



SISTEM PEMESANAN MAKANAN DAN MINUMAN DI OSAKA RAMEN DEPOK BERBASIS JAVA

Tugas Akhir
diajukan untuk melengkapi
persyaratan mencapai
gelar sarjana

NAMA : ROBBY AWALDI
NPM : 201543501022

**PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK DAN ILMU KOMPUTER
UNIVERSITAS INDRAPRASTA PGRI
2019**

LEMBAR PERSETUJUAN UJIAN SKRIPSI/TUGAS AKHIR

Nama : Robby Awaldi
NPM : 201543501022
Fakultas : Teknik dan Ilmu Komputer
Program Studi : Informatika
Judul Skripsi/Tugas Akhir : Sistem Pemesanan Makanan dan Minuman Di
Osaka Ramen Depok Berbasis Java

Telah diperiksa dan disetujui untuk diujikan

Pembimbing Materi

Pembimbing Teknik

(Harry Dhika, M.Kom.)

(Meri Chrismes Aruan, S.Pd., M.Kom.)

LEMBAR PENGESAHAN

Nama : Robby Awaldi
NPM : 201543501022
Program Studi : Informatika
Fakultas : Teknik dan Ilmu Komputer
Judul : Sistem Pemesanan Makanan dan Minuman Di Osaka
Ramen Depok Berbasis Java

Panitia Ujian

Ketua : Prof. Dr. H. Sumaryoto _____
Sekretaris : Ir. H. Soepardi Harris, M.T _____
Anggota :

No.	Nama	Tanda Tangan
1.	Ni Wayan Parwati Septiani, M.M., M.Kom	
2.	Achmad Sarwandianto, M.Kom	
3.	Abdul Mufti, M.Kom	

LEMBAR PERNYATAAN

Yang bertanda tangan di bawah ini:

Nama : Robby Awaldi

NPM : 201543501022

Program Studi : Informatika

Dengan ini menyatakan bahwa skripsi/tugas akhir dengan judul Sistem Pemesanan Makanan dan Minuman Di Osaka Ramen Depok Berbasis Java beserta seluruh isinya adalah benar-benar karya saya sendiri. Saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika ilmu yang berlaku dalam masyarakat keilmuan. Atas pernyataan ini, saya siap menanggung risiko/sanksi apabila di kemudian hari ditemukan adanya pelanggaran etika keilmuan atau ada klaim dari pihak lain terhadap keaslian karya saya ini sesuai dengan Undang-undang Republik Indonesia Nomor 20 Tahun 2003 tentang Sistem Pendidikan Nasional Bab VI Pasal 25 ayat 2 dan Bab XX Pasal 70. Demikian pernyataan ini saya buat untuk dimanfaatkan sesuai dengan keperluan.

Jakarta,

Yang menyatakan,

Robby Awaldi

ABSTRAK

- A. Robby Awaldi, NPM : 201543501022
- B. **Sistem Pemesanan Makanan dan Minuman Di Osaka Ramen Depok Berbasis Java**. Skripsi/Tugas Akhir : Jakarta : Fakultas Teknik dan Ilmu Komputer : Program Studi Informatika : Universitas Indraprasta Persatuan Guru Republik Indonesia, Juli, 2019
- C. xv + 5 Bab + 91 halaman
- D. Kata Kunci : Sistem, Pemesanan, Java
- E. Tujuan penelitian adalah untuk memberikan solusi kepada tempat penelitian terkait untuk menerapkan media digital dalam proses pemesanan makanan dan minuman. Metode pengembangan sistem yang digunakan adalah metode *waterfall*. Sedangkan metode pengumpulan data yang digunakan yaitu observasi, studi literatur, dan wawancara. Hasil yang diperoleh dari penelitian ini adalah sistem dapat membantu restoran Osaka Ramen dalam melayani pemesanan dengan cepat, akurat, dan efisien terutama saat pengunjung sedang ramai sehingga pekerjaan menjadi lebih mudah.
- F. Daftar Pustaka: 1. Buku 18 buah (Tahun 2010 – 2015)
2. 2 Jurnal
- G. Pembimbing: (**Harry Dhika, M.Kom.**) Pembimbing Materi
(**Meri Chrismes Aruan, S.Pd., M.Kom.**) Pembimbing Teknik

“Allah mencintai pekerjaan yang apabila bekerja ia menyelesaikannya dengan baik”

(HR. Thabrani)

“Skripsi ini
Penulis persembahkan
kepada Mama, Ayah, dan Ade”

KATA PENGANTAR

Penulis memanjatkan puji syukur ke hadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya kepada penulis, sehingga akhirnya penulis dapat menyelesaikan skripsi/tugas akhir ini tepat pada waktunya.

Skripsi/tugas akhir yang berjudul “Sistem Pemesanan Makanan dan Minuman Di Osaka Ramen Depok Berbasis Java” ini ditulis untuk memenuhi salah satu syarat guna memperoleh gelar sarjana pada Universitas Indraprasta PGRI. Pada kesempatan yang baik ini, izinkanlah penulis menyampaikan rasa hormat dan ucapan terima kasih kepada semua pihak yang dengan tulus ikhlas telah memberikan bantuan dan dorongan kepada penulis dalam menyelesaikan skripsi/tugas akhir ini, terutama kepada:

1. Bapak Harry Dhika, M.Kom. selaku Dosen Pembimbing Materi Universitas Indraprasta PGRI.
2. Ibu Meri Chrismes Aruan, S.Pd., M.Kom. selaku Dosen Pembimbing Teknik Universitas Indraprasta PGRI.
3. Bapak Taufik Hidayat selaku pemilik Osaka Ramen Depok dan seluruh pegawai Osaka Ramen Depok yang telah memberikan kesempatan kepada penulis untuk melakukan penelitian serta membantu dalam proses pembuatan tugas akhir.
4. Bapak Prof. Dr. H. Sumaryoto selaku Rektor Universitas Indraprasta PGRI.
5. Ibu Mei Lestari, M.Kom. selaku Ketua Program Studi Informatika.
6. Seluruh Dosen dan Staff Informatika Universitas Indraprasta PGRI.

7. Kedua orang tua penulis yang senantiasa menyayangi, mendoakan, serta memberikan dukungan moral dan moril.
8. Seluruh kerabat dan kawan seperjuangan yang telah membantu dan memberikan dukungan yang luar biasa khususnya Nadia Rizky, Aditya Maulana Kahfi, Surianto, Syamsir Achmad Hidayat, Erlangga Ario Tejo, Duhan Ferdiansyah, Toharudin, Mochamad Rizki Apriyana.

Penulis menyadari bahwa skripsi/tugas akhir ini masih banyak kekurangan, baik bentuk, isi, maupun teknik penyajiannya. Oleh sebab itu, kritik dan saran yang bersifat membangun dari berbagai pihak akan penulis terima dengan tangan terbuka serta sangat diharapkan. Semoga kehadiran skripsi/tugas akhir ini memenuhi sasarannya.

Jakarta, Juli 2019

Penulis

DAFTAR ISI

LEMBAR PERSETUJUAN	i
LEMBAR PENGESAHAN	ii
LEMBAR PERNYATAAN	iii
ABSTRAK	iv
LEMBAR MOTO	v
KATA PENGANTAR.....	vi
DAFTAR ISI.....	viii
DAFTAR TABEL	x
DAFTAR GAMBAR.....	xi
DAFTAR SIMBOL	xiii
DAFTAR LAMPIRAN	xv
BAB I PENDAHULUAN.....	1
A. Latar Belakang Masalah	1
B. Identifikasi Masalah	2
C. Batasan Masalah.....	2
D. Rumusan Masalah	3
E. Tujuan Penelitian.....	3
F. Kegunaan Penelitian.....	4
G. Sistematika Penulisan.....	4
BAB II LANDASAN TEORI, PENELITIAN YANG RELEVAN DAN KERANGKA BERPIKIR	7
A. Landasan Teori	7
B. Penelitian Yang Relevan	15
C. Kerangka Berpikir	19
BAB III METODE PENELITIAN	21
A. Waktu dan Tempat Penelitian	21
B. Desain Penelitian	22
C. Metode Pengumpulan Data	23
D. Langkah - Langkah Pengembangan Sistem	24

BAB IV ANALISIS SISTEM BERJALAN DAN RANCANGAN SISTEM YANG DIUSULKAN.....	26
A. Profil Perusahaan.....	26
B. Struktur Organisasi Perusahaan	27
C. Proses Bisnis Sistem Berjalan	28
D. Aturan Bisnis Sistem Berjalan	29
E. Dekomposisi Fungsi Sistem	30
F. Analisis Masukan (<i>Input</i>), Proses dan Keluaran (<i>Output</i>) Sistem Berjalan ..	30
G. Diagram Alir Data (DAD) Sistem Berjalan (Diagram Konteks, Nol, Rinci)	32
H. Analisis Permasalahan.....	33
I. Alternatif Penyelesaian Masalah	34
J. Aturan Bisnis Sistem Diusulkan	35
K. Dekomposisi Fungsi Sistem Diusulkan.....	36
L. Rancangan Masukan, Proses, dan Keluaran.....	36
M. Diagram Alir Data (DAD) Sistem yang Diusulkan (Diagram Konteks, Nol, Rinci)	41
N. Kamus Data Sistem yang Diusulkan	44
O. Spesifikasi Proses Sistem yang Diusulkan	48
P. Bagan Terstruktur Sistem yang Diusulkan.....	54
Q. Spesifikasi Modul yang Diusulkan	61
R. Rancangan Basis Data Sistem yang Diusulkan.....	62
S. Rancangan Layar, Rancangan <i>Form</i> Masukan Data, dan Rancangan Keluaran	67
T. Tampilan dan Penjelasan Layar, Tampilan Format Masukan, dan Tampilan Keluaran	78
BAB V SIMPULAN DAN SARAN.....	90
A. Simpulan.....	90
B. Saran.....	91
DAFTAR PUSTAKA	
LAMPIRAN	

DAFTAR TABEL

Tabel 2.1 Folder Penting XAMPP	12
Tabel 2.2 Penelitian Yang Relevan	16
Tabel 3.1 Jadwal Penelitian.....	21
Tabel 4.1 Spesifikasi File Menu.....	65
Tabel 4.2 Spesifikasi File Detail Ramen.....	66
Tabel 4.3 Spesifikasi File Transaksi	66
Tabel 4.4 Spesifikasi File Pesanan.....	67
Tabel 4.5 Spesifikasi File Level.....	67

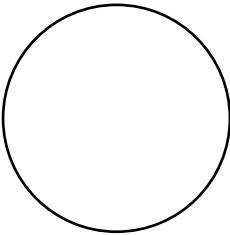

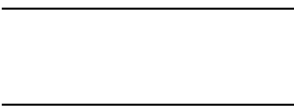

DAFTAR GAMBAR

Gambar 2.1 IntelliJ IDEA Community 2019.1.1	11
Gambar 2.2 Kerangka Berpikir	19
Gambar 3.1 Model Waterfall	24
Gambar 4.1 Struktur Organisasi.....	27
Gambar 4.2 Dekomposisi Fungsi Sistem	30
Gambar 4.3 Diagram Konteks Sistem Berjalan	32
Gambar 4.4 Diagram Nol Sistem Berjalan	33
Gambar 4.5 Dekomposisi Fungsi Sistem Diusulkan	36
Gambar 4.6 Diagram Konteks Sistem yang Diusulkan	41
Gambar 4.7 Diagram Nol Sistem yang Diusulkan.....	42
Gambar 4.8 Diagram Rinci Level 1 Proses 1 Sistem yang Diusulkan.....	42
Gambar 4.9 Diagram Rinci Level 1 Proses 2.....	43
Gambar 4.10 Diagram Rinci Level 1 Proses 3.....	43
Gambar 4.11 Diagram Rinci Level 1 Proses 4.....	44
Gambar 4.12 Bagan Terstruktur Menampilkan Katalog.....	54
Gambar 4.13 Bagan Terstruktur Mengirim Pesanan.....	54
Gambar 4.14 Bagan Terstruktur Menerima Konfirmasi Pesanan.....	55
Gambar 4.15 Bagan Terstruktur Request Pembayaran	55
Gambar 4.16 Bagan Terstruktur Mencetak Bill	56
Gambar 4.17 Bagan Terstruktur Melakukan Pembayaran	56
Gambar 4.18 Bagan Terstruktur Menyimpan Transaksi.....	57
Gambar 4.19 Bagan Terstruktur Mencetak Laporan Pemesanan.....	57
Gambar 4.20 Bagan Terstruktur Mencetak Laporan Pemasukan	58
Gambar 4.21 Bagan Terstruktur Mencetak Laporan Menu Favorit.....	58
Gambar 4.22 Bagan Terstruktur Mencetak Laporan Kunjungan.....	59
Gambar 4.23 Bagan Terstruktur Menampilkan Daftar Menu	59
Gambar 4.24 Bagan Terstruktur Tambah Menu	60
Gambar 4.25 Bagan Terstruktur Hapus Menu	60
Gambar 4.26 Bagan Terstruktur Ubah Menu.....	61
Gambar 4.27 Bentuk Tidak Normal.....	62
Gambar 4.28 Normalisasi Pertama	63
Gambar 4.29 Normalisasi Kedua	63
Gambar 4.30 Normalisasi Ketiga.....	64
Gambar 4.31 Diagram ERD.....	64
Gambar 4.32 Rancangan Antarmuka	68
Gambar 4.33 Rancangan Tampilan Navigasi.....	68
Gambar 4.34 Rancangan Tampilan Menu Ramen	69

Gambar 4.35 Rancangan Tampilan Menu Minuman, Cemilan, dan Lainnya	70
Gambar 4.36 Rancangan Tampilan Daftar Pesanan	71
Gambar 4.37 Rancangan Tampilan Setting	72
Gambar 4.38 Rancangan Tampilan Sign in	73
Gambar 4.39 Rancangan Tampilan Side Bar	74
Gambar 4.40 Rancangan Tampilan Halaman Utama.....	75
Gambar 4.41 Rancangan Tampilan Daftar Menu	76
Gambar 4.42 Rancangan Tampilan Laporan	77
Gambar 4.43 Tampilan Navigasi	78
Gambar 4.44 Tampilan Daftar Ramen	78
Gambar 4.45 Tampilan Daftar Minuman.....	79
Gambar 4.46 Tampilan Daftar Cemilan.....	79
Gambar 4.47 Tampilan Daftar Lainnya	80
Gambar 4.48 Tampilan Daftar Pesanan	81
Gambar 4.49 Tampilan Setting	82
Gambar 4.50 Tampilan Sign in	83
Gambar 4.51 Tampilan Side Bar.....	84
Gambar 4.52 Tampilan Halaman Utama	84
Gambar 4.53 Tampilan Daftar Menu	85
Gambar 4.54 Tampilan Laporan	86
Gambar 4.55 Tampilan Laporan Pemesanan	87
Gambar 4.56 Tampilan Laporan Menu Favorit	87
Gambar 4.57 Tampilan Laporan Pemasukan	88
Gambar 4.58 Tampilan Laporan Kunjungan	89

DAFTAR SIMBOL

A. Simbol Diagram Alir Data




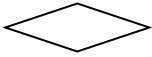
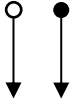
No.	Simbol <i>Data Flow Diagram</i>	Keterangan
1.		Proses , adalah suatu langkah-langkah yang dilakukan komputer untuk mengelola data dari masukan sehingga mengeluarkan suatu keluaran data atau informasi.
2.		Entitas , adalah suatu keberadaan yang memiliki keunikan dan beda dari yang lain. Entitas mengirim data menuju proses dan menerima hasil dari proses.
3.		Simpanan Data , merupakan basis data untuk menyimpan data hasil proses sistem sehingga data tersebut dapat digunakan kembali atau diambil kembali.
4.		Alur Data , menunjukkan arah data yang dikirim dari suatu entitas ke proses atau sebaliknya dan dari proses ke basis data atau sebaliknya.

B. Simbol Kamus Data

No.	Simbol	Arti
1.	=	Disusun atau terdiri atas
2.	+	Dan

3.	[]	baik ...atau...
4.	{ } ⁿ	n kali diulang/bernilai banyak
5.	()	Data operasional
6.	*...*	Batas komentar

C. Simbol Bagan Terstruktur

No	Simbol	Keterangan
1.		Module Menunjukkan suatu modul.
2.		Connection Untuk menghubungkan suatu modul dengan modul yang lainnya.
3.		Loop Menunjukkan suatu perulangan di dalam modul.
4.		Decision Menunjukkan suatu penyeleksian kondisi di dalam modul.
5.		Couple Menunjukkan suatu data atau elemen kontrol yang dikirim dari suatu modul ke modul lainnya. Panah dengan lingkaran kosong menunjukkan data dikirim dan panah dengan lingkaran diblok menunjukkan elemen kontrol yang dikirim.

DAFTAR LAMPIRAN

Lampiran 1	Daftar Riwayat Hidup Penulis
Lampiran 2	Surat Keterangan Mitra
Lampiran 3	Kartu Asistensi 1
Lampiran 4	Kartu Asistensi 2
Lampiran 5	Listing Program

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Pada saat ini perkembangan teknologi informasi berkembang dengan pesat dan banyak dimanfaatkan untuk memenuhi aktivitas atau pekerjaan manusia. Banyak pekerjaan manusia yang sebelumnya menggunakan sistem manual dapat diubah menjadi sistem yang terkomputerisasi. Meskipun teknologi informasi sudah berkembang pesat, tetapi masih saja terdapat sistem manual yang masih dipertahankan. Salah satu sistem manual yang masih dipertahankan sampai saat ini adalah sistem pemesanan makanan menggunakan cara konvensional.

Sistem pemesanan bersifat konvensional mengandalkan pelayan untuk melayani pemesanan setiap pelanggan dengan mencatat pesanan menggunakan alat tulis dan kertas. Sistem pemesanan yang bersifat konvensional ini sering kali terdapat permasalahan yang ditimbulkan, salah satunya adalah saat tempat makan tersebut sedang ramai, pelayan akan sangat sibuk melayani pelanggan, dan pelanggan yang baru datang harus menunggu pelayan agar dapat melakukan pemesanan. Masalah ini terjadi di Osaka Ramen Depok yang masih menggunakan sistem pemesanan bersifat konvensional.

Osaka Ramen Depok adalah resto yang menjual bermacam-macam jenis ramen khas Jepang dan juga terdapat berbagai masakan Indonesia. Seperti yang dijelaskan di atas, Osaka Ramen Depok masih menggunakan sistem

pemesanan yang bersifat konvensional. Oleh karena itu, perlu dibuat “**Sistem Pemesanan Makanan Dan Minuman Di Osaka Ramen Depok Berbasis Java**”. Sistem pemesanan yang dibuat diharapkan dapat membantu mempercepat proses pemesanan di Osaka Ramen Depok.

B. Identifikasi Masalah

Berdasarkan latar belakang yang dijelaskan di atas dapat dilihat permasalahan yang terjadi di Osaka Ramen Depok, maka penulis akan menyimpulkan beberapa identifikasi masalah yang ada, yaitu:

1. Pencatatan pesanan terkadang mengalami kesalahan karena menggunakan kertas.
2. Isi laporan penjualan sering mengalami kesalahan dikarenakan masih menghitung manual.
3. Pelanggan yang ingin memesan sering terabaikan di saat resto sedang ramai.
4. Proses pembayaran masih terbilang lama dan berisiko terdapat kesalahan karena dihitung secara manual menggunakan kalkulator.

C. Batasan Masalah

Berdasarkan latar belakang dan permasalahan yang diidentifikasi di atas, maka penulis melakukan pembatasan masalah, yaitu:

1. Sistem pemesanan yang dibuat hanya untuk mencatat pemesanan makanan dan minuman pelanggan yang ada di resto.
2. Laporan yang dibuat adalah laporan transaksi pemesanan setiap hari.
3. Fitur yang dapat digunakan pelanggan yaitu menampilkan daftar menu, melakukan pemesanan menu, dan melakukan pembayaran.

