	$0{,}35_D$	$2{,}1_G$	$3{,}5_I$	$4{,}5_J$

Keterangan:

c = *current*

∞ = tak hingga (tidak terhubung)

d = distance

Tabel 4. 10 Rumus Perhitungan Manual *Dijkstra Node* D

Node	**Rumus (km)**	***Node***	**Rumus (km)**
A	= *c+d* = B + 1,0 = 2,05 + 1,0 = 3,05 km	G	= *c+d* = H + 1,3 = 0,35 + 1,3 = 1,65 km

B	= *c* + *d* = C + 1,1 = 0,95+ 1,1 = 2,05 km	H	= *c*+*d* = D + 0,35 = 0 + 0,35 = 0,35 km
C	= *c*+*d* = D + 0,95 = 0 + 0,95 = 0,95 km	I	= *c*+*d* = G + 0,45 = 1,65 + 0,45 = 2,1 km
D	= *c*+*d* = 0 + 0 = 0 km	J	= *c*+*d* = I + 1,4 = 2,1 + 1,4 = 3,5 km
E	= *c*+*d* = D + 0,55 = 0 + 0.55 = 0,55 km	K	= *c*+*d* = J + 1,0 = 3,5 + 1,0 = 4,5 km
F	= *c*+*d* = E + 0,35 = 0,55 + 0,35 = 0,9 km		

Berdasarkan perhitungan *node* D, dapat disimpulakan *node* D mempunyai panjang lintasan masing-masing, yakni:

a. D → A = D-C-B-A, dengan panjang lintasan 3,05km
b. D → B = D-C-B, dengan panjang lintasan 2,05km
c. D → C = dengan panjang lintasan 0,95km
d. D → E = dengan panjang lintasan 0,55km
e. D → F = D-E-F, dengan panjang lintasan 0,9km
f. D → G = D-H-G, dengan panjang lintasan1,65km
g. D → H = dengan panjang lintasan 0,35km
h. D → I = D-H-G-I, dengan panjang lintasan 2,1km
i. D → J = D-H-G-I-J, dengan panjang lintasan 3,5km
j. D → K = D-H-G-I-J-K, dengan panjang lintasan 4,5km

5. *Node* E

Tabel 4. 11 Hasil Perhitungan Manual *Dijkstra Node* E

c	A	B	C	D	E	F	G	H	I	J	K
	∞	∞	∞	∞	0_E	∞	∞	∞	∞	∞	∞
E	∞	∞	∞	$0{,}55_E$	0_E	$0{,}35_E$	∞	∞	∞	∞	∞
F	∞	∞	∞	$0{,}55_E$	0_E	$0{,}35_E$	∞	∞	∞	∞	∞
D	∞	∞	$1{,}5_D$	$0{,}55_E$	0_E	$0{,}35_E$	∞	$0{,}9_D$	∞	∞	∞
H	∞	∞	$1{,}5_D$	$0{,}55_E$	0_E	$0{,}35_E$	$2{,}2_H$	$0{,}9_D$	∞	∞	∞
C	∞	$2{,}6_C$	$1{,}5_D$	$0{,}55_E$	0_E	$0{,}35_E$	$2{,}2_H$	$0{,}9_D$	∞	∞	∞
G	∞	$2{,}6_C$	$1{,}5_D$	$0{,}55_E$	0_E	$0{,}35_E$	$2{,}2_H$	$0{,}9_D$	$2{,}65_G$	∞	∞
B	$3{,}6_B$	$2{,}6_C$	$1{,}5_D$	$0{,}55_E$	0_E	$0{,}35_E$	$2{,}2_H$	$0{,}9_D$	$2{,}65_G$	∞	∞
I	$3{,}6_B$	$2{,}6_C$	$1{,}5_D$	$0{,}55_E$	0_E	$0{,}35_E$	$2{,}2_H$	$0{,}9_D$	$2{,}65_G$	$4{,}05_I$	∞
A	$3{,}6_B$	$2{,}6_C$	$1{,}5_D$	$0{,}55_E$	0_E	$0{,}35_E$	$2{,}2_H$	$0{,}9_D$	$2{,}65_G$	$4{,}05_I$	∞
J	$3{,}6_B$	$2{,}6_C$	$1{,}5_D$	$0{,}55_E$	0_E	$0{,}35_E$	$2{,}2_H$	$0{,}9_D$	$2{,}65_G$	$4{,}05_I$	$5{,}05_J$

Keterangan:

c = *current*

∞ = tak hingga (tidak terhubung)

d = distance

Tabel 4. 12 Rumus Perhitungan Manual *Dijkstra Node* E

Node	**Rumus (km)**	***Node***	**Rumus (km)**
A	= *c*+*d* = B + 1,0 = 2,6 + 1,0 = 3,6 km	G	= *c*+*d* = H + 1,3 = 0,9 + 1,3 = 2,2 km
B	= *c* + *d* = C + 1,1 = 1,5+ 1,1 = 2,6 km	H	= *c*+*d* = D + 0,35 = 0,55 + 0,35 = 0,9 km
C	= *c*+*d* = D + 0,95 = 0,55 + 0,95 = 1,5 km	I	= *c*+*d* = G + 0,45 = 2,2 + 0,45 = 2,65 km

D	= *c+d* = E + 0,55 = 0 + 0,55 = 0,55 km	J	= *c+d* = I + 1,4 = 2,65 + 1,4 = 4,05 km
E	= *c+d* = 0 + 0 = 0 km	K	= *c+d* = J + 1,0 = 4,05 + 1,0 = 5,05 km
F	= *c+d* = E + 0,35 = 0 + 0,35 = 0,35 km		

Berdasarkan perhitungan *node* E, dapat disimpulakan *node* E mempunyai panjang lintasan masing-masing, yakni:

a. E → A = E-D-C-B-A dengan panjang lintasan 3,6km
b. E → B = E-D-C-B, dengan panjang lintasan 2,6km
c. E → C = E-D-C, dengan panjang lintasan 1,5km
d. E → D = dengan panjang lintasan 0,55km
e. E → F = dengan panjang lintasan 0,35km
f. E → G = E-D-H-G, dengan panjang lintasan 2,2km
g. E → H = E-D-H, dengan panjang lintasan 0,9km
h. E → I = E-D-H-G-I, dengan panjang lintasan 2,65km
i. E → J = E-D-H-G-I-J, dengan panjang lintasan 4,05km
j. E → K = E-D-H-G-I-J-K, dengan panjang lintasan 5,05km

6. *Node* F

Tabel 4. 13 Hasil Perhitungan Manual *Dijkstra Node* F

c	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**
	∞	∞	∞	∞	∞	0_F	∞	∞	∞	∞	∞
F	∞	∞	∞	∞	$0{,}35_F$	0_F	∞	∞	∞	∞	∞
E	∞	∞	∞	$0{,}9_E$	$0{,}35_F$	0_F	∞	∞	∞	∞	∞
D	∞	∞	$1{,}85_D$	$0{,}9_E$	$0{,}35_F$	0_F	∞	$1{,}25_D$	∞	∞	∞

H	∞	∞	$1{,}85_D$	$0{,}9_E$	$0{,}35_F$	0_F	$2{,}55_H$	$1{,}25_D$	∞	∞	∞
C	∞	$2{,}95_C$	$1{,}85_D$	$0{,}9_E$	$0{,}35_F$	0_F	$2{,}55_H$	$1{,}25_D$	∞	∞	∞
G	∞	$2{,}95_C$	$1{,}85_D$	$0{,}9_E$	$0{,}35_F$	0_F	$2{,}55_H$	$1{,}25_D$	$3{,}0_G$	∞	∞
B	$3{,}95_B$	$2{,}95_C$	$1{,}85_D$	$0{,}9_E$	$0{,}35_F$	0_F	$2{,}55_H$	$1{,}25_D$	$3{,}0_G$	∞	∞
I	$3{,}95_B$	$2{,}95_C$	$1{,}85_D$	$0{,}9_E$	$0{,}35_F$	0_F	$2{,}55_H$	$1{,}25_D$	$3{,}0_G$	$4{,}4_I$	∞
A	$3{,}95_B$	$2{,}95_C$	$1{,}85_D$	$0{,}9_E$	$0{,}35_F$	0_F	$2{,}55_H$	$1{,}25_D$	$3{,}0_G$	$4{,}4_I$	∞
J	$3{,}95_B$	$2{,}95_C$	$1{,}85_D$	$0{,}9_E$	$0{,}35_F$	0_F	$2{,}55_H$	$1{,}25_D$	$3{,}0_G$	$4{,}4_I$	$5{,}4_J$

Keterangan:

c = *current*

∞ = tak hingga (tidak terhubung)

d = distance

Tabel 4. 14 Hasil Perhitungan Manual *Dijkstra Node* F

Node	**Rumus (km)**	***Node***	**Rumus (km)**
A	= *c*+*d* = B + 1,0 = 2,95 + 1,0 = 3,95 km	G	= *c*+*d* = H + 1,3 = 1,25 + 1,3 = 2,55 km
B	= *c* + *d* = C + 1,1 = 1,85+ 1,1 = 2,95 km	H	= *c*+*d* = D + 0,35 = 0,9 + 0,35 = 1,25 km
C	= *c*+*d* = D + 0,95 = 0,9 + 0,95 = 1,85 km	I	= *c*+*d* = G + 0,45 = 2,55 + 0,45 = 3 km
D	= *c*+*d* = E + 0,55 = 0,35 + 0,55 = 0,9 km	J	= *c*+*d* = I + 1,4 = 3 + 1,4 = 4,4 km
E	= *c*+*d* = F + 0,35 = 0 + 0,35 = 0,35 km	K	= *c*+*d* = J + 1,0 = 4,4 + 1,0 = 5,4 km
F	= *c*+*d* = 0 + 0 = 0 km		

Berdasarkan perhitungan *node* F, dapat disimpulakan *node* F mempunyai panjang lintasan masing-masing, yakni:

a. F → A = F-E-D-C-B-A dengan panjang lintasan 3,95km
b. F → B = F-E-D-C-B, dengan panjang lintasan 2,95km
c. F → C = F-E-D-C, dengan panjang lintasan 1,85km
d. F → D = F-E-D, dengan panjang lintasan 0,9km
e. F → E = dengan panjang lintasan 0,35km
f. F → G = F-E-D-H-G, dengan panjang lintasan 2,55km
g. F → H = F-E-D-H, dengan panjang lintasan 1,25km
h. F → I = F-E-D-H-G-I, dengan panjang lintasan 3,0km
i. F → J = F-E-D-H-G-I-J, dengan panjang lintasan 4,4km
j. F → K = F-E-D-H-G-I-K, dengan panjang lintasan 5,4km

7. *Node* G

Tabel 4. 15 Hasil Perhitungan Manual *Dijkstra Node* G

c	A	B	C	D	E	F	G	H	I	J	K
	∞	∞	∞	∞	∞	∞	0_G	∞	∞	∞	∞
G	∞	$2{,}0_G$	∞	∞	∞	∞	0_G	$1{,}3_G$	$0{,}45_G$	∞	∞
I	∞	$2{,}0_G$	∞	∞	∞	∞	0_G	$1{,}3_G$	$0{,}45_G$	$1{,}85_I$	∞
H	∞	$2{,}0_G$	∞	$1{,}65_H$	∞	∞	0_G	$1{,}3_G$	$0{,}45_G$	$1{,}85_I$	∞
D	∞	$2{,}0_G$	$2{,}6_D$	$1{,}65_H$	$2{,}2_D$	∞	0_G	$1{,}3_G$	$0{,}45_G$	$1{,}85_I$	∞
J	∞	$2{,}0_G$	$2{,}6_D$	$1{,}65_H$	$2{,}2_D$	∞	0_G	$1{,}3_G$	$0{,}45_G$	$1{,}85_I$	$2{,}85_J$
B	$3{,}0_B$	$2{,}0_G$	$2{,}6_D$	$1{,}65_H$	$2{,}2_D$	∞	0_G	$1{,}3_G$	$0{,}45_G$	$1{,}85_I$	$2{,}85_J$
E	$3{,}0_B$	$2{,}0_G$	$2{,}6_D$	$1{,}65_H$	$2{,}2_D$	$2{,}55_E$	0_G	$1{,}3_G$	$0{,}45_G$	$1{,}85_I$	$2{,}85_J$
F	$3{,}0_B$	$2{,}0_G$	$2{,}6_D$	$1{,}65_H$	$2{,}2_D$	$2{,}55_E$	0_G	$1{,}3_G$	$0{,}45_G$	$1{,}85_I$	$2{,}85_J$
C	$3{,}0_B$	$2{,}0_G$	$2{,}6_D$	$1{,}65_H$	$2{,}2_D$	$2{,}55_E$	0_G	$1{,}3_G$	$0{,}45_G$	$1{,}85_I$	$2{,}85_J$
K	$3{,}0_B$	$2{,}0_G$	$2{,}6_D$	$1{,}65_H$	$2{,}2_D$	$2{,}55_E$	0_G	$1{,}3_G$	$0{,}45_G$	$1{,}85_I$	$2{,}85_J$

Keterangan:

c = *current*

∞ = tak hingga (tidak terhubung)

d = distance

Tabel 4. 16 Hasil Perhitungan Manual *Dijkstra Node* G

Node	**Rumus (km)**	***Node***	**Rumus (km)**
A	= *c+d* = B + 1,0 = 2 + 1,0 = 3 km	G	= *c+d* = 0 + 0 = 0 km
B	= *c + d* = G + 2,0 = 0 + 2,0 = 2,0 km	H	= *c+d* = G + 1,3 = 0 + 1,3 = 1,3 km
C	= *c+d* = D + 0,95 = 1,65 + 0,95 = 2,6 km	I	= *c+d* = G + 0,45 = 0 + 0,45 = 0,45 km
D	= *c+d* = H + 0,35 = 1,3 + 0,35 = 1,65 km	J	= *c+d* = I + 1,4 = 0,45 + 1,4 = 1,85 km
E	= *c+d* = D + 0,55 = 1,65 + 0,55 = 2,2 km	K	= *c+d* = J + 1,0 = 1,85 + 1,0 = 2,85 km
F	= *c+d* = E + 0,35 = 2,2 + 0,35 = 2,55 km		

Berdasarkan perhitungan *node* G, dapat disimpulakan *node* G mempunyai panjang lintasan masing-masing, yakni:

a. G → A = G-B-A dengan panjang lintasan 3,0km
b. G → B = dengan panjang lintasan 2,0km
c. G → C = G-H-D-C, dengan panjang lintasan 2,6km
d. G → D = G-H-D, dengan panjang lintasan 1,65km
e. G → E = G-H-D-E, dengan panjang lintasan 2,2km
f. G → F = G-H-D-E-F, dengan panjang lintasan 2,55km

g. G → H = dengan panjang lintasan 1,3km

h. G → I = dengan panjang lintasan 0,45km

i. G → J = G-I-J, dengan panjang lintasan 1,85km

j. G → K = G-I-J-K, dengan panjang lintasan 2,85km

8. *Node* H

Tabel 4. 17 Hasil Perhitungan Manual *Dijkstra Node* H

c	A	B	C	D	E	F	G	H	I	J	K
	∞	∞	∞	∞	∞	∞	∞	0_H	∞	∞	∞
H	∞	∞	∞	$0{,}35_H$	∞	∞	$1{,}3_H$	0_H	∞	∞	∞
D	∞	∞	$1{,}3_D$	$0{,}35_H$	$0{,}9_D$	∞	$1{,}3_H$	0_H	∞	∞	∞
E	∞	∞	$1{,}3_D$	$0{,}35_H$	$0{,}9_D$	$1{,}25_E$	$1{,}3_H$	0_H	∞	∞	∞
F	∞	∞	$1{,}3_D$	$0{,}35_H$	$0{,}9_D$	$1{,}25_E$	$1{,}3_H$	0_H	∞	∞	∞
C	∞	$2{,}4_C$	$1{,}3_D$	$0{,}35_H$	$0{,}9_D$	$1{,}25_E$	$1{,}3_H$	0_H	∞	∞	∞
G	∞	$2{,}4_C$	$1{,}3_D$	$0{,}35_H$	$0{,}9_D$	$1{,}25_E$	$1{,}3_H$	0_H	$1{,}75_G$	∞	∞
I	∞	$2{,}4_C$	$1{,}3_D$	$0{,}35_H$	$0{,}9_D$	$1{,}25_E$	$1{,}3_H$	0_H	$1{,}75_G$	$3{,}15_I$	∞
B	$3{,}4_B$	$2{,}4_C$	$1{,}3_D$	$0{,}35_H$	$0{,}9_D$	$1{,}25_E$	$1{,}3_H$	0_H	$1{,}75_G$	$3{,}15_I$	∞
J	$3{,}4_B$	$2{,}4_C$	$1{,}3_D$	$0{,}35_H$	$0{,}9_D$	$1{,}25_E$	$1{,}3_H$	0_H	$1{,}75_G$	$3{,}15_I$	$4{,}15_J$
A	$3{,}4_B$	$2{,}4_C$	$1{,}3_D$	$0{,}35_H$	$0{,}9_D$	$1{,}25_E$	$1{,}3_H$	0_H	$1{,}75_G$	$3{,}15_I$	$4{,}15_J$

Keterangan:

c = *current*

∞ = tak hingga (tidak terhubung)

d = distance

Tabel 4. 18 Hasil Perhitungan Manual *Dijkstra Node* H

Node	**Rumus (km)**	***Node***	**Rumus (km)**
A	= *c*+*d* = B + 1,0 = 2,4 + 1,0 = 3,4 km	G	= *c*+*d* = H + 1,3 = 0 + 1,3 = 1,3 km

B	= *c* + *d* = C + 1,1 = 1,3 + 1,1 = 2,4 km	H	= *c*+*d* = 0 + 0 = 0 km
C	= *c*+*d* = D + 0,95 = 0,35 + 0,95 = 1,3 km	I	= *c*+*d* = G + 0,45 = 1,3 + 0,45 = 1,75 km
D	= *c*+*d* = H + 0,35 = 0 + 0,35 = 0,35 km	J	= *c*+*d* = I + 1,4 = 1,75 + 1,4 = 3,15 km
E	= *c*+*d* = D + 0,55 = 0,35 + 0,55 = 0,9 km	K	= *c*+*d* = J + 1,0 = 3,15 + 1,0 = 4,15 km
F	= *c*+*d* = E + 0,35 = 0,9 + 0,35 = 1,25 km		

Berdasarkan perhitungan *node* H, dapat disimpulakan *node* H mempunyai panjang lintasan masing-masing, yakni:

a. H → A = H-D-C-B-A, dengan panjang lintasan 3,4km
b. H → B = H-D-C-B, dengan panjang lintasan 2,4km
c. H → C = H-D-C, dengan panjang lintasan 1,3km
d. H → D = dengan panjang lintasan0,35km
e. H → E = H-D-E, dengan panjang lintasan 0,9km
f. H → F = H-D-E-F, dengan panjang lintasan 1,25km
g. H → G = dengan panjang lintasan 1,3km
h. H → I = H-G-I, dengan panjang lintasan 1,75km
i. H → J = H-G-I-J, dengan panjang lintasan 3,15km
j. H → K = H-G-I-J-K, dengan panjang lintasan 4,15km

9. *Node* I

Tabel 4. 19 Hasil Perhitungan Manual *Dijkstra Node* I

c	A	B	C	D	E	F	G	H	I	J	K
	∞	∞	∞	∞	∞	∞	∞	∞	0_I	∞	∞
I	∞	∞	∞	∞	∞	∞	$0{,}45_I$	∞	0_I	$1{,}4_I$	∞
G	∞	$2{,}45_G$	∞	∞	∞	∞	$0{,}45_I$	$1{,}75_G$	0_I	$1{,}4_I$	∞
J	∞	$2{,}45_G$	∞	∞	∞	∞	$0{,}45_I$	$1{,}75_G$	0_I	$1{,}4_I$	$2{,}4_J$
H	∞	$2{,}45_G$	∞	$2{,}1_H$	∞	∞	$0{,}45_I$	$1{,}75_G$	0_I	$1{,}4_I$	$2{,}4_J$
D	∞	$2{,}45_G$	$3{,}05_D$	$2{,}1_H$	$2{,}65_D$	∞	$0{,}45_I$	$1{,}75_G$	0_I	$1{,}4_I$	$2{,}4_J$
K	∞	$2{,}45_G$	$3{,}05_D$	$2{,}1_H$	$2{,}65_D$	$2{,}75_K$	$0{,}45_I$	$1{,}75_G$	0_I	$1{,}4_I$	$2{,}4_J$
B	$3{,}45_B$	$2{,}45_G$	$3{,}05_D$	$2{,}1_H$	$2{,}65_D$	$2{,}75_K$	$0{,}45_I$	$1{,}75_G$	0_I	$1{,}4_I$	$2{,}4_J$
E	$3{,}45_B$	$2{,}45_G$	$3{,}05_D$	$2{,}1_H$	$2{,}65_D$	$2{,}75_K$	$0{,}45_I$	$1{,}75_G$	0_I	$1{,}4_I$	$2{,}4_J$
F	$3{,}45_B$	$2{,}45_G$	$3{,}05_D$	$2{,}1_H$	$2{,}65_D$	$2{,}75_K$	$0{,}45_I$	$1{,}75_G$	0_I	$1{,}4_I$	$2{,}4_J$
C	$3{,}45_B$	$2{,}45_G$	$3{,}05_D$	$2{,}1_H$	$2{,}65_D$	$2{,}75_K$	$0{,}45_I$	$1{,}75_G$	0_I	$1{,}4_I$	$2{,}4_J$

Keterangan:

c = *current*

∞ = tak hingga (tidak terhubung)

d = distance

Tabel 4. 20 Hasil Perhitungan Manual *Dijkstra Node* I

Node	**Rumus (km)**	***Node***	**Rumus (km)**
A	= *c*+*d* = B + 1,0 = 2,45 + 1,0 = 3,45 km	G	= *c*+*d* = I + 0,45 = 0 + 0,45 = 0,45 km
B	= *c* + *d* = G + 2,0 = 0,45 + 2,0 = 2,45 km	H	= *c*+*d* = G + 1,3 = 0,45 + 1,3 = 1,75 km
C	= *c*+*d* = D + 0,95 = 2,1 + 0,95 = 3,05 km	I	= *c*+*d* = 0 + 0 = 0 km

D	= *c+d* = H + 0,35 = 1,75 + 0,35 = 2,1 km	J	= *c+d* = I + 1,4 = 0 + 1,4 = 1,4 km
E	= *c+d* = D + 0,55 = 2,1 + 0,55 = 2,65 km	K	= *c+d* = J + 1,0 = 1,4 + 1,0 = 2,4 km
F	= *c+d* = K + 0,35 = 2,4 + 0,35 = 2,75 km		

Berdasarkan perhitungan *node* I, dapat disimpulakan *node* I mempunyai panjang lintasan masing-masing, yakni:

a. I → A = I-G-B-A, dengan panjang lintasan 3,45km
b. I → B = I-G-B, dengan panjang lintasan 2,45km
c. I → C = I-G-H-D-C, dengan panjang lintasan 3,05km
d. I → D = I-G-H-D, dengan panjang lintasan 2,1km
e. I → E = I-G-H-D-E, dengan panjang lintasan 2,65km
f. I → F = I-J-K-F, dengan panjang lintasan 2,75km
g. I → G = dengan panjang lintasan 0,45km
h. I → H= I-G-H, dengan panjang lintasan 1,75km
i. I → J = dengan panjang lintasan 1,4km
j. I → K = I-J-K, dengan panjang lintasan 2,4km

10. *Node* J

Tabel 4. 21 Hasil Perhitungan Manual *Dijkstra Node* J

c	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**
	∞	∞	∞	∞	∞	∞	∞	∞	∞	0_J	∞
J	∞	∞	∞	∞	∞	∞	∞	∞	$1{,}4_J$	0_J	$1{,}0_J$
K	∞	∞	∞	∞	∞	$1{,}35_K$	∞	∞	$1{,}4_J$	0_J	$1{,}0_J$
F	∞	∞	∞	∞	$1{,}7_F$	$1{,}35_K$	∞	∞	$1{,}4_J$	0_J	$1{,}0_J$

I	∞	∞	∞	∞	$1{,}7_F$	$1{,}35_K$	$1{,}85_I$	∞	$1{,}4_J$	0_J	$1{,}0_J$
E	∞	∞	∞	$2{,}25_E$	$1{,}7_F$	$1{,}35_K$	$1{,}85_I$	∞	$1{,}4_J$	0_J	$1{,}0_J$
G	∞	$3{,}85_G$	∞	$2{,}25_E$	$1{,}7_F$	$1{,}35_K$	$1{,}85_I$	$3{,}15_G$	$1{,}4_J$	0_J	$1{,}0_J$
D	∞	$3{,}85_G$	$3{,}2_D$	$2{,}25_E$	$1{,}7_F$	$1{,}35_K$	$1{,}85_I$	$2{,}6_D$	$1{,}4_J$	0_J	$1{,}0_J$
H	∞	$3{,}85_G$	$3{,}2_D$	$2{,}25_E$	$1{,}7_F$	$1{,}35_K$	$1{,}85_I$	$2{,}6_D$	$1{,}4_J$	0_J	$1{,}0_J$
C	∞	$3{,}85_G$	$3{,}2_D$	$2{,}25_E$	$1{,}7_F$	$1{,}35_K$	$1{,}85_I$	$2{,}6_D$	$1{,}4_J$	0_J	$1{,}0_J$
B	$4{,}85_B$	$3{,}85_G$	$3{,}2_D$	$2{,}25_E$	$1{,}7_F$	$1{,}35_K$	$1{,}85_I$	$2{,}6_D$	$1{,}4_J$	0_J	$1{,}0_J$

Keterangan:

c = *current*

∞ = tak hingga (tidak terhubung)

d = distance

Tabel 4. 22 Hasil Perhitungan Manual *Dijkstra Node* J

Node	**Rumus (km)**	***Node***	**Rumus (km)**
A	= *c*+*d* = B + 1,0 = 3,85 + 1,0 = 4,85 km	G	= *c*+*d* = I + 0,45 = 1,4 + 0,45 = 1,85 km
B	= *c* + *d* = G + 2,0 = 1,85 + 2,0 = 3,85 km	H	= *c*+*d* = D + 0,35 = 2,25 + 0,35 = 2,6 km
C	= *c*+*d* = D + 0,95 = 2,25 + 0,95 = 3,2 km	I	= *c*+*d* = J + 1,4 = 0 + 1,4 = 1,4 km
D	= *c*+*d* = E + 0,55 = 1,7 + 0,55 = 2,25 km	J	= *c*+*d* = 0 + 0 = 0 km
E	= *c*+*d* = F + 0,35 = 1,35 + 0,35 = 1,7 km	K	= *c*+*d* = J + 1,0 = 0 + 1,0 = 1,0 km
F	= *c*+*d* = K + 0,35 = 1,0 + 0,35 = 1,35 km		

Berdasarkan perhitungan *node* J, dapat disimpulakan *node* J mempunyai panjang lintasan masing-masing, yakni:

a. J → A = J-I-G-B-A, dengan panjang lintasan 4,85km
b. J → B = J-I-G-B, dengan panjang lintasan 3,85km
c. J → C = J-K-F-E-D-C, dengan panjang lintasan 3,2km
d. J → D = J-K-F-E-D, dengan panjang lintasan 2,25km
e. J → E = J-K-F-E, dengan panjang lintasan 1,7km
f. J → F = J-K-F, dengan panjang lintasan 1,35km
g. J → G = J-I-G, dengan panjang lintasan 1,85km
h. J → H = J-K-F-E-D-H, dengan panjang lintasan 2,6km
i. J → I = dengan panjang lintasan 1,4km
j. J → K = dengan panjang lintasan 1km

11. *Node* K

Tabel 4. 23 Hasil Perhitungan Manual *Dijkstra Node* K

c	A	B	C	D	E	F	G	H	I	J	K
	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0_K
K	∞	∞	∞	∞	∞	$0{,}35_K$	∞	∞	∞	∞	0_K
F	∞	∞	∞	∞	$0{,}7_F$	$0{,}35_K$	∞	∞	∞	∞	0_K
E	∞	∞	∞	$1{,}25_E$	$0{,}7_F$	$0{,}35_K$	∞	∞	∞	∞	0_K
D	∞	∞	$2{,}2_D$	$1{,}25_E$	$0{,}7_F$	$0{,}35_K$	∞	$1{,}6_D$	∞	∞	0_K
H	∞	∞	$2{,}2_D$	$1{,}25_E$	$0{,}7_F$	$0{,}35_K$	$2{,}9_H$	$1{,}6_D$	∞	∞	0_K
C	∞	$3{,}3_C$	$2{,}2_D$	$1{,}25_E$	$0{,}7_F$	$0{,}35_K$	$2{,}9_H$	$1{,}6_D$	∞	∞	0_K
G	∞	$3{,}3_C$	$2{,}2_D$	$1{,}25_E$	$0{,}7_F$	$0{,}35_K$	$2{,}9_H$	$1{,}6_D$	$3{,}35_G$	∞	0_K
B	$4{,}3_B$	$3{,}3_C$	$2{,}2_D$	$1{,}25_E$	$0{,}7_F$	$0{,}35_K$	$2{,}9_H$	$1{,}6_D$	$3{,}35_G$	∞	0_K
I	$4{,}3_B$	$3{,}3_C$	$2{,}2_D$	$1{,}25_E$	$0{,}7_F$	$0{,}35_K$	$2{,}9_H$	$1{,}6_D$	$3{,}35_G$	$4{,}75_I$	0_K
A	$4{,}3_B$	$3{,}3_C$	$2{,}2_D$	$1{,}25_E$	$0{,}7_F$	$0{,}35_K$	$2{,}9_H$	$1{,}6_D$	$3{,}35_G$	$4{,}75_I$	0_K

