BAB 1 PENDAHULUAN

1.1 Latar Belakang

Restoran merupakan bangunan yang digunakan secara komersil yang melayani konsumen baik makanan dan/atau minuman (Marsum, 2005). Bagi orang yang sibuk dengan pekerjaan dan tidak sempat untuk memasak, rumah makan atau restoran dapat menjadi solusinya. Usaha restoran/rumah makan berskala menengah dan besar memiliki rata-rata tamu per harinya sebanyak 227 orang dan tempat duduk yang tersedia sebanyak 131 tempat duduk per usaha pada tahun 2015. Dilihat dari lokasi usaha, sebagian besar usaha restoran/rumah makan bertempat di kawasan pertokoan atau perkantoran, yaitu sebesar 54,57 persen. Sedangkan di lokasi objek wisata hanya sebesar 15,71 persen (BPS, 2017)

Menurut data yang didapatkan dari kuesioner pada **Error! Reference source not found.** yang dibagikan kepada 265 orang, 176 orang menjawab bahwa mengantre tetap menjadi permasalahan yang mereka alami pada saat berada di restoran dan 80 orang menjawab pelayanan yang ada kurang baik. Dari kuesioner tersebut dan wawancara yang telah dilakukan pada restoran juga didapatkan bahwa restoran di Malang masih memberikan daftar menu secara manual kepada pelanggannya yang membuat pihak restoran harus mencetak baru menunya setiap kali terdapat perubahan daftar menu restoran. Selain itu, pemesanan menu restoran secara manual masih menggunakan kertas, sehingga pelanggan harus menuliskan menu yang ingin dipesan melalui kertas yang diberikan oleh pelayan restoran. Hal ini akan membuang-buang waktu dan tenaga para pelanggan apalagi bagi mereka yang hanya memiliki waktu istirahat yang sedikit. Dengan keterbatasan karyawan restoran dan jumlah pelanggan yang sangat ramai membuat pelayanan di restoran tersebut sangat lama. Dengan data tersebut dapat diambil kesimpulan bahwa usaha restoran memang selalu ramai akan pelanggan

Berdasarkan permasalahan yang telah dipaparkan, penulis berusaha membuat aplikasi manajemen antrean pesanan menu restoran yang diharapkan dapat mengotomatiskan sistem antrean di restoran dengan membuat sistem terpusat yang dibantu dengan teknologi yang berkembang pada era saat ini. Salah satu teknologi yang dapat menjadi solusi adalah *Progressive Website Application (PWA)*. PWA menggunakan kapabilitas *modern website* untuk membawa suatu website menjadi *app-like* yang membuat pengalaman pengguna lebih menyerupai aplikasi *smartphone*, sehingga akan membuat aplikasi *mobile web* menjadi lebih cepat, dapat diandalkan, dan *engaging*. (Karpagam, 2017)

Pengembangan aplikasi ini dalam hal autentikasi restoran akan lebih mudah dengan menggunakan Kode QR. *Quick Response Code* atau yang biasa disebut sebagai Kode QR adalah gambar digital dua dimensi dimana dapat dengan mudah dibaca oleh kamera pada perangkat *mobile* manapun. Sekarang ini Kode QR sangat populer karena perkembangan *mobile*. Penggunaan *mobile device* mencapai 15.6% pada tahun 2001 hingga 74.9% (2010) (Cata, T., Patel, P. S., & Sakaguchi, 2013).

Teknologi akan terus berkembang dari waktu ke waktu. Dengan adanya teknologi, banyak permasalahan dapat diatasi dan dipermudah. Sehingga seharusnya pihak restoran dapat memanfaatkan teknologi tersebut untuk membuat sistem restorannya agar sistem pelayanan dalam restoran dapat terotomatiskan.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah penulis jabarkan sebelumnya, maka dapat dirumuskan rumusan masalahnya sebagai berikut:

1. Bagaimana menyederhanakan proses manajemen antrean pesanan menu restoran dengan aplikasi manajemen antrean pesanan menu restoran?
2. Bagaimana menyederhanakan proses pemesanan menu restoran dengan aplikasi manajemen antraen pesanan menu restoran?
3. Bagaimana hasil pengujian validasi pada aplikasi manajemen antrean pesanan menu restoran dengan menggunakan pengujian blackbox?
4. Bagaimana hasil tingkat *usability* dalam penggunaan aplikasi manajemen antrean pesanan menu restoran dengan pengujian *usability*?

1.3 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

5. Mengetahui bagaimana menyederhanakan proses manajemen antrean pesanan menu restoran dengan aplikasi manajemen antrean pesanan menu restoran.
6. Mengetahui bagaimana menyederhanakan proses pemesanan menu restoran dengan aplikasi manajemen antraen pesanan menu restoran.
7. Mengetahui bagaimana hasil pengujian validasi pada aplikasi manajemen antrean pesanan menu restoran dengan menggunakan pengujian blackbox.
8. Mengetahui bagaimana hasil tingkat *usability* aplikasi dalam penggunaan aplikasi manajemen antrean pesanan menu restoran dengan pengujian *usability*.

1.4 Manfaat

1. Manfaat bagi penulis

   Dapat mengembangkan sistem aplikasi perangkat bergerak sesuai dengan materi perkuliahan yang telah dipelajari sebelumnya.

2. Manfaat bagi peneliti selanjutnya

   Dapat menjadi acuan untuk pengembangan aplikasi yang serupa.

3. Manfaat bagi pengguna

   Dapat membantu untuk melakukan pemesanan makanan di restoran tanpa harus mengantre di depan kasir.

1.5 Batasan Masalah

Penelitian pengembangan sistem memiliki beberapa batasan masalah, yaitu:

1. Sistem dikembangkan menggunakan *library React JS.*
2. Konsep yang akan digunakan untuk membuat sistem adalah *Progressive Website Application.*
3. Sistem dapat berjalan dengan maksimal jika menggunakan *minimum browser* versi Crome 40, Safari 11.1, Firefox 44, dan Edge 17
4. Sistem menggunakan koneksi internet.
5. Data yang diolah bersumber dari restoran Ayam Bakar Wong Solo di Malang yang beralamat di Jl. soekarno hatta no.501 D ruko c-d, Mojolangu, Kec. Lowokwaru, Kota Malang, Jawa Timur 65142.
6. Target pengguna sistem adalah pelanggan yang menggunakan Ayam Bakar Wong Solo sebagai tempat makan.

1.6 Sistematika Pembahasan

Skripsi ini disusun menjadi tujuh bab yang terdiri dari:

1. Bab 1 – PENDAHULUAN

   Berisi tentang latar belakang, rumusan masalah, tujuan dari penelitian, manfaat dari penelitian, batasan penelitian, dan sistematika pembahasan.

2. Bab 2 – LANDASAN KEPUSTAKAAN

   Memuat kajian-kajian kepustakaan yang relevan yang akan digunakan sebagai referensi dalam melakukan penelitian ini.

3. Bab 3 – METODOLOGI PENELITIAN

   Memuat alur penelitian sebagai proses penyelesaian masalah dalam penelitian.

4. Bab 4 – ANALISIS KEBUTUHAN

   Memuat proses penggalian analisis kebutuhan dalam proses pengembangan sistem.

5. Bab 5 – PENGEMBANGAN METODOLOGI KERANGKA KERJA SCRUM

   Memuat perancangan dan pemodelan sistem berdasarkan data yang telah didapat di tahap analisis kebutuhan serta implementasi pengembangan sistem berdasarkan pemodelan yang telah dilakukan sebelumnya.

6. Bab 6 – PENGUJIAN

   Memuat pengujian sistem yang dilakukan oleh responden dan menganalisis hasil yang telah diperoleh, Seperti Pengujian *Black Box* dan analisis hasil pengujian.

7. Bab 7 – PENUTUP

Bab penutup berisi kesimpulan dan saran yang merupakan uraian bab-bab sebelumnya. Hasil kesimpulan dan saran yang diperoleh diharapkan dapat bermanfaat dalam pengembangan sistem aplikasi selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Dalam melakukan penelitian ini, terdapat beberapa penelitian sebelumnya yang relevan dengan manajemen antrean pesanan menu restoran, yaitu penelitian yang berjudul Aplikasi Reservasi Menu Restoran Berbasis Web dan *Mobile* Android di Cowek Ireng (Rasid, Supriyono, & Setiawan, 2018). Pada penelitian ini, terdapat masalah antrean panjang di restoran yang diatasi dengan membuat aplikasi pemesanan menu restoran pada restoran Cowek Ireng. Hasil dari penelitian ini diharapkan bisa menunjang proses manajemen pengelolaan pemesanan menu, pengelolaan data *user* seperti koki, kasir, pelanggan, dan pimpinan, serta pengelolaan reservasi menu dan tempat.

Penelitian yang relevan lainnya adalah penelitian yang berjudul *Online Food Ordering System* (Singh & Kanade, 2018). Penelitian ini membahas industri makanan selalu menjadi lahan yang mengundang profit bukan hanya bagi pemilik restoran, namun juga kepada pengguna/*customer* dan distributor. Perubahan yang besar di indsutri makanan dan juga semakin meningkatnya teknologi dengan memanfaatkan internet dapat membuat pemesanan menu makanan secara *online* di restoran menjadi kebutuhan masyarakat. Pengguna dapat dengan mudah memesan menu makanan di restoran menggunakan *real-time online food ordering*, menelusuri proses pembuatan makanan, dan memberikan *feedback* kepada menu makanan dan juga restoran agar dapat meningkatkan produktivitas restoran. Inisial implementasi system dilakukan pada 2 *restoran*/mess pada 5 area.

Penelitian tentang *Customer-Oriented Restaurant Management System* (CORMS) juga membahas tentang manajemen antrean pesanan menu restoran. Penelitian ini menjadikan sebuah sistem *multiplatform* (terutama pada *tablet*) yang dapat mengatur pemesanan di sebuah restoran tidak hanya pada pemesanannya saja, namun juga mengatur dan mengubah menu-menu di restoran dengan *user interface* yang baik. Selain itu dengan menampilkan rekomendasi menu makanan pada restoran akan memberikan *user experience* yang baik kepada *customer* (Davis, Francis, Sukumaran, E, & Nair, 2017).

2.2 Manajemen Antrean Pesanan Menu Restotoran

Tingginya minat pembeli pada sebuah restoran dapat mengakibatkan panjangnya antrean, khususnya pada hari libur. Antrean pada sebuah restoran seringkali membuat pelanggan tidak puas dan bahkan akhirnya memilih untuk meninggalkan restoran. Untuk mengatasi hal tersebut, perlu dilakukan manajemen antrean pesanan menu restoran. Manajemen antrean pemesanan menu di sebuah restoran dapat menggunakan satu kasir, yang membuat pelanggannya masuk pada *waiting line*, lalu pelanggan menunggu dan mendapatkan pelayanan. Untuk mengatasi waktu tunggu yang terlalu lama, restoran dapat memberikan fasilitas tambahan untuk pelayanan kasir restoran, sehingga membuat sistem antrean menjadi *Multichannel-Single Phase* yang mana

terdapat dua jalur antrean dengan dua fasilitas pelayanan kasir. (Susila, Panji, & Prima, 2007)

Manajemen antrean pesanan menu restoran dengan berfokus kepada *Customer* dapat dengan menggantikan sistem yang sudah ada yaitu penggunaan kertas untuk pemesanan dengan sistem baru yang menggunakan teknologi sebagai pengganti penggunaan kertas dalam pemesanan menu di restoran. Dorongan untuk sistem *online food ordering* semakin meningkat dari hari ke hari, hal ini dikarenakan dapat memudahkan akses dan fleksibilitas dari pekerjaan sehari-hari di sebuah restoran. Dengan teknologi manajemen antrean pesanan menu restoran, pihak restoran dapat dengan mudah mengganti menu restorannya dan mengatur pemesanan yang ada di restoran tersebut. (Davis et al., 2017)

2.3 Aplikasi Perangkat Bergerak

Aplikasi perangkat bergerak (*mobile application)* adalah suatu set program atau *software* yang berjalan atau beroperasi pada sebuah perangkat bergererak atau *mobile device*. Aplikasi tersebut akan melakukan tindakan atau tugas-tugas khusus yang diberikan kepada penggunanya. *Mobile Application* adalah suatu segmen yang baru dan cepat dalam teknologi informasi dan komunikasi. Aplikasi perangkat bergerak memiliki kapasitas yang ringan, ramah bagi pengguna, dapat diunduh, dan dijalankan pada *mobile phone*.

Beberapa aplikasi *mobile* merupakan aplikasi *pre-installed* yang sudah terinstall dalam sebuah smartphone, dan aplikasi perangkat bergerak lainnya dapat diunduh oleh pengguna melalui internet pada sebuah *market* yang disediakan oleh *developer* dan kemudian di-*install* pada perangkat. Dengan banyaknya aplikasi perangkat bergerak yang dikembangkan oleh *developer,* maka *market* dari aplikasi perangkat bergerak pun semakin meningkat. Sehingga beberapa aplikasi dapat di-*install* pada *platform* yang berbeda-beda seperti iPhone, Blackberry, Android, Symbian, dan Windows. Walaupun aplikasi *mobile* memiliki beberapa batasan seperti layar yang kecil, akses navigasi, dan kecepatan yang lambat; penggunaan aplikasi perangkat bergerak terus meningkat dari hari ke hari, banyak masyarakat lebih memilih untuk menggunakan aplikasi perangkat bergerak dibandingkan dengan aplikasi *desktop* untuk melakukan hal yang ringan (Islam & Mazumder, 2010).

Terdapat beberapa perbedaan antara aplikasi web dan perangkat bergerak terhadap *services, device, network, user,* dan *usage context.* Perangkat *mobile* memiliki ukuran tampilan yang relatif kecil dan juga memiliki bentuk tampilan yang berbeda dengan *desktop computer*. Selain itu, terdapat juga batasan pada *input, CPU, memory, bandwidth,* dan *data transfer rate*. Sehingga dapat disimpulkan bahwa perangkat bergerak lebih baik digunakan untuk tujuan yang terbatas. Batasan yang dimiliki oleh pernagkat *mobile* dijelaskan pada Tabel 2.1

Tabel 2.1 Keterbatasan perangkat *mobile*

Batasan	***Mobile services***	***Desktop computers***
Small display	Ya	Tidak
Limited input possibilities	Ya	Tergantung pada perangkat
CPU	Ya	Tidak
Small memory	Ya	Tidak
Limited bandwidth	Ya	Tidak
Small data transfer rate	Ya	Tidak
High latency	Ya	Tidak
Cost of use	Ya	Tidak

Walaupun terdapat perbedaan baik pada perangkat maupun *network* antara *web* dan *mobile*, pada kenyataannya perbedaan yang terbesar adalah pada pengguna itu sendiri. Pengguna aplikasi *web* lebih familiar dengan penggunaan computer maupun WWW. Berbeda dengan pengguna perangkat bergerak, mereka tidak terlalu familiar dengan hal tersebut. Ditambah lagi, pengguna perangkat bergerak cenderung lebih tidak sabar dan memiliki kebutuhan yang bervariasi. Mereka tidak suka *browsing* dengan perangkat bergerak karena akan terasa lebih lambat dan terganggu karena *display* yang kecil (Oinas-Kukkonen & Kurkela, 2003).

2.4 Progressive Website Application

Progressive Website Application atau biasa disebut PWA menggabungkan yang terbaik dari aplikasi website dan *mobile*. PWA dibangun menggunakan menggunakan teknologi *website application* yang bertindak selayaknya seperti sebuah aplikasi *native mobile.* Ide tentang PWA ini pertama kali disokong oleh *engineer* Google, Alex Russel, pada Juni 2015. Konsep dari PWA ini adalah teknologi, konsep desain, dan Web APIs yang membuat aplikasi website yang biasanya diakses melalui browser menyediakan pengalaman *app-like* seperti *push notification, work offline,* tampilan atau *app-shell* terlihat dan terasa seperti aplikasi *native mobile*, dan *load from home screen*.

Progressive Website Application memiliki beberapa fitur, seperti:

1. Dapat diandalkan

   Memuat aplikasi secara instan, karena dapat berjalan secara *offline,* sehingga ketika apliaksi dibuka, maka *service worker* yang bekerja selayaknya *client proxy* akan menghandle seluruh request sesuai yang sudah termuat di dalam *cache* saat aplikasi pertama kali dimuat atau diperbarui.

2. Cepat

   Merespon interaksi pengguna dengan cepat dengan animasi yang halus serta tanpa *Janky Scrolling*. Hal ini dikarenakan waktu untuk memuat membutuhkan lebih dari 3 detik, 53% pengguna akan meninggalkan situs tersebut.

3. Mengikat pengguna dan dapat diinstall

   Banyak cara untuk mengikat pengguna, seperti *user experience* yang baik dengan menerapkan *design* seperti aplikasi *native mobile*, fitur instalasi aplikasi ke *home screen* tanpa perlu instalasi melalui *store*, dan dengan adanya *push notification* maka dapat mengikat kembali penggunanya. *Developer* dapat menggunakan *app manifest* untuk mengatur *icon* pada *home screen*, mengontrol bagaimana tampilan ketika aplikasi akan dibuka, *screen orientation,* dan pengaturan *window.*

4. *Fresh*

   Selalu *up-to-date* dengan fitur yang terbaru, hal ini dapat diatur menggunakan *service worker*. Selain itu, terdapat fitur *background sync* untuk sinkronisasi data pengguna yang prosesnya berada di *background.*

Progressive Website Application akan meningkatkan performa website dengan membuatnya menjadi lebih cepat, dapat diandalkan, dan mengikat pengguna (Karpagam, 2017).

2.5 React JS

React JS merupakan suatu *UI library* yang dikembangkan oleh Facebook yang memiliki fasilitas *interactive, stateful,* dan *reusable UI components. Library* ini telah digunakan oleh Facebook. React JS menjadi *library* yang paling baik dalam proses *rendering user interface* yang sangat kompleks namun dengan performa yang tinggi. Dasar dari React adalah konsep yang menggunakan *Virtual DOM* dimana dapat digunakan untuk *rendering* pada *client-side* ataupun *server-side*. Cara kerjanya adalah dengan manipulasi DOM sesuai dengan perubahan *state* secara *up-to-date* (Kumar & Singh, 2016).

2.6 Kode QR

Denso Wave menciptakan *Kode QR* pada tahun 1994. Denso Wave menggunakan Kode QR sebagai trademark yang telah tergistrasi untuk melacak produknya. Perluasan penggunaan *Kode QR* kemudian dipromosikan oleh Denso Wave sebagai metode yang cepat dan mudah untuk *tracking* pada suatu produk. Konsep dibalik *Quick Response Code* diantaranya:

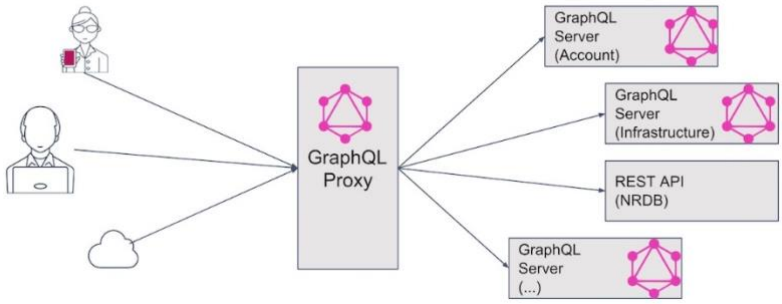
1. Penyimpanan yang lebih besar
2. Variasi data yang dapat disembunyikan dalam *Kode QR* adalah teks, nama, URL, SMS, email, dan kontak informasi
3. Dapat dilakukan *decoding* pada platform yang berbeda

4. Digunakan sebagai *online payment*
5. Menyimpan tipe data yang berbeda-beda seperti *numeric, alphabetic, binary,* dan Kanji

Kode QR telah digunakan dalam skala besar pada pemasaran atau kampanye sejak awal tahun 1990-an untuk menciptakan interaksi dengan konsumen. Denso Wave menggunakan ekstensi teknologi ini karena potensinya dalam perdagangan otomotif. Kode QR Pertama kali dating ke pasar sebagai iklan produk pada tahun 2011 ketika industry telekomunikasi sedang menaik. Hingga saat ini, Kode QR telah menjadi teknologi yang populer pada *smartphone* (Ozdenizci, Aydin, Coskun, & Ok, n.d.).

2.7 GraphQL

Graphql adalah sebuah konsep baru dalam membangun sebuah *application programming interface* (API) dan diimplementasikan pada sisi *server*. Posisi Graphql berada pada sisi klien dan *server* yang berhubungan dalam suatu API untuk mengambil dan memanipulasi data. Graphql didesain untuk berkolaborasi dengan bahasa pemrograman *server* yang lain dengan baik (Buna, 2016). Diagram yang menggambarkan bagaiman cara GraphQL bekerja dapat dilihat pada Gambar 2.1.



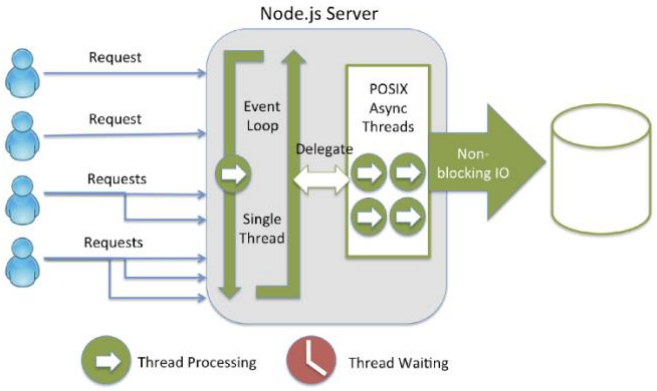
Gambar 2.1 Graphql diagram

(sumber: blog.newrelic.com)

Pada Gambar 2.1 menunjukkan bahwa *client* mengirimkan *request* ke GraphQL *server* melalui *proxy* GraphQL. Kemudian *request* tersebut diolah oleh GraphQL *server*.

2.8 Node JS

Node js merupakan suatu perangkat lunak untuk pengembangan aplikasi dari bahasa pemrograman Javascript yang memungkinkan Javascript dijalankan pada sisi *server*. Node.js berjalan dengan basis *event* dimana pada suatu kode program dijalankan hingga selesai. Setelah kode program selesai dijalankan, baru dialihkan ke kode program selanjutnya. Node JS memiliki banyak modul yang berguna sehingga tidak perlu menulis semua kode dari awal (Kiessling, 2015).



Gambar 2.2 Node JS flow

(sumber: strongloop.com)

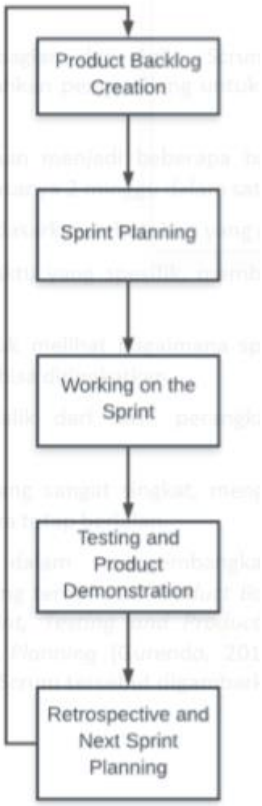
2.9 Model Pengembangan Perangkat Lunak

Pengembangan model perangkat lunak memiliki beberapa model pengembangan dalam proses pelaksanaan penelitian yaitu metodologi *Agile* dengan memanfaatkan kerangka kerja Scrum. Model untuk *software engineering* sangat banyak, seperti model *spiral, prototype, waterfall, incremental, RAD,* dan metode formal (Roger, 1997).

Pengembangan sistem manajemen antrean pesanan menu restoran menggunakan metodologi *Agile* dengan menerapkan kerangka kerja Scrum karena metode ini dapat digunakan dalam pengembangan sistem yang memiliki tingkat kompleksitas tinggi dan perubahan terus-menerus. Salah satu bagian dari metodologi Agile adalah Scrum. Karakteristik dari kerangka kerja scrum untuk memudahkan pengembang dalam bekerja adalah:

1. Menentukan *sprint* atau pemecahan durasi pekerjaan sehingga menjadi beberapa bagian. Biasanya dipecahkan 2 minggu dalam satu *sprint.*
2. Menentukan prioritas dan perencanaan *sprint* yang didasarkan pada kebutuhan terpenting pada saat itu.
3. Mengevaluasi bagaimana proses perjalanan dalam satu *sprint* dan melihat peluang apa saja yang mungkin bisa ditingkatka, sehingga akan mendapatkan *feedback* dari pengerjaan perangkat lunak.
4. *Daily meeting* untuk mengidentifikasi halangan yang mungkin terjadi dan memastikan seluruh proses dapat tetap berjalan dengan baik.

Ada beberapa tahapan dalam mengembangkan aplikasi dengan menggunakna metode Scrum yang terdiri atas *Product Backlog Creation, Sprint Planning, Working on the Sprint, Testing and Product Demonstration*, dan *Restrospective and Next Sprint Planning* (Gurendo, 2015). Adapun alur dari tahapan-tahapan dalam metode Scrum tersebut digambarkan pada Gambar 2.3.



Gambar 2.3 Alur Pengembangan dalam Metode Scrum

Product Backlog merupakan fitur-fitur yang diimplementasikan melalui pengembangan perangkat lunak. Cara untuk membuat *product backlog* dalam fase *product backlog creation* adalah dengan melakukan wawancara, dan hasil dari wawancara kepada calon pelanggan dijadikan sebuah *user story*. Setelah mendapatkan seluruh *backlog* dari sistem, kemudian *product backlog* tersebut diurutkan berdasarkan prioritas.

Sprint Planning adalah perencanaan *sprint* secara bertahap yang dilakukan melalui proses pengembangan perangkat lunak. Langkah pertama yang dilakukan adalah menentukan durasi dari *sprint* tersebut. Semakin pendek durasi *sprint*, maka akan semakin banyak frekuensi rilis dari sebuah aplikasi. Selanjutnya tim dapat menentukan daftar prioritas utama penyelesaian *product backlog,* Setelah itu, setiap orang dapat memberikan informasi berapa lama masing-masing dari seseorang dapat menjalankan *task* yang diberikan.

Working on the Sprint merupakan proses untuk mengimplementasikan perangkat lunak yang akan dikembangkan. Untuk memantau kinerja dari sebuah tim, biasanya digunakan *Task Board* yang dapat merepresentasikan status pengerjaan dari *product backlog* yang ada. Status tersebut dapat berupa *Todo, In Progress, Testing,* dan *Done*.

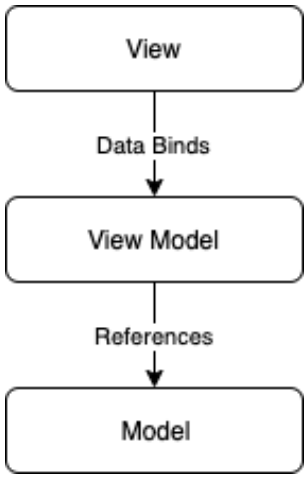
Testing and Product Demonstration adalah menguji aplikasi yang telah dibuat dalam satu *sprint*. Kemudian setelah lolos uji, tim dapat mendemonstrasikan hasil

pekerjaan mereka kepada *stakeholder*, sehingga orang-orang yang memiliki kepentingan dapat memutuskan aplikasi yang akan digunakan selanjutnya.

Proses *retrospective and Next Sprint Planning* merupakan proses diskusi hasil dari *sprint* yang telah dilakukan. Diskusi tersebut berisi tentang bagian apa saja yang berjalan lancar dan bagaimana cara tim untuk memperbaiki kesalahan untuk diimplementasikan pada *sprint* selanjutnya.

2.10 MVVM *Design Pattern*

Model View ViewModel (MVVM) adalah *design pattern* atau *architectural pattern* yang digunakan untuk memisahkan antara *User Interface* dan *Application Logic.* D*esign pattern* MVVM membentuk *linear* yang terdiri dari View-View Model-Model, seperti diagram pada Gambar 2.4 (Vanus et al., 2016)



Gambar 2.4 Pattern MVVM

Konsep dasar dari MVVM adalah View Model yang menangani komunikasi antara View dan Model. View akan selalu mengamati perubahan data yang dilakukan pada logic yang terdapat di View Model, sedangkan pengelolaan data pada View Model akan mereferensi data yang tersimpan pada Model. Sehingga View Model tidak akan memerdulikan View yang melakukan *binding* ke View Model. (Kouraklis, 2016)

Penerapan *design pattern* yang memisahkan antara bisnis dengan *logic* dan meningkatkan *reusability* pada kode program dapat memudahkan pengembangan, pengujian, dan *maintain* aplikasi. Hal ini dikarenakan *developer* dan *page designer* dapat saling kooperatif, sehingga ketika terdapat perubahan UI, tidak perlu lagi mengubah *logic* yang sudah ada untuk memanipulasi data. (Li et al., 2015)

2.11 Google Material Design

Material Design adalah *visual language* yang menyintesis prinsip klasik dari *design* yang baik dengan inovasi teknologi dan sains. Material adalah sistem *guidline*, komponen, dan *tools* yang dapat diadaptasi dan dapat didukung oleh praktik terbaik dari desain *user interface*. Didukung oleh kode *open-source*,

Material merampingkan kolaborasi antara *designers* dan *developers*, dan membantu tim untuk membangun produk yang indah dengan cepat.

