

**PENGEMBANGAN APLIKASI MANAJEMEN ANTREAN  
PESANAN MENU 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**

**PENGEMBANGAN APLIKASI MANAJEMEN ANTREAN PESANAN MENU RESTORAN  
DENGAN MEMANFAATKAN KODE QR**

**SKRIPSI**

**Diajukan untuk memenuhi sebagian persyaratan  
memperoleh gelar Sarjana Komputer**

**Disusun Oleh :  
Yahya Sahaja  
NIM: 165150207111086**

**Telah diperiksa dan disetujui oleh:**

**Dosen Pembimbing I**

**Dosen Pembimbing 2**

**Agi Putra Kharisma, S.T., M.T.  
NIK: 201304 860430 1 001**

**Tri Afirianto, S.T., M.T.  
NIK: 201309 851213 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 Maret 2019

Yahya Sahaia

NIM: 165150207111086

## **PRAKATA**

Segala puji dan syukur bagi Allah SWT atas limpahan rahmat dan hidayah-Nya penulis dapat menyelesaikan skripsi yang berjudul “Pengembangan Aplikasi Mobile Pendeteksi Penyakit pada Tanaman Cabai dengan Menggunakan Ximilar Custom Image Recognition”. Dalam kesempatan ini, penulis ingin memberikan ucapan terima kasih kepada seluruh pihak yang telah membantu penulis dalam proses penyusunan skripsi ini, diantaranya:

1. Kedua orang tua penulis, yaitu Bapak Hari Prihatin dan Ibu Sri Nastiti beserta seluruh keluarga besar yang terus memberikan dukungan dan motivasi serta doa yang tidak kunjung putus.
2. Bapak Issa Arwani, S.Kom., M.Sc. sebagai dosen pembimbing I dan Bapak Ir. Handoko, M.Sc. sebagai dosen pembimbing II atas segala bimbingan, masukan, dan saran yang bermanfaat bagi penulis selama proses penelitian skripsi ini.
3. Seluruh Bapak dan Ibu Dosen Fakultas Ilmu Komputer, Universitas Brawijaya atas ketesediannya membimbing dan memberikan ilmu yang bermanfaat kepada penulis.
4. Seluruh teman dan sahabat penulis yang secara tidak langsung memberikan ilmu serta motivasi dalam penyusunan skripsi ini.
5. Semua pihak yang tidak semuanya bisa dituliskan disini yang terlibat secara langsung maupun tidak langsung dalam penyusunan skripsi ini.

Penulis sadar masih banyak kekurangan dan kesalahan yang terdapat dalam penyusunan skripsi ini baik dari sisi teknis penelitian maupun penyajian materi. Maka dari itu, penulis mengharapkan kritik dan saran yang membangun dari pembaca. Semoga karya tulis ini bermanfaat dan dapat memberikan manfaat bagi pihak yang membutuhkan.

Malang, 1 Maret 2020

Penulis

yahya.sahaja2@gmail.com

## ABSTRAK

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

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

Permasalahan yang terjadi di sebuah restoran dalam melayani pelanggan sangat beragam. Kebanyakan Restoran di Malang masih memberikan daftar menu secara manual kepada pelanggannya yang membuat pihak restoran harus mencetak baru menunya setiap kali terdapat perubahan daftar menu restoran. Selain itu, pemesanan menu restoran secara manual masih menggunakan kertas, sehingga pelanggan harus menuliskan menu yang ingin dipesan melalui kertas yang diberikan oleh pelayan restoran. Hal ini akan membuang-buang waktu dan tenaga para pelanggan. Dengan keterbatasan karyawan restoran dan jumlah pelanggan yang sangat ramai membuat pelayanan di restoran tersebut sangat lama. Berdasarkan permasalahan tersebut, penulis membuat aplikasi manajemen antrean pesanan menu restoran yang diharapkan dapat mengotomatiskan sistem antrean di restoran. Aplikasi yang dikembangkan berupa dua aplikasi, yaitu aplikasi untuk pelanggan dan aplikasi untuk operator restoran. Penelitian ini menggunakan metode pengembangan perangkat lunak SCRUM dikarenakan kebutuhan pengguna yang memiliki banyak perubahan di setiap fase pengembangan. Hasil dari penelitian ini berfokus pada penyelesaian masalah dengan menyederhanakan proses manajemen antrean pesanan menu restoran dan menyederhanakan pemesanan menu restoran. Selain itu, peneliti juga melakukan dua pengujian, yaitu pengujian validasi dan *usability* untuk mengetahui tingkat kebergunaan aplikasi bagi pengguna. Dari pengujian validasi didapatkan hasil bahwa validasi yang dilakukan di setiap *sprint* terhadap kebutuhan fungsionalitas dinyatakan valid, sedangkan dari hasil pengujian *usability* didapatkan hasil bahwa aplikasi untuk pelanggan mendapatkan nilai *usability* sebesar 82,25% yang masuk ke dalam kategori A, dengan *adjective rating excellent*. Sedangkan aplikasi untuk operator restoran mendapatkan nilai *usability* sebesar 96,25% yang masuk ke dalam kategori A, dengan *adjective rating excellent*.

Kata kunci: manajemen restoran, pemesanan menu restoran, SCRUM, *progressive website application, pwa, usability*.

## ABSTRACT

**Yahya Sahaja, Restaurant Menu Ordering Application Development with Utilizing QR Code Technology**

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

*The problems happen in a restaurant for serving customers are too diverse. Many of restaurants in Malang are still just giving the menu list manually to the customers, that the restaurant should print the updated menu. Besides, ordering in a restaurant is still just using paper, that the customers should write the menu manually to order, and give it back to the restaurant waiters. The impact is that it can take too much time and costs more energy to the waiter as well as handling the restaurant menu ordering, even if there are so many customers should be served. Based on those problems, the writer wants to make an application that can handle restaurant food ordering queue management, hopefully this application could automate queue system in a restaurant. This application is built with following SCRUM framework methodology, because the requirements are still developing within the application development. The result of this research was focused to solve the problems as well as simplify the queue management process and menu ordering in a restaurant. The researcher would also test the system with validation and usability test. The usability test was done for determining the usability level as the customers view point. For the result, validation testing states that the validations for every sprint is valid and usability testing states that the usability value for customers application got 82,25% score belonging in an A category with adjective rating excellent, whereas the application for restaurant operator got 96.25% belonging as an A category with adjective rating excellent.*

Kata kunci: restaurant management, restaurant ordering, SCRUM framework, progressive website application, pwa, usability testing.

## DAFTAR ISI

PERSETUJUAN .....	ii
PERNYATAAN ORISINALITAS .....	iii
PRAKATA .....	iv
ABSTRAK.....	v
ABSTRACT .....	vi
DAFTAR ISI .....	vii
DAFTAR TABEL.....	x
DAFTAR GAMBAR.....	xiii
DAFTAR LAMPIRAN .....	xvi
BAB 1 PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	2
1.3 Tujuan .....	2
1.4 Manfaat.....	2
1.5 Batasan Masalah .....	3
1.6 Sistematika Pembahasan .....	3
BAB 2 LANDASAN KEPUSTAKAAN .....	5
2.1 Kajian Pustaka .....	5
2.2 Manajemen Antrean Pesanan Menu Restotoran .....	5
2.3 Aplikasi Perangkat Bergerak .....	6
2.4 Progressive Website Application .....	7
2.5 React JS .....	8
2.6 Kode QR .....	8
2.7 GraphQL.....	9
2.8 Node JS.....	9
2.9 Model Pengembangan Perangkat Lunak.....	10
2.9.1 Scrum .....	10
2.10 MVVM <i>Design Pattern</i> .....	12
2.11 Google Material Design .....	12
2.12 Pengujian Perangkat Lunak .....	13

2.12.1 <i>Blackbox Testing</i> .....	13
2.12.2 <i>Usability Testing</i> .....	13
2.12.3 <i>Regression Testing</i> .....	14
BAB 3 METODOLOGI PENELITIAN .....	15
3.1 Jenis Penelitian .....	15
3.2 Tempat dan Waktu Penelitian .....	15
3.3 Diagram Alir Metode .....	15
3.4 Analisis Kebutuhan .....	16
3.4.1 Pembuatan <i>Backlog</i> Produk dan Perancangan <i>Sprint</i> .....	17
3.5 Perancangan dan Implementasi .....	17
3.6 Evaluasi .....	18
3.6.1 Evaluasi dan Demonstrasi Produk.....	18
3.6.2 Retrospektif dan Perancangan <i>Sprint</i> Selanjutnya .....	18
3.7 Pengujian .....	18
3.8 Pengambilan Kesimpulan dan Saran .....	18
BAB 4 ANALISIS KEBUTUHAN .....	19
4.1 Gambaran Umum Sistem.....	19
4.1.1 Deskripsi Umum Sistem .....	19
4.1.2 Lingkungan Sistem .....	20
4.2 Analisis Kebutuhan Perangkat Lunak.....	20
4.2.1 Identifikasi Aktor .....	20
4.2.2 Pembuatan User Story .....	21
4.2.3 Kebutuhan Fungsional Sistem.....	22
4.2.4 Kebutuhan Non-Fungsional Sistem.....	24
4.2.5 Use Case Diagram .....	25
BAB 5 PENGEMBANGAN METODOLOGI SCRUM .....	29
5.1 Perancangan Basis Data.....	29
5.2 Spesifikasi Lingkungan Sistem Implementasi .....	31
5.2.1 Spesifikasi Perangkat Keras.....	31
5.2.2 Spesifikasi Perangkat Lunak .....	31
5.3 Batasan Implementasi .....	31
5.4 Implementasi Basis Data.....	32



5.5 Backlog Produk .....	35
5.6 Perancangan <i>Sprint</i> .....	36
5.7 Pengerjaan <i>Sprint</i> .....	37
5.7.1 <i>Sprint</i> Pertama .....	37
5.7.2 <i>Sprint</i> Kedua .....	64
5.7.3 <i>Sprint</i> Ketiga .....	93
BAB 6 Pengujian .....	101
6.1 Pengujian <i>Usability</i> .....	101
6.1.1 Analisis Hasil Pengujian Usability untuk Pelanggan Restoran .....	105
6.1.2 Analisis Pengujian Usability Aplikasi untuk Operator Restoran .....	108
BAB 7 Penutup .....	112
7.1 Kesimpulan.....	112
7.2 Saran .....	112
DAFTAR REFERENSI .....	114

## DAFTAR TABEL

Tabel 2.1 Keterbatasan perangkat <i>mobile</i> .....	7
Tabel 4.1 Daftar aktor dalam Sistem.....	20
Tabel 4.2 User story hasil wawancara Pelanggan.....	21
Tabel 4.3 User story hasil wawancara Operator Restoran .....	22
Tabel 4.4 Kebutuhan Fungsional Sistem .....	23
Tabel 4.5 Kebutuhan Non-Fungsional Sistem .....	25
Tabel 5.1 Penjelasan Perancangan Data .....	30
Tabel 5.2 Spesifikasi Perangkat Keras .....	31
Tabel 5.3 Spesifikasi Perangkat Keras .....	31
Tabel 5.4 Hasil implementasi basis data pada <i>entity</i> OrderItem.....	32
Tabel 5.5 Hasil implementasi <i>relational entity</i> .....	33
Tabel 5.6 Implementasi <i>realtime database entity</i> Orders .....	34
Tabel 5.7 Backlog Produk untuk Sprint Pertama .....	35
Tabel 5.8 Backlog Produk untuk Sprint Kedua .....	36
Tabel 5.9 Backlog Produk untuk Sprint Ketiga .....	36
Tabel 5.10 Skenario <i>Use Case</i> Login Pelanggan .....	37
Tabel 5.11 Skenario <i>Use Case</i> Melakukan Pemesanan Menu .....	38
Tabel 5.12 Skenario <i>Use Case</i> Melihat Riwayat Pemesanan Menu.....	38
Tabel 5.13 Skenario <i>Use Case</i> Mengubah Pesanan .....	39
Tabel 5.14 Skenario <i>Use Case</i> Menambahkan Menu Favorit .....	39
Tabel 5.15 Skenario <i>Use Case</i> Melihat Daftar Menu Favorit .....	40
Tabel 5.16 Skenario <i>Use Case</i> Melihat Rekomendasi Menu Restoran .....	40
Tabel 5.17 Skenario <i>Use Case</i> Logout Pelanggan.....	41
Tabel 5.18 Perancangan algoritme <i>create order</i> .....	53
Tabel 5.19 Implementasi kode program <i>create order</i> .....	53
Tabel 5.20 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Login Pelanggan .....	61
Tabel 5.21 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Melakukan Pemesanan.....	62
Tabel 5.22 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Melihat Riwayat Pemesanan .....	62
Tabel 5.23 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Mengubah Pesanan .....	62

Tabel 5.24 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Melihat Menu Restoran yang Paling Banyak Disukai oleh Pelanggan .....	63
Tabel 5.25 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Melihat Daftar Menu <i>Favorite</i> .....	63
Tabel 5.26 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Menambah Menu <i>Favorite</i> .....	63
Tabel 5.27 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Logout Pelanggan.....	63
Tabel 5.28 Skenario <i>Use Case</i> Login Operator Restoran.....	64
Tabel 5.29 Skenario <i>Use Case</i> Melihat Daftar Pesanan .....	65
Tabel 5.30 Skenario <i>Use Case</i> Mengubah Status Pesanan .....	65
Tabel 5.31 Skenario <i>Use Case</i> Melihat Daftar Kategori Menu Restoran .....	66
Tabel 5.32 Skenario <i>Use Case</i> Mengubah Kategori Menu Restoran .....	66
Tabel 5.33 Skenario <i>Use Case</i> Menambah Kategori Menu Restoran .....	67
Tabel 5.34 Skenario <i>Use Case</i> Menghapus Kategori Menu Restoran.....	67
Tabel 5.35 Skenario <i>Use Case</i> Melihat Daftar Menu Restoran.....	68
Tabel 5.36 Skenario <i>Use Case</i> Mengubah Menu Restoran.....	68
Tabel 5.37 Skenario <i>Use Case</i> Menambah Menu Restoran.....	69
Tabel 5.38 Skenario <i>Use Case</i> Menghapus Menu Restoran .....	69
Tabel 5.39 Skenario <i>Use Case</i> Mengubah Ketersediaan Menu Restoran.....	70
Tabel 5.40 Skenario <i>Use Case</i> Mengatur Ulang Ketersediaan Menu Restoran ....	70
Tabel 5.41 Perancangan algoritme mengubah status pesanan menjadi “PROCESS” .....	83
Tabel 5.42 Implementasi kode program <i>create order</i> .....	84
Tabel 5.43 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Login Operator Restoran....	89
Tabel 5.44 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Mengubah Status Pesanan .....	90
Tabel 5.45 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Melihat Daftar Kategori Menu Restoran .....	90
Tabel 5.46 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Menambah Kategori Menu Restoran .....	90
Tabel 5.47 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Mengubah Kategori Menu Restoran .....	90
Tabel 5.48 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Menghapus Kategori Menu Restoran .....	91
Tabel 5.49 Hasil pengujian <i>blackbox</i> pada <i>Use Case</i> Melihat Daftar Menu Restoran .....	91

Tabel 5.50 Hasil pengujian blackbox pada <i>Use Case</i> Menambah Menu Restoran	91
Tabel 5.51 Hasil pengujian blackbox pada <i>Use Case</i> Mengubah Menu Restoran	92
Tabel 5.52 Hasil pengujian blackbox pada <i>Use Case</i> Menghapus Menu Restoran .....	92
Tabel 5.53 Hasil pengujian blackbox pada <i>Use Case</i> Mengubah Ketersediaan Menu Restoran.....	92
Tabel 5.54 Hasil pengujian blackbox pada <i>Use Case</i> Mengatur Ulang Ketersediaan Menu Restoran.....	92
Tabel 5.55 Skenario <i>Use Case</i> Mengubah Profil Restoran .....	93
Tabel 5.56 Skenario <i>Use Case</i> Logout Operator Restoran .....	94
Tabel 5.57 Perancangan algoritme mengubah status pesanan menjadi “PROCESS” .....	97
Tabel 5.58 Implementasi kode program <i>update restaurant</i> .....	97
Tabel 5.59 Hasil pengujian blackbox pada <i>Use Case</i> Mengubah Profil Restoran .	99
Tabel 5.60 Hasil pengujian blackbox pada <i>Use Case</i> Logout Operator Restoran .	99
Tabel 6.1 Task Scenario Untuk Pelanggan Restoran .....	101
Tabel 6.2 <i>Task Scenario</i> Untuk Operator Restoran.....	102
Tabel 6.3 Skor Skala Likert dari Setiap Pertanyaan.....	103
Tabel 6.4 Kuesioner SUPR-Qm untuk Pelanggan Restoran.....	103
Tabel 6.5 Kuesioner SUPR-Qm untuk Operator Restoran .....	104
Tabel 6.6 Responden Pengujian <i>Usability</i> Pelanggan Restoran.....	106
Tabel 6.7 Task Completion Rate.....	106
Tabel 6.8 Hasil Pengujian Kuesioner SUPR-Qm untuk Pelanggan Restoran .....	107
Tabel 6.9 Task Completion Rate.....	109
Tabel 6.10 Hasil Pengujian Kuesioner SUPR-Qm untuk Operator Restoran.....	109

## DAFTAR GAMBAR

Gambar 2.1 Graphql diagram.....	9
Gambar 2.2 Node JS flow .....	10
Gambar 2.3 Alur Pengembangan dalam Metode Scrum .....	11
Gambar 2.4 Pattern MVVM .....	12
Gambar 3.1 Alur Metodologi Penelitian .....	16
Gambar 4.1 Identitas Kebutuhan.....	23
Gambar 4.2 Use case diagram aplikasi manajemen antrean pesanan menu restoran untuk pelanggan.....	26
Gambar 4.3 Use case diagram aplikasi manajemen antrean pesanan menu restoran untuk restoran.....	27
Gambar 5.1 <i>Entity Relational Diagram (ERD)</i> sistem manajemen antrean pesanan menu restoran.....	29
Gambar 5.2 <i>Sequence Diagram</i> Use Case Login Pelanggan.....	42
Gambar 5.3 <i>Sequence Diagram</i> Use Case Melakukan Pemesanan Menu.....	42
Gambar 5.4 <i>Sequence Diagram</i> Use Case Melihat Riwayat Pemesanan Menu ...	43
Gambar 5.5 <i>Sequence Diagram</i> Use Case Mengubah Pesanan.....	44
Gambar 5.6 <i>Sequence Diagram</i> Use Case Menambah Menu <i>Favorite</i> .....	44
Gambar 5.7 <i>Sequence Diagram</i> Use Case Menambah Menu <i>Favorite</i> .....	45
Gambar 5.8 <i>Sequence Diagram</i> Use Case Melihat Daftar Menu Rekomendasi ...	45
Gambar 5.9 <i>Sequence Diagram</i> Use Case Logout Pelanggan .....	46
Gambar 5.10 <i>Class Diagram</i> sistem pada <i>sprint</i> pertama .....	47
Gambar 5.11 <i>Wireframe Sign-in</i> .....	47
Gambar 5.12 <i>Wireframe Scan</i> .....	48
Gambar 5.13 <i>Wireframe Choose Restaurant Menu</i> .....	49
Gambar 5.14 <i>Wireframe Checkout</i> .....	49
Gambar 5.15 <i>Wireframe Orders</i> .....	50
Gambar 5.16 <i>Wireframe Order Detail</i> .....	50
Gambar 5.17 <i>Wireframe Update Order</i> .....	51
Gambar 5.18 <i>Wireframe Home</i> .....	51
Gambar 5.19 <i>Wireframe Favorite Menu</i> .....	52
Gambar 5.20 <i>Wireframe Account</i> .....	53

Gambar 5.21 Hasil Implementasi Antarmuka <i>Sign-in</i> .....	55
Gambar 5.22 Hasil Implementasi Antarmuka <i>Scan</i> .....	55
Gambar 5.23 Hasil Implementasi Antarmuka <i>Choose Restaurant Menu</i> .....	56
Gambar 5.24 Hasil Implementasi Antarmuka <i>Checkout</i> .....	57
Gambar 5.25 Hasil Implementasi Antarmuka <i>Orders</i> .....	58
Gambar 5.26 Hasil Implementasi Antarmuka <i>Order Detail</i> .....	58
Gambar 5.27 Hasil Implementasi Antarmuka <i>Update Order</i> .....	59
Gambar 5.28 Hasil Implementasi Antarmuka <i>Home</i> .....	60
Gambar 5.29 Hasil Implementasi Antarmuka <i>Favorite Menu</i> .....	60
Gambar 5.30 Hasil Implementasi Antarmuka <i>Account</i> .....	61
Gambar 5.31 <i>Sequence Diagram</i> Use Case Login Restaurant .....	71
Gambar 5.32 <i>Sequence Diagram</i> Use Case Melihat Daftar Pesanan .....	72
Gambar 5.33 <i>Sequence Diagram</i> Use Case Mengubah Status Pesanan .....	72
Gambar 5.34 <i>Sequence Diagram</i> Use Case Melihat Daftar Kategori Menu Restoran .....	73
Gambar 5.35 <i>Sequence Diagram</i> Use Case Menambah Kategori Menu Restoran .....	73
Gambar 5.36 <i>Sequence Diagram</i> Use Case Mengubah Kategori Menu Restoran .....	74
Gambar 5.37 <i>Sequence Diagram</i> Use Case Menghapus Kategori Menu Restoran .....	75
Gambar 5.34 <i>Sequence Diagram</i> Use Case Melihat Daftar Menu Restoran .....	75
Gambar 5.35 <i>Sequence Diagram</i> Use Case Menambah Menu Restoran .....	76
Gambar 5.36 <i>Sequence Diagram</i> Use Case Mengubah Menu Restoran .....	76
Gambar 5.37 <i>Sequence Diagram</i> Use Case Menghapus Menu Restoran .....	77
Gambar 5.42 <i>Sequence Diagram</i> Use Case Mengubah Ketersediaan Menu Restoran .....	78
Gambar 5.43 <i>Sequence Diagram</i> Use Case Mengatur Ulang Ketersediaan Menu Restoran .....	78
Gambar 5.40 <i>Class Diagram</i> aplikasi manajemen antrean pesanan menu restoran pada <i>sprint</i> ketiga .....	79
Gambar 5.41 <i>Wireframe Sign-in</i> .....	80
Gambar 5.41 <i>Wireframe Orders</i> .....	80
Gambar 5.42 <i>Wireframe Order Detail</i> .....	81
Gambar 5.43 <i>Wireframe Menu Availability</i> .....	82

Gambar 5.44 <i>Wireframe Category List</i> .....	82
Gambar 5.45 <i>Wireframe Menu List</i> .....	83
Gambar 5.46 Hasil Implementasi Antarmuka <i>Sign-in</i> .....	85
Gambar 5.47 Hasil Implementasi Antarmuka <i>Orders</i> .....	86
Gambar 5.48 Hasil Implementasi Antarmuka <i>Order Detail</i> .....	86
Gambar 5.49 Hasil Implementasi Antarmuka <i>Menu Availability</i> .....	87
Gambar 5.50 Hasil Implementasi Antarmuka <i>Category List</i> .....	88
Gambar 5.51 Hasil Implementasi Antarmuka <i>Menu List</i> .....	89
Gambar 5.53 <i>Sequence Diagram</i> Use Case Mengubah Profil Restoran .....	95
Gambar 5.54 <i>Sequence Diagram</i> Use Case Logout Operator Restoran .....	95
Gambar 5.55 Class Diagram aplikasi manajemen antrean pesanan menu restoran pada <i>sprint</i> ketiga.....	96
Gambar 5.56 <i>Wireframe Settings</i> .....	97
Gambar 5.57 Hasil Implementasi Antarmuka <i>Settings</i> .....	99

## DAFTAR LAMPIRAN

No table of contents entries found.



