PENGEMBANGAN APLIKASI MANAJEMEN ANTREAN RESTORAN DENGAN MEMANFAATKAN TEKNOLOGI KODE QR

SKRIPSI

Untuk memenuhi sebagian persyaratan   
memperoleh gelar Sarjana Komputer

Disusun oleh:

Nama YAHYA SAHAJA

NIM: 165150207111086



PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2019

PERSETUJUAN

JUDUL SKRIPSI

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

memperoleh gelar Sarjana Komputer

Disusun Oleh :

Yahya Sahaja

NIM: 165150207111086

Telah diperiksa dan disetujui oleh:

|  |  |
| --- | --- |
| Dosen Pembimbing I  Agi Putra Kharisma, S.T., M.T.  NIK: 201304 860430 1 001 | Dosen Pembimbing 2  Tri Afirianto, S.T., M.T.  NIK: 201309 851213 1 001 |

Mengetahui

Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.

NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

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

Malang, 1 Desember 2019



­

Yahya Sahaja

NIM: 165150207111086

PRAKATA

Bagian ini memuat pernyataan resmi untuk menyampaikan rasa terima kasih penulis kepada berbagai pihak yang telah membantu penyelesaian skripsi ini. Nama-nama penerima ucapan terima kasih sebaiknya dituliskan lengkap, termasuk gelar akademik, dan pihak-pihak yang tidak terkait dihindari untuk dituliskan. Bahasa yang digunakan seharusnya mengikuti kaidah bahasa Indonesia yang baku. Prakata boleh diakhiri dengan paragraf yang menyatakan bahwa penulis menerima kritik dan saran untuk pengembangan penelitian selanjutnya. Terakhir, prakata ditutup dengan mencantumkan kota dan tanggal penulisan prakata, lalu diikuti dengan kata “Penulis”.

Malang, 27 Oktober 2019

Penulis

yahya.sahaja2@gmail.com

ABSTRAK

Yahya Sahaja, Pengembangan Aplikasi Manajemen Antrean Restoran dengan Memanfaatkan Teknologi Kode QR

Pembimbing: Agi Putra Kharisma, S.T., M.T. dan Tri Afirianto, S.T., M.T.

Bagian ini diisi dengan abstrak dalam Bahasa Indonesia. Abstrak adalah uraian singkat (umumnya 200-300 kata) yang merupakan intisari dari sebuah skripsi. Abstrak membantu pembaca untuk mendapatkan gambaran secara cepat dan akurat tentang isi dari sebuah skripsi. Melalui abstrak, pembaca juga dapat menentukan apakah akan membaca skripsi lebih lanjut. Oleh karena itu, abstrak sebaiknya memberikan gambaran yang padat tetapi tetap jelas dan akurat tentang (1) apa dan mengapa penelitian dikerjakan: sedikit latar belakang, pertanyaan atau masalah penelitan, dan/atau tujuan penelitian; (2) bagaimana penelitian dikerjakan: rancangan penelitian dan metodologi/metode dasar yang digunakan dalam penelitian; (3) hasil penting yang diperoleh: temuan utama, karakteristik artefak, atau hasil evaluasi artefak yang dibangun; (4) hasil pembahasan dan kesimpulan: hasil dari analisis dan pembahasan temuan atau evaluasi artefak yang dibangun, yang dikaitkan dengan pertanyaan/tujuan penelitian.

Yang harus dihindari dalam sebuah abstrak diantaranya (1) penjelasan latar belakang yang terlalu panjang; (2) sitasi ke pustaka lainnya; (3) kalimat yang tidak lengkap; (3) singkatan, jargon, atau istilah yang membingungkan pembaca, kecuali telah dijelaskan dengan baik; (4) gambar atau tabel; (5) angka-angka yang terlalu banyak.

Di akhir abstrak ditampilkan beberapa kata kunci (normalnya 5-7) untuk membantu pembaca memposisikan isi skripsi dengan area studi dan masalah penelitian. Kata kunci, beserta judul, nama penulis, dan abstrak biasanya dimasukkan dalam basis data perpustakaan. Kata kunci juga dapat diindeks dalam basis data sehingga dapat digunakan untuk proses pencarian tulisan ilmiah yang relevan. Oleh karena itu pemilihan kata kunci yang sesuai dengan area penelitian dan masalah penelitian cukup penting. Pemilihan kata kunci juga bisa didapatkan dari referensi yang dirujuk.

Kata kunci: abstrak, skripsi, intisari, kata kunci, artefak

ABSTRACT

Student Name, Skripsi Title

Supervisors: First Supervisor’s Name and Second Supervisor’s Name

The absract of your skripsi in English is written here.

DAFTAR ISI

[PERSETUJUAN ii](#_Toc25098227)

[PERNYATAAN ORISINALITAS iii](#_Toc25098228)

[PRAKATA iv](#_Toc25098229)

[ABSTRAK v](#_Toc25098230)

[ABSTRACT vi](#_Toc25098231)

[DAFTAR ISI vii](#_Toc25098232)

[DAFTAR TABEL x](#_Toc25098233)

[DAFTAR GAMBAR xi](#_Toc25098234)

[DAFTAR LAMPIRAN xii](#_Toc25098235)

[BAB 1 PENDAHULUAN 1](#_Toc25098236)

[1.1 Latar Belakang 1](#_Toc25098237)

[1.2 Rumusan Masalah 2](#_Toc25098238)

[1.3 Tujuan 2](#_Toc25098239)

[1.4 Manfaat 2](#_Toc25098240)

[1.5 Batasan Masalah 3](#_Toc25098241)

[1.6 Sistematika Pembahasan 3](#_Toc25098242)

[BAB 2 LANDASAN KEPUSTAKAAN 5](#_Toc25098243)

[2.1 Kajian Pustaka 5](#_Toc25098244)

[2.2 Manajemen Antrean Restoran 5](#_Toc25098245)

[2.3 Aplikasi Perangkat Bergerak 6](#_Toc25098246)

[2.4 Progressive Website Application 7](#_Toc25098247)

[2.5 React JS 8](#_Toc25098248)

[2.6 QR Code 8](#_Toc25098249)

[2.7 GraphQL 9](#_Toc25098250)

[2.8 Node JS 9](#_Toc25098251)

[2.9 Model Pengembangan Perangkat Lunak 10](#_Toc25098252)

[2.9.1 Scrum 10](#_Toc25098253)

[2.10 MVVM *Design Pattern* 12](#_Toc25098254)

[2.11 Google Material Design 12](#_Toc25098255)

[2.12 Pengujian Perangkat Lunak 13](#_Toc25098256)

[*2.12.1 Blackbox Testing* 13](#_Toc25098257)

[*2.12.2 Usability Testing* 13](#_Toc25098258)

[*2.12.3 Regression Testing* 14](#_Toc25098259)

[BAB 3 METODOLOGI PENELITIAN 15](#_Toc25098260)

[3.1 Diagram Alir Metode 15](#_Toc25098261)

[3.2 Analisis Kebutuhan 15](#_Toc25098262)

[*3.2.1* Pembuatan *Backlog* Produk dan Perancangan *Sprint* 16](#_Toc25098263)

[3.3 Perancangan dan Implementasi 16](#_Toc25098264)

[3.4 Evaluasi 17](#_Toc25098265)

[3.4.1 Evaluasi dan Demonstrasi Produk 17](#_Toc25098266)

[*3.4.2* Retrospektif dan Perancangan *Sprint* Selanjutnya 17](#_Toc25098267)

[3.5 Pengujian 17](#_Toc25098268)

[3.6 Pengambilan Kesimpulan dan Saran 18](#_Toc25098269)

[BAB 4 ANALISIS KEBUTUHAN 19](#_Toc25098270)

[4.1 Gambaran Umum Sistem 19](#_Toc25098271)

[4.1.1 Deskripsi Umum Sistem 19](#_Toc25098272)

[4.1.2 Lingkungan Sistem 20](#_Toc25098273)

[4.2 Analisis Kebutuhan Perangkat Lunak 20](#_Toc25098274)

[4.2.1 Identifikasi Aktor 20](#_Toc25098275)

[4.2.2 Pembuatan User Story 21](#_Toc25098276)

[4.2.3 Kebutuhan Fungsional Sistem 22](#_Toc25098277)

[4.2.4 Kebutuhan Non-Fungsional Sistem 24](#_Toc25098278)

[4.2.5 Perancangan Durasi Sprint 24](#_Toc25098279)

[4.2.6 Pembagian Backlog Produk 25](#_Toc25098280)

[4.2.7 Use Case Diagram 27](#_Toc25098281)

[4.2.8 Use Case Scenario 29](#_Toc25098282)

[BAB 5 Perancangan dan Implementasi 39](#_Toc25098283)

[5.1 Sequence Diagram 39](#_Toc25098284)

[5.2 Subbab Lima Tiga 39](#_Toc25098285)

[5.2.1 Contoh Struktur Penelitian Implementatif Pembangunan 39](#_Toc25098286)

[5.2.2 Contoh Struktur Penelitian Nonimplementatif Eksperimental 40](#_Toc25098287)

[BAB 6 Penutup 41](#_Toc25098288)

[6.1 Kesimpulan 41](#_Toc25098289)

[6.2 Saran 41](#_Toc25098290)

[DAFTAR REFERENSI 42](#_Toc25098291)

DAFTAR TABEL

[Tabel 2.1 Keterbatasan perangkat *mobile* 7](#_Toc23840260)

[Tabel 4.1 Daftar aktor dalam Sistem 20](#_Toc23840261)

[Tabel 4.2 User story hasil wawancara Pelanggan 21](#_Toc23840262)

[Tabel 4.3 User story hasil wawancara Operator Restoran 21](#_Toc23840263)

[Tabel 4.4 Kebutuhan Fungsional Sistem 23](#_Toc23840264)

[Tabel 4.5 Backlog Produk untuk Sprint Pertama 25](#_Toc23840265)

[Tabel 4.6 Backlog Produk untuk Sprint Kedua 25](#_Toc23840266)