4. Pembayaran yang dapat dilakukan hanya melalui pembayaran tunai/*cash*.

D. Rumusan Masalah

Berdasarkan pembatasan masalah, maka perumusan masalah dalam penelitian ini yaitu:

1. Bagaimana caranya membuat sistem pemesanan makanan dan minuman yang terkomputerisasi?
2. Bagaimana cara mengelola data transaksi menjadi laporan yang bermanfaat dan akurat?
3. Bagaimana cara membuat *user interface* yang mudah digunakan dan tidak membingungkan pelanggan?
4. Seperti apakah proses pembayaran apabila menggunakan sistem yang terkomputerisasi?

E. Tujuan Penelitian

Tujuan dari penelitian tugas akhir ini adalah untuk menghasilkan sebagai berikut:

1. Membantu proses pencatatan makanan dan minuman yang selama ini dalam bentuk manual ke dalam komputerisasi.
2. Dapat membuat laporan yang lebih akurat untuk setiap pembukuan.
3. Dapat mempermudah pelanggan untuk memesan makanan dan minuman.
4. Dapat mempercepat proses pembayaran dan menghindari kesalahan saat menghitung total pembayaran.

F. Kegunaan Penelitian

Adapun kegunaan dari penelitian ini dibagi menjadi beberapa aspek, yaitu sebagai berikut:

1. Aspek Sistem

- a. Untuk memudahkan proses pemesanan yang sebelumnya masih menggunakan manual menjadi lebih cepat dan efisien.
- b. Sistem tersebut membuat proses pemesanan lebih mudah tanpa harus memanggil pelayan.

2. Aspek Manajerial

- a. Lebih efisien kinerja karyawan karena tidak ada yang merangkap tugas kerja.
- b. Lebih terkoordinasi dalam sistem pelayanan di Osaka Ramen Depok.

3. Aspek Penelitian Lanjutan

- a. Membuat sistem yang terhubung dengan cabang-cabang Osaka Ramen lainnya.
- b. Membuat sistem yang tidak hanya bekerja di jaringan lokal tetapi dapat diakses melalui *online*.

G. Sistematika Penulisan

Dalam usaha pemberian gambaran secara singkat, isi skripsi ini akan terbagi menjadi lima bab yang selanjutnya akan dijabarkan dalam beberapa sub-bab. Sistematika pembahasan adalah sebagai berikut:

BAB I PENDAHULUAN

Dalam bab ini penulis memberikan gambaran awal tentang latar belakang, identifikasi masalah, batasan masalah, rumusan masalah, tujuan penelitian, kegunaan penelitian, dan sistematika penulisan.

**BAB II LANDASAN TEORI, PENELITIAN YANG
RELEVAN DAN KERANGKA BERPIKIR**

Pada bab ini penulis menguraikan teori-teori atas konsep-konsep yang melandasi pembahasan dalam tugas akhir. Penelitian yang relevan berisi deskripsi singkat penelitian lain yang masih berkaitan dengan penelitian ini. Sedangkan kerangka berpikir menggambarkan bagan pola pikir penelitian dari awal sampai akhir kegiatan.

BAB III METODE PENELITIAN

Dalam bab ini menjelaskan tentang waktu dan tempat penelitian, desain penelitian, serta metode pengumpulan data dan langkah-langkah pengembangan sistem.

**BAB IV ANALISIS SISTEM BERJALAN DAN RANCANGAN
SISTEM YANG DIUSULKAN**

Pada bab ini penulis membahas mengenai profil perusahaan, struktur organisasi perusahaan, analisis sistem berjalan, analisis permasalahan, rancangan sistem yang diusulkan dan rancangan layar beserta penjelasannya.

BAB V SIMPULAN DAN SARAN

Pada bab ini berisi mengenai simpulan dan saran yang berkaitan dengan sistem pemesanan makanan dan minuman di Osaka Ramen Depok.

BAB II

LANDASAN TEORI, PENELITIAN YANG RELEVAN DAN KERANGKA BERPIKIR

A. Landasan Teori

Sebagai landasan pembahasan pada bab berikutnya diperlukan beberapa teori yang mendukung di antaranya:

1. Sistem

“Sistem adalah kumpulan dari elemen yang saling berhubungan dan bekerja sama untuk mencapai suatu tujuan” (Sujarweni, 2015). “Sistem adalah kumpulan dari sub sistem atau komponen apa pun baik berupa fisik yang berhubungan satu sama lain dan bekerja sama secara sistematis untuk mencapai satu tujuan tertentu” (Susanto, 2013). “Sistem sebagai kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisir, saling berinteraksi satu sama lain, dan terpadu” (Sutabri, 2012).

2. Pengertian Pemesanan

“Pemesanan adalah suatu aktivitas yang dilakukan oleh konsumen sebelum membeli” (Rahman, 2015). Pemesanan memiliki arti yang luas tergantung objek apa yang dituju. Dalam penelitian ini pengertian pemesanan adalah suatu proses yang dilakukan pelanggan restoran untuk memesan makanan dan minuman yang tersedia di daftar menu.

3. Java

“Bahasa pemrograman Java merupakan salah satu dari sekian banyak bahasa pemrograman yang dapat dijalankan di berbagai sistem operasi termasuk telepon genggam” (Nofriadi, 2015).

“Java merupakan bahasa pemrograman berorientasi objek yang dikembangkan oleh Sun Microsystem yang dimulai oleh James Gosling dan dirilis pada tahun 1995, saat ini Sun Microsystem telah di akuisisi oleh Oracle Corporation” (Enterprise, 2015).

“Bahasa Java memberi harapan menjadi perekat universal yang menghubungkan pemakai dengan informasi dari *web server*, basis data, penyedia informasi dan sumber-sumber lain” (Hariyanto, 2014).

Java dapat digunakan di berbagai *platform* dikarenakan program Java berjalan menggunakan *Java virtual machine (JVM)*. JVM inilah yang membuat program Java hanya perlu ditulis sekali dan dapat digunakan di berbagai sistem operasi dan *platform*. Java termasuk bahasa pemrograman berorientasi objek dikarenakan dalam bahasa Java semua sintaksis atau kode harus berada di dalam suatu *class* yang nantinya akan diimplementasikan menjadi sebuah objek.

4. FXML

FXML adalah sebuah bahasa *markup* berbasis XML yang digunakan untuk membuat tampilan atau antarmuka pada aplikasi berbasis JavaFX. FXML dapat dihasilkan secara otomatis menggunakan aplikasi *scene builder* sehingga dapat mempermudah dalam merancang tampilan

antarmuka. Penggunaan FXML juga dimaksud untuk memisahkan kode rancangan tampilan dengan kode logika bisnis yang bertujuan membuat kode program lebih rapih dan mudah untuk dilakukan pengembangan lebih lanjut.

5. MySQL

“MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 (enam) juta instalasi di seluruh dunia” (Faizal & Irnawati, 2015).

“MySQL adalah perangkat lunak basis data. MySQL merupakan tipe data relasional yang artinya MySQL menyimpan datanya dalam bentuk tabel-tabel yang saling berhubungan” (Winarno, Zaki, & SmitDev Community, 2014).

“MySQL adalah sistem yang berguna untuk melakukan proses pengaturan koleksi-koleksi struktur data (*database*) baik yang meliputi proses pembuatan atau proses pengelolaan *database*” (Ahmar, 2013).

6. Jaringan Komputer

“Jaringan komputer adalah suatu himpunan interkoneksi sejumlah komputer. Dalam bahasa populer dapat dijelaskan bahwa jaringan komputer adalah kumpulan beberapa komputer, dan perangkat lain seperti *router*, *switch*, dan sebagainya” (Sofana, 2013).

“Jaringan komputer merupakan sebuah sistem yang terdiri atas komputer dan perangkat jaringan lainnya yang bekerja sama untuk mencapai suatu tujuan yang sama” (Andi, 2015).

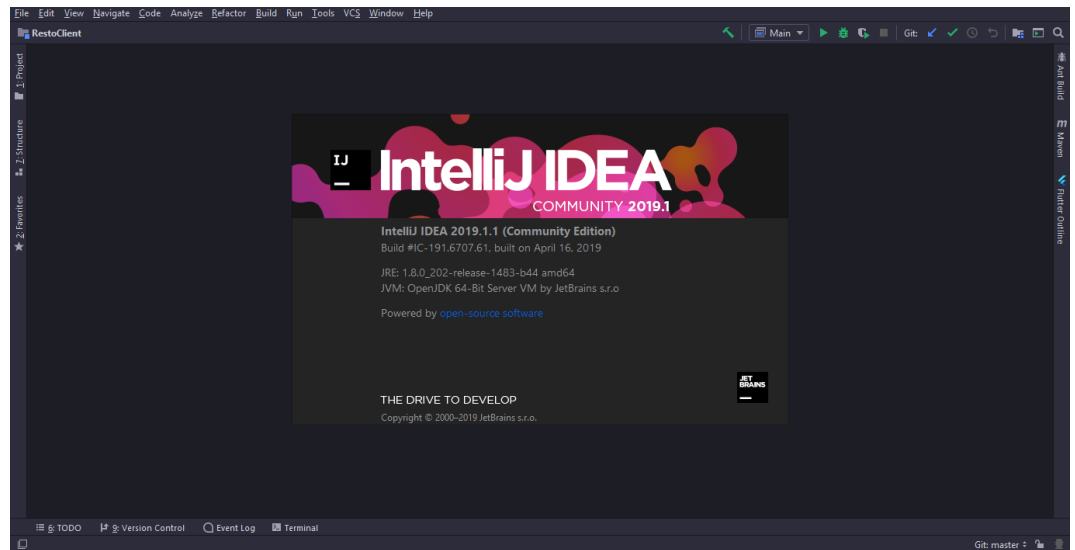
Berdasarkan penjelasan yang dikemukakan para ahli di atas dapat disimpulkan bahwa jaringan komputer adalah suatu sistem yang berupa sejumlah komputer dan perangkat jaringan lain yang saling terkoneksi satu sama lain, saling mengirim informasi dan berkomunikasi sehingga dapat mencapai suatu tujuan yang sama.

7. JSON

“JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer” (Juansyah Andi, 2015). Pertukaran data dengan menggunakan format JSON sangat ideal karena format JSON berbasis teks yang independen tetapi menggunakan konvensi yang akrab bagi programmer. JSON dibangun di atas dua struktur, yaitu:

- a. Koleksi pasangan nama atau nilai. Dalam berbagai bahasa pemrograman, ini direalisasikan sebagai objek.
- b. Daftar dari nilai. Dalam kebanyakan bahasa pemrograman, ini direalisasikan sebagai *array*.

8. IntelliJ IDEA



Gambar 2.1
IntelliJ IDEA Community 2019.1.1
Sumber : Dokumen Pribadi

IntelliJ IDEA adalah IDE (*Integrated Development Environment*) yang digunakan untuk mengembangkan program atau aplikasi menggunakan bahasa pemrograman Java. IntelliJ IDEA adalah perangkat lunak yang mirip seperti Netbeans, akan tetapi IntelliJ IDEA memiliki beberapa kelebihan seperti memberikan saran yang lebih lengkap saat proses penulisan kode berlangsung.

9. XAMPP

“XAMPP adalah perangkat lunak *open source* yang diunggah secara gratis dan bisa dijalankan di semua sistem operasi seperti Windows, Linux, Solaris, dan Mac” (Buana, 2014).

Di dalam folder utama XAMPP terdapat beberapa folder penting yang perlu diketahui yaitu sebagai berikut:

Tabel 2.1
Folder Penting XAMPP

Folder	Keterangan
<i>Apache</i>	Folder utama dari <i>Apache Webserver</i>
<i>Htdocs</i>	Folder utama untuk menyimpan data-data latihan web, baik <i>PHP</i> maupun <i>HTML</i> biasa
<i>Manual</i>	Berisi <i>sub folder</i> yang di dalamnya terdapat manual program dan <i>database</i> , termasuk manual <i>PHP</i> dan <i>MySQL</i>
<i>MySQL</i>	Folder utama untuk <i>database MySQL Server</i>
<i>PHP</i>	Folder utama untuk program <i>PHP</i>

Sumber : (Nugroho, 2014)

10. ERD

“ERD adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam *system* secara abstrak. ERD mendokumentasikan data dengan mengidentifikasi jenis entitas dan hubungannya” (Ladjamudin, 2013). ERD terdiri dari 3(tiga) tingkatan yaitu:

a. Entitas (*entity*)

Entitas adalah suatu objek yang memiliki ciri khas yang membedakan dari objek-objek lainnya. Entitas dapat berupa apa saja dan tidak harus berbentuk fisik, dapat juga berupa sesuatu yang abstrak.

b. Identifikasi (*attribute*)

Identifikasi adalah penjelasan dari suatu entitas sehingga entitas memiliki suatu karakteristik tertentu. Suatu entitas diidentifikasi

menjadi sekumpulan atribut yang menjelaskan detail dari entitas tersebut.

c. Hubungan (*relation*)

Relasi adalah hubungan antara satu entitas dengan entitas lain. Relasi menjelaskan bagaimana satu entitas memiliki peran dengan entitas lain sehingga setiap entitas saling terhubung. Hubungan antar dua entitas dapat dikategorikan menjadi tiga macam yaitu:

- 1) Hubungan satu dengan satu (1:1), yaitu satu entitas hanya memiliki satu hubungan saja dengan entitas lain dan begitu sebaliknya.
- 2) Hubungan satu dengan banyak (1:M) atau (M:1), yaitu satu entitas hanya memiliki satu hubungan dengan entitas lain tetapi entitas lain memiliki banyak hubungan.
- 3) Hubungan banyak dengan banyak (M:M), yaitu satu entitas memiliki banyak hubungan dengan entitas lain, begitu juga sebaliknya.

11. Normalisasi

“Normalisasi adalah proses untuk mengevaluasi dan memperbaiki struktur tabel untuk meminimalkan kesamaan data, sehingga mengurangi kemungkinan anomali data” (Coronel, Morris, & Rob, 2013).

Untuk melakukan normalisasi ada beberapa tahapan yang harus dilakukan, yaitu:

a. Bentuk Normal Pertama (*1NF/First Normal Form*)

Bentuk normal pertama dimulai dengan menyajikan data dalam format *tabular*, di mana setiap sel memiliki nilai tunggal dan tidak ada grup berulang. Untuk menghilangkan grup berulang, menghilangkan nol dengan memastikan bahwa setiap atribut grup berulang berisi nilai data yang sesuai

b. Bentuk Normal Kedua (*2NF/Second Normal Form*)

Konversi ke 2NF dilakukan hanya ketika 1NF memiliki kunci primer komposit. Jika 1NF memiliki atribut primer tunggal kunci, maka tabel secara otomatis dalam 2NF.

c. Bentuk Normal Ketiga (*3NF/Third Normal Form*)

Untuk setiap ketergantungan transitif, tulis determinannya sebagai kunci primer untuk tabel baru. Penentunya adalah atribut apa pun yang nilainya menentukan nilai lain dalam satu baris.

d. Bentuk Normal Boyce-Codd (*BCNF/Boyce Codd Normal Form*)

Setiap penentu dalam tabel adalah kunci kandidat. BCNF tidak boleh berisi lebih dari satu kunci kandidat.

e. Bentuk Normal Keempat (*4NF/Fourth Normal Form*)

Tidak ada ketergantungan bernilai banyak selain kunci kandidat.

f. Bentuk Normal Kelima (*5NF/Fifth Normal Form*)

Tidak kehilangan dekomposisi ke dalam tabel yang lebih kecil.

12. DFD

“*Data Flow Diagram* (DFD) merupakan alat untuk membuat diagram yang serbaguna” (Yakub, 2012). “*Data Flow Diagram* atau dalam Bahasa Indonesia menjadi Diagram Alir Data (DAD) adalah representasi grafik yang menggambarkan aliran informasi dan transformasi informasi yang diaplikasikan sebagai data yang mengatur dari masukan (*input*) dan keluaran (*output*)” (Sukamto & Shalahuddin, 2014).

13. Kamus Data (*Data Dictionary*)

Kamus data berfungsi sebagai tempat informasi suatu data yang mengenai definisi, struktur dan pemakaian dari sistem elemen. Elemen adalah unit data yang terkecil yang terdapat pada suatu sistem informasi. Kamus data berperan menyajikan suatu data yang ada pada sistem informasi tersebut.

“Kamus data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga masukan (*input*) dan keluaran (*output*) dapat dipahami secara umum (memiliki standar cara penulisan)” (Sukamto & Shalahuddin, 2014).

B. Penelitian Yang Relevan

Hasil penelitian yang bisa dijadikan acuan atau pembanding dalam kajian penelitian masalah sebagai berikut :

Tabel 2.2
Penelitian Yang Relevan

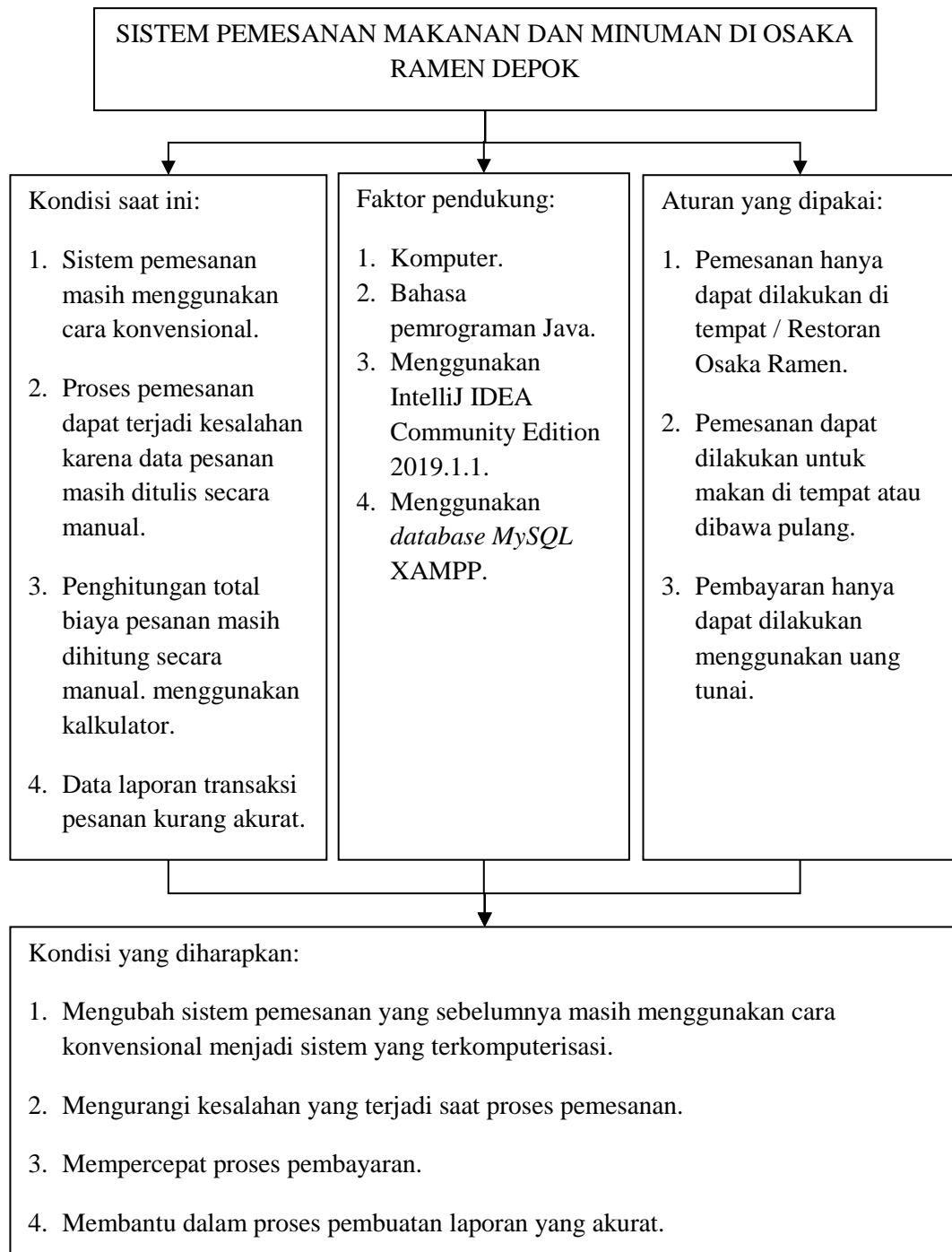
Identitas Penulis	Judul	Tujuan Penelitian	Hasil Penelitian
Reni Rosmitalia 11540088 (2016) di Universitas Islam Negeri Raden Fatah Palembang	Sistem Pemesanan Makanan di Rumah Makan Palapa Indah Berbasis <i>Web Service</i> Menggunakan Mobile Android	Membuat sistem pemesanan makanan berbasis <i>web service</i> dengan <i>platform</i> android pada Rumah Makan Palapa Indah	Setelah melakukan <i>requirement planning</i> , <i>user design</i> dan berakhir dengan pembuatan program yang sesungguhnya, maka hasil yang dicapai oleh penulisan adalah sebuah sistem pemesanan makanan di rumah makan palapa indah berbasis <i>web service</i> menggunakan <i>mobile</i> android dengan menggunakan bahasa pemrograman Java dan XML.
Abdul Haris 10510321 (2014) di Universitas Komputer	Sistem Informasi Pemesanan Makanan dan Minuman di	Mengetahui sistem informasi yang sedang berjalan di R.M Pondok Kapau, merancang	Mempercepat dalam pelayanan pemesanan makanan dan minuman karena pemesanan telah menggunakan aplikasi