# BAB 1 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.

## 1.1 Latar Belakang

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

Menurut data yang didapatkan dari kuesioner pada LAMPIRAN A 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 daftar menu secara manual kepada pelanggannya yang membuat pihak restoran harus mencetak baru menunya setiap kali terdapat perubahan daftar menu restoran. Selain itu, pemesanan menu restoran secara manual masih menggunakan kertas, sehingga pelanggan harus menuliskan menu yang ingin dipesan melalui kertas yang diberikan oleh pelayan restoran. Hal ini akan membuang-buang waktu dan tenaga para pelanggan apalagi bagi mereka yang hanya memiliki waktu istirahat yang sedikit. Dengan keterbatasan karyawan restoran dan jumlah pelanggan yang sangat ramai membuat pelayanan di restoran tersebut sangat lama. Dengan data tersebut dapat diambil kesimpulan bahwa usaha restoran memang selalu ramai akan pelanggan

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

Pengembangan aplikasi ini dalam hal autentikasi restoran akan lebih mudah dengan menggunakan Kode QR. *Quick Response Code* atau yang biasa disebut

sebagai Kode QR adalah gambar digital dua dimensi dimana dapat dengan mudah dibaca oleh kamera pada perangkat *mobile* manapun. Sekarang ini Kode QR sangat populer karena perkembangan *mobile*. Penggunaan *mobile device* mencapai 15.6% pada tahun 2001 hingga 74.9% (2010) (Cata, T., Patel, P. S., & Sakaguchi, 2013).

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

## **1.2 Rumusan Masalah**

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

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

## **1.3 Tujuan**

Tujuan dari penelitian ini adalah sebagai berikut:

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

## **1.4 Manfaat**

1. Manfaat bagi penulis

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

2. Manfaat bagi peneliti selanjutnya

Dapat menjadi acuan untuk pengembangan aplikasi yang serupa.

### 3. Manfaat bagi pengguna

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

## 1.5 Batasan Masalah

Pengembangan dalam penelitian ini memiliki beberapa batasan, diantaranya adalah:

1. Sistem yang akan dibuat menggunakan menggunakan *framework React JS*.
2. Sistem yang akan dibuat menggunakan konsep *Progressive Website Application*.
3. 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.
4. Sistem yang akan dibuat harus terkoneksi dengan internet.
5. Data yang diolah bersumber dari restoran Ayam Bakar Wong Solo di Malang yang beralamat di Jl. Soekarno Hatta no.501 D Ruko C-d, Mojolangu, Kec. Lowokwaru, Kota Malang, Jawa Timur 65142.
6. Target pengguna sistem adalah pelanggan yang menggunakan Ayam Bakar Wong Solo sebagai tempat makan.

## 1.6 Sistematika Pembahasan

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

### 1. Bab 1 – PENDAHULUAN

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

### 2. Bab 2 – LANDASAN KEPUSTAKAAN

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

### 3. Bab 3 – METODOLOGI PENELITIAN

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

### 4. Bab 4 – ANALISIS KEBUTUHAN

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

### 5. Bab 5 – PENGEMBANGAN METODOLOGI SCRUM

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.

6. 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.

7. 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.

## BAB 2 LANDASAN KEPUSTAKAAN

### 2.1 Kajian Pustaka

Dalam melakukan penelitian ini, terdapat beberapa penelitian sebelumnya yang relevan dengan manajemen antrean pesanan menu restoran, yaitu penelitian yang berjudul Aplikasi Reservasi Menu Restoran Berbasis Web dan Mobile Android di Cowek Ireng (Rasid, Supriyono, & Setiawan, 2018). Pada penelitian ini, terdapat masalah antrean panjang di restoran yang diatasi dengan membuat aplikasi pemesanan menu restoran pada restoran Cowek Ireng. Hasil dari penelitian ini 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 industri makanan dan juga semakin meningkatnya teknologi dengan memanfaatkan internet dapat membuat pemesanan menu makanan secara *online* di restoran menjadi kebutuhan masyarakat. Pengguna dapat dengan mudah memesan menu makanan di restoran menggunakan *real-time online food ordering*, menelusuri proses pembuatan makanan, dan memberikan *feedback* kepada menu makanan dan juga restoran agar dapat meningkatkan produktivitas restoran. Inisial implementasi system dilakukan pada 2 restoran/mess pada 5 area.

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

### 2.2 Manajemen Antrean Pesanan Menu Restoran

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

sehingga membuat sistem antrean menjadi *Multichannel-Single Phase* yang mana terdapat dua jalur antrean dengan dua fasilitas pelayanan kasir. (Susila, Panji, & Prima, 2007)

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

### 2.3 Aplikasi Perangkat Bergerak

Aplikasi perangkat bergerak (*mobile application*) adalah suatu set program atau *software* yang berjalan atau beroperasi pada sebuah perangkat bergerak 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 perangkat *mobile* dijelaskan pada Tabel 2.1

**Tabel 2.1 Keterbatasan perangkat *mobile***

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

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

## **2.4 Progressive Website Application**

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

*Progressive Website Application* memiliki beberapa fitur, seperti:

1. Dapat diandalkan

Memuat aplikasi secara instan, karena dapat berjalan secara *offline*, sehingga ketika aplikasi dibuka, maka *service worker* yang bekerja selayaknya *client proxy* akan handle seluruh request sesuai yang

sudah termuat di dalam *cache* saat aplikasi pertama kali dimuat atau diperbarui.

## 2. Cepat

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

## 3. Mengikat pengguna dan dapat diinstall

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

## 4. Fresh

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

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

## 2.5 React JS

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

## 2.6 Kode QR

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

### 1. Penyimpanan yang lebih besar

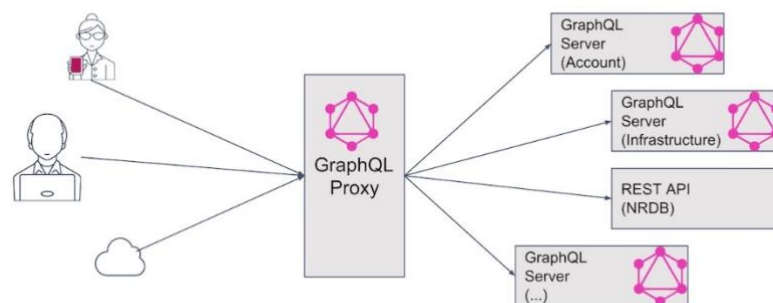


2. Variasi data yang dapat disembunyikan dalam *Kode QR* adalah teks, nama, URL, SMS, email, dan kontak informasi
3. Dapat dilakukan *decoding* pada platform yang berbeda
4. Digunakan sebagai *online payment*
5. Menyimpan tipe data yang berbeda-beda seperti *numeric*, *alphabetic*, *binary*, dan Kanji

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

## 2.7 GraphQL

Graphql adalah sebuah konsep baru dalam membangun sebuah *application programming interface* (API) dan diimplementasikan pada sisi *server*. Posisi Graphql berada pada sisi klien dan *server* yang berhubungan dalam suatu API untuk mengambil dan memanipulasi data. Graphql didesain untuk berkolaborasi dengan 9ahasa pemrograman *server* yang lain dengan baik (Buna, 2016).

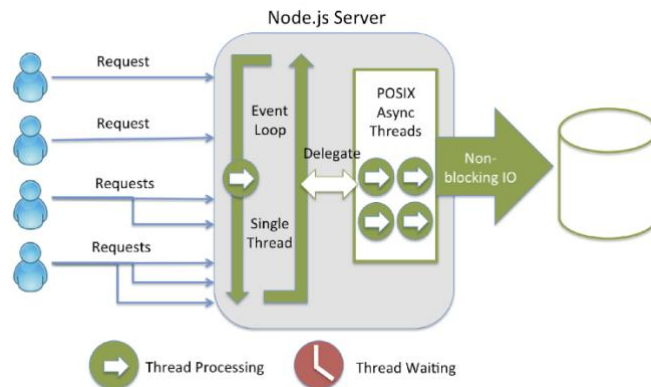


**Gambar 2.1 Graphql diagram**

(sumber: [blog.newrelic.com](http://blog.newrelic.com))

## 2.8 Node JS

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



**Gambar 2.2 Node JS flow**

(sumber: strongloop.com)

## 2.9 Model Pengembangan Perangkat Lunak

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).

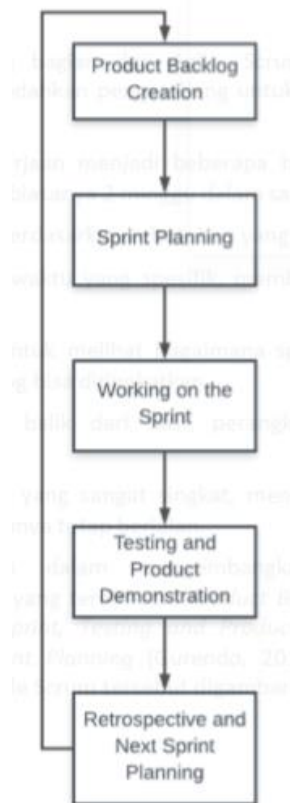
### 2.9.1 Scrum

Pengembangan sistem manajemen antrean pesanan menu 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:

1. Memecah durasi pekerjaan menjadi beberapa bagian yang kemudian disebut dengan *sprint* (biasanya 2 minggu dalam satu *sprint*).
2. Merencanakan *sprint* berdasarkan kebutuhan yang penting pada saat itu.
3. Tidak memperkirakan waktu yang spesifik, membandingkan banyaknya pekerjaan.
4. Mengevaluasi *sprint* untuk melihat bagaimana *sprint* itu berjalan, dan melihat apakah ada yang bisa ditingkatkan.
5. Mendapatkan umpan balik dari hasil perangkat lunak yang telah dikerjakan.
6. Pertemuan setiap hari yang sangat singkat, mengidentifikasi halangan, dan memastikan semuanya tetap berjalan.

Ada beberapa tahapan dalam mengembangkan aplikasi dengan menggunakan metode Scrum yang terdiri atas *Product Backlog Creation*, *Sprint Planning*, *Working on the Sprint*, *Testing and Product Demonstration*, dan

*Restrospective and Next Sprint Planning* (Gurendo, 2015). Adapun alur dari tahapan-tahapan dalam metode Scrum tersebut digambarkan pada Gambar 2.3.



**Gambar 2.3 Alur Pengembangan dalam Metode Scrum**

*Product Backlog* 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*.

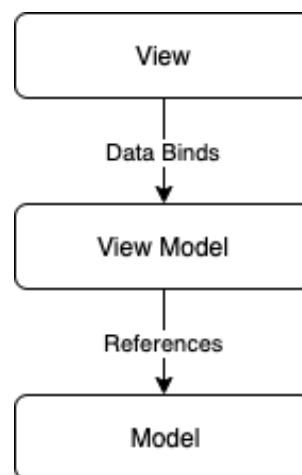
*Testing and Product Demonstration* merupakan hasil dari setiap *sprint* yang telah dilakukan. Tim membuat ulasan dan mendemonstrasikan hasil pekerjaan

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.

## 2.10 MVVM Design Pattern

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



**Gambar 2.4 Pattern MVVM**

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

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

## 2.11 Google Material Design

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

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

Berikut adalah prinsip dari Material Design:

1. Material adalah metafora

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

2. *Bold*, grafis, *intentional*

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

3. *Motion provides meaning*

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

4. Pondasi yang fleksibel

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

5. *Cross-platform*

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

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

## 2.12 Pengujian Perangkat Lunak

### 2.12.1 *Blackbox Testing*

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

### 2.12.2 *Usability Testing*

*Usability testing* atau uji ketergunaan sebuah aplikasi 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).

### **2.12.3 Regression Testing**

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

## **BAB 3 METODOLOGI PENELITIAN**

### **3.1 Jenis Penelitian**

Jenis penelitian yang digunakan dalam penelitian ini adalah penelitian implementatif-pengembangan (*development*). Karena pada penelitian ini terdapat penerapan prinsip-prinsip rekayasa, pemodelan sistem, dan tahapan pengembangan yang lengkap seperti tahap analisis, perancangan, implementasi, dan pengujian.

### **3.2 Tempat dan Waktu Penelitian**

#### **1. Tempat Penelitian**

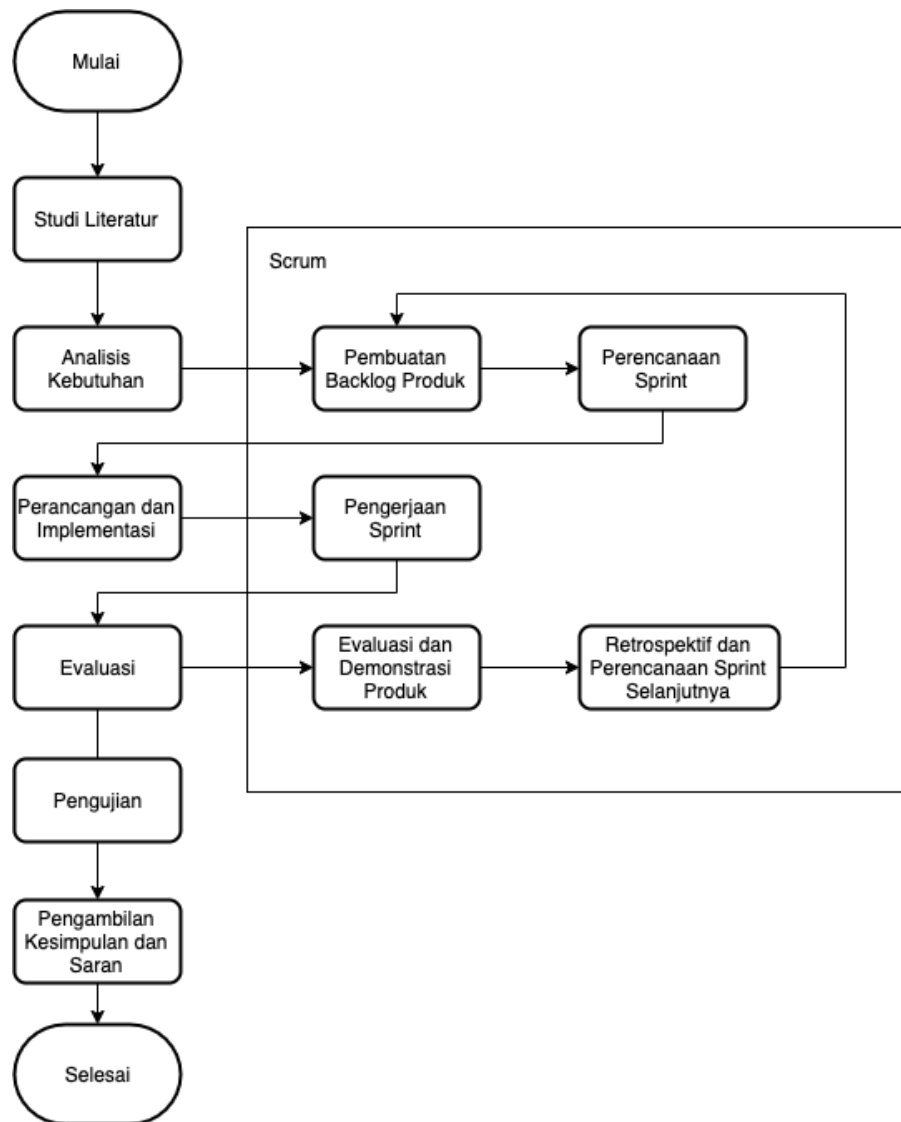
Penelitian ini akan dilaksanakan di Ayam Bakar Wong Solo yang memiliki alamat di Jl. Soekarno Hatta no.501 D Ruko C-D, Mojolangu, Kec. Lowokwaru, Kota Malang, Jawa Timur 65142.

#### **2. Waktu Penelitian**

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

### **3.3 Diagram Alir Metode**

Pada bab ini akan membahas alur pelaksanaan atau tahapan-tahapan dalam pengembangan sistem manajemen antrean pesanan menu restoran. Hal ini dilakukan agar proses dapat terarah dengan baik dan sesuai dengan tujuan. Metode dalam pengembangan perangkat lunak sistem manajemen antrean pesanan menu 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**

### 3.4 Analisis Kebutuhan

Analisis kebutuhan merupakan tahap dalam mempelajari kebutuhan calon pelanggan agar mendapatkan definisi kebutuhan sistem atau tahap untuk menentukan kebutuhan dari seluruh elemen sistem. 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:

1. Menjelaskan mengenai identifikasi aktor.
2. Menjelaskan mengenai pembuatan *User Story*.
3. Menjelaskan mengenai daftar kebutuhan fungsional.
4. Menjelaskan mengenai pembuatan *backlog product*.



5. Menjelaskan mengenai perencanaan *sprint*.
6. Menjelaskan mengenai analisis data.
7. Menjelaskan mengenai *use case diagram*.
8. Menjelaskan mengenai *use case* skenario.

#### **3.4.1 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.

### **3.5 Perancangan dan Implementasi**

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

#### **1. Perancangan Arsitektur**

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

#### **2. 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*

#### **3. 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 penelitian ini menggunakan serangkaian kode program. Kode program tersebut menggunakan bahasa pemrograman Javascript. Tahapan implementasi sistem diantaranya

adalah menjelaskan spesifikasi sistem, batasan implementasi, implementasi basis data, implementasi kode program, dan implementasi antarmuka.

### **3.6 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.

#### **3.6.1 Evaluasi dan Demonstrasi Produk**

Merupakan tahap evaluasi sistem yang akan dilakukan terhadap sistem yang telah dikembangkan, pengujian yang dilakukan adalah dengan menggunakan metode pengujian *Black Box Validation* dan *Regression Test*. Evaluasi sistem bertujuan untuk mengetahui jalannya kebutuhan fungsional yang didefinisikan sebelumnya apakah berjalan sesuai yang diinginkan. Tahapan evaluasi sistem diantaranya adalah pengujian *Black Box* (Validasi) dan *Regression Test*.

#### **3.6.2 Retrospektif dan 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.

### **3.7 Pengujian**

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

### **3.8 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 jawaban untuk permasalahan yang telah dirumuskan dalam penelitian ini.

## BAB 4 ANALISIS KEBUTUHAN

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

### 4.1 Gambaran Umum Sistem

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

#### 4.1.1 Deskripsi Umum Sistem

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

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

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

lagi mencetak ulang menu ketika terdapat perubahan pada menu restoran. Selain itu, antrian 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 antrian pesanan menu restoran untuk restoran akan membuat seluruh tahapan pemrosesan pesanan pelanggan dari mulai pesanan dibuat, diurutkan berdasarkan waktu yang tercatat saat pesanan dibuat, hingga akhirnya pesanan siap disajikan akan dibantu oleh sistem. Alur kerja dari sistem aplikasi manajemen antrian pesanan menu restoran untuk restoran adalah pesanan yang dibuat oleh pelanggan akan masuk ke antrian daftar pemesanan. Setelah itu koki restoran dapat melihat urutan prioritas menu apa saja yang harus disajikan terlebih dahulu. Kemudian setelah menu yang dipesan untuk satu pelanggan siap disajikan, pelayan dapat mengantarkan menu makanannya ke pelanggan yang memesan menu tersebut.

#### 4.1.2 Lingkungan Sistem

Aplikasi manajemen antrian pesanan menu 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*.