[Tabel 4.7 Backlog Produk untuk Sprint Ketiga 26](#_Toc23840267)

[Tabel 4.8 Skenario *Use Case* Login Pelanggan 28](#_Toc23840268)

[Tabel 4.9 Skenario *Use Case* Melakukan Pemesanan Menu 28](#_Toc23840269)

[Tabel 4.10 Skenario *Use Case* Melihat Riwayat Pemesanan Menu 29](#_Toc23840270)

[Tabel 4.11 Skenario *Use Case* Menambahkan Menu Favorit 29](#_Toc23840271)

[Tabel 4.12 Skenario *Use Case* Melihat Daftar Menu Favorit 30](#_Toc23840272)

[Tabel 4.13 Skenario *Use Case* Melihat Rekomendasi Menu Restoran 30](#_Toc23840273)

[Tabel 4.14 Skenario *Use Case* Logout Pelanggan 31](#_Toc23840274)

[Tabel 4.15 Skenario *Use Case* Login Operator Restoran 31](#_Toc23840275)

[Tabel 4.16 Skenario *Use Case* Melihat Daftar Pesanan 32](#_Toc23840276)

[Tabel 4.17 Skenario *Use Case* Mengubah Pesanan 32](#_Toc23840277)

[Tabel 4.18 Skenario *Use Case* Menghapus Pesanan 33](#_Toc23840278)

[Tabel 4.19 Skenario *Use Case* Melihat Daftar Menu Restoran 33](#_Toc23840279)

[Tabel 4.20 Skenario *Use Case* Mengubah Menu Restoran 34](#_Toc23840280)

[Tabel 4.21 Skenario *Use Case* Menambah Menu Restoran 34](#_Toc23840281)

[Tabel 4.22 Skenario *Use Case* Menghapus Menu Restoran 35](#_Toc23840282)

[Tabel 4.23 Skenario *Use Case* Mengatur Stock Menu Restoran 35](#_Toc23840283)

[Tabel 4.24 Skenario *Use Case* Mengubah Profil Restoran 36](#_Toc23840284)

[Tabel 4.25 Skenario *Use Case* Logout Operator Restoran 36](#_Toc23840285)

DAFTAR GAMBAR

[Gambar 2.1 Graphql diagram 9](#_Toc23079171)

[Gambar 2.2 Node JS flow 10](#_Toc23079172)

[Gambar 2.3 Alur Pengembangan dalam Metode Scrum 11](#_Toc23079173)

[Gambar 2.4 Pattern MVVM 12](#_Toc23079174)

[Gambar 3.1 Alur Metodologi Penelitian 15](#_Toc23079175)

DAFTAR LAMPIRAN

**No table of contents entries found.**

# PENDAHULUAN

Bagian utama skripsi terdiri dari beberapa komponen atau bab yang tersusun dengan alur yang logis. Pendahuluan merupakan komponen/bab pertama yang harus menjelaskan apa yang dikerjakan dalam skripsi dan mengapa ini dikerjakan.

## 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 yang dibagikan kepada 265 orang, 176 orang menjawab bahwa mengantri 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 menu secara manual yang membuat pihak restoran harus mencetak baru menunya. Ini juga yang menyebabkan pihak restoran harus mencetak ulang dan pelanggan juga harus menulis manual menu yang dipesan. Sebagian besar restoran juga masih memberikan menu secara manual yang membuat antrian yang sangat panjang. 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 ingin membuat aplikasi “Sans” yang diharapkan dapat mengotomatiskan sistem antrian di restoran dengan membuat sistem terpusat yang dibantu dengna 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 QR Code. *Quick Response Code* atau yang biasa disebut sebagai QR Code adalah gambar digital dua dimensi dimana dapat dengan mudah dibaca oleh kamera pada perangkat *mobile* manapun. Sekarang ini QR Code 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.

## Rumusan Masalah

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

1. Apakah kebutuhan fungsional dan non-fungsional aplikasi manajemen antrian restoran?
2. Bagaimana hasil perancangan aplikasi manajemen antrian restoran?
3. Bagaimana hasil implementasi aplikasi manajemen antrean restoran?
4. Bagaimana hasil pengujian dalam aplikasi manajemen antrean restoran?

## Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

Mengetahui apa saja kebutuhan fungsional dan non-fungsional aplikasi manajemen antrian restoran.

Mengetahui bagaimana hasil perancangan aplikasi manajemen antrian restoran.

Mengetahui bagaimana hasil implementasi aplikasi manajemen antrian restoran.

Mengetahui bagaimana hasil pengujian dalam aplikasi manajemen antrian restoran.

## Manfaat

Manfaat bagi penulis

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

Manfaat bagi peneliti selanjutnya

Dapat menjadi acuan untuk pengembangan aplikasi yang serupa.

Manfaat bagi pengguna

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

## Batasan Masalah

Pengembangan dalam penelitian ini memiliki beberapa batasan, diantaranya adalah:

Sistem yang akan dibuat menggunakan menggunakan *framework React JS.*

Sistem yang akan dibuat menggunakan konsep *Progressive Website Application.*

Sistem yang akan dibuat hanya dapat dijalankan pada *operating system* dengan *version* minimal pada *platform* Android Jelly Bean versi 4.0 dan *platform* iOS versi 9.0.

Sistem yang akan dibuat harus terkoneksi dengan internet.

Data yang diolah bersumber dari restoran Ayam Bakar Wong Solo di Malang.

Target pengguna sistem adalah pelanggan yang sering menggunakan Ayam Bakar Wong Solo sebagai tempat makan.

## Sistematika Pembahasan

Dokumen pada skripsi ini disusun menjadi beberapa bab yang terdiri dari:

Bab 1 – PENDAHULUAN

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

Bab 2 – LANDASAN KEPUSTAKAAN

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

Bab 3 – METODOLOGI PENELITIAN

Memuat alur kerja penelitian sebagai proses penyelesaian masalah dalam penelitian ini.

Bab 4 – ANALISIS KEBUTUHAN

Memuat hal-hal yang terkait seputar proses penggalian kebutuhan dalam proses pengembangan sistem.

Bab 5 – PERANCANGAN DAN IMPLEMENTASI

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

Bab 6 – PENGUJIAN

Memuat hal-hal yang berkaitan dengan pengujian sistem yang dilakukan oleh responden dan menganalisis hasil yang telah didapat. Pengujian terdiri dari pengujian *Black Box* dan analisis hasil pengujian.

Bab 7 – PENUTUP

Pada bab ini berisi kesimpulan dari keseluruhan uraian bab-bab sebelumnya. Serta saran-saran dari hasil yang diperoleh dan yang diharapkan dapat bermanfaat dalam pembelajaran selanjutnya.

# LANDASAN KEPUSTAKAAN

## Kajian Pustaka

Dalam melakukan penelitian ini, terdapat beberapa penelitian sebelumnya yang relevan dengan manajemen antrian 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 antrian panjang di restoran yang diatasi dengan membuat aplikasi pemesanan menu restoran pada restoran Cowek Ireng. Hasil dari penelitian ini dapat menunjang proses manajemen restoran menjadi lebih baik dengan pengelolaan pemesanan menu, reservasi menu dan tempat, dan pengelolaan data *user* seperti pimpinan, koki, kasir, koki, dan pelanggan.

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. Menggunakan *real-time online food ordering*, pengguna dapat dengan mudah memesan menu makanan di restoran, menelusuri proses pembuatan makanan, dan memberikan feedback kepada menu makanan dan juga restoran agar dapat meningkatkan produktivitas restoran. Inisial implementasi system dilakukan pada 2 *restaurant*/mess pada 5 area.

Penelitian tentang *Customer-Oriented Restaurant Management System* (CORMS) juga membahas tentang manajemen antrian restoran. Penelitian ini menjadikan sebuah system 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)

## Manajemen Antrean Restoran

Tingginya minat pembeli pada sebuah restoran dapat mengakibatkan panjangnya antria, khususnya pada hari libur. Antrian pada sebuah restoran seringkali membuat pelanggan tidak puas dan bahkan akhirnya memilih untuk meninggalkan restoran. Untuk mengatasi hal tersebut, perlu dilakukan manajemen antrian restoran. Manajemen antrian 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 antrian menjadi *Multichannel-Single Phase* yang mana terdapat dua jalur antrian dengan dua fasilitas pelayanan kasir. (Susila, Panji, & Prima, 2007)

Manajemen antrian 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 antrian restoran, pihak restoran dapat dengan mudah mengganti menu restorannya dan mengatur pemesanan yang ada di restoran tersebut. (Davis et al., 2017)

## 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).

## 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:

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.

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.

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.*

*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).

## 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).

## QR Code

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

Penyimpanan yang lebih besar

Variasi data yang dapat disembunyikan dalam *QR Code* adalah teks, nama, URL, SMS, email, dan kontak informasi

Dapat dilakukan *decoding* pada platform yang berbeda

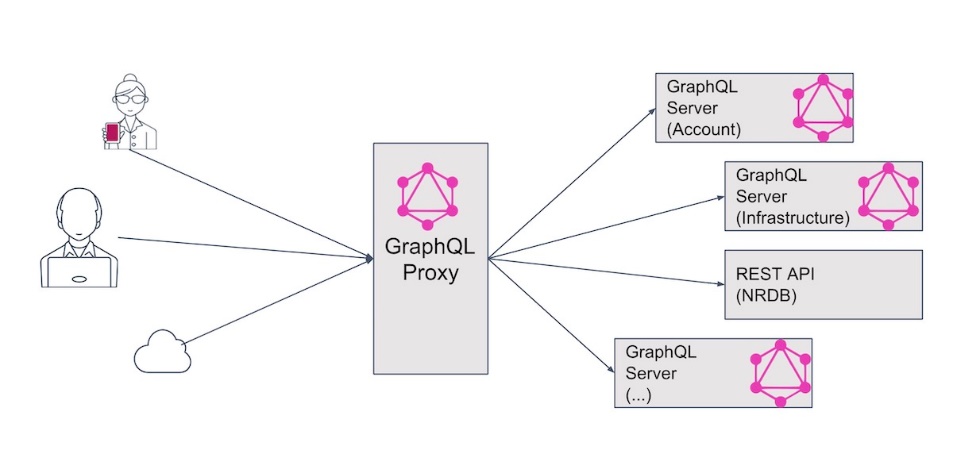
Digunakan sebagai *online payment*

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 et al., n.d.).