Indonesia	R.M Pondok	sistem informasi	<i>mobile</i> android,
Bandung	Kapau Berbasis Android	pemesanan makanan dan minuman berbasis android di R.M Pondok Kapau agar dapat mendukung dan meningkatkan pelayanannya. Melakukan pengujian sistem informasi pemesanan makanan dan minuman berbasis android yang diusulkan di R.M Pondok Kapau.	mengurangi kesalahan dalam pemesanan makanan dan minuman karena telah tertata dengan baik pada aplikasi ini.
Dian Galih Tegar di Universitas	Sistem Informasi	Merancang aplikasi pemesanan	Aplikasi dapat mengatasi pemesanan
Dian Nuswantoro	Pemesanan makanan dan Minuman Pada Omahe Cafe and Resto	makanan dan minuman berbasis <i>client server</i> dengan <i>platform</i> <i>android</i> pada	makanan dan minuman dengan <i>platform</i> android berbasis <i>client server</i> . Aplikasi <i>mobile</i> android ini bisa memberitahu

	Berbasis <i>Client Server</i> Dengan <i>Platform</i> Android	Omahe Cafe and Resto, Menghasilkan aplikasi yang dapat memberitahu pesanan pelanggan ke bagian dapur dan kasir.	pesanan pelanggan ke bagian dapur dan kasir, cepat, selain itu juga keamanannya bisa lebih terjamin.
Liliany Candra dan Ari Amir	Aplikasi Pemesanan Makanan Pada Bangka Original	Menghasilkan aplikasi yang dapat memberitahu pesanan pelanggan ke bagian dapur, bar, kasir dan pelayan tidak perlu mencatat pesanan sehingga membantu mempercepat sistem pemesanan makanan pada restoran.	Sebuah aplikasi pemesanan makanan pada Bangka Original Cafe yang terintegrasi di mana pemesanan makanan dari pelanggan akan dimasukkan di <i>mobile</i> android oleh pelayan, kemudian <i>view</i> di bagian dapur, bar untuk diproses dan dicetak di bagian kasir.

Sumber : Dokumen Pribadi

C. Kerangka Berpikir



Gambar 2.2
Kerangka Berpikir
Sumber : Dokumen Pribadi

1. Penjelasan Dari Gambar Kerangka Berpikir

Pada kondisi saat ini pemesanan di Osaka Ramen Depok terdapat beberapa permasalahan karena sistem pemesanan yang digunakan di Osaka Ramen Depok masih dengan cara konvensional. Cara ini terbilang kuno karena masih menggunakan kertas untuk mencatat pesanan pelanggan. Saat melakukan pembayaran, kasir harus menghitung terlebih dahulu menggunakan kalkulator untuk mengetahui total pembayaran. Setiap harinya karyawan atau kasir Osaka Ramen harus membuat laporan penjualan secara manual kemudian dicatat ke buku laporan. Pembuatan laporan seperti itu tidaklah akurat dan dapat mengalami kesalahan saat menghitung transaksi pembayaran.

Dengan bantuan beberapa faktor pendukung seperti komputer, bahasa pemrograman Java, IntelliJ IDEA, dan XAMPP, serta mengikuti aturan yang dipakai seperti pemesanan hanya dapat dilakukan di tempat/restoran, pemesanan dapat dilakukan untuk makan di tempat atau dibawa pulang, dan pembayaran hanya dapat dilakukan menggunakan uang tunai, maka dapat dibuat sebuah sistem pemesanan berbasis Java sehingga diharapkan dapat mengubah sistem pemesanan yang sebelumnya masih menggunakan cara konvensional menjadi sistem yang terkomputerisasi, mengurangi kesalahan yang terjadi saat proses pemesanan, mempercepat proses pembayaran, dan membantu dalam proses pembuatan laporan sehingga lebih akurat.

BAB III

METODE PENELITIAN

A. Waktu dan Tempat Penelitian

1. Waktu Penelitian

Waktu penelitian berlangsung dari bulan Februari 2019 sampai dengan bulan Juni 2019, dengan perincian jadwal sebagai berikut:

Tabel 3.1
Jadwal Penelitian

No.	Kegiatan	Februari				Maret				April				Mei				Juni			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1.	Identifikasi																				
	Analisis dan																				
2.	Pengumpulan Data																				
	Perancangan																				
3.	Sistem																				
4.	Desain																				
	Pengkodean																				
5.	dan Pengujian																				
6.	Implementasi dan Evaluasi																				
7.	Penulisan Laporan																				

Sumber : Dokumen Pribadi

2. Tempat Penelitian

Tempat penelitian penulis dilakukan di Osaka Ramen di Jl. Keadilan No. 23G, Rangkapan Jaya Baru, Pancoran Mas, Kota Depok, Jawa Barat.

B. Desain Penelitian

Desain penelitian yang dilakukan menggunakan metode deskriptif, yaitu metode untuk berusaha mengumpulkan, menyajikan, serta menganalisis data sehingga dapat memberikan gambaran yang jelas. Dalam penelitian ini terdapat beberapa jenis data dan sumber data yang digunakan.

1. Jenis Data yang Dikumpulkan

a. Data Kuantitatif

Data yang diperoleh dari mitra yang berhubungan dengan penelitian seperti tanggapan pegawai mengenai proses pemesanan yang digunakan selama ini di Osaka Ramen Depok.

b. Data Kualitatif

Data yang berupa penjelasan langsung dari pemilik Osaka Ramen Depok atau pegawai yang bersangkutan dengan permasalahan penelitian.

2. Sumber Data yang Digunakan

a. Data primer

Data yang didapatkan dari sumber penelitian, dengan cara wawancara langsung. Dalam penelitian ini penulis memperoleh data dari pemilik/*owner* dari Osaka Ramen Depok.

b. Data Sekunder

Data yang diperoleh dari laporan-laporan, dokumentasi, dan sumber-sumber lainnya yang berhubungan dengan permasalahan yang sedang dihadapi.

C. Metode Pengumpulan Data

Metode yang digunakan dalam pengumpulan data pada penelitian ini adalah observasi, studi literatur, dan wawancara.

1. Observasi

Observasi adalah pengamatan suatu objek untuk mendapatkan pemahaman yang mendalam dengan cara merasakan langsung atau melihat objek yang ingin diamati. Tujuan observasi yaitu untuk mengetahui secara langsung sistem atau metode pemesanan yang digunakan di Osaka Ramen Depok.

2. Studi Literatur

Pada tahap ini penulis mengumpulkan literatur dari buku-buku referensi dan jurnal yang berhubungan dengan permasalahan dalam penelitian ini.

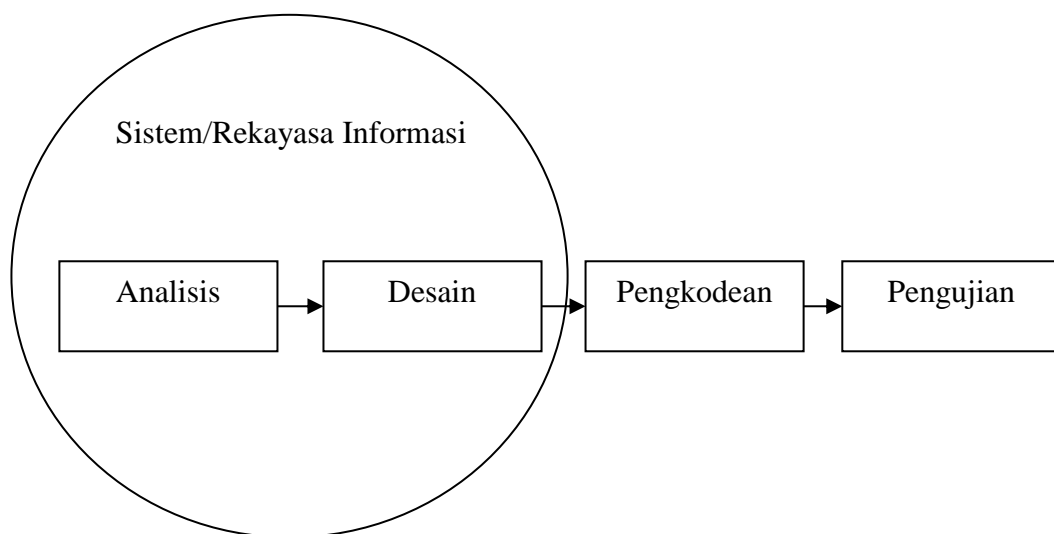
3. Wawancara

Wawancara adalah suatu teknik pengumpulan data melalui tanya jawab antara narasumber dan pewawancara. Pada penelitian ini dilakukan wawancara di mana narasumber adalah pemilik atau *owner* dari Osaka Ramen Depok dan pewawancara adalah penulis sendiri.

D. Langkah - Langkah Pengembangan Sistem

Langkah-langkah pengembangan sistem yang digunakan penulis dalam menyelesaikan penelitian adalah dengan menggunakan model *waterfall*.

“Model *waterfall* menyediakan pendekatan dalam membangun perangkat lunak secara terurut dimulai dari analisa, desain, pengkodean, pengujian, dan tahap pendukung (*support*)” (Rosa & Shalahuddin, 2013).



Gambar 3.1
Model *Waterfall*
Sumber : (Rosa & Shalahuddin, 2013)

1. Tahapan Metode *Waterfall*

a. Analisis

Analisis dapat dilakukan dengan cara proses pengumpulan data seperti wawancara, studi literatur, atau observasi. Untuk membuat sistem yang dapat memenuhi kebutuhan pengguna, dibutuhkan informasi berupa kebutuhan-kebutuhan pengguna terhadap sistem. Maka dari itu sebaiknya perlu dipelajari apa saja yang dibutuhkan pengguna agar terciptanya sistem yang dapat bermanfaat.

b. Desain

Setelah kebutuhan sistem dianalisis, tahap selanjutnya adalah melakukan perancangan dari data hasil analisa menjadi desain sistem agar mempermudah dalam melakukan implementasi.

c. Pengkodean

Setelah desain sistem dilakukan, maka yang perlu dilakukan adalah mengubah desain sistem ke suatu bentuk bahasa yang dapat dimengerti oleh komputer. Pada tahap ini desain sistem akan diimplementasikan menggunakan bahasa pemrograman yang akan dikerjakan oleh programmer.

d. Pengujian

Pengujian dilakukan untuk memastikan apakah sistem yang dibuat sudah sesuai dengan yang diharapkan. Apabila masih terjadi kekurangan atau kesalahan, maka akan dilakukan perbaikan sampai program sesuai dengan harapan.

2. Keunggulan dan Kelemahan Metode *Waterfall*

a. Keunggulan

- 1) Tahapan tidak membingungkan karena dilakukan secara berurut.
- 2) Mudah diterapkan dalam mengembangkan sistem yang tidak terlalu besar.

b. Kelemahan

- 1) Tidak cocok diterapkan untuk mengembangkan sistem yang rumit dan besar.

BAB IV

ANALISIS SISTEM BERJALAN DAN RANCANGAN SISTEM YANG DIUSULKAN

A. Profil Perusahaan

1. Sejarah Osaka Ramen Depok

Osaka Ramen adalah restoran ramen khas Jepang yang menyediakan beberapa variasi ramen dengan harga terjangkau. Osaka Ramen juga menyediakan berbagai makanan dan minuman lokal Indonesia sehingga menu yang dijual tidak hanya ramen saja. Osaka Ramen Depok yang beralamat di Jl. Keadilan No. 23G, Rangkapan Jaya Baru, Pancoran Mas, Depok mulai beroperasi tahun 2011.

2. Visi dan Misi Osaka Ramen

a. Visi

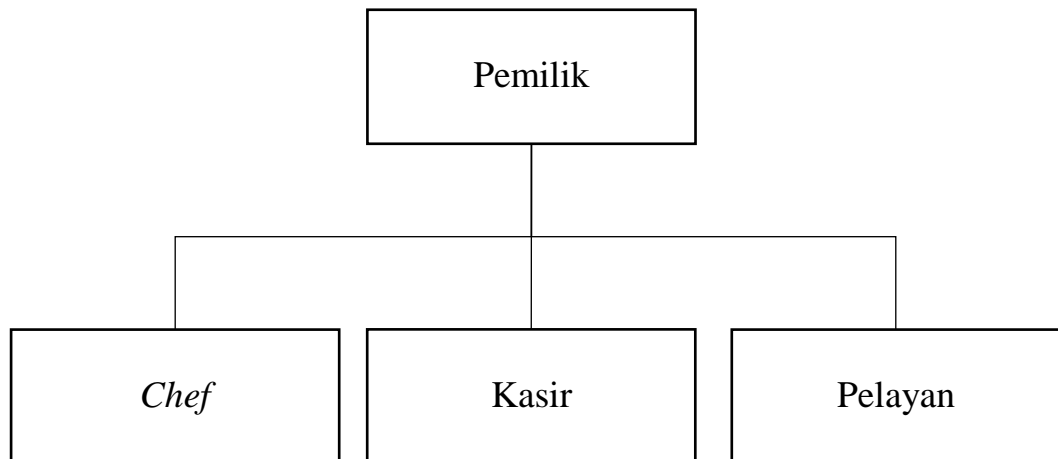
Mengenalkan masakan Jepang khususnya ramen kepada masyarakat kelas menengah ke bawah.

b. Misi

Memberikan cita rasa ramen yang berkualitas dengan harga terjangkau untuk masyarakat kelas menengah ke bawah.

B. Struktur Organisasi Perusahaan

1. Gambar Struktur Organisasi



Gambar 4.1
Struktur Organisasi
Sumber : Pemilik Osaka Ramen Depok

2. Deskripsi Kerja

Dari struktur organisasi yang ada di Osaka Ramen, akan diuraikan tugas dan tanggung jawabnya sebagai berikut:

a. Pemilik

- 1) Membuat perencanaan, strategi dan kebijakan yang menyangkut operasi Osaka Ramen.
- 2) Melakukan kontrol secara keseluruhan atas operasi Osaka Ramen.
- 3) Memegang kendali atas keputusan penting yang bersifat umum yang berkaitan dengan finansial.

b. Chef

- 1) Menyajikan makanan dan minuman sesuai pesanan pelanggan.
- 2) Mengawasi jalannya operasional dapur.

c. Kasir

- 1) Menjalankan proses penjualan dan pembayaran.
- 2) Melakukan pencatatan atas semua transaksi.
- 3) Melakukan pelaporan penjualan kepada pemilik Osaka Ramen.

d. Pelayan

- 1) Menyajikan makanan dan minuman kepada pelanggan dengan sopan, ramah, dan memberikan pelayanan terbaik demi kepuasan pelanggan.
- 2) Membersihkan dan mengatur semua meja, kursi, dan peralatan lainnya yang ada di restoran.
- 3) Memastikan bahwa semua minuman dan makanan yang disajikan sesuai dengan pesanan pelanggan.

C. Proses Bisnis Sistem Berjalan

1. Proses Pemesanan

- a. Pelanggan datang ke Osaka Ramen dan duduk di tempat yang diinginkan.
- b. Pelayan akan menghampiri pelanggan yang baru datang dan memberikan buku menu, kertas dan pulpen untuk menulis pesanan.
- c. Pelanggan yang sudah selesai memilih menu dan menuliskannya di kertas akan memanggil pelayan dan memberikan kertas pesanan.
- d. Pelayan akan memberitahukan kepada bagian dapur/*chef* untuk menyajikan makanan dan minuman sesuai pesanan.
- e. Setelah makanan dan minuman disajikan, pelayan akan membawakan makanan dan minuman ke meja pelanggan.

2. Proses Pembayaran

- a. Setelah pelanggan selesai makan, pelanggan menghampiri kasir untuk melakukan pembayaran.
- b. Kasir akan menanyakan nomor meja pelanggan tersebut.
- c. Setelah kasir mengetahui nomor meja pelanggan, kasir akan mencari kertas pesanan dan menghitung total harga secara manual.
- d. Kasir memberitahukan total harga pesanan, selanjutnya pelanggan membayar pesanan tersebut.
- e. Kasir memberikan bukti pembayaran dan kembalian apabila uang pelanggan melebihi total harga pesanan.

D. Aturan Bisnis Sistem Berjalan

Dengan menganalisis proses-proses bisnis sistem berjalan maka dapat diketahui aturan bisnis sistem berjalan sebagai berikut:

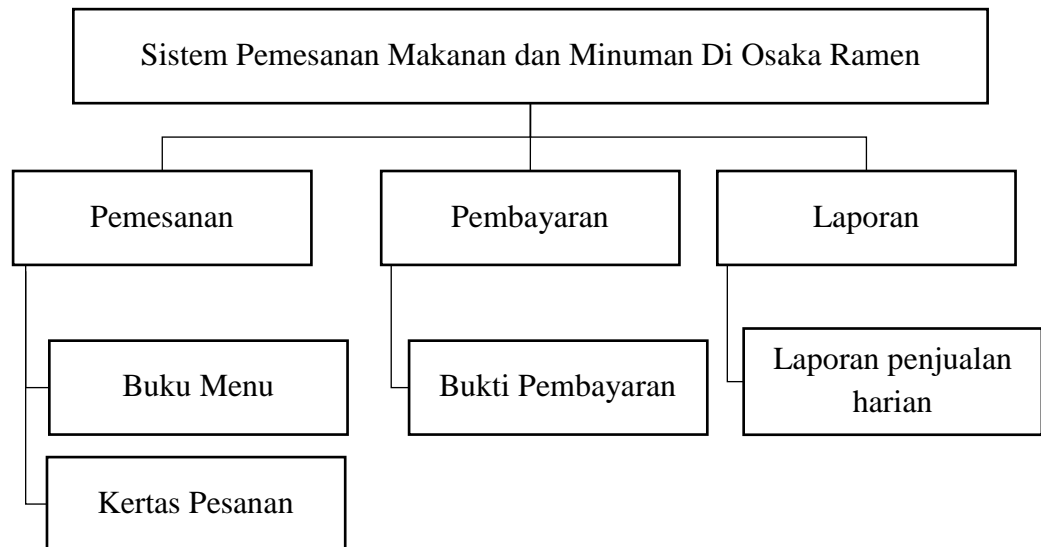
1. Pemesanan

Pemesanan dilakukan secara manual menggunakan kertas dan pulpen untuk mencatat pesanan. Pelayan membiarkan pelanggan untuk mencatat pesanan sendiri.