### 4.2 Analisis Kebutuhan Perangkat Lunak

#### 4.2.1 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 antrian pesanan menu restoran dapat dilihat pada Tabel 4.1

**Tabel 4.1 Daftar aktor dalam Sistem**

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

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

#### 4.2.2 Pembuatan User Story

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 pesanan menu restoran memiliki dua jenis pengguna, yaitu pelanggan dan operator restoran. Untuk menggali *user story* dari calon pengguna pelanggan, pertanyaan yang dilakukan dalam wawancara terhadap pelanggan difokuskan kepada pertanyaan berupa 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

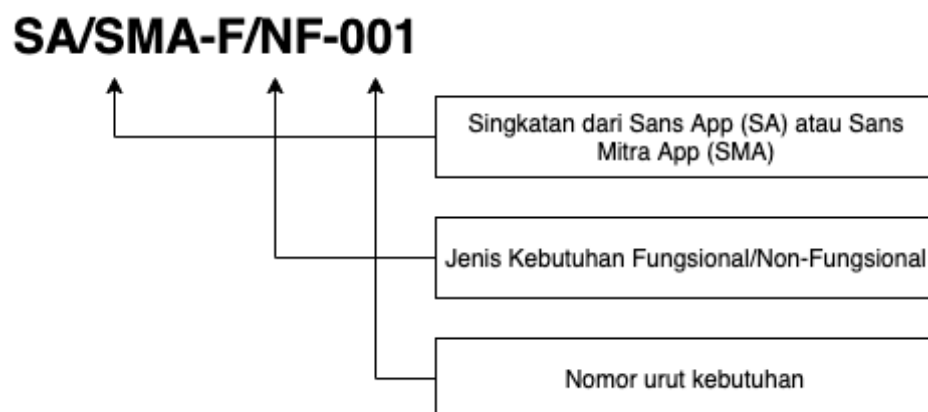
3	Sebagai pelanggan saya ingin melihat makanan favorite atau rekomendasi di sebuah restoran
4	Sebagai pelanggan saya ingin menyimpan menu-menu restoran yang saya sukai

**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 <i>up-to-date</i>

#### 4.2.3 Kebutuhan Fungsional Sistem

Kebutuhan fungsional sistem merupakan sebuah daftar yang mendeskripsikan kemampuan dan layanan yang disediakan oleh sistem. Kebutuhan fungsional 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 fungsional, 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



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

**Tabel 4.4 Kebutuhan Fungsional Sistem**

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

13	SMA-F-002	Sistem harus mampu menampilkan daftar pesanan pelanggan
14	SMA-F-003	Sistem harus mampu mengubah status pesanan pelanggan
15	SMA-F-004	Sistem harus mampu menampilkan daftar kategori menu restoran
16	SMA-F-005	Sistem harus mampu mengubah kategori menu restoran
17	SMA-F-006	Sistem harus mampu menambah kategori menu restoran
18	SMA-F-007	Sistem harus mampu menghapus kategori menu restoran
19	SMA-F-008	Sistem harus mampu menampilkan daftar menu restoran
20	SMA-F-009	Sistem harus mampu mengubah menu restoran
21	SMA-F-010	Sistem harus mampu menambah menu restoran
22	SMA-F-011	Sistem harus mampu menghapus menu restoran
23	SMA-F-012	Sistem harus mampu mengubah status ketersediaan menu restoran
24	SMA-F-013	Sistem harus mampu mengatur ulang ketersediaan menu restoran
25	SMA-F-014	Sistem harus mampu mengubah profil restoran
26	SMA-F-015	Sistem harus mampu mengeluarkan operator restoran dari sistem

#### 4.2.4 Kebutuhan Non-Fungsional Sistem

Kebutuhan non-fungsional adalah 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 pesanan menu restoran didaftarkan pada Tabel 4.5.

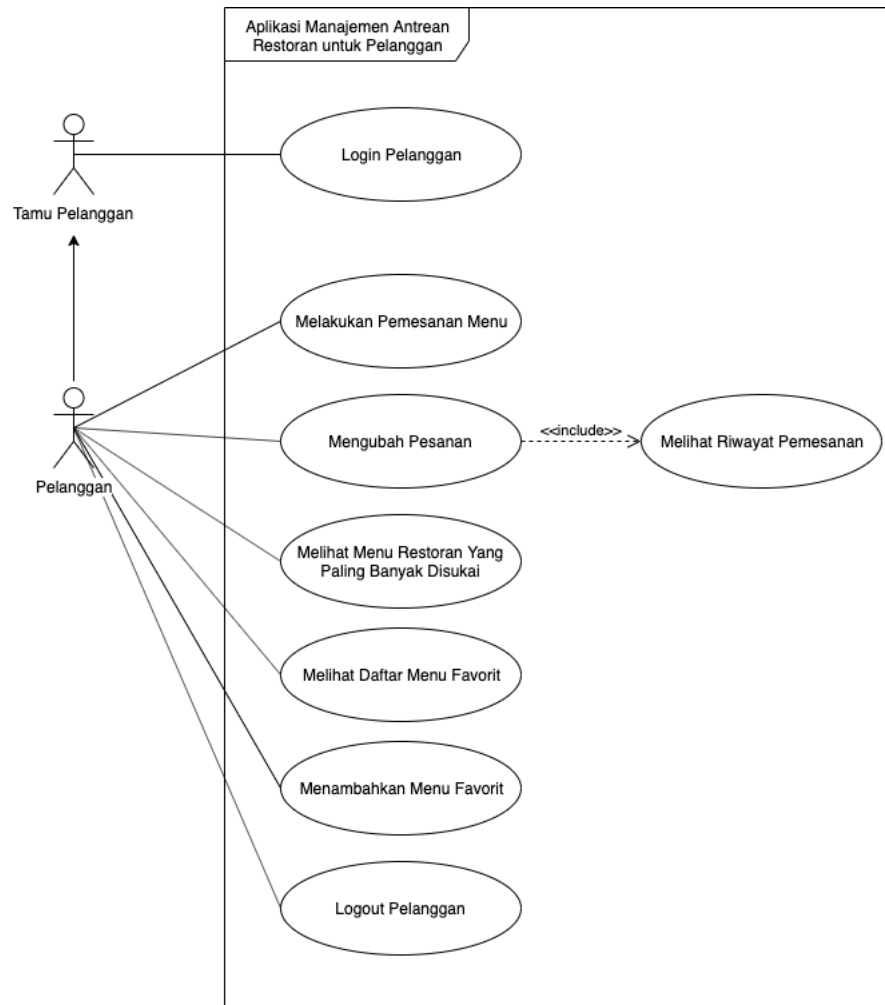


**Tabel 4.5 Kebutuhan Non-Fungsional Sistem**

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

#### **4.2.5 Use Case Diagram**

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



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



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

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

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

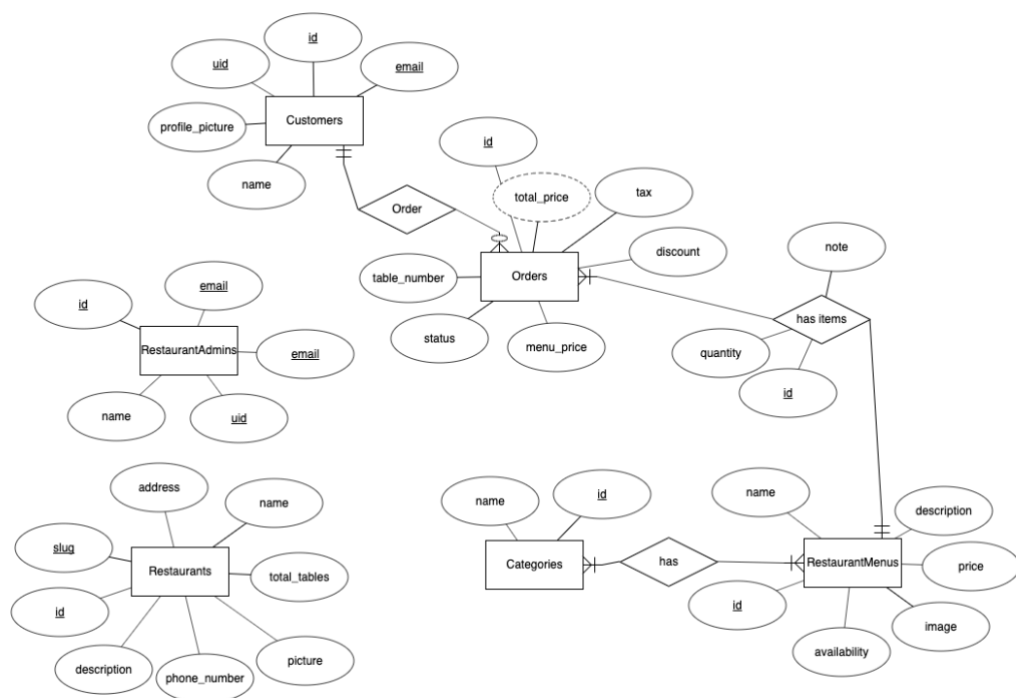
Pada *use case diagram* Login Operator Restoran, aktor yang terlibat adalah Tamu Operator yang merepresentasikan operator restoran yang belum masuk ke dalam sistem, sehingga dikenal sebagai tamu dan tidak dapat mengakses fitur-fitur lain dari sistem kecuali Login Operator Restoran. Setelah Tamu Operator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

## BAB 5 PENGEMBANGAN METODOLOGI SCRUM

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

### 5.1 Perancangan Basis Data

Basis data adalah komponen yang dibutuhkan oleh sistem untuk menyimpan seluruh data yang dipakai dan diolah oleh sistem. Perancangan basis data dilakukan untuk merancang seluruh komponen dalam basis data yang akan dijadikan referensi untuk diimplementasikan pada tahap implementasi basis data. Perancangan basis data yang dibuat dalam penelitian ini mereferensi pada analisis yang telah dilakukan sebelumnya. Perancangan basis data dalam penelitian ini digambarkan melalui *Entity Relational Diagram* atau ERD pada Gambar 2.1



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

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

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

**Tabel 5.1 Penjelasan Perancangan Data**

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

	Phone_number	Nomor telepon restoran
	Name	Nama restoran
	Address	Alamat restoran
	Total_tables	Total meja restoran
	Picture	Banner restoran

## 5.2 Spesifikasi Lingkungan Sistem Implementasi

### 5.2.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk mengembangkan perangkat lunak dijelaskan pada Tabel 5.2

**Tabel 5.2 Spesifikasi Perangkat Keras**

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

### 5.2.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk mengembangkan perangkat lunak dijelaskan pada Tabel 5.2

**Tabel 5.3 Spesifikasi Perangkat Keras**

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

## 5.3 Batasan Implementasi

Batasan implementasi dalam pengimplementasian perangkat lunak adalah sebagai berikut:

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

## 5.4 Implementasi Basis Data

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

**Tabel 5.4 Hasil implementasi basis data pada *entity* OrderItem**

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

Implementasi *relational entity* dapat dilihat pada Tabel 5.5



**Tabel 5.5 Hasil implementasi *relational entity***

1	import connection from './connection'
2	import { events, DB_CONNECTED } from '../events'
3	import Customer from './Customer'
4	import Category from './Category'
5	import OrderItem from './OrderItem'
6	import RestaurantMenu from './RestaurantMenu'
7	import RestaurantAdmin from './RestaurantAdmin'
8	import Restaurant from './Restaurant'
9	import SocialMedia from './SocialMedia'
10	import './Favorite'
11	import './Upload'
12	
13	// associate Restaurant with RestaurantMenu
14	Restaurant.hasMany(RestaurantMenu, { foreignKey:
15	'restaurant_id' })
16	RestaurantMenu.belongsTo(Restaurant, { foreignKey:
17	'restaurant_id' })
18	
19	// associate RestaurantAdmin with Restaurant
20	Restaurant.belongsTo(RestaurantAdmin, { foreignKey:
21	'restaurant_admin_id' })
22	RestaurantAdmin.hasMany(Restaurant, { foreignKey:
23	'restaurant_admin_id' })
24	
25	// associate Restaurant with Category
26	Restaurant.hasMany(Category, { foreignKey: 'restaurant_id'
27	})
28	Category.belongsTo(Restaurant, { foreignKey: 'restaurant_id'
29	})
30	
31	// associate RestaurantMenu with Category
32	Category.hasMany(RestaurantMenu, { foreignKey: 'category_id'
33	})
34	RestaurantMenu.belongsTo(Category, { foreignKey:
35	'category_id' })
36	
37	// associate RestaurantMenu with Customer as Favorite Menu
38	RestaurantMenu.belongsToMany(Customer, {
39	through: 'Favorite',
40	foreignKey: 'menu_id',
41	as: 'Favorites',
42	})
43	Customer.belongsToMany(RestaurantMenu, {
44	through: 'Favorite',
45	foreignKey: 'customer_id',
46	as: 'Favorites',
47	})
48	
49	// associate OrderItem with RestaurantMenu
50	RestaurantMenu.hasMany(OrderItem, { foreignKey: 'menu_id' })
51	OrderItem.belongsTo(RestaurantMenu, {
52	foreignKey: 'menu_id',
53	as: 'RestaurantMenus'
54	})
55	
56	import { giveSeeds } from '../seeders'
57	

```

58 let force = true
59
60 connection
61   .sync({
62     force
63   })
64   .then(async () => {
65     console.log('database synchronized')
66
67     events.emit(DB_CONNECTED)
68     if (force) giveSeeds()
69   })
70   .catch(err => {
71     console.log(err)
72   })
73
74 export default connection

```

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

**Tabel 5.6 Implementasi *realtime database entity* Orders**

```

1  {
2    "Orders" : {
3      "-LuVaTxnhJk1CpfPZJrt" : {
4        "created_at" : 1574652479577,
5        "customer" : {
6          "email" : "yahya.sahaja2@gmail.com",
7          "id" : 1,
8          "name" : "Yahya Sahaja",
9          "profile_picture" :
10         "https://lh3.googleusercontent.com/a-
11         /AAuE7mAjp1I9Q0CHL_E3yIT9f7zQxOI8iqvJsbFSTe0f_Q",
12         "uid" : "ytXc5lD8T8XtwowFyTCtZsVzw7E2"
13       },
14       "customer_id" : "1",
15       "discount" : 0,
16       "id" : "-LuVaTxnhJk1CpfPZJrt",
17       "menu_price" : 75000,
18       "order_items" : [ {
19         "menu_id" : "17",
20         "note" : "",
21         "quantity" : 2,
22         "restaurant_menu" : {
23           "availability" : true,
24           "category_id" : 8,
25           "description" : "Gurih Manis/Pedas",
26           "id" : 17,
27           "image" :
28           "/images/wongsolo/MenuPaketAyamBakar.png",
29           "name" : "Ayam Bakar",
30           "price" : 25000,
31           "restaurant_id" : 4
32         }
33       }
34     }
35   }

```

35	}
36	}

## 5.5 Backlog Produk

Pembagian *backlog* produk memiliki tujuan untuk membagi *backlog* yang sudah dibuat sebelumnya ke dalam beberapa *sprint*. Urutan *backlog* yang akan dimasukkan dalam urutan *sprint* disesuaikan dengan prioritas fitur pada sebuah backlog, sehingga *backlog-backlog* yang lebih penting akan direncanakan pada *sprint* yang lebih dulu. Oleh karena itu, hasil pemecahan *backlog* untuk setiap *sprint* akan dijelaskan pada **Error! Reference source not found.**, **Error! Reference source not found.**, dan **Error! Reference source not found.**

**Tabel 5.7 Backlog Produk untuk Sprint Pertama**

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

**Tabel 5.8 Backlog Produk untuk Sprint Kedua**

No	Kode	Deskripsi Kebutuhan
1	SMA-F-001	Sistem harus mampu menyediakan tempat untuk memasukkan tamu operator restoran ke sistem
2	SMA-F-002	Sistem harus mampu menampilkan daftar pesanan pelanggan
3	SMA-F-003	Sistem harus mampu mengubah status pesanan pelanggan
4	SMA-F-004	Sistem harus mampu menampilkan daftar kategori menu restoran
5	SMA-F-005	Sistem harus mampu mengubah kategori menu restoran
6	SMA-F-006	Sistem harus mampu menambah kategori menu restoran
7	SMA-F-007	Sistem harus mampu menghapus kategori menu restoran
8	SMA-F-008	Sistem harus mampu menampilkan daftar menu restoran
9	SMA-F-009	Sistem harus mampu mengubah menu restoran
10	SMA-F-010	Sistem harus mampu menambah menu restoran
11	SMA-F-011	Sistem harus mampu menghapus menu restoran
12	SMA-F-012	Sistem harus mampu mengubah ketersediaan menu restoran
13	SMA-F-013	Sistem harus mampu mengatur ulang ketersediaan menu restoran

**Tabel 5.9 Backlog Produk untuk Sprint Ketiga**

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

## 5.6 Perancangan *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 fungsional sistem. Sebuah durasi sprint tidak boleh melebihi waktu selama 4 minggu, dikarenakan jika durasi sprint terlalu lama, prospektif 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:

1. Semakin pendek durasi *sprint*, maka akan cepat umpan balik yang akan diterima, sehingga dapat dilakukan perbaikan sistem lebih cepat
2. Membantu peneliti untuk menjadi lebih fokus
3. Calon pengguna akan menerima respon perbaikan dari umpan balik yang telah diberikan menjadi lebih cepat melalui rilis produk *incremental* berikutnya

## 5.7 Pengerjaan *Sprint*

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

### 5.7.1 *Sprint* Pertama

#### 5.7.1.1 *Use Case Scenario*

*Use Case Scenario* adalah skenario apa saja yang dapat terjadi pada sebuah Use Case. *Use Case Scenario* dari sistem manajemen antrean pesanan menu restoran pada *sprint* pertama dijabarkan pada **Error! Reference source not found.** sampai dengan Tabel 5.56.

**Tabel 5.10 Skenario *Use Case* Login Pelanggan**

<i>Use Case</i>	Login Pelanggan
Kode terkait kebutuhan	SA-F-001
<i>Actor</i>	Tamu pelanggan
<i>Target</i>	Pelanggan terdaftar atau masuk ke dalam sistem
<i>Pre-Condition</i>	Pelanggan belum terdaftar atau masuk ke dalam sistem
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Sistem menampilkan halaman <i>Login</i></li> <li>2. Tamu pelanggan menekan tombol <i>Sign In With Google</i>.</li> <li>3. Sistem menampilkan akun Google yang tersedia.</li> </ol>

	4. Tamu memilih akun Google atau menambahkan akun Google baru. 5. Sistem memproses otentikasi pelanggan.
<i>Alternative Flow</i>	-
<i>Post Condition</i>	Pelanggan Tamu pelanggan berhasil masuk ke dalam sistem dan status tamu berubah menjadi pelanggan

**Tabel 5.11 Skenario *Use Case* Melakukan Pemesanan Menu**

<i>Use Case</i>	Melakukan Pemesanan Menu
Kode terkait kebutuhan	SA-F-002, SA-F-003, SA-F-004, dan SA-F-005
<i>Actor</i>	Pelanggan
<i>Target</i>	Pelanggan melakukan pemesanan menu restoran
<i>Pre-Condition</i>	Pelanggan sudah berada di halaman <i>home</i>
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Pelanggan menekan tombol <i>scan</i></li> <li>2. Sistem menampilkan halaman <i>Scan</i> Kode QR</li> <li>3. Pelanggan melakukan <i>scan</i> pada Kode QR meja</li> <li>4. Sistem menampilkan halaman restoran</li> <li>5. Pelanggan memilih makanan yang hendak dipesan dengan menekan tombol <i>add</i> sebanyak jumlah yang diinginkan dan menekan tombol <i>checkout</i> apabila pesanan sudah sesuai</li> <li>6. Sistem menampilkan halaman <i>checkout</i></li> <li>7. Pelanggan menekan tombol <i>order now</i> apabila pesannya telah tepat</li> <li>8. Sistem menampilkan pesan bahwa pesanan telah berhasil dibuat</li> </ol>
<i>Alternative Flow</i>	<ol style="list-style-type: none"> <li>1. Apabila stock habis saat data menu sudah dibuat dan telah memilih menu, sistem akan menampilkan pesan <i>error</i> bahwa stok menu sudah habis</li> </ol>
<i>Post Condition</i>	Pesanan menu telah dilakukan dan dibuat

**Tabel 5.12 Skenario *Use Case* Melihat Riwayat Pemesanan Menu**

<i>Use Case</i>	Melihat Riwayat Pemesanan Menu
Kode kebutuhan	SA-F-006

terkait	
<i>Actor</i>	Pelanggan
<i>Target</i>	Pelanggan dapat melihat daftar pemesanan yang pernah dilakukan dalam aplikasi manajemen antrean pesanan menu restoran untuk pelanggan
<i>Pre-Condition</i>	Pelanggan sudah berada di halaman <i>home</i>
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Pelanggan memilih halaman <i>orders</i></li> <li>2. Sistem menampilkan halaman <i>orders</i>, terdapat dua bagian yaitu bagian pesanan yang masih dalam proses dan yang telah selesai.</li> </ol>
<i>Alternative Flow</i>	-
<i>Post Condition</i>	Pelanggan berhasil mengakses daftar riwayat pesanan dalam halaman <i>orders</i> .

**Tabel 5.13 Skenario Use Case Mengubah Pesanan**

<i>Use Case</i>	Melihat Riwayat Pemesanan Menu
Kode kebutuhan terkait	SA-F-006
<i>Actor</i>	Pelanggan
<i>Target</i>	Pelanggan dapat mengubah pesanan ketika status pesanan adalah "INLINE"
<i>Pre-Condition</i>	Pelanggan sudah berada di halaman <i>transaction page</i>
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Pelanggan memilih menekan tombol "Update Order"</li> <li>2. Sistem menampilkan halaman <i>update order</i></li> <li>3. Pelanggan memilih menu yang ingin dipesan dan menekan tombol "Update Order"</li> <li>4. Sistem mengubah pesanan pelanggan</li> </ol>
<i>Alternative Flow</i>	-
<i>Post Condition</i>	Pelanggan berhasil mengubah pesanan ketika status pesanan adalah "INLINE"

**Tabel 5.14 Skenario Use Case Menambahkan Menu Favorit**

<i>Use Case</i>	Menambahkan Menu Favorit
Kode kebutuhan	SA-F-007

terkait	
<i>Actor</i>	Pelanggan
<i>Target</i>	Pelanggan dapat menambahkan menu favorit suatu restoran ke dalam daftar menu favorit
<i>Pre-Condition</i>	Pelanggan telah berada di halaman restoran
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Pelanggan menekan tombol favorit yang berbentuk hati pada menu kesukaannya.</li> <li>2. Sistem menambahkan menu menjadi menu favorit pelanggan</li> </ol>
<i>Alternative Flow</i>	<ol style="list-style-type: none"> <li>1. Apabila menu restoran sudah dihapus sebelum menu favorit yang diinginkan pelanggan dipilih, sistem akan menampilkan pesan <i>error</i> bahwa menu restoran sudah tidak tersedia</li> </ol>
<i>Post Condition</i>	Menu berhasil ditambahkan ke dalam daftar menu favorit.