## 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 ahasa pemrograman *server* yang lain dengan baik (Buna, 2016).

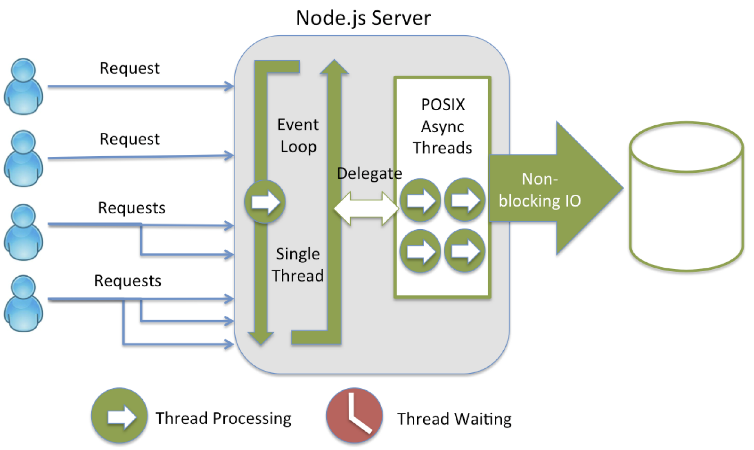


Gambar 2.1 Graphql diagram

(sumber: blog.newrelic.com)

## 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)

## Model Pengembangan Perangkat Lunak

Dalam proses pengembangan perangkat lunak, terdapat beberapa model pengembangan yang ditawarkan, salah satunya adalah model yang dipakai dalam penelitian ini yaitu Scrum. Model-model proses untuk *software engineering* seperti *model sekuensial linier* atau *model waterfall, model spiral,* *model prototype, model RAD, model inkremental, model pengembangan konkuren dan model metode formal* (Roger, 1997).

### Scrum

Pengembangan sistem manajemen antrean restoran menggunakan metode Scrum karena metode ini dapat digunakan digunakan dalam pengembangan sistem yang memiliki tingkat kompleksitas tinggi dan perubahan terus-menerus. Scrum adalah salah satu bagian dari *Agile*. Scrum memiliki beberapa karakteristik yang akan memudahkan pengembang untuk bekerja. Karakteristik dari Scrum adalah:

Memecah durasi pekerjaan menjadi beberapa bagian yang kemudian disebut dengan *sprint* (biasanya 2 minggu dalam satu *sprint*).

Merencanakan *sprint* berdasarkan kebutuhan yang penting pada saat itu.

Tidak memperkirakan waktu yang spesifik, membandingkan banyaknya pekerjaan.

Mengevaluasi *sprint* untuk melihat bagaimana *sprint* itu berjalan, dan melihat apakah ada yang bisa ditingkatkan.

Mendapatkan umpan balik dari hasil perangkat lunak yang telah dikerjakan.

Pertemuan setiap hari yang sangat singkat, mengidentifikasi halangan, dan memastikan semuanya tetap berjalan.

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 2.1.



Gambar 2.3 Alur Pengembangan dalam Metode Scrum

*Product Backlog* adalah sebuah daftar yang terdiri atas fitur-fitur yang akan diimplementasikan selama proses pengembangan aplikasi. Fitur-fitur tersebut diurutkan berdasarkan prioritas. Dalam fase *product backlog* *creation* daftar fitur-fitur dibuat berdasarkan hasil wawancara kepada calon pelanggan yang akan dijadikan *User Story*.

*Sprint Planning* merupakan tahap perencanaan *sprint* yang akan dilakukan dalam masa pengembangan aplikasi. Durasi *sprint* tersebut harus ditentukan terlebih dahulu. *Sprint* yang singkat akan membuat aplikasi tersebut rilis dengan frekuensi yang lebih sering. Setelah itu, tim akan menentukan *User Story* yang paling penting untuk diselesaikan terlebih dahulu. Setiap anggota tim harus menentukan berapa lama mereka akan menyelesaikan suatu task.

*Working on the Sprint* merupakan proses implementasi dari aplikasi yang dikembangkan. Untuk melacak kinerja tim, biasanya sebuah *Task* *Board* digunakan. *Task* *Board* berisi *User Story* yang telah direncanakan pada tahap sebelumnya. Ada beberapa kolom dalam *Task* *Board* yang bisa merepresentasikan status pengerjaan dari *User Story* yang bersangkutan seperti *To Do, In Progress, Testing,* dan *Done*.

*Testing and Product* *Demonstration* merupakan hasil dari setiap *sprint* yang telah dilakukan. Tim membuat ulasan dan mendemonstrasikan hasl pekerjaan mereka. Setelah itu *stakeholder* dapat menentukan keputusan selanjutnya yang berkaitan dengan kelanjutan pengembangan aplikasi selanjutnya.

*Retrospective* *and* *Next* *Sprint* *Planning* bertujuan untuk mendiskusikan hasil *sprint* dan menentukan cara untuk memperbaiki proses pengembangan pada *sprint* selanjutnya. Tim memberikan kesimpulan apa yang berjalan lancar dan apa yang bisa diperbaiki pada iterasi berikutnya.

## 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)

## 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:

Material adalah metafora

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

*Bold*, grafis, *intentional*

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

*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*.

Pondasi yang fleksibel

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

*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).

## Pengujian Perangkat Lunak

### *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).

### *Usability Testing*

*Usability testing* atau uji ketergunaan sebuah aplikasi merupakan salah satu 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 sejauh mana suatu produk dapat digunakan oleh yang sudah ditentukan untuk mencapai suatu tujuan tertentu sesuai dengan efektivitas, efisiensi, dan kepuasan penggunaan yang telah ditetapkan (Standard, 1998).

### *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).

# METODOLOGI PENELITIAN

## Diagram Alir Metode

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



Gambar 3.1 Alur Metodologi Penelitian

## 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. Tahap ini merupakan masa analisis kebutuhan sistem. Semua kebutuhan sistem akan dideskripsikan secara lengkap dalam fase ini. Pada tahap analisis kebutuhan terdapat dua bagian yaitu gambaran umum sistem dan analisis kebutuhan perangkat lunak. Di dalam analisis kebutuhan perangkat lunak diantaranya adalah:

Menjelaskan mengenai identifikasi aktor.

Menjelaskan mengenai pembuatan *User Story.*

Menjelaskan mengenai daftar kebutuhan fungsional.

Menjelaskan mengenai pembuatan *backlog product.*

Menjelaskan mengenai perencanaan *sprint.*

Menjelaskan mengenai analisis data.

Menjelaskan mengenai *use case diagram.*

Menjelaskan mengenai *use case* skenario.

### Pembuatan *Backlog* Produk dan Perancangan *Sprint*

Dalam fase analisis kebutuhan, hal yang pertama yang harus dilakukan adalah mendeskripsikan secara umum aplikasi yang akan dibuat agar lebih mudah dipahami. Setelah itu, menentukan target pelanggan aplikasi yang selanjutnya akan menjadi aktor dalam sistem. Kemudian, mendapatkan *User Story* dari aktor yang sudah ditentukan sebelumnya.

Setelah *User Story* didapatkan, selanjutnya adalah membuat daftar fungsionalitas sistem berdasarkan *User Story* yang didapatkan. Fungsionalitas inilah yang akan menjadi acuan utama dalam implementasi selanjutnya.

Selanjutnya adalah menentukan durasi *sprint* yang akan digunakan. Setelah mengetahui durasi *sprint*, maka *backlog* produk kemudian dibagi berdasarkan durasi *sprint* yang telah ditentukan. Durasi *sprint* ditentukan berdasarkan kompleksitas dari fungsionalitas yang dibuat.

## 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 restoran. Tahapan Perancangan Sistem diantaranya adalah :

Perancangan Arsitektur

Dalam perancangan arsitektur ini akan dilakukan pemodelan dengan menggunakan diagram UML seperti *sequence diagram* dan *class diagram*.

Perancangan Komponen

Dalam perancangan komponen ini akan dituliskan beberapa sampel algoritme utama yang diambil dari setiap klas pada *controller*. Algoritme ini akan ditulis dalam bentuk *pseudocode*

Perancangan Antarmuka

Perancangan antarmuka dari sistem yang akan dibangun ini terdiri dari tata letak komponen yang harus disediakan oleh sistem berdasarkan kebutuhan sistem. Dalam perancangan antarmuka ini akan dituliskan beberapa sampel antarmuka utama berdasarkan level pelanggan.

Pada tahap Implementasi yang dilakukan dalam menyelesaikan laporan ini menggunakan serangkaian kode program. Kode program tersebut menggunakan bahasa pemrograman Javascript. Tahapan implementasi sistem diantaranya adalah menjelaskan spesifikasi sistem, batasan impelentasi, implementasi basis data, implementasi kode program, dan implementasi antarmuka.

## Evaluasi

Setiap kali sebuah sprint telah dilakukan, evaluasi harus dilakukan untuk menentukan bahwa semua *backlog* produk telah terselesaikan secara menyeluruh atau belum. Jika ternyata masih ada *backlog* produk yang belum terselesaikan, maka *backlog* tersebut akan dimasukkan ke dalam *backlog* produk untuk *sprint* selanjutnya.