Keterangan:

c = *current*

∞ = tak hingga (tidak terhubung)

d = distance

Tabel 4. 24 Hasil Perhitungan Manual *Dijkstra Node* K

Node	Rumus (km)	Node	Rumus (km)
A	= *c+d* = B + 1,0 = 3,3 + 1,0 = 4,3 km	G	= *c+d* = H + 1,3 = 1,6 + 1,3 = 2,9 km
B	= *c + d* = C + 1,1 = 2,2 + 1,1 = 3,3 km	H	= *c+d* = D + 0,35 = 1,25 + 0,35 = 1,6 km
C	= *c+d* = D + 0,95 = 1,25 + 0,95 = 2,2 km	I	= *c+d* = G + 0,45 = 2,9 + 0,45 = 3,35 km
D	= *c+d* = E + 0,55 = 0,7 + 0,55 = 1,25 km	J	= *c+d* = I + 1,4 = 3,35 + 1,4 = 4,75 km
E	= *c+d* = F + 0,35 = 0,35 + 0,35 = 0,7 km	K	= *c+d* = 0 + 0 = 0 km
F	= *c+d* = K + 0,35 = 0 + 0,35 = 0,35 km		

Berdasarkan perhitungan *node* K, dapat disimpulakan *node* K mempunyai panjang lintasan masing-masing, yakni:

a. K → A = K-F-E-D-C-B-A, dengan panjang lintasan 4,3km
b. K → B = K-F-E-D-C-B, dengan panjang lintasan 3,3km
c. K → C = K-F-E-D-C, dengan panjang lintasan 2,2km
d. K → D = K-F-E-D, dengan panjang lintasan 1,25km
e. K → E = K-F-E, dengan panjang lintasan 0,7km
f. K → F = dengan panjang lintasan 0,35km

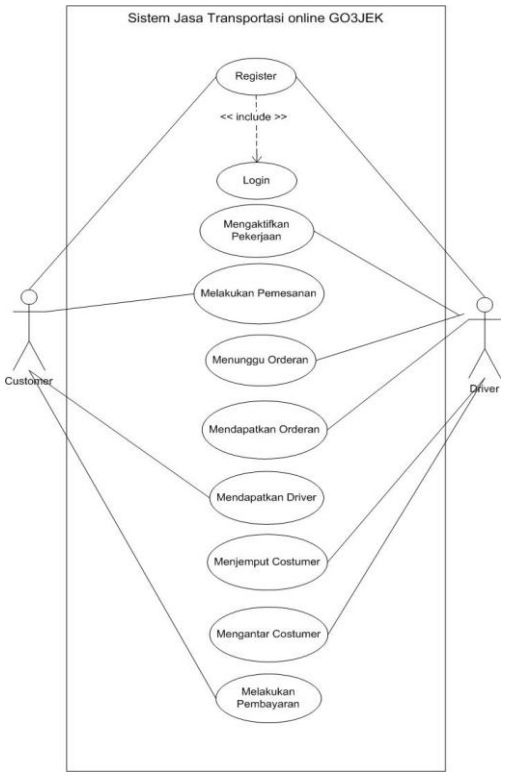
g. K → G = K-F-E-D-H-G, dengan panjang lintasan 2,9km
h. K → H = K-F-E-D-H, dengan panjang lintasan 1,6km
i. K → I = K-F-E-D-H-G-I, dengan panjang lintasan 3,35km
j. K → J = K-F-E-D-H-G-I-J, dengan panjang lintasan 4,75km

4.3 Desain Model Proses

Dalam subbab ini akan dibahas lebih detail mengenai alur sistem usulan yang telah dibahas sebelumnya, pada subbab ini akan membahas hal seperti analisa sistem melalui diagram UML dan analisa *database* yang akan dibuat.

4.3.1 *Use Case Diagram*

Usecase diagram bertujuan untuk menjelaskan beberapa aktifitas-aktifitas yang dilakukan oleh aktor yang merupakan *user* dalam sebuah sistem yang akan dikembangkan ataupun dibangun



Gambar 4. 7 *Usecase Diagram* Sistem Jasa Transportasi *Online* GO3JEK

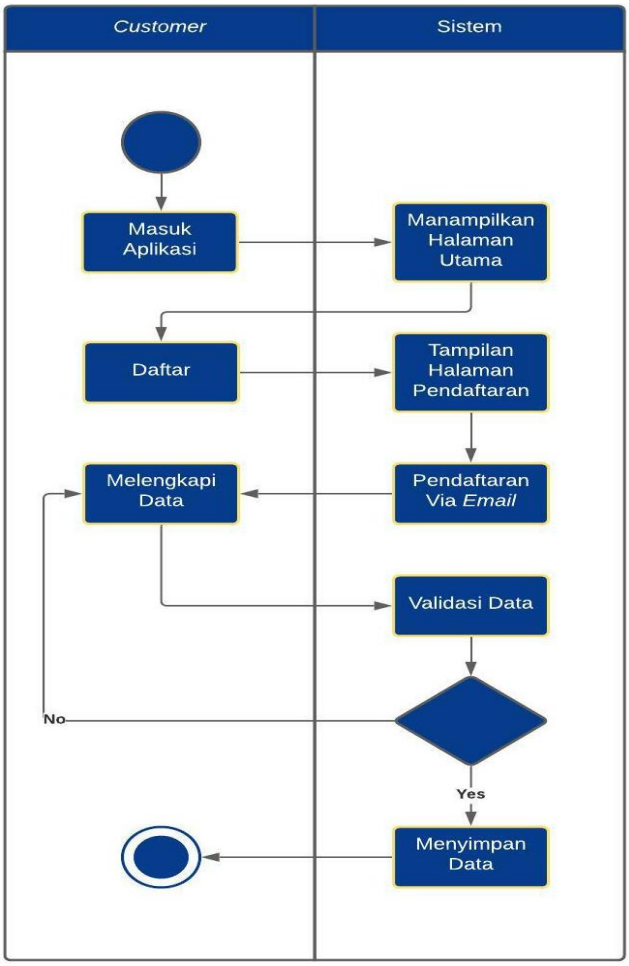
Pada diagram *usecase* yang telah dirancang diatas, dapat dilihat bahwa pada sistem jasa transportasi *online* GO3JEK mempunyai dua aktor yakni *customer* dan juga *driver*, terlihat kedua aktor mempunyai aktifitas masing-masing. *Customer* dan *driver* dapat melakukan *register* ataupun pembuatan akun, dan juga *login*. Aktor *customer* mempunyai beberapa aktifitas lainnya seperti melakukan pemesanan, mendapatkan *driver*, dan juga melakukan pembayaran terhadap *driver* nantinya. Sedangkan aktor *driver* mempunyai beberapa aktifitas lainnya seperti mengaktifkan pekerjaan, menunggu orderan masuk, mendapatkan orderan, menjemput *customer*, dan juga mengantarkan *customer*.

4.3.2 *Activity Diagram*

Secara sederhana *activity diagram* merupakan sebiah alur atupun *flow* mengenai aktifitas-aktifitas apa saja yang dapat dilakukan oleh seorang aktor ataupun *user*. Pada subbab ini penulis memaparkan *activity* menjadi dua, masing-masing untuk *activity* dari sisi *customer* dan *activity* untuk sisi *driver*. Berikut merupakan beberapa *activity* yang menjadi pembahasan pada aplikasi transportasi *online* GO3JEK.

1. *Activity Diagram* Registrasi (*Customer*)

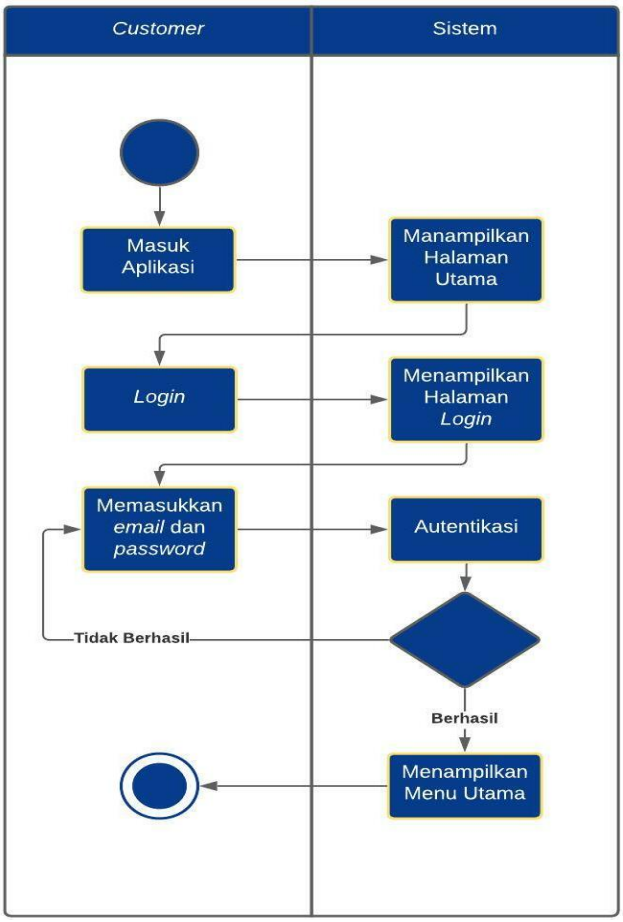
   Dalam melakukan registrasi, *customer* dapat melakukan registrasi dengan menggunakan *email* sebagai alternatifnya. *Email* yang terdaftar nantinya, merupakan sebuah data penting yang utama dalam sistem untuk melakukan *auththenticatien* ataupun autentikasi. Adapun *activity diagram* yang dirancang penulis, yakni:



Gambar 4. 8 *Activity Diagram* Registrasi *Customer*

Pada *diagram activity* yang dirancang penulis, *customer* memulai aktifitas dengan melakukan pendaftaran akun ataupun *registrasi* menggunakan *email*. Kemudian *customer* akan melakukan pengisian data untuk kelengkapan data informasi dan data diri. Kemudian sistem akan melakukan validasi apakah email yang ditulisan sudah benar ataupun apakah data-data yang dibutuhkan sudah dilengkapi atau tidak. Apabila data yang dimasukkan salah ataupun tidak lengkap, maka sistem akan mengarahkan *customer* untuk melengkapi ataupun memperbaiki data yang akan dimasukkan. Namun apabila data yang dimasukkan sudah benar, maka sistem akan menyimpan data *customer* tersebut kedalam *database*.

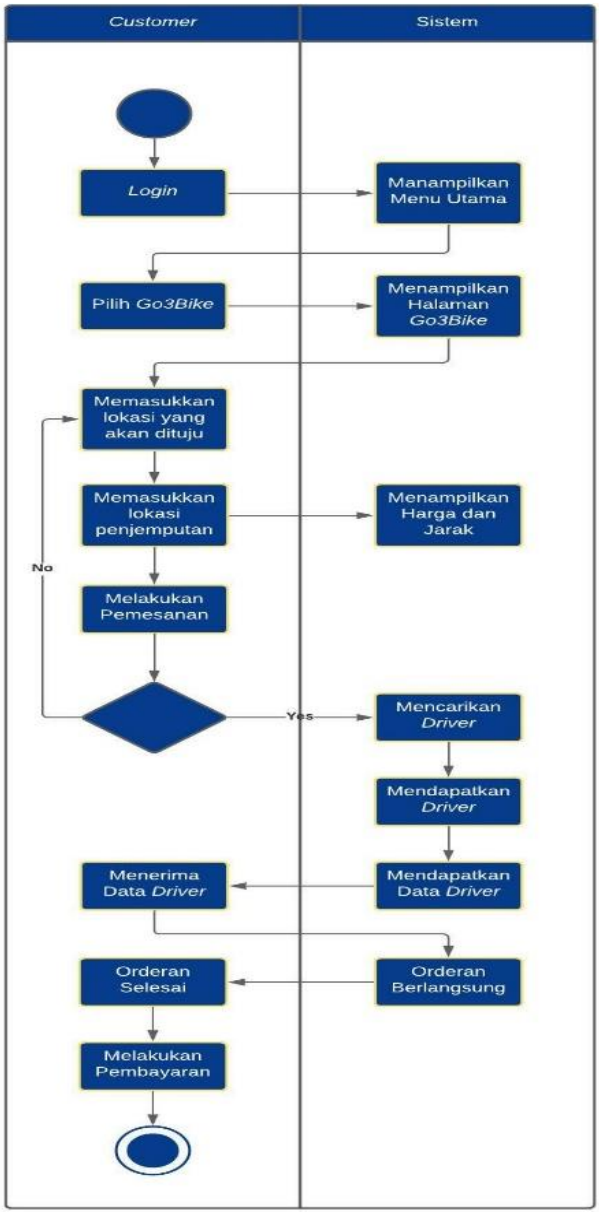
2. *Activity Diagram Login* (*Customer*)



Gambar 4. 9 *Activity Diagram Login Customer*

Pada gambar diatas, *customer* melakukan proses *login*, dimana *customer* memulai aktifitasnya dengan membuka aplikasi terlebih dahulu, kemudian sistem akan menampilkan interface halaman utama. Setelah itu *customer* memilih *login*. Setelah berada pada halaman *login*, kemudian *customer* memasukkan *email* dan *password* yang sudah didaftarkan ataupun terdaftar didalam *database*. Setelah memasukkan *email* dan *password*, sistem akan melakukan autentikasi ataupun *authentication*, apabila proses autentikasi berhasil, maka sistem akan menampilkan halaman menu utama. Namun apabila proses autentikasi tidak berhasil maka *customer* tersebut harus melakukan proses *login* dengan benar.

3. *Activity Diagram* Orderan (*Customer*)

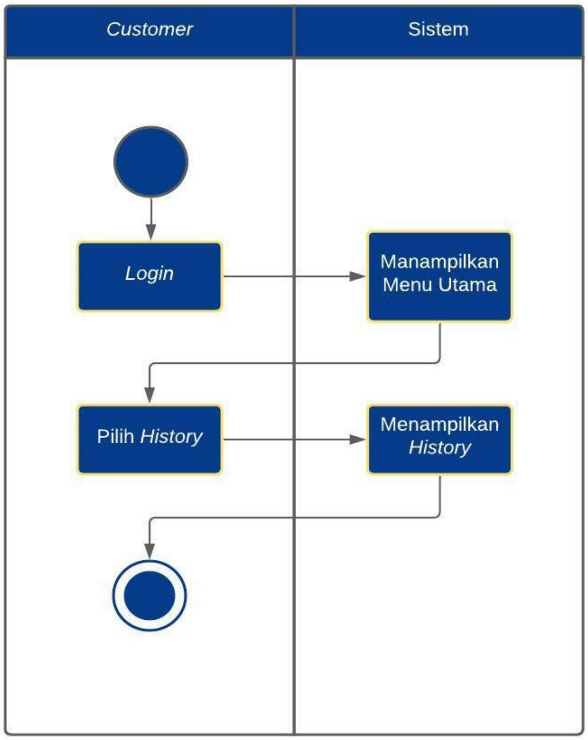


Gambar 4. 10 *Activity Diagram* Orderan *Customer*

Dalam melakukan orderan, seorang aktor *customer* pastinya mempunyai aktifitas-aktifitas yang akan dilalui agar dapat melakukan pesanan terhadap jasa transportasi *online* yang akan dirancang. Pada *activity* ini, *user* dapat melakukan pemesanan jasa *transportasi online*, dimana *customer* setelah melakukan *login* dan berada pada halaman utama, *customer* dapat memilih menu *Go3Bike* kemudian memasukkan alamat yang akan dituju, dan juga

memasukkan alamat penjemputan. Setelah mengisi data alamat yang dituju dan juga alamat penjemputan, maka *customer* dapat langsung melakukan proses transaksi ataupun melakukan pemesanan. Setelah melakukan pemesanan, sistem secara otomatis akan mencarikan *driver* yang sedang *online*. Orderan akan segera berlangsung ketika sistem sudah mendapatkan *driver*. Ketika orderan telah selasai, maka *customer* akan melakukan aktifitas pembayaran. Adapun aktifitas-aktifitas pada *customer* telah dirancang oleh penulis, yakni:

4. *Activity Diagram History* (*Customer*)



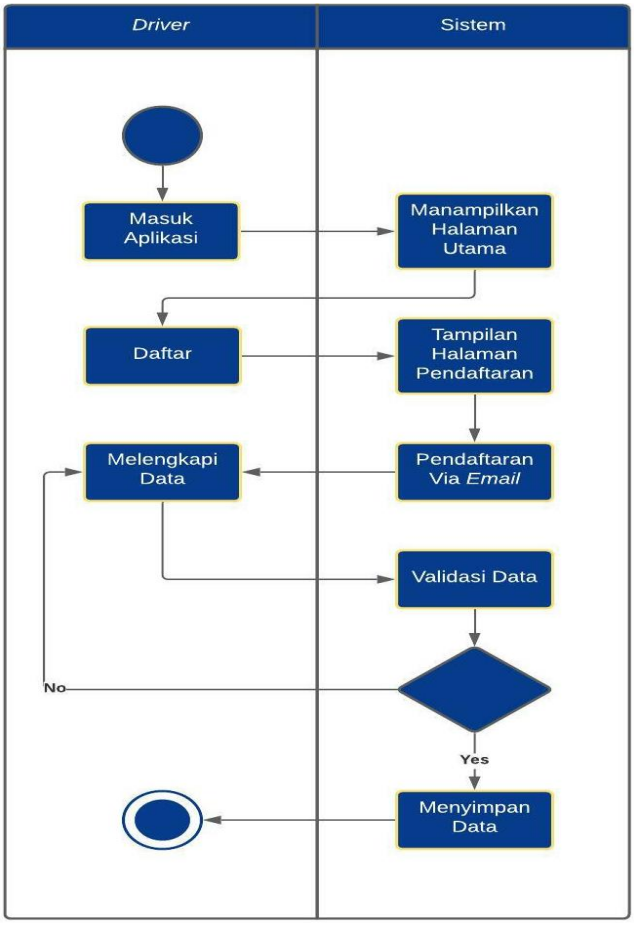
Gambar 4. 11 *Activity Diagram History Customer*

Pada gambar diatas merupakan sebuah aktifitas ataupun *flow* dari *activity diagram* laporan transaksi ataupun *history*, dimana *user* selaku *customer* dapat melihat laporan transaksi ataupun orderan yang sudah pernah dijalani oleh *customer* sendiri. *User* akan masuk kedalam aplikasi, setelah melakukan proses *login*, *user* dapat memilih menu *history* pada tampilan

menu utama. Setelah memilih menu *history*, sistem akan menampilkan *history* ataupun transaksi yang sudah pernah dilakukan oleh *user* selaku *customer* tersebut.

5. *Activity diagram* Registrasi (*Driver*)

Dalam melakukan registrasi, *driver* dapat melakukan registrasi dengan menggunakan *email* sebagai alternatifnya. *Email* yang terdaftar nantinya, merupakan sebuah data penting yang utama dalam sistem untuk melakukan *auththenticatien* ataupun autentikasi. Adapun *activity diagram* yang dirancang penulis, yakni:

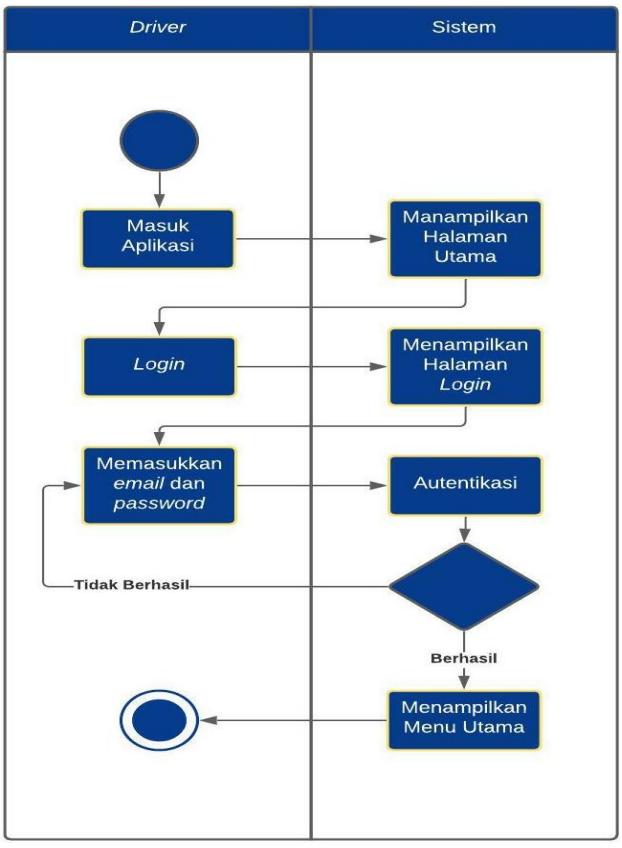


Gambar 4. 12 *Activity Diagram* Registrasi *Driver*

Pada *diagram activity* yang dirancang penulis, *driver* memulai aktifitas dengan melakukan pendaftaran akun ataupun *registrasi* menggunakan

email. Kemudian *driver* akan melakukan pengisian data untuk kelengkapan data informasi dan data diri. Kemudian sistem akan melakukan validasi apakah *email* yang ditulisan sudah benar ataupun apakah data-data yang dibutuhkan sudah dilengkapi atau tidak. Apabila data yang dimasukkan salah ataupun tidak lengkap, maka sistem akan mengarahkan *driver* untuk melengkapi ataupun memperbaiki data yang akan dimasukkan. Namun apabila data yang dimasukkan sudah benar, maka sistem akan menyimpan data *driver* tersebut kedalam *database*.

6. *Activity Diagram Login* (*Driver*)

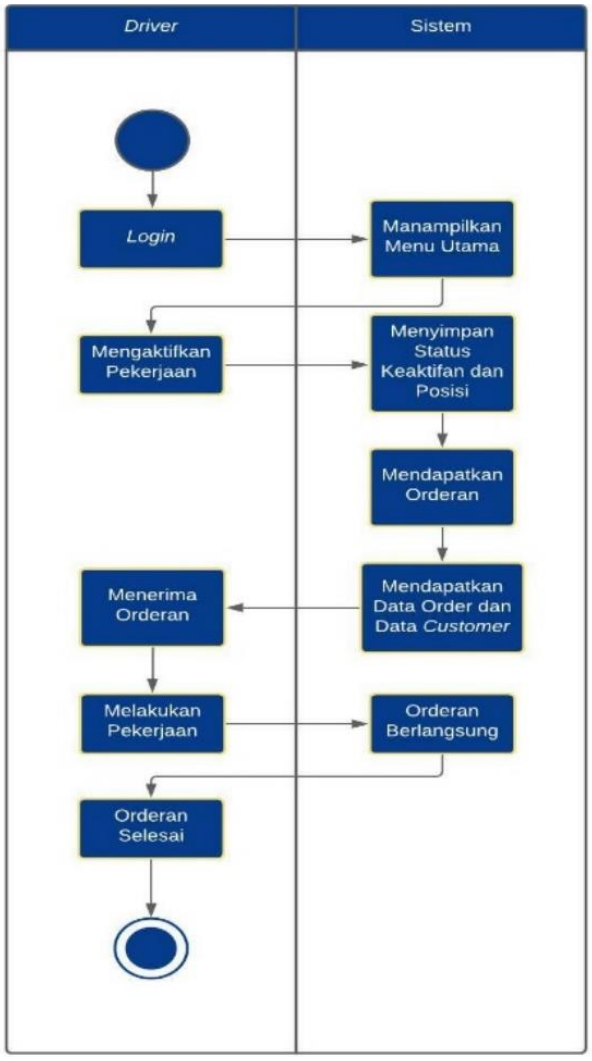


Gambar 4. 13 *Activity Diagram Login Driver*

Pada *activity diagram* login, *driver* melakukan proses *login*, dimana *driver* memulai aktifitasnya dengan membuka aplikasi terlebih dahulu, kemudian sistem akan menampilkan *interface* halaman utama dan *driver* dapat

memilih menu *login*. Setelah itu *driver* memasukkan *email* dan *password* yang sudah didaftarkan ataupun terdaftar didalam *database*. Setelah memasukkan *email* dan *password*, sistem akan melakukan autentikasi ataupun *authentication*, apabila proses autentikasi berhasil, maka sistem akan menampilkan halaman menu utama. Namun apabila proses autentikasi tidak berhasil maka *driver* tersebut harus melakukan proses *login* dengan benar. Adapun *flow* ataupun aktifitas aktor dari *driver* tersebut telah dipaparkan oleh penulis kedalam diagram, yakni:

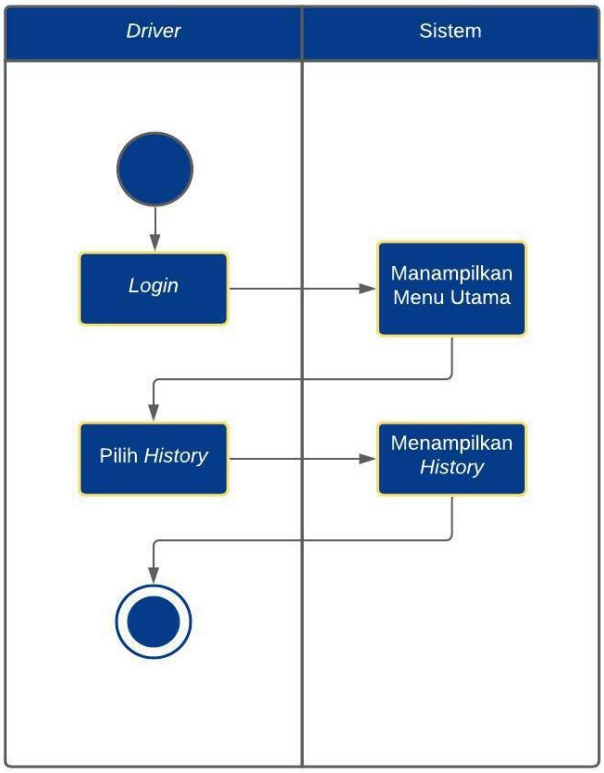
7. *Activity Diagram* Pekerjaan (*Driver*)



Gambar 4. 14 *Activity Diagram* Pekerjaan *Driver*

Pada *activity diagram* pekerjaan, *user* selaku *driver* mempunyai beberapa *flow* ataupun aktivitas dalam melakukan proses pekerjaan, dimana setelah *user* melakukan proses *login* pada aplikasi, selanjutnya *user* akan diarahkan sistem kepada menu utama, dimana pada halaman menu utama tersebut *user* harus mengaktifkan pekerjaan. Setelah mengaktifkan pekerjaan, sistem akan malacak dan menyimpan posisi *user*, kemudian sistem akan meneruskan data orderan yang masuk kepada *driver*. Kemudian *driver* akan melakukan pekerjaan. Untuk lebih jelas, penulis telah membuat diagram *activity* pekerjaan, yakni:

8. *Activity Diagram History* (*Driver*)



Gambar 4. 15 *Activity Diagram History Driver*

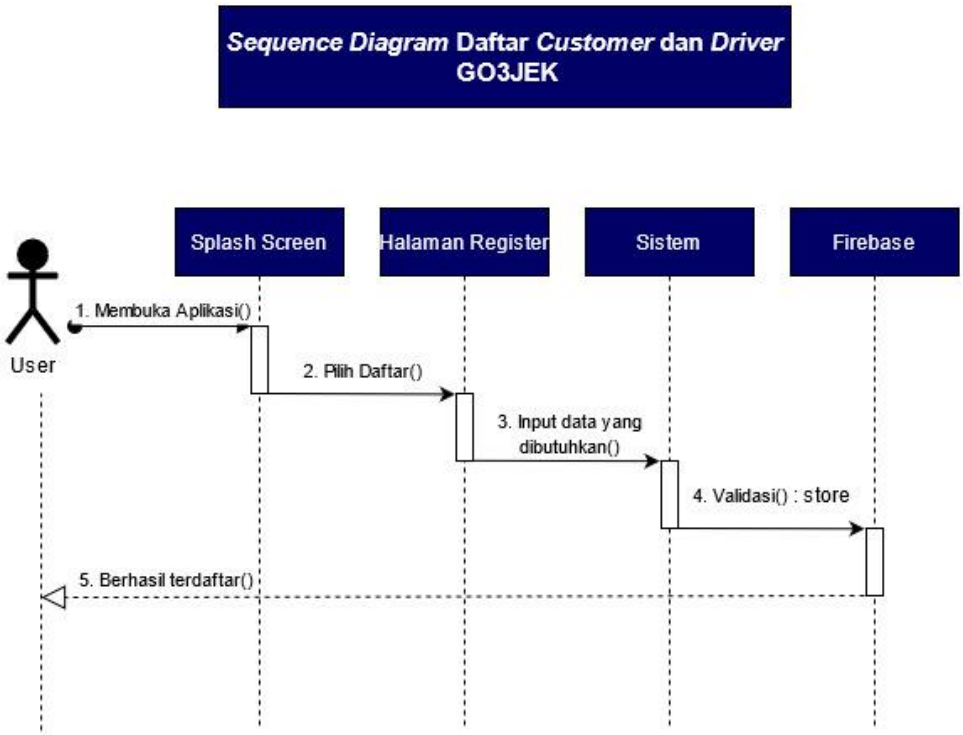
Pada gambar diatas merupakan sebuah aktifitas ataupun *flow* dari *activity diagram* laporan transaksi ataupun *history*, dimana *user* selaku *driver* dapat melihat laporan transaksi ataupun orderan yang sudah pernah

dijalani oleh *driver* sendiri. *User* akan masuk kedalam aplikasi, setelah melakukan proses *login*, *user* dapat memilih menu *history* pada tampilan menu utama. Setelah memilih menu *history*, sistem akan menampilkan *history* ataupun transaksi yang sudah pernah dilakukan oleh *user* selaku *driver* tersebut.