2. Pembayaran

Pembayaran boleh dilakukan setelah pelanggan selesai makan atau saat makanan dan minuman tiba di meja.

E. Dekomposisi Fungsi Sistem



Gambar 4.2
Dekomposisi Fungsi Sistem
Sumber : Dokumen Pribadi

F. Analisis Masukan (*Input*), Proses dan Keluaran (*Output*) Sistem Berjalan

1. Analisis Masukan (*Input*)

- | | |
|-----------------|---|
| a. Nama Masukan | : Data pesanan |
| Fungsi | : Untuk mengetahui pesanan pelanggan |
| Sumber | : Pelanggan |
| Media | : Kertas |
| Frekuensi | : Setiap ada pemesanan |
| Keterangan | : Berisi tentang menu yang dipilih dan jumlah pesanan |

2. Analisis Proses

a. Proses Pemesanan

Proses pemesanan terdiri dari proses penerimaan pesanan menggunakan kertas untuk mencatat pesanan sampai penyajian pesanan.

b. Proses Pembayaran

Proses pembayaran yaitu melakukan penghitungan harga dari kertas pesanan sampai mencatat data pesanan beserta harga ke bukti pembayaran.

c. Proses Pembuatan Laporan

Proses ini yaitu menghitung total transaksi setiap harinya dan menghitung secara manual pemasukan yang didapat kemudian dicatat ke buku laporan.

3. Analisis Keluaran (*Output*)

a. Nama Keluaran : Bukti pembayaran

Fungsi : Sebagai bukti transaksi pembayaran

Sumber : Kasir

Media : Kertas

Frekuensi : Setiap terjadi pembayaran

Keterangan : Berisi tentang daftar pesanan, harga tiap pesanan, dan total harga pesanan

b. Nama Keluaran : Laporan harian

Fungsi : Untuk mengetahui total pemasukan setiap harinya

Sumber : Kasir

Media : Buku laporan

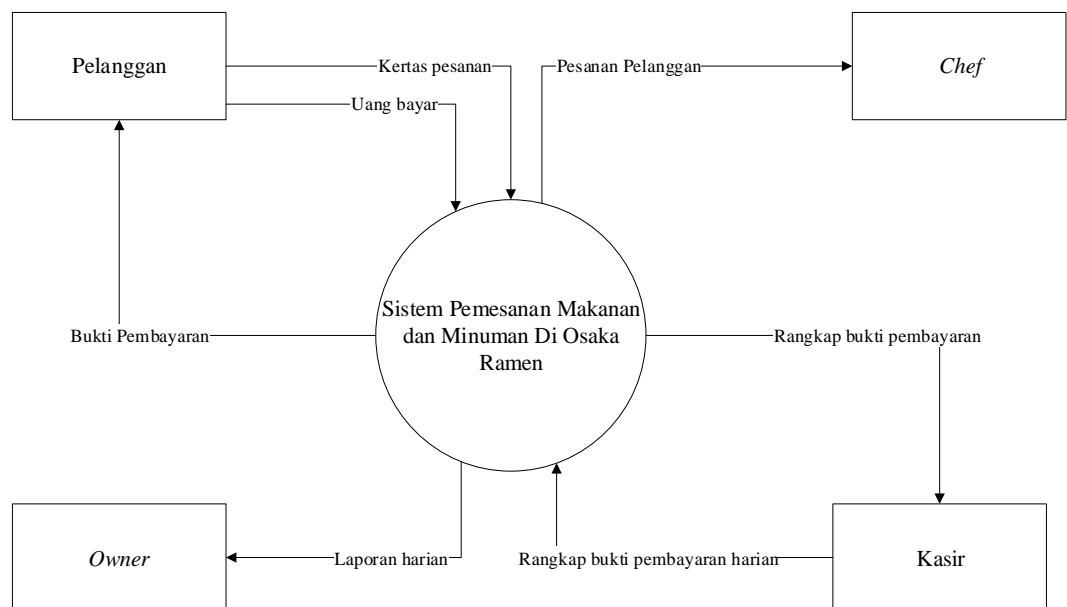
Frekuensi : Setiap hari

Keterangan : Berisi tentang data laporan berupa
total pemasukan dalam sehari

G. Diagram Alir Data (DAD) Sistem Berjalan (Diagram Konteks, Nol, Rinci)

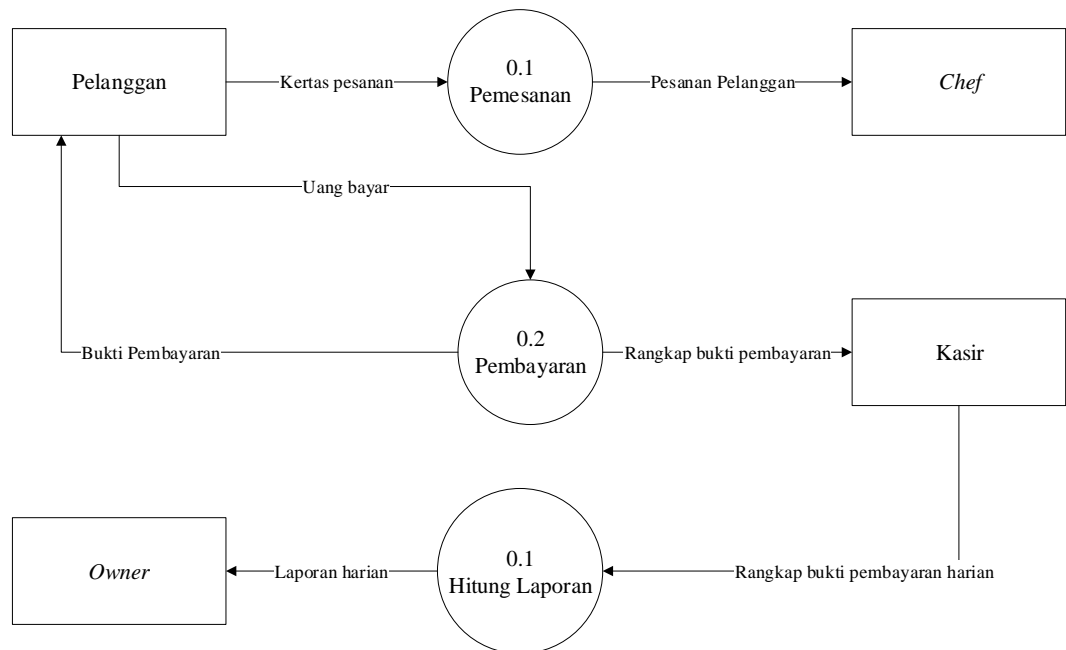
1. Diagram Konteks Sistem Berjalan

Proses diagram konteks dalam sistem berjalan yang berlangsung pada
Osaka Ramen adalah:



Gambar 4.3
Diagram Konteks Sistem Berjalan
Sumber : Dokumen Pribadi

2. Diagram Nol Sistem Berjalan



Gambar 4.4
Diagram Nol Sistem Berjalan
Sumber : Dokumen Pribadi

H. Analisis Permasalahan

Dari hasil penelitian yang penulis lakukan maka penulis dapat menganalisis permasalahan yang ada pada pemesanan di Osaka Ramen yaitu:

1. Proses pemesanan makanan dan minuman yang dilakukan masih menggunakan kertas dan pulpen dalam pencatatan pesanan sehingga terkadang terjadi kesalahan pesanan.
2. Perhitungan total pembayaran masih manual menggunakan kalkulator yang kemungkinan dapat terjadi kesalahan dan membutuhkan waktu saat proses menghitung total pembayaran setiap transaksi.
3. Proses pembuatan laporan masih manual dan harus dihitung semua transaksi setiap harinya.

I. Alternatif Penyelesaian Masalah

Alternatif penyelesaian masalah yang akan penulis buat adalah dengan membuat sistem pemesanan makanan dan minuman secara terkomputerisasi, yang bertujuan untuk mempermudah dalam proses pengolahan data pesanan dan data transaksi penjualan. Sistem ini akan dibuat dengan bahasa pemrograman Java yang dibagi menjadi dua jenis yaitu aplikasi *server* dan aplikasi *client* yang saling terhubung melalui jaringan komputer dan saling berinteraksi mengirimkan data melalui format *JSON*. Aplikasi *client* didesain untuk digunakan pelanggan yang berisi untuk menampilkan daftar menu makanan dan minuman serta mengelola pesanan pelanggan, sedangkan aplikasi *server* digunakan pelayan Osaka Ramen untuk mengelola pesanan masuk dan melakukan transaksi pembayaran.

Dengan menggunakan sistem ini diharapkan proses pemesanan dan pembayaran di Osaka Ramen menjadi lebih mudah, efektif, mengecilkan risiko kesalahan manusia, serta membuat daya tarik pengunjung atau pelanggan Osaka Ramen yang memberikan pengalaman dalam memesan makanan dan minuman di tempat makan dengan cara yang modern dan masa kini. Sistem ini baik dari sisi *aplikasi server* maupun *client* didesain dengan *interface* yang semudah mungkin dipahami sehingga pelanggan atau pelayan Osaka Ramen tidak kesulitan saat mengoperasikannya.

J. Aturan Bisnis Sistem Diusulkan

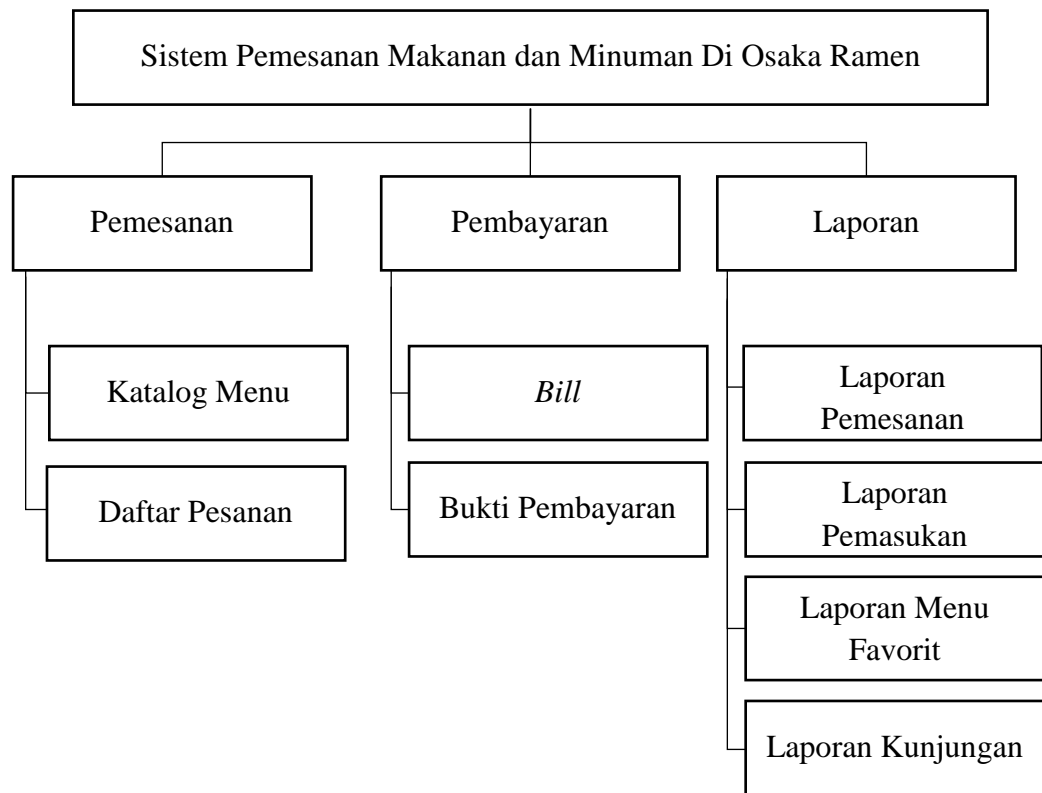
1. Pemesanan

Pelanggan memesan menggunakan komputer yang disediakan di setiap meja yang sudah terdapat aplikasi untuk memesan. Setelah memesan melalui aplikasi *client*, pelanggan hanya tinggal menunggu pesanan diantarkan ke meja tersebut. Jika pelanggan merasa kurang dengan pesannya, pelanggan dapat memesan kembali menggunakan aplikasi yang sama. Tetapi jika pelanggan telah menekan tombol bayar di dalam aplikasi tersebut, pelanggan harus menyelesaikan pembayaran terlebih dahulu jika ingin kembali memesan.

2. Pembayaran

Untuk melakukan pembayaran pelanggan dapat menekan tombol bayar di aplikasi *client* dan menunggu pelayan untuk mengantarkan *bill* atau tagihan pembayaran. Setelah itu pelanggan dapat membayar tagihan kepada pelayan. Pelayan akan kembali ke komputer *server* untuk menyelesaikan pembayaran dan mencetak bukti pembayaran.

K. Dekomposisi Fungsi Sistem Diusulkan



Gambar 4.5
Dekomposisi Fungsi Sistem Diusulkan
Sumber : Dokumen Pribadi

L. Rancangan Masukan, Proses, dan Keluaran

1. Rancangan Masukan

- a. Nama Masukan : Data pesanan
- Fungsi : Untuk mengetahui pesanan pelanggan
- Sumber : Pelanggan
- Frekuensi : Setiap ada pemesanan
- Keterangan : Berisi tentang menu yang dipilih dan jumlah pesanan

- b. Nama Masukan : Data pembayaran
- Fungsi : Untuk melakukan transaksi pembayaran
- Sumber : Kasir
- Frekuensi : Setiap ada transaksi pembayaran
- Keterangan : Berisi tentang uang tunai yang dibayar pelanggan
- c. Nama Masukan : Data menu baru
- Fungsi : Untuk mendata menu baru ke dalam sistem
- Sumber : Pemilik/*Chef*
- Frekuensi : Setiap ada menu baru
- Keterangan : Berisi tentang data menu baru

2. Rancangan Proses

a. Proses Pemesanan

Pelanggan akan memasukkan data pesanan dengan cara memilih menu di dalam katalog, memasukkan level apabila menu tersebut adalah ramen, dan memasukkan jumlah menu yang ingin dipesan, kemudian pelanggan akan masuk ke daftar pesanan untuk memastikan pesanan-pesanan yang dipilih pelanggan sudah benar. Setelah pelanggan merasa data sudah benar, selanjutnya pelanggan harus menekan tombol pesan di dalam daftar pesanan dan mengkonfirmasi jika data sudah benar.

Data tersebut akan dikirim ke komputer server melalui jaringan komputer lokal yang ada di Osaka Ramen. Di komputer *server*, pelayan atau *chef* akan mengkonfirmasi apakah pesanan dapat disajikan atau tidak. Apabila pesanan dapat disajikan, maka di komputer *client* yang ada di meja pelanggan tersebut dapat melihat bahwa pesannya sedang diproses.

b. Proses Pembayaran

Pelanggan dapat melakukan pembayaran apabila semua pesanan sudah dikonfirmasi saat melakukan pesanan. Pembayaran dapat dilakukan dengan cara masuk ke dalam daftar pesanan kemudian menekan tombol bayar. Apabila permintaan berhasil maka akan muncul sebuah *pop up* yang berisi pesan kepada pelanggan untuk menunggu pelayan mengantarkan tagihan atau *bill*. Data permintaan tersebut akan dikirim ke komputer *server* yang nantinya akan dikonfirmasi oleh kasir untuk mencetak *bill* pembayaran. Pembayaran dapat dilakukan di meja langsung saat pelayan mengantarkan tagihan. Pelayan akan kembali ke komputer *server* untuk memasukkan jumlah uang tunai pelanggan dan mencetak bukti pembayaran. Apabila transaksi sudah selesai, pelayan akan menyimpan data pesanan dengan menekan tombol simpan. Data tersebut akan disimpan ke *database* agar nantinya dapat diolah menjadi data laporan.

c. Proses Pembuatan Laporan

Data-data transaksi yang disimpan di *database* akan dibaca dan akan dikelola sistem untuk dibuat beberapa laporan. Terdapat empat laporan yang dapat dihasilkan yaitu laporan pemesanan, laporan pemasukan, laporan menu favorit, dan laporan kunjungan.

3. Rancangan Keluaran

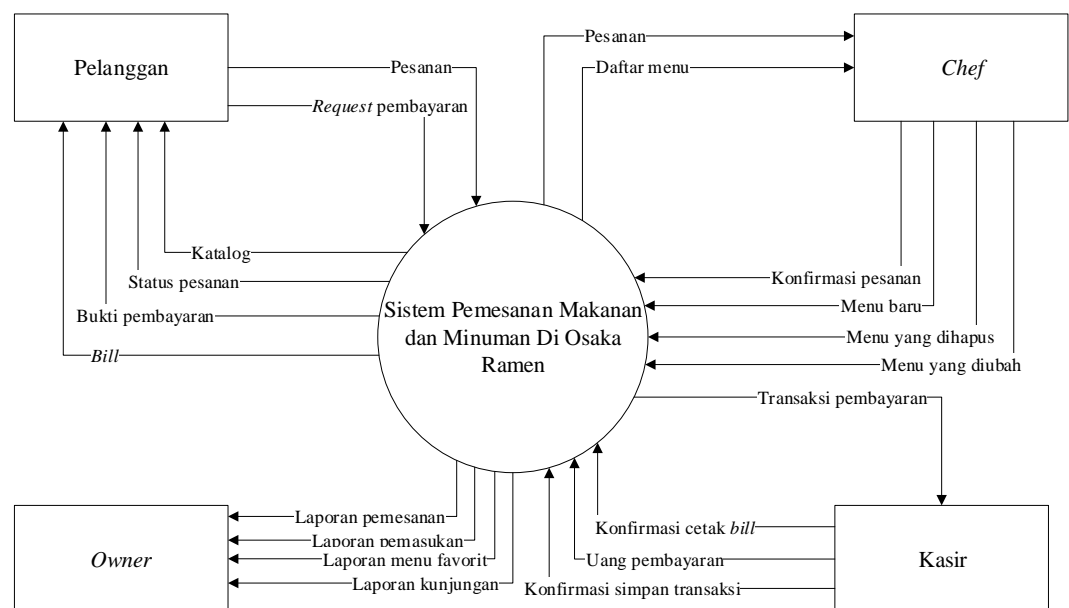
- | | |
|------------------|--|
| a. Nama Keluaran | : <i>Bill</i> |
| Fungsi | : Untuk memberitahukan total biaya pesanan ke pelanggan |
| Sumber | : Kasir |
| Frekuensi | : Setiap pelanggan meminta <i>bill</i> |
| Keterangan | : Berisi tentang data pesanan dan total pembayaran |
| b. Nama Keluaran | : Bukti pembayaran |
| Fungsi | : Untuk memberikan bukti pelanggan telah melakukan pembayaran |
| Sumber | : Kasir |
| Frekuensi | : Setiap pelanggan melakukan pembayaran |
| Keterangan | : Berisi tentang data pesanan, total pembayaran, uang tunai pelanggan, data kembalian pelanggan. |

- c. Nama Keluaran : Laporan pemesanan
- Fungsi : Untuk merekam semua data pemesanan yang terjadi dalam sehari
- Sumber : Kasir
- Frekuensi : Setiap hari
- Keterangan : Berisi tentang data pemesanan seperti nama menu, jumlah, harga, dan total harga
- d. Nama Keluaran : Laporan pemasukan
- Fungsi : Untuk mengetahui total pemasukan dalam satu hari
- Sumber : Kasir
- Frekuensi : Setiap bulan
- Keterangan : Berisi tentang total pemasukan dalam satu hari
- e. Nama Keluaran : Laporan menu favorit
- Fungsi : Untuk mengetahui menu yang sering diminati pengunjung dan melakukan evaluasi terhadap daftar menu
- Sumber : Kasir
- Frekuensi : Setiap bulan
- Keterangan : Berisi tentang daftar menu dan jumlah pesanan setiap menu