Berikut adalah prinsip dari Material Design:

1. Material adalah metafora

   Terinspirasi dari dunia fisik dan teksturnya, termasuk bagaimana memantulkan cahaya dan merepresentasikan *shadows*. Material menata kembali medium kertas dan tinta

2. *Bold*, grafis, *intentional*

   Dipandu oleh metode desain *typography, grids, scale, color,* dan *imagery* untuk membuat hirarki, makna, dan fokus yang ditanamkan pada pengguna

3. *Motion provides meaning*

   Motion memusatkan pada perhatian dan menjaga kontinuitas, melalui *feedback* yang halus dan transisi koheren. Ketika suatu elemen muncul di layar, akan memberikan pengaruh interaksi pada *environment*.

4. Pondasi yang fleksibel

   Didesain untuk mengekspresikan sebuah merek, oleh karena itu, material terintegrasi oleh *code-base* yang dapat diubah dan disesuaikan

5. *Cross-platform*

   Material menggunakan *shared components* yang dapat digunakan pada berbagai *platform* seperti Android, iOS, dan *web*

Guideline Material Design membantu untuk membuat produk yang indah dan cepat. Pengembang dapat melakukan *theming* untuk kustomisasi desain (Developer, 2019).

2.12 Pengujian Perangkat Lunak

2.12.1 Blackbox Testing

Black Box Testing merupakan pengujian perangkat lunak untuk mengetahui fungsi, masukan dan keluaran dari perangkat lunak telah sesuai dengan spesifikasi yang dibutuhkan atau belum dari segi spesifikasi fungsional tanpa menguji desain dan kode program (Rosa & Salahuddin, 2011) . Uji coba *black box* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya yaitu fungsi-fungsi yang salah atau hilang, *interface*, struktur data atau akses *database* eksternal, kesalahan performa ataupun inisialisasi dan terminasi (Ayuliana, 2009).

2.12.2 Usability Testing

Usability testing atau uji ketergunaan sebuah aplikasi yang merupakan metode evaluasi yang digunakan untuk mengukur tingkat kemudahan dan kenyamanan penggunaan dan interaksi pengguna terhadap sebuah sistem

informasi (Henriyadi & Mulyati, 2016). Usability menurut standar internasional ISO 9241 (Ergonomic Requirements for Office Work with Visual Display Terminals) adalah pengukuran suatu produk sejauh mana dapat digunakan untuk mencapai suatu tujuan secara efektivitas, efisiensi, dan kepuasan penggunaan yang telah ditetapkan (Standard, 1998).

2.12.3 Regression Testing

Regression testing adalah pengujian yang berfokus pada variasi yang terjadi dalam daur hidup sebuah perangkat lunak, dan menghasilkan kualiatas software yang akan memberikan efek samping. *Regression testing* digunakan untuk memonitor perubahan pada sebuah perangkat lunak dan memberikan timbal balik terhadap perubahan tersebut secara terurut dan konsisten. Ketika perangkat lunak diubah, beberapa aspek dari perangkat luna kseperti konfigurasi dan program sebelumnya juga dapat berubah, maka *regression testing* akan digunakan untuk menjamin bahwa perubahan tersebut tidak akan memberikan sebuah *error* atau bugs pada fitur lain dikarenakan akan diuji secara keseluruhan setiap penambahan fitur baru (Xiaowen, 2013).

BAB 3 METODOLOGI PENELITIAN

3.1 Jenis Penelitian

Penelitian ini menggunakan jenis implementatif-pengembangan (*development*). Karena dapat menerapkan prinsip-prinsip rekayasa, pemodelan sistem, dan tahapan pengembangan yang lengkap seperti tahap analisis, perancangan, implementasi, dan pengujian.

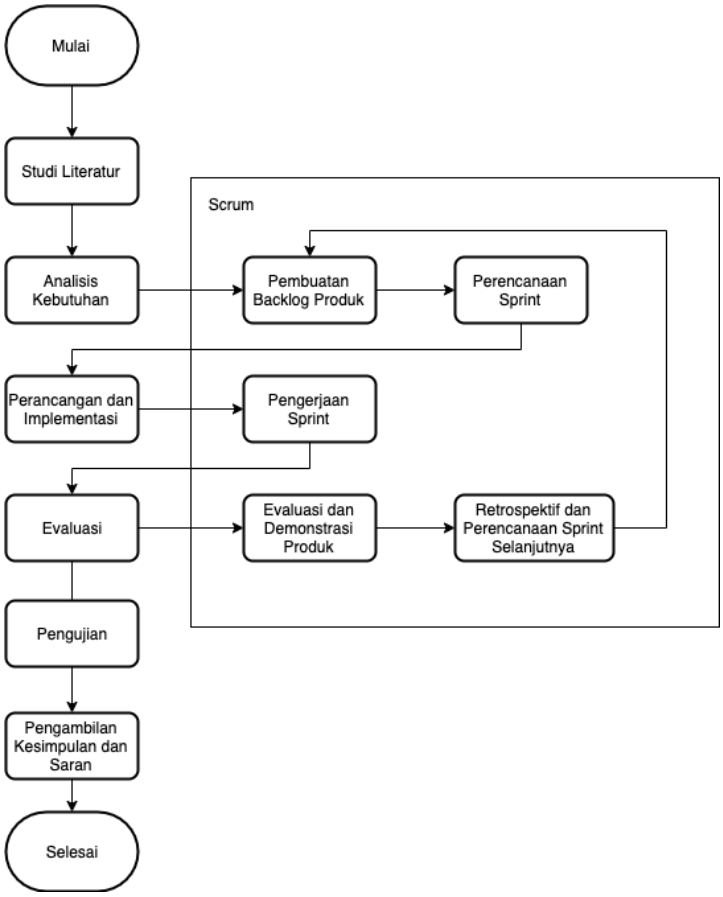
3.2 Tempt dan Waktu Penelitian

1. Tempat Penelitian

   Penelitian dilaksanakan di Ayam Bakar Wong Solo yang memiliki alamat di Jl. soekarno hatta no.501 D ruko c-d, Mojolangu, Kec. Lowokwaru, Kota Malang, Jawa Timur 65142.

2. Waktu Penelitian

   Proses penelitian yang penulis lakukan akan dilaksanakan dalam waktu dua bulan yaitu pada tanggal 1 September 2019 hingga 8 Desember 2019.



Gambar 3.1 Alur Metodologi Penelitian

3.3 Diagram Alir Metode

Pada bab ini akan membahas alur pelaksanaan atau tahapan-tahapan dalam pengembangan sistem manajemen antrean pesanan menu restoran. Hal ini dilakukan agar proses dapat berjalan dengan baik dan sesuai dengan tujuan. Metode dalam pengembangan perangkat lunak sistem manajemen antrean pesanan menu restoran adalah dengan metode SDLC Scrum. Beberapa tahapan yang dilakukan pada penelitian ini dapat digambarkan dalam Gambar 3.1.

3.4 Analisis Kebutuhan

Analisis kebutuhan merupakan tahap dalam mempelajari kebutuhan calon pelanggan agar mendapatkan definisi kebutuhan sistem atau tahap untuk menentukan kebutuhan dari seluruh elemen sistem. Pada fase ini, seluruh kebutuhan sistem akan dideskripsikan secara lengkap. Tahap analisis kebutuhan terdapat dua bagian, yaitu gambaran umum sistem dan analisis kebutuhan perangkat lunak. Analisis kebutuhan perangkat lunak diantaranya adalah:

1. Menjelaskan mengenai identifikasi aktor.
2. Menjelaskan mengenai pembuatan *User Story.*
3. Menjelaskan mengenai daftar kebutuhan fungsional.
4. Menjelaskan mengenai pembuatan *backlog product.*
5. Menjelaskan mengenai perencanaan *sprint.*
6. Menjelaskan mengenai analisis data.
7. Menjelaskan mengenai *use case diagram.*
8. Menjelaskan mengenai *use case* skenario.

Pada fase analisis kebutuhan, hal yang harus dilakukan pertama kali adalah mendeskripsikan aplikasi secara umum, sehingga pembuatan aplikasi yang akan dibuat dapat lebih mudah untuk dipahami. Selanjutnya adalah penentuan pengguna sistem yang kemudian akan menjadi aktor di dalam sistem. Setelah itu, peneliti mendapatkan *user story* dari masing-masing aktor yang sudah ditentukan. *User story* yang telah didapatkan dapat digunakan untuk menentukan daftar fungsionalitas dari sistem. Daftar fungsionalitas ini akan menjadi rujukan utama dalam mengimplementasikan sistem.

Setelah mendapatkan fungsionalitas dari sistem, kemudian peneliti membagi *backlog* atau fungsionalitas menjadi beberapa *sprint*. Pembagian *backlog* produk ke beberapa *sprint* ini dilakukan dengan mengurutkan *backlog* sesuai dengan prioritas yang dibutuhkan sistem untuk dapat bekerja terlebih dahulu. Setelah itu peneliti menentukan durasi *sprint* yang akan digunakan oleh tim berdasarkan tingkat kompleksitas dari fungsionalitas yang telah dibuat. Selanjutnya dilakukan pembagian *backlog* produk beradasarkan durasi *sprint*.

3.5 Perancangan dan Implementasi

Tahapan ini adalah tahap dalam menentukan perangkat keras *(hardware)*, arsitekturndiaya, dan sistem persyaratan yang dibutuhkan. Dalam tahapan ini, dilakukan segala persiapan akan sesuatu yang dibutuhkan dalam melakukan implementasi pada aplikasi manajemen antrean pesanan menu restoran. Tahapan Perancangan Sistem diantaranya adalah :

1. Perancangan Arsitektur

   Perancangan arsitektur dilakukan dengan pemodelan sistem menggunakan diagram *Unified Modelling Language.* Diagram UML yang digunakan adalah *sequence diagram* dan *class diagram*.

2. Perancangan Komponen

   Perancanngan komponen adalah melakukan perancangan berupa sampel algoritme utama dalam bentuk *pseudocode* yang diambil pada *class controller*, sehingga akan mendefinisikan *logic* dari suatu program.

3. Perancangan Antarmuka

   Proses perancangan antarmuka dari sistem terdiri dari *layout* atau tata letak komponen berdasarkan kebutuhan dari sistem. Rancangan antarmuka ini akan diguankan dalam implementasi antarmuka yang dihasilkan setelah program diimplementasikan.

Pada tahap implementasi, peneliti menuliskan serangkaian kode program untuk mewujudkan perancangan yang telah dirancang sebelumnya. Kode program tersebut menggunnakan bahasa pemrograman Javascript dengan menggunakan React JS sebagai *framework* untuk aplikasi pengguna, sedangkan pengolahan data dilakukan menggunakan Node JS sebagai Javascript *runtime* untuk memproses data dan GraphQL sebagai *resolver* dan *query language*. Tahap implementasi sistem diantaranya adalah menjelaskan spesifikasi sistem, dilanjutkan dengan mendefinisikan batasan implementasi, dan kemudian implementasi kode program dan antarmuka.

3.6 Evaluasi

Evaluasi digunakan untuk menentukan telah terselesaikannya semua *backlog* yang telah dibuat pada setiap *sprint*. Tim harus menyelesaikan seluruh *backlog* yang sudah direncanakan untuk dikerjakan selama satu sprint*,* jika ada produk *backlog* yang belum selesai, maka *sprint* selanjutnya dapat menangani produk *backlog* tersebut.

3.6.1 Evaluasi dan Demonstrasi Produk

Merupakan tahap evaluasi sistem yang akan dilakukan terhadap sistem yang telah dikembangkan, pengujian yang dilakukan adalah dengan menggunakan metode pengujian *Black Box Validation* dan *Regression Test.* Evaluasi sistem bertujuan untuk mengetahui jalannya kebutuhan fungsional yang didefinisikan

sebelumnya apakah berjalan sesuai yang diinginkan. Tahapan evaluasi sistem diantaranya adalah pengujian *Black Box* (Validasi) dan *Regression Test*.

3.6.2 Retrospektif dan Perencanaan *Sprint* Selanjutnya

Setelah *sprint* pada suatu iterasi telah selesai dilakukan sesuai tahapan maka retrospektif merupakan tahap berikutnya, yang bertujuan untuk mengevaluasi secara utuh proses dalam suatu *sprint*. Evaluasi tersebut akan menghasilkan kesimpulan dari *sprint*. Kemudian hasil retrospektif akan menjadi bahan rujukan untuk merancang kembali *sprint* yang akan dilakukan pada fase iterasi berikutnya.

3.7 Pengujian

Setelah semua proses pengembangan perangkat lunak dalam seluruh *sprint* telah dilakukan, maka sistem harus diuji kepada calon pengguna yang akan menggunakan sistem ini. Pengujian dalam penelitian ini menggunakan *Usability Testing* yang akan mengevaluasi tingkat kebergunaan aplikasi yang berdampak pada tingkat kepuasan pengguna terhadap sistem yang telah dikembangkan, sehingga pengguna dapat dengan nyaman dan puas berinteraksi terhadap sistem.

3.8 Pengambilan Kesimpulan dan Saran

Hasil pengujian dan analisis sistem akan menjadi bahan untuk pengambilan kesimpulan. Kesimpulan diambil untuk mengetahui keberhasilan dan menjawab permasalahan yang telah dirumuskan dalam penelitian. Setelah pengambilan kesimpulan dilanjutkan dengan memberikan saran berdasarkan hasil evaluasi dari penelitian ini.

BAB 4 ANALISIS KEBUTUHAN

Bab ini berisi tentang proses-proses yang akan dilalui untuk mendapatkan produk *backlog*. Untuk menghasilkan sebuah produk yang dapat digunakan, proses yang dilalui akan dilaksanakan dalam durasi *sprint* yang mana pengerjaan sebuah sprint mengacu pada produk *backlog*. Bab ini juga membahas tentang hasil wawancara yang kemudian akan diolah menjadi *User Story*, pemecahan produk *backlog,* dan durasi waktu *sprint.*

4.1 Gambaran Umum Sistem

Gambaran umum sistem merupakan gambaran yang dapat merepresentasikan seluruh proses dari sebuah sistem yang terdiri dari deskripsi umum sistem dan lingkungan sistem.

4.1.1 Deskripsi Umum Sistem

Penelitian ini memiliki tujuan untuk membangun sistem bernama manajemen antrean pesanan menu restoran, yaitu aplikasi manajemen antrean pesanan menu restoran dengan memanfaatkan teknologi kode QR. Aplikasi yang akan dibuat dalam sistem manajemen antrean pesanan menu restoran terdapat dua aplikasi, yaitu aplikasi manajemen antrean pesanan menu restoran untuk pelanggan dan aplikasi manajemen antrean pesanan menu restoran untuk restoran. Kedua aplikasi tersebut dapat berjalan pada platform *website*.

Aplikasi manajemen antrean pesanan menu restoran untuk pelanggan dikembangkan untuk pemesanan makanan yang ada di restoran dengan mendaftarkan nomor antrenya sesuai dengan meja yang pelanggan pilih ketika telah memasuki restoran. Dengan aplikasi manajemen antrean pesanan menu restoran untuk pelanggan, pemesanan makanan dengan metode konvensional dapat diganti dengan menggunakan teknologi digital, sehingga pelanggan tidak perlu lagi memanggil pelayan untuk memesan menu restoran dan pelanggan hanya perlu memilih menu restoran yang akan ditampilkan di layar *smartphone* pelanggan. Pertama-tama, pelanggan memasuki restoran dan memilih tempat duduk yang tersedia dan pelanggan inginkan. Di setiap meja yang ada direstoran sudah terdapat Kode QR yang tertempel, kemudian pelanggan dapat memindai Kode QR tersebut menggunakan kamera yang terdapat pada *smartphone*. Setelah itu pelanggan akan diarahkan ke website dengan halaman sesuai dengan posisi meja dan restoran yang tertanam pada Kode QR pada setiap meja. Setelah halaman menu restoran ditampilkan oleh sistem, maka pelanggan dapat memilih menu yang ingin mereka pesan, kemudian melakukan checkout, memilih metode pembayaran, dan melakukan pemesanan. Setelah itu, pelanggan akan mendapatkan nomor antre untuk dilayani oleh restoran.

Aplikasi manajemen antrean pesanan menu restoran untuk restoran dikembangkan untuk manajemen antrean pesanan di sebuah restoran. Dengan aplikasi manajemen antrean pesanan menu restoran, pihak restoran tidak perlu

lagi mencetak ulang menunya ketika terdapat perubahan pada menu restoran. Selain itu, antre pesanan makanan oleh pelanggan di restoran akan diatur secara otomatis oleh sistem, sehingga tidak memerlukan lagi tenaga manusia untuk mengurutkan pesanan yang sudah dipesan oleh pelanggan. Aplikasi manajemen antrean pesanan menu restoran untuk restoran akan membuat seluruh tahapan pemrosesan pesanan pelanggan dari mulai pesanan dibuat, diurutkan berdasar waktu yang tercatat saat pesanan dibuat, hingga akhirnya pesanan siap disajikan akan dibantu oleh sistem. Alur kerja dari sistem aplikasi manajemen antrean pesanan menu restoran untuk restoran adalah pesanan yang dibuat oleh pelanggan akan masuk ke antrean daftar pemesanan. Setelah itu koki restoran dapat melihat urutan prioritas menu apa saja yang harus disajikan terlebih dahulu. Kemudian setelah menu yang dipesan untuk satu pelanggan siap disajikan, pelayan dapat mengantarkan menu makanannya ke pelanggan yang memesan menu tersebut.

4.1.2 Lingkungan Sistem

Aplikasi manajemen antrean pesanan menu restoran untuk pelanggan dan restoran dikembangkan menggunakan *text editor* Visual Studio Code. *Library* yang digunakan untuk mengembangkan sistem adalah *library* React JS, sedangkan Javascript sebagai bahasa pemrograman yang digunakan. Penanganan *logic* untuk mengolah data yang dilakukan di dalam server ditanganin menggunakan Node JS dan GraphQL sebagai *resolver* dan *query language*. Basis data atau *database* yang digunakan adalah basis data MySQL. Aplikasi ini nantinya diharapkan dapat berjalan pada platform *website*.

4.2 Analisis Kebutuhan Perangkat Lunak

4.2.1 Identifikasi Aktor

Aktor merupakan seluruh pengguna aplikasi yang terlibat di dalam sistem. Orang yang akan melakukan interaksi secara langsung dengan sistem direpresentasikan sebagai aktor. Adapun aktor-aktor yang terlibat secara langsung dalam sistem manajemen antrean pesanan menu restoran dapat dilihat pada Tabel 4.1

Tabel 4.1 Daftar aktor dalam Sistem

Aktor	Deskripsi
Tamu Pelanggan	Pelanggan restoran yang baru membuka aplikasi manajemen antrean pesanan menu restoran dan belum terdaftar di dalam sistem aplikasi manajemen antrean pesanan menu restoran untuk pelanggan
Pelanggan	Pelanggan restoran yang telah terdaftar di dalam sistem aplikasi manajemen antrean pesanan menu restoran untuk pelanggan

Tamu Restoran	Orang yang belum terdaftar dalam aplikasi manajemen antrean pesanan menu restoran untuk restoran
Operator Restoran	Kapten Stelling atau biasa disebut operator, yaitu orang yang mengatur pesanan makanan di restoran Wong Solo. Operator restoran yang sudah terdaftar dan dapat mengoperasikan fitur dan fungsionalitas aplikasi manajemen antrean pesanan menu restoran untuk restoran.

4.2.2 Pembuatan User Story

User story merupakan ekspektasi dari calon pengguna sistem terhadap hal-hal yang ingin dicapai. Wawancara yang dilakukan kepada calon pengguna sistem akan menghasilkan *user story. User story* akan dijadikan sebagai dasar dari pembuatan fungsionalitas-fungsionalitas yang akan dibangun pada sistem yang akan dibuat. Kesimpulan dari *user story* akan menjadi dasar kebutuhan pengguna dalam sebuah sistem.

Sistem aplikasi manajemen antrean pesanan menu restoran memiliki dua jenis pengguna, yaitu pelanggan dan operator restoran. Untuk menggali *user story* dari calon pengguna pelanggan, pertanyaan yang dilakukan dalam wawancara terhadap pelanggan difokuskan kepada pertanyaan berupa permasalahan apa saja yang dialami oleh pelanggan ketika ingin memesan makanan di sebuah restoran dan ekspektasi dari pelanggan terhadap fitur-fitur yang akan disediakan jika terdapat sebuah aplikasi yang dapat membantu pelanggan dalam melakukan pemesanan menu makanan di suatu restoran. Sedangkan untuk menggali *user story* dari calon pengguna operator restoran, wawancara akan dilakukan terhadap perwakilan operator restoran Ayam Bakar Wong Solo. Pertanyaan-pertanyaan yang akan diajukan kepada operator restoran berupa kendala apa saja yang dialami ketika melakukan perannya untuk mengatur penyajian makanan untuk pelanggan. *User story* yang didapatkan dari hasil wawancara yang telah dilakukan dapat dilihat pada Tabel 4.2 dan Tabel 4.3.

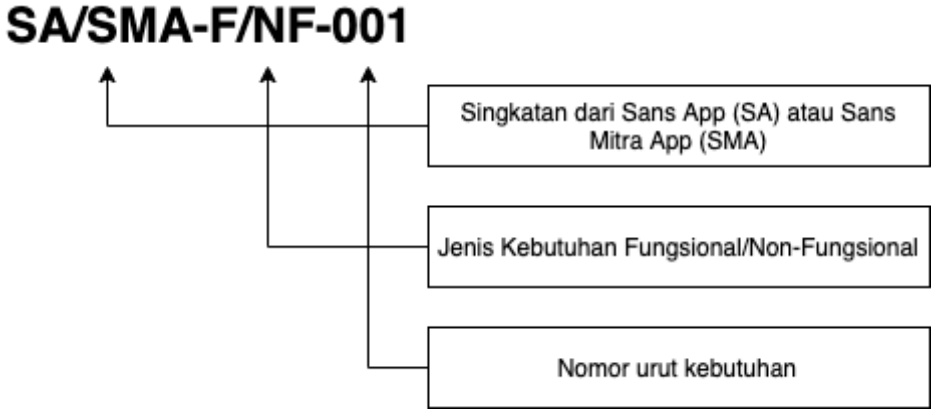
Tabel 4.2 User story hasil wawancara Pelanggan

No	User Story
1	Sebagai pelanggan saya ingin memesan makanan secara online dan mudah melalui aplikasi
2	Sebagai pelanggan saya ingin melakukan pembayaran makanan di restoran secara tunai ke kasir atau pembayaran online
3	Sebagai pelanggan saya ingin melihat makanan favorite atau rekomendasi di sebuah restoran
4	Sebagai pelanggan saya ingin menyimpan menu-menu restoran yang saya sukai

Tabel 4.3 User story hasil wawancara Operator Restoran

No	User Story
1	Sebagai operator restoran saya ingin bisa mengatur menu restoran secara sistematis melalui komputer
2	Sebagai operator restoran saya ingin bisa mengatur menu makanan restoran
3	Sebagai operator restoran saya ingin melihat daftar pesanan menu restoran dalam bentuk tabel
4	Sebagai operator restoran saya ingin bisa mengatur stok menu restoran
5	Sebagai operator restoran saya ingin bisa melihat transaksi pemesanan menu restoran secara *up-to-date*

4.2.3 Kebutuhan Fungsional Sistem



Gambar 4.1 Identitas Kebutuhan

Kebutuhan fungsional sistem merupakan sebuah daftar yang mendeskripsikan layanan dan kemampuan yang disediakan oleh sistem. Kebutuhan fungsinoal dibuat berdasarkan *User Story* yang telah didapatkan pada subbab sebelumnya. Kemudian setelah membuat kebutuhan sistem dibuat, tahap selanjutnya adalah membuat kebutuhan tersebut menjadi *backlog* produk pada *sprint*. Dalam pembuatan kebutuhan fungsinoal, setiap kebutuhan fungsional sistem diberi suatu identitas sehingga kebutuhan tersebut menjadi lebih mudah untuk direferensikan pada seluruh tahap penelitian. Format identitas untuk setiap kebutuhan dapat dilihat pada Gambar 4.1

SA/SMA melambangkan singkatan dari nama aplikasi, yaitu Sans App (SA) dan Sans Mitra App (SMA). Aplikasi Sans App adalah aplikasi manajemen antrean pesanan menu restoran untuk pelanggan dan aplikasi Sans Mitra App adalah aplikasi manajemen antrean pesanan menu restoran untuk restoran. Sedangkan untuk berikutnya, F melambangkan jenis kebutuhan fungsinoal (F) atau kebutuhan non-fungsional (NF). Kode 001 merepresentasikan nomor urut kebutuhan. Daftar kebutuhan fungsional sistem dapat dilihat pada Tabel 4.4

Tabel 4.4 Kebutuhan Fungsional Sistem

No	Kode	Deskripsi Kebutuhan
1	SA-F-001	Sistem harus mampu menyediakan tempat untuk mendaftarkan dan memasukkan tamu pelanggan ke sistem melalui akun Google pelanggan
2	SA-F-002	Sistem harus mampu menyediakan pengambilan gambar melalui kamera untuk pindai Kode QR pada meja restoran
3	SA-F-003	Sistem harus mampu menampilkan daftar meu restoran yangn telah dipidai pada meja restoran
4	SA-F-004	Sistem harus mampu menyediakan ringkasan pemesanan dan menampilkan harga
5	SA-F-005	Sistem harus mampu membuat pesanan
6	SA-F-006	Sistem harus mampu menampilkan daftar riwayat pemesanan yang telah dilakukan oleh pelanggan
7	SA-F-007	Sistem harus mampu mengubah pesanan saat status pesanan masih belum diproses
8	SA-F-008	Sistem harus mampu menambahkan menu restoran yang disukai oleh pelanggan
9	SA-F-009	Sistem harus mampu menampilkan daftar menu restoran yang disukai oleh pelanggan
10	SA-F-010	Sistem harus mampu menampilkan daftar menu restoran yang paling banyak disukai
11	SA-F-011	Sistem harus mampu mengeluarkan pelanggan restoran dari sistem
12	SMA-F-001	Sistem harus mampu menyediakan tempat untuk memasukkan tamu operator restoran ke sistem
13	SMA-F-002	Sistem harus mampu menampilkan daftar pesanan pelanggan
14	SMA-F-003	Sistem harus mampu mengubah status pesanan pelanggan
15	SMA-F-004	Sistem harus mampu menampilkan daftar kategori menu restoran
16	SMA-F-005	Sistem harus mampu mengubah kategori menu restoran
17	SMA-F-006	Sistem harus mampu menambah kategori menu restoran
18	SMA-F-007	Sistem harus mampu menghapus kategori menu restoran
19	SMA-F-008	Sistem harus mampu menampilkan daftar menu restoran
20	SMA-F-009	Sistem harus mampu mengubah menu restoran
21	SMA-F-010	Sistem harus mampu menambah menu restoran
22	SMA-F-011	Sistem harus mampu menghapus menu restoran
23	SMA-F-012	Sistem harus mampu mengubah status ketersediaan menu restoran

24	SMA-F-013	Sistem harus mampu mengatur ulang ketersediaan menu restoran
25	SMA-F-014	Sistem harus mampu mengubah profil restoran
26	SMA-F-015	Sistem harus mampu mengeluarkan operator restoran dari sistem

4.2.4 Kebutuhan Non-Fungsional Sistem

Kebutuhan non-fungsional adalah merupakan batasan fitur yang berfungsi sebagai sistem, batasan yang dimaksud adalah batasan dalam proses pengembangan, waktu dan batasan yang ditentukan oleh standar yang ada. Kebutuhan non-fungsional lebih diaplikasikan pada sistem secara keseluruhan, dibandingkan dengan pada fitur atau layanan spesifik dari sebuah sistem. Sehingga, kebutuhan non-fungsional seperti performa, keamanan, atau ketersediaan, biasanya akan menggambarkan karakteristik dari sebuah sistem secara keseluruhan (Vaduva, Baltac, Florescu, Floricica, & Jitaru, 1983). Kebutuhan non-fungsional dari sistem manajemen antrean pesanan menu restoran didaftarkan pada Tabel 4.5.

Tabel 4.5 Kebutuhan Non-Fungsional Sistem

No	Kode	Nama Kebutuhan	Deskripsi Kebutuhan
1	SA-NF-001	*Usability*	Kemudahan penggunaan. Pengukuran *Usability* berdasarkan kemampuan pengguna dalam menyelesaikan sebuah skenario menggunakan aplikasi ini. Dengan begitu, pengujian *usability* memiliki target untuk mendapatkan tingkat kepuasan responden yang memuaskan.