### 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*.

### Retrospektif dan Perancangan *Sprint* Selanjutnya

Setelah semua tahapan selesai dalam sebuah iterasi, maka selanjutnya dilakukan retrospektif. Retrospektif ini bertujuan untuk mengevaluasi seluruh proses *sprint* yang telah dilakukan sebelumnya untuk mengetahui hasil dan kesimpulan dari *sprint* yang telah dilakukan. Setelah itu, semua proses diulang kembali dari awal untuk iterasi selanjutnya dengan ikut mempertimbangkan hasil dan kesimpulan yang telah didapatkan dari fase retrospektif*.*

## Pengujian

Setelah semua prosespengembangan 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 kepuasan pengguna terhadap sistem yang telah dikembangkan sehingga pengguna dapat berinteraksi dengan nyaman dan puas terhadap sistem.

## Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan diambil sesuai dengan hasil dari pengujian dan analisis sehingga dapat diketahui inti dari penelitian ini. Kesimpulan yang ada akan menjadi menjawab untuk permasalahan yang telah dirumuskan dalam penelitian ini.

# 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.*

## Gambaran Umum Sistem

Tujuan dari penelitian ini adalah untuk membuat aplikasi manajemen antrean restoran dengan memanfaatkan kode QR. Gambaran umum sistem terdiri dari deskripsi umum sistem dan lingkungan sistem

### Deskripsi Umum Sistem

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

Aplikasi manajemen antrean 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 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 QR Code yang tertempel, kemudian pelanggan dapat memindai QR Code 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 QR Code 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 restoran untuk restoran dikembangkan untuk manajemen antrean pesanan di sebuah restoran. Dengan aplikasi manajemen antrean 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 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 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.

### Lingkungan Sistem

Aplikasi manajemen antrean restoran untuk pelanggan dan restoran dikembangkan menggunakan *text editor* Visual Studio Code dengan menggunakan *library* React JS dengan bahasa pemrograman Javascript. Basis data yang digunakan adalah MySQL dengan *server* yang disajikan menggunakan Node JS. Aplikasi ini nantinya diharapkan dapat berjalan pada platform *website*.

## Analisis Kebutuhan Perangkat Lunak

### Identifikasi Aktor

Aktor adalah semua orang yang terlibat di dalam sistem. Aktor akan merepresentasikan orang yang melakukan interaksi secara langsung dengan sistem. Aktor yang terlibat dalam sistem manajemen antrean 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 restoran dan belum terdaftar di dalam sistem aplikasi manajemen antrean restoran untuk pelanggan |
| Pelanggan | Pelanggan restoran yang telah terdaftar di dalam sistem aplikasi manajemen antrean restoran untuk pelanggan |
| Tamu Restoran | Orang yang belum terdaftar dalam aplikasi manajemen antrean 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 restoran untuk restoran. |

### Pembuatan User Story

Wawancara yang dilakukan kepada calon pengguna sistem akan menghasilkan *user story.* *User story* merupakan ekspektasi dari calon pengguna sistem terhadap hal-hal yang ingin dicapai. *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 restoran memiliki dua jenis pengguna, yaitu pelanggan dan operator restoran. Untuk menggali *user story* dari calonpengguna pelanggan, pertanyaan yang dilakukan dalam wawancara terhadap pelanggan difokuskan kepada pertanyaan berupa kendala 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 masing-masing dari operator restoran berupa kendala apa saja yang dialami ketika melakukan perannya masing-masing untuk mengatur penyajian makanan untuk pelanggan. *User story* yang didapatkan dari hasil wawancara yang telah dilakukan dapat dilihat pada Tabel 4.2

Tabel 4.2 User story hasil wawancara Pelanggan

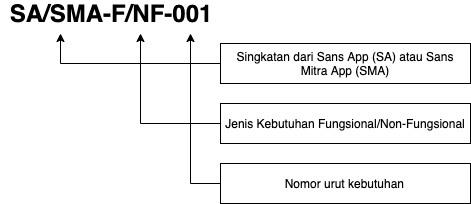
|  |  |
| --- | --- |
| No | User Story |
| 1 | Sebagai pelanggan saya ingin memesan makanan secara online dan mudah melalui aplikasi |
| 2 | Sebagai pelanggan saya ingin melakukan pembayaran makanan di restoran secara tunai ke kasir atau pembayaran online |
| 3 | Sebagai pelanggan saya ingin melihat 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* |

### Kebutuhan Fungsional Sistem

Kebutuhan fungsional sistem merupakan sebuah daftar yang mendeskripsikan kemampuan dan layanan 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 Identitas Kebutuhan



Gambar 4.1 Identitas Kebutuhan

SA/SMA melambangkan singkatan dari nama aplikasi, yaitu Sans App (SA) dan Sans Mitra App (SMA). Aplikasi Sans App adalah aplikasi manajemen antrean restoran untuk pelanggan dan aplikasi Sans Mitra App adalah aplikasi manajemen antrean 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 penngambilan gambar melalui kamera untuk pindai QR Code 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 menambahkan menu restoran yang disukai oleh pelanggan |
| 8 | SA-F-008 | Sistem harus mampu menampilkan daftar menu restoran yang disukai oleh pelanggan |
| 9 | SA-F-009 | Sistem harus mampu menampilkan daftar rekomendasi menu restoran |
| 10 | SA-F-010 | Sistem harus mampu mengeluarkan pelanggan restoran dari sistem |
| 11 | SMA-F-001 | Sistem harus mampu menyediakan tempat untuk mendaftarkan dan memasukkan tamu operator restoran ke sistem melalui akun Google operator restoran |
| 12 | SMA-F-002 | Sistem harus mampu menampilkan daftar pesanan pelanggan |
| 13 | SMA-F-003 | Sistem harus mampu mengubah pesanan pelanggan |
| 14 | SMA-F-004 | Sistem harus mampu menghapus pesanan pelanggan |
| 15 | SMA-F-005 | Sistem harus mampu menampilkan daftar menu restoran |
| 16 | SMA-F-006 | Sistem harus mampu mengubah menu restoran |
| 17 | SMA-F-007 | Sistem harus mampu menambah menu restoran |
| 18 | SMA-F-008 | Sistem harus mampu menghapus menu restoran |
| 19 | SMA-F-009 | Sistem harus mampu mengubah profil restoran |
| 20 | SMA-F-010 | Sistem harus mampu mengubah status ketersediaan menu restoran |
| 21 | SMA-F-011 | Sistem harus mampu mengeluarkan operator restoran dari sistem |

### Kebutuhan Non-Fungsional Sistem

Kebutuhan non-fungsional adalah batasan-batasan fitur atau fungsi dari sebuah sistem. Batasan-batasan tersebut adalah batasan waktu, batasan dalam proses pengembangan, 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 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 untuk menyelesaikan sebuah skenario menggunakan aplikasi ini. Dengan begitu, pengujian *usability* memiliki target untuk mendapatkan tingkat kepuasan responden yang memuaskan. |
| 2 | SA-NF-002 | *Security* | Keamanan sistem. Sistem harus mampu menjamin bahwa pengguna yang menggunakan sistem harus berada di dalam lingkungan restoran |

### Perancangan Durasi 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 berdasarkan kebutuhan fungsinoal sistem. Sebuah durasi sprint tidak boleh melebihi waktu selama 4 minggu, dikarenakan jika durasi sprint terlalu lama, prespektif akan sebuah hal yang akan dibangun bisa berubah, kompleksitas bisa meningkat, dan risiko juga 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:

Semakin pendek durasi *sprint,* maka akan cepat umpan balik yang akan diterima, sehingga dapat dilakukan perbaikan sistem lebih cepat

Membantu peneliti untuk menjadi lebih focus

Calon pengguna akan menerima respon perbaikan dari umpan balik yang telha diberikan menjadi lebih cepat melalui rilis produk *incremental* berikutnya

### Pembagian 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, berikut adalah hasil pembagian *sprint*:

Tabel 4.6 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 penngambilan gambar melalui kamera untuk pindai QR Code pada meja restoran |
| 3 | SA-F-006 | Sistem harus mampu menampilkan daftar meu restoran yangn telah dipidai pada meja restoran |
| 4 | SA-F-007 | Sistem harus mampu menampilkan daftar meu restoran yangn telah dipidai pada meja restoran |
| 5 | SA-F-008 | Sistem harus mampu membuat pesanan |

Tabel 4.7 Backlog Produk untuk Sprint Kedua

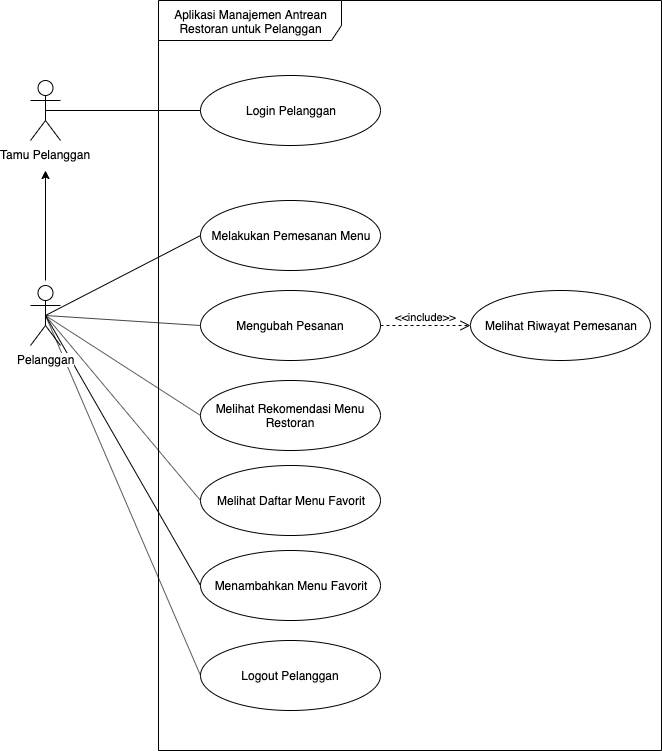
|  |  |  |
| --- | --- | --- |
| No | Kode | Deskripsi Kebutuhan |
| 1 | SMA-F-001 | Sistem harus mampu menyediakan tempat untuk mendaftarkan dan memasukkan tamu operator restoran ke sistem melalui akun Google operator restoran |
| 2 | SMA-F-002 | Sistem harus mampu menampilkan daftar pesanan pelanggan |
| 3 | SMA-F-003 | Sistem harus mampu mengubah pesanan pelanggan |
| 4 | SMA-F-004 | Sistem harus mampu menghapus pesanan pelanggan |
| 5 | SMA-F-005 | Sistem harus mampu menampilkan daftar menu restoran |
| 6 | SMA-F-006 | Sistem harus mampu mengubah menu restoran |
| 7 | SMA-F-007 | Sistem harus mampu menambah menu restoran |
| 8 | SMA-F-008 | Sistem harus mampu menghapus menu restoran |