- f. Nama Keluaran : Laporan kunjungan
- Fungsi : Untuk mengetahui jumlah pengunjung yang datang dalam satu bulan
- Sumber : Kasir
- Frekuensi : Setiap bulan
- Keterangan : Berisi tentang jumlah pengunjung setiap bulan

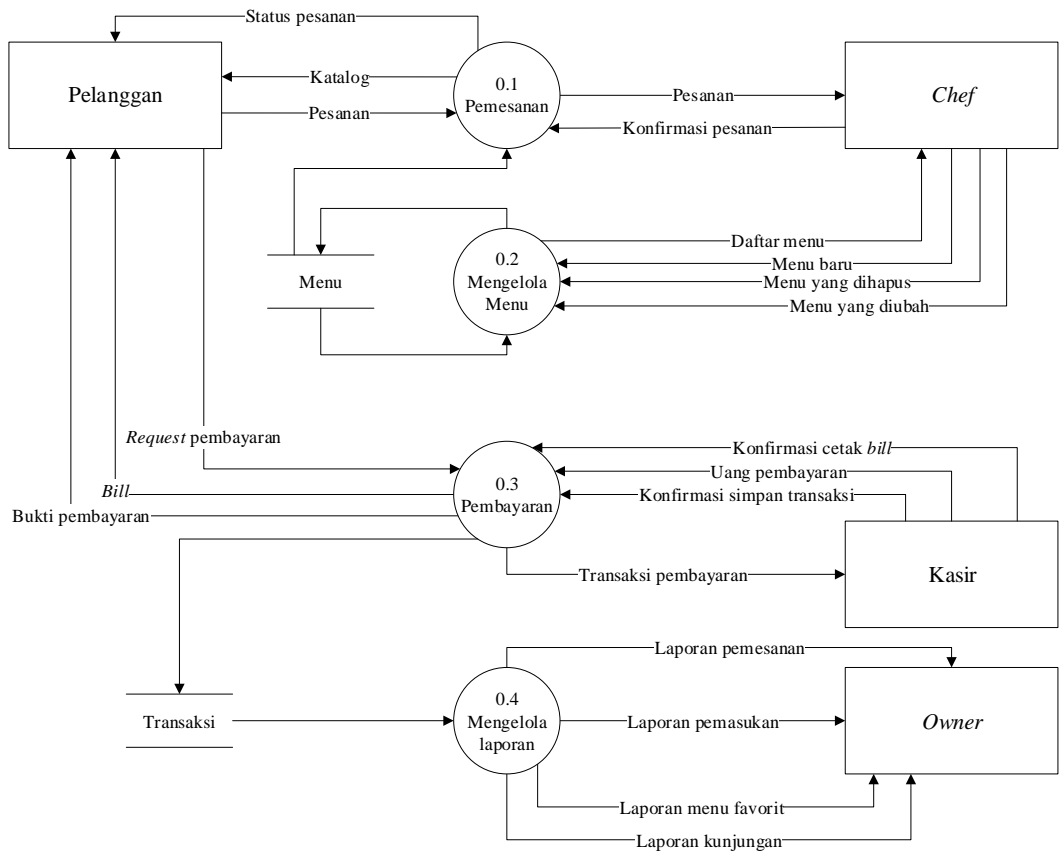
M. Diagram Alir Data (DAD) Sistem yang Diusulkan (Diagram Konteks, Nol, Rinci)

1. Diagram Konteks Sistem yang Diusulkan



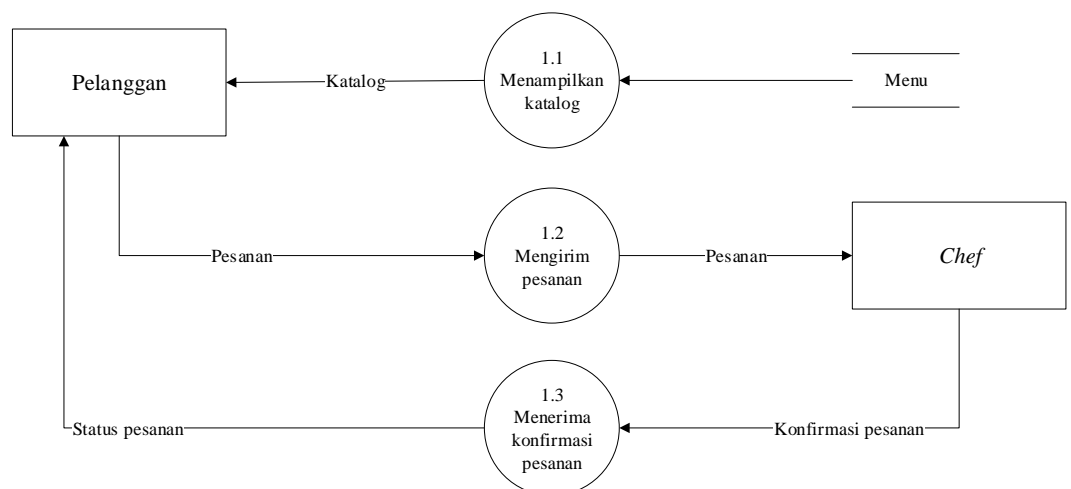
Gambar 4.6
Diagram Konteks Sistem yang Diusulkan
Sumber : Dokumen Pribadi

2. Diagram Nol Sistem yang Diusulkan



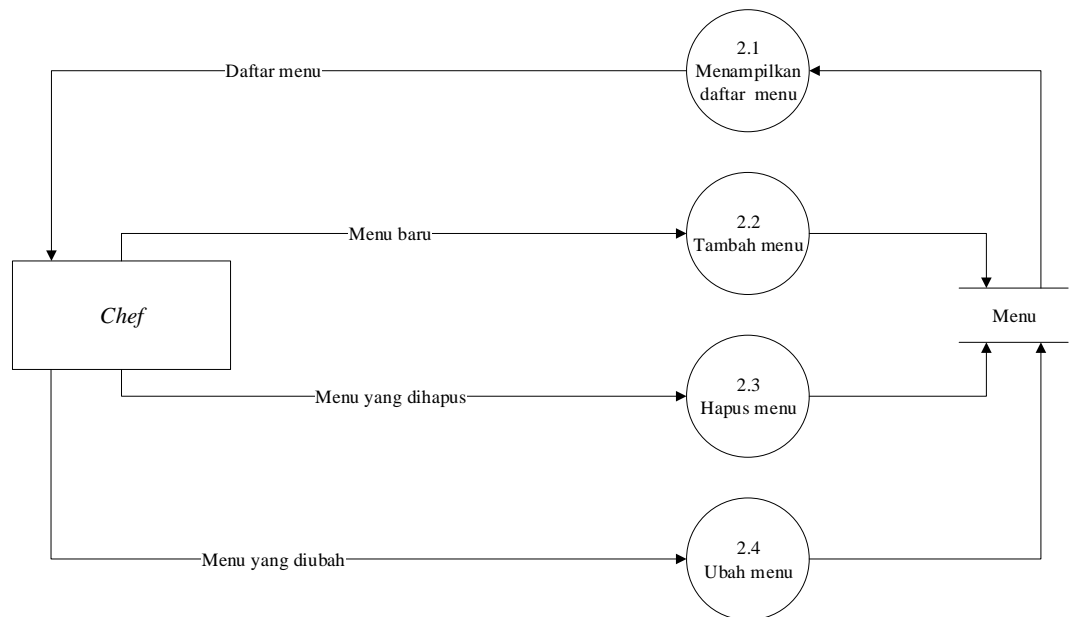
Gambar 4.7
Diagram Nol Sistem yang Diusulkan
Sumber : Dokumen Pribadi

3. Diagram Rinci Level 1 Proses 1



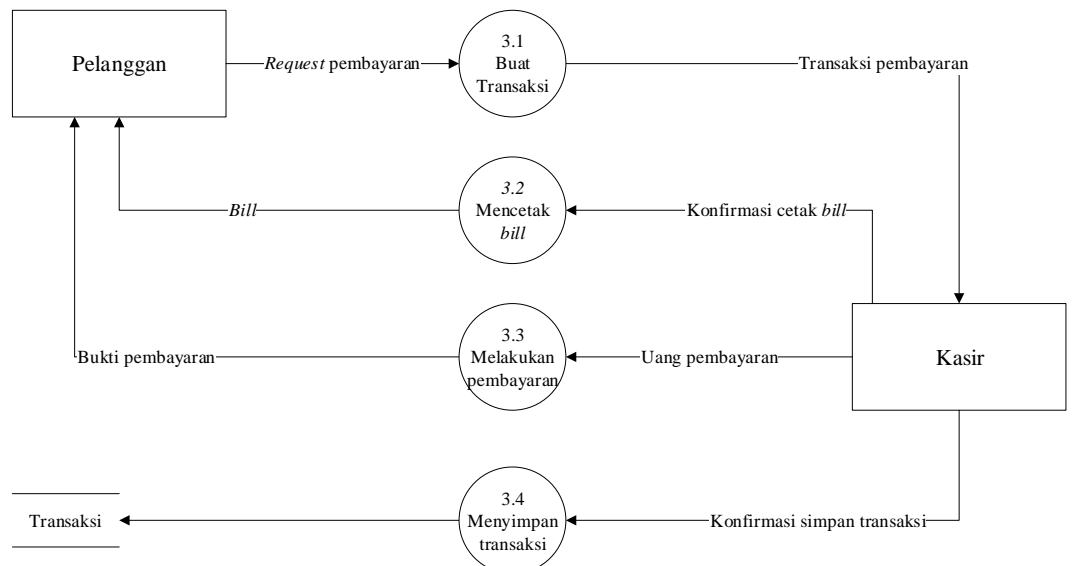
Gambar 4.8
Diagram Rinci Level 1 Proses 1 Sistem yang Diusulkan
Sumber : Dokumen Pribadi

4. Diagram Rinci Level 1 Proses 2



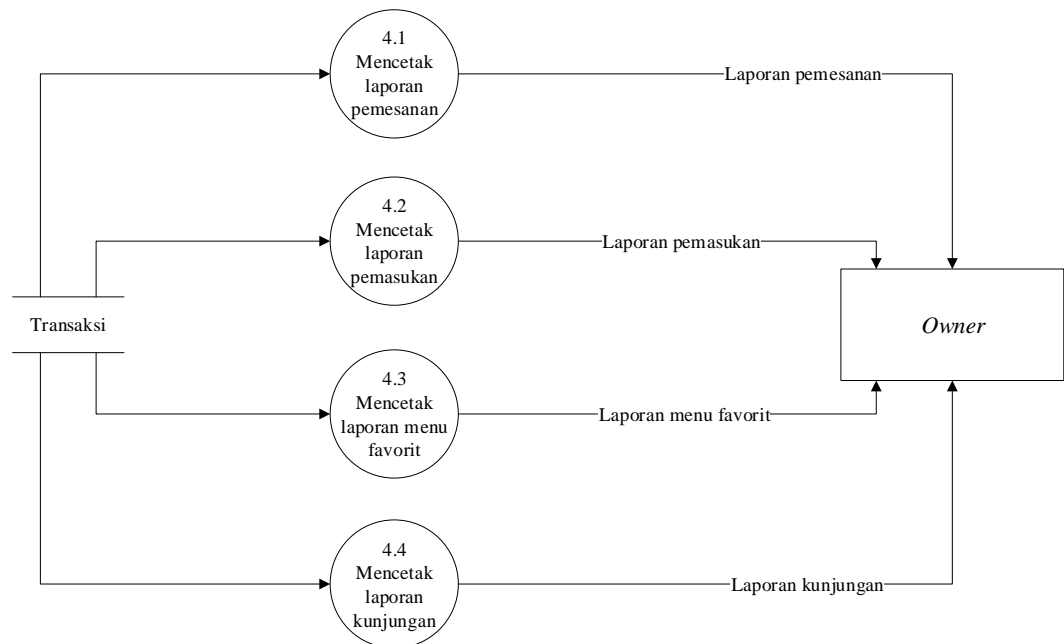
Gambar 4.9
Diagram Rinci Level 1 Proses 2
Sumber : Dokumen Pribadi

5. Diagram Rinci Level 1 Proses 3



Gambar 4.10
Diagram Rinci Level 1 Proses 3
Sumber : Dokumen Pribadi

6. Diagram Rinci Level 1 Proses 4



Gambar 4.11
Diagram Rinci Level 1 Proses 4
Sumber : Dokumen Pribadi

N. Kamus Data Sistem yang Diusulkan

Kamus data merupakan data-data atau informasi yang menjelaskan atau merincikan suatu aliran data pada diagram alir data (DAD) yang diusulkan sebagai berikut:

1. Nama arus data : Katalog
- Alias : Buku menu, Daftar menu
- Bentuk data : Data komputer
- Arus data : Menu – proses pemesanan – pelanggan,
Menu – proses mengelola menu -
chef

Penjelasan	: Berisi data menu makanan dan minuman
Periode	: Setiap pelanggan melihat menu
Volume	: Rata – rata 200 pelanggan setiap hari
Struktur data	: nama_menu + tipe + harga_menu
2. Nama arus data	: Pesanan
Alias	: Pesanan
Bentuk data	: Data komputer
Arus data	: Pelanggan – proses pemesanan - <i>chef</i>
Penjelasan	: Mencatat pesanan pelanggan
Periode	: Setiap pelanggan memesan
Volume	: Rata – rata 400 makanan dan minuman
Struktur data	: nama_menu + jumlah + level + harga_level
3. Nama arus data	: Transaksi
Alias	: <i>Bill</i> , bukti pembayaran
Bentuk data	: Data komputer, kertas
Arus data	: Proses pembayaran – kasir, Proses pembayaran – pelanggan, Proses pembayaran - transaksi
Penjelasan	: Mencatat transaksi pembayaran pelanggan

Periode	: Setiap pelanggan melakukan transaksi pembayaran
Volume	: Rata- rata 200 pengunjung sehari
Struktur data	: id_transaksi + no_meja + tanggal + daftar_pesanan
4. Nama arus data	: Laporan pemesanan
Alias	: Laporan pemesanan
Bentuk data	: Data komputer, kertas
Arus data	: Transaksi - proses mengelola laporan - <i>owner</i>
Penjelasan	: Rekaman semua transaksi setiap hari
Periode	: Setiap hari
Volume	: 1 kali sehari
Struktur data	: pukul + no_meja + nama_menu + jumlah + harga + total_harga
5. Nama arus data	: Laporan pemasukan
Alias	: Laporan pemasukan
Bentuk data	: Data komputer, kertas
Arus data	: Transaksi - proses mengelola laporan - <i>owner</i>
Penjelasan	: Rekaman semua transaksi setiap bulan
Periode	: Setiap bulan

Volume	: 1 kali sebulan
Struktur data	: tanggal + total pemasukan
6. Nama arus data	: Laporan menu favorit
Alias	: Laporan menu favorit
Bentuk data	: Data komputer, kertas
Arus data	: Transaksi - proses mengelola laporan – <i>owner</i>
Penjelasan	: Jumlah pesanan berdasarkan menu makanan dan minuman
Periode	: Setiap bulan
Volume	: 1 kali sebulan
Struktur data	: nama_menu + tipe + harga + total_dipesan
7. Nama arus data	: Laporan kunjungan
Alias	: Laporan kunjungan
Bentuk data	: Data komputer, kertas
Arus data	: Transaksi - proses mengelola laporan – <i>owner</i>
Penjelasan	: Jumlah kunjungan setiap bulan
Periode	: Setiap bulan
Volume	: 1 kali sebulan
Struktur data	: tanggal + total_kunjungan

O. Spesifikasi Proses Sistem yang Diusulkan

Spesifikasi proses menjelaskan spesifikasi dari setiap proses pada diagram rinci sistem pemesanan makanan dan minuman di Osaka Ramen yang diusulkan sebagai berikut:

1. Proses : 1.1
 - Nama proses : Menampilkan katalog
 - Masukan : Daftar menu
 - Keluaran : Katalog
 - Uraian : Daftar menu diambil dari basis data
untuk ditampilkan kepada
pelanggan berdasarkan kategori
menu tersebut
2. Proses : 1.2
 - Nama proses : Mengirim pesanan
 - Masukan : Pesanan
 - Keluaran : Pesanan pelanggan
 - Uraian : Pesanan pelanggan dari komputer
client dikirim ke komputer *server*
dan ditampilkan ke *chef* sehingga
pesanan dapat langsung dibuat
3. Proses : 1.3
 - Nama proses : Menerima konfirmasi pesanan
 - Masukan : Konfirmasi pesanan

- | | | |
|----------|---|--|
| Keluaran | : | Status pesanan |
| Uraian | : | Pesanan akan dikonfirmasi oleh
<i>chef</i> atau pelayan apakah pesanan
tersebut dapat disajikan atau tidak,
jika iya status pesanan akan diubah
menjadi “diproses”, jika tidak
pesanan akan dihapus |
4. Proses : 2.1
- | | | |
|-------------|---|--|
| Nama proses | : | Buat transaksi |
| Masukan | : | <i>Request</i> pembayaran |
| Keluaran | : | Transaksi pembayaran |
| Uraian | : | Pelanggan yang ingin membayar
akan dibuat data transaksi
pembayaran dan data tersebut akan
ditampilkan kepada kasir untuk
diproses |
5. Proses : 2.2
- | | | |
|-------------|---|---|
| Nama proses | : | Mencetak <i>bill</i> |
| Masukan | : | Konfirmasi cetak <i>bill</i> |
| Keluaran | : | <i>Bill</i> |
| Uraian | : | Data transaksi pembayaran
pelanggan akan dicetak dalam
bentuk <i>bill</i> dan akan diserahkan |

kepada pelanggan sebagai tagihan
pembayaran

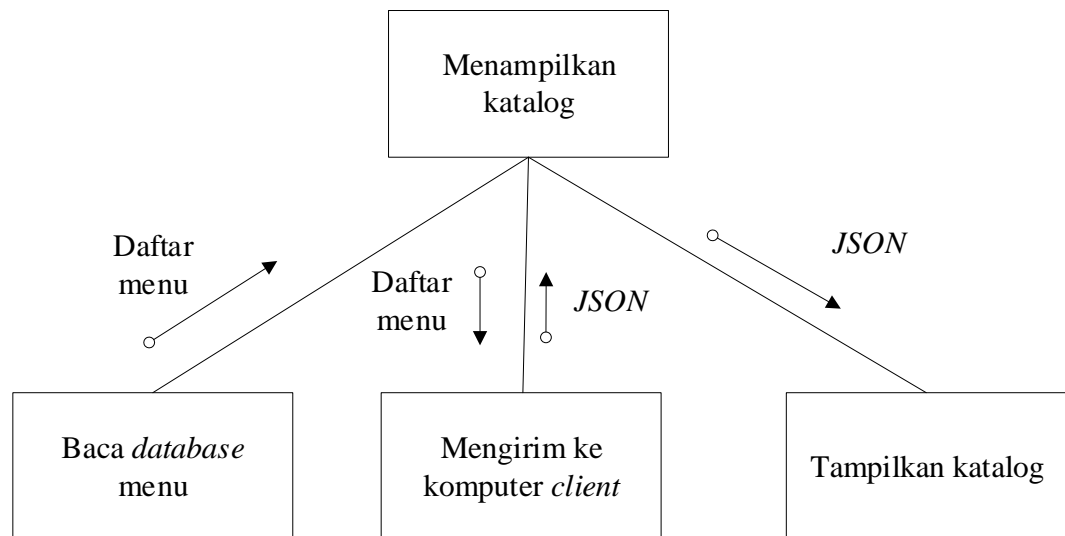
6. Proses : 2.3
 - Nama proses : Melakukan pembayaran
 - Masukan : Uang pembayaran
 - Keluaran : Bukti pembayaran
 - Uraian : Pelanggan yang mendapatkan *bill*
akan melakukan pembayaran
dengan memberikan uang
pembayaran kepada kasir, uang
pembayaran tersebut dimasukkan
oleh kasir ke dalam sistem,
kemudian sistem akan mencetak
bukti pembayaran untuk diserahkan
kepada pelanggan
7. Proses : 2.4
 - Nama proses : Menyimpan transaksi
 - Masukan : Konfirmasi simpan pesanan
 - Keluaran : Transaksi disimpan di *database*
 - Uraian : Setelah kasir melakukan
pembayaran, kasir akan
menyimpan data transaksi tersebut
ke *database*