4.2.5 Use Case Diagram

Use case diagram merepresentasikan interaksi antara sistem dan lingkungannya. Interaksi yang digambarkan dari *use case diagram* mendeskripsikan kelakuan sistem dari sudut pandang aktor. Use case yang dibuat akan merepresentasikan seluruh interaksi yang mungkin terjadi yang akan dideskripsikan pada kebutuhan sistem. Aktor dari suatu proses, yang mana dapat menjadi aktor ataupun sistem lain, digambarkan dengan figur *stick* (Vaduva et al., 1983). *Use case diagram* dari sistem ini digambarkan pada Gambar **4.2** dan Gambar 4.3.

Gambar 4.2 Use case diagram aplikasi manajemen antrean pesanan menu restoran untuk pelanggan

Gambar 4.3 Use case diagram aplikasi manajemen antrean pesanan menu restoran untuk restoran

Aplikasi yang akan dibuat dari sistem akan dibagi menjadi dua aplikasi, yaitu aplikasi manajemen antrean pesanan menu restoran untuk pelanggan dan aplikasi manajemen antrean pesanan menu restoran untuk operator restoran. Aplikasi

manajemen antrean pesanan menu restoran akan mengatur pemesanan yang dilakukan oleh pelanggan restoran beserta seluruh informasi tentang restoran. Pada *use case diagram* Login Pelanggan, aktor yang terlibat adalah Tamu Pelanggan yang merepresentasikan pelanggan yang belum masuk ke dalam sistem, sehingga dikenal sebagai tamu dan tidak dapat mengakses fitur-fitur lain dari sistem kecuali Login Pelanggan. Setelah Tamu Pelanggan masuk ke dalam sistem menggunakan *login with google account*, status pengguna akan berubah menjadi Pelanggan. Kemudian aktor Pelanggan akan dapat mengakses seluruh fitur pada aplikasi aplikasi manajemen antrean pesanan menu restoran untuk pelanggan, yaitu mencakup Melakukan Pesanan Menu, Melihat Riwayat Pemesanan, Melihat Rekomendasi Menu Restoran, Melihat Daftar Menu Favorit, Menambahkan Menu Favorit, dan Logout Pelanggan.

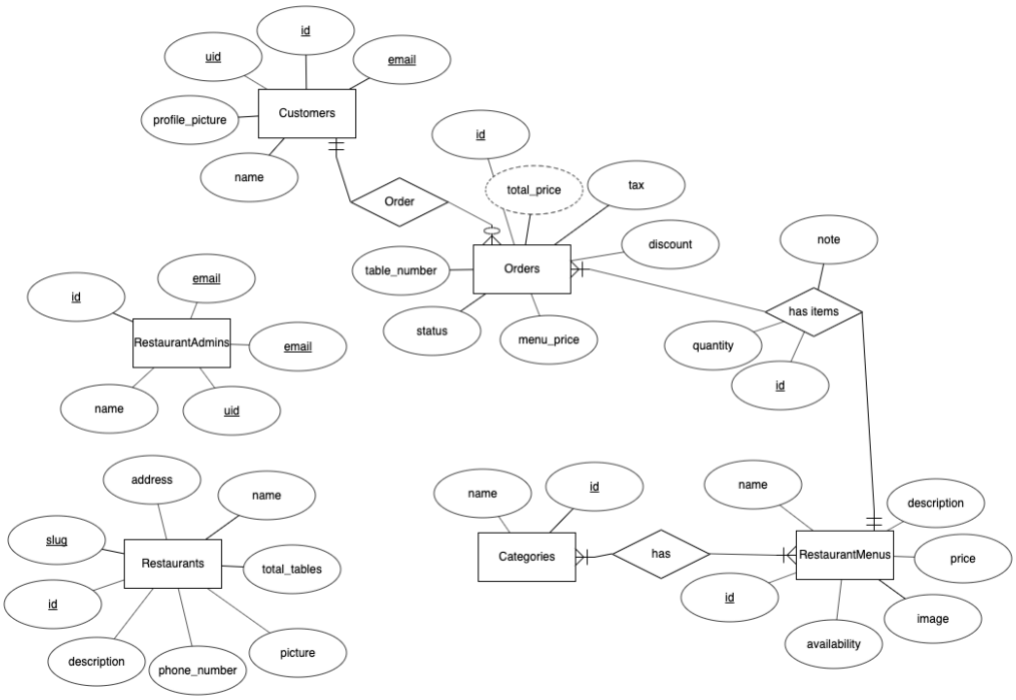
Pada *use case diagram* Login Operator Restoran, aktor yang terlibat adalah Tamu Operator yang merepresentasikan operator restoran yang belum masuk ke dalam sistem, sehingga dikenal sebagai tamu dan tidak dapat mengakses fitur-fitur lain dari sistem kecuali Login Operator Restoran. Setelah Tamu Operator Restoran masuk ke dalam sistem menggunakan *login with google account*, status pengguna akan berubah menjadi Operator Restoran. Kemudian aktor Operator Restoran akan dapat mengakses seluruh fitur pada aplikasi manajemen antrean pesanan menu restoran untuk pelanggan, yaitu mencakup Melihat Daftar Pesanan, Mengubah Pesanan, Menghapus Pesanan, Melihat Daftar Menu Restoran, Mengubah Menu Restoran, Menambah Menu Restoran, Menghapus Menu Restoran, Mengatur Stock Menu Restoran, Mengubah Profil Restoran, dan Logout Operator Restoran.

BAB 5 PENGEMBANGAN METODOLOGI KERANGKA KERJA SCRUM

Bab ini membahas tentang tahap implementasi dari pengembangan sistem dengan menerapkan metodologi kerangka kerja Scrum. Penerapan metodologi kerangka kerja Scrum untuk implementasi sistem dimulai dari penentuan *backlog product,* kemudian dilakukan pemecahan *backlog product* menjadi beberapa sprint, dan setelah itu dilanjutkan dengan pengerjaan *sprint*. Dalam pengerjaan *sprint,* akan dijelaskan hasil pengerjaan setiap *sprint* seperti Use case *scenario*, *sequence diagram*, *class diagram,* dan hasil evaluasi dan demonstrasi produk.

5.1 Perancangan Basis Data

Basis data adalah komponen yang dibutuhkan oleh sistem untuk menyimpan seluruh data yang dipakai dan diolah oleh sistem. Perancangan basis data dilakukan untuk merancang seluruh komponen dalam basis data yang akan dijadikan referensi untuk diimplementasikan pada tahap implementasi basis data. Perancangan basis data yang dibuat dalam penelitian ini mereferensi pada analisis yang telah dilakukan sebelumnya. Perancangan basis data dalam penelitian ini digambarkan melalui *Entity Relational Diagram* atau ERD pada Gambar 5.1.



Gambar 5.1 *Entity Relational Diagram (ERD)* sistem manajemen antrean pesanan menu restoran

Penjelasan dari *Entity Relational Diagram* sistem manajemen antrean pesanan menu restoran dapat dilihat pada Tabel 5.1. Entitas-entitas yang terdapat

dalam ERD memiliki hubungan atau relationship. Customer memiliki relationship 1:m terhadap Orders, Orders memiliki relationship n:m terhadap RestaurantMenus, dan RestaurantMenus memiliki relationship 1:m terhadap *Categories*.

Tabel 5.1 Penjelasan Perancangan Data

Nama Entitas	Attributes	Deskripsi
Customers	Id	Identifier entitas
	Uid	Identifier firebase google sign-in
	Email	Email pelanggan
	Profile_picture	Profile picture pelanggan
	Name	Nama pelanggan
Orders	Id	Identifier entitas
	Tax	Pajak pemesanan
	Menu_price	Harga menu yang dipesan
	Total_price	Total harga setelah dikurangi pajak
	Status	Status pemesanan
	Table_number	Nomor meja pelanggan
RestaurantMenus	Id	Identifier entitas
	Name	Nama menu
	Description	Deskripsi menu
	Image	Gambar menu
	Price	Harga menu
	Availability	Status ketersediaan menu
Categories	Id	Identifier entitas
	Name	Nama kategori
RestaurantAdmins	Id	Nama pelanggan
	Uid	Identifier firebase google sign-in
	Email	Email pelanggan
	Name	Nama pelanggan
Restaurants	Id	Identifier entitas
	Slug	Identifier sebagai ‘username’
	Description	Deskripsi restoran
	Phone_number	Nomor telepon restoran
	Name	Nama restoran
	Address	Alamat restoran
	Total_tables	Total meja restoran

	Picture	Banner restoran

5.2 Spesifikasi Lingkungan Sistem Implementasi

5.2.1 Spesifikasi Perangkat Keras

Spesifikasi pernagkat keras yang digunakan untuk pengembangkan perangkat lunak dijelaskan pada Tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Keras

Perangkat	MacBook Pro (13-inch, 2016, Four Thunderbolt 3 Ports)
Processor	2,9 GHz Intel Core i5
Ram	8 GB 2133 MHz LPDDR3
GPU	Intel Iris Graphics 550 1536 MB
Storage	C02TKAC3HF1P
OS	MacOS Mojave

5.2.2 Spesifikasi Perangkat Lunak

Spesifikasi pernagkat lunak yang digunakan untuk pengembangkan perangkat lunak dijelaskan pada Tabel 5.3.

Tabel 5.3 Spesifikasi Perangkat Keras

Editor Dokumentasi	Microsoft Word for Mac versi 16.31
Editor Perancangan	Draw.io dan erdplus.com
Editor Pemrograman	Visual Studio Code
Bahasa Pemrograman	Javascript
Perangkat Uji	Redmi Note 7

5.3 Batasan Implementasi

Batasan implementasi dalam pengimplementasian perangkat lunak adalah sebagai berikut:

1. Perangkat lunak *progressive website application* dijalankan pada lingkungan *browser* Chrome Version 78.0.3904.108 (Official Build) (64-bit)
2. Basis data yang digunakan untuk menyimpan data adalah mysql dan untuk menyimpan data yang bersifat *realtime* menggunakan *service realtime database* dari Firebase

5.4 Implementasi Basis Data

Implementasi penyimpanan data dilakukan dengan menggunakan Sequelize sebagai object-relational mapping (ORM) yang akan memetakan seluruh *entity*, *attribute,* dan *relational* yang telah digambarkan pada diagram Entity Relational Diagram sistem. Salah satu hasil implementasi basis data dari penggunaan ORM pada MySql dapat dilihat pada Tabel 5.4.

Tabel 5.4 Hasil implementasi basis data pada *entity* OrderItem

```
1   import Sequelize from 'sequelize'
2   import connection from './connection'
3
4   export default connection.define(
5     'RestaurantMenu',
6     {
7       name: {
8         type: Sequelize.STRING,
9         allowNull: false
10      },
11      description: {
12        type: Sequelize.STRING,
13        allowNull: true
14      },
15      price: {
16        type: Sequelize.INTEGER.UNSIGNED,
17        allowNull: false
18      },
19      image: {
20        type: Sequelize.STRING,
21        allowNull: false,
22        defaultValue: ''
23      },
24      availability: {
25        type: Sequelize.BOOLEAN,
26        allowNull: false,
27        defaultValue: true,
28      }
29    },
30    {
31      underscored: true,
32      timestamps: false
33    }
34  )
```

Implementasi *relational entity* dapat dilihat pada Tabel 5.5.

Tabel 5.5 Hasil implementasi *relational entity*

```
1   import connection from './connection'
2   import { events, DB_CONNECTED } from '../events'
3   import Customer from './Customer'
4   import Category from './Category'
5   import OrderItem from './OrderItem'
6   import RestaurantMenu from './RestaurantMenu'
7   import RestaurantAdmin from './RestaurantAdmin'
8   import Restaurant from './Restaurant'
```

```
import SocialMedia from './SocialMedia'
import './Favorite'
import './Upload'

// associate Restaurant with RestaurantMenu
Restaurant.hasMany(RestaurantMenu, { foreignKey:
'restaurant_id' })
RestaurantMenu.belongsTo(Restaurant, { foreignKey:
'restaurant_id' })

// associate RestaurantAdmin with Restaurant
Restaurant.belongsTo(RestaurantAdmin, { foreignKey:
'restaurant_admin_id' })
RestaurantAdmin.hasMany(Restaurant, { foreignKey:
'restaurant_admin_id' })

// associate Restaurant with Category
Restaurant.hasMany(Category, { foreignKey: 'restaurant_id'
})
Category.belongsTo(Restaurant, { foreignKey:
'restaurant_id' })

// associate RestaurantMenu with Category
Category.hasMany(RestaurantMenu, { foreignKey:
'category_id' })
RestaurantMenu.belongsTo(Category, { foreignKey:
'category_id' })

// associate RestaurantMenu with Customer as Favorite Menu
RestaurantMenu.belongsToMany(Customer, {
  through: 'Favorite',
  foreignKey: 'menu_id',
  as: 'Favorites',
})
Customer.belongsToMany(RestaurantMenu, {
  through: 'Favorite',
  foreignKey: 'customer_id',
  as: 'Favorites',
})

// associate OrderItem with RestaurantMenu
RestaurantMenu.hasMany(OrderItem, { foreignKey: 'menu_id'
})
OrderItem.belongsTo(RestaurantMenu, {
  foreignKey: 'menu_id',
  as: 'RestaurantMenus'
})

import { giveSeeds } from '../seeders'

let force = true

connection
  .sync({
    force
  })
  .then(async () => {
    console.log('database synchronized')

```

```
68      events.emit(DB_CONNECTED)
69      if (force) giveSeeds()
70    })
71    .catch(err => {
72      console.log(err)
73    })
74
      export default connection
```

Implementasi penyimpanan data yang bersifat *realtime database* menggunakan Firebase terdapat pada *entity* Orders. Hasil implementasi penyimpanan data *entity* Orders dapat dilihat pada Tabel 5.6.

Tabel 5.6 Implementasi *realtime database entity* Orders

```
1   {
2     "Orders" : {
3       "-LuVaTxnhJk1CpfPZJrt" : {
4         "created_at" : 1574652479577,
5         "customer" : {
6           "email" : "yahya.sahaja2@gmail.com",
7           "id" : 1,
8           "name" : "Yahya Sahaja",
9           "profile_picture" :
10  "https://lh3.googleusercontent.com/a-
11  /AAuE7mAJp1I9Q0CHL_E3yIT9f7zQxOI8iqvJsbFSTe0f_Q",
12          "uid" : "ytXc5lD8T8XtwowFyTCtZsVzw7E2"
13        },
14        "customer_id" : "1",
15        "discount" : 0,
16        "id" : "-LuVaTxnhJk1CpfPZJrt",
17        "menu_price" : 75000,
18        "order_items" : [ {
19          "menu_id" : "17",
20          "note" : "",
21          "quantity" : 2,
22          "restaurant_menu" : {
23            "availability" : true,
24            "category_id" : 8,
25            "description" : "Gurih Manis/Pedas",
26            "id" : 17,
27            "image" :
28  "/images/wongsolo/MenuPaketAyamBakar.png",
29            "name" : "Ayam Bakar",
30            "price" : 25000,
31            "restaurant_id" : 4
32          }
33        }
34      }
35    }
36  }
```

5.5 Backlog Produk

Pembagian *backlog* produk memiliki tujuan untuk membagi *backlog* yang sudah dibuat sebelumnya ke dalam beberapa *sprint*. Urutan *backlog* yang akan

dimasukkan dalam urutan *sprint* disesuaikan dengan prioritas fitur pada sebuah backlog, sehingga *backlog-backlog* yang lebih penting akan direncanakan pada *sprint* yang lebih dulu. Oleh karena itu, hasil pemecahan *backlog* untuk setiap *sprint* akan dijelaskan pada Tabel 5.7, Tabel 5.8, dan Tabel 5.9.

Tabel 5.7 Backlog Produk untuk Sprint Pertama

No	Kode	Deskripsi Kebutuhan
1	SA-F-001	Sistem harus mampu menyediakan tempat untuk mendaftarkan dan memasukkan tamu pelanggan ke sistem melalui akun Google pelanggan
2	SA-F-002	Sistem harus mampu menyediakan pengambilan gambar melalui kamera untuk pindai Kode QR pada meja restoran
3	SA-F-003	Sistem harus mampu menampilkan daftar meu restoran yangn telah dipidai pada meja restoran
4	SA-F-004	Sistem harus mampu menyediakan ringkasan pemesanan dan menampilkan harga
5	SA-F-005	Sistem harus mampu membuat pesanan
6	SA-F-006	Sistem harus mampu menampilkan daftar riwayat pemesanan yang telah dilakukan oleh pelanggan
7	SA-F-007	Sistem harus mampu mengubah pesanan saat status pesanan masih belum diproses
8	SA-F-008	Sistem harus mampu menambahkan menu restoran yang disukai oleh pelanggan
9	SA-F-009	Sistem harus mampu menampilkan daftar menu restoran yang disukai oleh pelanggan
10	SA-F-010	Sistem harus mampu menampilkan daftar menu restoran yang paling banyak disukai
11	SA-F-011	Sistem harus mampu mengeluarkan pelanggan restoran dari sistem

Tabel 5.8 Backlog Produk untuk Sprint Kedua

No	Kode	Deskripsi Kebutuhan
1	SMA-F-001	Sistem harus mampu menyediakan tempat untuk memasukkan tamu operator restoran ke sistem
2	SMA-F-002	Sistem harus mampu menampilkan daftar pesanan pelanggan
3	SMA-F-003	Sistem harus mampu mengubah status pesanan pelanggan
4	SMA-F-004	Sistem harus mampu menampilkan daftar kategori menu restoran
5	SMA-F-005	Sistem harus mampu mengubah kategori menu restoran
6	SMA-F-006	Sistem harus mampu menambah kategori menu restoran
7	SMA-F-007	Sistem harus mampu menghapus kategori menu restoran

8	SMA-F-008	Sistem harus mampu menampilkan daftar menu restoran
9	SMA-F-009	Sistem harus mampu mengubah menu restoran
10	SMA-F-010	Sistem harus mampu menambah menu restoran
11	SMA-F-011	Sistem harus mampu menghapus menu restoran
12	SMA-F-012	Sistem harus mampu mengubah ketersediaan menu restoran
13	SMA-F-013	Sistem harus mampu mengatur ulang ketersediaan menu restoran

Tabel 5.9 Backlog Produk untuk Sprint Ketiga

No	Kode	Deskripsi Kebutuhan
1	SMA-F-014	Sistem harus mampu mengubah profil restoran
2	SMA-F-015	Sistem harus mampu mengeluarkan operator restoran dari sistem

5.6 Perencanaan *Sprint*

Perancangan Durasi Sprint adalah perancangan untuk menentukan seberapa lama durasi sprint akan dilakukan dalam satu sprint. Untuk menentukan durasi sprint dalam metode *Scrum*, dasar yang digunakan adalah waktu, bukan didasarkan kepada kebutuhan fungsinoal sistem. Maksimal dari sebuah durasi *sprint* adalah sebanyak 4 minggu, hal ini dikarenakan jika durasi *sprint* terlalu lama, prespektif pada sebuah hal yang akan dibangun dapat berubah, meningkatnya kompleksitas, dan juga risiko yang meningkat (Schwaber & Sutherland, 2017).

Durasi sprint yang dilakukan dalam penelitian ini adalah selama 2 minggu dalam satu *sprint*. Jumlah *sprint* yang akan dilakukan sebanyak 3 kali, sehingga total durasi *sprint* yang dilakukan adalah selama 6 minggu. Durasi selama 2 minggu dalam satu *sprint* ini dirasa cukup oleh peneliti untuk digunakan dalam penelitian ini. Hal ini dikarenakan terdapat beberapa keuntungan, diantaranya:

1. Semakin pendek durasi *sprint,* maka akan cepat umpan balik yang akan diterima, sehingga dapat dilakukan perbaikan sistem lebih cepat.
2. Membantu peneliti untuk menjadi lebih focus.
3. Respon perbaikan dari umpan balik yang diterima oleh calon pengguna akan menjadi lebih cepat melalui rilis produk *incremental* berikutnya

5.7 Pengerjaan *Sprint*

Dalam pengerjaan *sprint,* akan dijelaskan hasil pengerjaan setiap *sprint* seperti *use case scenario*, *sequence diagram*, *class diagram,* perancangan antarmuka, implementasi kode program, implementasi antar muka dan hasil evaluasi dan demonstrasi produk.

5.7.1 *Sprint* Pertama

5.7.1.1 Use Case Scenario

Use Case Scenario adalah skenario apa saja yang dapat terjadi pada sebuah Use Case. *Use Case Scenario* dari sistem manajemen antrean pesanan menu restoran pada *sprint* pertama dijabarkan pada Tabel 5.10 sampai dengan Tabel 5.17.

Tabel 5.10 Skenario *Use Case* Login Pelanggan

Use Case	Login Pelanggan
Kode terkait kebutuhan	SA-F-001
Actor	Tamu pelanggan
Target	Pelanggan terdaftar atau masuk ke dalam sistem
Pre-Condition	Pelanggan belum terdaftar atau masuk ke dalam sistem
Main Flow	1. Sistem menampilkan halaman *Login* 2. Tamu pelanggan menekan tombol *Sign In With Google.* 3. Sistem menampilkan akun Google yang tersedia. 4. Tamu memilih akun Google atau menambahkan akun Google baru. 5. Sistem memproses otentikasi pelanggan.
Alternative Flow	-
Post Condition	Pelang Tamu pelanggan berhasil masuk ke dalam sistem dan status tamu berubah menjadi pelanggan

Tabel 5.11 Skenario *Use Case* Melakukan Pemesanan Menu

Use Case	Melakukan Pemesanan Menu
Kode terkait kebutuhan	SA-F-002, SA-F-003, SA-F-004, dan SA-F-005
Actor	Pelanggan
Target	Pelanggan melakukan pemesanan menu restoran
Pre-Condition	Pelanggan sudah berada di halaman *home*
Main Flow	1. Pelanggan menekan tombol *scan* 2. Sistem menampilkan halaman *Scan* Kode QR 3. Pelanggan melakukan *scan* pada Kode QR meja 4. Sistem menampilkan halaman restoran 5. Pelanggan memilih makanan yang hendak dipesan dengan menekan tombol *add* sebanyak jumlah yang

	diinginkan dan menekan tombol *checkout* apabila pesanan sudah sesuai 6. Sistem menampilkan halaman *checkout* 7. Pelanggan menekan tombol *order now* apabila pesanannya telah tepat 8. Sistem menampilkan pesan bahwa pesanan telah berhasil dibuat
Alternative Flow	1. Apabila stock habis saat data menu sudah dibuat dan telah memilih menu, sistem akan menampilkan pesan *error* bahwa stok menu sudah habis
Post Condition	Pesanan menu telah dilakukan dan dibuat

Tabel 5.12 Skenario *Use Case* Melihat Riwayat Pemesanan Menu

Use Case	Melihat Riwayat Pemesanan Menu
Kode kebutuhan terkait	SA-F-006
Actor	Pelanggan
Target	Pelanggan dapat melihat daftar pemesanan yang pernah dilakukan dalam aplikasi manajemen antrean pesanan menu restoran untuk pelanggan
Pre-Condition	Pelanggan sudah berada di halaman *home*
Main Flow	1. Pelanggan memilih halaman *orders* 2. Sistem menampilkan halaman *orders,* terdapat dua bagian yaitu bagian pesanan yang masih dalam proses dan yang telah selesai.
Alternative Flow	-
Post Condition	Pelanggan berhasil mengakses daftar riwayat pesanan dalam halaman *orders*.

Tabel 5.13 Skenario *Use Case* Mengubah Pesanan

Use Case	Melihat Riwayat Pemesanan Menu
Kode kebutuhan terkait	SA-F-006
Actor	Pelanggan
Target	Pelanggan dapat mengubah pesanan ketika status pesanan adalah "INLINE"
Pre-Condition	Pelanggan sudah berada di halaman *transaction page*

Main Flow	1. Pelanggan memilih menekan tombol “Update Order” 2. Sistem menampilkan halaman *update order* 3. Pelanggan memilih menu yang ingin dipesan dan menekan tombol “Update Order” 4. Sistem mengubah pesanan pelanggan
Alternative Flow	-
Post Condition	Pelanggan berhasil mengubah pesanan ketika status pesanan adalah “INLINE”

Tabel 5.14 Skenario *Use Case* Menambahkan Menu Favorit

Use Case	Menambahkan Menu Favorit
Kode kebutuhan terkait	SA-F-007
Actor	Pelanggan
Target	Pelanggan dapat menambahkan menu favorit suatu restoran ke dalam daftar menu favorit
Pre-Condition	Pelanggan telah berada di halaman restoran
Main Flow	1. Pelanggan menekan tombol favorit yang berbentuk hati pada menu kesukaannya. 2. Sistem menambahkan menu menjadi menu favorit pelanggan
Alternative Flow	1. Apabila menu restoran sudah dihapus sebelum menu favorit yang diinginkan pelanggan dipilih, sistem akan menampilkan pesan *error* bahwa menu restoran sudah tidak tersedia
Post Condition	Menu berhasil ditambahkan ke dalam daftar menu favorit.

Tabel 5.15 Skenario *Use Case* Melihat Daftar Menu Favorit

Use Case	Menambahkan Menu Favorit
Kode kebutuhan terkait	SA-F-008
Actor	Pelanggan
Target	Pelanggan dapat melihat daftar menu favorit yang pernah difavoritkan.
Pre-Condition	Pelanggan telah berada di halaman restoran
Main Flow	1. Pelanggan memilih halaman *favorite.* 2. Sistem menampilkan halaman *favorite* dan daftar menu favorit.

Alternative Flow	-
Post Condition	Pelanggan melihat daftar menu favorit mereka

Tabel 5.16 Skenario *Use Case* Melihat Rekomendasi Menu Restoran

Use Case	Melihat Rekomendasi Menu Restoran
Kode kebutuhan terkait	SA-F-009
Actor	Pelanggan
Target	Pelanggan dapat melihat rekomendasi menu restoran
Pre-Condition	Pelanggan sudah berada di halaman *home*
Main Flow	1. Pelanggan memilih halaman *home* 2. Sistem menampilkan halaman *home* yang berisikan list menu restoran sesuai dengan lokasi pengguna
Alternative Flow	-
Post Condition	Pelanggan melihat rekomendasi menu restoran.

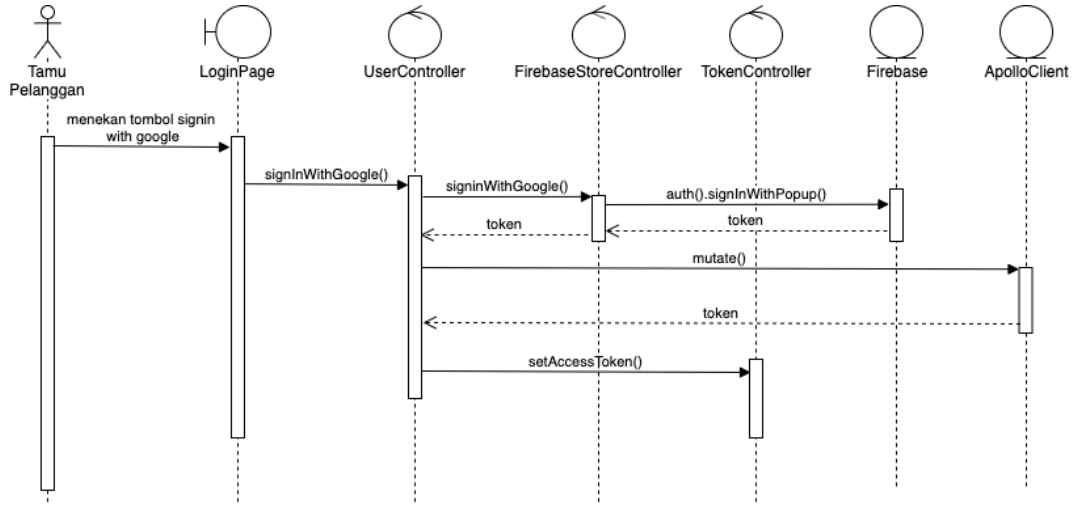
Tabel 5.17 Skenario *Use Case* Logout Pelanggan

Use Case	Logout Pelanggan
Kode kebutuhan terkait	SA-F-010
Actor	Pelanggan
Target	Otentikasi pelanggan dapat keluar dari sistem.
Pre-Condition	Pelanggan sudah berada di halaman *home*
Main Flow	1. Pelanggan memilih halaman *more* 2. Sistem menampilkan halaman *more* 3. Pelanggan menekan tombol *Logout* 4. Sistem menampilkan dialog konfirmasi untuk *logout* 5. Pelanggan memilih tombol *Logout* 6. Sistem menampilkan halaman login
Alternative Flow	1. Jika Pelanggan memilih tombol cancel logout pada dialog, dialog tertutup kembali
Post Condition	Otentikasi pelanggan telah keluar dari sistem.