Tabel 4.8 Backlog Produk untuk Sprint Ketiga

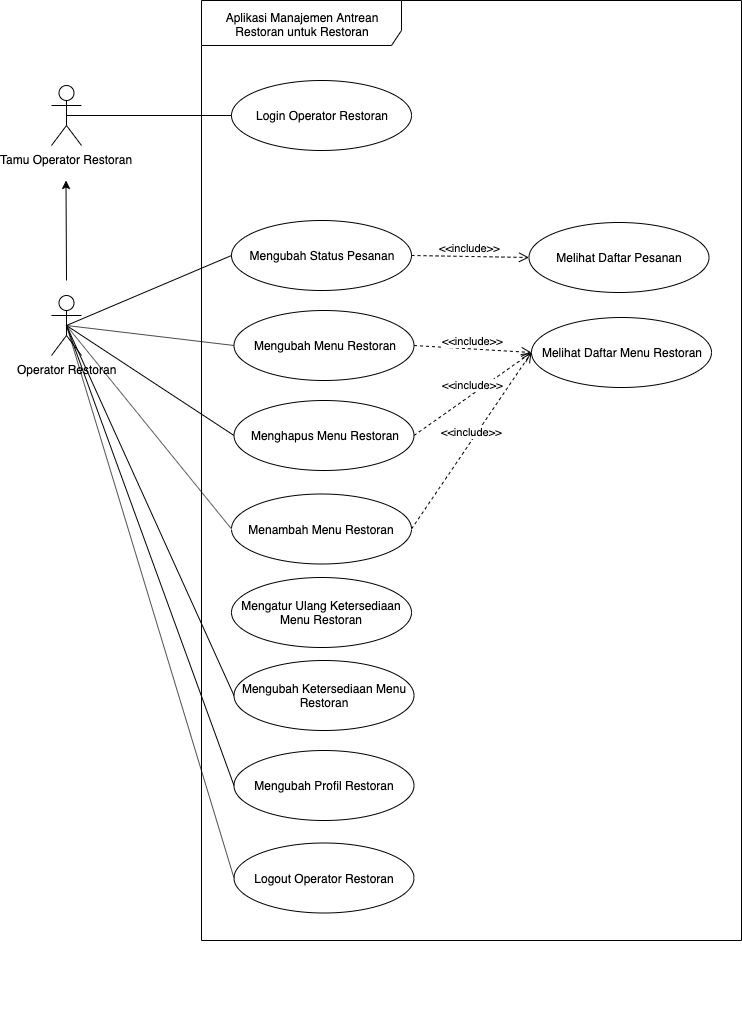
|  |  |  |
| --- | --- | --- |
| No | Kode | Deskripsi Kebutuhan |
| 1 | SA-F-006 | Sistem harus mampu menampilkan daftar riwayat pemesanan yang telah dilakukan oleh pelanggan |
| 2 | SA-F-007 | Sistem harus mampu menambahkan menu restoran yang disukai oleh pelanggan |
| 3 | SA-F-008 | Sistem harus mampu menampilkan daftar menu restoran yang disukai oleh pelanggan |
| 4 | SA-F-009 | Sistem harus mampu menampilkan daftar rekomendasi menu restoran |
| 5 | SA-F-010 | Sistem harus mampu mengeluarkan pelanggan restoran dari sistem |
| 6 | SMA-F-009 | Sistem harus mampu mengubah profil restoran |
| 7 | SMA-F-010 | Sistem harus mampu mengubah status ketersediaan menu restoran |
| 8 | SMA-F-011 | Sistem harus mampu mengeluarkan operator restoran dari sistem |

### 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



Gambar 4.2 Use case diagram aplikasi manajemen antrean restoran untuk pelanggan



Gambar 4.3 Use case diagram aplikasi manajemen antrean restoran untuk restoran

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

### Use Case Scenario

*Use Case Scenario* adalah skenario apa saja yang dapat terjadi pada sebuah Use Case. *Use Case Scenario* dari sistem manajemen antrean restoran dijabarkan pada Tabel 4.8 sampai dengan Tabel 4.25 sebagai berikut:

Tabel 4.9 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* | Sistem menampilkan halaman *Login*  Tamu pelanggan menekan tombol *Sign In With Google.*  Sistem menampilkan akun Google yang tersedia.  Tamu memilih akun Google atau menambahkan akun Google baru.  Sistem memproses otentikasi pelanggan. |
| *Alternative Flow* | - |
| *Post Condition* | Pelang Tamu pelanggan berhasil masuk ke dalam sistem dan status tamu berubah menjadi pelanggan |

Tabel 4.10 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* | Pelanggan menekan tombol *scan*  Sistem menampilkan halaman *Scan* QR Code  Pelanggan melakukan *scan* pada QR Code meja  Sistem menampilkan halaman restoran  Pelanggan memilih makanan yang hendak dipesan dengan menekan tombol *add* sebanyak jumlah yang diinginkan dan menekan tombol *checkout* apabila pesanan sudah sesuai  Sistem menampilkan halaman *checkout*  Pelanggan menekan tombol *order now* apabila pesanannya telah tepat  Sistem menampilkan pesan bahwa pesanan telah berhasil dibuat |
| *Alternative Flow* | 1. Apabila saldo pembayaran online tidak mencukupi, sistem menampilkan pesan error bahwa saldo tidak mencukupi 2. 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 4.11 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 restoran untuk pelanggan |
| *Pre-Condition* | Pelanggan sudah berada di halaman *home* |
| *Main Flow* | Pelanggan memilih halaman *orders*  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 4.12 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* | Pelanggan menekan tombol favorit yang berbentuk hati pada menu kesukaannya.  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 4.13 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* | Pelanggan memilih halaman *favorite.*  Sistem menampilkan halaman *favorite* dan daftar menu favorit. |
| *Alternative Flow* | - |
| *Post Condition* | Pelanggan melihat daftar menu favorit mereka |

Tabel 4.14 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 4.15 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* | Pelanggan memilih halaman *more*  Sistem menampilkan halaman *more*  Pelanggan menekan tombol *Logout*  Sistem menampilkan dialog konfirmasi untuk *logout*  Pelanggan memilih tombol *Logout*  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. |

Tabel 4.16 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* | Sistem menampilkan halaman *Login*  Tamu operator restoran menekan tombol *Sign In With Google.*  Sistem menampilkan akun Google yang tersedia.  Tamu memilih akun Google atau menambahkan akun Google baru.  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 4.17 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 4.18 Skenario *Use Case* Mengubah 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 memilih pesanan pelanggan yang ingin diubah 2. Sistem menampilkan formulir *dialog* untuk mengubah pesanan 3. Operator restoran menginputkan perubahan pesanan dan menekan tombol *save* 4. Sistem menyimpan perubahan pesanan |
| *Alternative Flow* | - |
| *Post Condition* | Operator restoran telah mengubah pesanan pelanggan |

Tabel 4.19 Skenario *Use Case* Menghapus Pesanan

|  |  |
| --- | --- |
| *Use Case* | Menghapus Pesanan |
| Kode terkait kebutuhan | SMA-F-004 |
| *Actor* | Operator Restoran |
| *Target* | Operator restoran dapat menghapus pesanan pelanggan |
| *Pre-Condition* | Operator restoran telah berada di halaman daftar pesanan dan daftar pesanan telah ditampilkan oleh sistem |
| *Main Flow* | 1. Operator restoran memilih pesanan pelanggan yang ingin dihapus 2. Sistem menampilkan *dialog* konfirmasi penghapusan pesanan 3. Operator restoran menekan tombol *delete* 4. Sistem menghapus pesanan pelanggan yang dipilih dan mengeluarkan pesan bahwa pesanan telah dihapus |
| *Alternative Flow* | - |
| *Post Condition* | Operator restoran telah menghapus pesanan pelanggan |

Tabel 4.20 Skenario *Use Case* Melihat Daftar Menu Restoran

|  |  |
| --- | --- |
| *Use Case* | Melihat Daftar Menu Restoran |
| Kode terkait kebutuhan | SMA-F-005 |
| *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 menu restoran 2. Sistem menampilkan halaman daftar menu restoran |
| *Alternative Flow* | - |
| *Post Condition* | Operator restoran telah melihat daftar menu restoran |

Tabel 4.21 Skenario *Use Case* Mengubah Menu Restoran

|  |  |
| --- | --- |
| *Use Case* | Mengubah Menu Restoran |
| Kode terkait kebutuhan | SMA-F-006 |
| *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 4.22 Skenario *Use Case* Menambah Menu Restoran

|  |  |
| --- | --- |
| *Use Case* | Menambah Menu Restoran |
| Kode terkait kebutuhan | SMA-F-007 |
| *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 4.23 Skenario *Use Case* Menghapus Menu Restoran

|  |  |
| --- | --- |
| *Use Case* | Menghapus Pesanan |
| Kode terkait kebutuhan | SMA-F-008 |
| *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 4.24 Skenario *Use Case* Mengatur Stock Menu Restoran

|  |  |
| --- | --- |
| *Use Case* | Mengatur Stock Menu Restoran |
| Kode terkait kebutuhan | SMA-F-009 |
| *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 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 stock menu restoran |
| *Alternative Flow* | - |
| *Post Condition* | Operator restoran telah mengubah ketersediaan stock menu restoran |

Tabel 4.25 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 4.26 Skenario *Use Case* Logout Operator Restoran