8. Proses : 3.1
- Nama proses : Mencetak laporan pemesanan
- Masukan : Daftar transaksi
- Keluaran : Laporan pemesanan
- Uraian : Data transaksi diambil dari
database kemudian dipilah
berdasarkan tanggal. Data tersebut
akan dibuat menjadi *file* pdf agar
dapat dicetak
9. Proses : 3.2
- Nama proses : Mencetak laporan pemasukan
- Masukan : Daftar transaksi
- Keluaran : Laporan pemasukan
- Uraian : Data transaksi diambil dari
database kemudian dipilah
berdasarkan tanggal. Data tersebut
akan dibuat menjadi *file* pdf agar
dapat dicetak
10. Proses : 3.3
- Nama proses : Mencetak laporan menu favorit
- Masukan : Daftar transaksi
- Keluaran : Laporan menu favorit

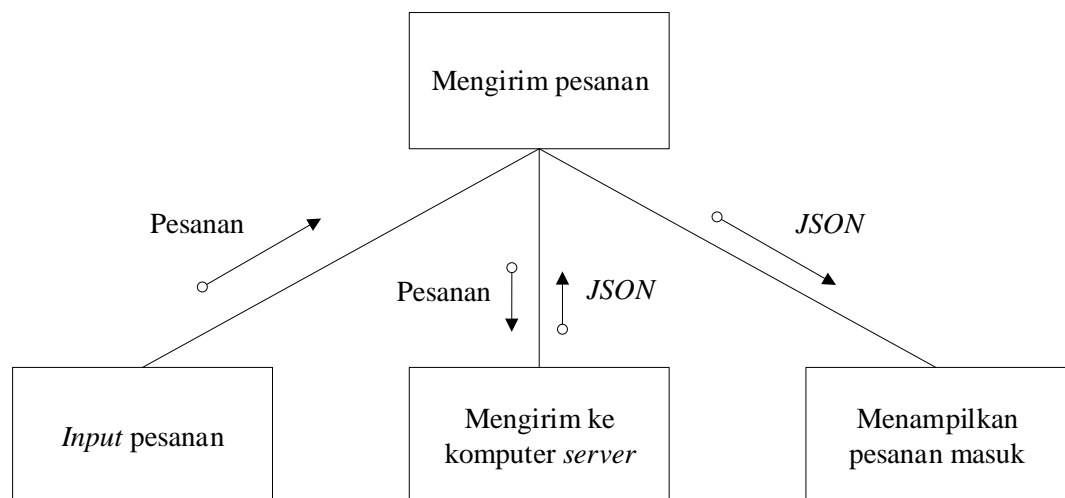
- Uraian : Data transaksi diambil dari *database* kemudian dipilah berdasarkan tanggal. Data tersebut akan dibuat menjadi *file* pdf agar dapat dicetak
11. Proses : 3.4
- Nama proses : Mencetak laporan kunjungan
- Masukan : Daftar transaksi
- Keluaran : Laporan menu favorit
- Uraian : Data transaksi diambil dari *database* kemudian dipilah berdasarkan tanggal. Data tersebut akan dibuat menjadi *file* pdf agar dapat dicetak
12. Proses : 4.1
- Nama proses : Menampilkan daftar menu
- Masukan : Daftar menu
- Keluaran : Daftar menu
- Uraian : Daftar menu diambil dari *database* kemudian ditampilkan kepada *chef* melalui layar
13. Proses : 4.2
- Nama proses : Tambah menu

- | | | |
|----------|---|--|
| Masukan | : | Menu baru |
| Keluaran | : | Menu disimpan di <i>database</i> |
| Uraian | : | Data menu baru dimasukkan oleh <i>chef</i> kemudian data tersebut akan disimpan ke <i>database</i> |
14. Proses : 4.3
- | | | |
|-------------|---|---|
| Nama proses | : | Hapus menu |
| Masukan | : | Menu yang dihapus |
| Keluaran | : | Menu dihapus dari <i>database</i> |
| Uraian | : | Data menu yang tidak ingin lagi dipasarkan akan dipilih oleh <i>chef</i> kemudian data tersebut akan dihapus dari <i>database</i> |
15. Proses : 4.4
- | | | |
|-------------|---|--|
| Nama proses | : | Ubah menu |
| Masukan | : | Menu yang diubah |
| Keluaran | : | Menu diubah di <i>database</i> |
| Uraian | : | Data menu yang tidak sesuai akan diubah oleh <i>chef</i> kemudian data tersebut akan diubah di <i>database</i> |

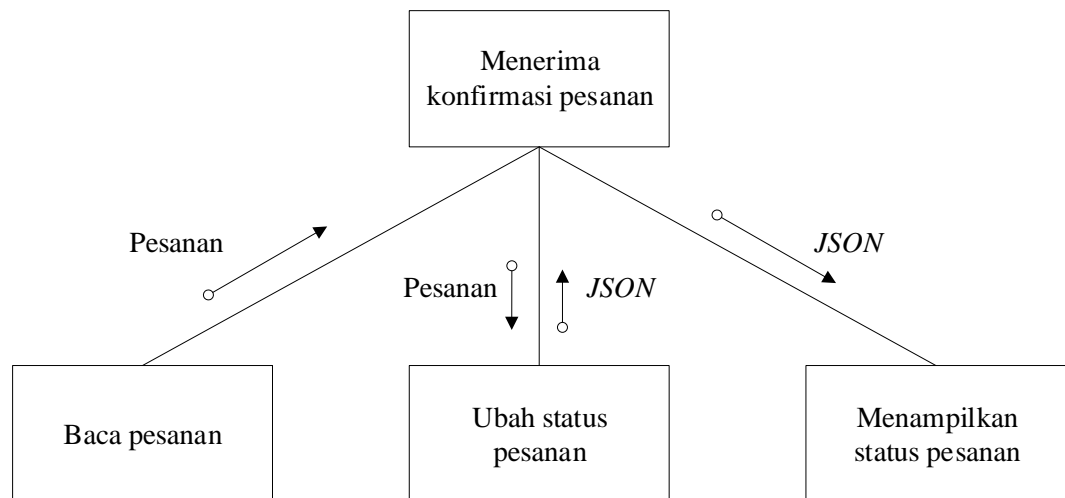
P. Bagan Terstruktur Sistem yang Diusulkan



Gambar 4.12
Bagan Terstruktur Menampilkan Katalog
Sumber : Dokumen Pribadi



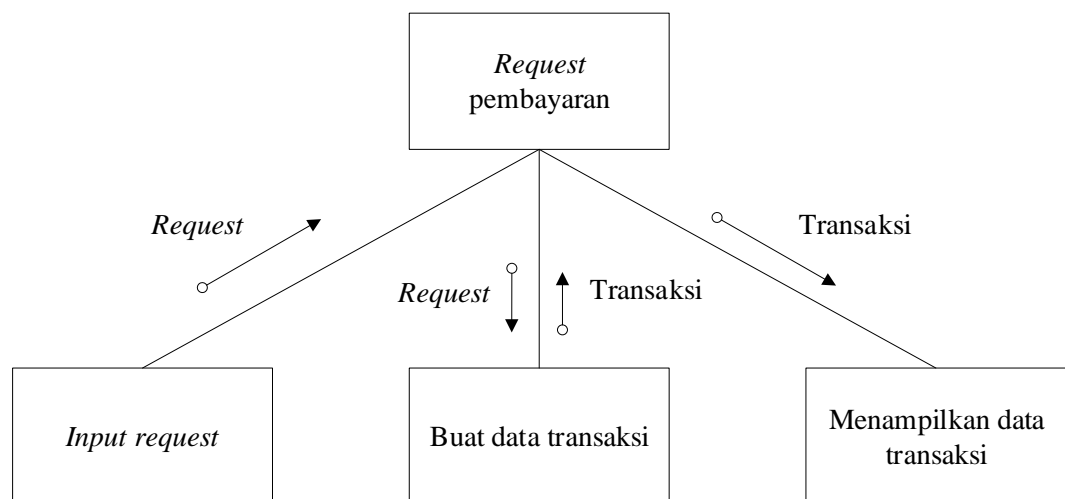
Gambar 4.13
Bagan Terstruktur Mengirim Pesanan
Sumber : Dokumen Pribadi



Gambar 4.14

Bagan Terstruktur Menerima Konfirmasi Pesanan

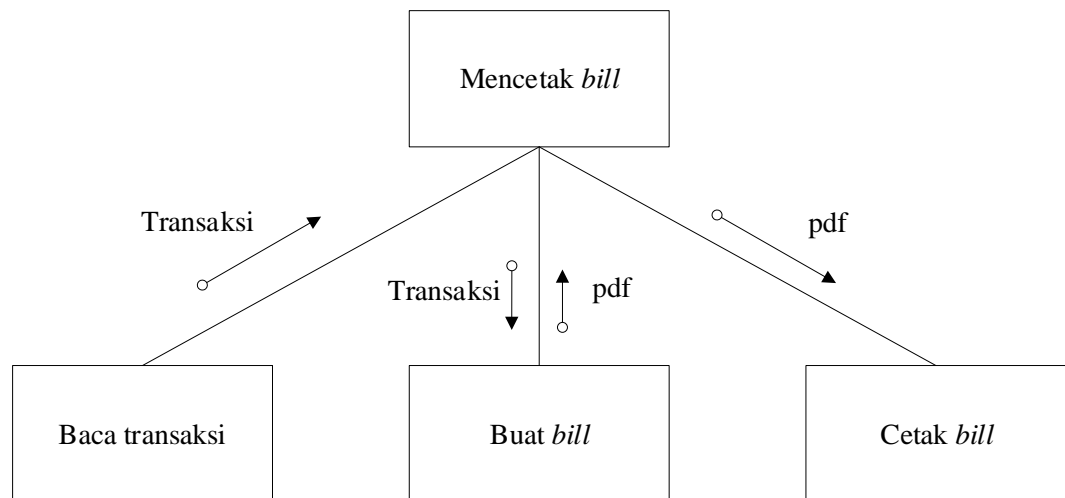
Sumber : Dokumen Pribadi



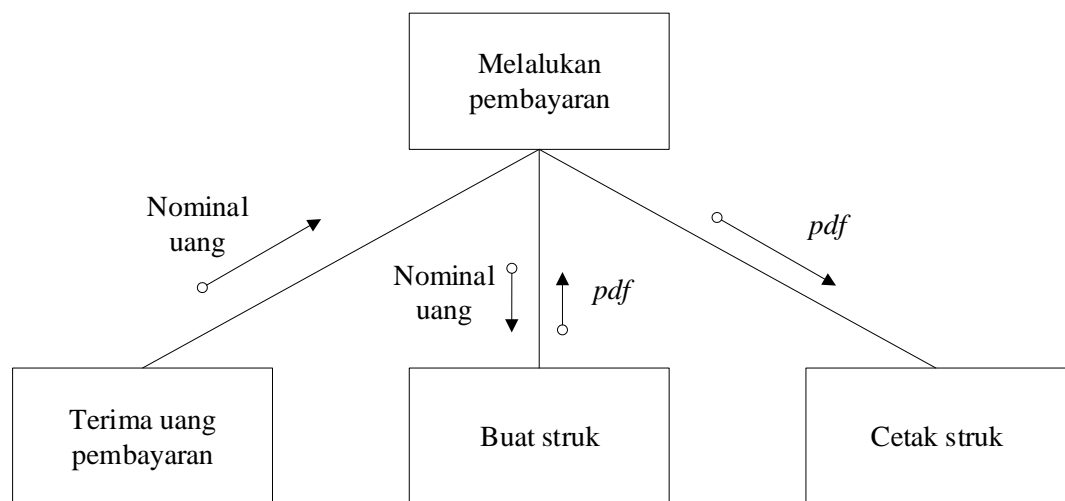
Gambar 4.15

Bagan Terstruktur *Request* Pembayaran

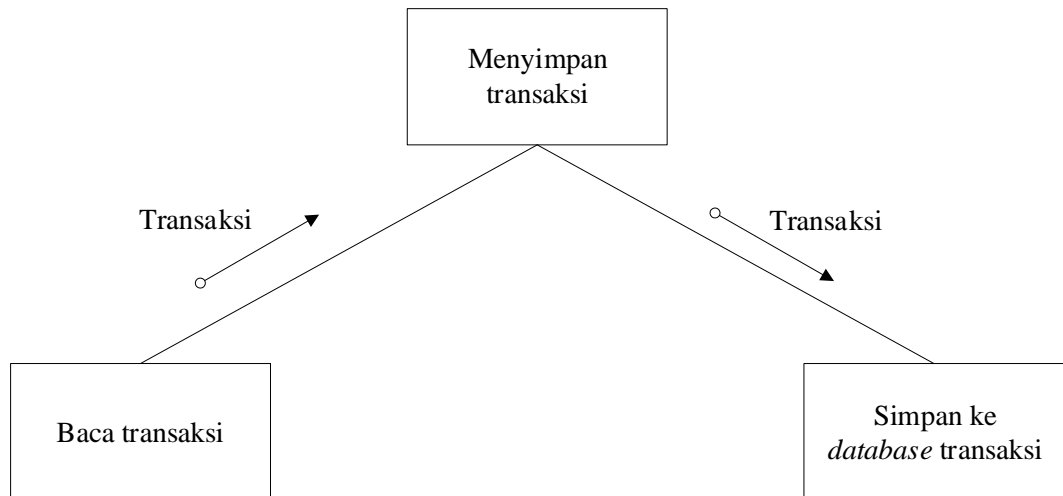
Sumber : Dokumen Pribadi



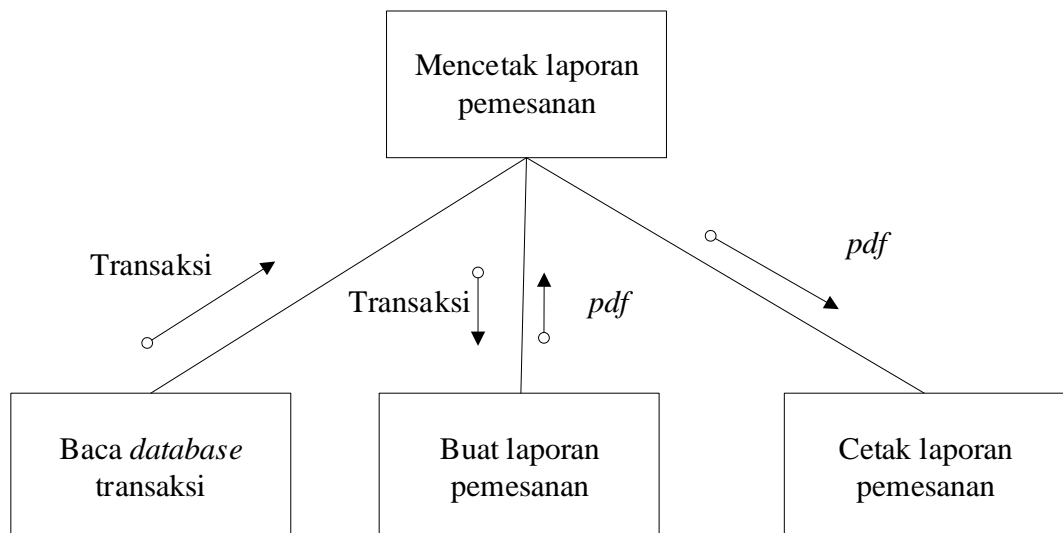
Gambar 4.16
Bagan Terstruktur Mencetak Bill
Sumber : Dokumen Pribadi



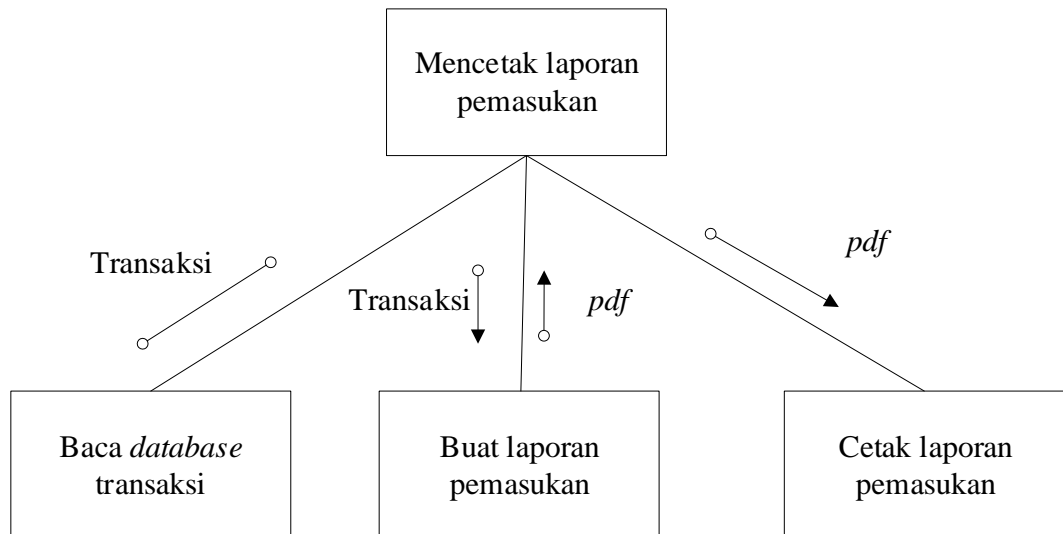
Gambar 4.17
Bagan Terstruktur Melakukan Pembayaran
Sumber : Dokumen Pribadi



Gambar 4.18
Bagan Terstruktur Menyimpan Transaksi
Sumber : Dokumen Pribadi

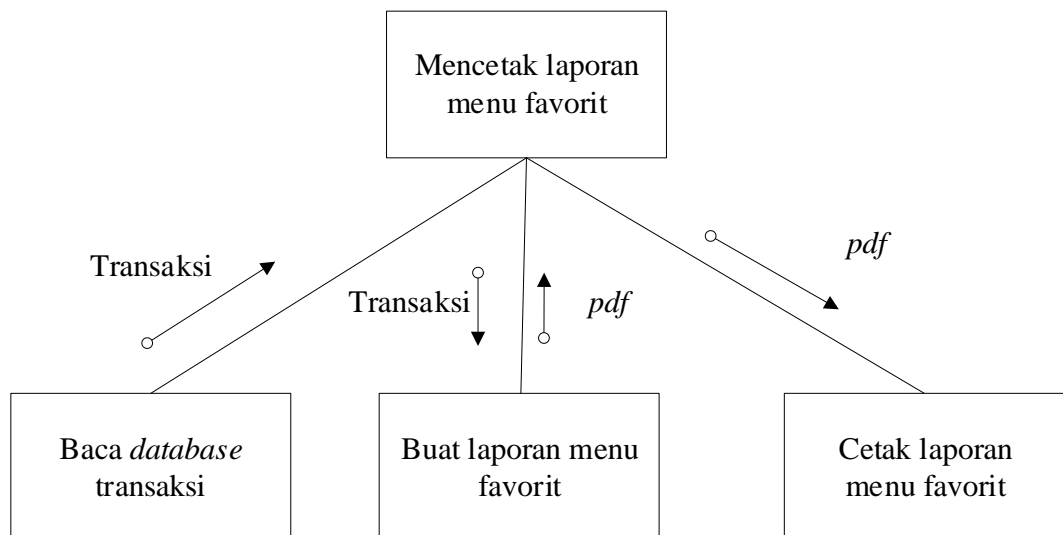


Gambar 4.19
Bagan Terstruktur Mencetak Laporan Pemesanan
Sumber : Dokumen Pribadi