5.7.1.2 Sequence Diagram

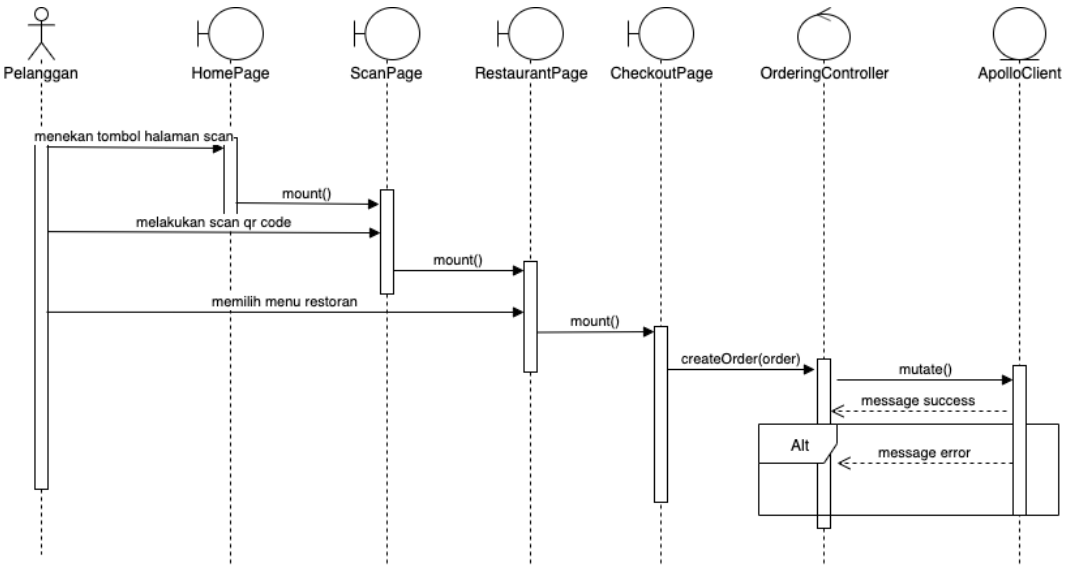
Sequence diagram menggambarkan urutas proses yang terjadi di dalam sistem untuk mencapai tujuan dari kebutuhan fungsionalitas sistem. Daftar *sequence*

diagram dalam sistem manajemen antrean pesanan menu restoran di *sprint* pertama dapat dilihat pada Gambar 5.2 sampai Gambar 5.9.



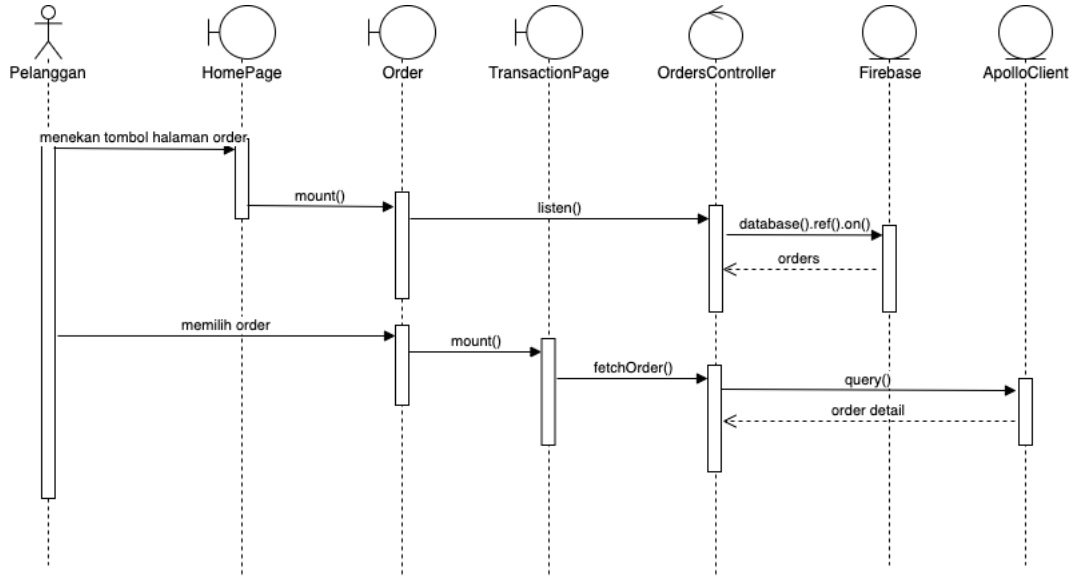
Gambar 5.2 *Sequence Diagram* Use Case Login Pelanggan

Pada *sequence diagram* Login Pelanggan pada Gambar 5.2 dijelaskan bahwa aktor tamu pelanggan menekan tombol *signin with google* pada halaman LoginPage. Kemudian aktivitas tersebut akan memicu pemanggilan fungsi `signinWIthGoogle` pada UserController. Setelah pelanggan berhasil *login* dengan *credentials* akun google pelanggan, maka dilakukan *request* ke *server* untuk melakukan *signin*. Nilai kembalian dari request *login* adalah *token* yang digunakan untuk *authorization* pengguna saat melakukan *request* untuk mengakses *endpoints* pada *server*. Kemudian dilakukan pemanggilan fungsi `setAccessToken()` pada TokenController untuk menyimpan data pada *local storage browser* dan mengatur *default header* dari *service request data axios*.



Gambar 5.3 *Sequence Diagram* Use Case Melakukan Pemesanan Menu

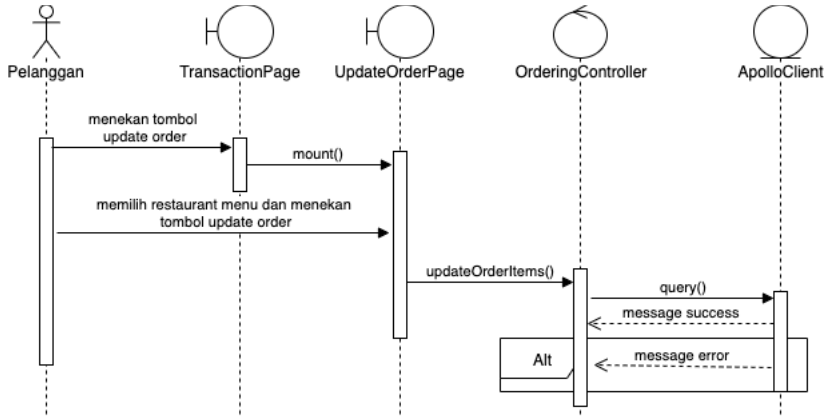
Pada *sequence diagram* Melakukan Pemesanan Menu pada Gambar 5.3 dijelaskan bahwa aktor pelanggan menekan tombol halaman scan pada LoginPage. Kemudian aktivitas tersebut memicu *mounting component* pada halaman ScanPage, sehingga halaman ScanPage ditampilkan oleh sistem. Kemudian pelanggan melakukan *scanning* Kode QR pada meja yang ada di restoran yang dipilih oleh pelanggan. Kemudian dilakukan *mounting component* pada halaman RestaurantPage. Setelah RestaurantPage ditampilkan oleh sistem, pelanggan memilih menu restoran. Setelah memilih menu restoran dan menekan tombol *Checkout*, maka dilakukan *mounting component* pada halaman *CheckoutPage*. Setelah pelanggan memeriksa pesanannya dan mengonfirmasi kebenaran pesanan yang dipilih, maka pelanggan menekan tombol *Order*. Aktivitas tersebut memicu pemanggilan fungsi `createOrder` pada *OrderingController*. Selanjutnya dilakukan *request* ke ApolloClient untuk melakukan *create order*. Setelah berhasil maka akan mendapatkan kembalian data berupa message *success.* Namun apabila terdapat *error*, maka akan mendapatkan kembalian data berupa *message error*.



Gambar 5.4 *Sequence Diagram* Use Case Melihat Riwayat Pemesanan Menu

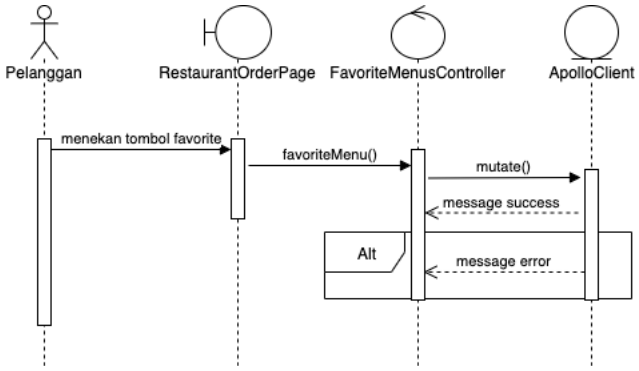
Pada *sequence diagram* Melihat Riwayat Pemesanan Menu pada Gambar 5.4 dijelaskan bahwa aktor Pelanggan menekan tombol halaman order pada halaman HomePage. Kemudian dilakukan *mounting component* halaman OrderPage, setelah itu memanggil fungsi `fetchOrders()` pada OrdersController. Fungsi tersebut kemudian melakukan *request orders* ke API dan memberikan nilai kembalian berupa *orders*. Kemudian data yang didapat akan disimpan dalam *observable property* yang membuatnya menjadi *live data* sehingga segala *observable component* seperti OrdersPage akan otomatis melakukan *re-render* ketika *observable property orders* berubah. Setelah itu Pelanggan memilih riwayat order yang ingin dilihat lebih detail. Aktivitas tersebut akan memicu *mounting component* pada TransactionPage. Kemudian dilakukan pemanggilan fungsi

`fetchOrder()` pada OrdersController untuk melakukan *request order detail* ke ApolloClient. Kemudian didapatkan nilai kembalian berupa *order detail* yang datanya disimpan di dalam *observable property order*.



Gambar 5.5 *Sequence Diagram* Use Case Mengubah Pesanan

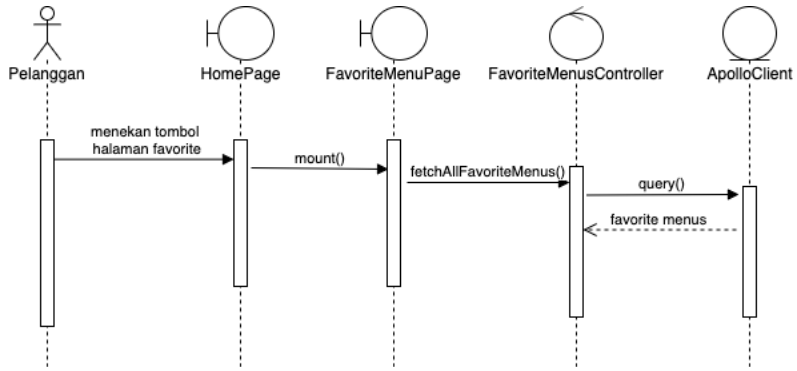
Pada *sequence diagram* Mengubah Pesanan pada Gambar 5.5 dijelaskan bahwa aktor Pelanggan menekan tombol update order pada halaman TransactionPage. Tombol update order ini akan muncul hanya ketika satus pesanan masih berupa *PROCESS*. Kemudian aktivitas tersebut akan memicu *mounting component* halaman UpdateOrderPage. Setelah itu Pelanggan memilih menu restoran yang sesuai dengan perubahan yang diinginkan oleh Pelanggan dan kemudian menekan tombol Update Order. Aktivitas tersebut akan memicu *request update order items* yang akan mengembalikan *message success* jika request berhasil dilakukan, namun apabila gagal, maka akan mengembalikan *message error*.



Gambar 5.6 *Sequence Diagram* Use Case Menambah Menu *Favorite*

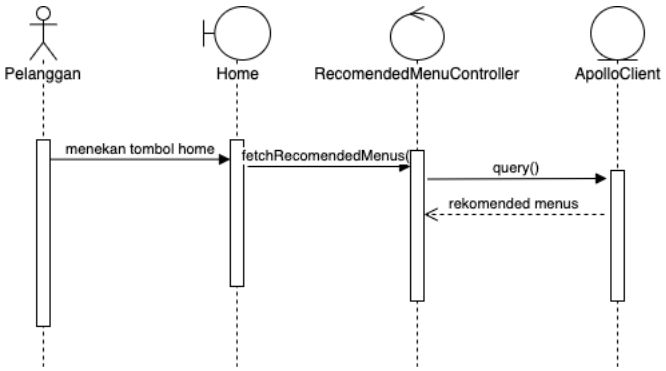
Pada *sequence diagram* Menambah Menu *Favorite* pada Gambar 5.6 dijelaskan bahwa aktor Pelanggan menekan tombol *favorite* berupa gambar hati pada menu restoran yang dipilih pelanggan di RestaurantOrderPage. Kemudian dilakukan pemanggilan fungsi `favoriteMenu()` pada FavoriteMenusController.

Aktivitas tersebut memicu *request favorite menu* ke Sans API, jika sukses akan mendapatkan pesan sukses, dan apabila gagal akan mendapatkan pesan *error*.



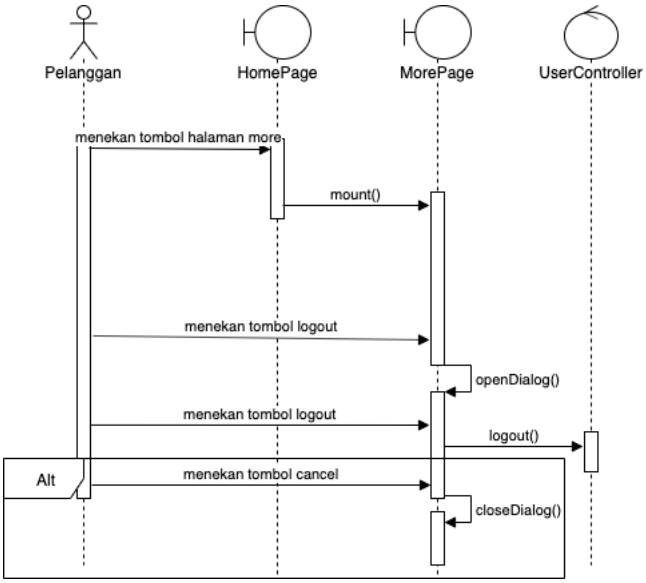
Gambar 5.7 *Sequence Diagram* Use Case Menambah Menu *Favorite*

Pada *sequence diagram* Menambah Menu *Favorite* pada Gambar 5.7 dijelaskan bahwa aktor Pelanggan menekan tombol *favorite* berupa gambar hati pada menu restoran yang dipilih pelanggan di RestaurantOrderPage. Kemudian dilakukan pemanggilan fungsi `favoriteMenu()` pada FavoriteMenusController. Aktivitas tersebut memicu *request favorite menu* ke Sans API, jika sukses akan mendapatkan pesan sukses, dan apabila gagal akan mendapatkan pesan *error*.



Gambar 5.8 *Sequence Diagram* Use Case Melihat Daftar Menu Rekomendasi

Pada *sequence diagram* Melihat Daftar Menu Rekomendasi pada Gambar 5.8 dijelaskan bahwa aktor Pelanggan menekan tombol home pada navigasi. Kemudian dilakukan pemanggilan fungsi `fetchAllRecomendedMenus()` pada RecomendedMenuController. Aktivitas tersebut akan memicu *request recommended menus*, dan mendapatkan nilai kembalian berupa *recommended menus* dan data tersebut disimpan dalam *observable property*.

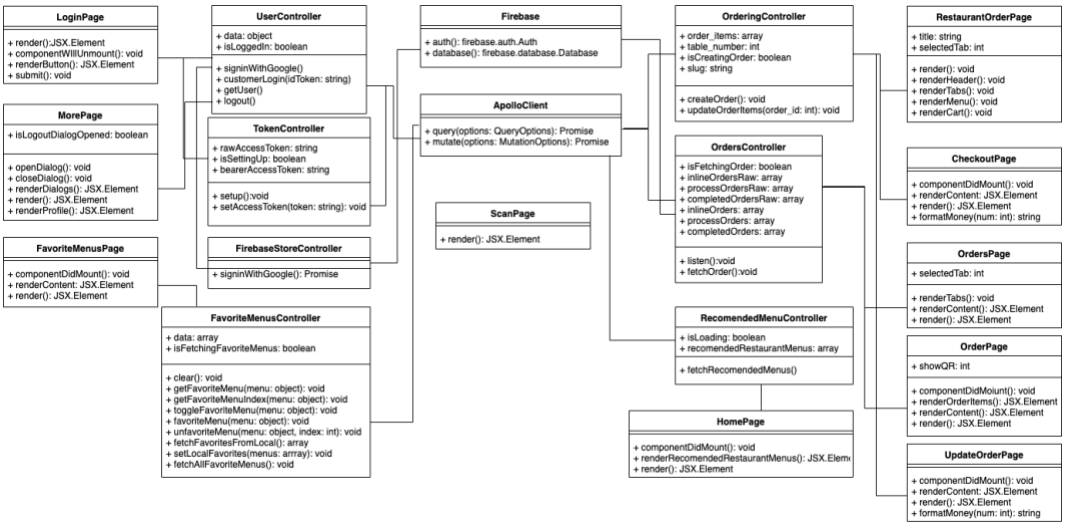


Gambar 5.9 *Sequence Diagram* Use Case Logout Pelanggan

Pada *sequence diagram* Logout Pelanggan pada Gambar 5.9 dijelaskan bahwa aktor Pelanggan menekan tombol more pada navigasi. Kemudian dilakukan *mounting component* MorePage. Setelah itu pelanggan menekan tombol logout pada halaman MorePage. Kemudian dilakukan *self-call* `openDialog()` sehingga sistem menampilkan dialog konfirmasi untuk logout. Kemudian pelanggan menekan tombol logout pada dialog. Aktivitas tersebut memicu pemanggilan fungsi `logout()` pada UserController untuk menghapus seluruh data otentikasi user pada *localStorage* browser. Apabila pelanggan menekan tombol *cancel*, maka dilakukan *self-call* `closeDialog()`.

5.7.1.3 Class Diagram

Class diagram digunakan untuk menggambarkan struktur statis dari sebuah sistem. Berdasarkan sequence diagram yang telah dibuat sebelumnya, maka akan didapatkan gambaran kelas-kelas beserta relasi, atribut, dan method yang ada di dalamnya. Class diagram dari *sprint* pertama ini digambarkan pada Gambar 5.10.

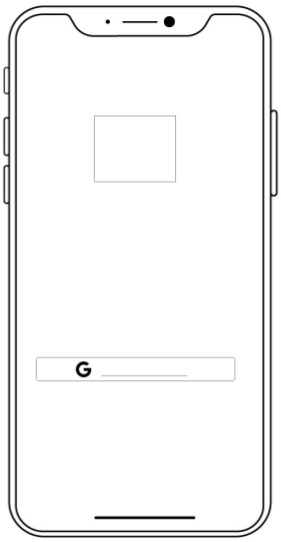


Gambar 5.10 *Class Diagram* sistem pada *sprint* pertama

5.7.1.4 Perancangan Antarmuka

Perancangan antarmuka digunakan untuk mempermudah proses implementasi GUI pada perangkat lunak dengan gambaran mengenai tata letak dan tampilan pada aplikasi. Pembuatan perancangan antarmuka dapat dibuat dalam bentuk *wireframe* menggunakan Adobe XD. *Wireframe* yang dibuat ditunjukkan pada Gambar 5.11 sampai Gambar 5.20.

(a) Perancangan Antarmuka *Sign-in*

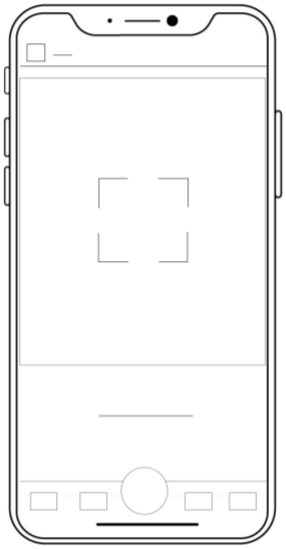


Gambar 5.11 *Wireframe Sign-in*

Pada Gambar 5.11 menunjukkan halaman *sign-in*. Halaman *sign-in* adalah halaman untuk memasukkan pengguna ke dalam sistem. Pada halaman *sign-in*, terdapat logo dan satu tombol bertuliskan “Sign in with Google”. Ketika tombol “Sign in with Google” ditekan oleh pengguna, maka sistem akan menampilkan

google authentication untuk memasukkan pengguna melalui akun *Google* pengguna.

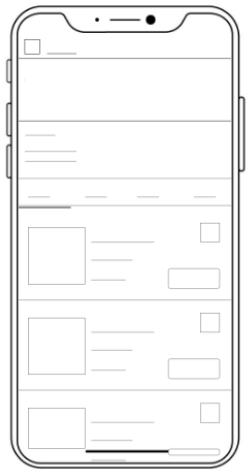
(b) **Perancangan Antarmuka *Scan***



Gambar 5.12 *Wireframe Scan*

Pada Gambar 5.11 menunjukkan halaman *scan*. Halaman *scan* adalah halaman untuk melakukan *scanning* kode QR yang terdapat pada meja restoran. Pada halaman *scan*, terdapat terdapat kamera dan *bottom navigation* yang memiliki tiga menu *home, orders, scan, favorite menu,* dan *account*. Ketika perangkat bergerak pelanggan diarahkan pada kode QR di meja pelanggan, maka sistem akan menampilkan halaman *Choose Restaurant Menu* sesuai dengan menu restoran dan nomor meja yang pelanggan pilih.

(c) **Perancangan Antarmuka *Choose Restaurant Menu***

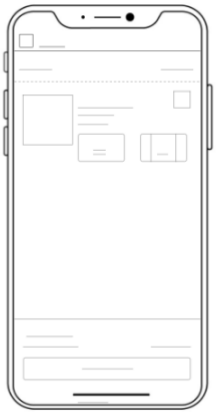


Gambar 5.13 *Wireframe Choose Restaurant Menu*

Pada Gambar 5.13 menunjukkan halaman *choose restaurant menu*. Halaman *choose restaurant menu* adalah halaman untuk melakukan pemilihan menu yang

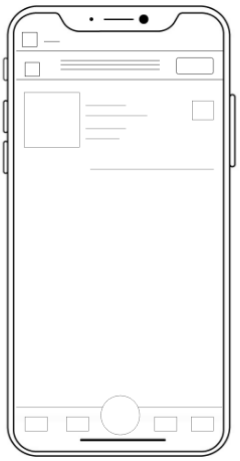
tersedia pada restoran. Pada halaman *choose restaurant menu*, terdapat terdapat deskripsi restoran, *menu categories* yang digambarkan dalam bentuk *tab bar*, dan pilihan menu dari setiap kategori dalam bentuk *card*. Kemudian dalam *menu card* terdapat gambar, nama, deskripsi, dan harga menu, beserta tombol untuk like, menambah, dan mengurangi menu yang diinginkan pelanggan.

(d) Perancangan Antarmuka *Checkout*



Gambar 5.14 *Wireframe Checkout*

Pada Gambar 5.14 menunjukkan halaman *checkout*. Halaman *choose checkout* adalah halaman untuk melakukan *review* pemesanan yang meliputi menu yang dipesan beserta estimasi harga. Pada halaman *checkout*, terdapat terdapat *menu card* yang menampilkan menu yang sudah dipilih pelanggan dari halaman *choose restaurant menu* sebelumnya. Selain itu, terdapat estimasi harga dan tombol "Order" untuk melakukan pemesanan menu restoran dan memasukkan pelanggan ke dalam antrean pemesanan menu restoran.

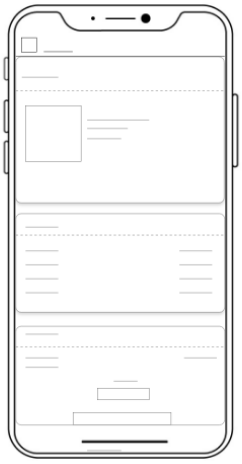


Gambar 5.15 *Wireframe Orders*

(e) **Perancangan Antarmuka *Orders***

Pada Gambar 5.15 menunjukkan halaman *orders*. Halaman *orders* adalah halaman untuk melihat riwayat pemesanan yang dilakukan oleh pelanggan. Pada halaman *orders*, terdapat daftar pesanan pelanggan yang ditampilkan dalam bentuk *card*. Ketika salah satu *order card* diklik, maka sistem akan menunjukkan halaman *Order Detail* yang berisi detail pemesanan.

(f) **Perancangan Antarmuka *Order Detail***



Gambar 5.16 *Wireframe Order Detail*

Pada Gambar 5.16 menunjukkan halaman *order detail*. Halaman *order detail* adalah halaman untuk melihat detail pesanan pelanggan. Pada halaman *order detail*, terdapat detail pemesanan yang dilakukan pelanggan yang meliputi data-data restoran, detil harga, status pemesanan, dan menu yang dipesan. Selain itu, jika status pesanan masih berstatus “INLINE”, maka akan terdapat tombol untuk melakukan *update* pada *order.*

(g) **Perancangan Antarmuka *Update Order***

Gambar 5.17 *Wireframe Update Order*

Pada Gambar 5.17 menunjukkan halaman *update order*. Halaman *update eorder* adalah halaman untuk melakukan perubahan pada pesanan yang telah

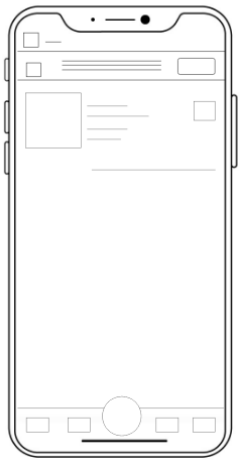
dilakukan pelanggan, jika status pesanan masih “INLINE”. Padahalaman *update order,* terdapat pilihan seluruh menu yang ada di restoran berupa *menu card*. Dalam *menu card* terdapat tombol untuk menambahkan dan mengurangi kuantitas dan catatan. Kemudian terdapat tombol *“Update Order”* untuk melakukan perubahan pada pesanan.

(h) **Perancangan Antarmuka *Home***

Gambar 5.18 *Wireframe Home*

Pada Gambar 5.18 menunjukkan halaman *home*. Halaman *home* adalah halaman untuk melihat urutan menu-menu pada restoran yang memiliki jumlah like terbanyak dari pelanggan. Pada halaman *home*, terdapat terdapat *card* yang menampilkan nama makanan dan gambar menu restoran yang sudah diurutkan sesuai dengan banyaknya jumlah like pada menu restoran tersebut.

(i) **Perancangan Antarmuka *Favorite Menu***

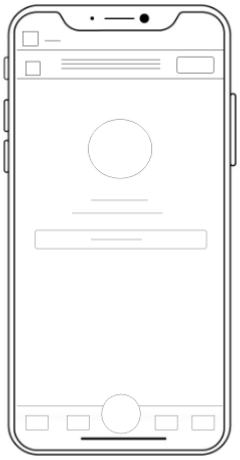


Gambar 5.19 *Wireframe Favorite Menu*

Pada Gambar 5.19 menunjukkan halaman *favorite menu*. Halaman *favorite menu* adalah halaman untuk melihat daftar menu yang telah disukai oleh

pelanggan. Pada halaman *favorite menu*, terdapat terdapat daftar menu berupa *card* yang menampilkan gambar menu, nama menu, deskripsi menu, harga menu, dan tombol like. Jika tombol like *diklik*, maka menu tersebut akan dihapus dari daftar *menu favorite*.

(j) **Perancangan Antarmuka *Account***



Gambar 5.20 *Wireframe Account*

Pada Gambar 5.20 menunjukkan halaman account. Halaman *account* adalah halaman untuk melihat profil pelanggan yang telah masuk ke dalam sistem. Pada halaman *account*, terdapat terdapat tombol *logout* untuk mengeluarkan pelanggan dari sistem.

5.7.1.5 Perancangan Algoritme

Pada bagian ini akan dilakukan perancangan algoritme untuk mendapatkan gambaran langkah-langkah dari suatu proses. Salah satu perancangan algoritme *create order* dapat dilihat pada Tabel 5.18.

Tabel 5.18 Perancangan algoritme *create order*

```
1  Begin
2    Deklarasi variable untuk input request
3    Request mutation graphql dengan input request
4    Catch error request
5    Begin
6      Show error message
7    End
8  end
```

Algoritme *create order* merupakan fungsi untuk membuat pesanan yang dipilih oleh pelanggan restoran. Diawali dengan mendeklarasikan seluruh variable yang menyesuaikan dengan *input* pada *field* yang dibutuhkan untuk melakukan *request.* Kemudian dilakukan *request* ke *server* dengan menginputkan data

variable yang telah dideklarasikan sebelumnya. Setelah itu jika terdapat *error*, maka pesan *error* akan ditampilkan oleh sistem.