**Tabel 5.15 Skenario *Use Case* Melihat Daftar Menu Favorit**

<i>Use Case</i>	Menambahkan Menu Favorit
Kode kebutuhan terkait	SA-F-008
<i>Actor</i>	Pelanggan
<i>Target</i>	Pelanggan dapat melihat daftar menu favorit yang pernah difavoritkan.
<i>Pre-Condition</i>	Pelanggan telah berada di halaman restoran
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Pelanggan memilih halaman <i>favorite</i>.</li> <li>2. Sistem menampilkan halaman <i>favorite</i> dan daftar menu favorit.</li> </ol>
<i>Alternative Flow</i>	-
<i>Post Condition</i>	Pelanggan melihat daftar menu favorit mereka

**Tabel 5.16 Skenario *Use Case* Melihat Rekomendasi Menu Restoran**

<i>Use Case</i>	Melihat Rekomendasi Menu Restoran
Kode kebutuhan terkait	SA-F-009
<i>Actor</i>	Pelanggan



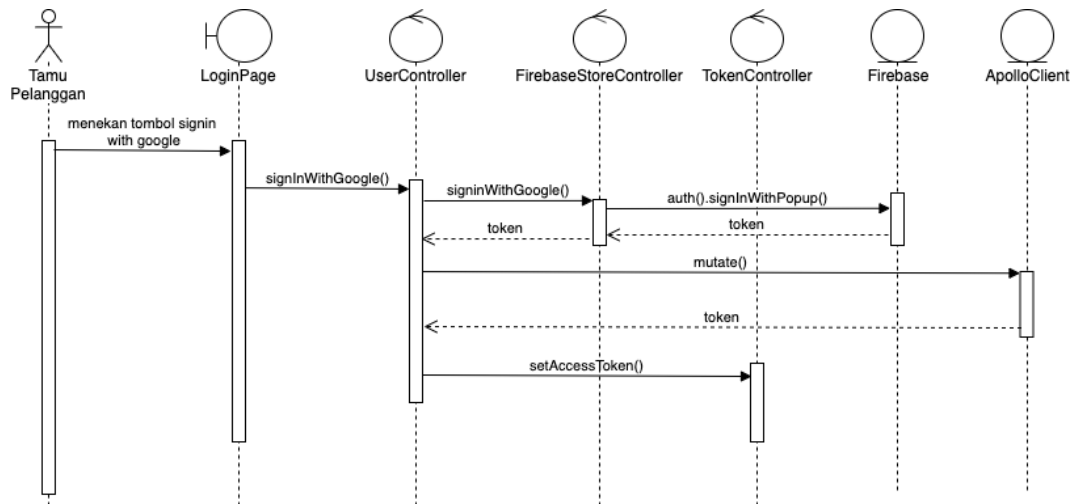
<i>Target</i>	Pelanggan dapat melihat rekomendasi menu restoran
<i>Pre-Condition</i>	Pelanggan sudah berada di halaman <i>home</i>
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Pelanggan memilih halaman <i>home</i></li> <li>2. Sistem menampilkan halaman <i>home</i> yang berisikan list menu restoran sesuai dengan lokasi pengguna</li> </ol>
<i>Alternative Flow</i>	-
<i>Post Condition</i>	Pelanggan melihat rekomendasi menu restoran.

**Tabel 5.17 Skenario *Use Case* Logout Pelanggan**

<i>Use Case</i>	Logout Pelanggan
Kode kebutuhan terkait	SA-F-010
<i>Actor</i>	Pelanggan
<i>Target</i>	Otentikasi pelanggan dapat keluar dari sistem.
<i>Pre-Condition</i>	Pelanggan sudah berada di halaman <i>home</i>
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Pelanggan memilih halaman <i>more</i></li> <li>2. Sistem menampilkan halaman <i>more</i></li> <li>3. Pelanggan menekan tombol <i>Logout</i></li> <li>4. Sistem menampilkan dialog konfirmasi untuk <i>logout</i></li> <li>5. Pelanggan memilih tombol <i>Logout</i></li> <li>6. Sistem menampilkan halaman login</li> </ol>
<i>Alternative Flow</i>	<ol style="list-style-type: none"> <li>1. Jika Pelanggan memilih tombol cancel logout pada dialog, dialog tertutup kembali</li> </ol>
<i>Post Condition</i>	Otentikasi pelanggan telah keluar dari sistem.

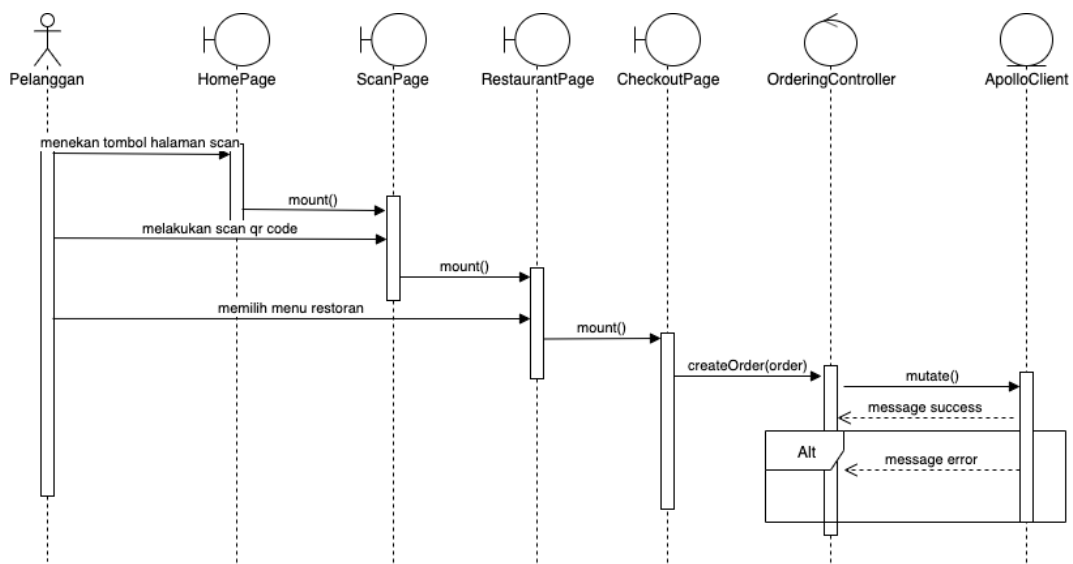
#### **5.7.1.2 Sequence Diagram**

*Sequence diagram* menggambarkan urutan proses yang terjadi di dalam sistem untuk mencapai tujuan dari kebutuhan fungsionalitas sistem. Daftar *sequence diagram* dalam sistem manajemen antrean pesanan menu restoran di *sprint* pertama dapat dilihat pada Gambar 5.2 sampai Gambar 5.9.



**Gambar 5.2 Sequence Diagram Use Case Login Pelanggan**

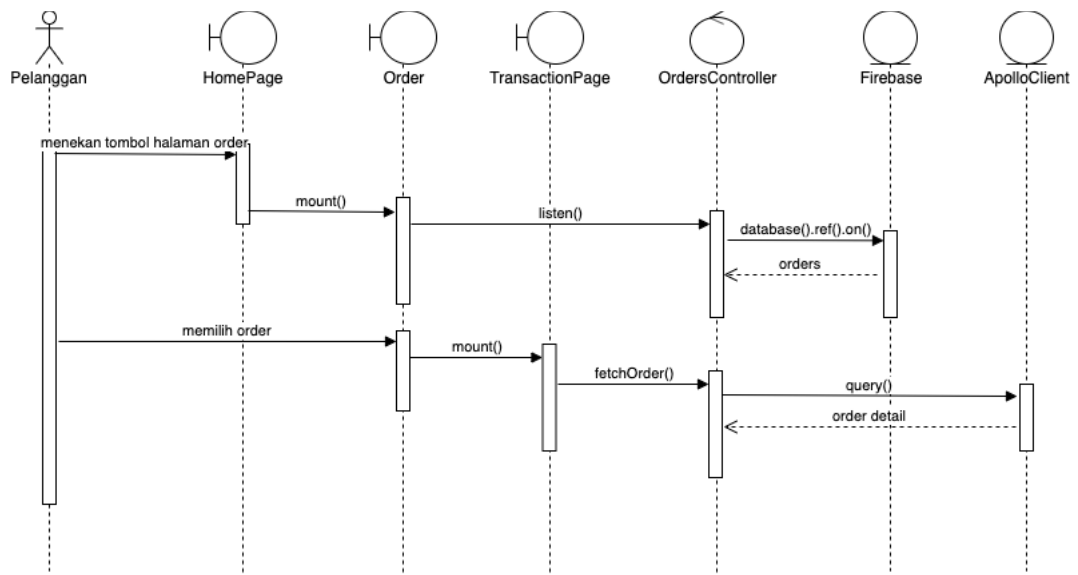
Pada *sequence diagram* Login Pelanggan pada Gambar 5.2 dijelaskan bahwa aktor tamu pelanggan menekan tombol *signin with google* pada halaman LoginPage. Kemudian aktivitas tersebut akan memicu pemanggilan fungsi *signInWithGoogle* pada UserController. Setelah pelanggan berhasil *login* dengan *credentials*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



**Gambar 5.3 Sequence Diagram Use Case Melakukan Pemesanan Menu**

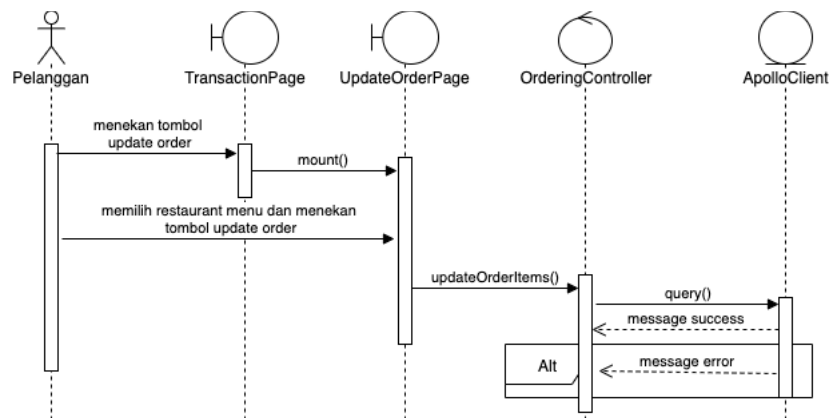
Pada *sequence diagram* Melakukan Pemesanan Menu pada Gambar 5.3 dijelaskan bahwa aktor pelanggan menekan tombol halaman scan pada

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



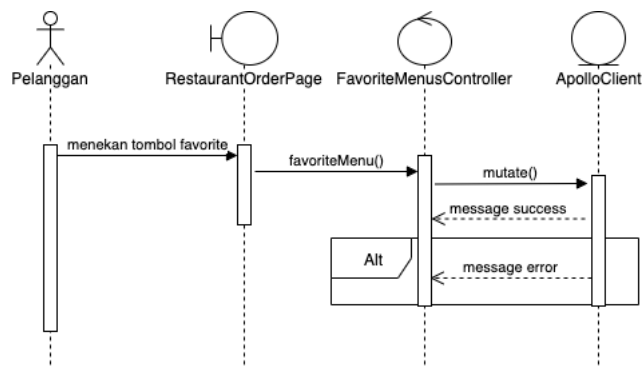
**Gambar 5.4 Sequence Diagram Use Case Melihat Riwayat Pemesanan Menu**

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



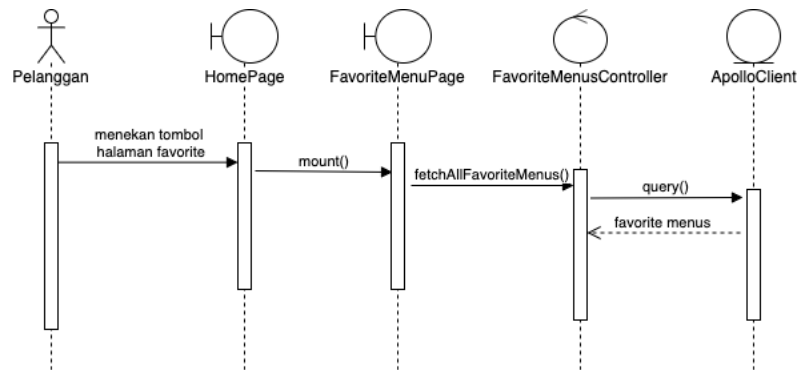
**Gambar 5.5 Sequence Diagram Use Case Mengubah Pesanan**

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



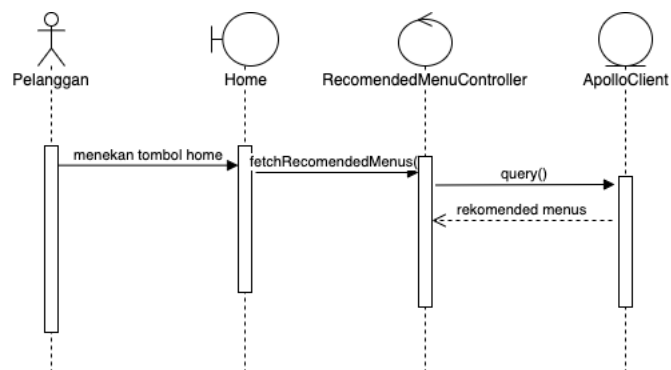
**Gambar 5.6 Sequence Diagram Use Case Menambah Menu Favorite**

Pada *sequence diagram* Menambah Menu *Favorite* pada Gambar 5.6 dijelaskan bahwa aktor Pelanggan menekan tombol *favorite* berupa gambar hati pada menu restoran yang dipilih pelanggan di RestaurantOrderPage. Kemudian dilakukan pemanggilan fungsi `favoriteMenu()` pada FavoriteMenusController. Aktivitas tersebut memicu *request favorite menu* ke Sans API, jika sukses akan mendapatkan pesan sukses, dan apabila gagal akan mendapatkan pesan *error*.



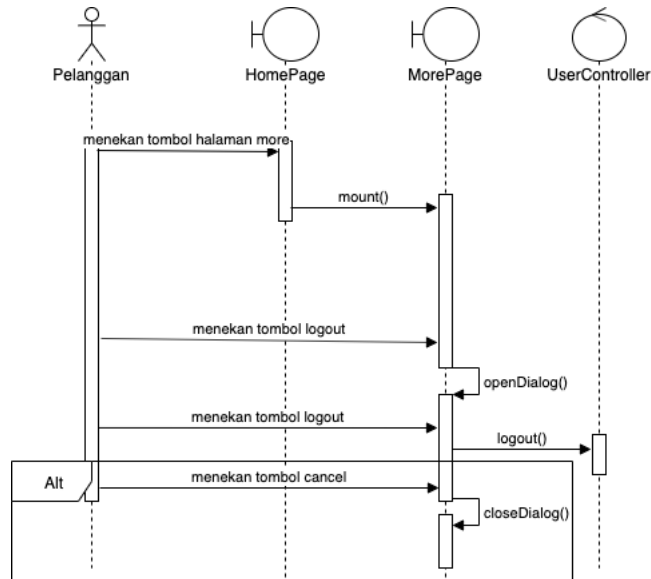
**Gambar 5.7 Sequence Diagram Use Case Menambah Menu *Favorite***

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



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

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

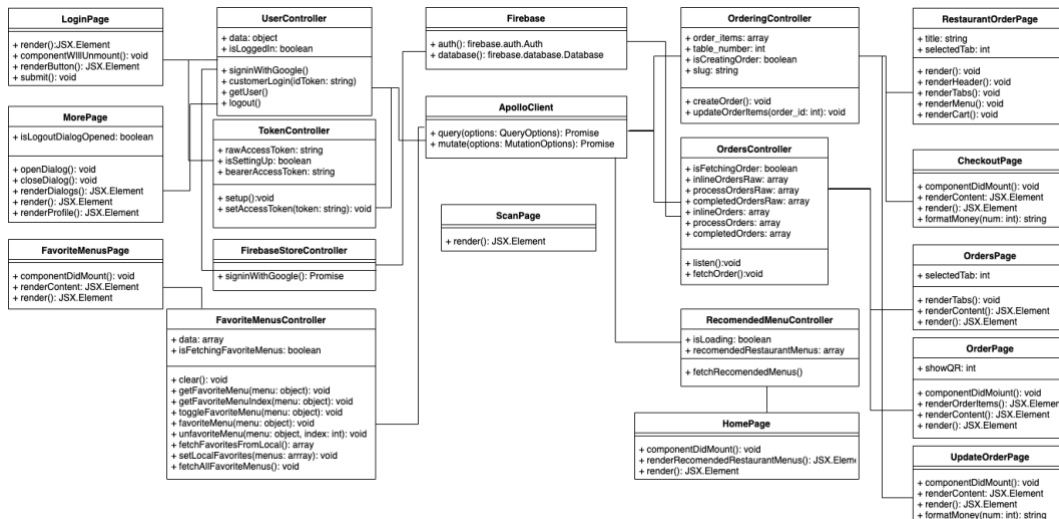


**Gambar 5.9 Sequence Diagram Use Case Logout Pelanggan**

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

### 5.7.1.3 Class Diagram

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

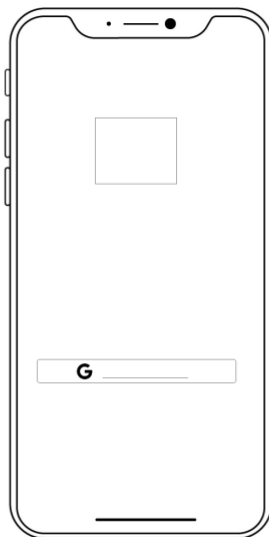


**Gambar 5.10 Class Diagram sistem pada sprint pertama**

#### 5.7.1.4 Perancangan Antarmuka

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

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

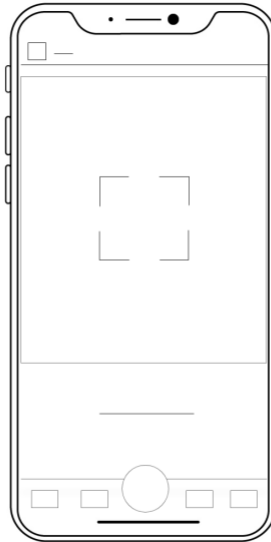


**Gambar 5.11 Wireframe Sign-in**

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

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

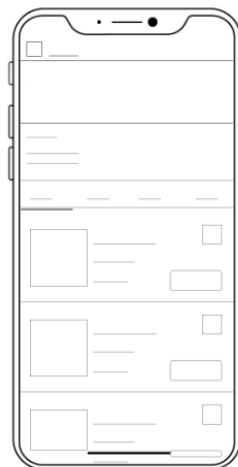
**(b) Perancangan Antarmuka *Scan***



**Gambar 5.12 Wireframe *Scan***

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

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

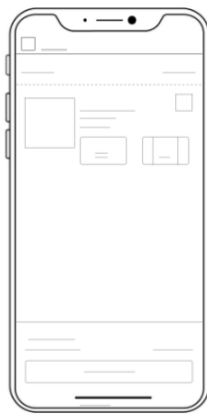




### **Gambar 5.13 Wireframe Choose Restaurant Menu**

Pada Gambar 5.13 menunjukkan halaman *choose restaurant menu*. Halaman *choose restaurant menu* adalah halaman untuk melakukan pemilihan menu yang tersedia pada restoran. Pada halaman *choose restaurant menu*, terdapat terdapat deskripsi restoran, *menu categories* yang digambarkan dalam bentuk *tab bar*, dan pilihan menu dari setiap kategori dalam bentuk *card*. Kemudian dalam *menu card* terdapat gambar, nama, deskripsi, dan harga menu, beserta tombol untuk like, menambah, dan mengurangi menu yang diinginkan pelanggan.

#### **(d) Perancangan Antarmuka Checkout**



### **Gambar 5.14 Wireframe Checkout**

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

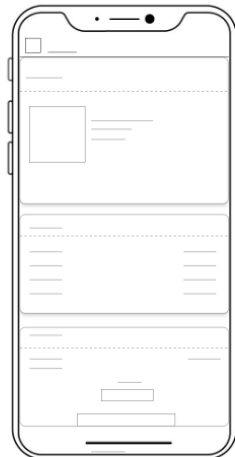
**(e) Perancangan Antarmuka *Orders***



**Gambar 5.15 Wireframe *Orders***

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

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

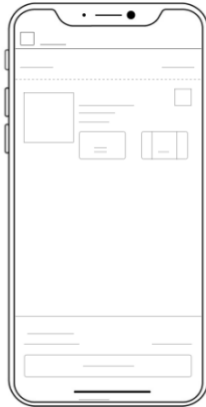


**Gambar 5.16 Wireframe *Order Detail***

Pada Gambar 5.16 menunjukkan halaman *order detail*. Halaman *order detail* adalah halaman untuk melihat detail pesanan pelanggan. Pada halaman *order detail*, terdapat detail pemesanan yang dilakukan pelanggan yang meliputi data-data restoran, detail harga, status pemesanan, dan menu yang dipesan. Selain itu,

jika status pesanan masih berstatus “INLINE”, maka akan terdapat tombol untuk melakukan *update* pada *order*.

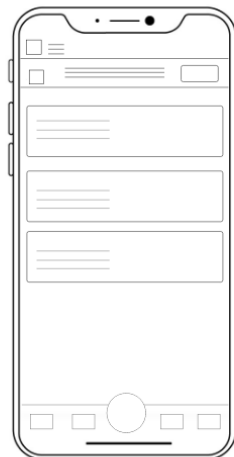
**(g) Perancangan Antarmuka *Update Order***



**Gambar 5.17 Wireframe *Update Order***

Pada Gambar 5.17 menunjukkan halaman *update order*. Halaman *update order* adalah halaman untuk melakukan perubahan pada pesanan yang telah dilakukan pelanggan, jika status pesanan masih “INLINE”. Pada halaman *update order*, terdapat pilihan seluruh menu yang ada di restoran berupa *menu card*. Dalam *menu card* terdapat tombol untuk menambahkan dan mengurangi kuantitas dan catatan. Kemudian terdapat tombol “*Update Order*” untuk melakukan perubahan pada pesanan.

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

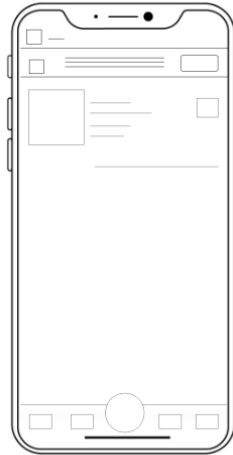


**Gambar 5.18 Wireframe *Home***

Pada Gambar 5.18 menunjukkan halaman *home*. Halaman *home* adalah halaman untuk melihat urutan menu-menu pada restoran yang memiliki jumlah

like terbanyak dari pelanggan. Pada halaman *home*, terdapat terdapat *card* yang menampilkan nama makanan dan gambar menu restoran yang sudah diurutkan sesuai dengan banyaknya jumlah like pada menu restoran tersebut.