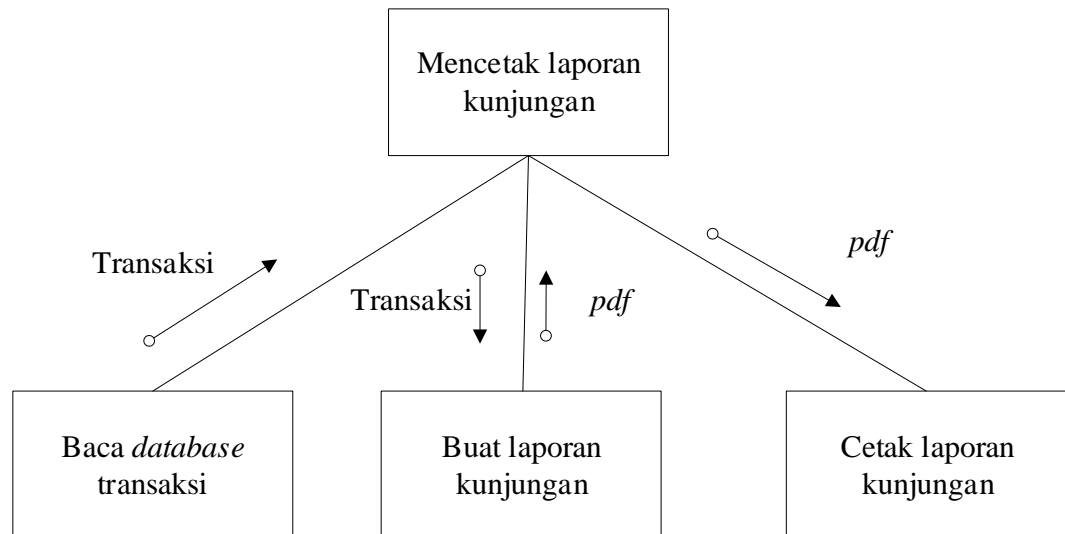
Gambar 4.20

Bagan Terstruktur Mencetak Laporan Pemasukan
Sumber : Dokumen Pribadi

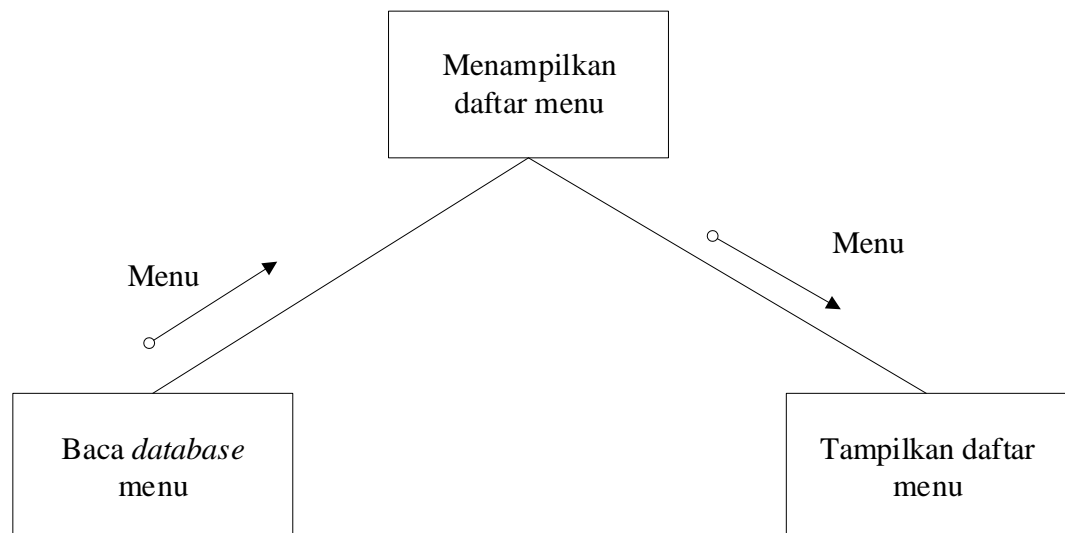


Gambar 4.21

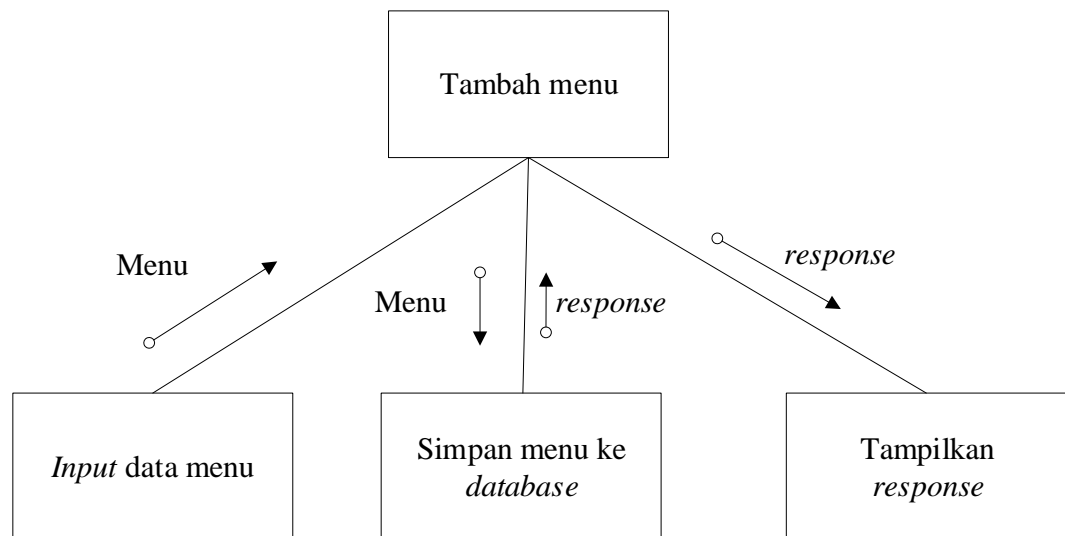
Bagan Terstruktur Mencetak Laporan Menu Favorit
Sumber : Dokumen Pribadi



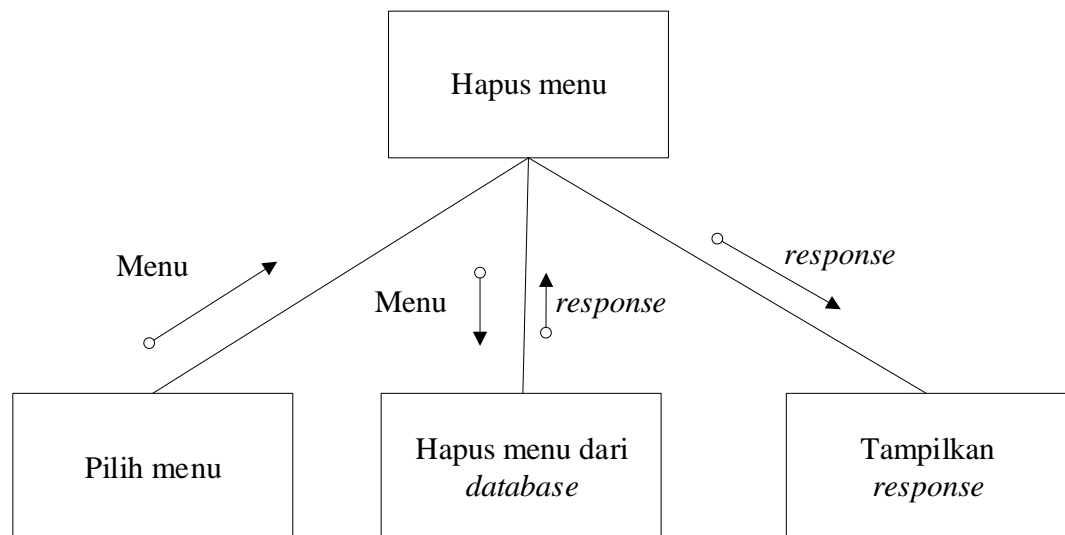
Gambar 4.22
Bagan Terstruktur Mencetak Laporan Kunjungan
Sumber : Dokumen Pribadi



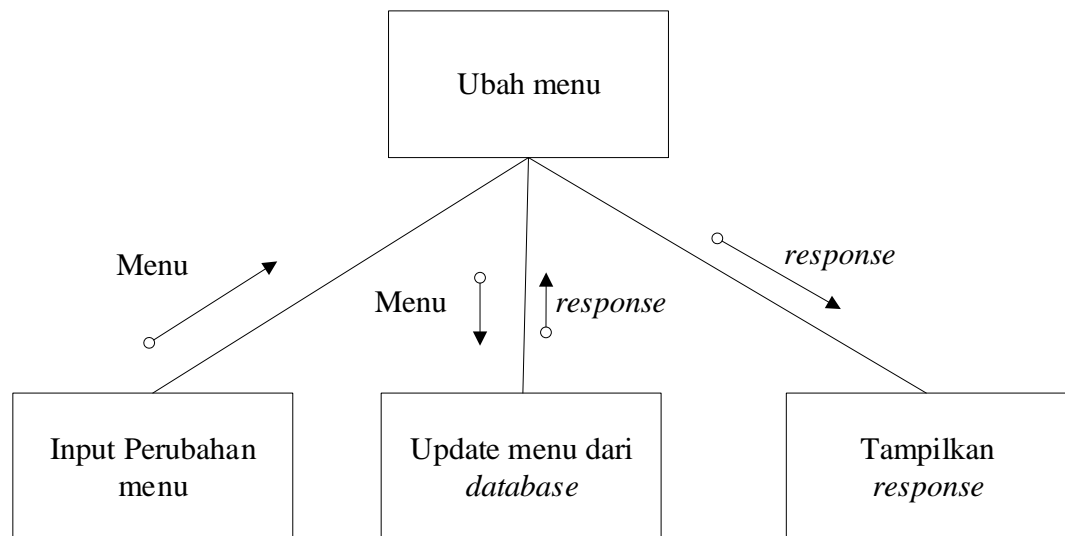
Gambar 4.23
Bagan Terstruktur Menampilkan Daftar Menu
Sumber : Dokumen Pribadi



Gambar 4.24
Bagan Terstruktur Tambah Menu
Sumber : Dokumen Pribadi



Gambar 4.25
Bagan Terstruktur Hapus Menu
Sumber : Dokumen Pribadi



Gambar 4.26
Bagan Terstruktur Ubah Menu
Sumber : Dokumen Pribadi

Q. Spesifikasi Modul yang Diusulkan

1. Modul pemesanan

Tampilkan daftar menu

Ambil data menu yang dipilih

Ambil masukan jumlah dan level menu yang dipilih

Buat data pesanan dari menu yang dipilih, jumlah, dan level

Mengubah data pesanan menjadi format *JSON*

Kirim data pesanan dalam format *JSON* tersebut ke *server*

Konfirmasi data pesanan

2. Modul *request bill*

Kirim *request* ke *server*

Buat transaksi berdasarkan nomor meja

Ambil data pesanan berdasarkan nomor meja

Hitung total pembayaran

Membuat *bill* dari data transaksi, data pesanan, dan total pembayaran

Cetak *bill*

3. Modul pembayaran

Ambil masukan nominal uang pembayaran

Ambil data transaksi berdasarkan nomor meja

Ambil data total pembayaran

Hitung kembalian dari uang kembalian dikurang total pembayaran

Membuat bukti pembayaran dari data transaksi, data pesanan, total pembayaran, uang pembayaran, dan kembalian

Cetak bukti pembayaran

R. Rancangan Basis Data Sistem yang Diusulkan

1. Normalisasi

a. Bentuk tidak normal (*Unnormalized*)

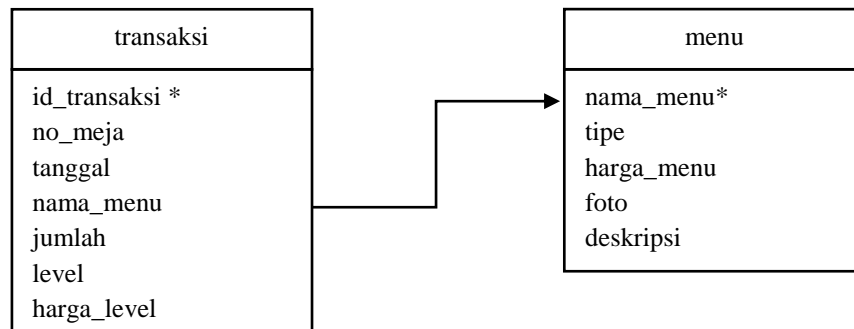
nama_menu
tipe
harga_menu
foto
deskripsi
level
harga_level
id_transaksi
no_meja
tanggal
id_pesanan
jumlah

Gambar 4.27

Bentuk Tidak Normal

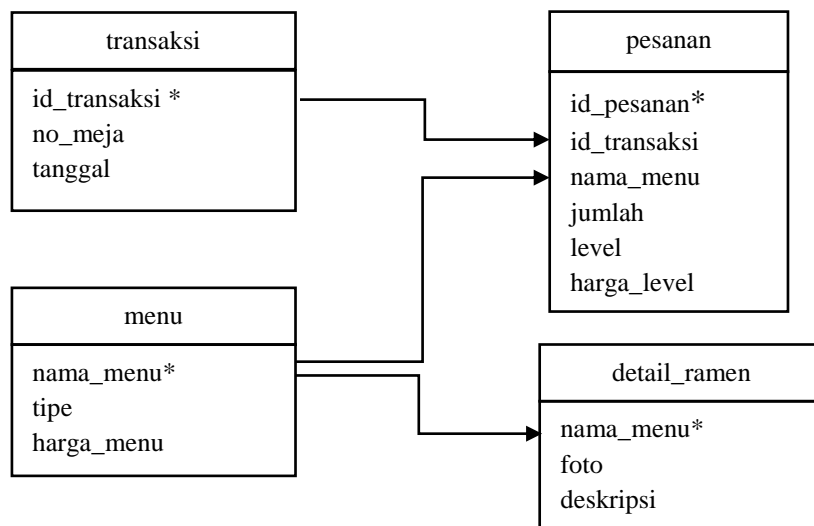
Sumber : Dokumen Pribadi

b. Normalisasi pertama (*First Normal Form*)



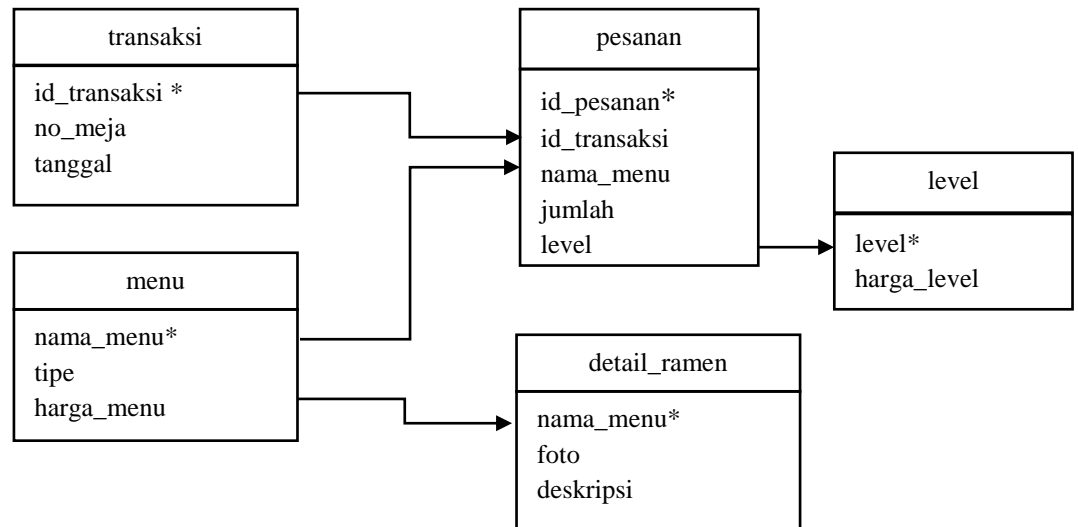
Gambar 4.28
Normalisasi Pertama
Sumber : Dokumen Pribadi

c. Normalisasi kedua (*Second Normal Form*)



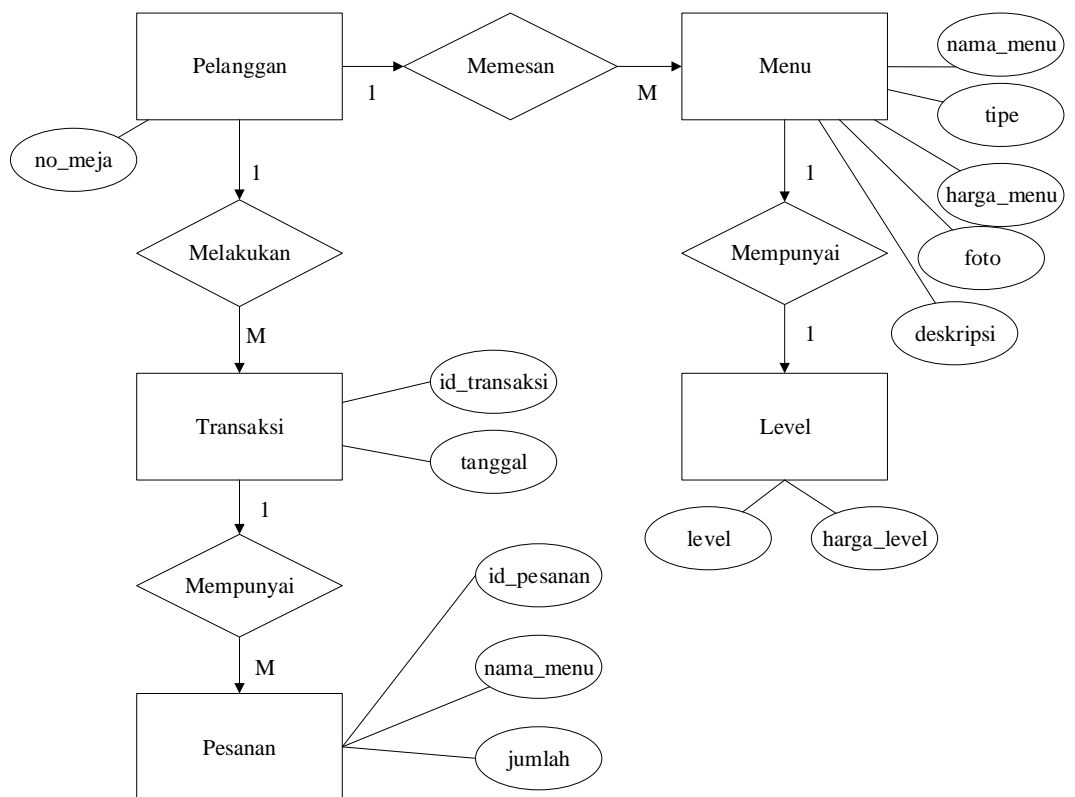
Gambar 4.29
Normalisasi Kedua
Sumber : Dokumen Pribadi

d. Normalisasi ketiga (*Third Normal Form*)



Gambar 4.30
Normalisasi Ketiga
Sumber : Dokumen Pribadi

2. ERD (*Entity Relationship Diagram*)



Gambar 4.31
Diagram ERD
Sumber : Dokumen Pribadi

3. Spesifikasi *File*

- a. Nama *file* : menu
- Media : *Harddisk*
- Primary key* : nama_menu
- Panjang *record* : 41
- Jumlah *record* : 41 x 31 (menu) = 1271
- Struktur :

Tabel 4.1
Spesifikasi *File Menu*

No.	Nama <i>Field</i>	<i>Type</i>	<i>Size</i>	Keterangan
1.	nama_menu	<i>varchar</i>	25	Nama menu
2.	tipe	<i>varchar</i>	10	Tipe menu
3.	harga_menu	<i>Int</i>	6	Harga menu