5.7.1.6 Implementasi Kode Program

Pada bagian ini akan dilakukan dilakukan implementasi dari algoritme yang telah dirancang sebelumnya. Bahasa pemrograman yang digunakan untuk melakukan implementasi pada sistem ini menggunakan bahasa pemrograman ECMAscript 2015 atau ES6. Salah satu implementasi kode program *create order* dapat dilihat pada Tabel 5.19.

Tabel 5.19 Implementasi kode program *create order*

```
@action
  async createOrder() {
    let {
      restaurant_id,
      order_items,
      table_number,
      customer_id,
    } = this

    let orderItems = toJS(order_items)
    orderItems.forEach(d => delete d.menu)

    let variables = {
      restaurant_id,
      order_items: orderItems,
      table_number,
      customer_id,
    }

    try {
      this.isCreatingOrder = true
      overlayLoading.show()
      let {
        data: {
          createOrder: order
        }
      } = await client.mutate({
        mutation: createOrderMutation,
        variables
      })

      this.isCreatingOrder = false
      overlayLoading.hide()
      snackbar.show('Order created!')
      return order
    } catch (err) {
      this.isCreatingOrder = false
      overlayLoading.hide()
      snackbar.show(err.message)
      console.log('ERROR WHILE CREATING ORDER',
err.message)
    }
  }
```

Pembahasan dari kode program diatas adalah sebagai berikut:

1. Baris 3 – 18: Deklarasi variable dengan menyesuaikan *field name* yang dibutuhkan pada *mutation graphql* untuk *create order*.
2. Baris 23 – 30: *Request mutation graphql* dengan menginputkan *variable* yang telah dideklarasikan sebelumnya, kemudian dilakukan *destructuring* untuk mendapatkan hasil dari *request*.
3. Baris 37 – 40: *error handling* dengan menampilkan pesan *error* ketika mendapatkan *error*

5.7.1.7 Implementasi Antarmuka

Implementasi antarmuka berisi hasil implementasi dari perancangan antarmuka yang telah dibuat sebelumnya. Implementasi antarmuka dibuat dengan menggunakan *library* React. Hasil dari implementasi antarmuka dapat dilihat pada Gambar 5.21 sampai Gambar 5.30.

(a) **Hasil Implementasi Antarmuka *Sign-in***

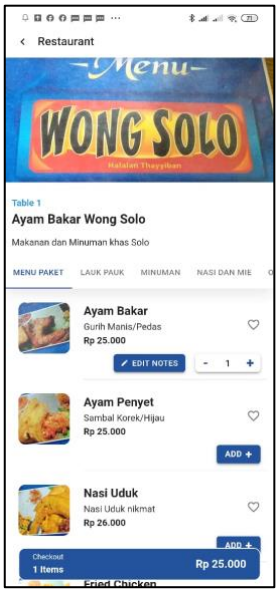
Gambar 5.21 Hasil Implementasi Antarmuka *Sign-in*

Pada Gambar 5.21 menunjukkan halaman *sign-in*. Halaman *sign-in* adalah halaman untuk memasukkan pengguna ke dalam sistem. Pada halaman *sign-in*, terdapat logo dan satu tombol bertuliskan “Sign in with Google”. Ketika tombol “Sign in with Google” ditekan oleh pengguna, maka sistem akan menampilkan *google authentication* untuk memasukkan pengguna melalui akun *Google* pengguna.

(b) Hasil Implementasi Antarmuka *Scan*

Gambar 5.22 Hasil Implementasi Antarmuka *Scan*

Pada Gambar 5.22 menunjukkan halaman *scan*. Halaman *scan* adalah halaman untuk melakukan *scanning* kode QR yang terdapat pada meja restoran. Pada halaman *scan*, terdapat terdapat kamera dan *bottom navigation* yang memiliki tiga menu *home, orders, scan, favorite menu,* dan *account*. Ketika perangkat bergerak pelanggan diarahkan pada kode QR di meja pelanggan, maka sistem akan menampilkan halaman *Choose Restaurant Menu* sesuai dengan menu restoran dan nomor meja yang pelanggan pilih.

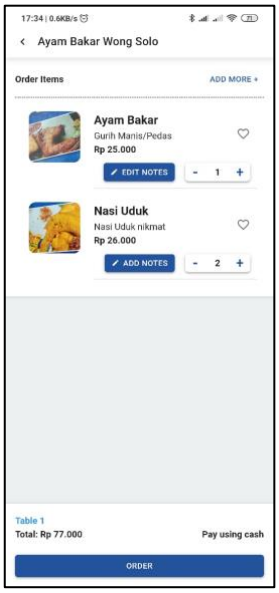


Gambar 5.23 Hasil Implementasi Antarmuka *Choose Restaurant Menu*

(c) Hasil Implementasi Antarmuka *Choose Restaurant Menu*

Pada Gambar 5.23 menunjukkan halaman *choose restaurant menu*. Halaman *choose restaurant menu* adalah halaman untuk melakukan pemilihan menu yang tersedia pada restoran. Pada halaman *choose restaurant menu*, terdapat terdapat deskripsi restoran, *menu categories* yang digambarkan dalam bentuk *tab bar*, dan pilihan menu dari setiap kategori dalam bentuk *card*. Kemudian dalam *menu card* terdapat gambar, nama, deskripsi, dan harga menu, beserta tombol untuk like, menambah, dan mengurangi menu yang diinginkan pelanggan.

(d) Hasil Implementasi Antarmuka *Checkout*



Gambar 5.24 Hasil Implementasi Antarmuka *Checkout*

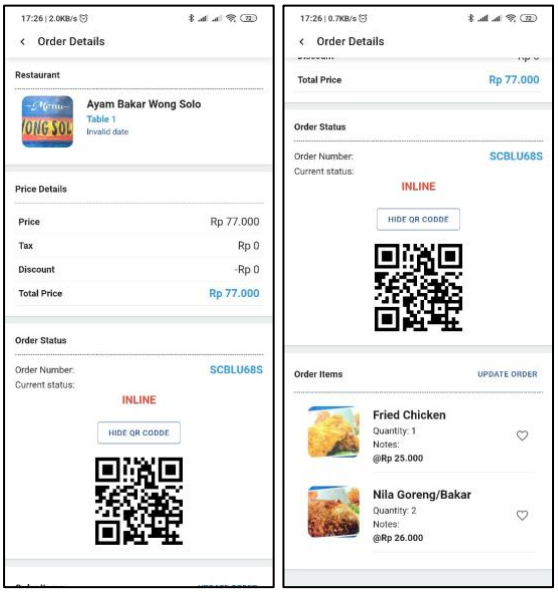
Pada Gambar 5.24 menunjukkan halaman *checkout*. Halaman *choose checkout* adalah halaman untuk melakukan *review* pemesanan yang meliputi menu yang dipesan beserta estimasi harga. Pada halaman *checkout*, terdapat terdapat *menu card* yang menampilkan menu yang sudah dipilih pelanggan dari halaman *choose restaurant menu* sebelumnya. Selain itu, terdapat estimasi harga dan tombol “Order” untuk melakukan pemesanan menu restoran dan memasukkan pelanggan ke dalam antrean pemesanan menu restoran.

(e) Hasil Implementasi Antarmuka *Orders*

Gambar 5.25 Hasil Implementasi Antarmuka *Orders*

Pada Gambar 5.25 menunjukkan halaman *orders*. Halaman *orders* adalah halaman untuk melihat riwayat pemesanan yang dilakukan oleh pelanggan. Pada halaman *orders*, terdapat daftar pesanan pelanggan yang ditampilkan dalam bentuk *card*. Ketika salah satu *order card* diklik, maka sistem akan menunjukkan halaman *Order Detail* yang berisi detail pemesanan.

(f) Hasil Implementasi Antarmuka *Order Detail*

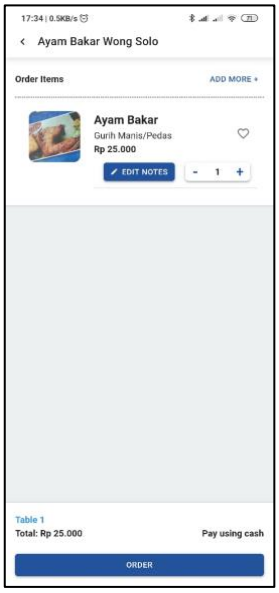


Gambar 5.26 Hasil Implementasi Antarmuka *Order Detail*

Pada Gambar 5.26 menunjukkan halaman *order detail*. Halaman *order detail* adalah halaman untuk melihat detail pesanan pelanggan. Pada halaman *order detail*, terdapat detail pemesanan yang dilakukan pelanggan yang meliputi data-

data restoran, detil harga, status pemesanan, dan menu yang dipesan. Selain itu, jika status pesanan masih berstatus “INLINE”, maka akan terdapat tombol untuk melakukan *update* pada *order.*

(g) Hasil Implementasi Antarmuka *Update Order*



Gambar 5.27 Hasil Implementasi Antarmuka *Update Order*

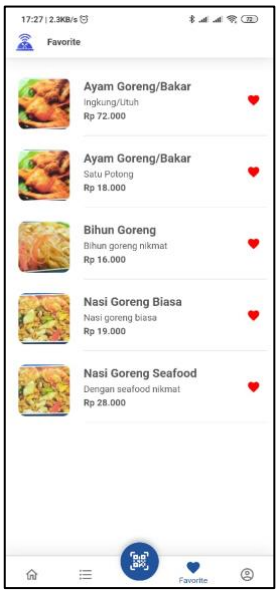
Pada Gambar 5.27 menunjukkan halaman *update order*. Halaman *update eorder* adalah halaman untuk melakukan perubahan pada pesanan yang telah dilakukan pelanggan, jika status pesanan masih “INLINE”. Padahalaman *update order,* terdapat pilihan seluruh menu yang ada di restoran berupa *menu card*. Dalam *menu card* terdapat tombol untuk menambahkan dan mengurangi kuantitas dan catatan. Kemudian terdapat tombol *“Update Order”* untuk melakukan perubahan pada pesanan.

Gambar 5.28 Hasil Implementasi Antarmuka *Home*

(h) Hasil Implementasi Antarmuka *Home*

Pada Gambar 5.28 menunjukkan halaman *home*. Halaman *home* adalah halaman untuk melihat urutan menu-menu pada restoran yang memiliki jumlah like terbanyak dari pelanggan. Pada halaman *home*, terdapat terdapat *card* yang menampilkan nama makanan dan gambar menu restoran yang sudah diurutkan sesuai dengan banyaknya jumlah like pada menu restoran tersebut.

(i) Hasil Implementasi Antarmuka *Favorite Menu*

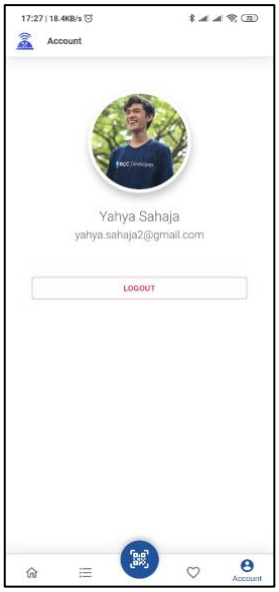


Gambar 5.29 Hasil Implementasi Antarmuka *Favorite Menu*

Pada Gambar 5.29 menunjukkan halaman *favorite menu*. Halaman *favorite menu* adalah halaman untuk melihat daftar menu yang telah disukai oleh pelanggan. Pada halaman *favorite menu*, terdapat terdapat daftar menu berupa *card* yang menampilkan gambar menu, nama menu, deskripsi menu, harga menu, dan tombol like. Jika tombol like *diklik*, maka menu tersebut akan dihapus dari daftar *menu favorite*.

(j) Hasil Implementasi Antarmuka *Account*

Pada Gambar 5.30 menunjukkan halaman account. Halaman *account* adalah halaman untuk melihat profil pelanggan yang telah masuk ke dalam sistem. Pada halaman *account*, terdapat terdapat tombol *logout* untuk mengeluarkan pelanggan dari sistem.



Gambar 5.30 Hasil Implementasi Antarmuka *Account*

5.7.1.8 *Evaluasi* dan Demonstrasi Produk

Setelah seluruh implementasi pada *sprint* pertama berhasil dilakukan, selanjutnya adalah melakukan evaluasi dan demonstrasi produk ke pengguna yang bersangkutan. Evaluasi pada *sprint* pertama dilakukan dengan pengujian *blackbox* dan *regression*, sedangkan demonstrasi produk dilakukan dengan mendemokan aplikasi kepada pelanggan restoran. Selanjutnya pelanggan diminta untuk memberikan *feedback* kepada aplikasi untuk dilakukan perbaikan pada *sprint* selanjutnya.

(a) Evaluasi

Evaluasi merupakan proses untuk validasi kesesuaian hasil implementasi dan kebutuhan sistem. Untuk melakukan evaluasi, dilakukan pengujian *black box* dan *regression*. Pengujian *blackbox* dilakukan dengan menguji setiap *use case* untuk memastikan apakah *expectation* dan *actual result* memiliki *validation* yang *valid* atau tidak *valid*. Hasil evaluasi pada *sprint* pertama dapat dilihat pada Tabel 5.20 hingga Tabel 5.27.

Tabel 5.20 Hasil pengujian *blackbox* pada *Use Case* Login Pelanggan

Test Case	*Expectation*	*Actual Result*	*Validation*
Login menggunakan akun Google pelanggan	Tamu pelanggan berhasil masuk ke dalam sistem menggunakan akun Google pelanggan	*Tamu pelanggan berhasil masuk ke dalam sistem menggunakan akun* Google pelanggan	*Valid*

Tabel 5.21 Hasil pengujian *blackbox* pada *Use Case* Melakukan Pemesanan

Test Case	*Expectation*	*Actual Result*	*Validation*

Melakukan pemesanan menu dari mulai *Scan* Kode QR, memilih menu yang ingin dipesan, melakukan *checkout*, dan melakukan pemesanan menu	Pelanggan berhasil melakukan pemesanan menu dan pesanan telah berhasil dibuat oleh sistem	Pelanggan berhasil melakukan pemesanan menu dan pesanan telah berhasil dibuat oleh sistem	*Valid*

Tabel 5.22 Hasil pengujian blackbox pada *Use Case* Melihat Riwayat Pemesanan

Test Case	*Expectation*	*Actual Result*	*Validation*
Melakukan pemesanan menu dari mulai *Scan* Kode QR, memilih menu yang ingin dipesan, melakukan *checkout*, dan melakukan pemesanan menu	Pelanggan berhasil melakukan pemesanan menu dan pesanan telah berhasil dibuat oleh sistem	Pelanggan berhasil melakukan pemesanan menu dan pesanan telah berhasil dibuat oleh sistem	*Valid*

Tabel 5.23 Hasil pengujian *blackbox* pada *Use Case* Mengubah Pesanan

Test Case	*Expectation*	*Actual Result*	*Validation*
Mengubah pesanan pelanggan ketika status pesanan adalah “INLINE”	Pelanggan berhasil mengubah pesanannya ketika status pesanan adalah “INLINE”	Pelanggan berhasil mengubah pesanannya ketika status pesanan adalah “INLINE”	*Valid*

Tabel 5.24 Hasil pengujian *blackbox* pada *Use Case* Melihat Menu Restoran yang Paling Banyak Disukai oleh Pelanggan

Test Case	*Expectation*	*Actual Result*	*Validation*
Melihat menu restoran yang paling banyak disukai oleh pelanggan-pelanggan di restoran	Daftar menu restoran yang paling banyak disukai oleh pelanggan berhasil ditampilkan oleh sistem	Daftar menu restoran yang paling banyak disukai oleh pelanggan berhasil ditampilkan oleh sistem	*Valid*

Tabel 5.25 Hasil pengujian blackbox pada *Use Case* Melihat Daftar Menu *Favorite*

Test Case	*Expectation*	*Actual Result*	*Validation*
Melihat daftar menu *favorite* yang	Daftar menu yang telah dijadikan sebagai menu	Daftar menu yang telah dijadikan sebagai menu	*Valid*

dilakukan oleh pelanggan	*favorite* berhasil ditampilkan oleh sistem	*favorite* berhasil ditampilkan oleh sistem	

Tabel 5.26 Hasil pengujian blackbox pada *Use Case* Menambah Menu Favorite

Test Case	*Expectation*	*Actual Result*	*Validation*
Melakukan penambahan menu restoran menjadi menu *favorite* pelanggan	Sistem berhasil menambahkan *menu* restoran yang dipilih pelanggan menjadi menu *favorite* pelanggan	Sistem berhasil menambahkan *menu* restoran yang dipilih pelanggan menjadi menu *favorite* pelanggan	*Valid*

Tabel 5.27 Hasil pengujian blackbox pada *Use Case* Logout Pelanggan

Test Case	*Expectation*	*Actual Result*	*Validation*
Melakukan logout dengan mengeluarkan pelanggan dari sistem	Sistem berhasil mengeluarkan pelanggan dari sistem	Sistem berhasil mengeluarkan pelanggan dari sistem	*Valid*

(b) Demonstrasi Produk

Setelah implementasi yang dilakukan divalidasi, selanjutnya dilakukan demonstrasi produk kepada pelanggan restoran untuk mendemokan aplikasi yang telah dibuat untuk mendapatkan pendapat dan saran dari pelanggan untuk dikembangkan pada *sprint* selanjutnya.

Dari demonstrasi produk pada *sprint* pertama ini, dadapatkan *feedback* dari operator restoran bahwa aplikasi sudah memuaskan dan mengatasi masalah pemesanan yang ada di restoran seperti penggunaan kertas untuk menulis pesanan, pemanggilan pelayan untuk memesan, dan penggunaan daftar menu dengan kertas yang tidak *update* terhadap ketersediaan menu restoran. Adapun terdapat foto yang menunjukkan demonstrasi produk yang dilakukan kepada pengguna dapat dilihat pada **Error! Reference source not found.**.

5.7.2 *Sprint* Kedua

5.7.2.1 Use Case Scenario

Use Case Scenario adalah skenario apa saja yang dapat terjadi pada sebuah Use Case. *Use Case Scenario* dari sistem manajemen antrean pesanan menu restoran pada *sprint* kedua dijabarkan pada Tabel 5.28 sampai dengan Tabel 5.40.

Tabel 5.28 Skenario *Use Case* Login Operator Restoran

Use Case	Login Operator Restoran
Kode terkait kebutuhan	SMA-F-001
Actor	Tamu Operator Restoran

Target	Operator restoran terdaftar atau masuk ke dalam sistem
Pre-Condition	Operator restoran belum terdaftar atau masuk ke dalam sistem
Main Flow	1. Sistem menampilkan halaman *Login* 2. Tamu operator restoran menekan tombol *Sign In With Google.* 3. Sistem menampilkan akun Google yang tersedia. 4. Tamu memilih akun Google atau menambahkan akun Google baru. 5. Sistem memproses otentikasi operator restoran.
Alternative Flow	Tamu operator restoran berhasil masuk ke dalam sistem dan status tamu berubah menjadi operator restoran
Post Condition	Operator restoran sudah terdaftar atau masuk ke dalam sistem

Tabel 5.29 Skenario *Use Case* Melihat Daftar Pesanan

Use Case	Melihat Daftar Pesanan
Kode terkait kebutuhan	SMA-F-002
Actor	Operator Restoran
Target	Operator restoran melihat daftar pesanan yang dibuat oleh pelanggan
Pre-Condition	Operator restoran telah berada di halaman dashboard
Main Flow	1. Operator restoran memilih halaman daftar pesanan 2. Sistem menampilkan halaman daftar pesanan dan menampilkan daftar pesanan yang dibuat oleh pelanggan dalam 3 tabel, yaitu tabel untuk status UNPAID, PROCESS, dan COMPLETED.
Alternative Flow	-
Post Condition	Operator restoran telah melihat daftar pesanan yang dibuat oleh pelanggan

Tabel 5.30 Skenario *Use Case* Mengubah Status Pesanan

Use Case	Mengubah Pesanan
Kode terkait kebutuhan	SMA-F-003
Actor	Operator Restoran
Target	Operator restoran dapat mengubah pesanan pelanggan
Pre-Condition	Operator restoran telah berada di halaman daftar pesanan dan daftar pesanan telah ditampilkan oleh sistem

Main Flow	1. Operator restoran menekan tombol “Process Order” 2. Sistem mengubah status pesanan dari “INLINE” menjadi “PROCESS”
Alternative Flow	1. Operator restoran menekan tombol “Complete Order” 2. Sistem mengubah status pesanan dari “PROCESS” menjadi “COMPLETED”
Post Condition	Operator restoran telah mengubah pesanan pelanggan

Tabel 5.31 Skenario *Use Case* Melihat Daftar Kategori Menu Restoran

Use Case	Melihat Daftar Kategori Menu Restoran
Kode terkait kebutuhan	SMA-F-004
Actor	Operator Restoran
Target	Operator restoran melihat daftar kategori menu restoran
Pre-Condition	Operator restoran telah berada di halaman *dashboard*
Main Flow	1. Operator restoran memilih halaman daftar kategori menu restoran 2. Sistem menampilkan halaman daftar kategori menu restoran
Alternative Flow	-
Post Condition	Operator restoran telah melihat daftar kategori menu restoran

Tabel 5.32 Skenario *Use Case* Mengubah Kategori Menu Restoran

Use Case	Mengubah Kategori Menu Restoran
Kode terkait kebutuhan	SMA-F-005
Actor	Operator Restoran
Target	Operator restoran dapat mengubah kategori menu restoran
Pre-Condition	Operator restoran telah berada di halaman daftar kategori menu restoran dan daftar kategori menu restoran telah ditampilkan oleh sistem

Main Flow	1. Operator restoran menekan tombol *edit* pada kategori yang ingin diubah 2. Sistem menampilkan formulir *dialog* untuk mengubah kategori menu restoran 3. Operator restoran menginputkan perubahan kategori menu restoran dan menekan tombol *save* 4. Sistem menyimpan perubahan kategori menu restoran
Alternative Flow	1. Jika Pelanggan memilih tombol cancel logout pada dialog, dialog tertutup kembali
Post Condition	Operator restoran telah mengubah kategori menu restoran

Tabel 5.33 Skenario *Use Case* Menambah Kategori Menu Restoran

Use Case	Menambah Kategori Menu Restoran
Kode terkait kebutuhan	SMA-F-006
Actor	Operator Restoran
Target	Operator restoran dapat menambah kategori menu restoran
Pre-Condition	Operator restoran telah berada di halaman daftar menu restoran dan formulir untuk menambah kategori menu restoran telah ditampilkan
Main Flow	1. Operator restoran menekan tombol floating action button bergambar logo “tambah” 2. Sistem menampilkan formulir dialog untuk menambah kategori menu restoran 3. Operator restoran menginputkan detail kategori menu restoran yang baru dan menekan tombol *save* 4. Sistem menyimpan kategori menu restoran
Alternative Flow	1. Jika Pelanggan memilih tombol cancel logout pada dialog, dialog tertutup kembali
Post Condition	Operator restoran telah menambah kategori menu restoran

Tabel 5.34 Skenario *Use Case* Menghapus Kategori Menu Restoran

Use Case	Menghapus Kategori Menu Restoran
Kode terkait kebutuhan	SMA-F-007
Actor	Operator Restoran
Target	Operator restoran dapat menghapus kategori menu restoran

Pre-Condition	Operator restoran telah berada di halaman daftar kategori menu restoran dan daftar kategori menu restoran telah ditampilkan oleh sistem
Main Flow	1. Operator restoran menekan tombbol “delete” pada kategori menu restoran yang ingin dihapus 2. Sistem menampilkan *dialog* konfirmasi penghapusan kategori menu restoran 3. Operator restoran menekan tombol *delete* 4. Sistem menghapus kategori menu restoran pelanggan yang dipilih dan mengeluarkan pesan bahwa kategori menu restoran telah dihapus

Tabel 5.35 Skenario *Use Case* Melihat Daftar Menu Restoran

Use Case	Melihat Daftar Menu Restoran
Kode terkait kebutuhan	SMA-F-008
Actor	Operator Restoran
Target	Operator restoran melihat daftar menu restoran
Pre-Condition	Operator restoran telah berada di halaman *dashboard*
Main Flow	1. Operator restoran memilih halaman daftar *category* 2. Sistem menampilkan halaman daftar *category* 3. Operator restoran memilih *category item* yang ingin dilihat menunya 4. Sistem menampilkan halaman menu restoran 5. Sistem menampilkan halaman daftar menu restoran
Alternative Flow	-
Post Condition	Operator restoran telah melihat daftar menu restoran

Tabel 5.36 Skenario *Use Case* Mengubah Menu Restoran

Use Case	Mengubah Menu Restoran
Kode terkait kebutuhan	SMA-F-009
Actor	Operator Restoran
Target	Operator restoran dapat mengubah menu restoran
Pre-Condition	Operator restoran telah berada di halaman daftar menu restoran dan daftar menu restoran telah ditampilkan oleh sistem

Main Flow	1. Operator restoran memilih menu restoran yang ingin diubah 2. Sistem menampilkan formulir *dialog* untuk mengubah menu restoran 3. Operator restoran menginputkan perubahan menu restoran dan menekan tombol *save* 4. Sistem menyimpan perubahan menu restoran
Alternative Flow	1. Jika Pelanggan memilih tombol cancel logout pada dialog, dialog tertutup kembali
Post Condition	Operator restoran telah mengubah menu restoran

Tabel 5.37 Skenario *Use Case* Menambah Menu Restoran

Use Case	Menambah Menu Restoran
Kode terkait kebutuhan	SMA-F-010
Actor	Operator Restoran
Target	Operator restoran dapat menambah menu restoran
Pre-Condition	Operator restoran telah berada di halaman daftar menu restoran dan formulir untuk menambah menu restoran telah ditampilkan
Main Flow	1. Operator restoran menekan tombol floating action button bergambar logo “tambah” 2. Sistem menampilkan formulir dialog untuk menambah menu restoran 3. Operator restoran menginputkan detail menu restoran yang baru dan menekan tombol *save* 4. Sistem menyimpan menu restoran
Alternative Flow	1. Jika Pelanggan memilih tombol cancel logout pada dialog, dialog tertutup kembali
Post Condition	Operator restoran telah menambah menu restoran

Tabel 5.38 Skenario *Use Case* Menghapus Menu Restoran

Use Case	Menghapus Pesanan
Kode terkait kebutuhan	SMA-F-011
Actor	Operator Restoran
Target	Operator restoran dapat menghapus menu restoran

Pre-Condition	Operator restoran telah berada di halaman daftar menu restoran dan daftar menu restoran telah ditampilkan oleh sistem
Main Flow	1. Operator restoran memilih menu restoran yang ingin dihapus 2. Sistem menampilkan *dialog* konfirmasi penghapusan menu restoran 3. Operator restoran menekan tombol *delete* 4. Sistem menghapus menu restoran pelanggan yang dipilih dan mengeluarkan pesan bahwa menu restoran telah dihapus
Alternative Flow	-
Post Condition	Operator restoran telah menghapus menu restoran pelanggan

Tabel 5.39 Skenario *Use Case* Mengubah Ketersediaan Menu Restoran

Use Case	Mengatur Stock Menu Restoran
Kode terkait kebutuhan	SMA-F-012
Actor	Operator Restoran
Target	Operator restoran dapat mengatur ketersediaan menu restoran
Pre-Condition	Operator restoran telah berada di halaman daftar menu restoran dan daftar menu restoran telah ditampilkan
Main Flow	1. Operator restoran memilih menu restoran yang ingin diubah stocknya menjadi tersedia atau tidak tersedia dengan menekan tombol *switch* pada menu restoran yang dipilih 2. Sistem mengubah ketersediaan menu restoran
Alternative Flow	-
Post Condition	Operator restoran telah mengubah ketersediaan menu restoran

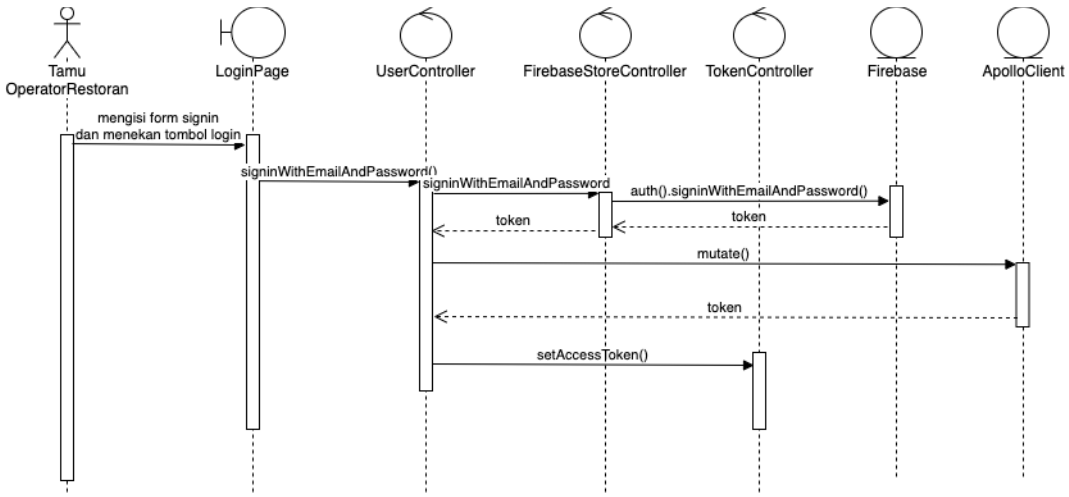
Tabel 5.40 Skenario *Use Case* Mengatur Ulang Ketersediaan Menu Restoran

Use Case	Mengatur Stock Menu Restoran
Kode terkait kebutuhan	SMA-F-013
Actor	Operator Restoran
Target	Operator restoran dapat mengatur ketersediaan stock menu restoran
Pre-Condition	Operator restoran telah berada di halaman daftar stock menu restoran dan daftar menu stock restoran telah ditampilkan

Main Flow	1. Operator restoran melakukan navigasi ke halaman "dashboard" 2. Sistem melakukan melakukan pengaturan ulang ketersediaan menu restoran menjadi tersedia
Alternative Flow	-
Post Condition	Operator restoran telah mengatur ulang ketersediaan menu restoran

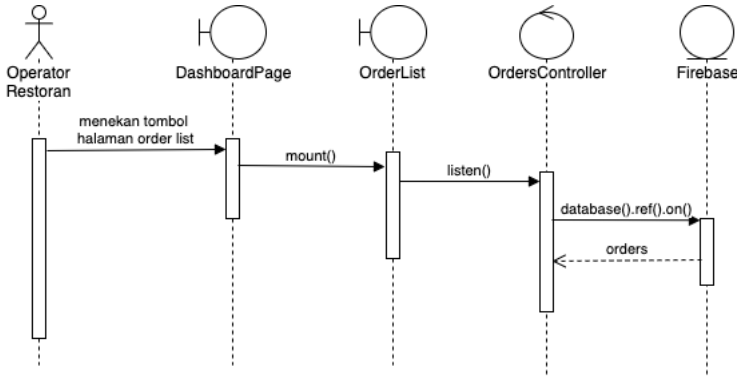
5.7.2.2 Sequence Diagram

Sequence diagram menggambarkan urutas proses yang terjadi di dalam sistem untuk mencapai tujuan dari kebutuhan fungsionalitas sistem. Daftar *sequence diagram* dalam sistem manajemen antrean pesanan menu restoran di *sprint* kedua dapat dilihat pada Gambar 5.31 sampai dengan Gambar 5.43.