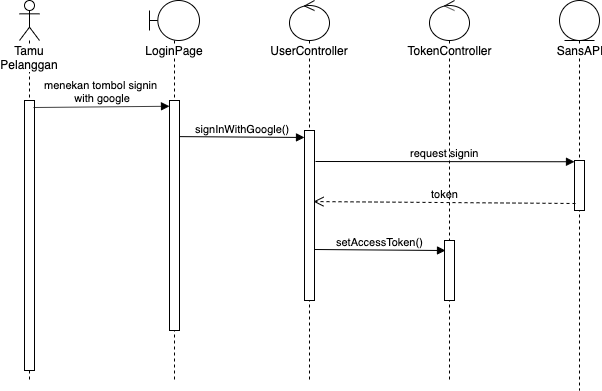
|  |  |
| --- | --- |
| *Use Case* | Logout Pelanggan |
| Kode kebutuhan terkait | SA-F-010 |
| *Actor* | Operator Restoran |
| *Target* | Otentikasi operator restoran dapat keluar dari sistem. |
| *Pre-Condition* | Pelanggan sudah berada di halaman *dashboard* |
| *Main Flow* | Pelanggan menekan tombol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

# Perancangan dan Implementasi

Bab ini membahas tentang perancangan dan implementasi sistem. Tujuan dilakukan perancangan adalah untuk membuat gambaran program sebelum diimplementasikan. Sehingga nantinya akan dijadikan rujukan pada fase implementasi sistem. Tahapan dari perancangan sistem terbagi menjadi beberapa tahap, yaitu membuat *Sequence Diagram, Class Diagram,* rancangan *Database*, dan diakhiri dengan membuat rancangan antarmuka.

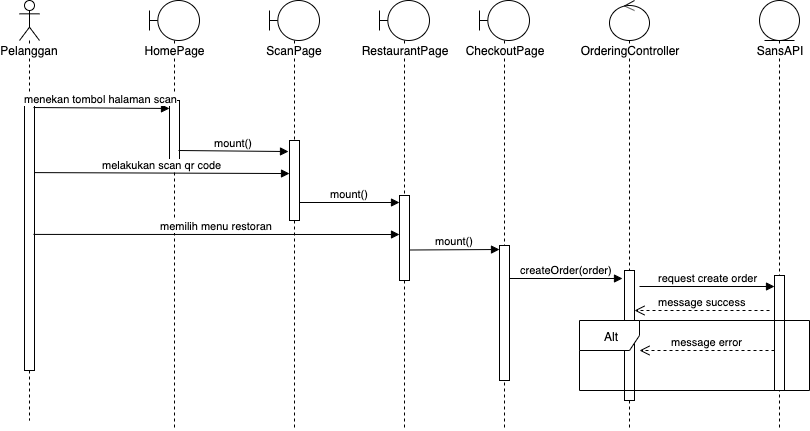
## *Sequence Diagram*

*Sequence diagram* menggambarkan urutas proses yang terjadi di dalam sistem untuk mencapai tujuan dari kebutuhan fungsionalitas sistem. Berikut adalah daftar *sequence diagram* dalam sistem manajemen antrean restoran. //cross reference diagram



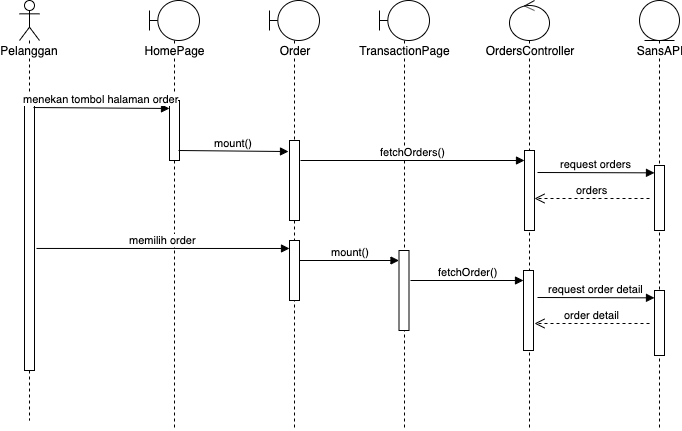
Gambar 5.1 *Sequence Diagram* Use Case Login Pelanggan

Pada *sequence diagram* Login Pelanggan pada Gambar 5.1 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*.



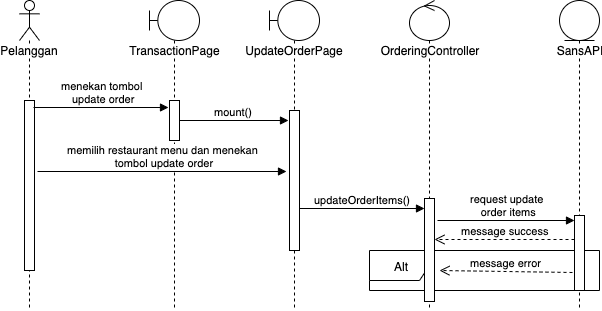
Gambar 5.2 *Sequence Diagram* Use Case Melakukan Pemesanan Menu

Pada *sequence diagram* Melakukan Pemesanan Menu pada Gambar 5.2 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 SansAPI 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*.



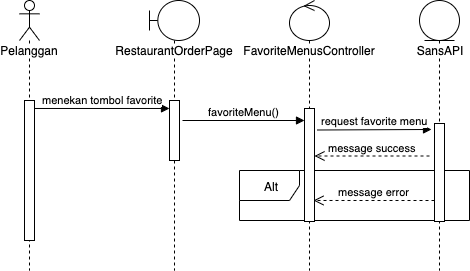
Gambar 5.3 *Sequence Diagram* Use Case Melihat Riwayat Pemesanan Menu

Pada *sequence diagram* Melihat Riwayat Pemesanan Menu pada Gambar 5.3 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 SansAPI. Kemudian didapatkan nilai kembalian berupa *order detail* yang datanya disimpan di dalam *observable property order*.



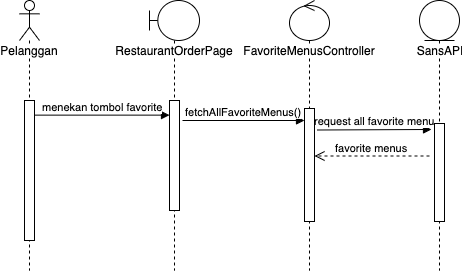
Gambar 5.4 *Sequence Diagram* Use Case Mengubah Pesanan

Pada *sequence diagram* Mengubah Pesanan pada Gambar 5.4 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*.



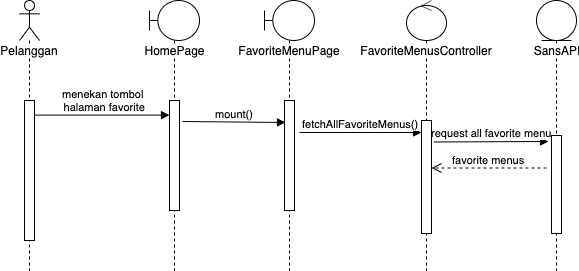
Gambar 5.5 *Sequence Diagram* Use Case Menambah Menu *Favorite*

Pada *sequence diagram* Menambah Menu *Favorite* pada Gambar 5.5 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.



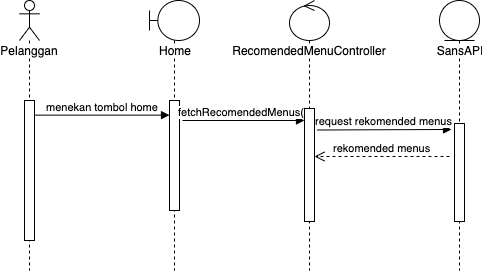
Gambar 5.6 *Sequence Diagram* Use Case Menambah Menu *Favorite*

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.



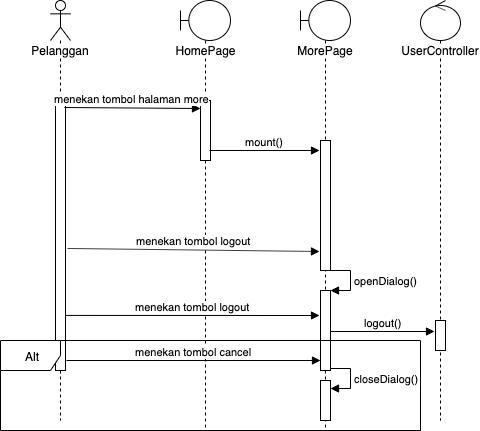
Gambar 5.7 *Sequence Diagram* Use Case Melihat Daftar Menu *Favorite*

Pada *sequence diagram* Melihat Daftar Menu *Favorite* pada Gambar 5.7 dijelaskan bahwa aktor Pelanggan menekan tombol halaman menu *favorite* berupa gambar hati pada HomePage. Kemudian dilakukan *mounting component* pada halaman RestaurantOrderPage. Setelah itu akan dilakukan pemanggilan fungsi *fetchAllFavoriteMenus()* pada FavoriteMenusController. Aktivitas tersebut memicu *request all favorite menu* ke Sans API, kemudian mendapatkan nilai kembalian berupa daftar *favorite menus*.



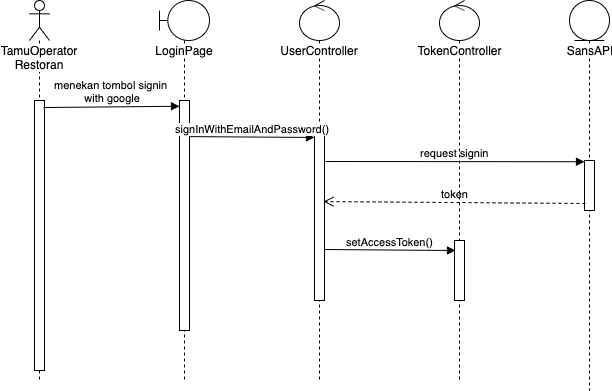
Gambar 5.8 *Sequence Diagram* Use Case Melihat Daftar Menu Rekomendasi

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*.