4.3.3 *Sequence Diagram*

Berikut ini merupakan beberapa *sequential diagram* yang dibuat oleh penulis yang akan digunakan dalam pengembangan aplikasi, yakni:

1. *Sequence Diagram* Register (*Customer* dan *Driver*)

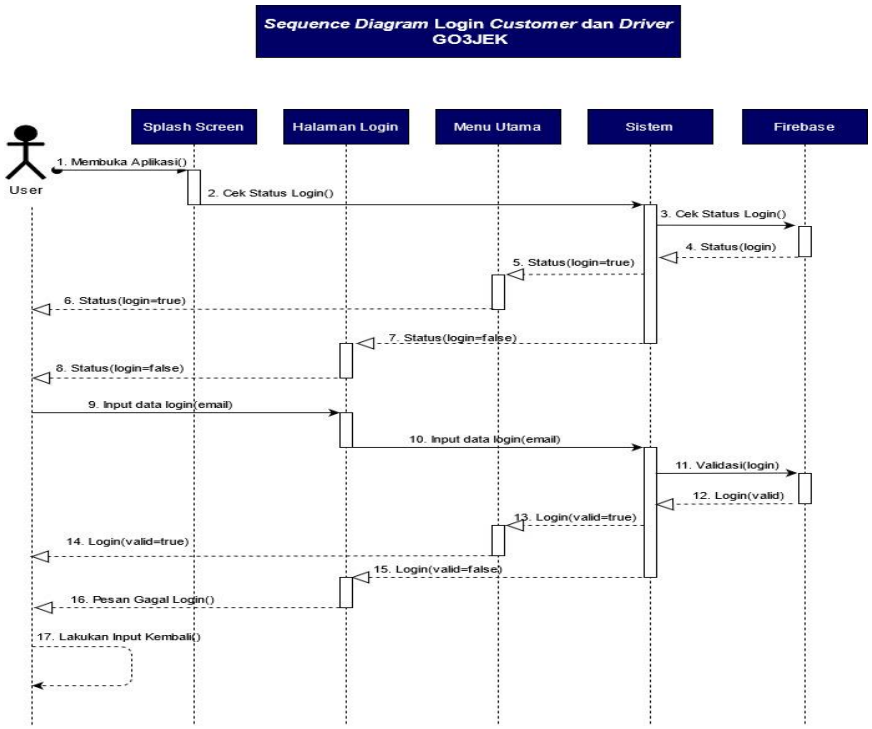


Gambar 4. 16 *Sequence Diagram Register*

Gambar diatas menjelaskan tentang rangkaian pesan antar objek saat *user* melakukan aktifitas Register ataupun daftar akun agar dapat menggunakan aplikasi tersebut, dimana *user* tersebut merupakan aktor *Customer* dan *Driver*. Dimulai ketika *user* memasukkan data-data diri kedalam halaman register, kemudian sistem akan melakukan validasi. Setelah itu sistem

akan meminta *request* terhadap *firebase* dan memberi sebuah *response* bahwa data yang diisi oleh *user* telah berhasil didaftarkan kedalam *database.*

2. *Sequence Diagram Login* (*Customer* dan *Driver*)

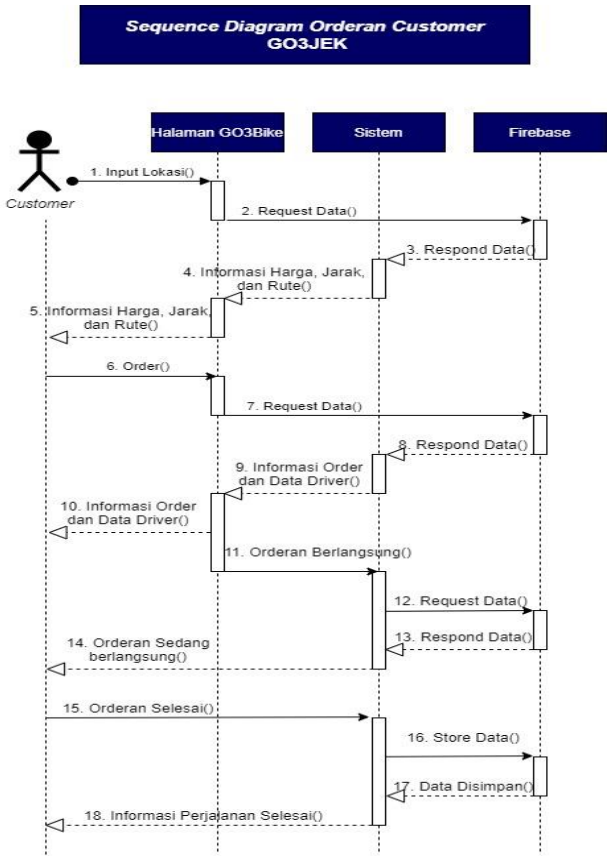


Gambar 4. 17 *Sequence Diagram Login*

Sequencel diagram diatas menjelaskan tentang rangkaian pesan antar objek saat *user* melakukan aktifasi *login*, dimana *user* tersebut merupakan sebuah aktor *customer* dan juga *driver*. Dimulai dimana *user* ataupun pengguna membuka aplikasi ini maka akan disambut dengan *splash screen* yang merupakan tampilan selamat datang. Selanjutnya sistem akan secara otomatis melakukan proses *session*, dimana proses ini bertujuan sebagai pengecekan terhadap *user* apakah pernah masuk ke aplikasi ini atau tidak. Dalam melakukan proses *session*, sistem melakukan *request* data kepada server dimana server tersebut merupakan *firebase*, dan server akan memberikan sebuah *response* apakah pengguna sudah pernah *login* atau

belum. Apabila *response* yang diberikan server bernilai benar, maka *user* akan dapat langsung menggunakan aplikasi, namun apabila *response* menghasilkan nilai negasi maka *user* akan diarahkan kehalaman *login*. Setelah berada pada halaman login, maka *user* akan memasukkan data *login* melalui *email* yang sudah terdaftar. *Email* dan *password* yang dimasukkan kemudian akan diproses oleh sistem, lalu sistem akan melakukan *validasi* kepada *firebase* untuk mengetahui apakah data tersebut ada atau tidak. Jika data tersebut ada, maka *user* akan mempunyai hak akses untuk menggunakan aplikasi, tetapi apabila data tersebut tidak ada maka *user* akan mengulang kembali proses *inputan*.

3. *Sequence Diagram* Orderan (*Customer*)



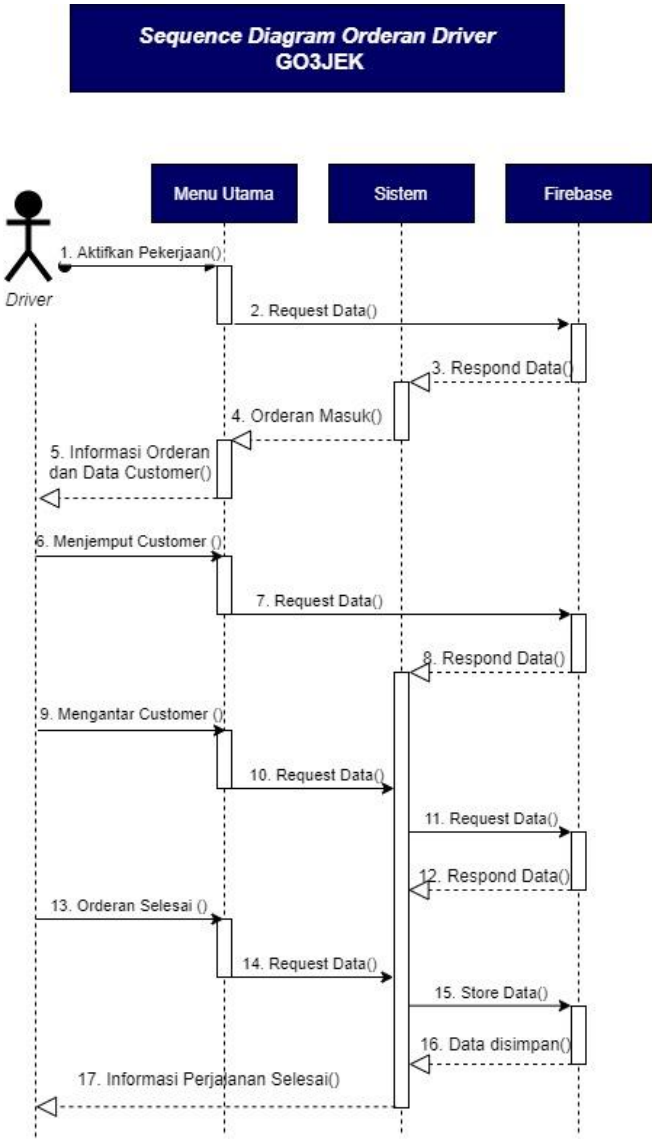
Gambar 4. 18 *Sequence Diagram* Orderan *Customer*

Pada gambar ini menjelaskan tentang skenario ataupun proses pemesanan *transportasi online* pada *user* selaku *customer*, diawali dengan *user* melakukan *input* lokasi penjemputan dan juga pengantaran pada halaman *GO3Bike*. Kemudian sistem akan melakukan *request* data terhadap *firebase*. Setelah itu *firebase* akan merespon data tersebut dan akan merekam data orderan yang masuk, lalu sistem akan secara otomatis mengeluarkan nilai mengenai jarak, harga, dan rute perjalanan mengenai perjalanan yang sudah di *input* oleh *user* tersebut, kemudian data ini akan diteruskan kedalam *interface* halaman *Go3bike* pada sistem. Proses selanjutnya yaitu *user* melakukan orderan, kemudian sistem akan melakukan *request* ataupun permintaan data kepada *server* selaku *firebase* untuk mendapatkan *driver*. Kemudian *firebase* akan memberikan sebuah *response*, yang berisikan data *driver* yang sudah didapatkan dan akan diteruskan kepada *user* selaku *customer* tersebut. Proses selanjutnya adalah melakukan perjalanan, setelah perjalanan selesai, *server* akan memasukkan data dan menjadikannya sebagai *database*.

4. *Sequence Diagram* Orderan (*Driver*)

Seorang *user* selaku *driver* memulai skenario ataupun proses dengan memngaktifkan pekerjaan pada *interface* ataupun menu utama. Setelah *user* mengaktifkan pekerjaan, maka sistem akan melakukan pengiriman status keaktifan *user* kepada *server* dan melakukan permintaan data kepada *server* selaku *firebase* mengenai letak posisi *user*. Proses selanjutnya, *firebase* akan memberikan sebuah *response* dan meneruskannya kepada sistem mengenai orderan beserta data-data *customer* yang sudah masuk ataupun terekam dari aktivitas *customer* dan akan menginformasikan kepada *user* selaku *driver* tersebut. Kemudian *driver* akan melakukan aktifitas penjemputan kepada *customer* dan memberikan *request* data kepada *firebase*, lalu *firebase* akan melakukan *response* data dan merekamnya kepada sistem. Tahap selanjutnya *driver* akan melakukan aktivitas pengantaran terhadap *customer*. Sama seperti

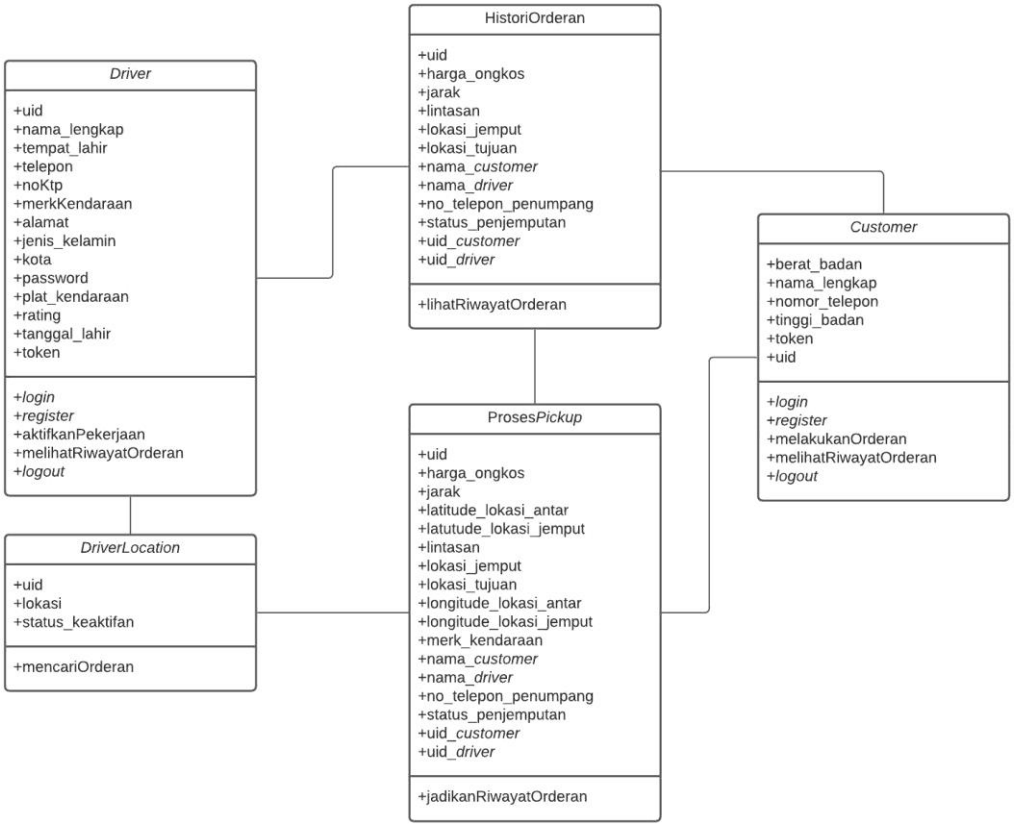
aktifitas menjemput *customer*, aktifitas mengantar *customer* pun, sistem akan meminta *request* data, lalu *firebase* akan memberikan sebuah *response* kepada sistem mengenai aktifitas yang sedang dilakukan. Setelah *user* telah selesai dan melakukan konfirmasi orderan telah selesai, maka data tersebut akan direkam dan disimpan oleh *server* sebagai *database*.



Gambar 4. 19 *Sequence Diagram* Orderan *Driver*

4.3.4 *Class Diagram*

Class diagram adalah sebuah deskripsi yang akan menunjukkan beberapa atribut yang saling berhubungan dalam sistem yang sedang dibangun. Adapun *class diagram* untuk sistem transportasi *online* yang akan dibangun penulis, yakni:



Gambar 4. 20 *Class Diagram* Aplikasi Jasa Transportasi *Online* GO3JEK

4.4 Desain *Database*

Setelah melakukan proses *modelling*, tahap selanjutnya adalah melakukan perancangan *database* berdasarkan data dan informasi yang telah didapatkan. Pada tahapan ini. Pada *database* jenis NoSql memiliki format JSON (*Javascript Object Notation*).

JSON merupakan sebuah format pertukaran data yang sangat ringan dan mudah dibaca serta diolah pengguna, sehingga mudah untuk diterjemahkan dan

dibuat oleh komputer. Dalam NoSql dikenal istilah *collection*, istilah ini dalam *database sql* dapat disamakan dengan tabel dan baris. Untuk memudahkan pembacaan dalam penelitian ini penulis menggambarkan gambaran *database* yang akan dirancang dalam bentuk tabel lalu mengimplemantasikan rancangan tersebut dalam bentuk JSON.

4.4.1 *Collection Customers*

Tabel 4. 25 *Collection Customers*

Key	***Type Data***	**Deskripsi**
uid_customer	Object	Unique ID

Tabel 4. 26 *Child* dari key uid_ustomers

Child	***Type Data***	**Deskripsi**
berat_badan	String	berat badan *user*
nama_lengkap	String	nama lengkap *user*
nomor_telepon	String	nomor telepon *user*
tinggi_badan	String	tinggi badan *user*
token	String	*firebase cloud messanging* token
uid	String	*UniqueID user*

Representasi JSON pada tabel *Customer*

```
{
        "Customers" : {
                "uid_customer" : {
                        "berat_badan" : berat badan pada user",
                        "nama_lengkap" : "nama lengkap pada user",
                        "nomor_telepon" : "085261637693",
                        "tinggi_badan" : "tinggi badan user"
                        "uid" : "mHepK7iH2bfTnjmr2ITebjVo3zF3"
                        "token" : "mHcsxxxxxxxxxxxxxxx"
                        }
        }
}
```

4.4.2 *Collection Driver*

Tabel 4. 27 *Collection Driver*

Key	***Type Data***	**Deskripsi**
uid_driver	Object	Unique ID

Tabel 4. 28 *Child* dari key uid_driver

Child	***Type Data***	**Deskripsi**
nama_lengkap	String	nama lengkap *driver*
alamat	String	alamat *driver*
email	String	*email driver*
jenis_kelamin	String	jenis kelamin *driver*
kota	String	kota asal *driver* sesuai ktp
merk_kendaraan	String	merk kendaraan *driver*
no_ktp	String	no ktp *driver*
password	String	password *driver*
plat_kendaraan	String	String
tanggal_lahir	String	tanggal lahir sesuai KTP
tempat_lahir	String	tempat lahir sesuai KTP
rating	String	rating *driver*
telepon	String	nomor telepon *driver*
token	String	*firebase cloud messanging* token
uid	String	*uniqueID driver*

Representasi JSON pada tabel *Driver*

```
{
        "Driver" : {
                "uid_driver" : {
                        "nama_lengkap" : "Nama lengkap driver",
                        "alamat": “alamat driver”,
                        "email" : "email driver"
                        "jenis_kelamin" : "laki-laki"
                        "kota" : "medan"
                        "merk_kendaraan" : "yamaha"
                        "no_ktp" : "06878xxx"
                        "password" : "0xxxx"
                        "plat_kendaraan" : "BKxxxxx"
                        "tanggal_lahir" : "xxxx"
                        "tempat_lahir" : "medan"
                        "rating" : "rating sementara"
                        "telepon" : "081276678912"
                       "token" : ""cZ_h0symTqa3s5mpoBbrUL:APA91bE8-UN-w.."
                        "uid" : "BmKrGeukIMW8eLxIXqsDsaoXCPm1"
                        }
        }
}
```

4.4.3 *Collection DriverLocation*

Tabel 4. 29 *Collection DriverLocation*

Key	***Type Data***	**Deskripsi**
Location	Object	Lokasi driver

Tabel 4. 30 *Child* dari key Location

Child	***Type Data***	**Deskripsi**
uid	Object	Uid driver

Tabel 4. 31 *Child* dari key uid location driver

Child	***Type Data***	**Deskripsi**
lat	String	Garis lintang (*vertical*) posisi *driver*
lng	String	Garis bujur (*horizontal*) posisi *driver*

Representasi JSON pada tabel *DriverLocation*

```
{
        "DriverLocation" : {
                "Medan" : {
                        "uid" : "unique id driver" : {
                                "lat" : "garis lintang driver",
                                "lng" : "garis bujur driver"
                                }
                        }
        }
}
```

4.4.4 *Collection HistoriAktifitas*

Tabel 4. 32 *Collection* HistoriAktifitas

Key	***Type Data***	**Deskripsi**
id_transaksi	Object	key

Tabel 4. 33 *Child* dari key id_transaksi

Child	***Type Data***	**Deskripsi**
harga_ongkos	String	harga ongos tertera
jarak	String	jarak *driver*
lintasan	String	lintasan hasil algoritma *dijkstra*
lokasi_jemput	String	lokasi jemput user
lokasi_tujuan	String	lokasi pengantaran
nama_customer	String	nama *user*
nama_driver	String	nama *driver*
no_telepon_penumpang	String	no telepon *user*
status_penjemputan	String	status orderan
uid_customer	String	unique id *user*
uid_driver	String	uinique id *driver*

Representasi JSON pada tabel HistoriAktifitas

```
{
        "HistoriAktifitas" : {
                "id_transaksi" : {
                        "harga_ongkos" : berat badan pada user",
                        "jarak" : "nama lengkap pada user",
                        "lintasan" : "085261637693",
                        "lokasi_jemput" : "tinggi badan user"
                        "lokasi_tujuan" : "mHepK7iH2bfTnjmr2ITebjVo3zF3",
                        "nama_customer" : "nama penumpang",
                        "nama_driver" : "nama driver",
                        "no_telepon_penumpang" : "08527…."
                         "status_penjemputan" : "status orderan"
                         "uid_customer" : "Qxtv71e8X0PGHfkoEuuWWDnITc12"
                         "uid_driver" : "BmKrGeukIMW8eLxIXqsDsaoXCPm1"
                        }
        }
}
```

4.4.5 *Collection* Proses*Pickup*

Tabel 4. 34 *Collection* Proses*Pickup*

Key	***Type Data***	**Deskripsi**
id_prosesOrderan	Object	Unique ID

Tabel 4. 35 *Child* dari *key* id_prosesOrderan

Child	***Type Data***	**Deskripsi**
harga_ongkos	String	harga ongos tertera
jarak	String	jarak *driver*
lintasan	String	lintasan hasil algoritma *dijkstra*
latitude_lokasi_antar	String	garis lintang lokasi pengantaran
latitude_lokasi_jemput	String	garis lintang lokasi penjemputan
lokasi_jemput	String	lokasi jemput user
lokasi_tujuan	String	lokasi pengantaran
longitude_lokasi_antar	String	garis bujur lokasi pengantaran

longitude_lokasi_jemput	String	garis bujur lokasi penjemputan
nama_customer	String	nama *user*
nama_driver	String	nama *driver*
no_telepon_penumpang	String	no telepon *user*
status_penjemputan	String	status orderan
uid_customer	String	unique id *user*
uid_driver	String	uinique id *driver*

Representasi dalam bentuk JSON

```
{
        "ProsesPickup" : {
                "id_prosesOrderan" : {
                        "harga_ongkos" : berat badan pada user",
                        "jarak" : "nama lengkap pada user",
                        "latitude_lokasi_jemput" : "garis lintang penjemputan",
                         "latitude_lokasi_antar" : “garis bujur pengantaran",
                        "lintasan" : "085261637693",
                        "lokasi_jemput" : "tinggi badan user"
                        "lokasi_tujuan" : "mHepK7iH2bfTnjmr2ITebjVo3zF3",
                        "longitude_lokasi_jemput" : “garis bujur penjemputan",
                        "longitude_lokasi_antar " : "garis bujur pengantaran",
                        "nama_customer" : "nama penumpang",
                        "nama_driver" : "nama driver",
                        "no_telepon_penumpang" : "08527…."
                         "status_penjemputan" : "status orderan"
                         "uid_customer" : "Qxtv71e8X0PGHfkoEuuWWDnITc12"
                         "uid_driver" : "BmKrGeukIMW8eLxIXqsDsaoXCPm1"
                        }
        }
}
```

4.5 Rancangan Antarmuka Sistem

Pada subbab rancangan antarmuka sistem ini menjelaskan tentang bagaimana antarmuka yang akan diimplementasikan oleh penulis dalam sistem. Rancangan antarmuka ataupun *interface* yang akan dibangun menggunakan *mockflow*, secara sederhana *mockflow* adalah sebuah *software* ataupun *tool cloudbased* yang mempunyai fitur *template library* dan *drag andr drop* untuk memudahkan pengguna dalam menggunakannya. *Tool* ini sangat cocok digunakan untuk para pengguna yang baru belajar *wireframing*.