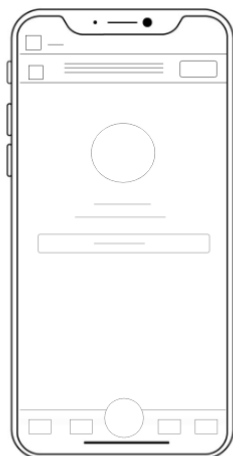
**(i) Perancangan Antarmuka *Favorite Menu***



**Gambar 5.19 Wireframe *Favorite Menu***

Pada Gambar 5.19 menunjukkan halaman *favorite menu*. Halaman *favorite menu* adalah halaman untuk melihat daftar menu yang telah disukai oleh pelanggan. Pada halaman *favorite menu*, terdapat terdapat daftar menu berupa *card* yang menampilkan gambar menu, nama menu, deskripsi menu, harga menu, dan tombol like. Jika tombol like *diklik*, maka menu tersebut akan dihapus dari daftar *menu favorite*.

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



### Gambar 5.20 Wireframe Account

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

#### 5.7.1.5 Perancangan Algoritme

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

**Tabel 5.18 Perancangan algoritme *create order***

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

Algoritme *create order* merupakan fungsi untuk membuat pesanan yang dipilih oleh pelanggan restoran. Diawali dengan mendeklarasikan seluruh variable yang menyesuaikan dengan *input* pada *field* yang dibutuhkan untuk melakukan *request*. Kemudian dilakukan *request* ke *server* dengan menginputkan data *variable* yang telah dideklarasikan sebelumnya. Setelah itu jika terdapat *error*, maka pesan *error* akan ditampilkan oleh sistem.

#### 5.7.1.6 Implementasi Kode Program

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

**Tabel 5.19 Implementasi kode program *create order***

1	@action
2	async createOrder() {
3	let {
4	restaurant_id,
5	order_items,
6	table_number,
7	customer_id,
8	} = this
9	
10	let orderItems = toJS(order_items)
11	orderItems.forEach(d => delete d.menu)
12	
13	let variables = {
14	restaurant_id,

15	order_items: orderItems,
16	table_number,
17	customer_id,
18	}
19	
20	try {
21	this.isCreatingOrder = true
22	overlayLoading.show()
23	let {
24	data: {
25	createOrder: order
26	}
27	} = await client.mutate({
28	mutation: createOrderMutation,
29	variables
30	})
31	
32	this.isCreatingOrder = false
33	overlayLoading.hide()
34	snackbar.show('Order created!')
35	return order
36	} catch (err) {
37	this.isCreatingOrder = false
38	overlayLoading.hide()
39	snackbar.show(err.message)
40	console.log('ERROR WHILE CREATING ORDER', err.message)
41	}
42	}

Pembahasan dari kode program diatas adalah sebagai berikut:

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

#### 5.7.1.7 Implementasi Antarmuka

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

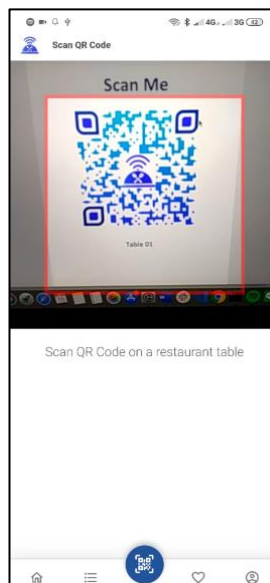
**(a) Hasil Implementasi Antarmuka *Sign-in***



**Gambar 5.21 Hasil Implementasi Antarmuka *Sign-in***

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

**(b) Hasil Implementasi Antarmuka *Scan***

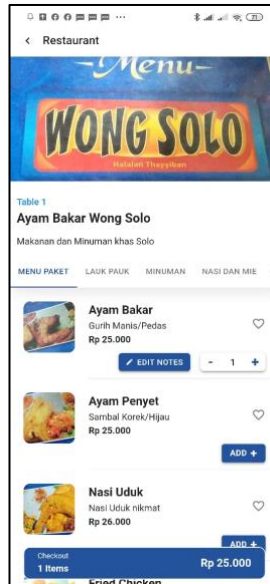


**Gambar 5.22 Hasil Implementasi Antarmuka *Scan***

Pada Gambar 5.11 menunjukkan halaman *scan*. Halaman *scan* adalah halaman untuk melakukan *scanning* kode QR yang terdapat pada meja restoran.

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

**(c) Hasil Implementasi Antarmuka *Choose Restaurant Menu***

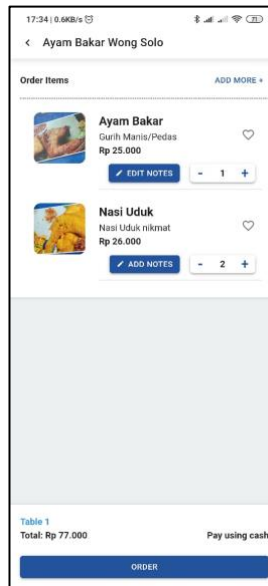


**Gambar 5.23 Hasil Implementasi Antarmuka *Choose Restaurant Menu***

Pada Gambar 5.13 menunjukkan halaman *choose restaurant menu*. Halaman *choose restaurant menu* adalah halaman untuk melakukan pemilihan menu yang tersedia pada restoran. Pada halaman *choose restaurant menu*, terdapat terdapat deskripsi restoran, *menu categories* yang digambarkan dalam bentuk *tab bar*, dan pilihan menu dari setiap kategori dalam bentuk *card*. Kemudian dalam *menu card* terdapat gambar, nama, deskripsi, dan harga menu, beserta tombol untuk like, menambah, dan mengurangi menu yang diinginkan pelanggan.



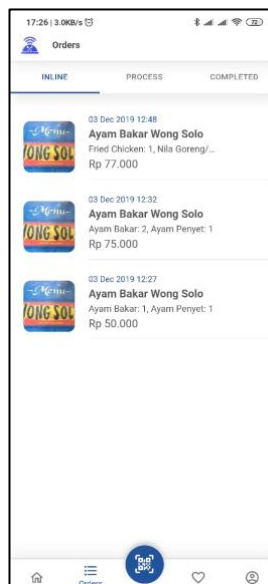
#### (d) Hasil Implementasi Antarmuka *Checkout*



**Gambar 5.24 Hasil Implementasi Antarmuka *Checkout***

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

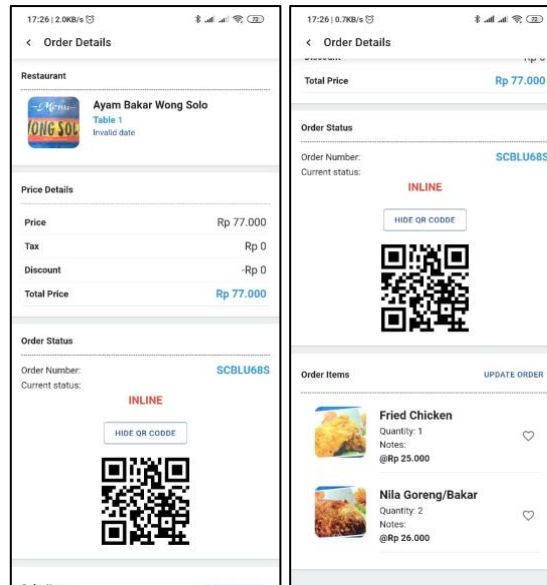
#### (e) Hasil Implementasi Antarmuka *Orders*



**Gambar 5.25 Hasil Implementasi Antarmuka *Orders***

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

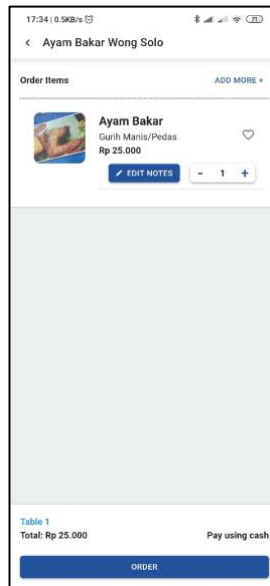
**(f) Hasil Implementasi Antarmuka *Order Detail***



**Gambar 5.26 Hasil Implementasi Antarmuka *Order Detail***

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

**(g) Hasil Implementasi Antarmuka *Update Order***



**Gambar 5.27 Hasil Implementasi Antarmuka *Update Order***

Pada Gambar 5.17 menunjukkan halaman *update order*. Halaman *update order* adalah halaman untuk melakukan perubahan pada pesanan yang telah dilakukan pelanggan, jika status pesanan masih “INLINE”. Pada halaman *update order*, terdapat pilihan seluruh menu yang ada di restoran berupa *menu card*. Dalam *menu card* terdapat tombol untuk menambahkan dan mengurangi kuantitas dan catatan. Kemudian terdapat tombol “*Update Order*” untuk melakukan perubahan pada pesanan.

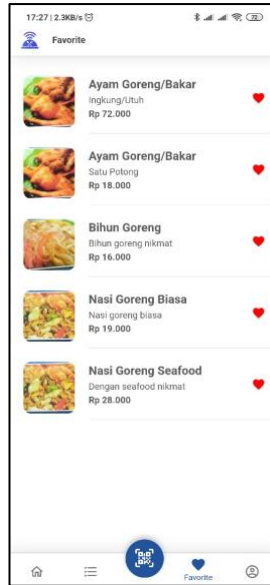
**(h) Hasil Implementasi Antarmuka *Home***



### Gambar 5.28 Hasil Implementasi Antarmuka *Home*

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

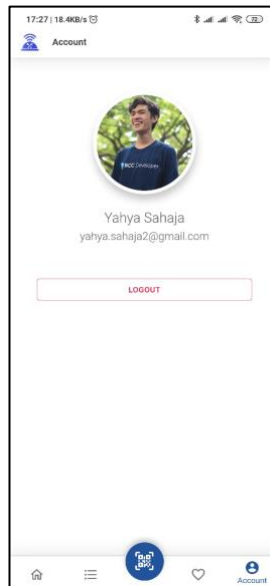
#### (i) Hasil Implementasi Antarmuka *Favorite Menu*



### Gambar 5.29 Hasil Implementasi Antarmuka *Favorite Menu*

Pada Gambar 5.19 menunjukkan halaman *favorite menu*. Halaman *favorite menu* adalah halaman untuk melihat daftar menu yang telah disukai oleh pelanggan. Pada halaman *favorite menu*, terdapat terdapat daftar menu berupa *card* yang menampilkan gambar menu, nama menu, deskripsi menu, harga menu, dan tombol like. Jika tombol like *diklik*, maka menu tersebut akan dihapus dari daftar *menu favorite*.

### (j) Hasil Implementasi Antarmuka *Account*



**Gambar 5.30 Hasil Implementasi Antarmuka *Account***

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

#### 5.7.1.8 *Evaluasi dan Demonstrasi Produk*

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

##### (a) *Evaluasi*

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

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

<i>Test Case</i>	<i>Expectation</i>	<i>Actual Result</i>	<i>Validation</i>
Login menggunakan akun Google	Tamu pelanggan berhasil masuk ke dalam sistem menggunakan akun	Tamu pelanggan berhasil masuk ke dalam sistem menggunakan akun	<i>Valid</i>

pelanggan	Google pelanggan	Google pelanggan	
-----------	------------------	------------------	--

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

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

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

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

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

<i>Test Case</i>	<i>Expectation</i>	<i>Actual Result</i>	<i>Validation</i>
Mengubah pesanan pelanggan ketika status pesanan adalah "INLINE"	Pelanggan berhasil mengubah pesanannya ketika status pesanan adalah "INLINE"	Pelanggan berhasil mengubah pesanannya ketika status pesanan adalah "INLINE"	<i>Valid</i>

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

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

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

<i>Test Case</i>	<i>Expectation</i>	<i>Actual Result</i>	<i>Validation</i>
Melihat daftar menu <i>favorite</i> yang dilakukan oleh pelanggan	Daftar menu yang telah dijadikan sebagai menu <i>favorite</i> berhasil ditampilkan oleh sistem	Daftar menu yang telah dijadikan sebagai menu <i>favorite</i> berhasil ditampilkan oleh sistem	<i>Valid</i>

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

<i>Test Case</i>	<i>Expectation</i>	<i>Actual Result</i>	<i>Validation</i>
Melakukan penambahan menu restoran menjadi menu <i>favorite</i> pelanggan	Sistem berhasil menambahkan menu restoran yang dipilih pelanggan menjadi menu <i>favorite</i> pelanggan	Sistem berhasil menambahkan menu restoran yang dipilih pelanggan menjadi menu <i>favorite</i> pelanggan	<i>Valid</i>

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

<i>Test Case</i>	<i>Expectation</i>	<i>Actual Result</i>	<i>Validation</i>
Melakukan logout dengan mengeluarkan pelanggan dari sistem	Sistem berhasil mengeluarkan pelanggan dari sistem	Sistem berhasil mengeluarkan pelanggan dari sistem	<i>Valid</i>

## (b) Demonstrasi Produk

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

Dari demonstrasi produk pada *sprint* pertama ini, didapatkan *feedback* dari operator restoran bahwa aplikasi sudah memuaskan dan mengatasi masalah pemesanan yang ada di restoran seperti penggunaan kertas untuk menulis pesanan, pemanggilan pelayan untuk memesan, dan penggunaan daftar menu dengan kertas yang tidak *update* terhadap ketersediaan menu restoran. Adapun terdapat foto yang menunjukkan demonstrasi produk yang dilakukan kepada pengguna dapat dilihat pada LAMPIRAN B.

### 5.7.2 *Sprint* Kedua

#### 5.7.2.1 *Use Case Scenario*

*Use Case Scenario* adalah skenario apa saja yang dapat terjadi pada sebuah *Use Case*. *Use Case Scenario* dari sistem manajemen antrean pesanan menu restoran pada *sprint* kedua dijabarkan pada Tabel 5.28 sampai dengan Tabel 5.39.

**Tabel 5.28 Skenario *Use Case* Login Operator Restoran**

<i>Use Case</i>	Login Operator Restoran
Kode terkait kebutuhan	SMA-F-001
<i>Actor</i>	Tamu Operator Restoran
<i>Target</i>	Operator restoran terdaftar atau masuk ke dalam sistem
<i>Pre-Condition</i>	Operator restoran belum terdaftar atau masuk ke dalam sistem
<i>Main Flow</i>	<ol style="list-style-type: none"><li>1. Sistem menampilkan halaman <i>Login</i></li><li>2. Tamu operator restoran menekan tombol <i>Sign In With Google</i>.</li><li>3. Sistem menampilkan akun Google yang tersedia.</li><li>4. Tamu memilih akun Google atau menambahkan akun Google baru.</li><li>5. Sistem memproses otentikasi operator restoran.</li></ol>
<i>Alternative Flow</i>	Tamu operator restoran berhasil masuk ke dalam sistem dan status tamu berubah menjadi operator restoran
<i>Post Condition</i>	Operator restoran sudah terdaftar atau masuk ke dalam sistem



**Tabel 5.29 Skenario Use Case Melihat Daftar Pesanan**

<i>Use Case</i>	Melihat Daftar Pesanan
Kode terkait kebutuhan	SMA-F-002
<i>Actor</i>	Operator Restoran
<i>Target</i>	Operator restoran melihat daftar pesanan yang dibuat oleh pelanggan
<i>Pre-Condition</i>	Operator restoran telah berada di halaman dashboard
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Operator restoran memilih halaman daftar pesanan</li> <li>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.</li> </ol>
<i>Alternative Flow</i>	-
<i>Post Condition</i>	Operator restoran telah melihat daftar pesanan yang dibuat oleh pelanggan

**Tabel 5.30 Skenario Use Case Mengubah Status Pesanan**

<i>Use Case</i>	Mengubah Pesanan
Kode terkait kebutuhan	SMA-F-003
<i>Actor</i>	Operator Restoran
<i>Target</i>	Operator restoran dapat mengubah pesanan pelanggan
<i>Pre-Condition</i>	Operator restoran telah berada di halaman daftar pesanan dan daftar pesanan telah ditampilkan oleh sistem
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Operator restoran menekan tombol "Process Order"</li> <li>2. Sistem mengubah status pesanan dari "INLINE" menjadi "PROCESS"</li> </ol>
<i>Alternative Flow</i>	<ol style="list-style-type: none"> <li>1. Operator restoran menekan tombol "Complete Order"</li> <li>2. Sistem mengubah status pesanan dari "PROCESS" menjadi "COMPLETED"</li> </ol>
<i>Post Condition</i>	Operator restoran telah mengubah pesanan pelanggan

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

<i>Use Case</i>	Melihat Daftar Kategori Menu Restoran
Kode terkait kebutuhan	SMA-F-004
<i>Actor</i>	Operator Restoran
<i>Target</i>	Operator restoran melihat daftar kategori menu restoran
<i>Pre-Condition</i>	Operator restoran telah berada di halaman <i>dashboard</i>
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Operator restoran memilih halaman daftar kategori menu restoran</li> <li>2. Sistem menampilkan halaman daftar kategori menu restoran</li> </ol>
<i>Alternative Flow</i>	-
<i>Post Condition</i>	Operator restoran telah melihat daftar kategori menu restoran

**Tabel 5.32 Skenario *Use Case* Mengubah Kategori Menu Restoran**

<i>Use Case</i>	Mengubah Kategori Menu Restoran
Kode terkait kebutuhan	SMA-F-005
<i>Actor</i>	Operator Restoran
<i>Target</i>	Operator restoran dapat mengubah kategori menu restoran
<i>Pre-Condition</i>	Operator restoran telah berada di halaman daftar kategori menu restoran dan daftar kategori menu restoran telah ditampilkan oleh sistem
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Operator restoran menekan tombol <i>edit</i> pada kategori yang ingin diubah</li> <li>2. Sistem menampilkan formulir <i>dialog</i> untuk mengubah kategori menu restoran</li> <li>3. Operator restoran menginputkan perubahan kategori menu restoran dan menekan tombol <i>save</i></li> <li>4. Sistem menyimpan perubahan kategori menu restoran</li> </ol>
<i>Alternative Flow</i>	<ol style="list-style-type: none"> <li>1. Jika Pelanggan memilih tombol cancel logout pada dialog, dialog tertutup kembali</li> </ol>
<i>Post Condition</i>	Operator restoran telah mengubah kategori menu restoran

**Tabel 5.33 Skenario *Use Case* Menambah Kategori Menu Restoran**

<i>Use Case</i>	Menambah Kategori Menu Restoran
Kode terkait kebutuhan	SMA-F-006
<i>Actor</i>	Operator Restoran
<i>Target</i>	Operator restoran dapat menambah kategori menu restoran
<i>Pre-Condition</i>	Operator restoran telah berada di halaman daftar menu restoran dan formulir untuk menambah kategori menu restoran telah ditampilkan
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Operator restoran menekan tombol floating action button bergambar logo “tambah”</li> <li>2. Sistem menampilkan formulir dialog untuk menambah kategori menu restoran</li> <li>3. Operator restoran menginputkan detail kategori menu restoran yang baru dan menekan tombol <i>save</i></li> <li>4. Sistem menyimpan kategori menu restoran</li> </ol>
<i>Alternative Flow</i>	<ol style="list-style-type: none"> <li>1. Jika Pelanggan memilih tombol cancel logout pada dialog, dialog tertutup kembali</li> </ol>
<i>Post Condition</i>	Operator restoran telah menambah kategori menu restoran

**Tabel 5.34 Skenario *Use Case* Menghapus Kategori Menu Restoran**

<i>Use Case</i>	Menghapus Kategori Menu Restoran
Kode terkait kebutuhan	SMA-F-007
<i>Actor</i>	Operator Restoran
<i>Target</i>	Operator restoran dapat menghapus kategori menu restoran
<i>Pre-Condition</i>	Operator restoran telah berada di halaman daftar kategori menu restoran dan daftar kategori menu restoran telah ditampilkan oleh sistem
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Operator restoran menekan tombol “delete” pada kategori menu restoran yang ingin dihapus</li> <li>2. Sistem menampilkan <i>dialog</i> konfirmasi penghapusan</li> </ol>

	<p>kategori menu restoran</p> <ol style="list-style-type: none"> <li>Operator restoran menekan tombol <i>delete</i></li> <li>Sistem menghapus kategori menu restoran pelanggan yang dipilih dan mengeluarkan pesan bahwa kategori menu restoran telah dihapus</li> </ol>
--	--

**Tabel 5.35 Skenario *Use Case* Melihat Daftar Menu Restoran**

<i>Use Case</i>	Melihat Daftar Menu Restoran
Kode terkait kebutuhan	SMA-F-008
<i>Actor</i>	Operator Restoran
<i>Target</i>	Operator restoran melihat daftar menu restoran
<i>Pre-Condition</i>	Operator restoran telah berada di halaman <i>dashboard</i>
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>Operator restoran memilih halaman daftar <i>category</i></li> <li>Sistem menampilkan halaman daftar <i>category</i></li> <li>Operator restoran memilih <i>category item</i> yang ingin dilihat menunya</li> <li>Sistem menampilkan halaman menu restoran</li> <li>Sistem menampilkan halaman daftar menu restoran</li> </ol>
<i>Alternative Flow</i>	-
<i>Post Condition</i>	Operator restoran telah melihat daftar menu restoran

**Tabel 5.36 Skenario *Use Case* Mengubah Menu Restoran**

<i>Use Case</i>	Mengubah Menu Restoran
Kode terkait kebutuhan	SMA-F-009
<i>Actor</i>	Operator Restoran
<i>Target</i>	Operator restoran dapat mengubah menu restoran
<i>Pre-Condition</i>	Operator restoran telah berada di halaman daftar menu restoran dan daftar menu restoran telah ditampilkan oleh sistem
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>Operator restoran memilih menu restoran yang ingin diubah</li> <li>Sistem menampilkan formulir <i>dialog</i> untuk mengubah menu restoran</li> </ol>

	3. Operator restoran menginputkan perubahan menu restoran dan menekan tombol <i>save</i> 4. Sistem menyimpan perubahan menu restoran
<i>Alternative Flow</i>	1. Jika Pelanggan memilih tombol cancel logout pada dialog, dialog tertutup kembali
<i>Post Condition</i>	Operator restoran telah mengubah menu restoran