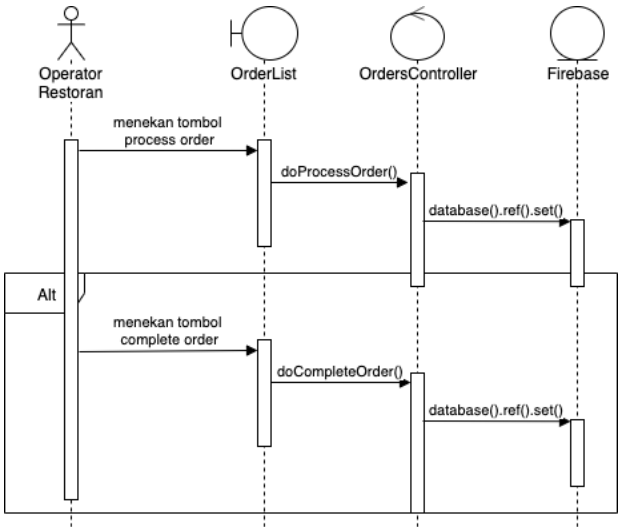
Gambar 5.31 *Sequence Diagram* Use Case Login Restaurant

Pada *sequence diagram* Login Restaurant pada Gambar 5.31 dijelaskan bahwa aktor Tamu Operator Restaurant menekan tombol "signin with google" pada halaman LoginPage. Kemudian dilakukan pemanggilan fungsi `signinWithEmailAndPassword()` pada UserController. Setelah pelanggan berhasil *login* dengan *credentials* akun google pelanggan, maka dilakukan *request* ke *server* untuk melakukan *signin*. Nilai kembalian dari request *login* adalah *token* yang digunakan untuk *authorization* pengguna saat melakukan *request* untuk mengakses *endpoints* pada *server*. Kemudian dilakukan pemanggilan fungsi `setAccessToken()` pada TokenController untuk menyimpan data pada *local storage browser* dan mengatur *default header* dari *service request data axios*.



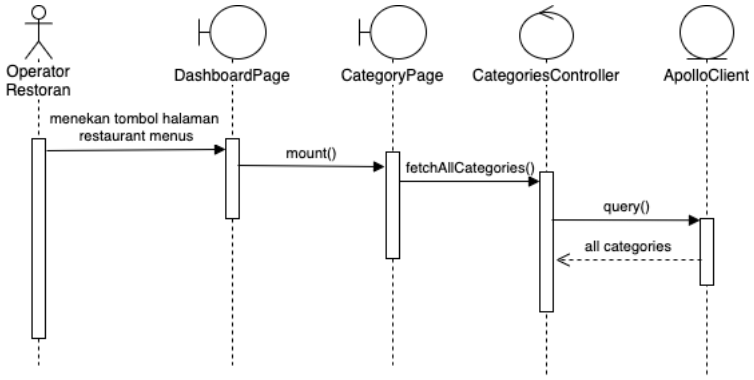
Gambar 5.32 *Sequence Diagram* Use Case Melihat Daftar Pesanan

Pada *sequence diagram* Melihat Daftar Pesanan pada Gambar 5.32 dijelaskan bahwa aktor operator restoran menekan tombol halaman order list pada navigasi DashboardPage. Kemudian dilakukan *mounting component* OrderList. Setelah itu dilakukan pemanggilan fungsi `listen()` pada OrdersController untuk membuat *event listener* pada model Orders pada Firebase Firestore. Kemudian setiap data Orders yang diterima dari *listening Orders* pada Firebase akan disimpan dalam *observable property* orders pada OrdersController.



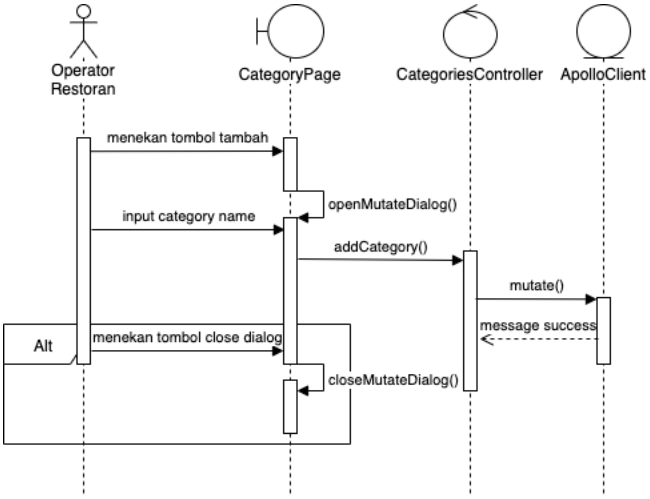
Gambar 5.33 *Sequence Diagram* Use Case Mengubah Status Pesanan

Pada *sequence diagram* Mengubah Status Pesanan pada Gambar 5.33 dijelaskan bahwa aktor operator restoran menekan tombol "process order" pada halaman OrderList. Aktivitas tersebut memicu pemanggilan `doProcessOrder()` pada OrdersController. Kemudian dilakukan *update field* pada model Orders pada Firebase Firestore. Terdapat *alternative flow* ketika operator restoran menekan tombol *complete order* pada halaman OrderList, maka akan memicu pemanggilan fungsi `doCompleteOrder()` pada OrdersController. Kemudian dilakukan *update order field* pada Firebase Firestore.



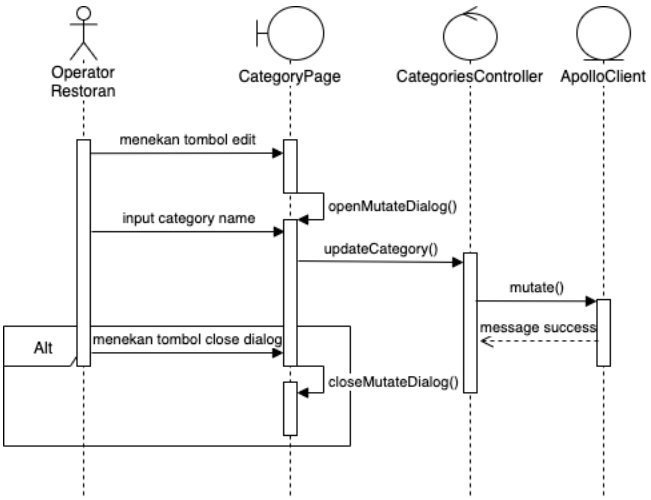
Gambar 5.34 *Sequence Diagram* Use Case Melihat Daftar Kategoi Menu Restoran

Pada *sequence diagram* Melihat Daftar Kategori Menu Restoran pada Gambar 5.34 dijelaskan bahwa aktor operator restoran menekan tombol halaman restoran menus pada halaman DashboardPage. Kemudian dilakukan *mounting component* pada CategoryPage. Setelah itu dilakukan pemanggilan `fetchAllCategories()` pada CategoriesController. Aktivitas tersebut memicu *request all categories* ke ApolloClient dan mendapatkan nilai kembalian berupa all categories yang disimpan dalam *observable property.*



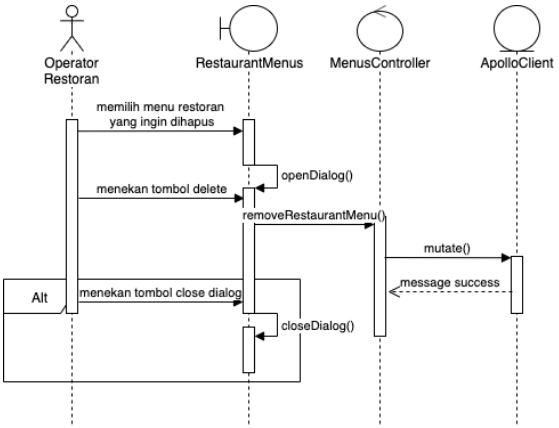
Gambar 5.35 *Sequence Diagram* Use Case Menambah Kategori Menu Restoran

Pada *sequence diagram* Menambah Kategori Menu Restoran pada Gambar 5.35 dijelaskan bahwa aktor operator restoran menekan tombol tambah pada *category item* halaman CategoryPage. Kemudian dilakukan *self-call* `openMutateDialog()`. Setelah itu operator restoran *input* menu restoran. Aktivitas tersebut memicu pemanggilan `addCategory()` pada CategoriesController dan melakukan *request add category* pada ApolloClient, kemudian mendapatkan nilai kembalian berupa *message success*. Apabila operator restoran menekan tombol *close dialog*, maka dilakukan *self-call* `closeMutateDialog()` pada CategoryPage.



Gambar 5.36 *Sequence Diagram* Use Case Mengubah Kategori Menu Restoran

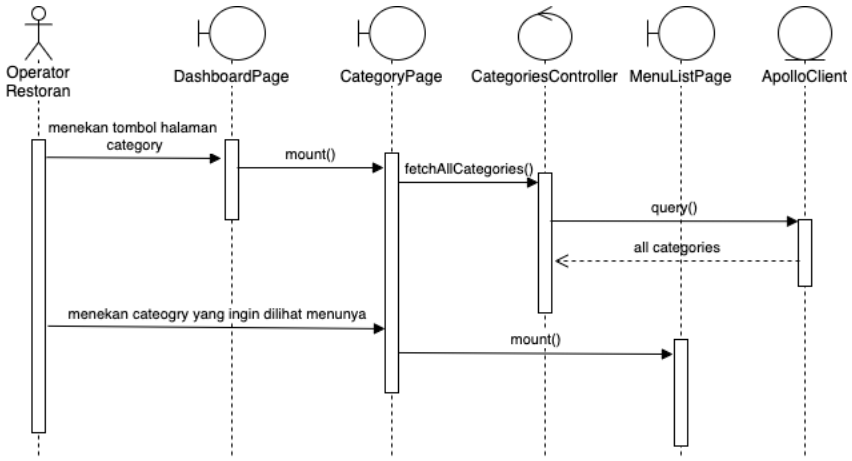
Pada *sequence diagram* Mengubah Kategori Menu Restoran pada Gambar 5.36 dijelaskan bahwa aktor operator restoran menekan tombol update tambah pada *category item* halaman CategoryPage. Kemudian dilakukan *self-call* `openMutateDialog()`. Setelah itu operator restoran *input* menu restoran. Aktivitas tersebut memicu pemanggilan `updateCategory()` pada CategoriesController dan melakukan *request update category* pada ApolloClient, kemudian mendapatkan nilai kembalian berupa *message success*. Apabila operator restoran menekan tombol *close dialog*, maka dilakukan *self-call* `closeMutateDialog()` pada CategoryPage.



Gambar 5.37 *Sequence Diagram* Use Case Menghapus Kategori Menu Restoran

Pada *sequence diagram* Menghapus Kategori Menu Restoran pada Gambar 5.37 dijelaskan bahwa aktor operator restoran menekan tombol tambah pada halaman CategoryPage. Kemudian dilakukan *self-call* `openMutateDialog()`. Setelah itu operator restoran *input* menu restoran. Aktivitas tersebut memicu pemanggilan `removeCategory()` pada CategoriesController dan melakukan *request remove category* pada ApolloClient, kemudian mendapatkan nilai

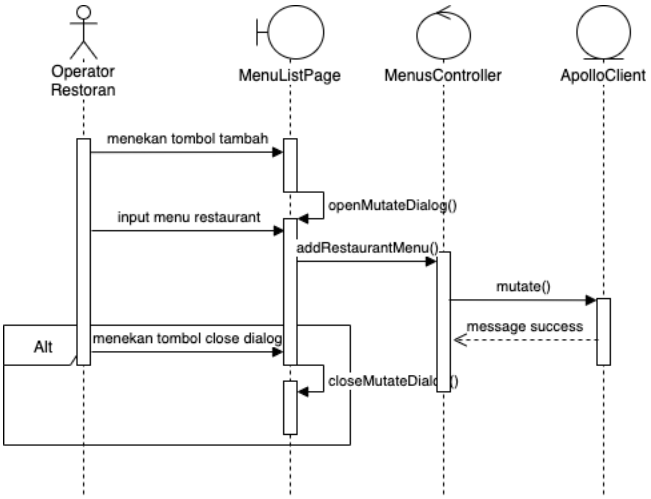
kembalian berupa *message success*. Apabila operator restoran menekan tombol *close dialog*, maka dilakukan *self-call* `closeMutateDialog()` pada CategoryPage.



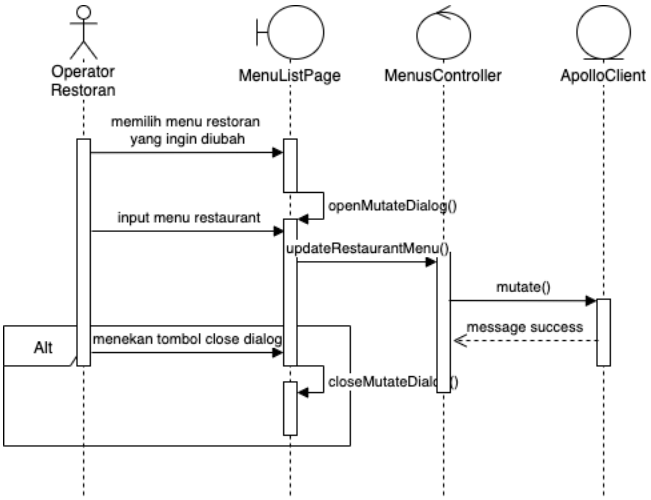
Gambar 5.38 *Sequence Diagram* Use Case Melihat Daftar Menu Restoran

Pada *sequence diagram* Melihat Menu Restoran pada Gambar 5.38 dijelaskan bahwa aktor operator restoran menekan tombol halaman restoran menus pada halaman DashboardPage. Kemudian dilakukan *mounting component* pada CategoryPage. Setelah itu dilakukan pemanggilan `fetchAllCategories()` pada CategoriesController. Aktivitas tersebut memicu *request all categories* ke ApolloClient dengan memanggil method `query()` dan mendapatkan nilai kembalian berupa all categories yang berisikan daftar kategori dan menunya. Kemudian data tersebut disimpan dalam *observable property.* Setelah itu operator restoran menekan *category item* yang ingin dilihat menunya dan kemudian sistem melakukan *mounting component* Menulistpage

Pada *sequence diagram* Menambah Menu Restoran pada Gambar 5.39 dijelaskan bahwa aktor operator restoran menekan tombol tambah pada halaman RestaurantMenus. Kemudian dilakukan *self-call* `openMutateDialog()`. Setelah itu operator restoran *input* menu restoran. Aktivitas tersebut memicu pemanggilan `addRestaurantMenu()` pada MenusController dan melakukan *request add restoran menu* pada ApolloClient, kemudian mendapatkan nilai kembalian berupa *message success*. Apabila operator restoran menekan tombol *close dialog*, maka dilakukan *self-call* `closeMutateDialog()` pada MenuList.

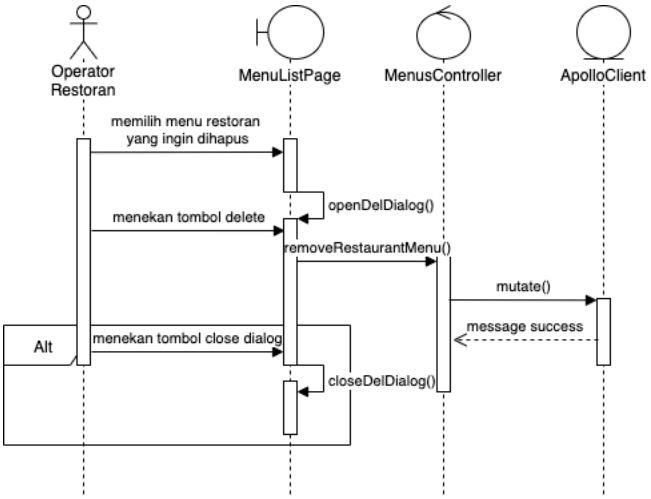


Gambar 5.39 *Sequence Diagram* Use Case Menambah Menu Restoran



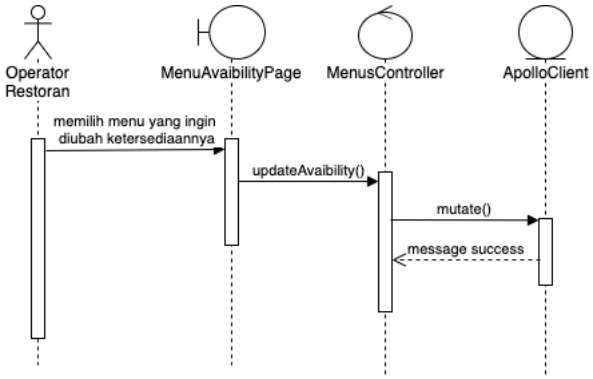
Gambar 5.40 *Sequence Diagram* Use Case Mengubah Menu Restoran

Pada *sequence diagram* Menambah Menu Restoran pada Gambar 5.40 dijelaskan bahwa aktor operator restoran memilih menu yang ingin diubah pada halaman MenuListPage. Kemudian dilakukan *self-call* `openMutateDialog()`. Setelah itu operator restoran *input* menu restoran. Aktivitas tersebut memicu pemanggilan `updateRestaurantMenu()` pada MenusController dan melakukan *request update restoran menu* pada ApolloClient, kemudian mendapatkan nilai kembalian berupa *message success*. Apabila operator restoran menekan tombol *close dialog*, maka dilakukan *self-call* `closeMutateDialog()` pada MenuListPage.



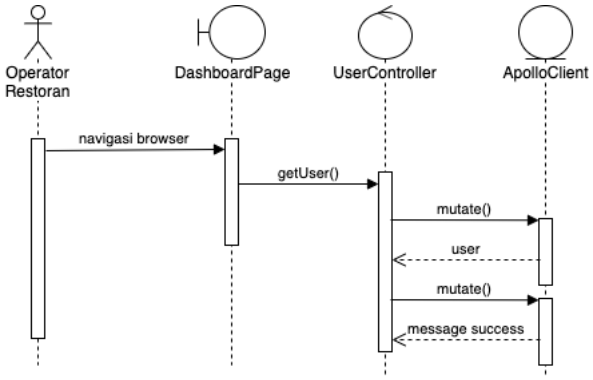
Gambar 5.41 *Sequence Diagram* Use Case Menghapus Menu Restoran

Pada *sequence diagram* Menghapus Menu Restoran pada Gambar 5.41 dijelaskan bahwa aktor operator restoran memilih menu yang ingin dihapus pada halaman MenuListPage. Kemudian dilakukan *self-call* `openDelDialog()`. Setelah itu operator restoran *input* menu restoran. Aktivitas tersebut memicu pemanggilan `removeRestaurantMenu()` pada MenusController dan melakukan *request remove restoran menu* pada ApolloClient, kemudian mendapatkan nilai kembalian berupa *message success*. Apabila operator restoran menekan tombol *close dialog*, maka dilakukan *self-call* `closeDelDialog()` pada MenuListPage



Gambar 5.42 *Sequence Diagram* Use Case Mengubah Ketersediaan Menu Restoran

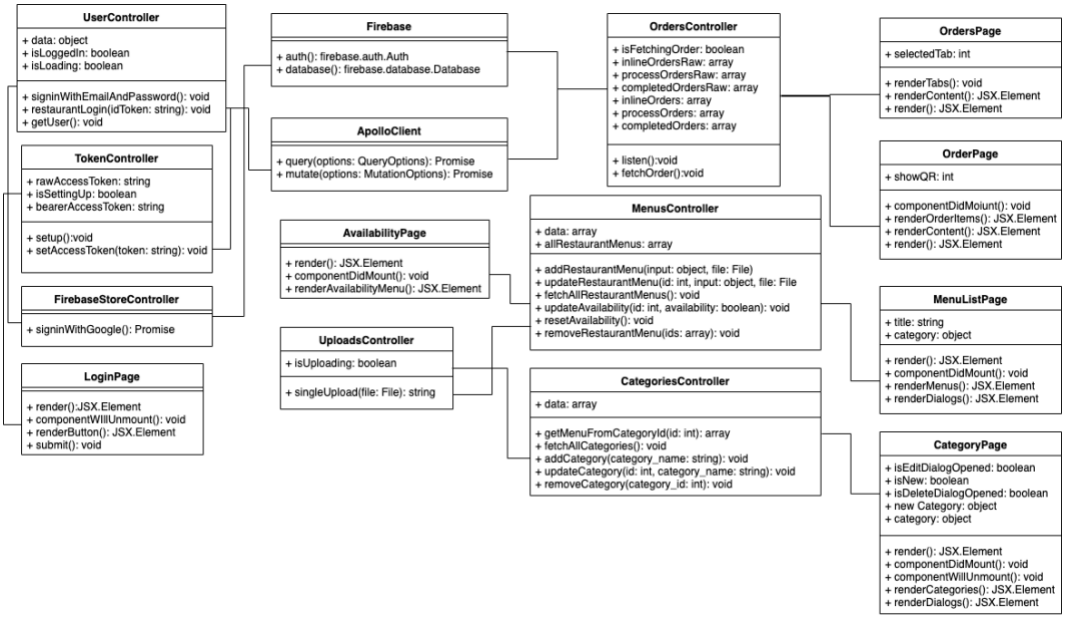
Pada *sequence diagram* Mengubah Ketersediaan Menu Restoran pada Gambar 5.42 dijelaskan bahwa aktor operator restoran memilih menu yang ingin diubah ketersediaannya pada halaman MenuAvaibilityPage. Kemudian dilakukan pemanggilan fungsi `updateAvaibility()` pada MenusController. Setelah itu dilakukan *request update menu avaibility* dan mendapatkan nilai kembalian *message success*.



Gambar 5.43 *Sequence Diagram* Use Case Mengatur Ulang Ketersediaan Menu Restoran

Pada *sequence diagram* Mengatur Ulang Ketersediaan Menu Restoran pada Gambar 5.43 dijelaskan bahwa aktor operator restoran melakukan navigasi browser ke halaman DashboardPage. Kemudian dilakukan pemanggilan fungsi `getUser()` pada UserController. Setelah itu dilakukan *request get user* pada ApolloClient. Nilai kembalian yang didapat berupa user yang disimpan dalam *observable property*. Kemudian dilakukan *request* untuk mengatur ulang ketersediaan menu ke ApolloClient dan mendapatkan nilai kembalian *message success*.

5.7.2.3 Class Diagram



Gambar 5.44 *Class Diagram* aplikasi manajemen antrean pesanan menu restoran pada *sprint* ketiga

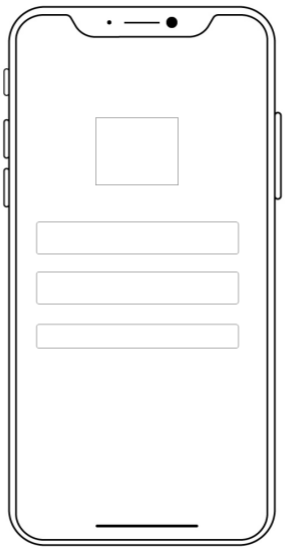
Class Diagram merupakan diagram yang merepresentasikan *class* yang akan digunakan dalam fase implementasi. Kelas-kelas yang terlibat dalam *class diagram* didasarkan pada *backlog* dalam satu *sprint* yang telah dibuat sebelumnya. *Class*

Diagram yang digunakan dalam *backlog sprint* ketiga ini digambarkan pada Gambar 5.44.

5.7.2.4 Perancangan Antarmuka

Perancangan antarmuka digunakan untuk mempermudah proses implementasi GUI pada perangkat lunak dengan gambaran mengenai tata letak dan tampilan pada aplikasi. Pembuatan perancangan antarmuka dapat dibuat dalam bentuk *wireframe* menggunakan Adobe XD. *Wireframe* yang dibuat ditunjukkan pada Gambar 5.45 sampai dengan Gambar 5.50.

(a) **Perancangan Antarmuka *Sign-in***

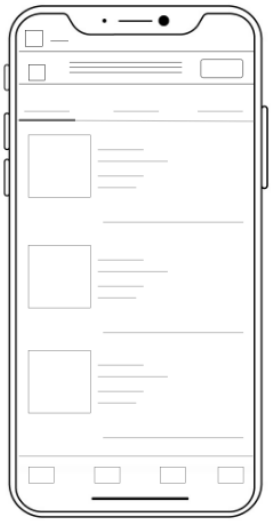


Gambar 5.45 *Wireframe Sign-in*

Pada Gambar 5.45 menunjukkan halaman *sign-in*. Halaman *sign-in* adalah halaman untuk memasukkan pengguna ke dalam sistem. Pada halaman *sign-in*, terdapat logo, *input* text email dan password, dan tombol “Sign in”. Ketika pengguna menekan tombol “Sign in” maka sistem akan melakukan otentikasi sesuai dengan *input* email dan *password* yang telah pengguna masukkan.

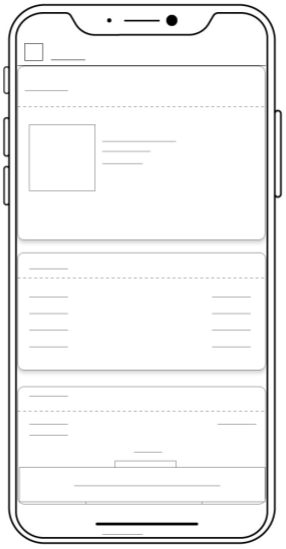
(b) **Perancangan Antarmuka *Orders***

Pada Gambar 5.46 menunjukkan halaman *orders*. Halaman *orders* menunjukkan halaman *orders*. Halaman *orders* adalah halaman untuk melihat riwayat pemesanan yang dilakukan oleh operator restoran. Pada halaman *orders*, terdapat daftar pesanan pelanggan pada restoran yang ditampilkan dalam bentuk *card*. Ketika salah satu *order card* diklik, maka sistem akan menunjukkan halaman *Order Detail* yang berisi detail pemesanan.