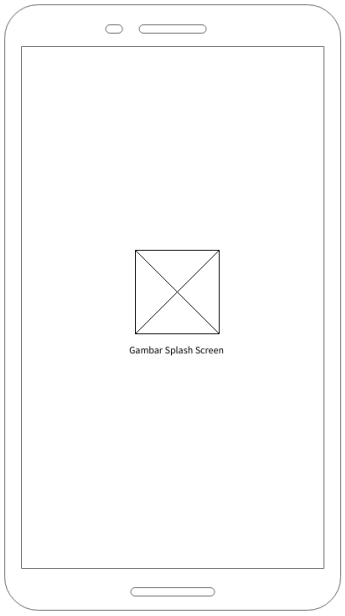
Dalam pengembangan aplikasi berbasis android terdapat beberapa komponen-komponen utama yang disebut *view* ataupun *widget*. Adapun beberapa komponen-komponen tersebut adalah sebagai berikut:

Tabel 4. 36 Komponen Widget pada Android

Nama Widget	Fungsi
Textview	Merupakan sebuah komponen yang menampilkan teks kepada pengguna, komponen ini cocok digunakan untuk menyampaikan informasi kepada pengguna dalam bentuk teks.
EditText	Merupakan sebuah komponen yang berfungsi sebagai wadah pengguna menginputkan suatu nilai/masukan yang dibutuhkan dalam sistem, contoh melakukan inputan data nama, tanggal lahir, dan sebagainya
ImageView	Merupakan sebuah komponen yang mempresentasikan suatu gambar kepada pengguna, misalkan gambar avatar pengguna.
Button	Merupakan sebuah komponen aksi, dimana user bisa memberikan aksi atau perintah kepada sistem sesuai kasusnya.
RecyclerView	Adalah sebuah komponen yang tedapat pada android yg mempresentasikan perulangan data dalam bentuk list yang bisa di kustomasi

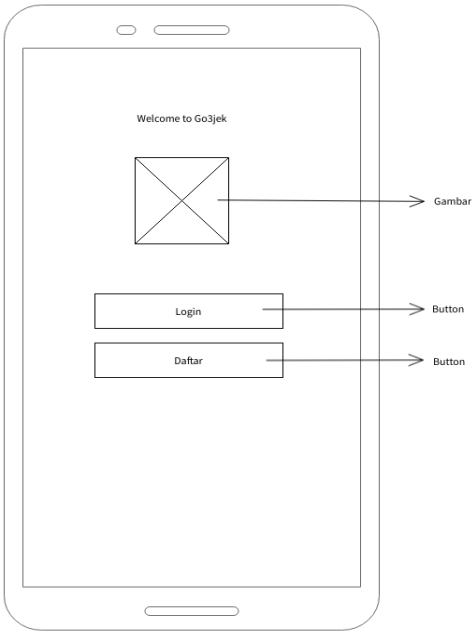
1. Tampilan *Mockflow Customer*

a) Rancangan Antarmuka *Splash Screen Customer*



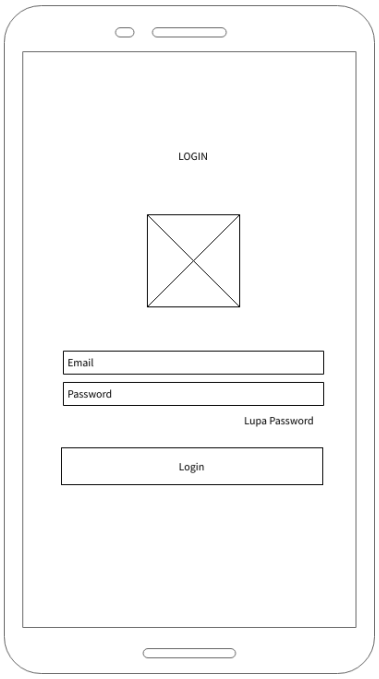
Gambar 4.21 Rancangan Antarmuka *Splash Screen Customer*

b) Rancangan Antarmuka *Welcome Screen Customer*



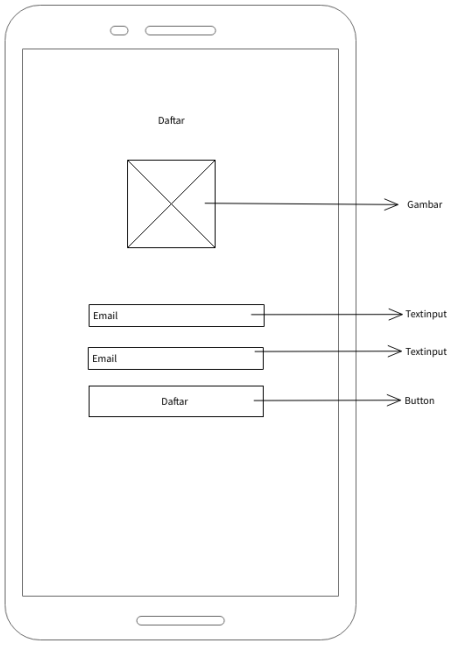
Gambar 4.22 Rancangan Antarmuka *Welcome Screen Customer*

c) Rancangan Antarmuka *Login Customer*



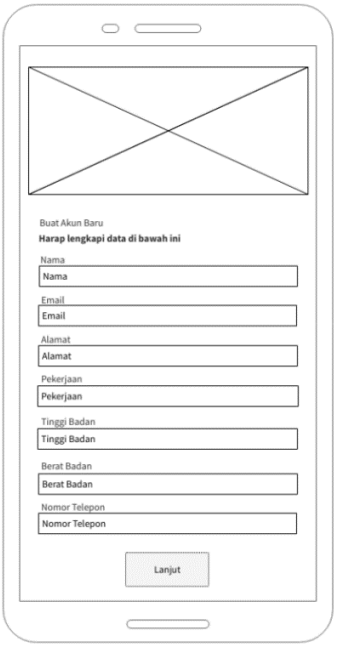
Gambar 4.23 Rancangan Antarmuka *Login Customer*

d) Rancangan Antarmuka Daftar *Customer*



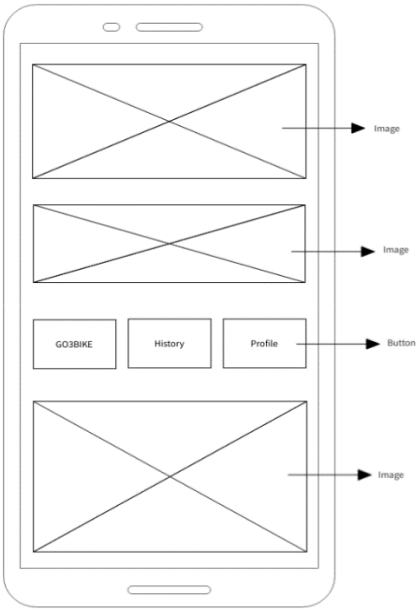
Gambar 4.24 Rancangan Antarmuka Daftar *Customer*

e) *Rancangan Antarmuka* Input *Data* Customer



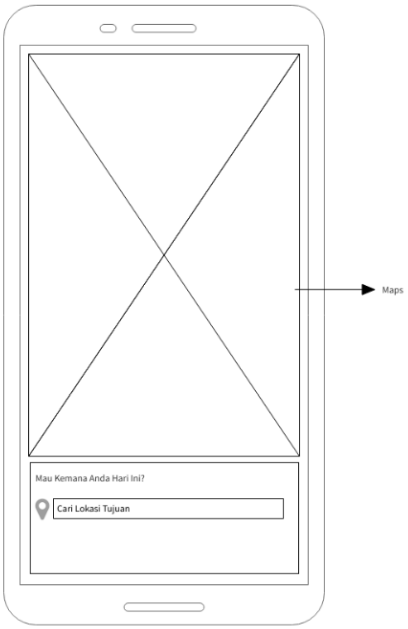
Gambar 4.25 Rancangan Antarmuka *Input* Data *Customer*

f) Rancangan Antarmuka Menu Utama



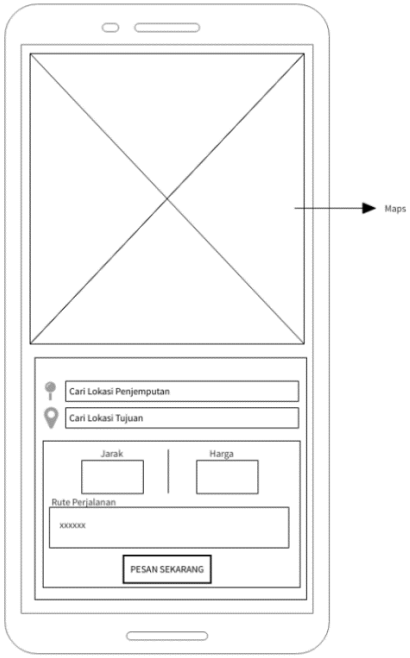
Gambar 4.26 Rancangan Antarmuka Menu Utama

g) Rancangan Antarmuka Halaman Go3Bike



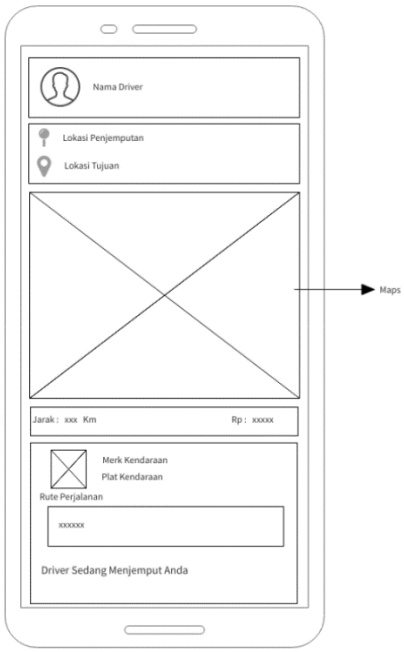
Gambar 4.27 Rancangan Antarmuka Halaman Go3Bike

h) Rancangan Antarmuka Halaman melakukan orderan



Gambar 4.28 Rancangan Antarmuka Melakukan Orderan

i) Rancangan Antarmuka Halaman Mendapatkan *Driver*



Gambar 4.29 Rancangan Antarmuka Halaman Mendapatkan *Driver*

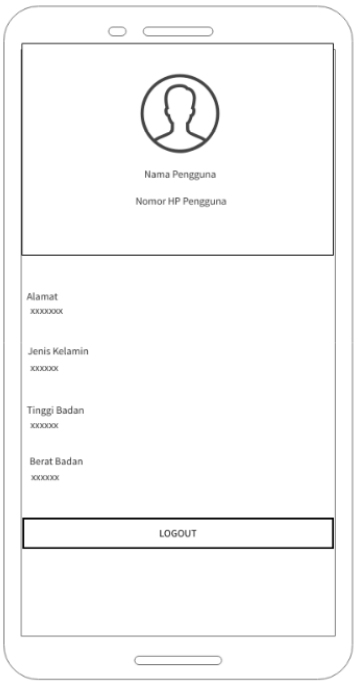
j) Rancangan Antarmuka Halaman Orderan Berlangsung

Gambar 4.30 Rancangan Antarmuka Halaman Orderan Berlangsung

k) Rancangan Antarmuka *History Customer*

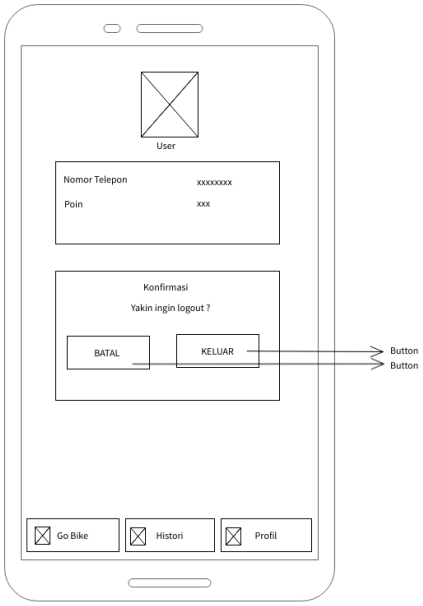
Gambar 4.31 Rancangan Antarmuka *HIstory Customer*

l) Rancangan Antarmuka Profil *Customer*



Gambar 4. 32 Rancangan Antarmuka Profil *Customer*

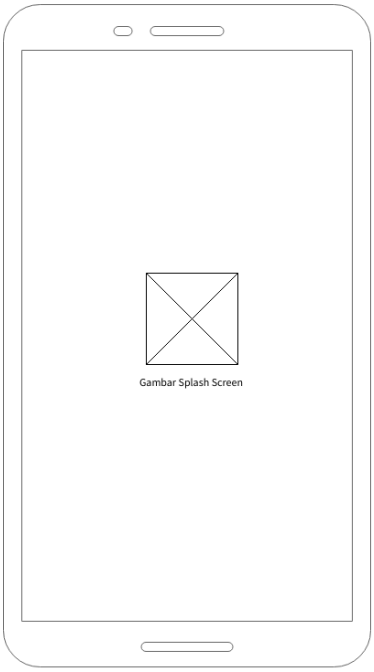
m) Rancangan Antarmuka Konfirmasi *Logout Customer*



Gambar 4.33 Rancangan Antarmuka Konfirmasi *Logout Customer*

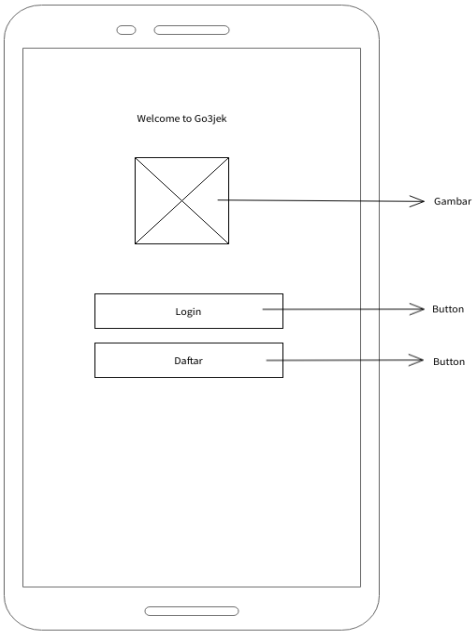
2. Tampilan *Mockflow Driver*

a) Rancangan Antarmuka *Splash Screen Driver*



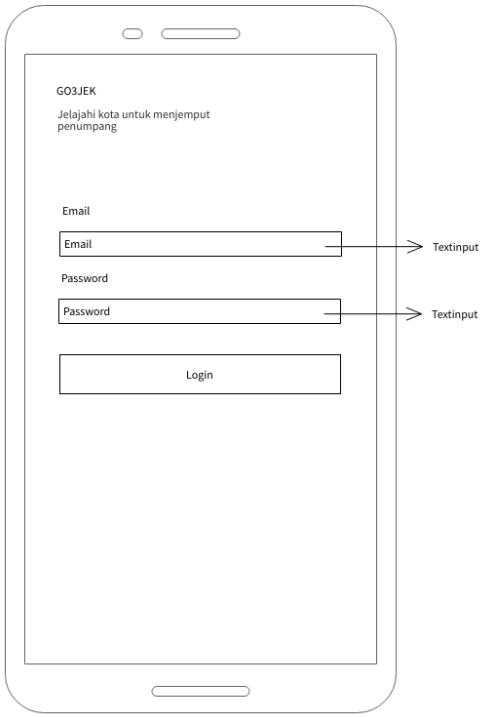
Gambar 4.34 Rancangan Antarmuka *Splash Screen Driver*

b) Rancangan Antarmuka *Wellcome Screen Driver*



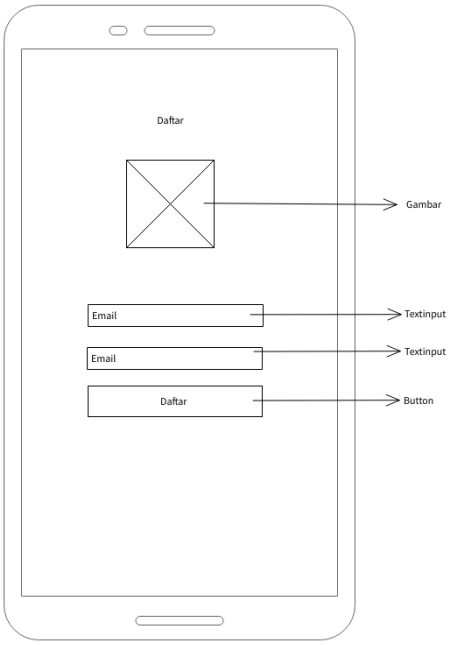
Gambar 4. 35 Rancangan Antarmuka *Wellcome Screen driver*

c) Rancangan Antarmuka *Login Driver*



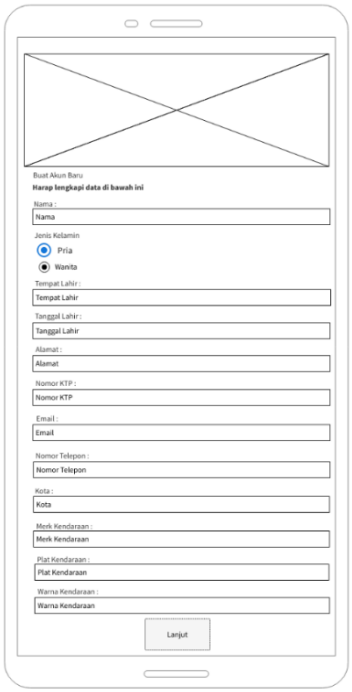
Gambar 4.36 Rancangan Antarmuka *Login Driver*

d) Rancangan Antarmuka Daftar *Driver*



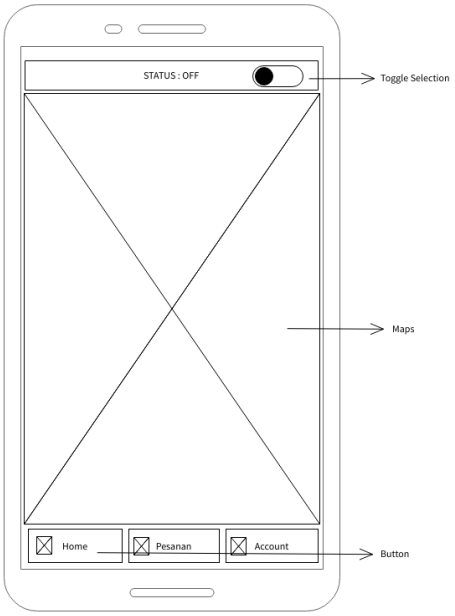
Gambar 4. 37 Rancangan Antarmuka Halaman Daftar *Driver*

e) Rancangan Antarmuka *Input* Data *Driver*



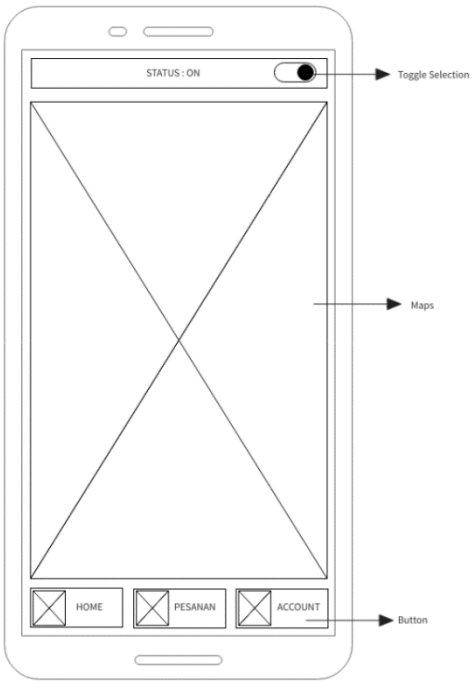
Gambar 4. 38 Rancangan Halaman Antarmuka Input Data *Driver*

f) Rancangan Antarmuka Menu Utama *Driver Offline*



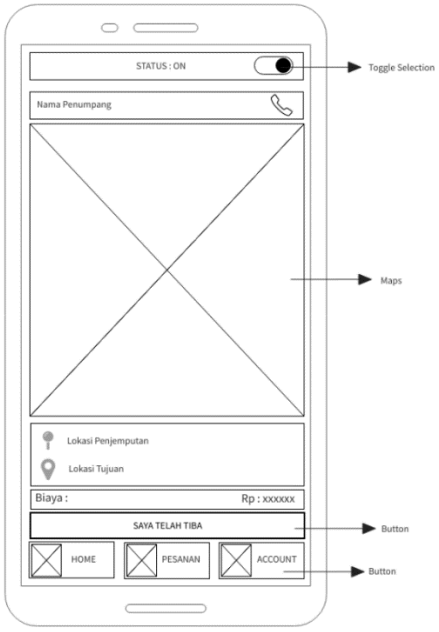
Gambar 4.39 Rancangan Antarmuka Halaman Utama *Driver Offline*

g) Rancangan Antarmuka Halaman Utama *Driver Online*



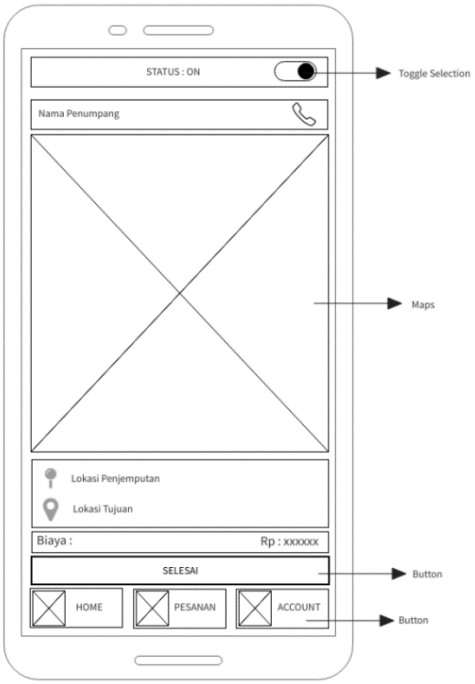
Gambar 4.40 Rancangan Antarmuka Halaman Utama *Driver Online*

h) Rancangan Antarmuka Transaksi Terima *Orderan Customer*



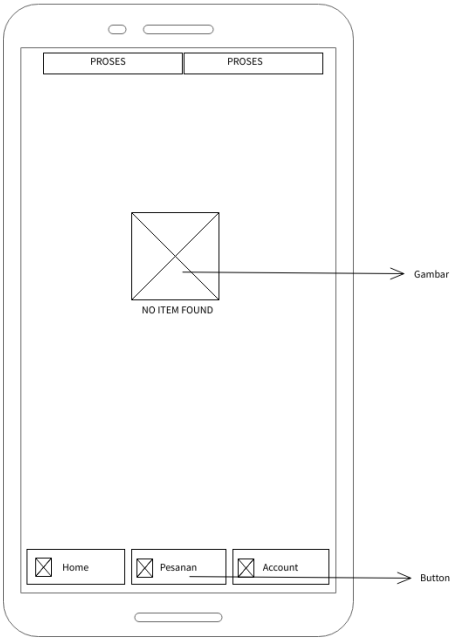
Gambar 4. 41 Rancangan Antarmuka Transaksi Terima *Orderan*

i) Rancangan Antarmuka Tampilan Orderan Berlangsung *Driver*



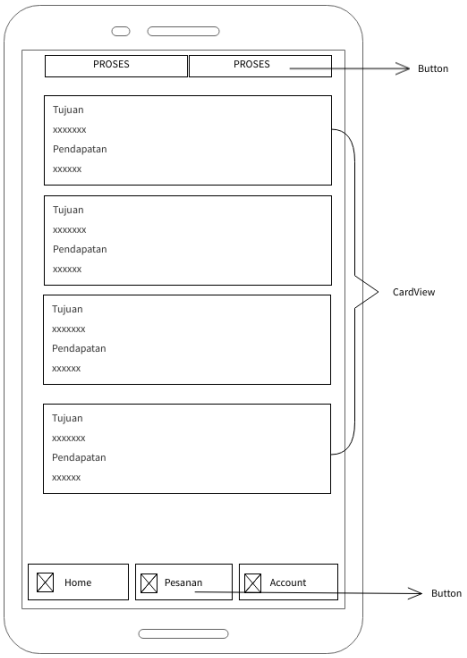
Gambar 4. 42 Rancangan Antarmuka Tampilan Orderan Berlangsung *Driver*

j) Rancangan Antarmuka Tidak Ada Transaksi Berjalan *Driver*



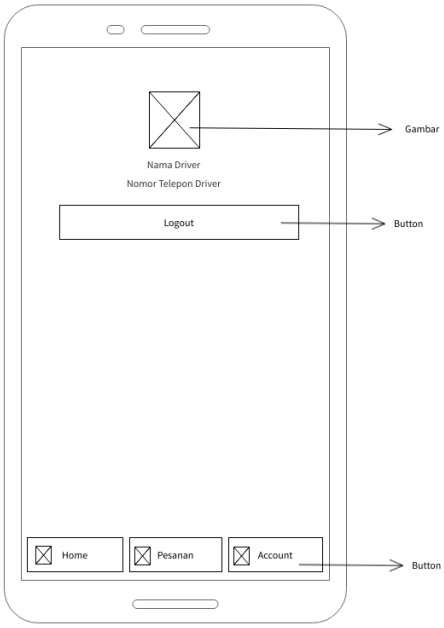
Gambar 4.43 Rancangan Antarmuka Tidak Ada Transaksi Berjalan *Driver*

k) Rancangan Antarmuka *History* Transaksi Selesai *Driver*



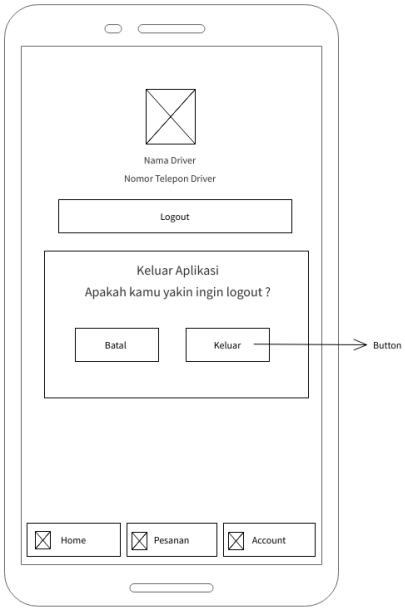
Gambar 4.44 Rancangan Antarmuka *History* Transaksi Selesai *Driver*

l) Rancangan Antarmuka Profil *Driver*



Gambar 4.45 Racangan Antarmuka Profil *Driver*

m) Rancangan Antarmuka *Logout Driver*



Gambar 4.46 Rancangan Antarmuka *Logout* Driver

4.6 Implementasi

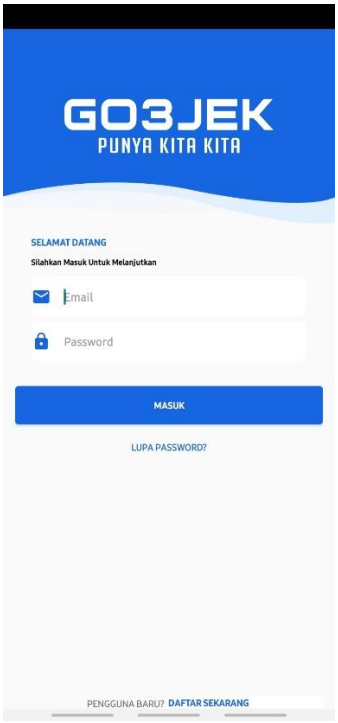
Implementasi antarmuka ataupun *interface* pada aplikasi jasa transportasi *online* menggunakan *firebase* dan *dijkstra* ini memiliki beberapa antarmuka diantaranya yaitu *interface login* dari sisi *customer* maupun *driver*, registrasi, halaman utama, proses orderan, histori orderan. Adapun implementasi antarmuka yang dimaksud seperti berikut:

4.6.1 Tampilan Antarmuka *Customer*

a) Tampilan *Splash Screen*

Gambar 4. 47 Tampilan *Splash Screen Customer*

b) Tampilan *Login*

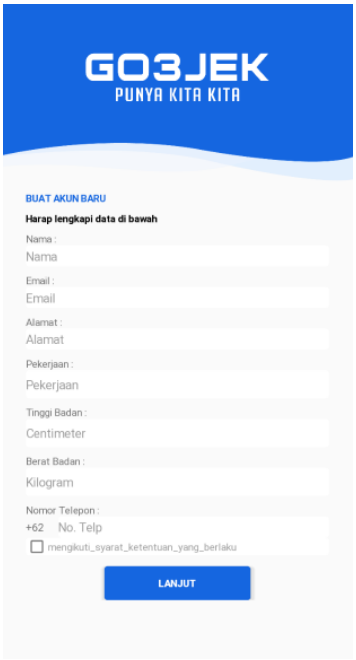


Gambar 4. 48 Tampilan *Login Customer*

c) Tampilan Daftar

Gambar 4. 49 Tampilan Daftar *Customer*

d) Tampilan *Input* Daata



Gambar 4. 50 Tampilan Isi Data *Customer*

e) Tampilan Menu Utama

Gambar 4. 51 Tampilan Halaman Menu *Customer*

f) Tampilan Halaman Go3Bike

Gambar 4. 52 Tampilan Halaman Go3Bike

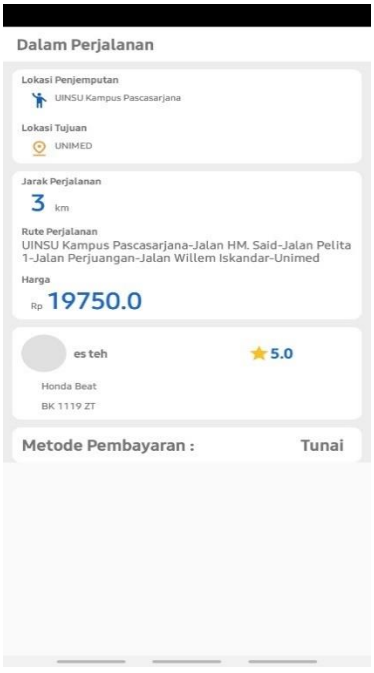
g) Tampilan Melakukan Orderan

Gambar 4. 53 Tampilan Melakukan Orderan

h) Tampilan Mendapatkan *Driver*

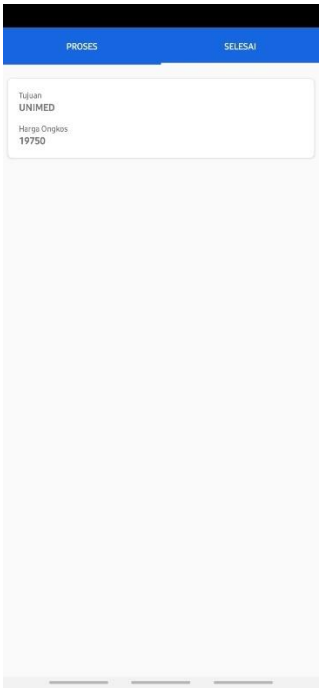
Gambar 4. 54 Tampilan Mendapatkan *Driver*

i) Tampilan Orderan Berlangsung



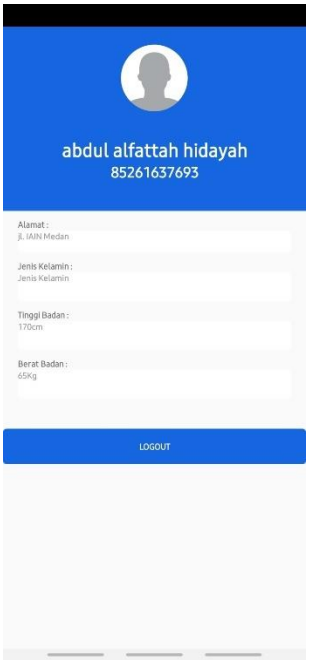
Gambar 4. 55 Tampilan Orderan Berlangsung *Customer*

j) Tampilan Menu *History*



Gambar 4. 56 Tampilan Menu *History Customer*

k) Tampilan Profil



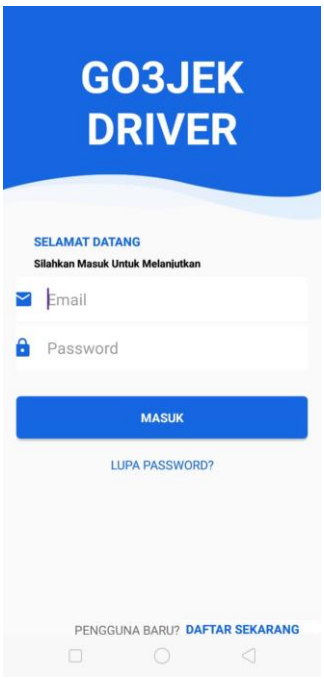
Gambar 4. 57 Tampilan Profil *Customer*

4.6.2 Tampilan Antarmuka *Driver*

a) Tampilan *Splash Screen*

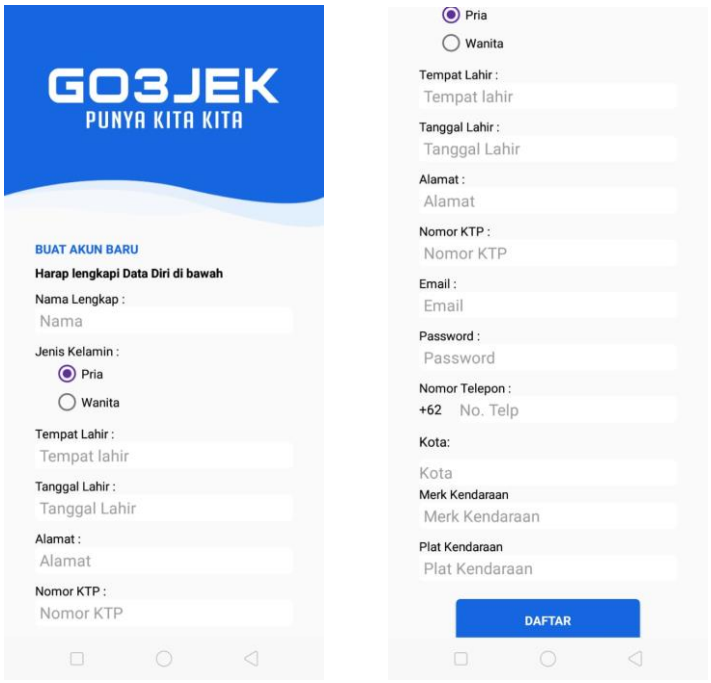
Gambar 4. 58 Tampilan *Splash Screen Driver*

b) Tampilan *Login*



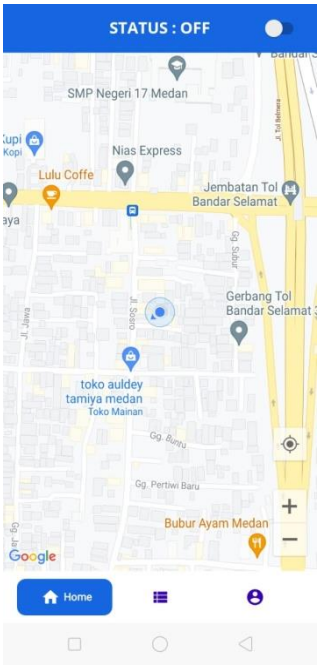
Gambar 4. 59 Tampilan *Login Driver*

c) Tampilan Daftar



Gambar 4. 60 Tampilan Halaman Daftar *Driver*

d) Tampilan Menu Utama (*offline*)