**Tabel 5.37 Skenario *Use Case* Menambah Menu Restoran**

<i>Use Case</i>	Menambah Menu Restoran
Kode terkait kebutuhan	SMA-F-010
<i>Actor</i>	Operator Restoran
<i>Target</i>	Operator restoran dapat menambah menu restoran
<i>Pre-Condition</i>	Operator restoran telah berada di halaman daftar menu restoran dan formulir untuk menambah menu restoran telah ditampilkan
<i>Main Flow</i>	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 <i>save</i> 4. Sistem menyimpan menu restoran
<i>Alternative Flow</i>	1. Jika Pelanggan memilih tombol cancel logout pada dialog, dialog tertutup kembali
<i>Post Condition</i>	Operator restoran telah menambah menu restoran

**Tabel 5.38 Skenario *Use Case* Menghapus Menu Restoran**

<i>Use Case</i>	Menghapus Pesanan
Kode terkait kebutuhan	SMA-F-011
<i>Actor</i>	Operator Restoran
<i>Target</i>	Operator restoran dapat menghapus menu restoran
<i>Pre-Condition</i>	Operator restoran telah berada di halaman daftar menu restoran dan daftar menu restoran telah ditampilkan oleh

	sistem
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Operator restoran memilih menu restoran yang ingin dihapus</li> <li>2. Sistem menampilkan <i>dialog</i> konfirmasi penghapusan menu restoran</li> <li>3. Operator restoran menekan tombol <i>delete</i></li> <li>4. Sistem menghapus menu restoran pelanggan yang dipilih dan mengeluarkan pesan bahwa menu restoran telah dihapus</li> </ol>
<i>Alternative Flow</i>	-
<i>Post Condition</i>	Operator restoran telah menghapus menu restoran pelanggan

**Tabel 5.39 Skenario *Use Case* Mengubah Ketersediaan Menu Restoran**

<i>Use Case</i>	Mengatur Stock Menu Restoran
Kode terkait kebutuhan	SMA-F-012
<i>Actor</i>	Operator Restoran
<i>Target</i>	Operator restoran dapat mengatur ketersediaan menu restoran
<i>Pre-Condition</i>	Operator restoran telah berada di halaman daftar menu restoran dan daftar menu restoran telah ditampilkan
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Operator restoran memilih menu restoran yang ingin diubah stocknya menjadi tersedia atau tidak tersedia dengan menekan tombol <i>switch</i> pada menu restoran yang dipilih</li> <li>2. Sistem mengubah ketersediaan menu restoran</li> </ol>
<i>Alternative Flow</i>	-
<i>Post Condition</i>	Operator restoran telah mengubah ketersediaan menu restoran

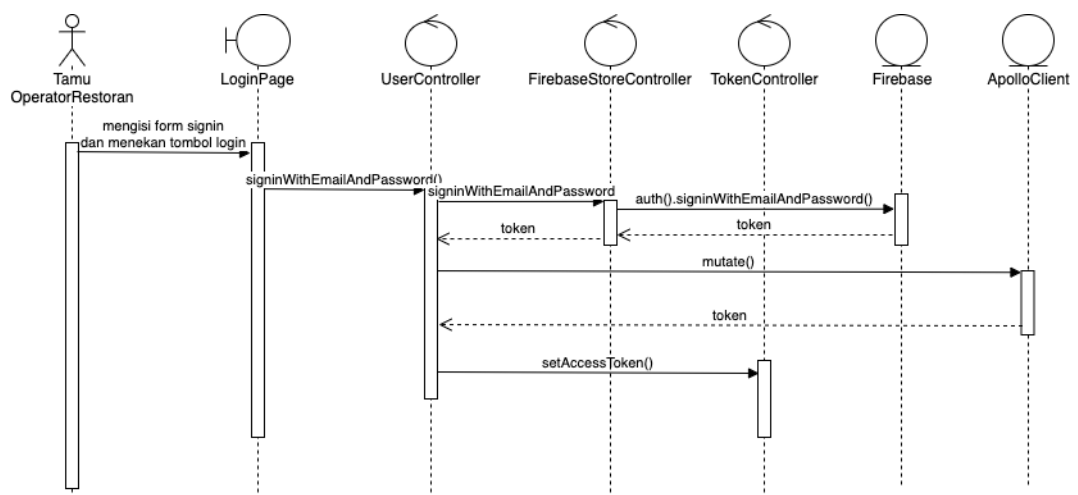
**Tabel 5.40 Skenario *Use Case* Mengatur Ulang Ketersediaan Menu Restoran**

<i>Use Case</i>	Mengatur Stock Menu Restoran
Kode terkait kebutuhan	SMA-F-013

<i>Actor</i>	Operator Restoran
<i>Target</i>	Operator restoran dapat mengatur ketersediaan stock menu restoran
<i>Pre-Condition</i>	Operator restoran telah berada di halaman daftar stock menu restoran dan daftar menu stock restoran telah ditampilkan
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Operator restoran melakukan navigasi ke halaman “dashboard”</li> <li>2. Sistem melakukan melakukan pengaturan ulang ketersediaan menu restoran menjadi tersedia</li> </ol>
<i>Alternative Flow</i>	-
<i>Post Condition</i>	Operator restoran telah mengatur ulang ketersediaan menu restoran

#### 5.7.2.2 Sequence Diagram

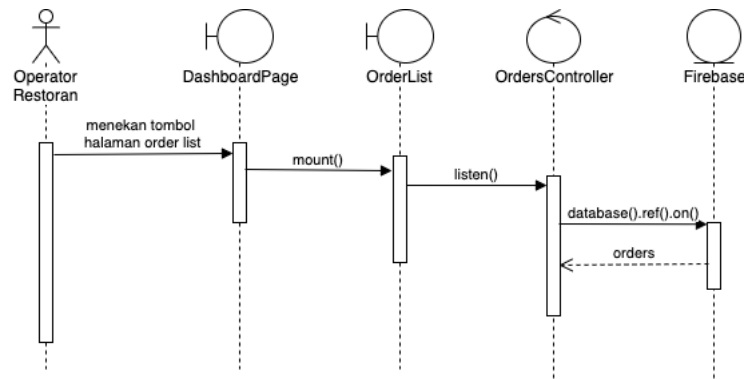
*Sequence diagram* menggambarkan urutan proses yang terjadi di dalam sistem untuk mencapai tujuan dari kebutuhan fungsionalitas sistem. Daftar *sequence diagram* dalam sistem manajemen antrian pesanan menu restoran di *sprint* kedua dapat dilihat pada Gambar 5.31 sampai Gambar 5.43.



**Gambar 5.31 Sequence Diagram Use Case Login Restaurant**

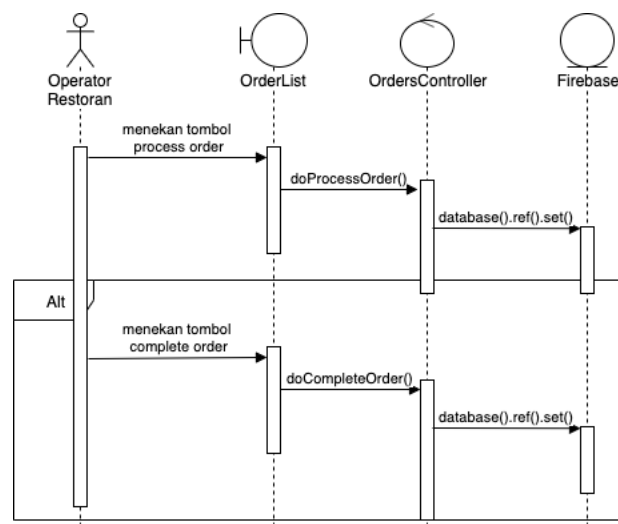
Pada *sequence diagram* Login Restaurant pada Gambar 5.31 dijelaskan bahwa aktor Tamu Operator Restaurant menekan tombol “signin with google” pada halaman LoginPage. Kemudian dilakukan pemanggilan fungsi `signinWithEmailAndPassword()` pada UserController. Setelah pelanggan berhasil login dengan *credentials* akun google pelanggan, maka dilakukan request ke server untuk melakukan *signin*. Nilai kembalian dari request login adalah

*token* yang digunakan untuk *authorization* pengguna saat melakukan *request* untuk mengakses *endpoints* pada *server*. Kemudian dilakukan pemanggilan fungsi *setAccessToken()* pada *TokenController* untuk menyimpan data pada *local storage browser* dan mengatur *default header* dari *service request data axios*.



**Gambar 5.32 Sequence Diagram Use Case Melihat Daftar Pesanan**

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

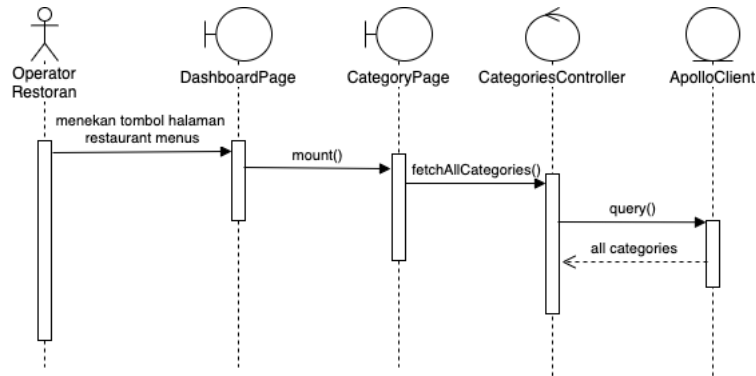


**Gambar 5.33 Sequence Diagram Use Case Mengubah Status Pesanan**

Pada *sequence diagram* Mengubah Status Pesanan pada Gambar 5.33 dijelaskan bahwa aktor operator restoran menekan tombol “process order” pada

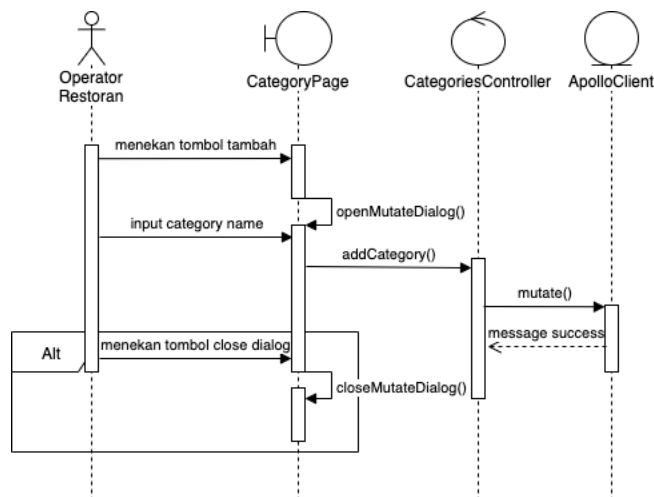


halaman OrderList. Aktivitas tersebut memicu pemanggilan `doProcessOrder()` pada `OrdersController`. Kemudian dilakukan *update field* pada model `Orders` pada `Firebase Firestore`. Terdapat *alternative flow* ketika operator restoran menekan tombol *complete order* pada halaman OrderList, maka akan memicu pemanggilan fungsi `doCompleteOrder()` pada `OrdersController`. Kemudian dilakukan *update order field* pada `Firebase Firestore`.



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

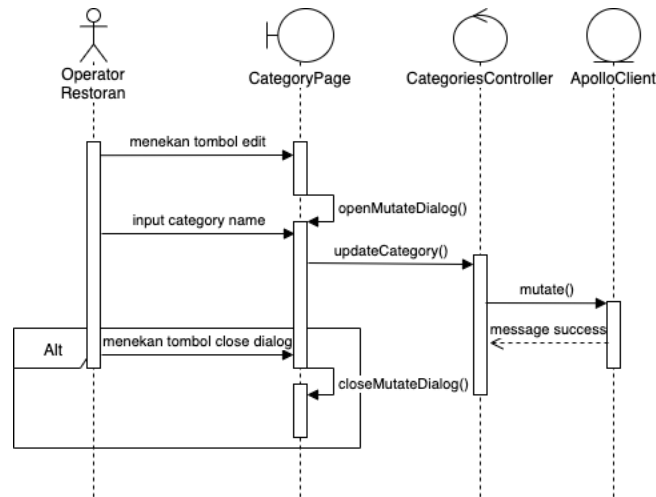
Pada *sequence diagram* Melihat Daftar Kategori Menu Restoran pada Gambar 5.38 dijelaskan bahwa aktor operator restoran menekan tombol halaman restoran menus pada halaman `DashboardPage`. Kemudian dilakukan *mounting component* pada `CategoryPage`. Setelah itu dilakukan pemanggilan `fetchAllCategories()` pada `CategoriesController`. Aktivitas tersebut memicu *request all categories* ke `ApolloClient` dan mendapatkan nilai kembalian berupa *all categories* yang disimpan dalam *observable property*.



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

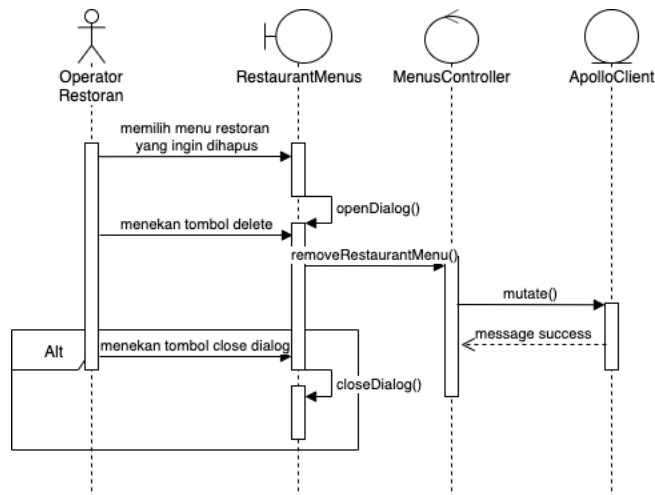
Pada *sequence diagram* Menambah Kategori Menu Restoran pada Gambar 5.39 dijelaskan bahwa aktor operator restoran menekan tombol tambah pada

*category item* halaman *CategoryPage*. Kemudian dilakukan *self-call* `openMutateDialog()`. Setelah itu operator restoran *input* menu restoran. Aktivitas tersebut memicu pemanggilan `addCategory()` pada *CategoriesController* dan melakukan *request add category* pada *ApolloClient*, kemudian mendapatkan nilai kembalian berupa *message success*. Apabila operator restoran menekan tombol *close dialog*, maka dilakukan *self-call* `closeMutateDialog()` pada *CategoryPage*.



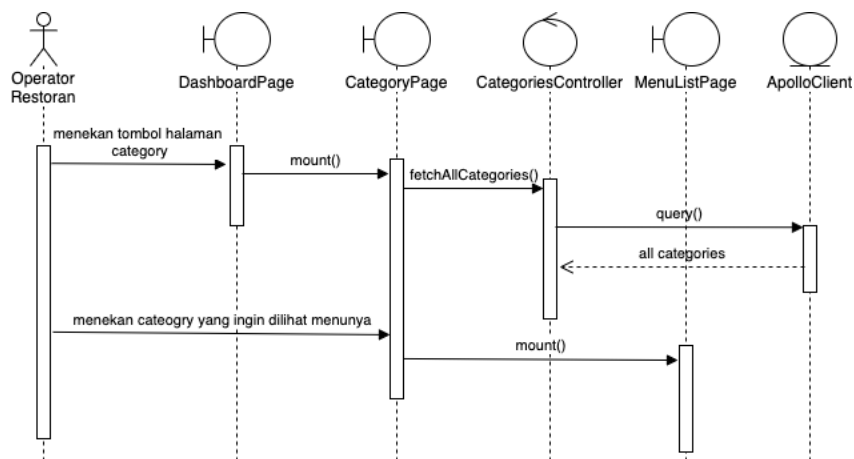
**Gambar 5.36 Sequence Diagram Use Case Mengubah Kategori Menu Restoran**

Pada *sequence diagram* Mengubah Kategori Menu Restoran pada Gambar 5.39 dijelaskan bahwa aktor operator restoran menekan tombol update tambah pada *category item* halaman *CategoryPage*. Kemudian dilakukan *self-call* `openMutateDialog()`. Setelah itu operator restoran *input* menu restoran. Aktivitas tersebut memicu pemanggilan `updateCategory()` pada *CategoriesController* dan melakukan *request update category* pada *ApolloClient*, kemudian mendapatkan nilai kembalian berupa *message success*. Apabila operator restoran menekan tombol *close dialog*, maka dilakukan *self-call* `closeMutateDialog()` pada *CategoryPage*.



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

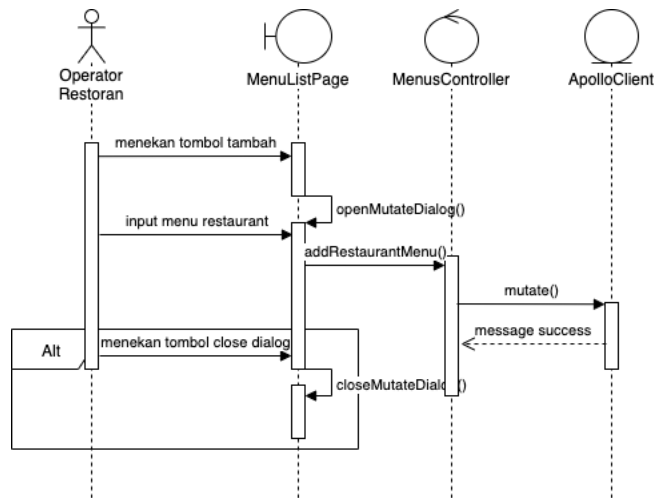
Pada *sequence diagram* Menghapus Kategori Menu Restoran pada Gambar 5.39 dijelaskan bahwa aktor operator restoran menekan tombol tambah pada halaman *CategoryPage*. Kemudian dilakukan *self-call* `openMutateDialog()`. Setelah itu operator restoran *input* menu restoran. Aktivitas tersebut memicu pemanggilan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



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

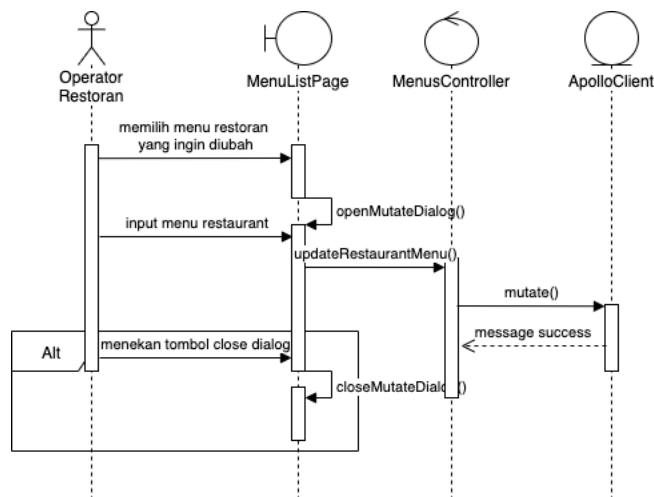
Pada *sequence diagram* Melihat Menu Restoran pada Gambar 5.38 dijelaskan bahwa aktor operator restoran menekan tombol halaman restoran menus pada halaman *DashboardPage*. Kemudian dilakukan *mounting component* pada *CategoryPage*. Setelah itu dilakukan pemanggilan `fetchAllCategories()` pada *CategoriesController*.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

operator restoran menekan *category item* yang ingin dilihat menyunya dan kemudian sistem melakukan *mounting component* Menulistpage



**Gambar 5.39 Sequence Diagram Use Case Menambah Menu Restoran**

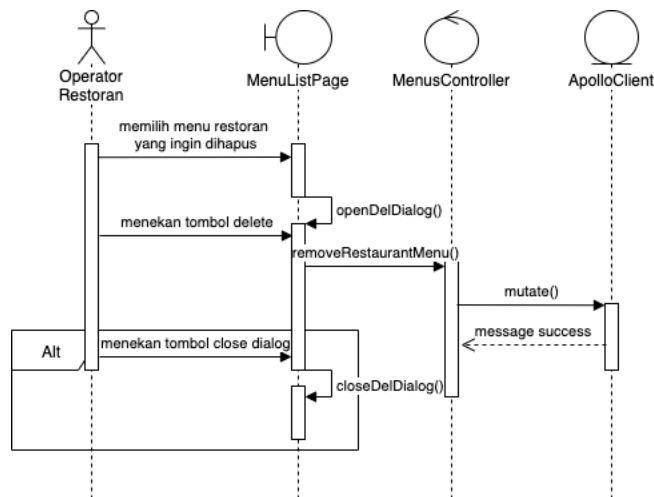
Pada *sequence diagram* Menambah Menu Restoran pada Gambar 5.39 dijelaskan bahwa aktor operator restoran menekan tombol tambah pada halaman RestaurantMenus. Kemudian dilakukan *self-call* `openMutateDialog()`. Setelah itu operator restoran *input* menu restoran. Aktivitas tersebut memicu pemanggilan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



**Gambar 5.40 Sequence Diagram Use Case Mengubah Menu Restoran**

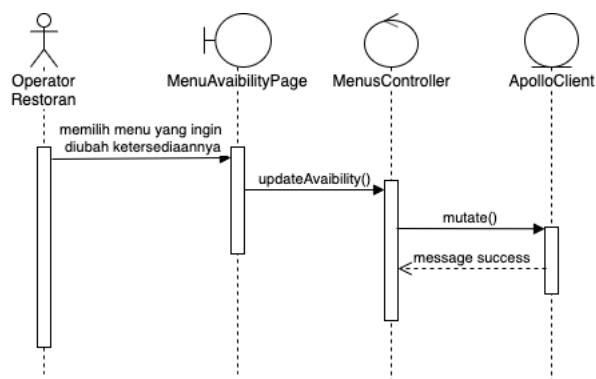
Pada *sequence diagram* Menambah Menu Restoran pada Gambar 5.40 dijelaskan bahwa aktor operator restoran memilih menu yang ingin diubah pada halaman `MenuListPage`. Kemudian dilakukan *self-call*

`openMutateDialog()`. Setelah itu operator restoran *input* menu restoran. Aktivitas tersebut memicu pemanggilan `updateRestaurantMenu()` pada `MenusController` dan melakukan *request update restoran menu* pada `ApolloClient`, kemudian mendapatkan nilai kembalian berupa *message success*. Apabila operator restoran menekan tombol *close dialog*, maka dilakukan *self-call* `closeMutateDialog()` pada `MenuListPage`.