Gambar 5.46 *Wireframe Orders*

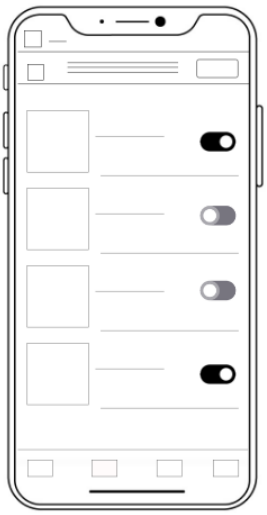
(c) **Perancangan Antarmuka *Order Detail***



Gambar 5.47 *Wireframe Order Detail*

Pada Gambar 5.47 menunjukkan halaman *order detail*. Halaman *order detail* adalah halaman untuk melihat detail pesanan pelanggan. Pada halaman *order detail*, terdapat detail pemesanan yang dilakukan pelanggan yang meliputi data-data restoran, detil harga, status pemesanan, dan menu yang dipesan. Selain itu, jika status pesanan masih berstatus "INLINE", maka akan terdapat tombol "*process order*" untuk mengubah status pesanan menjadi "PROCESS". Namun apabila pesanan masih berstatus "PROCESS", maka akan terdapat tombol *"complete order"* untuk mengubah status pesanan menjadi "COMPLETED".

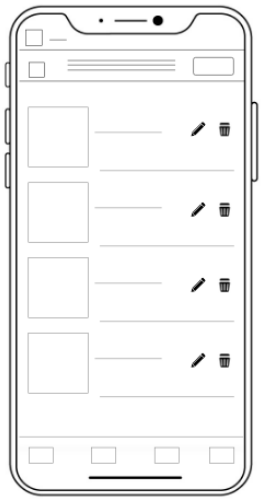
(d) **Perancangan Antarmuka *Menu Availability***



Gambar 5.48 *Wireframe Menu Availability*

Pada Gambar 5.48 menunjukkan halaman *menu availability*. Halaman *menu availability* adalah halaman untuk mengubah ketersediaan menu restoran menjadi tersedia atau tidak tersedia. Pada halaman *menu availability* terdapat daftar menu restoran dan *switch button* untuk mengubah ketersediaan menu restoran.

(e) **Perancangan Antarmuka *Category List***

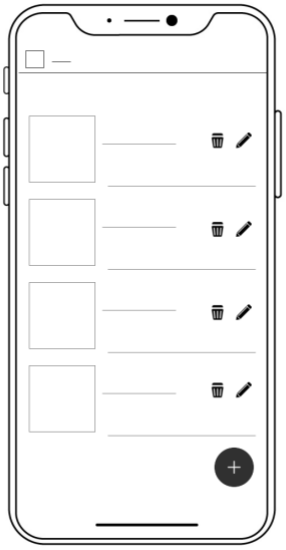


Gambar 5.49 *Wireframe Category List*

Pada Gambar 5.49 menunjukkan halaman *category list*. Halaman *category list* adalah halaman untuk melihat seluruh list kategori menu makanan yang ada di restoran. Pada halaman *category list*, terdapat terdapat *card* yang menampilkan

nama kategori dan gambar salah satu menu dalam kategori tersebut, tombol *edit*, dan tombol *delete*. *Category card* merupakan list yang dapat diklik untuk dapat membuka halaman *list menu* yang ada pada kategori tersebut. Selain itu, dalam setiap *category card*, terdapat tombol *add* berupa *floating action button* untuk menambah kategori. Tombol *edit* dan *delete* di setiap kategori digunakan untuk mengubah dan menghapus kategori, sedangkan tombol *add* digunakan untuk menambah kategori baru.

(f) **Perancangan Antarmuka *Menu List***



Gambar 5.50 *Wireframe Menu List*

Pada Gambar 5.50 menunjukkan halaman *menu list*. Halaman *menu list* adalah halaman untuk melihat seluruh list kategori menu makanan yang ada di restoran. Pada halaman *menu list*, terdapat terdapat *card* yang menampilkan nama menu, gambar menu, tombol *edit*, dan tombol *delete*. Selain itu, dalam setiap *menu card*, terdapat tombol *add* berupa *floating action button* untuk menambah menu. Tombol *edit* dan *delete* di setiap menu digunakan untuk mengubah dan menghapus menu, sedangkan tombol *add* digunakan untuk menambah menu baru dalam kategori yang telah dipilih oleh pelanggan.

5.7.2.5 Perancangan Algoritme

Pada bagian ini akan dilakukan perancangan algoritme untuk mendapatkan gambaran langkah-langkah dari suatu proses. Salah satu perancangan algoritme mengubah status pesanan menjadi “PROCESS” dapat dilihat pada Tabel 5.41.

Tabel 5.41 Perancangan algoritme mengubah status pesanan menjadi “PROCESS”

1 2 3	Begin Deklarasi id untuk inlineOrder yang ingin diubah statusnya

```
4     Request set firebase dengan referensi
    '/Orders/${id}/status' menjadi 'PROCESS'
5     Request set firebase firestore dengan referensi
6   '/Orders/${id}/updated_at' menjadi Date saat ini
7     Catch error request
8     Begin
9       Show error message
      End
    end
```

Algoritme mengubah status pesanan merupakan fungsi untuk mengubah status pesanan pelanggan. Diawali dengan deklarasi id yang ingin diubah statusnya. Kemudian dilakukan *request set* firebase dengan referensi `'/Orders/${id}/status'` menjadi 'PROCESS' sehingga status pesanan pelanggan dengan id tersebut akan menjadi 'PROCESS'. Kemudian dilakukan *request set* firebase dengan referensi `'/Orders/${id}/updated_at'` menjadi `Date.now()`. Setelah itu jika terdapat *error*, maka pesan *error* akan ditampilkan oleh sistem.

5.7.2.6 Implementasi Kode Program

Pada bagian ini akan dilakukan dilakukan implementasi dari algoritme yang telah dirancang sebelumnya. Bahasa pemrograman yang digunakan untuk melakukan implementasi pada sistem ini menggunakan bahasa pemrograman ECMAscript 2015 atau ES6. Salah satu implementasi kode program *process order* dapat dilihat pada Tabel 5.42.

Tabel 5.42 Implementasi kode program *process order*

```
1   @action
2     async doProcessOrder(id) {
3       if (!id) if (this.inlineDetailOrder) id =
    this.inlineDetailOrder.id
4       overlayLoading.show()
5       try {
6         await firebase
7           .database()
8           .ref(`/Orders/${id}/status`)
9           .set('PROCESS')
10        await firebase
11          .database()
12          .ref(`/Orders/${id}/updated_at`)
13          .set(Date.now())
14
15        return true
16      } catch (err) {
17        console.log('ERROR WHILE FETCHING WISH LIST ORDERS',
    err)
18      } finally {
19        overlayLoading.hide()
20      }
21    }
```

Pembahasan dari kode program diatas adalah sebagai berikut:

1. Baris 3: Mengubah variable id pada parameter id menjadi id yang terdapat pada inlineDetailOrder jika parameter id bernilai null.
2. Baris 6 – 9: *Request set* firebase dengan referensi `'/Orders/${id}/status'` menjadi 'PROCESS'.
3. Baris 10 – 13: *Request set* firebase dengan referensi `'/Orders/${id}/updated_at'` menjadi nilai dari `Date.now()`

5.7.2.7 Implementasi Antarmuka

Implementasi antarmuka berisi hasil implementasi dari perancangan antarmuka yang telah dibuat sebelumnya. Implementasi antarmuka dibuat dengan menggunakan *library* React. Hasil dari implementasi antarmuka dapat dilihat pada Gambar 5.51 sampai Gambar 5.56.

(a) Hasil Implementasi Antarmuka *Sign-in*

Gambar 5.51 Hasil Implementasi Antarmuka *Sign-in*

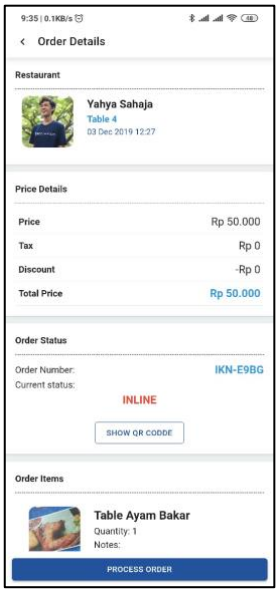
Pada Tabel 5.51 menunjukkan halaman *sign-in*. Halaman *sign-in* adalah halaman untuk memasukkan pengguna ke dalam sistem. Pada halaman *sign-in*, terdapat logo, *input* text email dan password, dan tombol "Sign in". Ketika pengguna menekan tombol "Sign in" maka sistem akan melakukan otentikasi sesuai dengan *input* email dan *password* yang telah pengguna masukkan.

(b) Hasil Implementasi Antarmuka *Orders*

Gambar 5.52 Hasil Implementasi Antarmuka *Orders*

Pada Tabel 5.52 menunjukkan halaman *orders*. Halaman *orders* menunjukkan halaman *orders*. Halaman *orders* adalah halaman untuk melihat riwayat pemesanan yang dilakukan oleh operator restoran. Pada halaman *orders*, terdapat daftar pesanan pelanggan pada restoran yang ditampilkan dalam bentuk *card*. Ketika salah satu *order card* diklik, maka sistem akan menunjukkan halaman *Order Detail* yang berisi detail pemesanan.

(c) Hasil Implementasi Antarmuka *Order Detail*

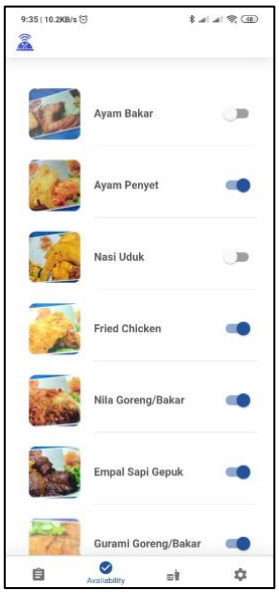


Gambar 5.53 Hasil Implementasi Antarmuka *Order Detail*

Pada Gambar 5.16 menunjukkan halaman *order detail*. Halaman *order detail* adalah halaman untuk melihat detail pesanan pelanggan. Pada halaman *order*

detail, terdapat detail pemesanan yang dilakukan pelanggan yang meliputi data-data restoran, detil harga, status pemesanan, dan menu yang dipesan. Selain itu, jika status pesanan masih berstatus "INLINE", maka akan terdapat tombol "*process order*" untuk mengubah status pesanan menjadi "PROCESS". Namun apabila pesanan masih berstatus "PROCESS", maka akan terdapat tombol *"complete order"* untuk mengubah status pesanan menjadi "COMPLETED".

(d) Hasil Implementasi Antarmuka *Menu Availability*



Gambar 5.54 Hasil Implementasi Antarmuka *Menu Availability*

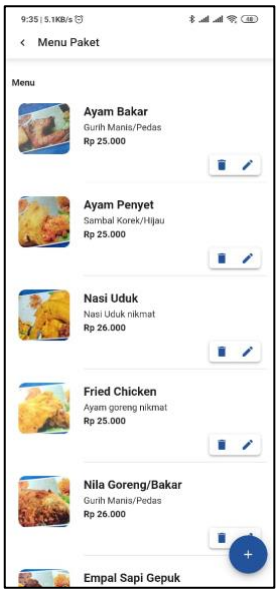
Pada Tabel 5.53 menunjukkan halaman *menu availability*. Halaman *menu availability* adalah halaman untuk mengubah ketersediaan menu restoran menjadi tersedia atau tidak tersedia. Pada halaman *menu availability* terdapat daftar menu restoran dan *switch button* untuk mengubah ketersediaan menu restoran.

(e) Hasil Implementasi Antarmuka *Category List*

Pada Tabel 5.54 menunjukkan halaman *category list*. Halaman *category list* adalah halaman untuk melihat seluruh list kategori menu makanan yang ada di restoran. Pada halaman *category list*, terdapat terdapat *card* yang menampilkan nama kategori dan gambar salah satu menu dalam kategori tersebut, tombol *edit*, dan tombol *delete*. *Category card* merupakan list yang dapat diklik untuk dapat membuka halaman *list menu* yang ada pada kategori tersebut. Selain itu, dalam setiap *category card*, terdapat tombol *add* berupa *floating action button* untuk menambah kategori. Tombol *edit* dan *delete* di setiap kategori digunakan untuk mengubah dan menghapus kategori, sedangkan tombol *add* digunakan untuk menambah kategori baru.

Gambar 5.55 Hasil Implementasi Antarmuka *Category List*

(f) Hasil Implementasi Antarmuka *Menu List*



Gambar 5.56 Hasil Implementasi Antarmuka *Menu List*

Pada Tabel 5.55 menunjukkan halaman *menu list*. Halaman *menu list* adalah halaman untuk melihat seluruh list kategori menu makanan yang ada di restoran. Pada halaman *menu list*, terdapat terdapat *card* yang menampilkan nama menu, gambar menu, tombol *edit*, dan tombol *delete*. Selain itu, dalam setiap *menu card*, terdapat tombol *add* berupa *floating action button* untuk menambah menu. Tombol *edit* dan *delete* di setiap menu digunakan untuk mengubah dan menghapus menu, sedangkan tombol *add* digunakan untuk menambah menu baru dalam kategori yang telah dipilih oleh pelanggan.

5.7.2.8 *Evaluasi* dan demonstrasi produk

Setelah seluruh implementasi pada *sprint* pertama berhasil dilakukan, selanjutnya adalah melakukan evaluasi dan demonstrasi produk ke pengguna yang bersangkutan. Evaluasi pada *sprint* kedua dilakukan dengan pengujian *blackbox* dan *regression*, sedangkan demonstrasi produk dilakukan dengan mendemokan aplikasi kepada operator restoran. Selanjutnya pelanggan diminta untuk memberikan *feedback* kepada aplikasi untuk dilakukan perbaikan pada *sprint* selanjutnya.

(a) Evaluasi

Evaluasi merupakan proses untuk validasi kesesuaian hasil implementasi dan kebutuhan sistem. Untuk melakukan evaluasi, dilakukan pengujian *black box* dan *regression*. Pengujian *blackbox* dilakukan dengan menguji setiap *use case* untuk memastikan apakah *expectation* dan *actual result* memiliki *validation* yang *valid* atau tidak *valid*. Hasil evaluasi pada *sprint* kedua dapat dilihat pada Tabel 5.43 sampai dengan Tabel 5.54.

Tabel 5.43 Hasil pengujian *blackbox* pada *Use Case* Login Operator Restoran

Test Case	*Expectation*	*Actual Result*	*Validation*
Login menggunakan email dan password dengan memasukkan email dan password operator restoran dan menekan tombol "Sign in"	Tamu operator restoran berhasil masuk ke dalam sistem menggunakan email dan password operator restoran	Tamu operator restoran berhasil masuk ke dalam sistem menggunakan email dan password operator restoran	*Valid*
Login menggunakan email atau password operator restoran yang salah	Tamu operator restoran tidak berhasil masuk ke dalam sistem dan sistem menampilkan pesan bahwa login gagal	Tamu operator restoran tidak berhasil masuk ke dalam sistem dan sistem menampilkan pesan bahwa login gagal	*Valid*

Tabel 5.44 Hasil pengujian *blackbox* pada *Use Case* Mengubah Status Pesanan

Test Case	*Expectation*	*Actual Result*	*Validation*
Mengubah status pesanan dari pelanggan dari "INLINE" menjadi "PROCESS"	Sistem berhasil mengubah status pesanan dari "INLINE" menjadi "PROCESS"	Sistem berhasil mengubah status pesanan dari "INLINE" menjadi "PROCESS"	*Valid*
Mengubah status pesanan dari pelanggan dari "PROCESS" menjadi "COMPLETED"	Sistem berhasil mengubah status pesanan dari "PROCESS" menjadi "COMPLETE"	Sistem berhasil mengubah status pesanan dari "PROCESS" menjadi "COMPLETED"	*Valid*

Tabel 5.45 Hasil pengujian blackbox pada *Use Case* Melihat Daftar Kategori Menu Restoran

Test Case	*Expectation*	*Actual Result*	*Validation*
Melihat daftar kategori menu restoran	Sistem berhasil menampilkan daftar kategori menu restoran	Sistem berhasil menampilkan daftar kategori menu restoran	*Valid*

Tabel 5.46 Hasil pengujian blackbox pada *Use Case* Menambah Kategori Menu Restoran

Test Case	*Expectation*	*Actual Result*	*Validation*
Menambahkan kategori menu restoran dengan menekan *Floating Action Button* "tambah"	Sistem berhasil menambahkan kategori menu restoran	Sistem berhasil menambahkan kategori menu restoran	*Valid*

Tabel 5.47 Hasil pengujian blackbox pada *Use Case* Mengubah Kategori Menu Restoran

Test Case	*Expectation*	*Actual Result*	*Validation*
Mengubah kategori menu restoran dengan menekan tombol "edit" pada daftar kategori menu restoran yang dipilih	Sistem berhasil mengubah kategori menu restoran	Sistem berhasil mengubah kategori menu restoran	*Valid*

Tabel 5.48 Hasil pengujian blackbox pada *Use Case* Menghapus Kategori Menu Restoran

Test Case	*Expectation*	*Actual Result*	*Validation*
Menghapus kategori menu restoran dengan menekan tombol "delete" pada daftar kategori menu restoran yang dipilih	Sistem berhasil menghapus kategori menu restoran	Sistem berhasil menghapus kategori menu restoran	*Valid*

Tabel 5.49 Hasil pengujian blackbox pada *Use Case* Melihat Daftar Menu Restoran

Test Case	*Expectation*	*Actual Result*	*Validation*

Melihat daftar menu restoran pada kategori yang telah dipilih oleh pelanggan	Sistem berhasil menampilkan daftar menu restoran	Sistem berhasil menampilkan daftar menu restoran	*Valid*

Tabel 5.50 Hasil pengujian blackbox pada *Use Case* Menambah Menu Restoran

Test Case	*Expectation*	*Actual Result*	*Validation*
Menambahkan menu restoran dengan menekan *Floating Action Button* "tambah"	Sistem berhasil menambahkan menu restoran	Sistem berhasil menambahkan menu restoran	*Valid*

Tabel 5.51 Hasil pengujian blackbox pada *Use Case* Mengubah Menu Restoran

Test Case	*Expectation*	*Actual Result*	*Validation*
Mengubah menu restoran dengan menekan tombol "edit" pada daftar menu restoran yang dipilih	Sistem berhasil mengubah menu restoran	Sistem berhasil mengubah menu restoran	*Valid*

Tabel 5.52 Hasil pengujian blackbox pada *Use Case* Menghapus Menu Restoran

Test Case	*Expectation*	*Actual Result*	*Validation*
Menghapus menu restoran dengan menekan tombol "delete" pada daftar menu restoran yang dipilih	Sistem berhasil menghapus menu restoran	Sistem berhasil menghapus menu restoran	*Valid*

Tabel 5.53 Hasil pengujian blackbox pada *Use Case* Mengubah Ketersediaan Menu Restoran

Test Case	*Expectation*	*Actual Result*	*Validation*
Mengubah ketersediaan menu restoran dengan menekan *switch button* pada menu restoran yang ingin diubah ketersediaannya	Sistem berhasil mengubah ketersediaan menu restoran	Sistem berhasil mengubah ketersediaan menu restoran	*Valid*

Tabel 5.54 Hasil pengujian blackbox pada *Use Case* Mengatur Ulang Ketersediaan Menu Restoran

Test Case	*Expectation*	*Actual Result*	*Validation*
Mengatur ulang ketersediaan menu restoran menjadi "tersedia" dengan melakukan akses kepada sistem pada hari yang berbeda dari terakhir kali operator restoran mengakses sistem	Sistem berhasil mengubah ketersediaan menu restoran kembali menjadi "tersedia"	Sistem berhasil mengubah ketersediaan menu restoran kembali menjadi "tersedia"	*Valid*

(b) Demonstrasi Produk

Setelah implementasi yang dilakukan divalidasi, selanjutnya dilakukan demonstrasi produk kepada operator restoran untuk mendemokan aplikasi yang telah dibuat untuk mendapatkan pendapat dan saran dari pelanggan untuk dikembangkan pada *sprint* selanjutnya.

Dari demonstrasi produk pada *sprint* kedua ini, dadapatkan *feedback* dari operator restoran bahwa urutan pesanan yang mengantre seharusnya diurutkan secara *ascending* dari yang terdahulu hingga yang terkini, sehingga pemilihan pesanan untuk diproses dapat lebih mudah. Selain itu, dibutuhkan *field* yang menampilkan tanggal dan waktu pesanan dibuat atau diubah. Adapun terdapat foto yang menunjukkan demonstrasi produk yang dilakukan kepada pengguna dapat dilihat pada **Error! Reference source not found.**.

5.7.3 Sprint Ketiga

5.7.3.1 Use Case Scenario

Use Case Scenario adalah skenario apa saja yang dapat terjadi pada sebuah Use Case. *Use Case Scenario* dari sistem manajemen antrean pesanan menu restoran pada *sprint* ketiga dijabarkan pada Tabel 5.55 sampai dengan Tabel 5.56.

Tabel 5.55 Skenario *Use Case* Mengubah Profil Restoran

Use Case	Mengubah Profil Restoran
Kode terkait kebutuhan	SMA-F-010
Actor	Operator Restoran
Target	Operator restoran dapat mengubah profil restoran
Pre-Condition	Operator restoran telah berada di halaman dashboard

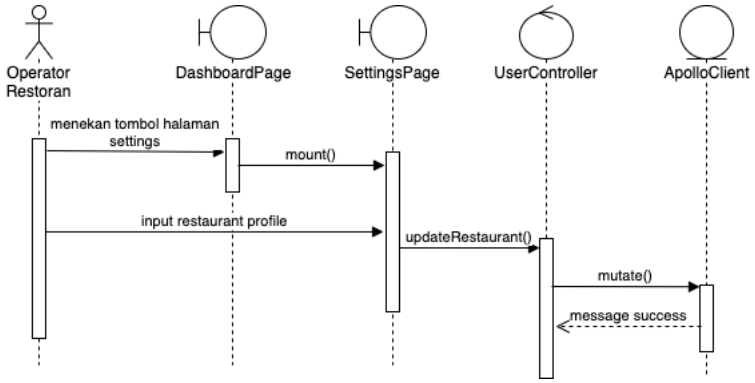
Main Flow	1. Operator restoran menekan tombol profil pada menu navigasi 2. Sistem menampilkan formulir untuk mengubah profil restoran 3. Operator restoran menginputkan perubahan profil restoran dan menekan tombol *save* 4. Sistem menyimpan perubahan profil restoran
Alternative Flow	-
Post Condition	Operator restoran telah mengubah profil restoran

Tabel 5.56 Skenario *Use Case* Logout Operator Restoran

Use Case	Logout Pelanggan
Kode kebutuhan terkait	SA-F-010
Actor	Operator Restoran
Target	Otentikasi operator restoran dapat keluar dari sistem.
Pre-Condition	Pelanggan sudah berada di halaman *dashboard*
Main Flow	1. Pelanggan menekan tombol *logout* pada menu navigasi 2. Sistem menampilkan dialog konfirmasi untuk *logout* 3. Pelanggan memilih tombol *Logout* 4. Sistem menampilkan halaman login dan menghapus informasi pengguna dari penyimpanan *local*
Alternative Flow	-
Post Condition	Otentikasi operator restoran telah keluar dari sistem.

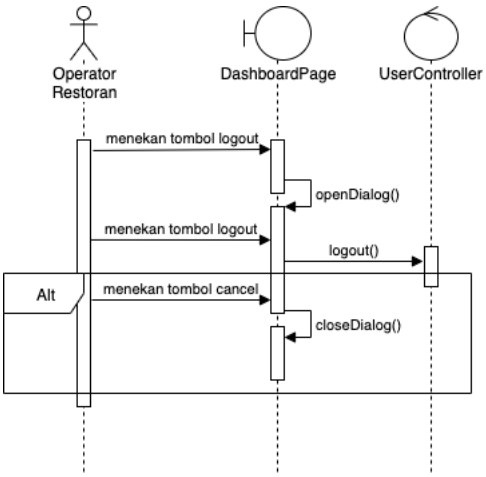
5.7.3.2 Sequence Diagram

Sequence diagram menggambarkan urutas proses yang terjadi di dalam sistem untuk mencapai tujuan dari kebutuhan fungsionalitas sistem. Daftar *sequence diagram* dalam sistem manajemen antrean pesanan menu restoran di *sprint* ketiga dapat dilihat pada Gambar 5.57 sampai Gambar 5.58.



Gambar 5.57 *Sequence Diagram* Use Case Mengubah Profil Restoran

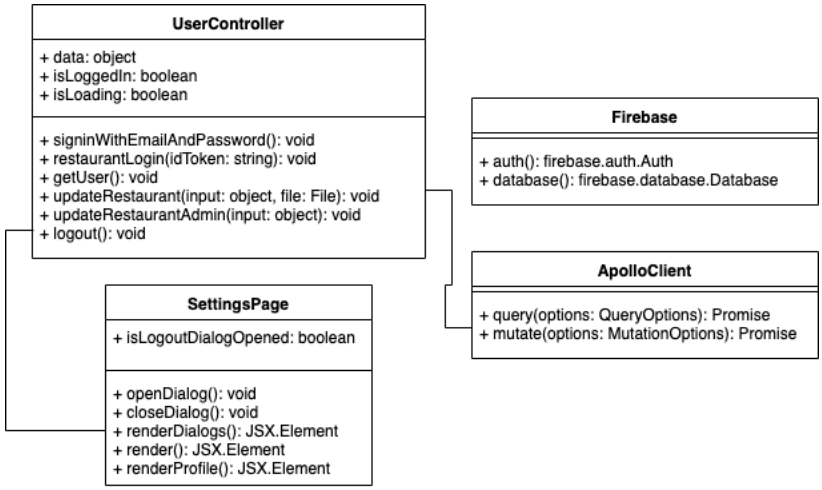
Pada *sequence diagram* Mengubah Profil Restoran pada Gambar 5.57 dijelaskan bahwa aktor operator restoran menekan tombol halaman settings pada DashboardPage. Kemudian dilakukan *mounting component* pada SettingsPage. Setelah itu operator restoran *input* profil restoran. Aktivitas tersebut akan memicu pemanggilan fungsi *updateRestaurant()* pada SettingsPage. Kemudian dilakukan *request update restoran* ke ApolloClient dan mendapatkan nilai kembalian *message success*.



Gambar 5.58 *Sequence Diagram* Use Case Logout Operator Restoran

Pada *sequence diagram* Logout Operator Restoran pada Gambar 5.58 dijelaskan bahwa aktor Operator Restoran menekan tombol logout pada halaman DashboardPage. Kemudian dilakukan *self-call openDialog()* sehingga sistem menampilkan dialog konfirmasi untuk logout. Kemudian pelanggan menekan tombol logout pada dialog. Aktivitas tersebut memicu pemanggilan fungsi *logout()* pada UserController untuk menghapus seluruh data otentikasi user pada *localStorage* browser. Apabila pelanggan menekan tombol *cancel,* maka dilakukan *self-call closeDialog()*.

5.7.3.3 Class Diagram



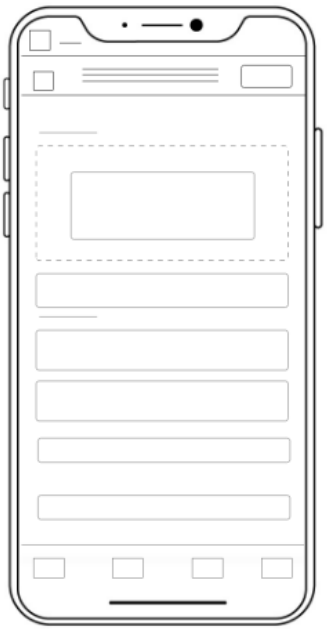
Gambar 5.59 Class Diagram aplikasi manajemen antrean pesanan menu restoran pada *sprint* ketiga

Class Diagram merupakan diagram yang merepresentasikan *class* yang akan digunakan dalam fase implementasi. *Classes* yang terlibat dalam *class diagram* didasarkan pada *backlog* yang telah dibuat sebelumnya. *Class Diagram* yang digunakan dalam *backlog sprint* ketiga ini digambarkan pada **Gambar 5.59**.

5.7.3.4 Perancangan Antarmuka

Perancangan antarmuka digunakan untuk mempermudah proses implementasi GUI pada perangkat lunak dengan gambaran mengenai tata letak dan tampilan pada aplikasi. Pembuatan perancangan antarmuka dapat dibuat dalam bentuk *wireframe* menggunakan Adobe XD. *Wireframe* yang dibuat ditunjukkan pada Gambar 5.60.

(a) **Perancangan Antarmuka *Settings***



Gambar 5.60 *Wireframe Settings*

Pada Gambar 5.60 menunjukkan halaman *settings*. Halaman *settings* adalah halaman untuk melihat profil operator restoran yang telah masuk ke dalam sistem. Pada halaman *settings*, terdapat *form* untuk melakukan perubahan pada profil restoran dan profil operator restoran. Selain itu, terdapat terdapat tombol *logout* untuk mengeluarkan pelanggan dari sistem.