Gambar 4. 61 Tampilan Menu Utama *Offline*

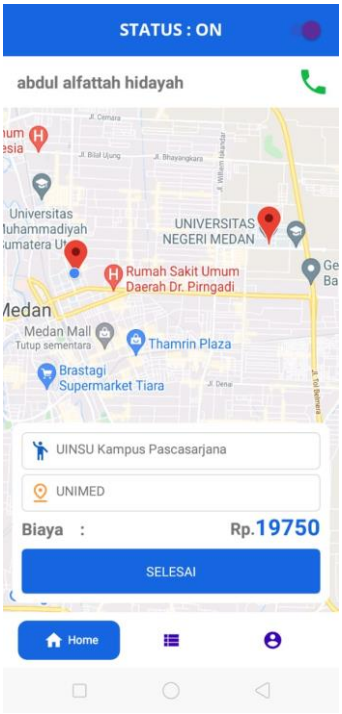
e) Tampilan Menu Utama (*online*)

Gambar 4. 62 Tampilan Menu Utama *Online*

f) Tampilan Orderan Masuk

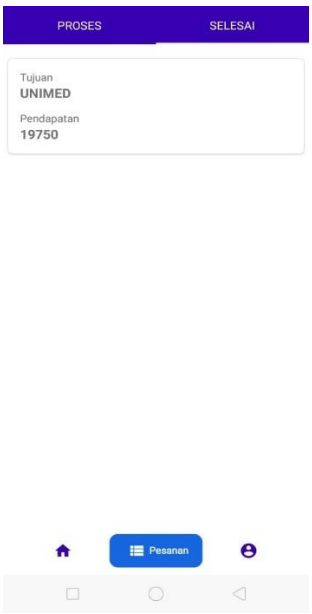
Gambar 4. 63 Tampilan Orderan Masuk

g) Tampilan Orderan Berlangsung



Gambar 4. 64 Tampilan Orderan Berlangsung

h) Tampilan *History*

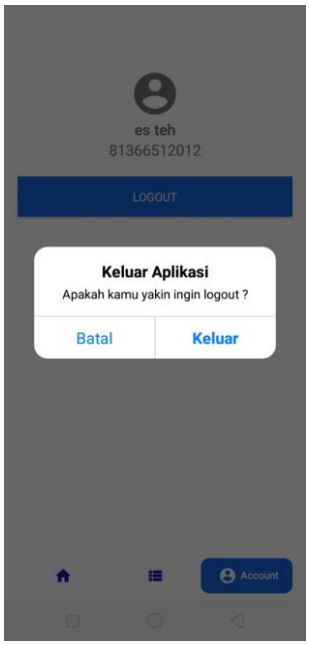


Gambar 4. 65 Tampilan *History Driver*

i) Tampilan Profil

Gambar 4. 66 Tampilan Profil *Driver*

j) Tampilan *Logout*



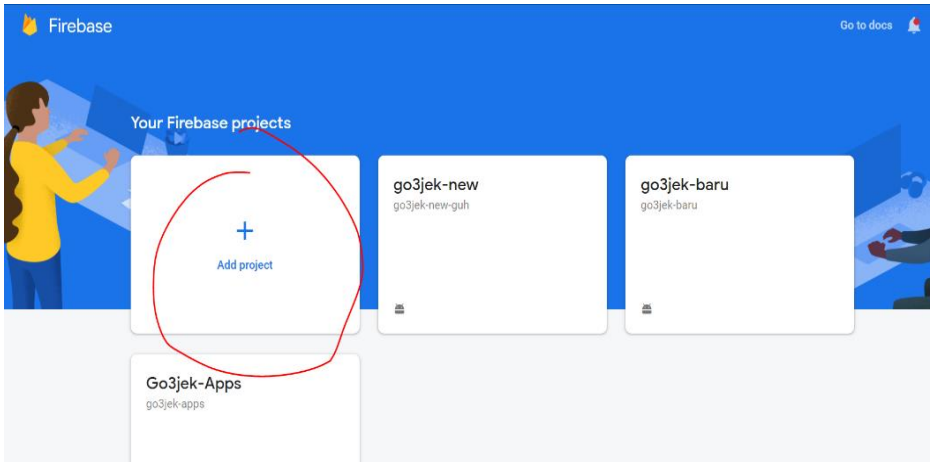
Gambar 4. 67 Tampilan *Logout Driver*

4.6.3 Implementasi *Firebase Auth*

Firebase banyak menyediakan *resourse authentication*, dan bekerjasama dengan perusahaan-perusahaan besar seperti *google, facebook, microsoft, github,* dan lainnya dengan mengambil data API pada perusahaan-perusahaan tersebut. Tentunya dalam kelebihan ini akan memudahkan pada *developer* dalam mengembangkan aplikasi yang didalamnya membutuhkan autentikasi dari bermacam-macam *resource* perusahaan-perusahaan besar di dunia. Autentikasi juga dapat dilakukan dengan menggunakan OTP (*One Time Password*) yang akan dikirimkan melalui *sms gateway* yang dikirimkan melalui nomor telepon.

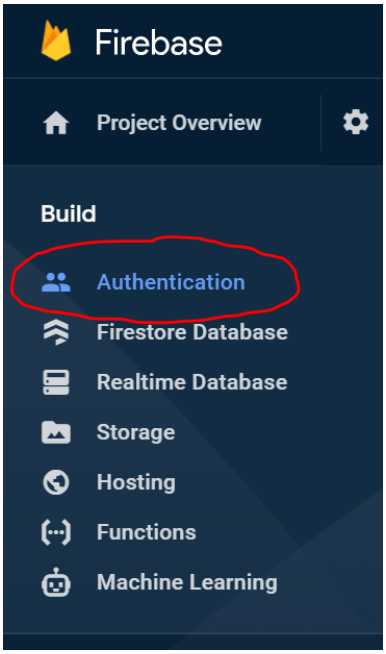
Adapun cara bagaimana mengimplementasikan layanan *firebase authentication* kedalam aplikasi android adalah sebagai berikut:

1. Melakukan pendaftaran *google account* kedalam layanan *firebase*. setelah melakukan pendaftaran. Pengguna akan dapat membuat sebuah projek pada aplikasi yang akan dibuat



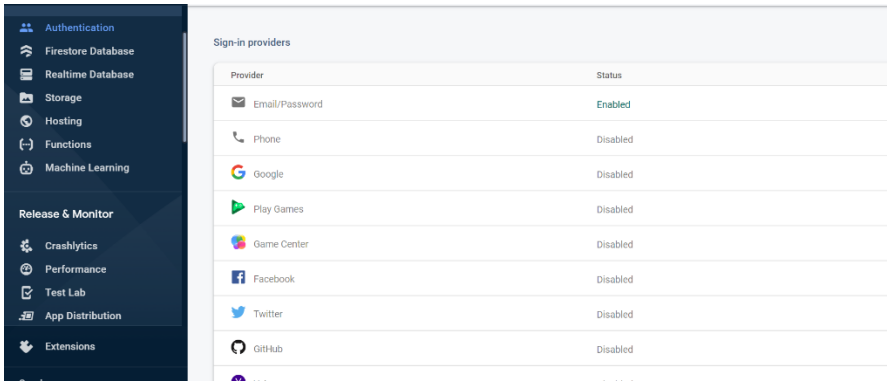
Gambar 4. 68 Halaman Menu Utama Pada *Firebase*

2. Setelah membuat projek, pengembang akan ditampilkan beberapa menu seperti berikut ini:



Gambar 4. 69 Layanan menu dari *project firebase*

3. Kemudian pengguna dapat mengaktifkan layanan autentikasi susuai kebutuhan pengguna ataupun pengembang. Pada penelitian ini penulis hanya menggunakan layanan *email* sebagai media autentikasi



Gambar 4. 70 Layanan yang disediakan *Firebase* sebagai autentikasi

4. Setelah melakukan pengaktifan layanan autentikasi yang diinginkan sesuai kebutuhan, selanjutnya adalah melakukan integrasi dengan aplikasi android yaitu dengan menggunakan *depedency java* dari *gradle*.

```
//Firebase Authentication
implementation 'com.google.firebase:firebase-auth-ktx'
```

4.6.3.1 Autentikasi Menggunakan *Email*

Untuk menggunakan autentikasi *email firebase*, pengguna harus melakukan *instance* kepada objek *FirebaseAuth*, dengan sintaks sebagai berikut:

```
private lateinit var auth: FirebaseAuth
// ...
// Initialize Firebase Auth
auth = Firebase.auth
```

Apabila telah melakukan *instance* kepada objek *FirebaseAuth*, selanjutnya adalah tahap melakukan pendaftaran akun baru dengan meneruskan alamat *email*. Adapun sintaksnya, yaitu:

```
auth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                // Sign in success, update UI with the
signed-in user's information
                Log.d(TAG, "createUserWithEmail:success")
                val user = auth.currentUser
                updateUI(user)
            } else {
                // If sign in fails, display a message to the
user.
                Log.w(TAG, "createUserWithEmail:failure",
task.exception)
                Toast.makeText(baseContext, "Authentication
failed.",
                        Toast.LENGTH_SHORT).show()
                updateUI(null)
            }
        }
```

Setelah sintakas *email* sudah berhasil diintegrasikan pada *firebase authentication*, selanjutnya adalah melakukan integrasi *function login* pada aplikasi. Adapun sintaks aktifitas *login* tidak jauh berbeda dengan sintaks membuat akun, yaitu seperti:

```
auth.signInWithEmailAndPassword(email, password)
```

```
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                // Sign in success, update UI with the
signed-in user's information
                Log.d(TAG, "signInWithEmail:success")
                val user = auth.currentUser
                updateUI(user)
            } else {
                // If sign in fails, display a message to the
user.
                Log.w(TAG, "signInWithEmail:failure",
task.exception)
                Toast.makeText(baseContext, "Authentication
failed.",
                        Toast.LENGTH_SHORT).show()
                updateUI(null)
            }
        }
```

Dalam menginisialisasi aktivitas pengguna apakah akun pengguna sudah *login* ataupun belum, dibutuhkan sintaks dalam mengintegrasikan aktivitas tersebut, yaitu:

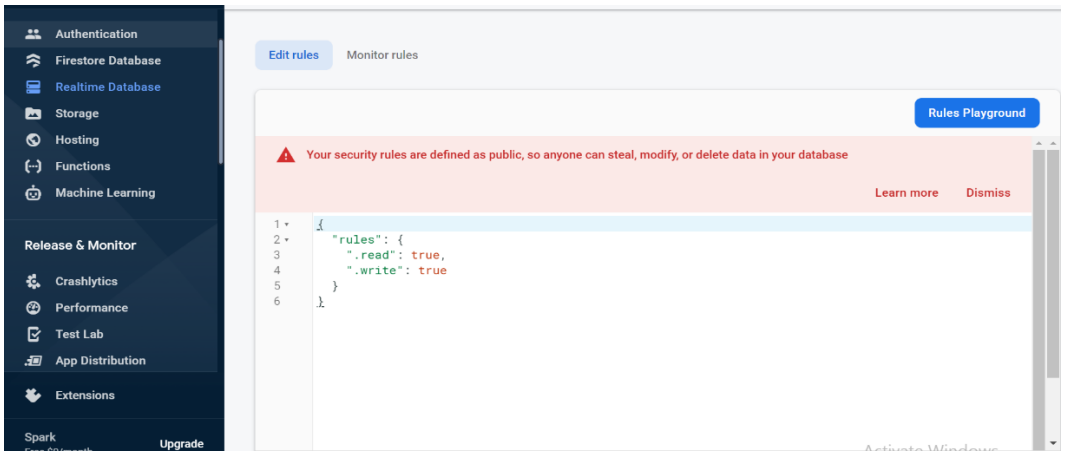
```
public override fun onStart() {
    super.onStart()
    // Check if user is signed in (non-null) and update UI
accordingly.
    val currentUser = auth.currentUser
    if(currentUser != null){
        reload();
    }
}
```

4.6.4 Implementasi *Firebase Realtime Database*

Dalam menambahkan *realtime database* kedalam aplikasi, pengguna wajib mendeklarasikan depedensi kedalam *gradle*

```
// firebase service
    implementation platform('com.google.firebase:firebase-
bom:27.0.0')
// Firebase Realtime Database
    implementation 'com.google.firebase:firebase-database'
```

Dalam mengimplementasikan *realtime database*, pengguna juga diwajibkan mengatur keamanan terkait hak akses siapa saja yang dapat melakukan proses *read* dan *write* data. Pada penelitian ini, aturan yang digunakan adalah dengan membenarkan seluruh hak akses *write* dan *read*, seperti berikut:



Gambar 4. 71 Aturan hak akses *realtime database* pada aplikasi GO3JEK

4.7 Pengujian Sistem

Pengujian sistem merupakan sebuah proses yang bertujuan untuk menguji kesesuaian pada sistem berdasarkan tujuan dari perencanaan pada penelitian yang telah dibuat. *Black box testing* merupakan sebuah tahapan pengujian yang didasarkan atas detail dan fungsi pada setiap menu pada sebuah sistem yang sudah dirancang. Adapun kerangka dan rancangan tabel pengujian aplikasi GO3JEK adalah sebagai berikut:

1. **Pengujian *Interface* Pendaftaran (Aplikasi *Customer*)**

Tabel 4. 37 Pengujian *BlackboxInterface* Pendaftaran *Customer*

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Keterangan
1	Pengujian pendaftaran menggunakan	*Customer* terdaftar dengan *email*	Sistem akan melakukan *insert* data ke	Berhasil

	firebase authentication layanan *email*	dan bisa melakukan *login*	*authentication* dan antarmuka menuju halaman utama	

2. Pengujian *Interface Login* (Aplikasi *Customer*)

Tabel 4. 38 Pengujian *BlackboxInterface Login Customer*

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Keterangan
1	Pengujian *Login* menggunakan *firebase authentication* layanan *email*	*Customer* akan menuju halaman ataupun *interface* menu utama	Sistem berhasil mengautentikasi dan menuju halaman menu utama	Berhasil

3. Pengujian *Interface* Menu Utama (Aplikasi *Customer*)

Tabel 4. 39 Pengujian *BlackboxInterface* Menu Utama*Customer*

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Keterangan
1	Menguji *button* GO3BIKE	Menuju halaman GO3BIKE	Sistem Menampilkan halaman GO3BIKE	Berhasil
2	Menguji *button history*	Menuju halaman *history*	Sistem Menampilkan halaman *history*	Berhasil
3	Menguji *button profile*	Menuju halaman *profile*	Sistem Menampilkan halaman *profile*	Berhasil

4. Pengujian *interface* GO3BIKE (Aplikasi *Customer*)

Tabel 4. 40 Pengujian *BlackboxInterface* GO3BIKE *Customer*

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Keterangan
1	Menguji *input* lokasi tujuan	Menampilkan list jalan yang sudah terdaftar dengan perhitungan *dijkstra*	Sistem menampilkan *icon marker* lokasi pada *maps*	Berhasil
2	Menguji *input* lokasi penjemputan	Menampilkan list jalan yang sudah terdaftar serta mengukur jarak dan rute perjalanan dengan perhitungan *dijkstra*, serta juga menghitug harga ongkos perjalanan	Sistem menampilkan *icon marker* lokasi pada *maps*, menampilkan jarak dan rute perjalanan melalui perhitungan algoritma *dijkstra*, serta menampilkan harga ongkos perjalanan	Berhasil
3	Menguji *button* pesan sekarang	Mendapatkan *driver* dan mendapatkan informasi data *driver*, serta posisi *driver*	Sistem menampilkan *toast* mencari *driver*, kemudian setelah mendapatkan *driver* sistem akan menampilkan data dari *driver*, beserta posisi *driver*	Berhasil

4	Menguji FCM yang diterima ketika *driver* sudah tiba di lokasi penjemputan	Menampilkan data perjalanan, dan juga data *driver*.	Sistem menampilkan data-data perjalanan beserta data *driver*, dan juga menampilkan informasi sedang dalam perjalanan	Berhasil

5. Pengujian *Interface History* (Aplikasi *Customer*)

Tabel 4. 41 Pengujian *BlackboxInterface History Customer*

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Keterangan
1	Menampilkan *list history* orderan	Sistem menampilkan *list* mengenai orderan	Sistem menampilkan *list-list* orderan yang ada	Berhasil

6. Pengujian *Interface Profile* (Aplikasi *Customer*)

Tabel 4. 42 Pengujian *BlackboxInterface Profile Customer*

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Keterangan
1	Menguji Tombol *Logout*	Sistem menampilkan *alert* dialog konfirmasi *logout*	Sistem menampilkan konfirmasi *logout*	Berhasil

7. Pengujian *Interface* Pendaftaran (Aplikasi *Driver*)

Tabel 4. 43 Pengujian *BlackboxInterface* Pendaftaran *Driver*

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Keterangan
1	Pengujian pendaftaran menggunakan *firebase authentication* layanan *email*	*Driver* terdaftar dengan *email* dan bisa melakukan *login*	Sistem akan melakukan *insert* data ke *authentication* dan antarmuka menuju halaman utama	Berhasil

8. Pengujian *Interface Login* (Aplikasi *Driver*)

Tabel 4. 44 Pengujian *BlackboxInterface Login Driver*

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Keterangan
1	Pengujian *Login* menggunakan *firebase authentication* layanan *email*	*Driver* menuju halaman ataupun *interface* menu utama	Sistem berhasil mengautentikasi dan menuju halaman menu	Berhasil

9. Pengujian *Interface* Menu Utama (Aplikasi *Driver*)

Tabel 4. 45 Pengujian *BlackboxInterface* Menu Utama *Driver*

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Keterangan
1	Pengujian pengaktifan tombol pekerjaan	*Driver* mendapatkan orderan setelah mengaktifkan pekerjaan	Sistem berhasil mendapatkan orderan	Berhasil

2	Pengujian tombol saya telah tiba	Setelah tombol saya telah tiba di *tap* nantinya akan berubah menjadi tombol selesai. Dan FCM tersebut dapat diterima pada aplikasi *Customer*	Sistem menampilkan tombol selesai, dan FCM diterima oleh *Customer*	Berhasil
3	Pengujian tombol selesai	Setelah tombol selesai di *tap* maka orderan akan selesai dan data orderan akan disimpan kedalam *database*	Orderan selesai, dan dapat mengambil orderan lainnya	Berhasil

10. Pengujian *Interface History* (Aplikasi *Driver*)

Tabel 4. 46 Pengujian *BlackboxInterface History Driver*

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Keterangan
1	Menampilkan *list history* orderan	Sistem menampilkan *list* mengenai orderan	Sistem menampilkan *list-list* orderan yang ada	Berhasil

11. Pengujian *Interface Profile* (Aplikasi *Driver*)

Tabel 4. 47 Pengujian *BlackboxInterface Profile Driver*

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Keterangan
1	Menguji Tombol *Logout*	Sistem menampilkan *alert* dialog konfirmasi *logout*	Sistem menampilkan konfirmasi *logout*	Berhasil

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah penulis lakukan yaitu mengimplementasikan *Firebase* dan algoritma *Dijkstra* dalam aplikasi transportasi *online* dapat ditarik beberapa kesimpulan, yaitu:

1. Peran firebase mempunyai pengaruh yang sangat baik terhadap aplikasi transportasi *online* yang dibangun, dikarenakan dapat mengakses *database* secara *realtime*, dimana *customer* dapat mengakses lokasi keberadaan *driver* melalui *maps* yang disediakan, dan juga status keaktifan *driver* yang menjadi masalah pada sistem yang sedang berjalan sebelumnya.
2. Algoritma *dijkstra* dapat dimanfaatkan dalam pemetaan pada aplikasi transportasi *online* yang dibuat untuk mencari rute-rute terpendek pada sebuah perjalanan dengan memanfaatkan *node-node* yang sudah ditentukan pada suatu *graf* yang berbobot

5.2 Saran

Saran yang dapat diberikan untuk pengembangan penelitian ini dimasa yang akan datang nantinya antara lain:

1. Perlu adanya pengembangan metode agar dapat menambahkan jumlah lokasi dan menampilkan gambar rute perjalanan pada maps yang sudah dihitung melalui algoritma *dijkstra*
2. Penambahan fitur memprioritaskan *driver* yang lebih dekat pada titik lokasi penjemputan agar dapat menerima orderan sesuai dengan jarak-jarak tertentu

DAFTAR PUSTAKA

AAbdullah, D. (2015). Perancangan Sistem Informasi Pendataan Siswa SMP Islam Swasta Darul Yatama Berbasis Web. *IJNS-Indonesian Journal on Networking and Security*, *4*(1).

Al Faruq, U. (2015). Rancang bangun aplikasi rekam medis poliklinik universitas trilogi. *Jurnal Informatika Ahmad Dahlan*, *9*(1).

Almuttaqin, G. (2016). Sistem Informasi Pendaftaran Pernikahan Berbasis Online Menggunakan Metode Waterfall (Study Kasus: Kantor Urusan Agama Kecamatan Mandau-Duri). *Jurnal Ilmiah Rekayasa dan Manajemen Sistem Informasi*, *2*(2), 52–55.

Angela, W., & Gani, A. (2016). Rancang Bangun Game Edukasi Berbasis Web Dan Android Menggunakan Adobe Flash Cs5 Dan Action Script 3.0. *IJIS-Indonesian Journal On Information System*, *1*(2).

Anindhita, W., Arisanty, M., & Rahmawati, D. (2016). Analisis Penerapan Teknologi Komunikasi Tepat Guna Pada Bisnis Transportasi Ojek Online (Studi pada Bisnis Gojek dan Grab Bike dalam Penggunaan Teknologi Komuniasi Tepat Guna untuk Mengembangkan Bisnis Transportasi). *Prosiding Seminar Nasional INDOCOMPAC*.

Anwar, S. N. (2015). *Perancangan Dan Implementasi Aplikasi Mobile Semarang Guidance Pada Android*.

Arifin, J., Zulita, L. N., & Hermawansyah, H. (2016). Perancangan Murottal Otomatis Menggunakan Mikrokontroller Arduino Mega 2560. *Jurnal Media Infotama*, *12*(1).

Azkiyah, R., Arifin, R., & Hufron, M. (2018). PENGARUH DIMENSI KUALITAS JASA TERHADAP LOYALITAS DENGAN KEPUASAN SEBAGAI VARIABEL INTERVENING PADA NASABAH BANK (Studi Kasus Pada Bank BRI Kantor Cabang Negara Bali). *Jurnal Ilmiah Riset Manajemen*, *7*(8).

Fajarianto, O., Iqbal, M., & Cahya, J. T. (2017). Sistem penunjang keputusan seleksi penerimaan karyawan dengan metode weighted product. *Jurnal Sisfotek Global*, *7*(1).

Fatoni, A., & Dwi, D. (2016). Rancang bangun sistem extreme programming sebagai metodologi pengembangan sistem. *PROSISKO: Jurnal Pengembangan Riset dan Observasi Sistem Komputer*, *3*(1).

Febriandirza, A. (2020). Perancangan Aplikasi Absensi Online Dengan Menggunakan Bahasa Pemrograman Kotlin. *Pseudocode*, *7*(2), 123–133.

Firly, N. (2018). *Create Your Own Android Application*. Elex Media Komputindo.

Harahap, M. K., & Khairina, N. (2017). Pencarian Jalur Terpendek dengan Algoritma Dijkstra. *SinkrOn*, *2*(2), 18–23.

Hartati, S., Dewi, N. A. K., Puastuti, D., Muslihudin, M., & Budi, N. S. (2017). Sistem Aplikasi Educhat Stmik Pringsewu Berbasis Android Sebagai Media Komunikasi dan Informasi. *Jurnal Nasional Teknologi dan Sistem Informasi*, *3*(1), 143–152.

Herliana, A., & Rasyid, P. M. (2016). Sistem Informasi monitoring pengembangan software pada tahap development berbasis web. *Jurnal Informatika*, *3*(1).

Ikhwan, A. (2020). *PENGEMBANGAN SISTEM RENTAL MOBIL*.

Irawan, M. D. (2017). Implementasi Kriptografi Vigenere Cipher dengan Php. *JurTI (Jurnal Teknologi Informasi)*, *1*(1), 11–21.

Jonathan, W., & Lestari, S. (2015). Sistem informasi UKM berbasis website pada desa Sumber Jaya. *Jurnal Teknologi Informasi dan Bisnis Pengabdian Masyarakat Darmajaya*, *1*(1), 1–16.

Juansyah, A. (2015). Pembangunan aplikasi child tracker berbasis assisted-global positioning system (a-gps) dengan platform android. *Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)*, *1*(1), 1–8.

Kusniyati, H., & Sitanggang, N. S. P. (2016). Aplikasi Edukasi Budaya Toba Samosir Berbasis Android. *Jurnal teknik informatika*, *9*(1).

Manueke, M., Tampi, G. B., & Londa, V. (2018). Persepsi Masyarakat Tentang Jasa Transportasi Berbasis Aplikasi Online Di Kota Manado (Studi Kasus Di Pt. Go-Jek). *Jurnal Administrasi Publik*, *4*(51).

Marisa, F., & Wijaya, I. D. (2016). Implementasi Google Speech Untuk Penentuan Level Pembelajaran Iqro'Berbasis Android. *JOINTECS (Journal of Information Technology and Computer Science)*, *1*(1).

Ningratri, Y. A. (2018). *Analisis Pengaruh Strategi Bauran Pemasaran Jasa (3P) terhadap Keputusan Mahasiswa Memilih STIM Sukma Medan*.

Pahendra, I., Thereza, N., & Mutho'a, M. F. (2019). PEMBUATAN APLIKASI OJEK ONLINE UNTUK MASYARAKAT SEPUTAR KAMPUS UNSRI INDRALAYA. *Prosiding Applicable Innovation of Engineering and Science Research*, *2019*, 475–479.

Pamungkas, C. A. (2019). Aplikasi Penghitung Jarak Koordinat Berdasarkan Latitude dan Longitude dengan Metode Euclidean Distance dan Metode Haversine. *Jurnal Informa*, *5*(2), 8–13.

Pratama, N. A., & Hermawan, C. (2016). Aplikasi Pembelajaran Tes Potensi Akademik Berbasis Android. *Jurnal Penelitian Dosen FIKOM (UNDA)*, *6*(1).

Samsudin, S., Irawan, M. D., & Harahap, A. H. (2019). MOBILE APP EDUCATION GANGGUAN PENCERNAAN MANUSIA BERBASIS MULTIMEDIA MENGGUNAKAN ADOBE ANIMATE CC. *JurTI (Jurnal Teknologi Informasi)*, *3*(2), 141–148.

Samsudin, S., Zufria, I., & Triase, T. (2019). *Augmented Reality Jejak Rasulullah SAW Dalam Menerima Wahyu Al-Qur'an*.