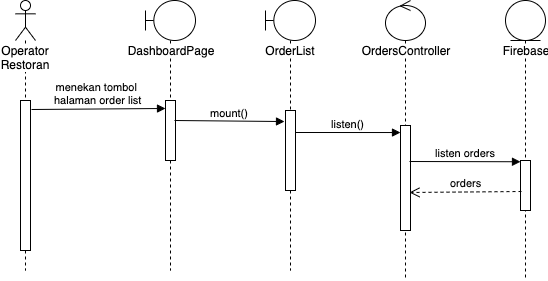
Gambar 5.9 *Sequence Diagram* Use Case Logout Pelanggan

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()*.



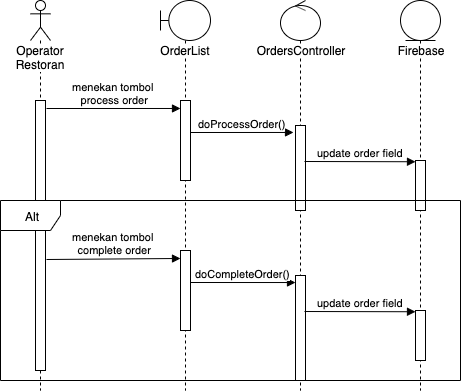
Gambar 5.10 *Sequence Diagram* Use Case Login Restaurant

Pada *sequence diagram* Login Restaurant pada Gambar 5.10 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*.



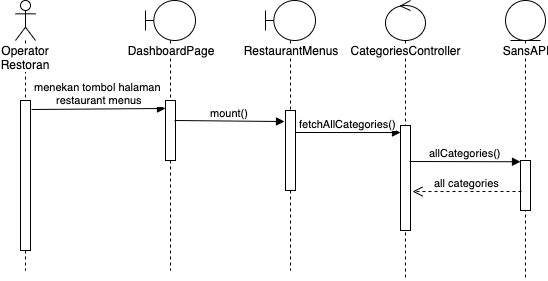
Gambar 5.11 *Sequence Diagram* Use Case Melihat Daftar Pesanan

Pada *sequence diagram* Melihat Daftar Pesanan pada Gambar 5.11 dijelaskan bahwa aktor operator restaurant 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.



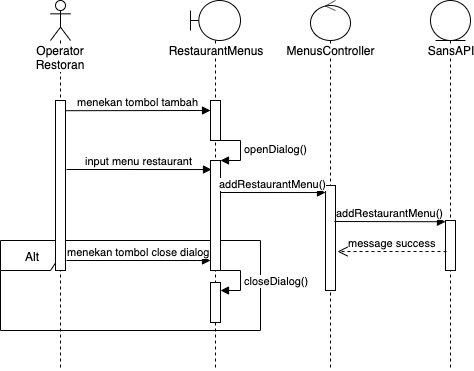
Gambar 5.12 *Sequence Diagram* Use Case Mengubah Status Pesanan

Pada *sequence diagram* Mengubah Status Pesanan pada Gambar 5.12 dijelaskan bahwa aktor operator restaurant 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.



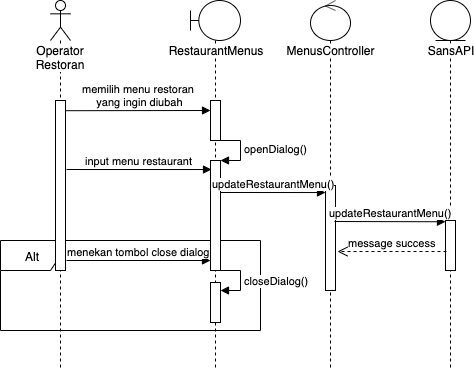
Gambar 5.13 *Sequence Diagram* Use Case Melihat Daftar Menu Restoran

Pada *sequence diagram* Melihat Menu Restoran pada Gambar 5.13 dijelaskan bahwa aktor operator restoran menekan tombol halaman restaurant menus pada halaman DashboardPage. Kemudian dilakukan *mounting component* pada RestaurantMenu. Setelah itu dilakukan pemanggilan fetchAllCategories() pada CategoriesController. Aktivitas tersebut memicu *request all categories* ke SansApi dan mendapatkan nilai kembalian berupa all categories yang disimpan dalam *observable property.*



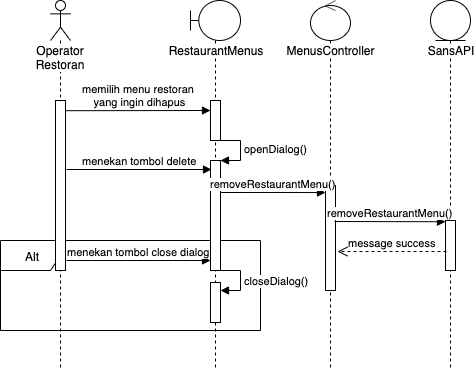
Gambar 5.14 *Sequence Diagram* Use Case Menambah Menu Restoran

Pada *sequence diagram* Menambah Menu Restoran pada Gambar 5.14 dijelaskan bahwa aktor operator restoran menekan tombol tambah pada halaman RestaurantMenus. Kemudian dilakukan *self-call* openDialog(). Setelah itu operator restoran input menu restoran. Aktivitas tersebut memicu pemanggilan addRestaurantMenu() pada MenusController dan melakukan *request add restaurant menu* pada SansAPI, kemudian mendapatkan nilai kembalian berupa *message success*. Apabila operator restoran menekan tombol *close dialog*, maka dilakukan *self-call* closeDialog() pada RestaurantMenus.



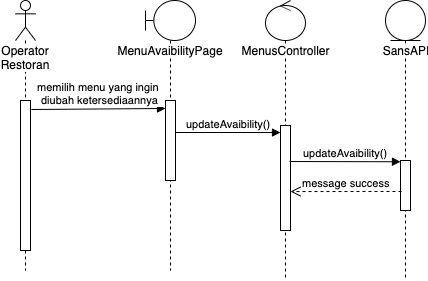
Gambar 5.15 *Sequence Diagram* Use Case Mengubah Menu Restoran

Pada *sequence diagram* Menambah Menu Restoran pada Gambar 5.15dijelaskan bahwa aktor operator restoran memilih menu yang ingin diubah pada halaman RestaurantMenus. Kemudian dilakukan *self-call* openDialog(). Setelah itu operator restoran input menu restoran. Aktivitas tersebut memicu pemanggilan updateRestaurantMenu() pada MenusController dan melakukan *request update restaurant menu* pada SansAPI, kemudian mendapatkan nilai kembalian berupa *message success*. Apabila operator restoran menekan tombol *close dialog*, maka dilakukan *self-call* closeDialog() pada RestaurantMenus.



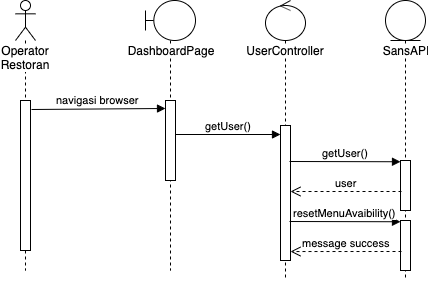
Gambar 5.16 *Sequence Diagram* Use Case Menghapus Menu Restoran

Pada *sequence diagram* Menghapus Menu Restoran pada Gambar 5.16 dijelaskan bahwa aktor operator restoran memilih menu yang ingin dihapus pada halaman RestaurantMenus. Kemudian dilakukan *self-call* openDialog(). Setelah itu operator restoran input menu restoran. Aktivitas tersebut memicu pemanggilan removeRestaurantMenu() pada MenusController dan melakukan *request remove restaurant menu* pada SansAPI, kemudian mendapatkan nilai kembalian berupa *message success*. Apabila operator restoran menekan tombol *close dialog*, maka dilakukan *self-call* closeDialog() pada RestaurantMenus.



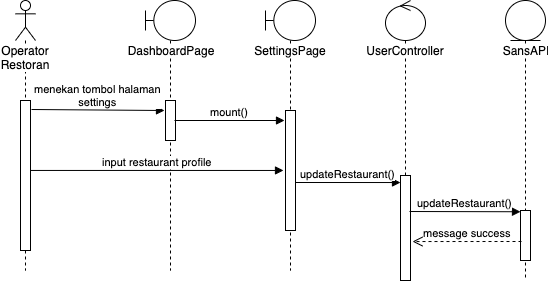
Gambar 5.17 *Sequence Diagram* Use Case Mengubah Ketersediaan Menu Restoran

Pada *sequence diagram* Mengubah Ketersediaan Menu Restoran pada Gambar 5.17 dijelaskan bahwa aktor operator restoran memilih menu yang ingin diubah ketersediaannya pada halaman MenuAvaibilityPage. Kemudian dilakukanpemanggilan fungsi updateAvaibility() pada MenusController. Setelah itu dilakukan *request update menu avaibility* dan mendapatkan nilai kembalian *message success*.



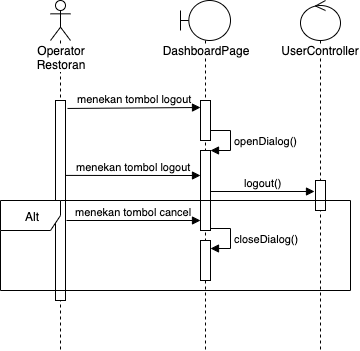
Gambar 5.18 *Sequence Diagram* Use Case Mengatur Ulang Ketersediaan Menu Restoran

Pada *sequence diagram* Mengatur Ulang Ketersediaan Menu Restoran pada Gambar 5.18 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 SansAPI. Nilai kembalian yang didapat berupa user yang disimpan dalam *observable property*. Kemudian dilakukan *request* untuk mengatur ulang ketersediaan menu ke SansAPI dan mendapatkan nilai kembalian *message success*.



Gambar 5.19 *Sequence Diagram* Use Case Mengubah Profil Restoran

Pada *sequence diagram* Mengubah Profil Restoran pada Gambar 5.19 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 restaurant* ke SansAPI dan mendapatkan nilai kembalian *message success*.