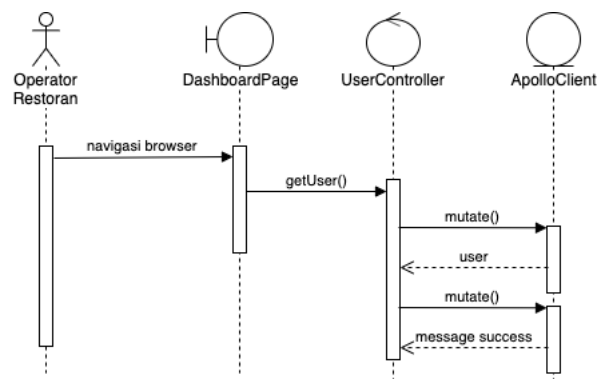
**Gambar 5.41 Sequence Diagram Use Case Menghapus Menu Restoran**

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



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

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

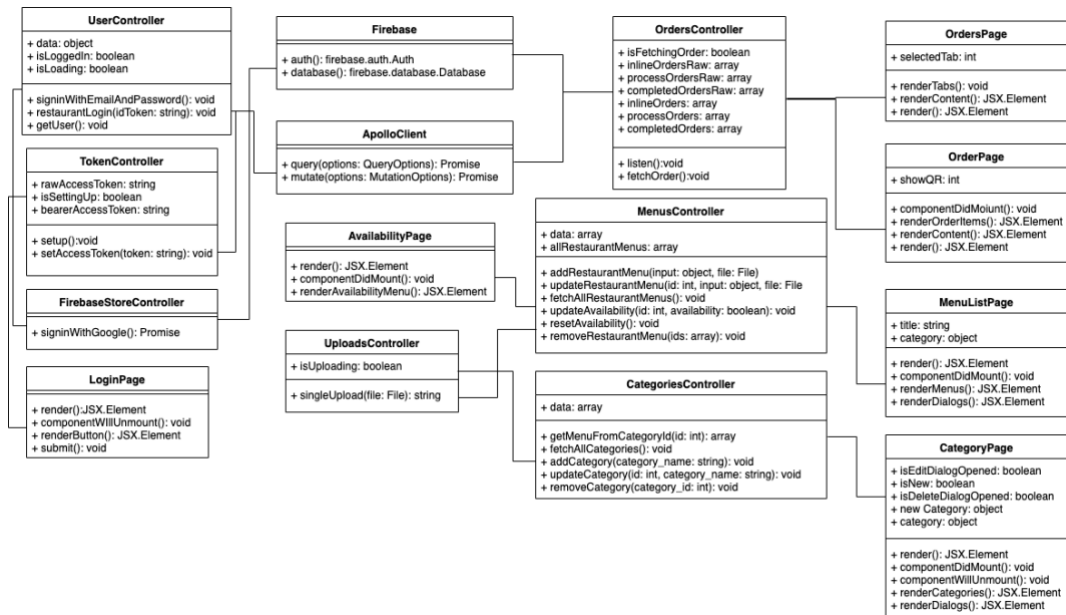


**Gambar 5.43 Sequence Diagram Use Case Mengatur Ulang Ketersediaan Menu Restoran**

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

### 5.7.2.3 Class Diagram

*Class Diagram* merupakan diagram yang merepresentasikan *class* yang akan digunakan dalam fase implementasi. *Classes* yang terlibat dalam *class diagram* didasarkan pada *backlog* yang telah dibuat sebelumnya. *Class Diagram* yang digunakan dalam *backlog sprint* ketiga ini digambarkan pada Gambar 5.44.

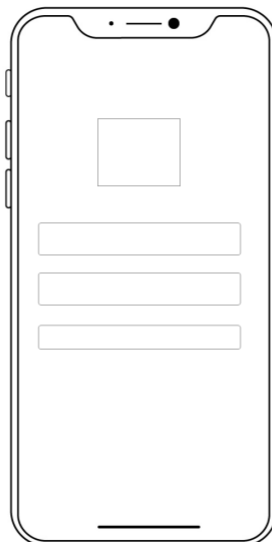


**Gambar 5.44 Class Diagram** aplikasi manajemen antrean pesanan menu restoran pada *sprint* ketiga

#### 5.7.2.4 Perancangan Antarmuka

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

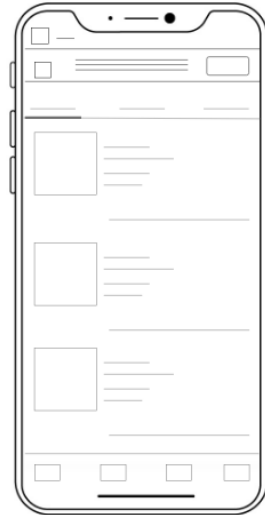
##### (a) Perancangan Antarmuka Sign-in



#### **Gambar 5.45 Wireframe Sign-in**

Pada Gambar 5.11 menunjukkan halaman *sign-in*. Halaman *sign-in* adalah halaman untuk memasukkan pengguna ke dalam sistem. Pada halaman *sign-in*, terdapat logo, *input text* email dan password, dan tombol “Sign in”. Ketika pengguna menekan tombol “Sign in” maka sistem akan melakukan otentikasi sesuai dengan *input* email dan *password* yang telah pengguna masukkan.

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



#### **Gambar 5.46 Wireframe Orders**

Pada Gambar 5.11 menunjukkan halaman *orders*. Halaman *orders* menunjukkan halaman *orders*. Halaman *orders* adalah halaman untuk melihat riwayat pemesanan yang dilakukan oleh operator restoran. Pada halaman *orders*, terdapat daftar pesanan pelanggan pada restoran yang ditampilkan dalam bentuk *card*. Ketika salah satu *order card* diklik, maka sistem akan menunjukkan halaman *Order Detail* yang berisi detail pemesanan.



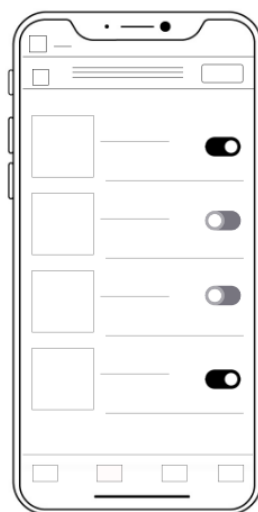
**(c) Perancangan Antarmuka *Order Detail***



**Gambar 5.47 Wireframe *Order Detail***

Pada Gambar 5.16 menunjukkan halaman *order detail*. Halaman *order detail* adalah halaman untuk melihat detail pesanan pelanggan. Pada halaman *order detail*, terdapat detail pemesanan yang dilakukan pelanggan yang meliputi data-data restoran, detail harga, status pemesanan, dan menu yang dipesan. Selain itu, jika status pesanan masih berstatus “*INLINE*”, maka akan terdapat tombol “*process order*” untuk mengubah status pesanan menjadi “*PROCESS*”. Namun apabila pesanan masih berstatus “*PROCESS*”, maka akan terdapat tombol “*complete order*” untuk mengubah status pesanan menjadi “*COMPLETED*”.

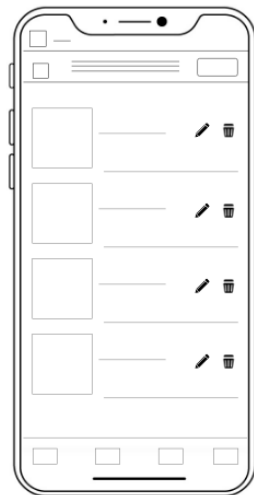
**(d) Perancangan Antarmuka *Menu Availability***



**Gambar 5.48 Wireframe Menu Availability**

Pada Gambar 5.17 menunjukkan halaman *menu availability*. Halaman *menu availability* adalah halaman untuk mengubah ketersediaan menu restoran menjadi tersedia atau tidak tersedia. Pada halaman *menu availability* terdapat daftar menu restoran dan *switch button* untuk mengubah ketersediaan menu restoran.

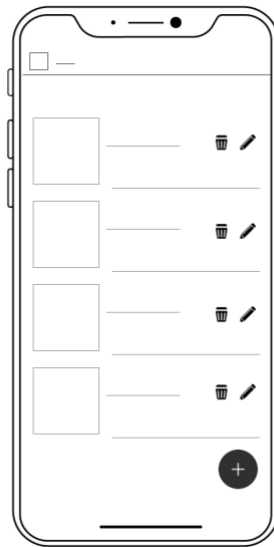
**(e) Perancangan Antarmuka *Category List***



**Gambar 5.49 Wireframe *Category List***

Pada Gambar 5.18 menunjukkan halaman *category list*. Halaman *category list* adalah halaman untuk melihat seluruh list kategori menu makanan yang ada di restoran. Pada halaman *category list*, terdapat terdapat *card* yang menampilkan nama kategori dan gambar salah satu menu dalam kategori tersebut, tombol *edit*, dan tombol *delete*. *Category card* merupakan list yang dapat diklik untuk dapat membuka halaman *list menu* yang ada pada kategori tersebut. Selain itu, dalam setiap *category card*, terdapat tombol *add* berupa *floating action button* untuk menambah kategori. Tombol *edit* dan *delete* di setiap kategori digunakan untuk mengubah dan menghapus kategori, sedangkan tombol *add* digunakan untuk menambah kategori baru.

**(f) Perancangan Antarmuka Menu List**



**Gambar 5.50 Wireframe Menu List**

Pada Gambar 5.18 menunjukkan halaman *menu list*. Halaman *menu list* adalah halaman untuk melihat seluruh list kategori menu makanan yang ada di restoran. Pada halaman *menu list*, terdapat terdapat *card* yang menampilkan nama menu, gambar menu, tombol *edit*, dan tombol *delete*. Selain itu, dalam setiap *menu card*, terdapat tombol *add* berupa *floating action button* untuk menambah menu. Tombol *edit* dan *delete* di setiap menu digunakan untuk mengubah dan menghapus menu, sedangkan tombol *add* digunakan untuk menambah menu baru dalam kategori yang telah dipilih oleh pelanggan.

**5.7.2.5 Perancangan Algoritme**

Pada bagian ini akan dilakukan perancangan algoritme untuk mendapatkan gambaran langkah-langkah dari suatu proses. Salah satu perancangan algoritme mengubah status pesanan menjadi "PROCESS" dapat dilihat pada Tabel 5.18.

**Tabel 5.41 Perancangan algoritme mengubah status pesanan menjadi "PROCESS"**

1	Begin
2	Deklarasi id untuk inlineOrder yang ingin diubah statusnya
3	Request set firebase dengan referensi `/Orders/\${id}/status` menjadi `PROCESS`
4	Request set firebase firestore dengan referensi `/Orders/\${id}/updated_at` menjadi Date saat ini
5	Catch error request
6	Begin
7	Show error message
8	End
9	end

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

### 5.7.2.6 Implementasi Kode Program

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

**Tabel 5.42 Implementasi kode program *create order***

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

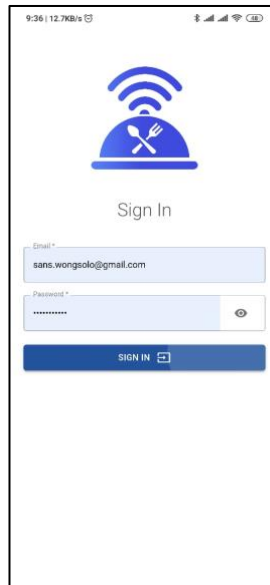
Pembahasan dari kode program diatas adalah sebagai berikut:

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

### 5.7.2.7 Implementasi Antarmuka

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

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



**Gambar 5.51 Hasil Implementasi Antarmuka *Sign-in***

Pada Gambar 5.11 menunjukkan halaman *sign-in*. Halaman *sign-in* adalah halaman untuk memasukkan pengguna ke dalam sistem. Pada halaman *sign-in*, terdapat logo, *input* text email dan password, dan tombol "Sign in". Ketika pengguna menekan tombol "Sign in" maka sistem akan melakukan otentikasi sesuai dengan *input* email dan *password* yang telah pengguna masukkan.

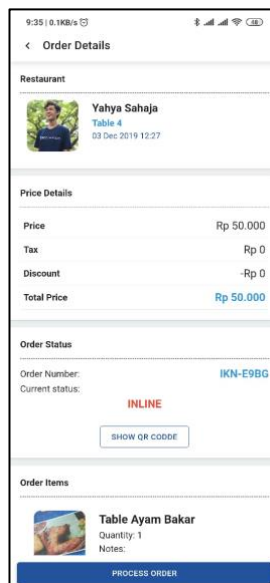
**(b) Hasil Implementasi Antarmuka *Orders***



**Gambar 5.52 Hasil Implementasi Antarmuka *Orders***

Pada Gambar 5.11 menunjukkan halaman *orders*. Halaman *orders* menunjukkan halaman *orders*. Halaman *orders* adalah halaman untuk melihat riwayat pemesanan yang dilakukan oleh operator restoran. Pada halaman *orders*, terdapat daftar pesanan pelanggan pada restoran yang ditampilkan dalam bentuk *card*. Ketika salah satu *order card* diklik, maka sistem akan menunjukkan halaman *Order Detail* yang berisi detail pemesanan.

**(c) Hasil Implementasi Antarmuka *Order Detail***

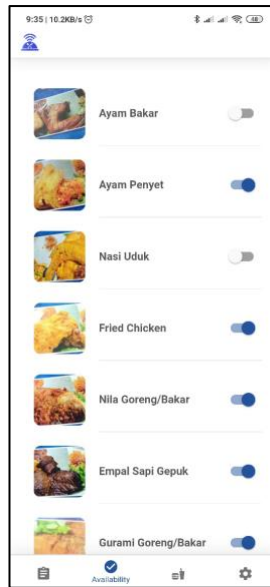


**Gambar 5.53 Hasil Implementasi Antarmuka *Order Detail***

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

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

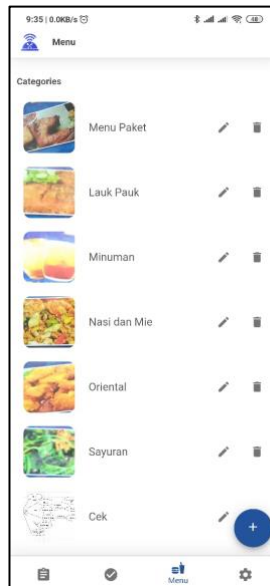
**(d) Hasil Implementasi Antarmuka *Menu Availability***



**Gambar 5.54 Hasil Implementasi Antarmuka *Menu Availability***

Pada Gambar 5.17 menunjukkan halaman *menu availability*. Halaman *menu availability* adalah halaman untuk mengubah ketersediaan menu restoran menjadi tersedia atau tidak tersedia. Pada halaman *menu availability* terdapat daftar menu restoran dan *switch button* untuk mengubah ketersediaan menu restoran.

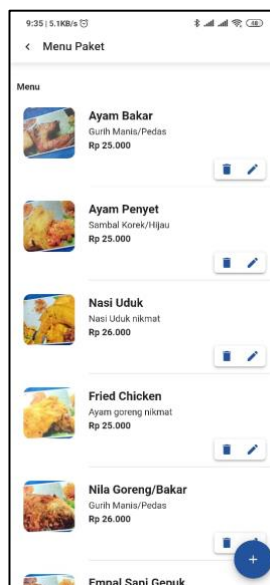
**(e) Hasil Implementasi Antarmuka *Category List***



**Gambar 5.55 Hasil Implementasi Antarmuka *Category List***

Pada Gambar 5.18 menunjukkan halaman *category list*. Halaman *category list* adalah halaman untuk melihat seluruh list kategori menu makanan yang ada di restoran. Pada halaman *category list*, terdapat terdapat *card* yang menampilkan nama kategori dan gambar salah satu menu dalam kategori tersebut, tombol *edit*, dan tombol *delete*. *Category card* merupakan list yang dapat diklik untuk dapat membuka halaman *list menu* yang ada pada kategori tersebut. Selain itu, dalam setiap *category card*, terdapat tombol *add* berupa *floating action button* untuk menambah kategori. Tombol *edit* dan *delete* di setiap kategori digunakan untuk mengubah dan menghapus kategori, sedangkan tombol *add* digunakan untuk menambah kategori baru.

**(f) Hasil Implementasi Antarmuka *Menu List***





### Gambar 5.56 Hasil Implementasi Antarmuka *Menu List*

Pada Gambar 5.18 menunjukkan halaman *menu list*. Halaman *menu list* adalah halaman untuk melihat seluruh list kategori menu makanan yang ada di restoran. Pada halaman *menu list*, terdapat terdapat *card* yang menampilkan nama menu, gambar menu, tombol *edit*, dan tombol *delete*. Selain itu, dalam setiap *menu card*, terdapat tombol *add* berupa *floating action button* untuk menambah menu. Tombol *edit* dan *delete* di setiap menu digunakan untuk mengubah dan menghapus menu, sedangkan tombol *add* digunakan untuk menambah menu baru dalam kategori yang telah dipilih oleh pelanggan.

#### 5.7.2.8 Evaluasi dan demonstrasi produk

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

##### (a) Evaluasi

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

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

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

	gagal	gagal	
--	-------	-------	--

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

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

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

<i>Test Case</i>	<i>Expectation</i>	<i>Actual Result</i>	<i>Validation</i>
Melihat daftar kategori menu restoran	Sistem berhasil menampilkan daftar kategori menu restoran	Sistem berhasil menampilkan daftar kategori menu restoran	<i>Valid</i>

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

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

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

<i>Test Case</i>	<i>Expectation</i>	<i>Actual Result</i>	<i>Validation</i>
------------------	--------------------	----------------------	-------------------

Mengubah kategori menu restoran dengan menekan tombol "edit" pada daftar kategori menu restoran yang dipilih	Sistem berhasil mengubah kategori menu restoran	Sistem berhasil mengubah kategori menu restoran	<i>Valid</i>
--	---	---	--------------

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

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

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

<i>Test Case</i>	<i>Expectation</i>	<i>Actual Result</i>	<i>Validation</i>
Melihat daftar menu restoran pada kategori yang telah dipilih oleh pelanggan	Sistem berhasil menampilkan daftar menu restoran	Sistem berhasil menampilkan daftar menu restoran	<i>Valid</i>

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

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

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

<i>Test Case</i>	<i>Expectation</i>	<i>Actual Result</i>	<i>Validation</i>
Mengubah menu restoran dengan menekan tombol “edit” pada daftar menu restoran yang dipilih	Sistem berhasil mengubah menu restoran	Sistem berhasil mengubah menu restoran	<i>Valid</i>

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

<i>Test Case</i>	<i>Expectation</i>	<i>Actual Result</i>	<i>Validation</i>
Menghapus menu restoran dengan menekan tombol “delete” pada daftar menu restoran yang dipilih	Sistem berhasil menghapus menu restoran	Sistem berhasil menghapus menu restoran	<i>Valid</i>

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

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

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

<i>Test Case</i>	<i>Expectation</i>	<i>Actual Result</i>	<i>Validation</i>
Mengatur ulang ketersediaan menu restoran menjadi “tersedia” dengan melakukan	Sistem berhasil mengubah ketersediaan menu restoran kembali menjadi “tersedia”	Sistem berhasil mengubah ketersediaan menu restoran kembali menjadi “tersedia”	<i>Valid</i>

akses kepada sistem pada hari yang berbeda dari terakhir kali operator restoran mengakses sistem			
--	--	--	--

### (b) Demonstrasi Produk

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

Dari demonstrasi produk pada *sprint* kedua ini, didapatkan *feedback* dari operator restoran bahwa urutan pesanan yang mengantre seharusnya diurutkan secara *ascending* dari yang terdahulu hingga yang terkini, sehingga pemilihan pesanan untuk diproses dapat lebih mudah. Selain itu, dibutuhkan *field* yang menampilkan tanggal dan waktu pesanan dibuat atau diubah. Adapun terdapat foto yang menunjukkan demonstrasi produk yang dilakukan kepada pengguna dapat dilihat pada LAMPIRAN B.

### 5.7.3 Sprint Ketiga

#### 5.7.3.1 Use Case Scenario

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

**Tabel 5.55 Skenario Use Case Mengubah Profil Restoran**

<i>Use Case</i>	Mengubah Profil Restoran
Kode terkait kebutuhan	SMA-F-010
<i>Actor</i>	Operator Restoran
<i>Target</i>	Operator restoran dapat mengubah profil restoran
<i>Pre-Condition</i>	Operator restoran telah berada di halaman dashboard
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Operator restoran menekan tombol profil pada menu navigasi</li> <li>2. Sistem menampilkan formulir untuk mengubah profil restoran</li> <li>3. Operator restoran menginputkan perubahan profil restoran dan menekan tombol <i>save</i></li> </ol>

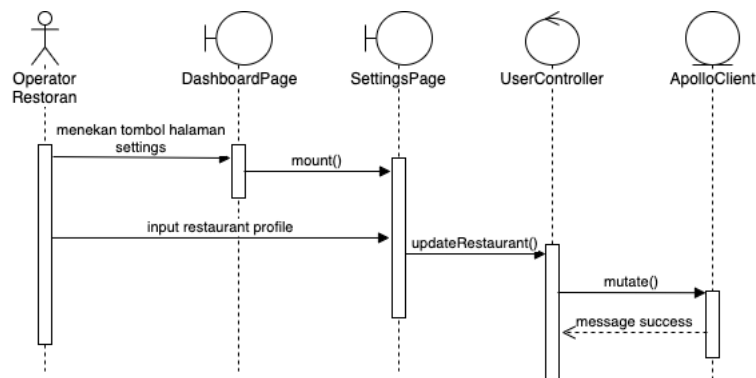
	4. Sistem menyimpan perubahan profil restoran
<i>Alternative Flow</i>	-
<i>Post Condition</i>	Operator restoran telah mengubah profil restoran

**Tabel 5.56 Skenario *Use Case* Logout Operator Restoran**

<i>Use Case</i>	Logout Pelanggan
Kode kebutuhan terkait	SA-F-010
<i>Actor</i>	Operator Restoran
<i>Target</i>	Otentikasi operator restoran dapat keluar dari sistem.
<i>Pre-Condition</i>	Pelanggan sudah berada di halaman <i>dashboard</i>
<i>Main Flow</i>	<ol style="list-style-type: none"> <li>1. Pelanggan menekan tombol <i>logout</i> pada menu navigasi</li> <li>2. Sistem menampilkan dialog konfirmasi untuk <i>logout</i></li> <li>3. Pelanggan memilih tombol <i>Logout</i></li> <li>4. Sistem menampilkan halaman login dan menghapus informasi pengguna dari penyimpanan <i>local</i></li> </ol>
<i>Alternative Flow</i>	-
<i>Post Condition</i>	Otentikasi operator restoran telah keluar dari sistem.