Sanad, E. A. W., Achmad, A., & Dewiani, D. (2018). Pemanfaatan Realtime Database di Platform Firebase Pada Aplikasi E-Tourism Kabupaten Nabire. *Jurnal Penelitian Enjiniring*, *22*(1), 20–26.

Setiani, B. (2015). Prinsip-prinsip pokok pengelolaan jasa transportasi udara. *Jurnal Ilmiah Widya*, *1*(1).

Sibarani, N. S., Munawar, G., & Wisnuadhi, B. (2018). Analisis Performa Aplikasi Android Pada Bahasa Pemrograman Java dan Kotlin. *Prosiding Industrial Research Workshop and National Seminar*, *9*, 319–324.

Suendri, S., Triase, T., & Afzalena, S. (2020). IMPLEMENTASI METODE JOB ORDER COSTING PADA SISTEM INFORMASI PRODUKSI BERBASIS WEB. *JS (JURNAL SEKOLAH)*, *4*(2), 97–106.

Susanti, E., Triyono, J., & Pi, R. (2016). Pengembangan sistem pemantau dan pengendali kendaraan menggunakan raspberry pi dan firebase. *Jurnal Informatika*, *1*, 144–153.

Tangkudung, E. S., Najoan, M. E., & Mamahit, D. J. (2018). Aplikasi Tata Cara Ibadah Berbasis Android. *Jurnal Teknik Informatika*, *13*(1).

Triase, T., & Aprilia, R. (2020). Implementasi Penyaluran Paket Online Shop Menggunakan Algoritma FIFO dan Dijkstra. *Query: Journal of Information Systems*, *4*(1).

Turang, D. A. O. (2015). Pengembangan Sistem Relay Pengendalian Dan Penghematan Pemakaian Lampu Berbasis Mobile. *Seminar Nasional Informatika (SEMNASIF)*, *1*(1).

Widiyanto, W. W. (2018). Analisa Metodologi Pengembangan Sistem Dengan Perbandingan Model Perangkat Lunak Sistem Informasi Kepegawaian Menggunakan Waterfall Development Model, Model Prototype, Dan Model Rapid Application Development (Rad). *Jurnal Informa: Jurnal Penelitian dan Pengabdian Masyarakat*, *4*(1), 34–40.

Yasin, A., & Yumarlin, M. Z. (2016). Evaluasi Web UJB menggunakan Golden Rules Of User Interface Design Theo Mandel. *SEMNASTEKNOMEDIA ONLINE*, *4*(1), 1-3–79.

Zakinah, N. (2019). *Efisiensi dan Dampak Ojek Online terhadap Kesejahteraan Driver Kota Makassar*. Universitas Islam Negeri Alauddin Makassar.

LAMPIRAN I

SURAT IJIN RISET

GO3JEK INDONESIA
Jl. Ringroad No.11 ABCDE , Kel. Tj. Sari,
Kec. Medan Selayang, Kota Medan, Sumatera Utara 20232
Phone : 0821 6338 7049 E-mail : go3jekgogogo@gmail.com

Nomor : 41/GO3JEK/XI/2020 Medan, 23 November 2020
Lampiran : -
Hal : **Balasan Permohonan Izin Riset**

Kepada Yth :
Dekan Bidang Akademik dan Kelembagaan
Universitas Islam Negeri Sumatera Utara
Jl. Williem Iskandar Pasar V Medan Estate 20371
Medan

Menanggapi surat saudara No. B.150/ST.I/ST.V.2/TL.00/11/2020 perihal " Permohonan Izin Riset ", pada mahasiswa :

No	Nama	NIM	Program Studi	Semester
1.	Abdul Alfattah Hidayah	0702162039	Sistem Informasi	IX (Sembilan)

Dengan ini diberitahukan bahwa **Kami Mengizinkan Mahasiswa Yang Dimaksud Untuk Melakukan Risetnya.**

Demikian surat balasan ini kami sampaikan, kami ucapkan terima kasih

Medan, 23 November 2020
Hormat Kami
GO3JEK INDONESIA

Borkat Hasibuan, S.Ag. MSP
Direktur Utama

LAMPIRAN II

BLACKBOX TESTING

Formulir Pengujian *Black-Box*

Tanggal Pengujian : 26 Agustus 2021

Nama Aplikasi : **GO3JEK**

Penguji : Ali Ikhwan, M.Kom

1. Pengujian *Interface* Pendaftaran (Aplikasi *Customer*)

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Hasil Pengujian	
				Sesuai	Tidak Sesuai
1	Pengujian pendaftaran menggunakan *firebase authentication* layanan *email*	*Customer* terdaftar dengan *email* dan bisa melakukan *login*	Sistem akan melakukan *insert* data ke *authentication* dan antarmuka menuju halaman utama	✓	

2. Pengujian *Interface Login* (Aplikasi *Customer*)

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Hasil Pengujian	
				Sesuai	Tidak Sesuai
1	Pengujian *Login* menggunakan *firebase authentication* layanan *email*	*Customer* akan menuju halaman ataupun *interface* menu utama	Sistem berhasil mengautentikasi dan menuju halaman menu utama	✓	

3. Pengujian *Interface* Menu Utama (Aplikasi *Customer*)

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Hasil Pengujian	
				Sesuai	Tidak Sesuai
1	Menguji *button* GO3BIKE	Menuju halaman GO3BIKE	Sistem Menampilkan halaman GO3BIKE	✓	
2	Menguji *button history*	Menuju halaman *history*	Sistem Menampilkan halaman *history*	✓	
3	Menguji *button profile*	Menuju halaman *profile*	Sistem Menampilkan halaman *profile*	✓	

4. **Pengujian *interface* GO3BIKE (Aplikasi *Customer*)**

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Hasil Pengujian	
				Sesuai	Tidak Sesuai
1	Menguji *input* lokasi tujuan	Menampilkan list jalan yang sudah terdaftar dengan perhitungan *dijkstra*	Sistem menampilkan *icon marker* lokasi pada *maps*	✓	
2	Menguji *input* lokasi penjemputan	Menampilkan list jalan yang sudah terdaftar serta mengukur jarak dan rute perjalanan dengan perhitungan *dijkstra,* serta juga menghitug harga ongkos perjalanan	Sistem menampilkan *icon marker* lokasi pada *maps,* menampilkan jarak dan rute perjalanan melalui perhitungan algoritma *dijkstra*, serta menampilkan harga ongkos perjalanan	✓	
3	Menguji *button* pesan sekarang	Mendapatkan *driver* dan mendapatkan informasi data *driver*, serta posisi *driver*	Sistem menampilkan *toast* mencari *driver*, kemudian setelah mendapatkan *driver* sistem akan menampilkan data dari *driver*, beserta posisi *driver*	✓	
4	Menguji FCM yang diterima ketika *driver* sudah tiba di lokasi penjemputan	Menampilkan data perjalanan, dan juga data *driver*.	Sistem menampilkan data-data perjalanan beserta data *driver*, dan juga menampilkan informasi sedang dalam perjalanan	✓	

5. **Pengujian *Interface History* (Aplikasi *Customer*)**

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Hasil Pengujian	
				Sesuai	Tidak Sesuai
1	Menampilkan *list history* orderan	Sistem menampilkan *list* mengenai orderan	Sistem menampilkan *list-list* orderan yang ada	✓	

6. **Pengujian *Interface Profile* (Aplikasi *Customer*)**

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Hasil Pengujian	
				Sesuai	Tidak Sesuai
1	Menguji Tombol *Logout*	Sistem menampilkan *alert* dialog konfirmasi *logout*	Sistem menampilkan konfirmasi *logout*	✓	

7. **Pengujian *Interface* Pendaftaran (Aplikasi *Driver*)**

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Hasil Pengujian	
				Sesuai	Tidak Sesuai
1	Pengujian pendaftaran menggunakan *firebase authentication* layanan *email*	*Driver* terdaftar dengan *email* dan bisa melakukan *login*	Sistem akan melakukan *insert* data ke *authentication* dan antarmuka menuju halaman utama	✓	

8. **Pengujian *Interface Login* (Aplikasi *Driver*)**

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Hasil Pengujian	
				Sesuai	Tidak Sesuai
1	Pengujian *Login* menggunakan *firebase authentication* layanan *email*	*Driver* menuju halaman ataupun *interface* menu utama	Sistem berhasil mengautentikasi dan menuju halaman menu utama	✓	

9. **Pengujian *Interface* Menu Utama (Aplikasi *Driver*)**

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Hasil Pengujian	
				Sesuai	Tidak Sesuai
1	Pengujian pengaktifan tombol pekerjaan	*Driver* mendapatkan orderan setelah mengaktifkan pekerjaan	Sistem berhasil mendapatkan orderan	✓	

2	Pengujian tombol saya telah tiba	Setelah tombol saya telah tiba di *tap* nantinya akan berubah menjadi tombol selesai. Dan FCM tersebut dapat diterima pada aplikasi *Customer*	Sistem menampilkan tombol selesai, dan FCM diterima oleh *Customer*	✓	
3	Pengujian tombol selesai	Setelah tombol selesai di *tap* maka orderan akan selesai dan data orderan akan disimpan kedalam *database*	Orderan selesai, dan dapat mengambil orderan lainnya	✓	

10. Pengujian *Interface History* (Aplikasi *Driver*)

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Hasil Pengujian	
				Sesuai	Tidak Sesuai
1	Menampilkan *list history* orderan	Sistem menampilkan *list* mengenai orderan	Sistem menampilkan *list-list* orderan yang ada	✓	

11. Pengujian *Interface Profile* (Aplikasi *Driver*)

No	Skenario Pengujian	Hasil yang diharapkan	Hasil yang Didapatkan	Hasil Pengujian	
				Sesuai	Tidak Sesuai
1	Menguji Tombol *Logout*	Sistem menampilkan *alert* dialog konfirmasi *logout*	Sistem menampilkan konfirmasi *logout*	✓	

Medan, 26 Agustus 2021

Penguji *Black-Box Testing*

ALI IKHWAN M.Kom

LAMPIRAN III

Source Code

LoginActivity.kt (*Customer*)

```
package
com.gotrijek.go3jekcustomer.beta

import android.content.Intent
import
androidx.appcompat.app.AppCompatActiv
ity
import android.os.Bundle
import android.view.View
import android.widget.*
import com.gotrijek.go3jekcustomer.R
import
com.gotrijek.go3jekcustomer.helper.Co
nstans
import
com.gotrijek.go3jekcustomer.helper.Se
ssionManager
import
com.gotrijek.go3jekcustomer.models.re
altimedb.Customer
import
com.gotrijek.go3jekcustomer.presenter
.beta.LoginPresenter
import
com.gotrijek.go3jekcustomer.presenter
.beta.LoginView
import
com.gotrijek.go3jekcustomer.ui.regist
er.RegisterActivity
import
kotlinx.android.synthetic.main.activi
ty_login2.*

class Login2Activity :
AppCompatActivity(), LoginView {

    private lateinit var edtEmail:
EditText
    private lateinit var edtPass:
EditText
    private lateinit var btnLogin:
Button
    private lateinit var progress:
ProgressBar
    private lateinit var session:
SessionManager
    private lateinit var
txtRegistrasi: TextView

    override fun
onCreate(savedInstanceState: Bundle?)
{
super.onCreate(savedInstanceState)

setContentView(R.layout.activity_logi
n2)

        // binding layouting
        edtEmail =
findViewById(R.id.email)
        edtPass =
findViewById(R.id.password)
        btnLogin =
findViewById(R.id.login)
        progress =
findViewById(R.id.progress)
        txtRegistrasi =
findViewById(R.id.registration)

        Toast.makeText(this,
Constans.token,
Toast.LENGTH_SHORT).show()

        // presenter
        val presenter =
LoginPresenter(this)

        // session manager
        session =
SessionManager(this)

        btnLogin.setOnClickListener {
            if
(edtEmail.text.toString().isEmpty()
||
edtPass.text.toString().isEmpty()){
                Toast.makeText(this,
"Field email dan password wajib diisi
terlebih dahulu",
Toast.LENGTH_SHORT).show()

return@setOnClickListener
            }

            val emailTxt =
edtEmail.text.toString()
            val pass =
edtPass.text.toString()

            // presenter executre dan
show progress

presenter.loginUser(emailTxt, pass)

presenter.updateToken(Constans.token,
```

```
session.uidCustomer.toString())
            progress.visibility =
View.VISIBLE
            btnLogin.visibility =
View.GONE
        }


txtRegistrasi.setOnClickListener {
            val myIntent =
Intent(this,
RegisterActivity::class.java)
            startActivity(myIntent)
            finish()
        }
    }

    override fun
onErrorMessage(messageError: String)
{
        progress.visibility =
View.GONE
        btnLogin.visibility =
View.VISIBLE
        Toast.makeText(this,
messageError,
Toast.LENGTH_SHORT).show()
    }

    override fun suksesLogin(state:
Boolean, customer: Customer?,
uidCustomer: String?) {
       if (state == true) {
            progress.visibility =
View.GONE
            btnLogin.visibility =
View.VISIBLE
            // set session
            session.login = true
            session.uidCustomer =
customer?.uid
            session.noTelepon =
customer?.nomor_telepon
            session.namaLengkap =
customer?.nama_lengkap
            //  intent
            val myIntent =
Intent(this,
MenuActivity::class.java)

myIntent.putExtra(TAG_CUSTOMER,
customer)
            startActivity(myIntent)
            finish()
       } else {
            // intent
            val myIntent =
Intent(this,
DaftarActivity::class.java)

myIntent.putExtra(UID_CUSTOMER,
uidCustomer)
            startActivity(myIntent)
            finish()
       }
    }

    companion object {
        const val TAG_CUSTOMER =
"CUSTOMER_LOGIN_TAG"
        const val UID_CUSTOMER =
"UID_CUSTOMER"
    }
}
```

DaftarActivity.kt (*Custoner*)

```
package
com.gotrijek.go3jekcustomer.beta

import android.content.Intent
import
androidx.appcompat.app.AppCompatActiv
ity
import android.os.Bundle
import android.widget.Button
import android.widget.CheckBox
import android.widget.EditText
import android.widget.Toast
import com.gotrijek.go3jekcustomer.R
import
com.gotrijek.go3jekcustomer.models.re
altimedb.Customer
import
com.gotrijek.go3jekcustomer.presenter
.beta.DaftarPresenter
import
com.gotrijek.go3jekcustomer.presenter
.beta.DaftarView

class DaftarActivity :
AppCompatActivity(), DaftarView {

    private lateinit var edtNama:
EditText
    private lateinit var edtAlamat:
EditText
    private lateinit var edtEmail:
EditText
    private lateinit var
edtPekerjaan: EditText
    private lateinit var
edtBeratBadan: EditText
    private lateinit var
edtTinggiBadan: EditText
    private lateinit var edtNotelp:
```

```
EditText
    private lateinit var checkSyarat:
CheckBox
    private lateinit var btnLanjut:
Button


    override fun
onCreate(savedInstanceState: Bundle?)
{

super.onCreate(savedInstanceState)

setContentView(R.layout.activity_daft
ar)

        // binding layouting
        edtNama =
findViewById(R.id.daftar_nama)
        edtAlamat =
findViewById(R.id.daftar_alamat)
        edtEmail =
findViewById(R.id.daftar_email)
        edtPekerjaan =
findViewById(R.id.daftar_pekerjaan)
        edtBeratBadan =
findViewById(R.id.daftar_berat_badan)
        edtTinggiBadan =
findViewById(R.id.daftar_tinggi_badan
)
        edtNotelp =
findViewById(R.id.daftar_no_telp)
        checkSyarat =
findViewById(R.id.daftar_chcek)
        btnLanjut =
findViewById(R.id.daftar_btn_lanjut)

        // presenter
        val presenter =
DaftarPresenter(this)

        val extras = intent?.extras
        if (extras != null) {


btnLanjut.setOnClickListener {
                if
(checkSyarat.isChecked) {

                    val uidCustomer =
extras.getString(Login2Activity.UID_C
USTOMER)
                    val nama =
edtNama.text.toString()
                    val alamat =
edtAlamat.text.toString()
                    val email =
edtEmail.text.toString()
                    val pekerjaan =
edtPekerjaan.text.toString()
                    val beratBadan =
edtBeratBadan.text.toString()
                    val tinggiBadan =
edtTinggiBadan.text.toString()
                    val noTelp =
edtNotelp.text.toString()

                    val dataCustomer
= Customer(

uidCustomer.toString(),
                        nama,
                        noTelp,
                        beratBadan,
                        tinggiBadan,
                        alamat,
                        pekerjaan,
                        email
                    )


presenter.addCustomer(dataCustomer)
                } else {

Toast.makeText(this, "Harap setujui
syarat dan ketentuan dahulu",
Toast.LENGTH_SHORT).show()
                }
            }
        }
    }

    override fun
suksesTerdaftar(state: Boolean,
customer: Customer) {
        if (state == true) {
            val intent = Intent(this,
MenuActivity::class.java)
            startActivity(intent)
            finish()
        }
    }

    override fun
onFailureMessage(msg: String) {
        Toast.makeText(this, msg,
Toast.LENGTH_SHORT).show()
    }
}
```

MenuActivity.kt (*Customer*)

```
package
com.gotrijek.go3jekcustomer.beta

import android.content.Intent
```

```
import
androidx.appcompat.app.AppCompatActiv
ity
import android.os.Bundle
import android.widget.LinearLayout
import com.gotrijek.go3jekcustomer.R
import
com.gotrijek.go3jekcustomer.ui.main.h
istori.HistoriFragment

class MenuActivity :
AppCompatActivity() {

    private lateinit var homeBike:
LinearLayout
    private lateinit var homeHistory:
LinearLayout
    private lateinit var
profileHistory: LinearLayout

    override fun
onCreate(savedInstanceState: Bundle?)
{

super.onCreate(savedInstanceState)

setContentView(R.layout.activity_menu
)

        homeBike =
findViewById(R.id.home_bike)
        homeHistory =
findViewById(R.id.home_history)
        profileHistory =
findViewById(R.id.home_profile)

        homeBike.setOnClickListener {
            val intent = Intent(this,
Go3BikeActivity::class.java)
            startActivity(intent)
        }


homeHistory.setOnClickListener {
            val myIntent =
Intent(this,
HistoriActivity::class.java)
            startActivity(myIntent)
        }


profileHistory.setOnClickListener {
            val myIntent =
Intent(this,
AkunActivity::class.java)
            startActivity(myIntent)
        }
    }
}
```

AkunActivity.kt (*Customer*)

```
package
com.gotrijek.go3jekcustomer.beta

import
androidx.appcompat.app.AppCompatActiv
ity
import android.os.Bundle
import android.view.View
import android.widget.TextView
import
com.google.firebase.auth.FirebaseAuth
import com.gotrijek.go3jekcustomer.R
import
com.gotrijek.go3jekcustomer.helper.Se
ssionManager
import
com.gotrijek.go3jekcustomer.models.re
altimedb.Customer
import
com.gotrijek.go3jekcustomer.presenter
.beta.AkunPresenter
import
com.gotrijek.go3jekcustomer.presenter
.beta.CustomerView
import
kotlinx.android.synthetic.main.activi
ty_akun.*

class AkunActivity :
AppCompatActivity(), CustomerView {

    override fun
onCreate(savedInstanceState: Bundle?)
{

super.onCreate(savedInstanceState)

setContentView(R.layout.activity_akun
)

        val session =
SessionManager(this)

        akun_nama.text =
session.namaLengkap
        akun_no_telepon.text =
session.noTelepon

        val presenter =
AkunPresenter(this)

presenter.getDataProfile(session.uidC
ustomer.toString())
```

```
akun_logout.setOnClickListener {
            session.login = false
            session.namaLengkap = ""
            session.noTelepon = ""
            session.token = ""
            session.uidCustomer = ""

FirebaseAuth.getInstance().signOut()
            finishAffinity()
            System.exit(0)
        }
    }

    override fun
dataCustomer(customer: Customer) {
        akun_loading.visibility =
View.GONE
        layout_form.visibility =
View.VISIBLE

        akun_alamat.text =
customer.alamat
        akun_berat_badan.text =
customer.berat_badan
        akun_tinggi_badan.text =
customer.tinggi_badan
    }
}
```