5.7.3.5 Perancangan Algoritme

Pada bagian ini akan dilakukan perancangan algoritme untuk mendapatkan gambaran langkah-langkah dari suatu proses. Salah satu perancangan algoritme mengubah status pesanan menjadi "PROCESS" dapat dilihat pada Tabel 5.57.

Tabel 5.57 Perancangan algoritme mengubah status pesanan menjadi "PROCESS"

```
1   Begin
2     Jika terdapat file, maka upload file
3     Jika pattern pada string input.picture adalah url, maka ubah
    input.picture menjadi path url
4     Request graphql mutation updateRestaurant untuk mengubah data
    restaurant sesuai dengan input
5     Catch error request
6     Begin
7       Show error message
8     End
9   end
```

Algoritme *update restaurant* merupakan fungsi untuk mengubah data restoran. Diawali dengan mendeklarasikan id untuk pesanan yang akan diubah

statusnya. Kemudian dilakukan *request firebase* pada referensi `'/Orders/${id}/status'` menjadi `'PROCESS'`.

5.7.3.6 Implementasi Kode Program

Pada bagian ini akan dilakukan dilakukan implementasi dari algoritme yang telah dirancang sebelumnya. Bahasa pemrograman yang digunakan untuk melakukan implementasi pada sistem ini menggunakan bahasa pemrograman ECMAscript 2015 atau ES6. Salah satu implementasi kode program *update restaurant* dapat dilihat pada Tabel 5.58.

Tabel 5.58 Implementasi kode program *update restaurant*

```
1   @action
2     async updateRestaurant(input, file) {
3       try {
4         this.isUpdatingRestaurant = true
5         overlayLoading.show()
6
7         if (file) {
8           input.picture = await uploads.singleUpload(file)
9         }
10
11        if (isURL(input.picture)) {
12          let url = new URL(input.picture)
13          input.picture = url.pathname
14        }
15
16        await client.mutate({
17          mutation: updateRestaurantMutation,
18          variables: {
19            input
20          }
21        })
22
23        await this.fetchRestaurant()
24        snackbar.show('Restaurant data updated')
25      } catch(err) {
26        this.isLoadingLogin = false
27        console.log('ERROR WHILE UPDATE RESTO', err.message)
28        snackbar.show(err.message)
29        return err.message
30      } finally {
31        this.isUpdatingRestaurant = false
32        overlayLoading.hide()
33      }
34    }
```

Pembahasan dari kode program diatas adalah sebagai berikut:

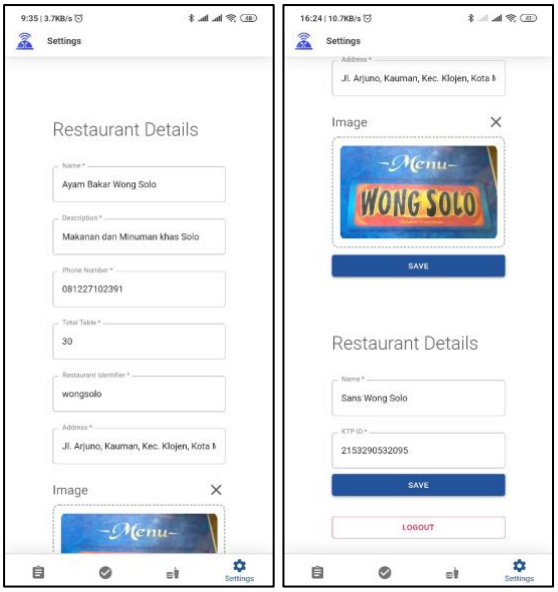
1. Baris 7 – 9: jika terdapat file, maka upload file terlebih dahulu
2. Baris 11 – 13: jika string `input.picture` memiliki *pattern* url, maka ubah nilai *input*.picture menjadi menjadi path dalam url `input.picture`.

3. Baris 10 – 13: *Request mutation graphql* untuk *update* restoran dengan data pada variable parameter `input`.

5.7.3.7 Implementasi Antarmuka

Implementasi antarmuka berisi hasil implementasi dari perancangan antarmuka yang telah dibuat sebelumnya. Implementasi antarmuka dibuat dengan menggunakan *library* React. Hasil dari implementasi antarmuka dapat dilihat pada Gambar 5.61.

(a)* Hasil Implementasi Antarmuka *Settings



Gambar 5.61 Hasil Implementasi Antarmuka *Settings*

Pada Gambar 5.61 menunjukkan halaman *settings*. Halaman *settings* adalah halaman untuk melihat profil operator restoran yang telah masuk ke dalam sistem. Pada halaman *settings*, terdapat *form* untuk melakukan perubahan pada profil restoran dan profil operator restoran. Selain itu, terdapat terdapat tombol *logout* untuk mengeluarkan pelanggan dari sistem.

5.7.3.8 *Evaluasi*

Pada *sprint ketiga,* evaluasi dilakukan hanya dengan menggunakan *blackbox testing* untuk menguji kesesuaian hasil implementasi dan kebutuhan sistem. Hasil evaluasi pada *sprint* ketiga dapat dilihat pada Tabel 5.59 sampai dengan Tabel 5.60.

Tabel 5.59 Hasil pengujian blackbox pada *Use Case* Mengubah Profil Restoran

Test Case	*Expectation*	*Actual Result*	*Validation*
Mengubah profil restoran dengan menginputkan data profil restoran dan	Sistem berhasil mengubah profil restoran sesuai dengan	Sistem berhasil mengubah profil restoran sesuai dengan data *input* operator restoran	*Valid*

menekan tombol "Save"	data *input* operator restoran		

Tabel 5.60 Hasil pengujian blackbox pada *Use Case* Logout Operator Restoran

Test Case	*Expectation*	*Actual Result*	*Validation*
Melakukan logout dengan mengeluarkan operator restoran dari sistem	Sistem berhasil mengeluarkan operator restoran dari sistem	Sistem berhasil mengeluarkan operator restoran dari sistem	*Valid*

BAB 6 PENGUJIAN

Bab ini membahas tentang pengujian aplikasi yang sudah diimplementasikan. Pengujian yang dilakukan menggunakan pengujian *Usability* untuk menguji kebergunaan aplikasi manajemen antrean pesanan menu restoran bagi penggunanya.

6.1 Pengujian *Usability*

Pengujian *usability* dilakukan untuk menguji kebergunaan aplikasi bagi pengguna. Pengujian dilakukan dengan cara memberikan *task scenario* kepada pengguna, kemudian pengguna akan melakukan aktivitas untuk menjalankan *task scenario* yang diberikan. *Task scenario* yang diberikan kepada pelanggan merupakan langkah-langkah dalam menjalankan aplikasi. Langkah-langkah tersebut mengacu pada fungsionalitas aplikasi.

Pada pengujian ini, *task scenario* diberikan kepada dua jenis pengguna, yaitu pelanggan restoran dan operator restoran. *Task scenario* untuk pelanggan dapat dilihat pada Tabel 6.1 dan *task scenario* untuk operator restoran dapat dilihat pada Tabel 6.2

Tabel 6.1 Task Scenario Untuk Pelanggan Restoran

Task	Nama *Task*	Langkah-langkah
1	Login	1. Menekan tombol "*account*" pada navigasi 2. Menekan tombol "*Sign in*" 3. Menekan tombol "*Sign In with* Google" 4. Login menggunakan akun Google Anda
2	Memesan menu restoran	1. Menekan tombol *Scan*/Pindai 2. Memindai QR Code yang terdapat pada meja Anda 3. Memilih menu restoran yang Anda inginkan 4. Menekan tombol "*Checkout*" 5. Memastikan kembali pesanan dengan benar dan menekan tombol "*Order*"
3	Melihat riwayat pesanan	1. Menekan tombol "*Orders*" pada navigasi 2. Memilih riwayat pesanan yang ingin dilihat lebih detail
4	Mengubah pesanan	1. Menekan tombol "*Update Order*" 2. Melakukan perubahan pemesanan menu 3. Menekan tombol update order
5	Menambahkan menu ke daftar menu favorit dan melihat menu favorite	1. Menekan tombol "*Orders*" pada navigasi 2. Memilih riwayat pesanan yang ingin dilihat lebih detail 3. Menekan tombol love berbentuk hati 4. Menekan tombol "*Back*" berbentuk panah 5. Menekan tombol "*Favorite Menu*" pada navigasi

6	Melihat menu rekomendasi	1. Menekan tombol "*Home*" pada navigasi
7	*Logout*	1. Menekan tombol "*Logout*" pada navigasi 2. Menekan tombol "*Logout*" pada konfirmasi dialog yang muncul

Tabel 6.2 *Task Scenario* Untuk Operator Restoran

Task	Nama *Task*	Langkah-langkah
1	*Login*	1. Menginputkan email dan password restoran Wong Solo yang sudah terdaftar dalam aplikasi 2. Menekan tombol "*Sign in*"
2	Mengelola pesanan pelanggan	1. Menekan tombol OrderList pada navigasi 2. Memilih pesanan yang ingin diproses pada "*Inline Order*" 3. Memastikan pesanan pelanggan 4. Menekan tombol "*Process Order*" 5. Memilih pesanan yang ingin diubah statusnya menjadi completed pada daftar "*Process Order*" 6. Menekan tombol "*Complete Order*" 7. Memilih pesanan yang sudah selesai untuk dilihat detailnya
3	*Menu availability*	1. Menekan tombol *Menu Avaibility* pada navigasi 2. Memilih menu yang ingin diatur ketersediaannya
4	Mengubah pesanan	1. Menekan tombol "*Update Order*" 2. Melakukan perubahan pemesanan menu 3. Menekan tombol *update order*
5	Manajemen menu restoran	1. Menekan tombol "*Restaurant Menu*" pada navigasi 2. Menekan tombol bericon "*plus*" untuk menambah menu makanan 6. Menginputkan menu restoran yang ingin ditambah 7. Menekan tombol "*Add*" 8. Memilih menu yang ingin diubah 9. Menginputkan menu restoran yang ingin diubah 10. Menekan tombol "*Update*" 11. Memilih menu yang ingin dihapus dengan menekan tombol "*checklist*" 12. Menekan tombol hapus bericon "*trash*" bewarna merah 13. Menekan tombol "*Delete*"
6	Mengubah profil restoran	1. Menekan tombol "*Settings*" pada navigasi 2. Menginputkan profil restoran yang ingin diperbarui 3. Menekan tombol "*Save*"

7	*Logout*	3. Menekan tombol "*Logout*" pada navigasi 4. Menekan tombol "*Logout*" pada konfirmasi dialog yang muncul

Pada Tabel 6.1 merupakan 7 *task scenario* yang diberikan kepada pelanggan restoran, sedangkan pada Tabel 6.2 merupakan 7 *task scenario* yang diberikan kepada operator restoran. Dalam melaksanakan *task scenario* yang diberikan, masing-masing pengguna harus menjalankan *task scenario* tersebut secara berurutan.

Kemudian responden diminta untuk mengisi kuesioner sebagai penilaian tentang usabilitas pada aplikasi ini. Pengujian pada penelitian ini menggunakan kuesioner *SUPER-Qm* yang terdiri dari 16 pertanyaan untuk masing-masing jenis pengguna yang dapat dilihat pada Tabel 6.4 untuk pengguna pelanggan restoran dan pada Tabel 6.5 untuk pengguna operator restoran. Setiap pertanyaan yang diberikan memiliki skor dengan menggunakan skala likert dalam rentang 1 sampai 5 yang dijelaskan pada Tabel 6.3.

Tabel 6.3 Skor Skala Likert dari Setiap Pertanyaan

Skor	Keterangan
1	Sangat Tidak Setuju (STS)
2	Tidak Setuju (TS)
3	Netral (N)
4	Setuju (S)
5	Sangat Setuju (SS)

Tabel 6.4 Kuesioner SUPR-Qm untuk Pelanggan Restoran

No	Pernyataan
1	Aplikasi ini penting untuk saya
2	Aplikasi ini merupakan aplikasi pemesanan menu restoran terbaik yang pernah saya gunakan
3	Saya tidak tahu apakah adalah aplikasi pemesanan menu restoran yang lebih baik dari aplikasi ini
4	Saya tidak akan menghapus aplikasi ini dari smartphone saya
5	Saya akan menyarankan aplikasi ini ke teman saya

6	Saya suka mengeksplorasi fitur-fitur yang terdapat dalam aplikasi ini
7	Aplikasi ini memiliki seluruh fitur dan fungsi yang saya inginkan pada aplikasi pemesanan menu restoran
8	Saya akan sering membuka aplikasi ini setiap kali saya ingin memesan menu di restoran Wong Solo
9	Aplikasi ini menyenangkan
10	Aplikasi ini bekerja dengan baik dengan fitur-fitur lain yang ada pada *smartphone* saya (contohnya kamera)
11	Saya akan menggunakan aplikasi ini ketika ingin memesan menu di restoran Wong Solo
12	Desain dari aplikasi ini memudahkan saya dalam menemukan informasi yang saya inginkan
13	Menurut saya aplikasi ini menarik
14	Aplikasi ini sesuai dengan kebutuhan saya
15	Melakukan navigasi dalam aplikasi ini mudah bagi saya
16	Aplikasi ini mudah digunakan

Tabel 6.5 Kuesioner SUPR-Qm untuk Operator Restoran

No	Pernyataan
1	Aplikasi ini penting untuk saya
2	Aplikasi ini merupakan aplikasi pemesanan menu restoran terbaik yang pernah saya gunakan
3	Saya tidak tahu apakah adalah aplikasi pemesanan menu restoran yang lebih baik dari aplikasi ini
4	Saya tidak akan menghapus aplikasi ini dari smartphone saya
5	Saya akan menyarankan aplikasi ini ke teman saya
6	Saya suka mengeksplorasi fitur-fitur yang terdapat dalam aplikasi ini

7	Aplikasi ini memiliki seluruh fitur dan fungsi yang saya inginkan pada aplikasi pemesanan menu restoran
8	Saya akan sering membuka aplikasi ini setiap kali saya ingin memesan menu di restoran Wong Solo
9	Aplikasi ini menyenangkan
10	Aplikasi ini bekerja dengan baik dengan fitur-fitur lain yang ada pada *smartphone* saya (contohnya kamera)
11	Saya akan menggunakan aplikasi ini ketika ingin memesan menu di restoran Wong Solo
12	Desain dari aplikasi ini memudahkan saya dalam menemukan informasi yang saya inginkan
13	Menurut saya aplikasi ini menarik
14	Aplikasi ini sesuai dengan kebutuhan saya
15	Melakukan navigasi dalam aplikasi ini mudah bagi saya
16	Aplikasi ini mudah digunakan

Pada Tabel 6.4 dan Tabel 6.5 merupakan 16 buah pertanyaan dengan menggunakan kuesioner SUPR-Qm yang nantinya akan didapatkan hasil secara kuantitatif terhadap kebergunaan dan kemudahan aplikasi yang dikembangkan.

Perhitungan SUPR-Q yang dilakukan untuk mendapatkan hasil kuantitatif dari skor pada pertanyaan kuesioner SUPR-Qm dilakukan dengan menggunakan rumus pada Persamaan 6.1.

$$Nilai\ SUPR - Q = \frac{jumlah\ nilai\ diperoleh}{jumlah\ nilai\ maksimal} \times 100\% \quad (6.1)$$

Persamaan SUPR-Q pada Persamaan 6.1 digunakan untuk perhitungan nilai SUPR-Qm. Kemudian nilai hasil perhitungan tersebut akan dikonversikan ke dalam kategori penilaian *usability*.

6.1.1 Analisis Hasil Pengujian Usability untuk Pelanggan Restoran

Pengujian *usability* untuk pelanggan restoran dilakukan kepada lima responden yang merupakan pelanggan restoran Ayam Bakar Wong Solo. Pengambilan responden dilakukan secara acak kepada pelanggan yang datang ke

restoran untuk memesan makanan dan memakan makanan di restoran. Adapun lima responden tersebut dijelaskan pada Tabel 6.6.

Tabel 6.6 Responden Pengujian *Usability* Pelanggan Restoran

No	Nama	Latar Belakang
1	Insan Nurzaman	Pelanggan yang kurang dari tiga kali dalam setahun memesan menu di restoran Wong Solo. Pengguna *smartphone* Android yang sering menggunakan *smartphone*-nya untuk penggunaan kamera atau GPS
2	Zakwan Dhiyaulhaq	Pelanggan yang sekali dalam setahun memesan menu di restoran Wong Solo. Pengguna *smartphone* Android yang menggunakan *smartphone*-nya untuk penggunaan finansial dan kamera atau GPS
3	Saiful Wardi Lubis	Pelanggan yang lebih dari lima kali dalam satu tahun memesan menu di restoran Wong Solo. Pengguna *smartphone* Android yang menggunakan *smartphone*-nya untuk penggunaan finansial dan hiburan
4	Izhari Adi	Pelanggan yang kurang dari tiga kali dalam satu tahun memesan menu di restoran Wong Solo. Pengguna *smartphone* iPhone yang menggunakan *smartphone*-nya untuk membaca berita dan penggunaan finansial
5	Nehru Priyambodo	Pelanggan yang kurang dari lima kali dalam satu tahun memesan menu di restoran Wong Solo. Pengguna *smartphone* Android yang menggunakan *smartphone*-nya untuk membaca berita dan penggunaan *social media.*

Kemudian responden diminta untuk mengoperasikan aplikasi berdasarkan *task scenario* dan moderator mengamati perilaku responden untuk mengetahui apakah responden dapat menyelesaikan *task scenario* yang diberikan atau tidak. Hasil dari penyelesaian *task scenario* dapat dilihat pada Tabel 6.7.

Tabel 6.7 Task Completion Rate

	User 1	*User 2*	*User 3*	*User 4*	*User 5*	*User 6*	*Completion Rate*
Task 1	√	√	√	√	√	√	*100%*
Task 2	√	√	√	√	√	√	*100%*
Task 3	√	√	√	√	√	√	*100%*
Task 4	√	√	√	√	√	√	*100%*
Task 5	√	√	√	√	√	√	*100%*
Task 6	√	√	√	√	√	√	*100%*
Task 7	√	√	√	√	√	√	*100%*

Pada Tabel 6.7 dapat disimpulkan bahwa semua responden dapat menyelesaikan tujuh *task scenario* yang diberikan oleh moderator dengan *completion rate* bernilai 100%.

Setelah responden menyelesaikan *task scenario* yang diberikan, selanjutnya responden diminta untuk mengisi kuesioner SUPR-Qm yang telah didefinisikan sebelumnya. Adapun hasil kuesioner SUPR-Qm yang telah diisi oleh responden pelanggan dapat dilihat pada **Error! Reference source not found.** dan hasil dari pengujian kuesioner SUPR-Qm untuk pelanggan restoran dapat dilihat pada Tabel 6.8.

Tabel 6.8 Hasil Pengujian Kuesioner SUPR-Qm untuk Pelanggan Restoran

No	Pertanyaan	Skor					Total Skor
		1	2	3	4	5	
1	Aplikasi ini penting untuk saya	0	0	0	5	0	20
2	Aplikasi ini merupakan aplikasi pemesanan menu restoran terbaik yang pernah saya gunakan	0	0	2	3	0	18
3	Saya tidak tahu apakah adalah aplikasi pemesanan menu restoran yang lebih baik dari aplikasi ini	0	0	1	3	1	20
4	Saya tidak akan menghapus aplikasi ini dari smartphone saya	0	0	1	2	2	21
5	Saya akan menyarankan aplikasi ini ke teman saya	0	0	0	4	1	21
6	Saya suka mengeksplorasi fitur-fitur yang terdapat dalam aplikasi ini	0	0	0	3	2	22
7	Aplikasi ini memiliki seluruh fitur dan fungsi yang saya inginkan pada aplikasi pemesanan menu restoran	0	1	0	3	1	19
8	Saya akan sering membuka aplikasi ini setiap kali saya ingin memesan menu di restoran Wong Solo	0	0	2	1	2	20
9	Aplikasi ini menyenangkan	0	0	2	0	3	21
10	Aplikasi ini bekerja dengan baik dengan fitur-fitur lain yang ada pada *smartphone* saya (contohnya kamera)	0	0	1	2	2	21
11	Saya akan menggunakan aplikasi ini ketika ingin memesan menu di restoran Wong Solo	0	0	1	3	1	20
12	Desain dari aplikasi ini memudahkan saya dalam menemukan informasi yang saya inginkan	0	0	0	1	4	24
13	Menurut saya aplikasi ini menarik	0	0	0	4	1	21
14	Aplikasi ini sesuai dengan kebutuhan saya	0	0	1	3	1	20

15	Melakukan navigasi dalam aplikasi ini mudah bagi saya	0	0	2	2	1	19
16	Aplikasi ini mudah digunakan	0	0	0	3	2	22
						Total Akhir	329
						Total Nilai Maksimum	400
						Nilai SUPR-Qm	82,25%

Tabel 6.8 menunjukkan penilaian tertinggi didapat pada pertanyaan *SUPR-Qm* nomor 12 dengan skor 24 dan penilaian terendah didapat pada pertanyaan SUPR-Qm nomor 2 dengan skor 18. Pada tabel juga menunjukkan total akhir yang bernilai 329, sehingga didapatkan nilai SUPR-Qm sebesar 82,25%.

Setelah mendapatkan nilai akhir SUPR-Qm sebesar 82,25%, kemudian nilai tersebut diinterpretasi ke kategori nilai *usability*. Penginterpretasian nilai 82,25% pada kategori nilai *usability* mendapatkan kategori A dengan *adjective rating Excellent*. Sehingga dapat disimpulkan bahwa aplikasi manajemen antrean pesanan menu restoran untuk pelanggan sangat baik dan dapat diterima oleh pelanggan restoran Ayam Bakar Wong Solo.

6.1.2 Analisis Pengujian Usability Aplikasi untuk Operator Restoran

Pengujian *Usability* kepada operator restoran diberikan kepada satu responden. Operator restoran di Restoran Ayam Bakar Wongsolo dikenal sebagai 'Kapten Stelling' sehingga pengujian dilakukan kepada 100% populasi, yaitu operator restoran pada Restoran Ayam Bakar Wong Solo, Ibu Dian Presetianingrum.

Kemudian responden diminta untuk mengoperasikan aplikasi berdasarkan *task scenario* dan moderator mengamati perilaku responden untuk mengetahui apakah responden dapat menyelesaikan *task scenario* yang diberikan atau tidak. Hasil dari penyelesaian *task scenario* dapat dilihat pada Tabel 6.9.

Tabel 6.9 Task Completion Rate

	User 1	*Completion Rate*
Task 1	√	*100%*
Task 2	√	*100%*
Task 3	√	*100%*
Task 4	√	*100%*
Task 5	√	*100%*
Task 6	√	*100%*
Task 7	√	*100%*

Pada Tabel 6.9 dapat disimpulkan bahwa semua responden dapat menyelesaikan tujuh *task scenario* yang diberikan oleh moderator dengan *completion rate* bernilai 100%.

Responden menyelesaikan *task scenario* kemudian responden mengisi kuesioner SUPR-Qm yang telah didefinisikan sebelumnya. Berikut hasil kuesioner SUPR-Qm responden operator restoran dapat dilihat pada **Error! Reference source not found.** dan hasil dari pengujian kuesioner SUPR-Qm untuk operator restoran dapat dilihat pada Tabel 6.10.

Tabel 6.10 Hasil Pengujian Kuesioner SUPR-Qm untuk Operator Restoran

No	Pertanyaan	Skor					Total Skor
		1	2	3	4	5	
1	Aplikasi ini penting untuk saya	0	0	1	0	0	3
2	Aplikasi ini merupakan aplikasi manajemen pemesanan menu restoran terbaik yang pernah saya gunakan	0	0	0	0	1	5
3	Saya tidak tahu apakah adalah aplikasi manajemen pemesanan menu restoran yang lebih baik dari aplikasi ini	0	0	0	0	1	5
4	Saya tidak akan menghapus aplikasi ini dari smartphone saya	0	0	1	0	0	3
5	Saya akan menyarankan aplikasi ini ke teman saya	0	0	0	1	0	4
6	Saya suka mengeksplorasi fitur-fitur yang terdapat dalam aplikasi ini	0	0	0	0	1	5
7	Aplikasi ini memiliki seluruh fitur dan fungsi yang saya inginkan pada aplikasi pemesanan menu restoran	0	0	0	0	1	5
8	Saya akan sering membuka aplikasi ini setiap kali saya ingin memesan menu di restoran Wong Solo	0	0	0	0	1	5
9	Aplikasi ini menyenangkan	0	0	0	0	1	5
10	Aplikasi ini bekerja dengan baik dengan fitur-fitur lain yang ada pada *smartphone* saya (contohnya kamera)	0	0	0	0	1	5
11	Saya akan menggunakan aplikasi ini untuk manajemen pesanan menu di restoran Wong Solo	0	0	0	0	1	5
12	Desain dari aplikasi ini memudahkan saya dalam menemukan informasi yang saya inginkan	0	0	0	0	1	5
13	Menurut saya aplikasi ini menarik	0	0	0	0	1	5
14	Aplikasi ini sesuai dengan kebutuhan saya	0	0	0	0	1	5

15	Melakukan navigasi dalam aplikasi ini mudah bagi saya	0	0	0	0	1	5
16	Aplikasi ini mudah digunakan	0	0	0	0	1	5
						Total Akhir	77
						Total Nilai Maksimum	80
						Nilai SUPR-Qm	96,25%

Tabel 6.10 menunjukkan bahwa 13 pertanyaan mendapatkan skor tertinggi yaitu 5, sedangkan skor terendah ditunjukkan pada pertanyaan nomor 1 dan 4 dengan skor 3. Pada tabel di atas menunjukkan total akhir yang bernilai 329, sehingga didapatkan nilai SUPR-Qm sebesar 96,25%.

Setelah mendapatkan nilai akhir SUPR-Qm sebesar 96,25%, kemudian nilai tersebut diinterpretasi ke kategori nilai *usability*. Penginterpretasian nilai 96,25% pada kategori nilai *usability* mendapatkan kategori A dengan *adjective rating Excellent*. Sehingga dapat disimpulkan bahwa aplikasi manajemen antrean pesanan menu restoran untuk pelanggan sangat baik dan dapat diterima oleh pelanggan restoran Ayam Bakar Wong Solo.

BAB 7 PENUTUP

Pada bagian ini membahas kesimpulan dan saran terhadap penelitian pengembangan aplikasi manajemen antrean pesanan menu restoran.

7.1 Kesimpulan

Berdasarkan hasil dari analisis kebutuhan, perancangan, implementasi, dan pengujian pada penelitian pengembangan aplikasi manajemen antrean pesanan menu restoran didapatkan tiga kesimpulan, antara lain:

1. Manajemen antrean pesanan menu restoran yang sebelumnya menggunakan kertas untuk mengatur antrean pesanan menu pelanggan di restoran dapat disederhanakan dengan menggunakan aplikasi manajemen antrean pesanan menu restoran sebagai aplikasi yang mengatur antrean pesanan menu restoran melalui *smartphone*.
2. Pemesanan menu restoran yang sebelumnya menggunakan kertas untuk mencatat menu yang ingin dipesan dan harus memanggil pelayan untuk melakukan pemesanan dapat disederhanakan dengan menggunakan aplikasi manajemen antrean pesanan menu restoran sebagai aplikasi untuk melakukan pemesanan menu restoran melalui *smartphone*.
3. Berdasarkan pengujian validasi yang telah dilakukan terhadap implementasi sistem yang dibuat, didapatkan hasil bahwa validasi yang dilakukan di setiap *sprint* terhadap kebutuhan fungsionalitas dinyatakan valid.
4. Berdasarkan pengujian *usability* yang telah dilakukan kepada pengguna secara langsung, didapatkan hasil bahwa aplikasi manajemen antrean pesanan menu restoran untuk pelanggan mendapatkan nilai *ustability* sebesar 82,25% yang masuk ke dalam kategori A, dengan *adjective rating excellent*. Sedangkan untuk aplikasi manajemen antrean pesanan menu restoran untuk operator restoran mendapatkan nilai *usability* sebesar 96,25% yang masuk ke dalam kategori A, dengan *adjective rating excellent*. Sehingga dapat disimpulkan bahwa aplikasi manajemen antrean pesanan menu restoran untuk pelanggan dan restoran sangat berguna bagi pelanggan dan operator restoran.

7.2 Saran

Berdasarkan penelitian yang telah dilakukan, peneliti merasa bahwa penelitian ini belum dapat dikatakan sempurna. Oleh karena itu, didapatkan saran yang dapat dikembangkan untuk penelitian selanjutnya, antara lain:

1. Perlu ditambahkan penanganan keamanan yang lebih protektif untuk menghindari terjadinya penggunaan yang berlebih dan pesanan palsu dari pelanggan.

2. Perlu dilakukan penelitian lebih lanjut untuk mendeteksi lokasi pengguna dan memastikan keberadaan pengguna dalam menggunakan aplikasi bahwa pengguna sedang berada di restoran untuk memesan menu restoran.