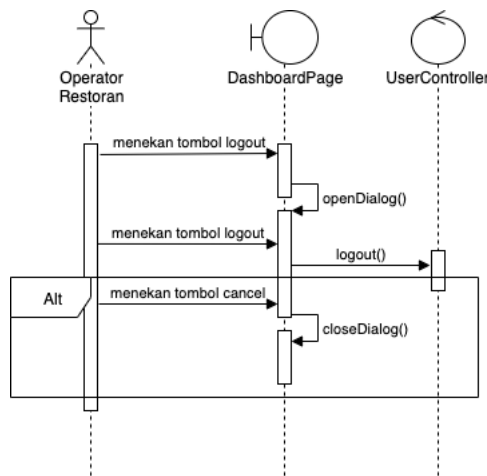
### 5.7.3.2 Sequence Diagram

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



**Gambar 5.57 Sequence Diagram Use Case Mengubah Profil Restoran**

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

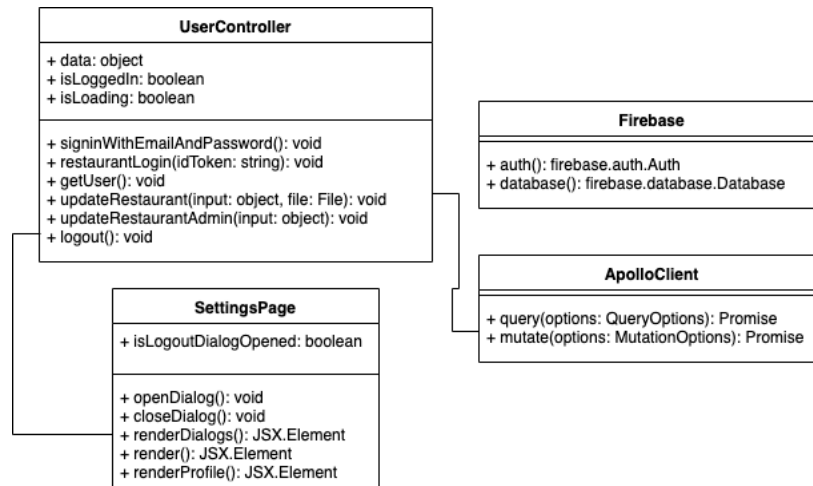


**Gambar 5.58 Sequence Diagram Use Case Logout Operator Restoran**

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

### 5.7.3.3 Class Diagram

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

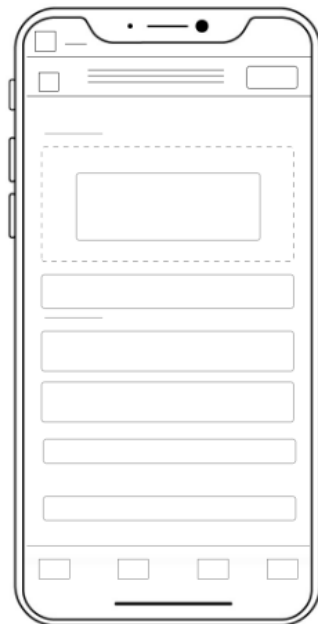


**Gambar 5.59 Class Diagram aplikasi manajemen antrean pesanan menu restoran pada *sprint* ketiga**

#### 5.7.3.4 Perancangan Antarmuka

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

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





### Gambar 5.60 Wireframe Settings

Pada Gambar 5.20 menunjukkan halaman *settings*. Halaman *settings* adalah halaman untuk melihat profil operator restoran yang telah masuk ke dalam sistem. Pada halaman *settings*, terdapat *form* untuk melakukan perubahan pada profil restoran dan profil operator restoran. Selain itu, terdapat tombol *logout* untuk mengeluarkan pelanggan dari sistem.

#### 5.7.3.5 Perancangan Algoritme

Pada bagian ini akan dilakukan perancangan algoritme untuk mendapatkan gambaran langkah-langkah dari suatu proses. Salah satu perancangan algoritme mengubah status pesanan menjadi "PROCESS" dapat dilihat pada Tabel 5.18.

**Tabel 5.57 Perancangan algoritme mengubah status pesanan menjadi "PROCESS"**

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

Algoritme *update restaurant* merupakan fungsi untuk mengubah data restoran. Diawali dengan mendeklarasikan id untuk pesanan yang akan diubah statusnya. Kemudian dilakukan *request firebase* pada referensi `'/Orders/${id}/status'` menjadi `'PROCESS'`.

#### 5.7.3.6 Implementasi Kode Program

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

**Tabel 5.58 Implementasi kode program *update restaurant***

1	@action
2	async updateRestaurant( <i>input</i> , <i>file</i> ) {
3	try {
4	this.isUpdatingRestaurant = true
5	overlayLoading.show()
6	
7	if ( <i>file</i> ) {
8	<i>input.picture</i> = await uploads.singleUpload( <i>file</i> )
9	}
10	
11	if (isURL( <i>input.picture</i> )) {
12	let url = new URL( <i>input.picture</i> )

13	<code>input.picture = url.pathname</code>
14	<code>}</code>
15	
16	<code>await client.mutate({</code>
17	<code>mutation: updateRestaurantMutation,</code>
18	<code>variables: {</code>
19	<code>input</code>
20	<code>}</code>
21	<code>})</code>
22	
23	<code>await this.fetchRestaurant()</code>
24	<code>snackbar.show('Restaurant data updated')</code>
25	<code>} catch(err) {</code>
26	<code>this.isLoadingLogin = false</code>
27	<code>console.log('ERROR WHILE UPDATE RESTO', err.message)</code>
28	<code>snackbar.show(err.message)</code>
29	<code>return err.message</code>
30	<code>} finally {</code>
31	<code>this.isUpdatingRestaurant = false</code>
32	<code>overlayLoading.hide()</code>
33	<code>}</code>
34	<code>}</code>

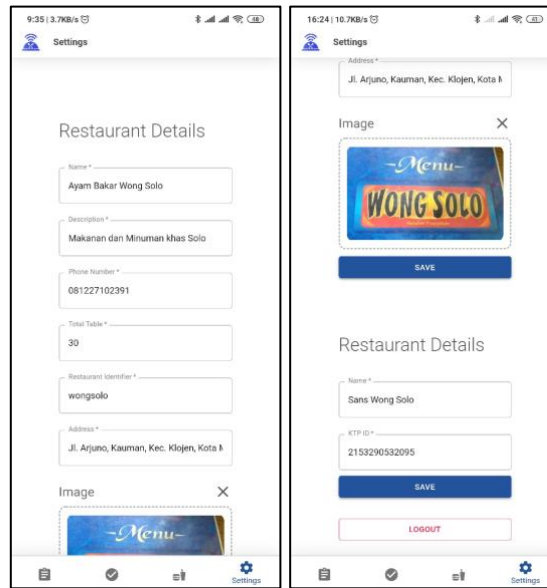
Pembahasan dari kode program diatas adalah sebagai berikut:

1. Baris 7 – 9: jika terdapat file, maka upload file terlebih dahulu
2. Baris 11 – 13: jika string `input.picture` memiliki *pattern* url, maka ubah nilai `input.picture` menjadi menjadi path dalam url `input.picture`.
3. Baris 10 – 13: *Request mutation graphql* untuk *update* restoran dengan data pada variable parameter `input`.

### 5.7.3.7 Implementasi Antarmuka

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

**(a) Hasil Implementasi Antarmuka *Settings***



**Gambar 5.61 Hasil Implementasi Antarmuka *Settings***

Pada Gambar 5.20 menunjukkan halaman *settings*. Halaman *settings* adalah halaman untuk melihat profil operator restoran yang telah masuk ke dalam sistem. Pada halaman *settings*, terdapat *form* untuk melakukan perubahan pada profil restoran dan profil operator restoran. Selain itu, terdapat terdapat tombol *logout* untuk mengeluarkan pelanggan dari sistem.

**5.7.3.8 Evaluasi**

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

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

<i>Test Case</i>	<i>Expectation</i>	<i>Actual Result</i>	<i>Validation</i>
Mengubah profil restoran dengan menginputkan data profil restoran dan menekan tombol "Save"	Sistem berhasil mengubah profil restoran sesuai dengan data <i>input</i> operator restoran	Sistem berhasil mengubah profil restoran sesuai dengan data <i>input</i> operator restoran	<i>Valid</i>

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

<i>Test Case</i>	<i>Expectation</i>	<i>Actual Result</i>	<i>Validation</i>
Melakukan logout dengan	Sistem berhasil mengeluarkan	Sistem berhasil mengeluarkan	<i>Valid</i>

mengeluarkan operator restoran dari sistem	operator restoran dari sistem	operator restoran dari sistem	
--	----------------------------------	----------------------------------	--

## BAB 6 PENGUJIAN

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

### 6.1 Pengujian *Usability*

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

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

**Tabel 6.1 Task Scenario Untuk Pelanggan Restoran**

Task	Nama Task	Langkah-langkah
1	Login	<ol style="list-style-type: none"><li>1. Menekan tombol "account" pada navigasi</li><li>2. Menekan tombol "Sign in"</li><li>3. Menekan tombol "Sign In with Google"</li><li>4. Login menggunakan akun google Anda</li></ol>
2	Memesan menu restoran	<ol style="list-style-type: none"><li>1. Menekan tombol Scan/Pindai</li><li>2. Memindai QR Code yang terdapat pada meja Anda</li><li>3. Memilih menu restoran yang Anda inginkan</li><li>4. Menekan tombol "Checkout"</li><li>5. Memastikan kembali pesanan dengan benar dan menekan tombol "Order"</li></ol>
3	Melihat riwayat pesanan	<ol style="list-style-type: none"><li>1. Menekan tombol "Orders" pada navigasi</li><li>2. Memilih riwayat pesanan yang ingin dilihat lebih detail</li></ol>
4	Mengubah pesanan	<ol style="list-style-type: none"><li>1. Menekan tombol "Update Order"</li><li>2. Melakukan perubahan pemesanan menu</li><li>3. Menekan tombol update order</li></ol>
5	Menambahkan menu ke daftar menu favorit dan melihat menu favorite	<ol style="list-style-type: none"><li>1. Menekan tombol "Orders" pada navigasi</li><li>2. Memilih riwayat pesanan yang ingin dilihat lebih detail</li><li>3. Menekan tombol love berbentuk hati</li><li>4. Menekan tombol "Back" berbentuk panah</li><li>5. Menekan tombol "Favorite Menu" pada navigasi</li></ol>

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

**Tabel 6.2 Task Scenario Untuk Operator Restoran**

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

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

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

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

**Tabel 6.3 Skor Skala Likert dari Setiap Pertanyaan**

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

**Tabel 6.4 Kuesioner SUPR-Qm untuk Pelanggan Restoran**

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

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

**Tabel 6.5 Kuesioner SUPR-Qm untuk Operator Restoran**

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



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

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

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

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

(6.1)

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

#### 6.1.1 Analisis Hasil Pengujian Usability untuk Pelanggan Restoran

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

restoran untuk memesan makanan dan memakan makanan di restoran. Adapun lima responden tersebut dijelaskan pada Tabel 6.6 dan **Error! Reference source not found.**

**Tabel 6.6 Responden Pengujian *Usability* Pelanggan Restoran**

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

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

**Tabel 6.7 Task Completion Rate**

	<i>User 1</i>	<i>User 2</i>	<i>User 3</i>	<i>User 4</i>	<i>User 5</i>	<i>User 6</i>	<i>Completion Rate</i>

<i>Task 1</i>	√	√	√	√	√	√	100%
<i>Task 2</i>	√	√	√	√	√	√	100%
<i>Task 3</i>	√	√	√	√	√	√	100%
<i>Task 4</i>	√	√	√	√	√	√	100%
<i>Task 5</i>	√	√	√	√	√	√	100%
<i>Task 6</i>	√	√	√	√	√	√	100%
<i>Task 7</i>	√	√	√	√	√	√	100%

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

Setelah responden menyelesaikan *task scenario* yang diberikan, selanjutnya responden diminta untuk mengisi kuesioner SUPR-Qm yang telah didefinisikan sebelumnya. Hasil dari pengujian kuesioner SUPR-Qm untuk pelanggan restoran dapat dilihat pada Tabel 6.8.

**Tabel 6.8 Hasil Pengujian Kuesioner SUPR-Qm untuk Pelanggan Restoran**

No	Pertanyaan	Skor					Total Skor
		1	2	3	4	5	
1	Aplikasi ini penting untuk saya	0	0	0	5	0	20
2	Aplikasi ini merupakan aplikasi pemesanan menu restoran terbaik yang pernah saya gunakan	0	0	2	3	0	18
3	Saya tidak tahu apakah adalah aplikasi pemesanan menu restoran yang lebih baik dari aplikasi ini	0	0	1	3	1	20
4	Saya tidak akan menghapus aplikasi ini dari smartphone saya	0	0	1	2	2	21
5	Saya akan menyarankan aplikasi ini ke teman saya	0	0	0	4	1	21
6	Saya suka mengeksplorasi fitur-fitur yang terdapat dalam aplikasi ini	0	0	0	3	2	22
7	Aplikasi ini memiliki seluruh fitur dan fungsi yang saya inginkan pada aplikasi pemesanan menu restoran	0	1	0	3	1	19
8	Saya akan sering membuka aplikasi ini setiap kali saya ingin memesan	0	0	2	1	2	20

	menu di restoran Wong Solo						
9	Aplikasi ini menyenangkan	0	0	2	0	3	21
10	Aplikasi ini bekerja dengan baik dengan fitur-fitur lain yang ada pada <i>smartphone</i> saya (contohnya kamera)	0	0	1	2	2	21
11	Saya akan menggunakan aplikasi ini ketika ingin memesan menu di restoran Wong Solo	0	0	1	3	1	20
12	Desain dari aplikasi ini memudahkan saya dalam menemukan informasi yang saya inginkan	0	0	0	1	4	24
13	Menurut saya aplikasi ini menarik	0	0	0	4	1	21
14	Aplikasi ini sesuai dengan kebutuhan saya	0	0	1	3	1	20
15	Melakukan navigasi dalam aplikasi ini mudah bagi saya	0	0	2	2	1	19
16	Aplikasi ini mudah digunakan	0	0	0	3	2	22
Total Akhir							329
Total Nilai Maksimum							400
Nilai SUPR-Qm							82,25%

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

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

### 6.1.2 Analisis Pengujian Usability Aplikasi untuk Operator Restoran

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

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

**Tabel 6.9 Task Completion Rate**

	<i>User 1</i>	<i>Completion Rate</i>
<i>Task 1</i>	√	100%
<i>Task 2</i>	√	100%
<i>Task 3</i>	√	100%
<i>Task 4</i>	√	100%
<i>Task 5</i>	√	100%
<i>Task 6</i>	√	100%
<i>Task 7</i>	√	100%

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

Setelah responden menyelesaikan *task scenario* yang diberikan, selanjutnya responden diminta untuk mengisi kuesioner SUPR-Qm yang telah didefinisikan sebelumnya. Hasil dari pengujian kuesioner SUPR-Qm untuk pelanggan restoran dapat dilihat pada Tabel 6.10.

**Tabel 6.10 Hasil Pengujian Kuesioner SUPR-Qm untuk Operator Restoran**

No	Pertanyaan	Skor					Total Skor
		1	2	3	4	5	
1	Aplikasi ini penting untuk saya	0	0	1	0	0	3
2	Aplikasi ini merupakan aplikasi manajemen pemesanan menu restoran terbaik yang pernah saya gunakan	0	0	0	0	1	5
3	Saya tidak tahu apakah adalah aplikasi manajemen pemesanan menu restoran yang lebih baik dari aplikasi ini	0	0	0	0	1	5
4	Saya tidak akan menghapus aplikasi	0	0	1	0	0	3

	ini dari smartphone saya						
5	Saya akan menyarankan aplikasi ini ke teman saya	0	0	0	1	0	4
6	Saya suka mengeksplorasi fitur-fitur yang terdapat dalam aplikasi ini	0	0	0	0	1	5
7	Aplikasi ini memiliki seluruh fitur dan fungsi yang saya inginkan pada aplikasi pemesanan menu restoran	0	0	0	0	1	5
8	Saya akan sering membuka aplikasi ini setiap kali saya ingin memesan menu di restoran Wong Solo	0	0	0	0	1	5
9	Aplikasi ini menyenangkan	0	0	0	0	1	5
10	Aplikasi ini bekerja dengan baik dengan fitur-fitur lain yang ada pada <i>smartphone</i> saya (contohnya kamera)	0	0	0	0	1	5
11	Saya akan menggunakan aplikasi ini untuk manajemen pesanan menu di restoran Wong Solo	0	0	0	0	1	5
12	Desain dari aplikasi ini memudahkan saya dalam menemukan informasi yang saya inginkan	0	0	0	0	1	5
13	Menurut saya aplikasi ini menarik	0	0	0	0	1	5
14	Aplikasi ini sesuai dengan kebutuhan saya	0	0	0	0	1	5
15	Melakukan navigasi dalam aplikasi ini mudah bagi saya	0	0	0	0	1	5
16	Aplikasi ini mudah digunakan	0	0	0	0	1	5
Total Akhir							77
Total Nilai Maksimum							80
Nilai SUPR-Qm							96,25%

Tabel 6.8 menunjukkan 13 pertanyaan mendapatkan skor tertinggi yaitu 5, sedangkan skor terendah ditunjukkan pada pertanyaan nomor dengan skor 3. Pada tabel juga menunjukkan total akhir yang bernilai 329, sehingga didapatkan nilai SUPR-Qm sebesar 96,25%.

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

## BAB 7 PENUTUP

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

### 7.1 Kesimpulan

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

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

### 7.2 Saran

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

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



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

## 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 Restaurant 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*, (2015), 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.

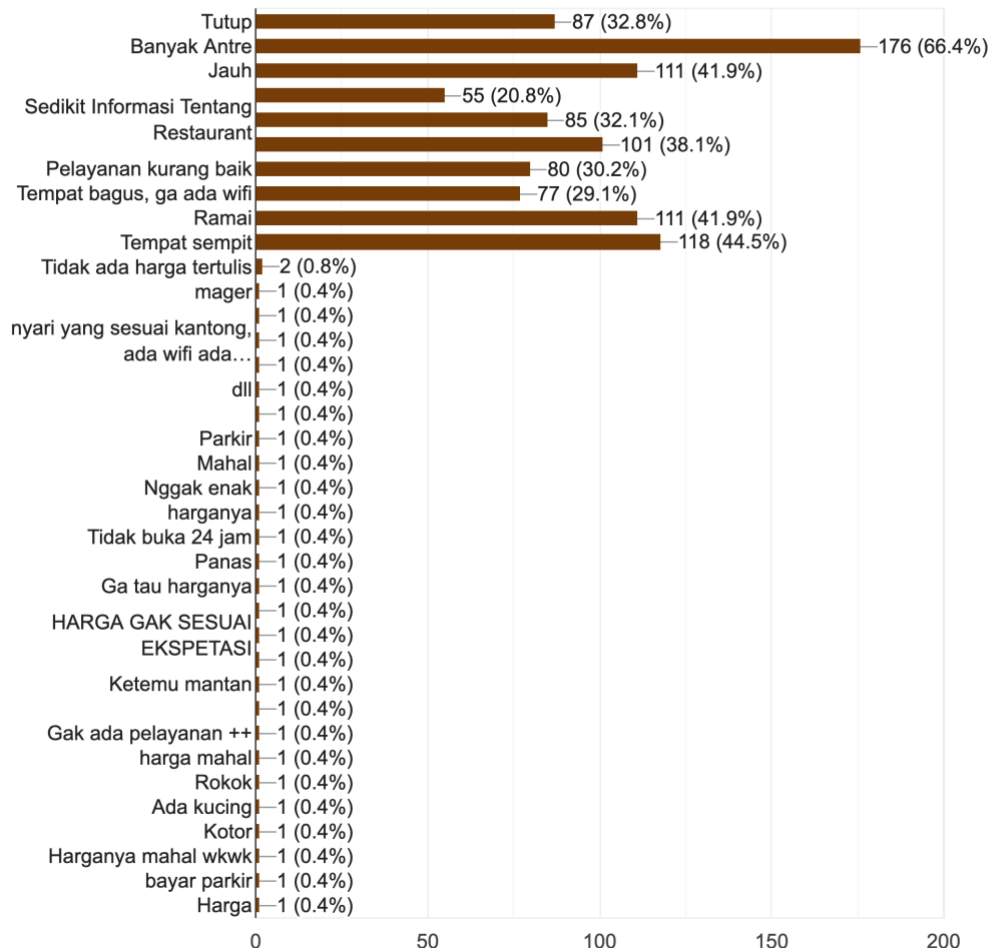
- Ozdenizci, B., Aydin, M. N., Coskun, V., & Ok, K. (n.d.). *Department of Information Technologies , ISIK University , Istanbul , Turkey { busraozdenizci , mnaydin , vedatcoskun , keremok }@ isikun . edu . tr.*
- Rasid, A., Supriyono, S., & Setiawan, R. (2018). Aplikasi Reservasi Menu Restoran Berbasis 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 (li). 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>

## LAMPIRAN A KUESIONER GOOGLE FORM

### A.1 Kuesioner menggunakan *Google Form* dengan pertanyaan kendala yang mungkin terjadi ketika datang ke restoran

Kendala apa saja yang mungkin terjadi ketika datang ke rumah makan?

265 responses



## LAMPIRAN B DEMONSTRASI PRODUK

### B.1 Demonstrasi Produk Sprint Pertama



### B.2 Demonstrasi Produk Sprint Kedua