Go3BikeActivity.kt (*Customer*)

```
package
com.gotrijek.go3jekcustomer.beta

import android.Manifest
import
android.annotation.SuppressLint
import android.app.Activity
import
android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.content.IntentFilter
import android.net.Uri
import
androidx.appcompat.app.AppCompatActiv
ity
import android.os.Bundle
import android.os.Looper
import android.util.Log
import android.view.View
import android.widget.*
import
androidx.localbroadcastmanager.conten
t.LocalBroadcastManager
import
com.gmail.samehadar.iosdialog.IOSDial
og
import
com.google.android.gms.location.*
import
com.google.android.gms.maps.CameraUpd
ateFactory
import
com.google.android.gms.maps.GoogleMap
import
com.google.android.gms.maps.MapView
import
com.google.android.gms.maps.OnMapRead
yCallback
import
com.google.android.gms.maps.model.Bit
mapDescriptorFactory
import
com.google.android.gms.maps.model.Lat
Lng
import
com.google.android.gms.maps.model.Mar
ker
import
com.google.android.gms.maps.model.Mar
kerOptions
import com.gotrijek.go3jekcustomer.R
import
com.gotrijek.go3jekcustomer.helper.Co
nstans
import
com.gotrijek.go3jekcustomer.helper.Dj
ikstraAlgorithm
import
com.gotrijek.go3jekcustomer.helper.Se
ssionManager
import
com.gotrijek.go3jekcustomer.models.Cu
stomLokasiDjikstra
import
com.gotrijek.go3jekcustomer.models.fc
m.DataNotifikasi
import
com.gotrijek.go3jekcustomer.models.fc
m.NotifikasiModel
import
com.gotrijek.go3jekcustomer.models.re
altimedb.DriverBio
import
com.gotrijek.go3jekcustomer.models.re
altimedb.LokasiDriver
import
com.gotrijek.go3jekcustomer.models.re
altimedb.Pesanan
import
com.gotrijek.go3jekcustomer.services.
MyFirebaseService
import
com.gotrijek.go3jekcustomer.ui.main.g
```

```
obike.GoBikeCallback
import
com.gotrijek.go3jekcustomer.ui.main.g
obike.GoBikeFragment
import
com.gotrijek.go3jekcustomer.ui.main.g
obike.GoBikePresenter
import
com.gotrijek.go3jekcustomer.ui.main.h
istori.proses.ProsesFragment
import
com.vmadalin.easypermissions.EasyPerm
issions
import io.reactivex.Observable
import
io.reactivex.android.schedulers.Andro
idSchedulers
import
io.reactivex.schedulers.Schedulers
import
kotlinx.android.synthetic.main.activi
ty_costumer_dalam_perjalanan.*
import
kotlinx.android.synthetic.main.activi
ty_go3bike.*
import
kotlinx.android.synthetic.main.activi
ty_go3bike.harga
import
kotlinx.android.synthetic.main.activi
ty_go3bike.jarak
import
kotlinx.android.synthetic.main.activi
ty_go3bike.titik_jemput
import
kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import
kotlinx.coroutines.Dispatchers.IO
import kotlinx.coroutines.delay
import kotlinx.coroutines.launch
import java.text.DecimalFormat

class Go3BikeActivity :
AppCompatActivity(),
OnMapReadyCallback, GoBikeCallback {

    private lateinit var mMapView:
MapView
    private lateinit var mMapsGogle:
GoogleMap
    private lateinit var presenter:
GoBikePresenter

    private lateinit var
layourMauKemana: LinearLayout
    private lateinit var
layoutSiapOrder: LinearLayout
    private lateinit var
layoutDataDiDapatkan: RelativeLayout

    private lateinit var
edtMauKemana: EditText
    private lateinit var
edtTitikJemput: TextView
    private lateinit var
edtTitikAntar: TextView
    private lateinit var btnTelefon:
ImageView

    private lateinit var
fusedLocationProvider:
FusedLocationProviderClient
    private lateinit var
locationCallback: LocationCallback
    private val locationRequest =
LocationRequest()

    private lateinit var session:
SessionManager

    private val iosLoadingPopup by
lazy {
        IOSDialog.Builder(this)

.setTitleColorRes(R.color.gray)
            .setTitle("Mencari
Driver")
            .setCancelable(false)
            .build()
    }

    // receiver broadcast
    private val receiver = object :
BroadcastReceiver() {
        override fun onReceive(ctx:
Context?, intent: Intent?) {
            val filter =
MyFirebaseService.INTENT_FILTER
            if
(intent?.action.equals(filter)) {
                val uidDriver =
intent?.extras?.getString(MyFirebaseS
ervice.UID_DRIVER)
                val lokasiJemput =
intent?.extras?.getString(MyFirebaseS
ervice.LOKASI_JEMPUT)
                val lokasiTujuan =
intent?.extras?.getString(MyFirebaseS
ervice.LOKASI_ANTAR)
                val jarak =
intent?.extras?.getDouble(MyFirebaseS
ervice.JARAK)
                val rute =
intent?.extras?.getString(MyFirebaseS
ervice.RUTE)
                val harga =
intent?.extras?.getDouble(MyFirebaseS
```

```
ervice.HARGA)
                val namaDriver =
intent?.extras?.getString(MyFirebaseS
ervice.NAMA_DRIVER)
                val merkKendaraan =
intent?.extras?.getString(MyFirebaseS
ervice.MERK_KENDARAAN)
                val platKendaraan =
intent?.extras?.getString(MyFirebaseS
ervice.PLAT_KENDARAAN)

                val myIntent =
Intent(this@Go3BikeActivity,
CostumerDalamPerjalananActivity::clas
s.java)

myIntent.putExtra(MyFirebaseService.U
ID_DRIVER, uidDriver)

myIntent.putExtra(MyFirebaseService.L
OKASI_JEMPUT, lokasiJemput)

myIntent.putExtra(MyFirebaseService.L
OKASI_ANTAR, lokasiTujuan)

myIntent.putExtra(MyFirebaseService.J
ARAK, jarak)

myIntent.putExtra(MyFirebaseService.R
UTE, rute)

myIntent.putExtra(MyFirebaseService.H
ARGA, harga)

myIntent.putExtra(MyFirebaseService.N
AMA_DRIVER, namaDriver)

myIntent.putExtra(MyFirebaseService.M
ERK_KENDARAAN, merkKendaraan)

myIntent.putExtra(MyFirebaseService.P
LAT_KENDARAAN, platKendaraan)


startActivity(myIntent)

            }
        }
    }

    override fun onStart() {
        super.onStart()

LocalBroadcastManager.getInstance(thi
s)

.registerReceiver(receiver,
IntentFilter(MyFirebaseService.INTENT
_FILTER))
    }

    override fun onStop() {
        super.onStop()

LocalBroadcastManager.getInstance(thi
s)

.unregisterReceiver(receiver)
    }


@SuppressLint("MissingPermission")
    override fun
onCreate(savedInstanceState: Bundle?)
{

super.onCreate(savedInstanceState)

setContentView(R.layout.activity_go3b
ike)

        // binding
        mMapView =
findViewById(R.id.map)
        edtMauKemana =
findViewById(R.id.locationSearch)
        layourMauKemana =
findViewById(R.id.layoutOrder)
        layoutSiapOrder =
findViewById(R.id.layoutOrder1)
        layoutDataDiDapatkan =
findViewById(R.id.layout_data_driver)
        edtTitikJemput =
findViewById(R.id.titik_jemput)
        edtTitikAntar =
findViewById(R.id.titik_antar)
        btnTelefon =
findViewById(R.id.call)

        // presenter
        presenter =
GoBikePresenter(this)

        // session manager
        session =
SessionManager(this)

        if(hasLocationPermission()) {
            // insialisasi map view
agar memunculkan map

mMapView.onCreate(savedInstanceState)

mMapView.getMapAsync(this)

            // fused location
            // insialisasi location
request
```

```
            locationRequest.priority
=
LocationRequest.PRIORITY_HIGH_ACCURAC
Y
            locationRequest.interval
= 5000

locationRequest.fastestInterval =
4000

locationRequest.smallestDisplacement

            // callback dari
fusedlcaotion
            locationCallback = object
: LocationCallback() {

@SuppressLint("CheckResult")
                override fun
onLocationResult(locationResult:
LocationResult?) {
                    val realtimeLat =
locationResult?.lastLocation?.latitud
e
                    val realtimeLng =
locationResult?.lastLocation?.longitu
de

                    val
latFromContant =
Constans.myLatLng?.latitude
                    val
lngFromConstant =
Constans.myLatLng?.longitude

                    // Jika realtime
latitde dan longitude tidak sama
dengan nilai lat dan lng pada object
constan
                    if (realtimeLat
!= latFromContant && realtimeLng !=
lngFromConstant) {
                        // ser
location (artinya lat dan lng
pengguna berubah)

Constans.myLatLng =
LatLng(realtimeLat ?: 0.0,
realtimeLng ?: 0.0)
                    }

                    // apabila di
temukan driver lebih dari 0 dari
hashset berikut
                    if
(Constans.driverFounded.size > 0) {

Observable.fromIterable(Constans.driv
erFounded)
```

```
.subscribeOn(Schedulers.newThread())

.observeOn(AndroidSchedulers.mainThre
ad())

.subscribe( { lokasiDriver:
LokasiDriver? ->

Log.d(GoBikeFragment.TAG,
"onLocationResult:
${lokasiDriver?.geoLocation?.latitude
}")

presenter.buatMarker(lokasiDriver as
LokasiDriver)
                            }, {

Log.d(GoBikeFragment.TAG,
"onLocationResult:
${it.localizedMessage}")
                            })
                    }
                }
            }

            // fused locatioon di
insialiasai
            fusedLocationProvider =
LocationServices.getFusedLocationProv
iderClient(this)

fusedLocationProvider.requestLocation
Updates(locationRequest,
locationCallback, Looper.myLooper())

            // event listener edt mau
ke mana hari ini

edtMauKemana.setOnClickListener {
                val myIntent =
Intent(this,
LokasiDjikstraActivity::class.java)

startActivityForResult(myIntent,
CODE_INTENT_MAU_KEMANA)
            }

        } else {
            requestPermission()
        }

    }

    override fun onResume() {
        mMapView.onResume()
        super.onResume()
    }
```

```
    @SuppressLint("SetTextI18n",
"UseCompatLoadingForDrawables")
    override fun
onActivityResult(requestCode: Int,
resultCode: Int, data: Intent?) {

super.onActivityResult(requestCode,
resultCode, data)
        if (resultCode ==
Activity.RESULT_OK && requestCode ==
CODE_INTENT_MAU_KEMANA) {

            val feedbackLokasi =
data?.extras?.getParcelable<CustomLok
asiDjikstra>(LokasiDjikstraActivity.T
AG_MAU_KEMANA)
            if (feedbackLokasi !=
null) {

layourMauKemana.visibility =
View.GONE

layoutSiapOrder.visibility =
View.VISIBLE

                pesan1.background =
getDrawable(R.drawable.form3)
                pesan1.text = "Mohon
Tunggu"
                pesan1.isEnabled =
false

                edtTitikJemput.text =
"UINSU Kampus Pascasarjana"
                edtTitikAntar.text =
feedbackLokasi.namaLokasi.toString()

                // save di nilai
constanst tujuan
                Constans.lokasiJemput
=
CustomLokasiDjikstra(edtTitikJemput.t
ext.toString(), 3.60055, 98.68158)
                Constans.lokasiAntar
=
CustomLokasiDjikstra(edtTitikAntar.te
xt.toString(),
feedbackLokasi.latitude,
feedbackLokasi.longitude)

                val listMarker =
ArrayList<CustomLokasiDjikstra>()
                // marker lokasi
jemput
                listMarker.add(

CustomLokasiDjikstra("UINSU Kampus
Pascasarjana", 3.60055, 98.68158)
                )
                // marker lokasi
antar
                listMarker.add(

CustomLokasiDjikstra(feedbackLokasi.n
amaLokasi, feedbackLokasi.latitude,
feedbackLokasi.longitude)
                )

                for (item in
listMarker) {

createMarker(item.latitude,
item.longitude, item.namaLokasi)
                }


mMapsGogle.animateCamera(CameraUpdate
Factory.zoomTo(11f))


CoroutineScope(IO).launch {
                    delay(3000)

                    val
jarakFromDjikstra: Double =
DjikstraAlgorithm.calculateDistance(e
dtTitikAntar.text.toString(),
edtTitikJemput.text.toString())
                    val desimalFormat
= DecimalFormat("#.##")
                    Constans.distance
= jarakFromDjikstra

                    var
textviewLintasan = ""
                    val stringBuilder
= StringBuilder()
                    val getLintasan =
DjikstraAlgorithm.getJalanYangDilewat
i(edtTitikAntar.text.toString(),
edtTitikJemput.text.toString())
                    for (item in
getLintasan) {

stringBuilder.append(item).append("-
")
                    }
                    textviewLintasan
=
stringBuilder.toString().dropLast(1)

CoroutineScope(Dispatchers.Main).laun
ch {
                        jarak.text =
desimalFormat.format(jarakFromDjikstr
a).toString()
                        val setHarga
: Double = jarakFromDjikstra * 5000
```

```
                        harga.text =
setHarga.toInt().toString()

rute_terpendek.text =
textviewLintasan

pesan1.background =
getDrawable(R.drawable.form1)
                        pesan1.text =
"Pesan Sekarang"

pesan1.isEnabled = true

                        // event
listener jemput

edtTitikJemput.setOnClickListener {
                            val
myIntent =
Intent(this@Go3BikeActivity,
LokasiDjikstraActivity::class.java)

startActivityForResult(myIntent,
CODE_INTENT_SET_JEMPUT)
                        }

                        // event
listener antar

edtTitikAntar.setOnClickListener {
                            val
myIntent =
Intent(this@Go3BikeActivity,
LokasiDjikstraActivity::class.java)

startActivityForResult(myIntent,
CODE_INTENT_SET_TUJUAN)
                        }

                        // event
listener button

pesan1.setOnClickListener {

if(Constans.listMarker.isNotEmpty())
{

Constans.myDriver?.let { lokasiDriver
->

presenter.temukanDriverByKey(lokasiDr
iver, edtTitikJemput.text.toString(),
setHarga, textviewLintasan)

iosLoadingPopup.show()
                                }
                            } else {

Toast.makeText(this@Go3BikeActivity,
"Driver kami saat ini sedang sibuk",
Toast.LENGTH_SHORT).show()
                                }
                            }

                        }
                    }
                }
            } else if (resultCode ==
Activity.RESULT_OK && requestCode ==
CODE_INTENT_SET_JEMPUT) {

                val dataFeedback =
data?.extras?.getParcelable<CustomLok
asiDjikstra>(LokasiDjikstraActivity.T
AG_MAU_KEMANA)

                jarak.text = "0"
                harga.text = "0"
                rute_terpendek.text =
"Mencari Rute Terpendek"
                pesan1.background =
getDrawable(R.drawable.form3)
                pesan1.text = "Mohon
Tunggu"
                pesan1.isEnabled = false

                edtTitikJemput.text =
dataFeedback?.namaLokasi.toString()
                edtTitikAntar.text =
Constans.lokasiAntar?.namaLokasi

                // save di nilai
constanst jemput
                Constans.lokasiJemput =
CustomLokasiDjikstra(edtTitikJemput.t
ext.toString(),
dataFeedback!!.latitude,
dataFeedback.longitude)

                val listMarker =
ArrayList<CustomLokasiDjikstra>()

                mMapsGogle.clear()


listMarker.add(CustomLokasiDjikstra(d
ataFeedback?.namaLokasi.toString(),
dataFeedback?.latitude!!.toDouble(),
dataFeedback.longitude))

listMarker.add(CustomLokasiDjikstra(C
onstans.lokasiAntar?.namaLokasi.toStr
ing(),
Constans.lokasiAntar!!.latitude,
Constans.lokasiAntar!!.longitude))

                for (item in listMarker)
{
```

```
createMarker(item.latitude,
item.longitude, item.namaLokasi)
            }

            // Tambahin marker driver
            val markerOptions =
MarkerOptions().apply {

position(LatLng(Constans.latDriver,
Constans.lngDriver))
                flat(true)

icon(BitmapDescriptorFactory.fromReso
urce(R.drawable.motorcycle_mini))
            }

mMapsGogle.addMarker(markerOptions)

            CoroutineScope(IO).launch
{
                delay(3000)

                val
jarakFromDjikstra: Double =
DjikstraAlgorithm.calculateDistance(e
dtTitikAntar.text.toString(),
edtTitikJemput.text.toString())
                val desimalFormat =
DecimalFormat("#.##")

                var textviewLintasan
= ""
                val stringBuilder =
StringBuilder()
                val getLintasan =
DjikstraAlgorithm.getJalanYangDilewat
i(edtTitikAntar.text.toString(),
edtTitikJemput.text.toString())
                for (item in
getLintasan) {

stringBuilder.append(item).append("-
")
                }
                textviewLintasan =
stringBuilder.toString().dropLast(1)

CoroutineScope(Dispatchers.Main).Laun
ch {
                    jarak.text =
desimalFormat.format(jarakFromDjikstr
a).toString()
                    val setHarga :
Double = jarakFromDjikstra * 5000
                    harga.text =
setHarga.toInt().toString()

rute_terpendek.text =
textviewLintasan
                    pesan1.background
= getDrawable(R.drawable.form1)
                    pesan1.text =
"Pesan Sekarang"
                    pesan1.isEnabled
= true

                    // event listener
jemput

edtTitikJemput.setOnClickListener {
                        val myIntent
= Intent(this@Go3BikeActivity,
LokasiDjikstraActivity::class.java)

startActivityForResult(myIntent,
CODE_INTENT_SET_JEMPUT)
                    }

                    // event listener
antar

edtTitikAntar.setOnClickListener {
                        val myIntent
= Intent(this@Go3BikeActivity,
LokasiDjikstraActivity::class.java)

startActivityForResult(myIntent,
CODE_INTENT_SET_TUJUAN)
                    }

                    // event listener
button

pesan1.setOnClickListener {

if(Constans.listMarker.isNotEmpty())
{

Constans.myDriver?.let { lokasiDriver
->

presenter.temukanDriverByKey(lokasiDr
iver, edtTitikJemput.text.toString(),
setHarga, textviewLintasan)

iosLoadingPopup.show()
                            }
                        } else {

Toast.makeText(this@Go3BikeActivity,
"Driver kami saat ini sedang sibuk",
Toast.LENGTH_SHORT).show()
                        }
                    }
                }
            }
```

```
        } else if(resultCode ==
Activity.RESULT_OK && requestCode ==
CODE_INTENT_SET_TUJUAN) {
            val dataFeedback =
data?.extras?.getParcelable<CustomLok
asiDjikstra>(LokasiDjikstraActivity.T
AG_MAU_KEMANA)

            jarak.text = "0"
            harga.text = "0"
            rute_terpendek.text =
"Mencari Rute Terpendek"
            pesan1.background =
getDrawable(R.drawable.form3)
            pesan1.text = "Mohon
Tunggu"
            pesan1.isEnabled = false

            edtTitikJemput.text =
Constans.lokasiJemput?.namaLokasi
            edtTitikAntar.text =
dataFeedback?.namaLokasi

            // save di nilai
constanst jemput
            Constans.lokasiAntar =
CustomLokasiDjikstra(edtTitikAntar.te
xt.toString(),
dataFeedback!!.latitude,
dataFeedback.longitude)

            // set constants tujuan
pengantaran

Constans.destinatioLatNLng =
LatLng(dataFeedback.latitude,
dataFeedback.longitude)

            val listMarker =
ArrayList<CustomLokasiDjikstra>()

            mMapsGogle.clear()


listMarker.add(CustomLokasiDjikstra(d
ataFeedback?.namaLokasi.toString(),
dataFeedback?.latitude!!.toDouble(),
dataFeedback.longitude))

listMarker.add(CustomLokasiDjikstra(C
onstans.lokasiAntar?.namaLokasi.toStr
ing(),
Constans.lokasiAntar!!.latitude,
Constans.lokasiAntar!!.longitude))

            for (item in listMarker)
{

createMarker(item.latitude,
item.longitude, item.namaLokasi)
            }

            // Tambahin marker driver
            val markerOptions =
MarkerOptions().apply {

position(LatLng(Constans.latDriver,
Constans.lngDriver))
                flat(true)

icon(BitmapDescriptorFactory.fromReso
urce(R.drawable.motorcycle_mini))
            }

mMapsGogle.addMarker(markerOptions)

            CoroutineScope(IO).launch
{
                delay(3000)

                val
jarakFromDjikstra: Double =
DjikstraAlgorithm.calculateDistance(e
dtTitikAntar.text.toString(),
edtTitikJemput.text.toString())
                val desimalFormat =
DecimalFormat("#.##")
                var textviewLintasan
= ""
                val stringBuilder =
StringBuilder()
                val getLintasan =
DjikstraAlgorithm.getJalanYangDilewat
i(edtTitikAntar.text.toString(),
edtTitikJemput.text.toString())
                for (item in
getLintasan) {

stringBuilder.append(item).append("-
")
                }
                textviewLintasan =
stringBuilder.toString().dropLast(1)

CoroutineScope(Dispatchers.Main).laun
ch {
                    jarak.text =
desimalFormat.format(jarakFromDjikstr
a).toString()
                    val setHarga :
Double = jarakFromDjikstra * 5000
                    harga.text =
setHarga.toInt().toString()

rute_terpendek.text =
textviewLintasan
                    pesan1.background
= getDrawable(R.drawable.form1)
```

```
                    pesan1.text =
"Pesan Sekarang"
                    pesan1.isEnabled
= true

                    // event listener
jemput

edtTitikJemput.setOnClickListener {
                        val myIntent
= Intent(this@Go3BikeActivity,
LokasiDjikstraActivity::class.java)

startActivityForResult(myIntent,
CODE_INTENT_SET_JEMPUT)
                    }

                    // event listener
antar

edtTitikAntar.setOnClickListener {
                        val myIntent
= Intent(this@Go3BikeActivity,
LokasiDjikstraActivity::class.java)

startActivityForResult(myIntent,
CODE_INTENT_SET_TUJUAN)
                    }

                    // event listener
button

pesan1.setOnClickListener {

if(Constans.listMarker.isNotEmpty())
{

Constans.myDriver?.let { lokasiDriver
->

presenter.temukanDriverByKey(lokasiDr
iver, edtTitikJemput.text.toString(),
setHarga, textviewLintasan)

iosLoadingPopup.show()
                            }
                        } else {

Toast.makeText(this@Go3BikeActivity,
"Driver kami saat ini sedang sibuk",
Toast.LENGTH_SHORT).show()
                        }
                    }
                }
            }
        }
    }
```

```
@SuppressLint("MissingPermission")
    override fun
onMapReady(pGoogleMap: GoogleMap?) {

        if (pGoogleMap != null) {
            mMapsGogle = pGoogleMap
            mMapView.onResume()

mMapsGogle.uiSettings.isZoomControlsE
nabled = true

mMapsGogle.isMyLocationEnabled = true

            // Apabila ini dari
histori fragment
            if
(intent?.extras?.getParcelable<Pesana
n>(ProsesFragment.DETAIL_PESANAN) !=
null) {
                val detailPesanan =
intent?.extras?.getParcelable<Pesanan
>(ProsesFragment.DETAIL_PESANAN)

layourMauKemana.visibility =
View.GONE

layoutDataDiDapatkan.visibility =
View.VISIBLE

                driver_name.text =
detailPesanan?.nama_driver ?: "null"
                jemput_transaksi.text
= detailPesanan?.lokasi_jemput
                tujuan_transaksi.text
= detailPesanan?.lokasi_tujuan
                jarakOrder2.text =
detailPesanan?.jarak
                hargaOrder2.text =
detailPesanan?.harga_ongkos
                rute_perjalanan.text
= detailPesanan?.lintasan

                // fitur intent nomor
telepon

if(detailPesanan?.no_telepon_penumpan
g.isNullOrEmpty()){

btnTelefon.setOnClickListener {

Toast.makeText(this, "Nomor telepon
tidak ditemukan",
Toast.LENGTH_SHORT).show()
                    }
                } else {

btnTelefon.setOnClickListener {
                        val myIntent
= Intent(Intent.ACTION_CALL,
```

```
Uri.parse("tel:" +
detailPesanan?.no_telepon_penumpang))

startActivity(myIntent)
                    }
                }


jenis_kendaraan_hehe.text =
detailPesanan?.merk_kendaraan
                plat_hehe.text =
detailPesanan?.plat_kendaraan

                // buat marker maps
                val listMarker =
ArrayList<CustomLokasiDjikstra>()
                listMarker.add(

CustomLokasiDjikstra(detailPesanan!!.
lokasi_jemput,
detailPesanan.latitude_lokasi_jemput,
detailPesanan.longitude_lokasi_jemput
)
                )
                listMarker.add(

CustomLokasiDjikstra(detailPesanan!!.
lokasi_tujuan,
detailPesanan.latitude_lokasi_antar,
detailPesanan.longitude_lokasi_antar)
                )

                for (item in
listMarker) {

createMarker(item.latitude,
item.longitude, item.namaLokasi)
                }


mMapsGogle.animateCamera(CameraUpdate
Factory.zoomTo(11f))

            }

            if (Constans.myLatLng ==
null) {

fusedLocationProvider.lastLocation.ad
dOnSuccessListener { location ->
                    if (location !=
null) {

Constans.myLatLng =
LatLng(location.latitude,
location.longitude)

mMapsGogle.animateCamera(CameraUpdate
Factory.newLatLngZoom(Constans.myLatL
ng, 17f))

                        // check
driver avalaible

presenter.getDriverAvalaible()

                    } else {

return@addOnSuccessListener
                    }
                }
            } else {
                // animate camera

mMapsGogle.animateCamera(CameraUpdate
Factory.newLatLngZoom(Constans.myLatL
ng, 17f))

                // check driver
avalaible

presenter.getDriverAvalaible()
                }
        }

    }

    override fun
markerReceived(lokasiDriver:
LokasiDriver) {
        // Jika key tidak ada pada
list hasmap
        val key = lokasiDriver.key as
String

        // set di constans
        Constans.latDriver =
lokasiDriver.geoLocation!!.latitude
        Constans.lngDriver =
lokasiDriver.geoLocation!!.longitude

        if
(!Constans.listMarker.containsKey(key
)) {

            // Tambahin dulu di
listMarker
            val markerOptions =
MarkerOptions().apply {

position(LatLng(lokasiDriver.geoLocat
ion?.latitude ?: 0.0,
lokasiDriver.geoLocation?.longitude
?: 0.0))
                flat(true)

icon(BitmapDescriptorFactory.fromReso
urce(R.drawable.motorcycle_mini))
```

```
            }

            Constans.listMarker[key]
= markerOptions
        }

        for (item in
Constans.listMarker) {
            val markerOption =
item.value

mMapsGogle.addMarker(markerOption)
        }


fusedLocationProvider.removeLocationU
pdates(locationCallback)
    }

    override fun
jarakEstimasi(distanceEstimation:
Double, destination: LatLng,
lintasan: String) {

    }

    override fun driverFinded(
        driverBio: DriverBio,
        lokasiTujuan: String,
        harga: Double,
        rute: String
    ) {
        // notif driver dengan FCM
        val dataNotifikasi =
NotifikasiModel(

driverBio.token.toString(),
            DataNotifikasi(
                "Ada Penumpang!",

driverBio.uid.toString(),

session.uidCustomer.toString(),

session.namaLengkap.toString(),

session.noTelepon.toString(),
                Constans.distance,
                harga,

Constans.lokasiAntar?.namaLokasi.toSt
ring(),

Constans.lokasiJemput?.latitude ?:
0.0,

Constans.lokasiJemput?.longitude ?:
0.0,
Constans.lokasiAntar?.latitude ?:
0.0,

Constans.lokasiAntar?.longitude ?:
0.0,
                rute,

driverBio.merkKendaraan.toString(),

driverBio.platKendaraan.toString(),
                Constans.token,

Constans.lokasiJemput?.namaLokasi.toS
tring()
            )
        )

        val myOrder = Pesanan(
            driverBio.uid.toString(),

session.namaLengkap.toString(),

driverBio.nama_lengkap.toString(),

Constans.lokasiJemput?.namaLokasi ?:
"null",

Constans.lokasiAntar?.namaLokasi ?:
"null",
            harga.toInt().toString(),

Constans.distance.toString(),
            "menngantar",

session.noTelepon.toString(),

session.uidCustomer.toString(),
            rute,

Constans.lokasiJemput!!.latitude,

Constans.lokasiJemput!!.longitude,

Constans.lokasiAntar!!.latitude,

Constans.lokasiAntar!!.longitude,

driverBio.merkKendaraan.toString(),

driverBio.platKendaraan.toString()
        )

        Constans.namaDriver =
driverBio.nama_lengkap.toString()

presenter.noticeDriver(dataNotifikasi
)

presenter.addToDatabase(myOrder)
```

```
    }

    override fun notifCallback(state:
Boolean, data: NotifikasiModel) {
        if (state == true) {
            iosLoadingPopup.hide()


layoutSiapOrder.visibility =
View.GONE

layoutDataDiDapatkan.visibility =
View.VISIBLE

            jarakOrder2.text =
data.data.estimasi_jarak.toString()
            hargaOrder2.text =
data.data.harga.toInt().toString()
            rute_perjalanan.text =
data.data.lintasan
            tujuan_transaksi.text =
Constans.lokasiAntar?.namaLokasi.toSt
ring()
            jemput_transaksi.text =
Constans.lokasiJemput?.namaLokasi.toS
tring()
            driver_name.text =
Constans.namaDriver
            jenis_kendaraan_hehe.text
= data.data.merk_kendaraan
            plat_hehe.text =
data.data.plat_kendaraan

        } else {
            iosLoadingPopup.hide()

            Toast.makeText(this,
"Terjadi kesalahan saat pemesanan",
Toast.LENGTH_SHORT).show()
        }
    }

    // function cek perizinan lokasi
    private fun
hasLocationPermission(): Boolean {
        return
EasyPermissions.hasPermissions(
            applicationContext,

Manifest.permission.ACCESS_FINE_LOCAT
ION,
Manifest.permission.ACCESS_COARSE_LOC
ATION
        )
    }

    // function minta izin akses
lokasi
    private fun requestPermission() {


EasyPermissions.requestPermissions(
            this,
            "Aplikasi ini butuh izin
lokasi",
            101,

Manifest.permission.ACCESS_FINE_LOCAT
ION,
Manifest.permission.ACCESS_COARSE_LOC
ATION)
    }

    // function add marker
    private fun createMarker(lat:
Double, lng: Double, snippet:
String): Marker {
        return mMapsGogle.addMarker(
            MarkerOptions().apply {
                position(LatLng(lat,
lng))

title(snippet.toString())
            }
        )
    }

    companion object {
        const val
CODE_INTENT_MAU_KEMANA = 1111
        const val
CODE_INTENT_SET_JEMPUT = 1112
        const val
CODE_INTENT_SET_TUJUAN = 1113
    }
}
```

LokasiDijkstraActivity.kt (*Customer*)

```
package
com.gotrijek.go3jekcustomer.beta

import android.app.Activity
import android.content.Intent
import
androidx.appcompat.app.AppCompatActiv
ity
import android.os.Bundle
import
androidx.recyclerview.widget.Recycler
View
import com.gotrijek.go3jekcustomer.R
import
com.gotrijek.go3jekcustomer.adapter.D
jikstraLokasiAdapter
import
com.gotrijek.go3jekcustomer.adapter.I
```

```
temLokasiCLidked
import
com.gotrijek.go3jekcustomer.helper.Co
nstans
import
com.gotrijek.go3jekcustomer.models.Cu
stomLokasiDjikstra

class LokasiDjikstraActivity :
AppCompatActivity(),
ItemLokasiCLidked {

    private lateinit var myAdapter:
DjikstraLokasiAdapter
    private lateinit var rvLokasi:
RecyclerView

    override fun
onCreate(savedInstanceState: Bundle?)
{

super.onCreate(savedInstanceState)

setContentView(R.layout.activity_loka
si_djikstra)

        // layouting
        rvLokasi =
findViewById(R.id.lokasi_rv)

        myAdapter =
DjikstraLokasiAdapter(this)

        val listLokasi =
Constans.generateLocation()

myAdapter.setListLokasi(listLokasi)

rvLokasi.setHasFixedSize(true)
        rvLokasi.adapter = myAdapter

    }

    override fun
onItemClicked(lokasi:
CustomLokasiDjikstra) {
        val myIntent = Intent()

myIntent.putExtra(TAG_MAU_KEMANA,
lokasi)
        setResult(Activity.RESULT_OK,
myIntent)
        finish()
    }

    companion object {
        const val TAG_MAU_KEMANA =
"MAU_KEMANA_TAG"
    }
}
```

DijkstraShortestPath.kt (*Customer*)

```
import java.util.*


class DijkstraShortestPath {

    fun
computeShortestPaths(sourceMyVertex:
MyVertex) {
        sourceMyVertex.distance = 0.0
        val priorityQueue =
PriorityQueue<MyVertex>()

priorityQueue.add(sourceMyVertex)
        sourceMyVertex.isVisited =
true


while(!priorityQueue.isEmpty()) {

            val actualVertex =
priorityQueue.poll()

            for (edge in
actualVertex.getAdjacenciesList()) {
                val v: MyVertex =
edge.targetMyVertex as MyVertex
                if (!v.isVisited) {
                    val newDistance:
Double = actualVertex.distance +
edge.weight
                    if (newDistance <
v.distance) {

priorityQueue.remove(v)
                        v.distance =
newDistance
                        v.predecessor
= actualVertex

priorityQueue.add(v)
                    }
                }
            }
            actualVertex.isVisited =
true
        }
    }


    fun
getShortestPathTo(targetMyVertex:
```

```
MyVertex?): List<MyVertex> {
        val path:
MutableList<MyVertex> = ArrayList()
        var vertex = targetMyVertex
        while (vertex != null) {
            path.add(vertex)
            vertex =
vertex.predecessor
        }
        path.reverse()
        return path
    }
}
```