Gambar 5.9 *Sequence Diagram* Use Case Logout Operator Restoran

Pada *sequence diagram* Logout Operator Restoran pada Gambar 5.9 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()*.

## Class Diagram

## Perancangan Basis Data

## Perancangan Algoritme

## Perancangan Antarmuka

## Subbab Lima Tiga

Contoh struktur skripsi untuk implementatif pembangunan dan nonimplementatif eksperimental dapat dilihat pada kedua subbab berikut.

### Contoh Struktur Penelitian Implementatif Pembangunan

Berikut ini adalah contoh bab-bab yang terdapat pada penelitian implementatif pembangunan sistem perangkat lunak.

Bab 1 Pendahuluan

Bab 2 Landasan Kepustakaan

Bab 3 Metodologi Penelitian

Bab 4 Persyaratan

Bab 5 Perancangan dan Implementasi

Bab 6 Pengujian

Bab 7 Penutup

Bab 1 sampai Bab 3 memuat informasi yang sesuai dengan panduan sebelumnya. Isi dari bab-bab berikutnya:

* Bab 4 Persyaratan:
* Pernyataan masalah (problem statement), yang lebih elaboratif daripada yang di Pendahuluan.
* Identifikasi pemangku kepentingan (stakeholders) dan aktor (actors) sistem.
* Daftar terstruktur persyaratan/kebutuhan perangkat lunak, secara fungsional, data, dan non-fungsional
* Use cases, use case diagrams, dan use case specifications, dan sebagainya.
* Bab 5 Perancangan dan Implementasi:
* Rancangan arsitektur: deskripsi struktur dan setiap komponen utama
* Representasi data dalam model data dan basis data
* Detil implementasi dari fungsi-fungsi utama yang menjadi fokus
* Bab 6 Pengujian dan Evaluasi
* Strategi, rencana, kasus, dan data pengujian
* Ringkasan hasil pengujian perangkat lunak, termasuk data dan analisisnya (detilnya di Lampiran)
* Evaluasi hasil proyek secara keseluruhan, misalkan
* Bab 7 Penutup
* Ringkasan dari capaian proyek
* Saran pengembangan lebih lanjut

Pada contoh struktur ini “hasil” tersebar di beberapa bab mulai Bab 4 Persyaratan sampai Bab 6, sedangkan “pembahasan” secara keseluruhan terhadap masalah penelitian terdapat di Bab 6. Yang dimaksud dengan pengujian dalam Bab 6 terfokus pada pengujian persyaratan perangkat lunak, sedangkan evaluasi berfungsi sebagai “pembahasan” secara keseluruhan, yaitu menentukan apakah “hasil” sudah menjawab masalah penelitian yang dirumuskan pada Bab 1.

Sebagai catatan, Bab 3 Metodologi umumnya menjelaskan model proses perangkat lunak yang digunakan. Jika strategi untuk setiap aktivitasnya (analisis persyaratan, perancangan, dan seterusnya) sudah dijelaskan di Bab 3 ini juga, maka bab-bab lainnya yang berhubungan dengan aktivitas-aktivitas ini masing-masing langsung dapat menjelaskan hasil pelaksanaan metodenya.

### Contoh Struktur Penelitian Nonimplementatif Eksperimental

Berikut ini adalah contoh bab-bab yang terdapat pada penelitian implementatif pembangunan sistem perangkat lunak.

Bab 1 Pendahuluan

Bab 2 Landasan Kepustakaan

Bab 3 Metodologi Penelitian

Bab 4 Hasil

Bab 5 Pembahasan

Bab 6 Penutup

Isi dari setiap bab dapat menyesuaikan dengan panduan yang telah dijelaskan sebelumnya. Jika diperlukan, Bab 4 dapat digabungkan dengan Bab 5, menjadi Hasil dan Pembahasan.

Struktur dasar ini cukup universal sehingga dapat digunakan juga untuk tipe-tipe penelitian lainnya, khususnya jika belum ada struktur lain yang lebih tematik dan cocok untuk penelitian yang bersangkutan.

# Penutup

Bagian ini memuat kesimpulan dan saran terhadap skripsi. Kesimpulan dan saran disajikan secara terpisah, dengan penjelasan sebagai berikut:

## Kesimpulan

Kesimpulan merupakan pernyataan-pernyataan yang singkat, jelas, dan tepat tentang hasil penelitian yang diperoleh berdasarkan tujuannya. Bagian ini merupakan penegasan dari yang telah dijelaskan pada bagian Pembahasan dan tidak memuat informasi yang baru. Bagian ini juga mencerminkan jawaban dari rumusan masalah (pertanyaan penelitian).

## Saran

Saran berisi pernyataan-pernyataan yang ringkas dan jelas tentang masalah-masalah atau hal-hal yang dapat dilakukan untuk mengembangkan penelitian ini lebih lanjut. Saran itu dapat diarahkan pada aspek metode, instrumen, populasi/sampel, dan sebagainya.

DAFTAR REFERENSI

Ayuliana. (2009). Testing dan Implementasi.

BPS. (2017). *Statistik Restoran/Rumah Makan 2015*.

Buna, S. (2016). *Learning Graphql and Relay*. Birmingham: Packt Publishing Ltd.

Cata, T., Patel, P. S., & Sakaguchi, T. (2013). *Article ID 813339, 7 pages Cancer*. *2013*. https://doi.org/10.5171/2013

Davis, R., Francis, N., Sukumaran, S. K., E, S. J., & Nair, U. (2017). Corms : an Automated Restaurent Management System. *International Research Journal of Engineering and Technology(IRJET)*, *4*(3), 2860–2867. Retrieved from https://irjet.net/archives/V4/i3/IRJET-V4I3720.pdf

Developer, G. (2019). Material Design Introduction. Retrieved from https://material.io/design/introduction/

Henriyadi, H., & Mulyati, R. (2016). USABILITY TESTING Sistem Informasi: Studi kasus pada Aplikasi Repositori Publikasi Badan Penelitian dan Pengembangan Pertanian. *Jurnal Perpustakaan Pertanian*, *23*(2), 54. https://doi.org/10.21082/jpp.v23n2.2014.p54-63

Islam, R., & Mazumder, T. (2010). Mobile application and its global impact. *International Journal of Engineering & …*, (06), 72–78. Retrieved from http://ijens.org/107506-0909 IJET-IJENS.pdf

Karpagam, V. (2017). Performance Enhancement of Webpage Using Progressive Web App Features. *International Journal of Innovative Research in Advanced Engineering*, *03*(4), 2349–2163.

Kiessling, M. (2015). *The Node Beginner Book: A Comprehensive Node.js Tutorial*.

Kouraklis, J. (2016). MVVM in delphi: Architecting and building model view viewmodel applications. *MVVM in Delphi: Architecting and Building Model View ViewModel Applications*, (October 2016), 1–143. https://doi.org/10.1007/978-1-4842-2214-0

Kumar, A., & Singh, R. K. (2016). Comparative analysis of angularjs and reactjs. *International Journal of Latest Trends in Engineering and Technology*, *7*(4), 225–227. https://doi.org/10.21172/1.74.030

Li, X., Chang, D., Pen, H., Zhang, X., Liu, Y., & Yao, Y. (2015). Application of MVVM design pattern in MES. *2015 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, IEEE-CYBER 2015*, (2012), 1374–1378. https://doi.org/10.1109/CYBER.2015.7288144

Marsum, W. (2005). *Restoran dan Segala Permasalahannya*. Jakarta: Andi.

Oinas-Kukkonen, H., & Kurkela, V. (2003). Developing successful mobile applications. *Proceedings of the IASTED International Conference on Computer Science and Technology*, (January 2003), 50–54.

Ozdenizci, B., Aydin, M. N., Coskun, V., & Ok, K. (n.d.). *Department of Information Technologies , ISIK University , Istanbul , Turkey { busraozdenizci , mnaydin , vedatcoskun , keremok }@ isikun . edu . tr*.

Rasid, A., Supriyono, S., & Setiawan, R. (2018). Aplikasi Reservasi Menu Restoran Berbasis Web Dan Mobile Android Di Cowek Ireng. *SITECH : Jurnal Sistem Informasi Dan Teknologi*, *1*(1), 21–30. https://doi.org/10.24176/sitech.v1i1.2273

Rosa, A., & Salahuddin, M. (2011). *Modul Pembelajaran Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek)*. Bandung: Modula.

Schwaber, K., & Sutherland, J. (2017). *2017 Scrum Guide*. *19*(6), 504. https://doi.org/10.1053/j.jrn.2009.08.012

Singh, A., & Kanade, V. (2018). *Online food ordering system*. 374–378.

Standard, I. (1998). *Iso 9241-11*. *1998*.

Susila, A. nur, Panji, D., & Prima, D. W. A. (2007). *Analisa Sistem Antrian Untuk Menentukan Tingkat Pelayanan Yang Optimal Pada Kasir (Server) Rumah Makan Kober Mie Setan Malang Dengan Metode Simulasi*. *23*(39870423), 946–952. https://doi.org/10.13989/j.cnki.0517-6611.2015.10.011

Vaduva, I., Baltac, V., Florescu, V., Floricica, I., & Jitaru, M. (1983). Software Engineering (Ii). In *Economic Computation and Economic Cybernetics Studies and Research* (Vol. 18). https://doi.org/10.1109/MC.1984.1659001

Vanus, J., Belesova, J., Martinek, R., Bilik, P., Zidek, J., & Koval, L. (2016). Development of Software Tool for Operational and Technical Functions Control in the Smart Home with KNX technology. *IFAC-PapersOnLine*, *49*(25), 431–436. https://doi.org/10.1016/j.ifacol.2016.12.088

Xiaowen, L. (2013). Research on regression testing methods for industry applications. *International Journal of Smart Home*, *7*(6), 111–122. https://doi.org/10.14257/ijsh.2013.7.6.11