DijkstraAlgorithm.kt (*Customer*)

```
package
com.gotrijek.go3jekcustomer.helper

import DijkstraShortestPath
import Edge
import MyVertex
import android.util.Log
import
com.google.android.gms.maps.model.Lat
Lng

object DjikstraAlgorithm {

    fun
calculateDistance(lokasiTujuan:
String, lokasiJemput: String) :
Double {

        var distance = 0.0

        // Lokasi tujuan UNIMED
dengan lokasi jemput random
        if (lokasiTujuan == "UNIMED"
&& lokasiJemput == "Jalan William
Iskandar") {
            val vertexA =
MyVertex("Unimed")
            val vertexB =
MyVertex("Jalan Willem Iskandar")
            val vertexC =
MyVertex("Jalan Perjuangan")
            val vertexD =
MyVertex("Jalan Pelita 1")
            val vertexE =
MyVertex("Jalan HM. Said")
            val vertexF =
MyVertex("UINSU Kampus Pascasarjana")
            val vertexG =
MyVertex("Jalan Gurilla")
            val vertexH =
MyVertex("Jalan Rakyat")
            val vertexI =
MyVertex("Jalan Sentosa Baru")
            val vertexJ =
MyVertex("Jalan Prof. M. Yamin")
            val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))
```

```
vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

            val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texB)
            distance =
vertexA.distance

        } else if (lokasiTujuan ==
"UNIMED" && lokasiJemput == "UINSU
Kampus Pascasarjana") {
            val vertexA =
MyVertex("Unimed")
            val vertexB =
MyVertex("Jalan Willem Iskandar")
            val vertexC =
MyVertex("Jalan Perjuangan")
            val vertexD =
MyVertex("Jalan Pelita 1")
            val vertexE =
MyVertex("Jalan HM. Said")
            val vertexF =
MyVertex("UINSU Kampus Pascasarjana")
            val vertexG =
MyVertex("Jalan Gurilla")
            val vertexH =
MyVertex("Jalan Rakyat")
            val vertexI =
MyVertex("Jalan Sentosa Baru")
            val vertexJ =
MyVertex("Jalan Prof. M. Yamin")
            val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))
```

```
            val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texF)
            distance =
vertexA.distance

        } else if (lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
Perjuangan") {
            val vertexA =
MyVertex("Unimed")
            val vertexB =
MyVertex("Jalan Willem Iskandar")
            val vertexC =
MyVertex("Jalan Perjuangan")
            val vertexD =
MyVertex("Jalan Pelita 1")
            val vertexE =
MyVertex("Jalan HM. Said")
            val vertexF =
MyVertex("UINSU Kampus Pascasarjana")
            val vertexG =
MyVertex("Jalan Gurilla")
            val vertexH =
MyVertex("Jalan Rakyat")
            val vertexI =
MyVertex("Jalan Sentosa Baru")
            val vertexJ =
MyVertex("Jalan Prof. M. Yamin")
            val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))
```

```
vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

            val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texC)
            distance =
vertexA.distance

        } else if (lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
Pelita I") {
            val vertexA =
MyVertex("Unimed")
            val vertexB =
MyVertex("Jalan Willem Iskandar")
            val vertexC =
```

```
MyVertex("Jalan Perjuangan")
            val vertexD =
MyVertex("Jalan Pelita 1")
            val vertexE =
MyVertex("Jalan HM. Said")
            val vertexF =
MyVertex("UINSU Kampus Pascasarjana")
            val vertexG =
MyVertex("Jalan Gurilla")
            val vertexH =
MyVertex("Jalan Rakyat")
            val vertexI =
MyVertex("Jalan Sentosa Baru")
            val vertexJ =
MyVertex("Jalan Prof. M. Yamin")
            val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))
```

```
vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

            val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texD)
            distance =
vertexA.distance

        } else if (lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
H.M Said") {
            val vertexA =
MyVertex("Unimed")
            val vertexB =
MyVertex("Jalan Willem Iskandar")
            val vertexC =
MyVertex("Jalan Perjuangan")
            val vertexD =
MyVertex("Jalan Pelita 1")
            val vertexE =
MyVertex("Jalan HM. Said")
            val vertexF =
MyVertex("UINSU Kampus Pascasarjana")
            val vertexG =
MyVertex("Jalan Gurilla")
            val vertexH =
MyVertex("Jalan Rakyat")
            val vertexI =
MyVertex("Jalan Sentosa Baru")
            val vertexJ =
MyVertex("Jalan Prof. M. Yamin")
            val vertexK =
```

```
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

            val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texE)
            distance =
vertexA.distance

        } else if (lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
Gurila") {
            val vertexA =
MyVertex("Unimed")
            val vertexB =
MyVertex("Jalan Willem Iskandar")
            val vertexC =
MyVertex("Jalan Perjuangan")
            val vertexD =
MyVertex("Jalan Pelita 1")
            val vertexE =
MyVertex("Jalan HM. Said")
            val vertexF =
MyVertex("UINSU Kampus Pascasarjana")
            val vertexG =
MyVertex("Jalan Gurilla")
            val vertexH =
MyVertex("Jalan Rakyat")
            val vertexI =
MyVertex("Jalan Sentosa Baru")
            val vertexJ =
MyVertex("Jalan Prof. M. Yamin")
            val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))
```

```
vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

            val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texG)
```

```
            distance =
vertexA.distance

        } else if (lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
Rakyat"){
            val vertexA =
MyVertex("Unimed")
            val vertexB =
MyVertex("Jalan Willem Iskandar")
            val vertexC =
MyVertex("Jalan Perjuangan")
            val vertexD =
MyVertex("Jalan Pelita 1")
            val vertexE =
MyVertex("Jalan HM. Said")
            val vertexF =
MyVertex("UINSU Kampus Pascasarjana")
            val vertexG =
MyVertex("Jalan Gurilla")
            val vertexH =
MyVertex("Jalan Rakyat")
            val vertexI =
MyVertex("Jalan Sentosa Baru")
            val vertexJ =
MyVertex("Jalan Prof. M. Yamin")
            val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
```

```
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

            val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texH)
            distance =
vertexA.distance
        } else if (lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
Sentosa Baru") {
            val vertexA =
MyVertex("Unimed")
            val vertexB =
MyVertex("Jalan Willem Iskandar")
            val vertexC =
MyVertex("Jalan Perjuangan")
            val vertexD =
MyVertex("Jalan Pelita 1")
            val vertexE =
MyVertex("Jalan HM. Said")
            val vertexF =
MyVertex("UINSU Kampus Pascasarjana")
            val vertexG =
MyVertex("Jalan Gurilla")
            val vertexH =
MyVertex("Jalan Rakyat")
            val vertexI =
MyVertex("Jalan Sentosa Baru")
            val vertexJ =
MyVertex("Jalan Prof. M. Yamin")
            val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))
```

```
vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

            val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texI)
            distance =
vertexA.distance
        } else if (lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
Prof.H.M Yamin") {
            val vertexA =
MyVertex("Unimed")
            val vertexB =
MyVertex("Jalan Willem Iskandar")
            val vertexC =
MyVertex("Jalan Perjuangan")
            val vertexD =
MyVertex("Jalan Pelita 1")
            val vertexE =
MyVertex("Jalan HM. Said")
            val vertexF =
MyVertex("UINSU Kampus Pascasarjana")
            val vertexG =
MyVertex("Jalan Gurilla")
            val vertexH =
MyVertex("Jalan Rakyat")
            val vertexI =
MyVertex("Jalan Sentosa Baru")
            val vertexJ =
MyVertex("Jalan Prof. M. Yamin")
            val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))
```

```
vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))
```

```
vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

            val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texJ)
            distance =
vertexA.distance
        } else if (lokasiTujuan ==
"UNIMED" && lokasiJemput ==
"Universitas HKBP Nomensen") {
            val vertexA =
MyVertex("Unimed")
            val vertexB =
MyVertex("Jalan Willem Iskandar")
            val vertexC =
MyVertex("Jalan Perjuangan")
            val vertexD =
MyVertex("Jalan Pelita 1")
            val vertexE =
MyVertex("Jalan HM. Said")
            val vertexF =
MyVertex("UINSU Kampus Pascasarjana")
            val vertexG =
MyVertex("Jalan Gurilla")
            val vertexH =
MyVertex("Jalan Rakyat")
            val vertexI =
MyVertex("Jalan Sentosa Baru")
            val vertexJ =
MyVertex("Jalan Prof. M. Yamin")
            val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))
```

```
vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

            val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texK)
            distance =
vertexA.distance
        }

// ini adalah function guna
mendapatkan rute djikstra
fun
getJalanYangDilewati(lokasiTujuan:
```

```
String, lokasiJemput: String):
List<MyVertex> {

    val path: ArrayList<MyVertex> =
java.util.ArrayList()

    if (lokasiTujuan == "UNIMED" &&
lokasiJemput == "Jalan William
Iskandar") {
        val vertexA =
MyVertex("Unimed")
        val vertexB = MyVertex("Jalan
Willem Iskandar")
        val vertexC = MyVertex("Jalan
Perjuangan")
        val vertexD = MyVertex("Jalan
Pelita 1")
        val vertexE = MyVertex("Jalan
HM. Said")
        val vertexF = MyVertex("UINSU
Kampus Pascasarjana")
        val vertexG = MyVertex("Jalan
Gurilla")
        val vertexH = MyVertex("Jalan
Rakyat")
        val vertexI = MyVertex("Jalan
Sentosa Baru")
        val vertexJ = MyVertex("Jalan
Prof. M. Yamin")
        val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

        val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texB)
        val result =
shortestPath.getShortestPathTo(vertex
A)

        path.addAll(result)
    } else if (lokasiTujuan ==
"UNIMED" && lokasiJemput == "UINSU
Kampus Pascasarjana") {
        val vertexA =
MyVertex("Unimed")
        val vertexB = MyVertex("Jalan
Willem Iskandar")
        val vertexC = MyVertex("Jalan
```

```
Perjuangan")
        val vertexD = MyVertex("Jalan
Pelita 1")
        val vertexE = MyVertex("Jalan
HM. Said")
        val vertexF = MyVertex("UINSU
Kampus Pascasarjana")
        val vertexG = MyVertex("Jalan
Gurilla")
        val vertexH = MyVertex("Jalan
Rakyat")
        val vertexI = MyVertex("Jalan
Sentosa Baru")
        val vertexJ = MyVertex("Jalan
Prof. M. Yamin")
        val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))
```

```
vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

        val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texF)
        val result =
shortestPath.getShortestPathTo(vertex
A)

        path.addAll(result)
    } else if (lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
Perjuangan") {
        val vertexA =
MyVertex("Unimed")
        val vertexB = MyVertex("Jalan
Willem Iskandar")
        val vertexC = MyVertex("Jalan
Perjuangan")
        val vertexD = MyVertex("Jalan
Pelita 1")
        val vertexE = MyVertex("Jalan
HM. Said")
        val vertexF = MyVertex("UINSU
Kampus Pascasarjana")
        val vertexG = MyVertex("Jalan
Gurilla")
        val vertexH = MyVertex("Jalan
Rakyat")
        val vertexI = MyVertex("Jalan
Sentosa Baru")
        val vertexJ = MyVertex("Jalan
```

```
Prof. M. Yamin")
        val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))
```

```
vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

        val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texC)
        val result =
shortestPath.getShortestPathTo(vertex
A)

        path.addAll(result)

    } else if(lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
Pelita I") {
        val vertexA =
MyVertex("Unimed")
        val vertexB = MyVertex("Jalan
Willem Iskandar")
        val vertexC = MyVertex("Jalan
Perjuangan")
        val vertexD = MyVertex("Jalan
Pelita 1")
        val vertexE = MyVertex("Jalan
HM. Said")
        val vertexF = MyVertex("UINSU
Kampus Pascasarjana")
        val vertexG = MyVertex("Jalan
Gurilla")
        val vertexH = MyVertex("Jalan
Rakyat")
        val vertexI = MyVertex("Jalan
Sentosa Baru")
        val vertexJ = MyVertex("Jalan
Prof. M. Yamin")
        val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
```

```
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))
```

```
        val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texD)
        val result =
shortestPath.getShortestPathTo(vertex
A)

        path.addAll(result)

    } else if(lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
H.M Said") {
        val vertexA =
MyVertex("Unimed")
        val vertexB = MyVertex("Jalan
Willem Iskandar")
        val vertexC = MyVertex("Jalan
Perjuangan")
        val vertexD = MyVertex("Jalan
Pelita 1")
        val vertexE = MyVertex("Jalan
HM. Said")
        val vertexF = MyVertex("UINSU
Kampus Pascasarjana")
        val vertexG = MyVertex("Jalan
Gurilla")
        val vertexH = MyVertex("Jalan
Rakyat")
        val vertexI = MyVertex("Jalan
Sentosa Baru")
        val vertexJ = MyVertex("Jalan
Prof. M. Yamin")
        val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))
```

```
vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

        val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texE)
        val result =
shortestPath.getShortestPathTo(vertex
A)

        path.addAll(result)

    } else if(lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
Gurila"){
        val vertexA =
MyVertex("Unimed")
        val vertexB = MyVertex("Jalan
Willem Iskandar")
        val vertexC = MyVertex("Jalan
Perjuangan")
        val vertexD = MyVertex("Jalan
Pelita 1")
        val vertexE = MyVertex("Jalan
HM. Said")
        val vertexF = MyVertex("UINSU
Kampus Pascasarjana")
        val vertexG = MyVertex("Jalan
Gurilla")
        val vertexH = MyVertex("Jalan
Rakyat")
        val vertexI = MyVertex("Jalan
Sentosa Baru")
        val vertexJ = MyVertex("Jalan
Prof. M. Yamin")
        val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))
```

```
vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

        val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texG)
        val result =
shortestPath.getShortestPathTo(vertex
A)

        path.addAll(result)

    } else if(lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
Rakyat") {
        val vertexA =
MyVertex("Unimed")
        val vertexB = MyVertex("Jalan
Willem Iskandar")
        val vertexC = MyVertex("Jalan
Perjuangan")
        val vertexD = MyVertex("Jalan
Pelita 1")
        val vertexE = MyVertex("Jalan
HM. Said")
        val vertexF = MyVertex("UINSU
Kampus Pascasarjana")
        val vertexG = MyVertex("Jalan
Gurilla")
        val vertexH = MyVertex("Jalan
Rakyat")
        val vertexI = MyVertex("Jalan
Sentosa Baru")
        val vertexJ = MyVertex("Jalan
Prof. M. Yamin")
        val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
```

```
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

        val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texH)
        val result =
shortestPath.getShortestPathTo(vertex
A)

        path.addAll(result)
    } else if(lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
Sentosa Baru"){
        val vertexA =
MyVertex("Unimed")
        val vertexB = MyVertex("Jalan
Willem Iskandar")
        val vertexC = MyVertex("Jalan
Perjuangan")
        val vertexD = MyVertex("Jalan
Pelita 1")
        val vertexE = MyVertex("Jalan
HM. Said")
        val vertexF = MyVertex("UINSU
Kampus Pascasarjana")
        val vertexG = MyVertex("Jalan
Gurilla")
        val vertexH = MyVertex("Jalan
Rakyat")
        val vertexI = MyVertex("Jalan
Sentosa Baru")
        val vertexJ = MyVertex("Jalan
Prof. M. Yamin")
        val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
```

```
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))
```

```
vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

        val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texI)
        val result =
shortestPath.getShortestPathTo(vertex
A)

        path.addAll(result)
    } else if(lokasiTujuan ==
"UNIMED" && lokasiJemput == "Jalan
Prof.H.M Yamin"){
        val vertexA =
MyVertex("Unimed")
        val vertexB = MyVertex("Jalan
Willem Iskandar")
        val vertexC = MyVertex("Jalan
Perjuangan")
        val vertexD = MyVertex("Jalan
Pelita 1")
        val vertexE = MyVertex("Jalan
HM. Said")
        val vertexF = MyVertex("UINSU
Kampus Pascasarjana")
        val vertexG = MyVertex("Jalan
Gurilla")
        val vertexH = MyVertex("Jalan
Rakyat")
        val vertexI = MyVertex("Jalan
Sentosa Baru")
        val vertexJ = MyVertex("Jalan
Prof. M. Yamin")
        val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))
```

```
vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))

vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

        val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texJ)
        val result =
shortestPath.getShortestPathTo(vertex
A)
```

```
        path.addAll(result)
    } else if(lokasiTujuan ==
"UNIMED" && lokasiJemput ==
"Universitas HKBP Nomensen") {
        val vertexA =
MyVertex("Unimed")
        val vertexB = MyVertex("Jalan
Willem Iskandar")
        val vertexC = MyVertex("Jalan
Perjuangan")
        val vertexD = MyVertex("Jalan
Pelita 1")
        val vertexE = MyVertex("Jalan
HM. Said")
        val vertexF = MyVertex("UINSU
Kampus Pascasarjana")
        val vertexG = MyVertex("Jalan
Gurilla")
        val vertexH = MyVertex("Jalan
Rakyat")
        val vertexI = MyVertex("Jalan
Sentosa Baru")
        val vertexJ = MyVertex("Jalan
Prof. M. Yamin")
        val vertexK =
MyVertex("Universitas HKBP Nommensen
Medan")


vertexA.addNeighbour(Edge(1.0,
vertexA, vertexB))

vertexB.addNeighbour(Edge(1.0,
vertexB, vertexA))

vertexB.addNeighbour(Edge(1.1,
vertexB, vertexC))

vertexB.addNeighbour(Edge(2.0,
vertexB, vertexG))

vertexC.addNeighbour(Edge(1.1,
vertexC, vertexB))

vertexC.addNeighbour(Edge(0.95,
vertexC, vertexD))

vertexD.addNeighbour(Edge(0.95,
vertexD, vertexC))

vertexD.addNeighbour(Edge(0.55,
vertexD, vertexE))

vertexD.addNeighbour(Edge(0.35,
vertexD, vertexH))

vertexE.addNeighbour(Edge(0.55,
vertexE, vertexD))
```

```
vertexE.addNeighbour(Edge(0.35,
vertexE, vertexF))

vertexF.addNeighbour(Edge(0.35,
vertexF, vertexE))

vertexG.addNeighbour(Edge(2.0,
vertexG, vertexB))

vertexG.addNeighbour(Edge(1.3,
vertexG, vertexH))

vertexG.addNeighbour(Edge(0.45,
vertexG, vertexI))

vertexH.addNeighbour(Edge(0.35,
vertexH, vertexD))

vertexH.addNeighbour(Edge(1.3,
vertexH, vertexG))

vertexI.addNeighbour(Edge(0.45,
vertexI, vertexG))

vertexI.addNeighbour(Edge(1.4,
vertexI, vertexJ))

vertexJ.addNeighbour(Edge(1.4,
vertexJ, vertexI))

vertexJ.addNeighbour(Edge(1.0,
vertexJ, vertexK))

vertexK.addNeighbour(Edge(0.35,
vertexK, vertexF))

        val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texK)
        val result =
shortestPath.getShortestPathTo(vertex
A)

        path.addAll(result)
    }
    val shortestPath =
DijkstraShortestPath()

shortestPath.computeShortestPaths(ver
texJ)
            val result =
shortestPath.getShortestPathTo(vertex
K)

            path.addAll(result)
        }
```

```
        return path
    }
}
```

LoginActivity.kt (*Driver*)

```
package
com.gotrijek.go3jekdriver.beta

import android.content.Intent
import
androidx.appcompat.app.AppCompatActiv
ity
import android.os.Bundle
import android.view.View
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import
com.gotrijek.go3jekdriver.MainActivit
y
import com.gotrijek.go3jekdriver.R
import
com.gotrijek.go3jekdriver.helper.Cons
tants
import
com.gotrijek.go3jekdriver.helper.Sess
ionManager
import
com.gotrijek.go3jekdriver.models.Driv
erInfo
import
com.gotrijek.go3jekdriver.presenter.l
ogin.LoginPresenter
import
com.gotrijek.go3jekdriver.view.login.
LoginViewInterface
import
kotlinx.android.synthetic.main.activi
ty_login2.*

class Login2Activity :
AppCompatActivity(),
LoginViewInterface {

    private lateinit var daftarAkun:
TextView
    private lateinit var emailEdt:
EditText
    private lateinit var passwordEdt:
EditText
    private lateinit var
sessionManager: SessionManager

    override fun
onCreate(savedInstanceState: Bundle?)
{

super.onCreate(savedInstanceState)

setContentView(R.layout.activity_logi
n2)

        // binding layout
        daftarAkun =
findViewById(R.id.registration)
        emailEdt =
findViewById(R.id.email)
        passwordEdt =
findViewById(R.id.password)

        // presenter
        val presenter =
LoginPresenter(this)

        // session manager
        sessionManager =
SessionManager(this)

        // event listener daftar akun
        daftarAkun.setOnClickListener
{
            val myIntent =
Intent(this,
DaftarActivity::class.java)
            startActivity(myIntent)
        }

        // akun listener btn daftar
        login.setOnClickListener {
            if
(emailEdt.text.isNotEmpty() ||
passwordEdt.text.isNotEmpty()) {
                val email =
emailEdt.text.toString()
                val password =
passwordEdt.text.toString()


presenter.loginAuthentication(email,
password)
            } else {

Toast.makeText(this@Login2Activity,
"Harap isi seluruh data terlebih
dahulu", Toast.LENGTH_SHORT).show()
            }
        }
    }

    override fun isLoading(status:
Boolean?) {
        if (status == true) {
            progress.visibility =
View.VISIBLE
```

```
            login.visibility =
View.GONE
        } else {
            progress.visibility =
View.GONE
            login.visibility =
View.VISIBLE
        }
    }

    override fun isMessage(message:
String) {
        Toast.makeText(this, message,
Toast.LENGTH_SHORT).show()
    }

    override fun
isSuccessLogin(stats: Boolean,
driver: DriverInfo?) {
        if (stats == true) {
            val intent = Intent(this,
MainActivity::class.java)

intent.putExtra(Constants.EXTRA_DRIVE
R_DATA, driver)
            sessionManager.login =
true

sessionManager.namaLengkap =
driver?.nama_lengkap

sessionManager.nomorTelepon =
driver?.telepon
            sessionManager.uidDriver
= driver?.uid

sessionManager.merkKendaraan =
driver?.merkKendaraan

sessionManager.platKendaraan =
driver?.platKendaraan
            startActivity(intent)
            finish()
        }
    }
}
```

Daftar.Activity.kt (*Driver*)

```
package
com.gotrijek.go3jekdriver.beta

import android.content.Intent
import
androidx.appcompat.app.AppCompatActiv
ity
import android.os.Bundle
import android.widget.*
import
com.gmail.samehadar.iosdialog.IOSDial
og
import
com.gotrijek.go3jekdriver.MainActivit
y
import com.gotrijek.go3jekdriver.R
import
com.gotrijek.go3jekdriver.helper.Sess
ionManager
import
com.gotrijek.go3jekdriver.models.Driv
erInfo
import
com.gotrijek.go3jekdriver.presenter.b
eta.DaftarInterface
import
com.gotrijek.go3jekdriver.presenter.b
eta.DaftarPresenter

class DaftarActivity :
AppCompatActivity(), DaftarInterface
{

    private lateinit var namaLengkap:
EditText
    private lateinit var
jenisKelamin: RadioGroup
    private lateinit var tempatLahir:
EditText
    private lateinit var
tanggalLahir: EditText
    private lateinit var alamat:
EditText
    private lateinit var noKTP:
EditText
    private lateinit var email:
EditText
    private lateinit var password:
EditText
    private lateinit var noTelp:
EditText
    private lateinit var kota:
EditText
    private lateinit var
merkKendaraan: EditText
    private lateinit var
platKendaraan: EditText
    private lateinit var daftarBtn:
Button
    private var jenisKelaminPilihan:
String = "Pria"

    private lateinit var
sessionManager: SessionManager

    private val iosLoadingPopup by
lazy {
```

```
        IOSDialog.Builder(this)

.setTitleColorRes(R.color.gray)
            .setTitle("Mohon Tunggu")
            .setCancelable(false)
            .build()
    }

    override fun
onCreate(savedInstanceState: Bundle?)
{

super.onCreate(savedInstanceState)

setContentView(R.layout.activity_daft
ar)

        // layouting name binding
        namaLengkap =
findViewById(R.id.nama)
        jenisKelamin =
findViewById(R.id.jenis_kelamin)
        tempatLahir =
findViewById(R.id.tempat_lahir)
        tanggalLahir =
findViewById(R.id.tanggal_lahir)
        alamat =
findViewById(R.id.alamat)
        noKTP =
findViewById(R.id.no_ktp)
        email =
findViewById(R.id.email)
        password =
findViewById(R.id.password)
        noTelp =
findViewById(R.id.phone)
        kota =
findViewById(R.id.kota)
        merkKendaraan =
findViewById(R.id.merk_kendaraan)
        platKendaraan =
findViewById(R.id.plat_kendaraan)
        daftarBtn =
findViewById(R.id.btn_lanjut)

        // get nilai dari masing2
widget

jenisKelamin.setOnCheckedChangeListen
er(object :
RadioGroup.OnCheckedChangeListener {
            override fun
onCheckedChanged(group: RadioGroup?,
checkId: Int) {
                val choose:
RadioButton = findViewById(checkId)
                jenisKelaminPilihan =
choose.text.toString()
            }
        })

        // presenter
        val presenter =
DaftarPresenter(this)

        // session manager
        sessionManager =
SessionManager(this)

        daftarBtn.setOnClickListener
{
            val dataDriver =
DriverInfo(

namaLengkap.text.toString(),
                jenisKelaminPilihan,

tempatLahir.text.toString(),

tanggalLahir.text.toString(),

alamat.text.toString(),

noKTP.text.toString(),

email.text.toString(),

noTelp.text.toString(),

noKTP.text.toString(),
                "",

merkKendaraan.text.toString(),

platKendaraan.text.toString(),

password.text.toString()
            )

presenter.registerToAuth(dataDriver)
        }
    }

    override fun
onLoginSuccess(status: Boolean,
driverInfo: DriverInfo) {
        if (status == true) {
            val myIntent =
Intent(this,
MainActivity::class.java)

            // set session manager
            sessionManager.login =
true

sessionManager.namaLengkap =
driverInfo.nama_lengkap
```

```
sessionManager.nomorTelepon =
driverInfo.telepon
            sessionManager.uidDriver
= driverInfo.uid

            startActivity(myIntent)
            finish()
        } else {
            Toast.makeText(this, "",
Toast.LENGTH_SHORT).show()
        }
    }

    override fun isLoading(state:
Boolean) {
        if (state == true) {
            iosLoadingPopup.show()
        } else {
            iosLoadingPopup.hide()
        }
    }


}
```

DetailPesananPresenter.kt (*Driver*)

```
package
com.gotrijek.go3jekdriver.presenter.p
esanan

import
com.google.firebase.database.Firebase
Database
import
com.gotrijek.go3jekdriver.helper.Cons
tants
import
com.gotrijek.go3jekdriver.models.Pesa
nan
import
com.gotrijek.go3jekdriver.view.pesana
n.detail.DetailPesananView

class DetailPesananPresenter(private
val detailPesananView:
DetailPesananView) {

    private val mDatabase by lazy {

FirebaseDatabase.getInstance()
    }

    fun addProsesPesanan(pesanan:
Pesanan) {
        val prosesPickupReference =
mDatabase.getReference(Constants.tb_p
roses_pickup)
        prosesPickupReference

.child(pesanan.uid_driver)
            .setValue(pesanan)
            .addOnCompleteListener {
task ->
                if
(task.isSuccessful) {

detailPesananView.isSucessDetailPesan
an(true)
                } else {

detailPesananView.isSucessDetailPesan
an(false)
                }
            }
            .addOnFailureListener {

detailPesananView.failureProses(it)
            }
    }

    fun
updateStatusPenjemputan(uidDriver:
String) {

        val valueUpdates =
HashMap<String, Any>()

valueUpdates["status_penjemputan"] =
"mengantar"


mDatabase.getReference(Constants.tb_p
roses_pickup)
            .child(uidDriver)

.updateChildren(valueUpdates)
            .addOnCompleteListener
{task ->
                if
(task.isSuccessful) {

detailPesananView.updateStatusToMenga
ntar(true)
                } else {

detailPesananView.updateStatusToMenga
ntar(false)
                }
            }
    }

    fun
addSelesaiPenjemputan(pesanan:
Pesanan) {
```

```
        val key =
mDatabase.reference.push().key.toStri
ng()
        val selesaiReference =
mDatabase.getReference(Constants.tb_h
istori_aktifitas)

        val pesanan = Pesanan(
            pesanan.uid_driver,
            pesanan.nama_customer,
            pesanan.nama_driver,
            pesanan.lokasi_jemput,
            pesanan.lokasi_tujuan,
            pesanan.harga_ongkos,
            pesanan.jarak,
            "selesai",

pesanan.no_telepon_penumpang,
            pesanan.uid_customer,
            pesanan.lintasan
        )

        selesaiReference.child(key)
            .setValue(pesanan)
            .addOnCompleteListener {
task ->
                if
(task.isSuccessful) {

mDatabase.getReference(Constants.tb_p
roses_pickup)

.removeValue()

.addOnCompleteListener { task ->
                            if
(task.isSuccessful) {

detailPesananView.makeSelesaiChildren
Sucess(true)
                            } else {

detailPesananView.failedResponse("Gag
al hapus children pickup location")
                            }
                        }
                }
            }

    }

}
```

ProsesPesananPresenter.kt (*Driver*)

```
package
com.gotrijek.go3jekdriver.presenter.p
esanan

import android.util.Log
import
com.google.firebase.database.DataSnap
shot
import
com.google.firebase.database.Database
Error
import
com.google.firebase.database.Firebase
Database
import
com.google.firebase.database.ValueEve
ntListener
import
com.gotrijek.go3jekdriver.helper.Cons
tants
import
com.gotrijek.go3jekdriver.models.Pesa
nan
import
com.gotrijek.go3jekdriver.view.pesana
n.proses.ProsesPesananView

class ProsesPesananPresenter(val
prosesPesananView: ProsesPesananView)
{

    private val mDatabase by lazy {

FirebaseDatabase.getInstance()
    }

    fun
getPenumpangYangDiproses(uidDriver:
String) {

mDatabase.getReference(Constants.tb_p
roses_pickup)
            .child(uidDriver)

.addListenerForSingleValueEvent(objec
t : ValueEventListener {
                override fun
onCancelled(error: DatabaseError) {

prosesPesananView.onCanceled(error.me
ssage)
                }

                override fun
onDataChange(snapshot: DataSnapshot)
{
                    var
pesananEntitas: Pesanan? = null
                    if
(snapshot.exists()) {
                        val
```

```
listPesanan = ArrayList<Pesanan>()

snapshot.children.map {

pesananEntitas =
snapshot.getValue(Pesanan::class.java
)
                        }

listPesanan.add(pesananEntitas!!)

prosesPesananView.getDataSuccess(list
Pesanan)
                    } else {

prosesPesananView.snapshotExist("Tida
k ada proses saat ini")
                    }
                }

            })
    }

}
```

SelesaiPesananPresenter.kt (*Driver*)

```
package
com.gotrijek.go3jekdriver.presenter.p
esanan

import android.util.Log
import
com.google.firebase.database.DataSnap
shot
import
com.google.firebase.database.Database
Error
import
com.google.firebase.database.Firebase
Database
import
com.google.firebase.database.ValueEve
ntListener
import
com.gotrijek.go3jekdriver.helper.Cons
tants
import
com.gotrijek.go3jekdriver.models.Pesa
nan
import
com.gotrijek.go3jekdriver.view.pesana
n.selesai.SelesaiView

class SelesaiPesananPresenter(private
val selesaiView: SelesaiView) {
    private val mDatabase by lazy {

FirebaseDatabase.getInstance()
    }

    fun
getAllYangSudahSelesai(uidDriver:
String) {

mDatabase.getReference(Constants.tb_h
istori_aktifitas)

.orderByChild("uid_driver")
            .equalTo(uidDriver)

.addListenerForSingleValueEvent(objec
t : ValueEventListener {
                override fun
onCancelled(error: DatabaseError) {

selesaiView.failureSnapshot(error.mes
sage)
                }

                override fun
onDataChange(snapshot: DataSnapshot)
{
                    if
(snapshot.exists()) {
                        val
listPesanan = ArrayList<Pesanan>()
                        for (item in
snapshot.children) {
                            val
pesanan =
item.getValue(Pesanan::class.java)

listPesanan.add(pesanan!!)
                        }

selesaiView.successSnapshot(listPesan
an)
                    } else {

selesaiView.failedSnapshot("Snapshot
Tidak Ada")
                    }
                }
        })
    }


}
```