Sumber : Dokumen Pribadi

- b. Nama *file* : detail_ramen
- Media : *Harddisk*
- Primary key* : nama_menu
- Panjang *record* : 16777390
- Jumlah *record* : 16777390 x 4 (menu) =
67109560
- Struktur :

Tabel 4.2
Spesifikasi *File* Detail Ramen

No.	Nama <i>Field</i>	<i>Type</i>	<i>Size</i>	Keterangan
1.	nama_menu	<i>varchar</i>	25	Nama menu
2.	foto	<i>mediumblob</i>	16777215	Foto menu
3.	deskripsi	<i>varchar</i>	150	Deskripsi menu

Sumber : Dokumen Pribadi

c. Nama *file* : transaksi

Media : *Harddisk*

Primary key : id_transaksi

Panjang *record* : 30

Jumlah *record* : 30 x 100 (pesanan) x 12
(bulan) = 36000

Struktur :

Tabel 4.3
Spesifikasi *File* Transaksi

No.	Nama <i>Field</i>	<i>Type</i>	<i>Size</i>	Keterangan
1.	id_transaksi	<i>varchar</i>	20	Id transaksi
2.	no_meja	<i>varchar</i>	2	Nomor meja
3.	Tanggal	<i>datetime</i>	8	Tanggal transaksi

Sumber : Dokumen Pribadi

d. Nama *file* : pesanan

Media : *Harddisk*

Primary key : id_pesanan

Panjang *record* : 62

Jumlah *record* : 62 x 5 (menu) x 100 (pesanan)
x 12 (bulan) = 372000

Struktur :

Tabel 4.4
Spesifikasi *File Pesanan*

No.	Nama <i>Field</i>	<i>Type</i>	<i>Size</i>	Keterangan
1.	id_transaksi	<i>varchar</i>	20	Id transaksi
2.	id_pesanan	<i>varchar</i>	12	Id Pesanan
3.	nama_menu	<i>varchar</i>	25	Nama menu
4.	jumlah	<i>Int</i>	3	Jumlah pesanan
5.	level	<i>varchar</i>	2	Level pesanan

Sumber : Dokumen Pribadi

e. Nama *file* : Level

Media : *Harddisk*

Primary key : level

Panjang *record* : 8

Jumlah *record* : 8 x 11 (level) = 88

Struktur :

Tabel 4.5
Spesifikasi *File Level*

No.	Nama <i>Field</i>	<i>Type</i>	<i>Size</i>	Keterangan
1.	level	<i>varchar</i>	2	Level pesanan
2.	harga_level	<i>Int</i>	6	Harga level

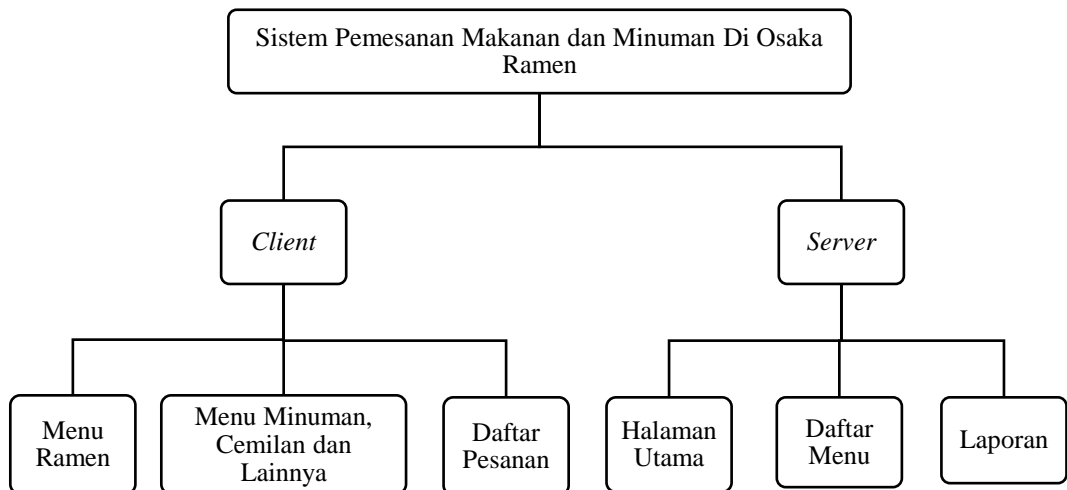
Sumber : Dokumen Pribadi

S. Rancangan Layar, Rancangan *Form* Masukan Data, dan Rancangan Keluaran

Rancangan antar muka atau *user interface* adalah rancangan tampilan grafis untuk dilihat pengguna dan dapat dimengerti dan digunakan pengguna

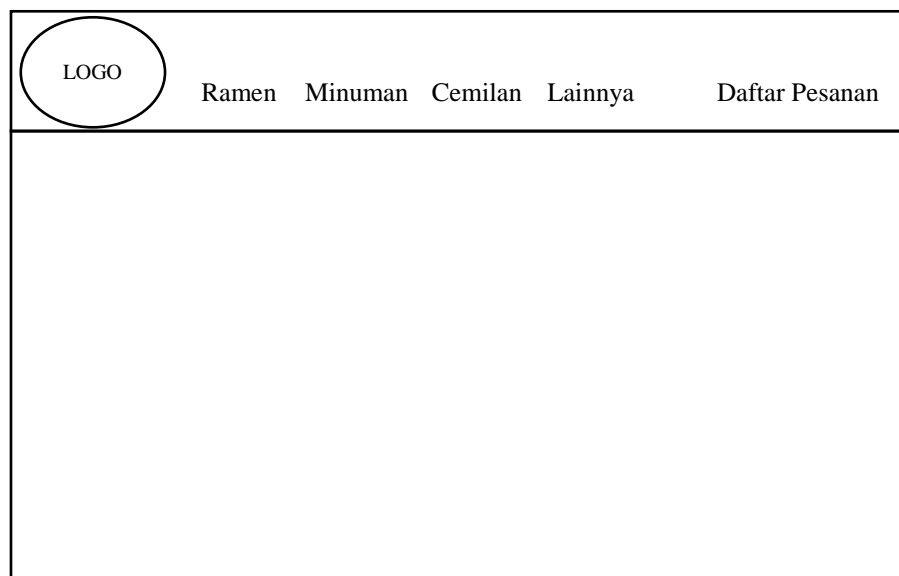
sehingga dapat terjadi adanya komunikasi antara pengguna dengan komputer.

Rancangan tersebut dapat diuraikan sebagai berikut:



Gambar 4.32
Rancangan Antarmuka
Sumber : Dokumen Pribadi

1. Rancangan Tampilan Navigasi



Gambar 4.33
Rancangan Tampilan Navigasi
Sumber : Dokumen Pribadi

Tampilan ini terdapat pada aplikasi *client* yang digunakan pelanggan untuk memesan menu makanan dan minuman Osaka Ramen.

Pada tampilan navigasi terdapat logo Osaka Ramen, empat kategori menu dan daftar pesanan. Tombol kategori-kategori ini digunakan untuk mengarahkan ke daftar menu sesuai kategori yaitu ramen, minuman, cemilan, dan lainnya. Daftar menu akan ditampilkan di bawah navigasi dalam satu jendela (*window*) yang sama. Sedangkan tombol daftar pesanan digunakan untuk memunculkan sebuah *pop up* atau dialog yang berisi daftar pesanan.

2. Rancangan Tampilan Menu Ramen

The diagram illustrates the layout of the Ramen Menu screen. It features a header bar at the top. Below it, the main content area contains a large circle labeled 'Gambar Ramen' (Ramen Image). To the right of the circle, the labels 'Nama Ramen' (Ramen Name), 'Deskripsi' (Description), and 'Harga' (Price) are positioned. At the bottom of the screen, there is a 'Level' dropdown menu, a quantity selector consisting of a '+' button, the number '1', and a '-' button, and a 'Pesanan' (Order) button.

Gambar 4.34
Rancangan Tampilan Menu Ramen
Sumber : Dokumen Pribadi

Tampilan menu ramen dapat ditampilkan jika tombol kategori ramen di navigasi ditekan. Pada tampilan menu ramen terdapat data mengenai ramen seperti gambar ramen, nama ramen, deskripsi mengenai ramen, dan harga. Selain informasi mengenai ramen, juga terdapat sebuah masukan seperti masukan tingkat level kepedasan ramen dan jumlah

ramen yang ingin dipesan, dan terdapat tombol pesan untuk mengeksekusi pesanan tersebut. Pada rancangan di atas, digunakan untuk menampung satu data ramen. Apabila data ramen lebih dari satu, maka rancangan tersebut akan diulang sebanyak jumlah data pada daftar ramen.

3. Rancangan Tampilan Menu Minuman, Cemilan, dan Lainnya

Gambar 4.35
Rancangan Tampilan Menu Minuman, Cemilan, dan Lainnya
Sumber : Dokumen Pribadi

Tampilan ini dapat ditampilkan jika tombol kategori menu selain ramen ditekan. Tampilan menu minuman, cemilan, dan lainnya digunakan untuk menampilkan daftar menu selain ramen yaitu minuman, cemilan, dan lainnya. Pada tampilan ini terdapat informasi mengenai menu yaitu nama menu, dan harga. Selain informasi mengenai menu juga terdapat masukan yaitu jumlah pesanan menu dan tombol pesan untuk mengeksekusi pesanan. Pada rancangan di atas, digunakan untuk

menampung satu data menu. Apabila data menu lebih dari satu, maka rancangan tersebut akan diulang sebanyak jumlah data pada daftar menu.

4. Rancangan Tampilan Daftar Pesanan

Daftar Pesanan

Nama	Jumlah	Harga	Total	Status

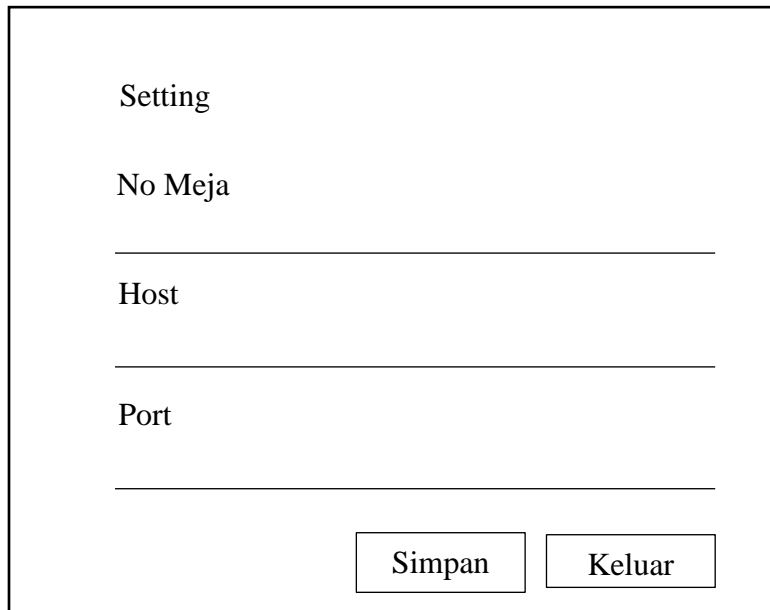
Total harga: Rp.0,00

Pesan Bayar Kembali

Gambar 4.36
Rancangan Tampilan Daftar Pesanan
Sumber : Dokumen Pribadi

Tampilan daftar pesanan akan tampil jika tombol daftar pesanan yang ada di navigasi ditekan. Pada tampilan daftar pesanan terdapat tabel daftar pesanan, total harga pembayaran dan tiga tombol di bawah yaitu tombol pesan, bayar, dan kembali.

5. Rancangan Tampilan *Setting*



Setting

No Meja

Host

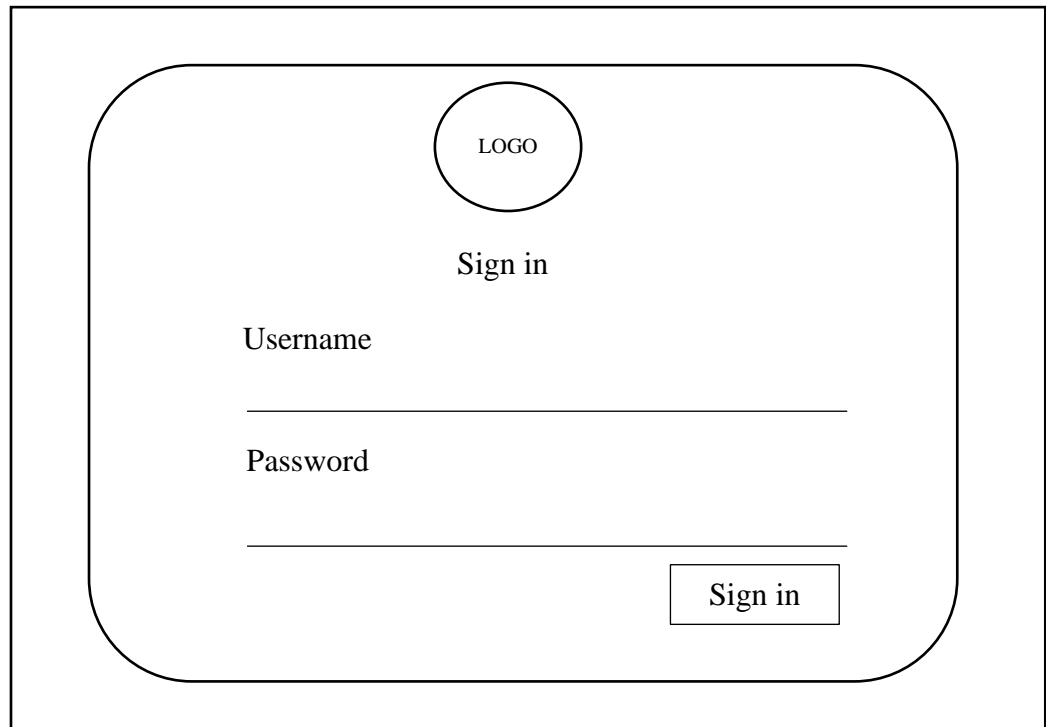
Port

Simpan Keluar

Gambar 4.37
Rancangan Tampilan *Setting*
Sumber : Dokumen Pribadi

Tampilan *setting* akan muncul saat logo pada aplikasi *client* ditekan tiga kali. Hal ini dimaksud agar pelanggan tidak mengetahui cara mengakses tampilan *setting*. Pada tampilan *setting* terdapat konfigurasi untuk mengubah nomor meja, alamat *host server*, dan *port server*.

6. Rancangan Tampilan *Sign In*

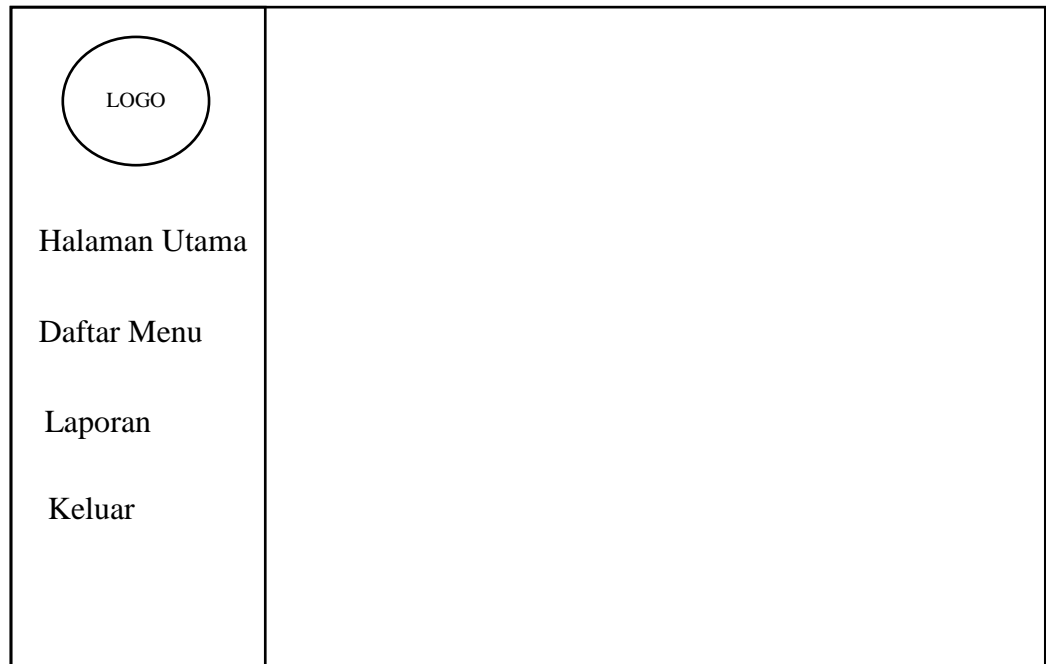


The image shows a wireframe for a 'Sign In' form. It is contained within a rounded rectangle. At the top center is a circle labeled 'LOGO'. Below the logo is the text 'Sign in'. Underneath that are two input fields: the first is labeled 'Username' and the second is labeled 'Password'. At the bottom right of the form is a rectangular button labeled 'Sign in'.

Gambar 4.38
Rancangan Tampilan *Sign in*
Sumber : Dokumen Pribadi

Tampilan *sign in* akan muncul saat program atau aplikasi *server* di jalankan. Pada tampilan *sign in* terdapat logo Osaka Ramen, masukan *username*, *password* dan tombol untuk melakukan *sign in*.

7. Rancangan Tampilan *Side Bar*



Gambar 4.39
Rancangan Tampilan *Side Bar*
Sumber : Dokumen Pribadi

Tampilan ini akan tampil jika pengguna berhasil melakukan *sign in*. Pada tampilan *side bar* terdapat logo, dan 3 menu yaitu halaman utama, daftar menu, dan laporan. Pada tampilan ini juga terdapat tombol untuk keluar dari aplikasi.

8. Rancangan Tampilan Halaman Utama

Pesanan Masuk					Pembayaran				
No meja	Nama	Jumlah	Terima	Tolak	No meja	Total harga	Bill	Struk	Simpan

Gambar 4.40
Rancangan Tampilan Halaman Utama
Sumber : Dokumen Pribadi

Tampilan halaman utama dapat diakses pada awal tampilan atau saat tombol halaman utama pada *side bar* ditekan. Pada tampilan ini terdapat dua tabel yaitu tabel pesanan masuk untuk memantau pesanan yang masuk dan mengkonfirmasi pesanan tersebut dan terdapat tabel pembayaran untuk memantau permintaan pelanggan untuk melakukan pembayaran.

9. Rancangan Tampilan Daftar Menu

Daftar Menu

Nama	Tipe	Harga

Tambah Menu

Nama

Tipe

▽

Harga

Deskripsi

Gambar

Pilih gambar...

Hapus

Tambah/Ubah

Level

Level	Harga

Ubah Level

Level

Harga

Ubah

Gambar 4.41
Rancangan Tampilan Daftar Menu
Sumber : Dokumen Pribadi

Tampilan daftar menu dapat ditampilkan saat tombol daftar menu pada *side bar* di tekan. Pada tampilan daftar menu terdapat tabel daftar menu dan *form* menu untuk mengelola data menu seperti menambah, menghapus, dan mengubah data menu.

10. Rancangan Tampilan Laporan

Dari _____ Sampai _____

Semua ▽

Cetak

Laporan Pemesanan

Pukul	No Meja	Nama	Jumlah	Harga	Total Harga

Menu Favorit

Diagram
Pie

Laporan Pemasukan

Diagram Garis

Laporan Kunjungan

Diagram Garis

Gambar 4.42
Rancangan Tampilan Laporan
Sumber : Dokumen Pribadi

Tampilan laporan dapat ditampilkan saat tombol laporan pada *side bar* di tekan. Tampilan laporan terdapat empat bagian yaitu laporan pemesanan, laporan menu favorit, laporan pemasukan, dan laporan kunjungan.

T. Tampilan dan Penjelasan Layar, Tampilan Format Masukan, dan Tampilan Keluaran

1. Tampilan Layar

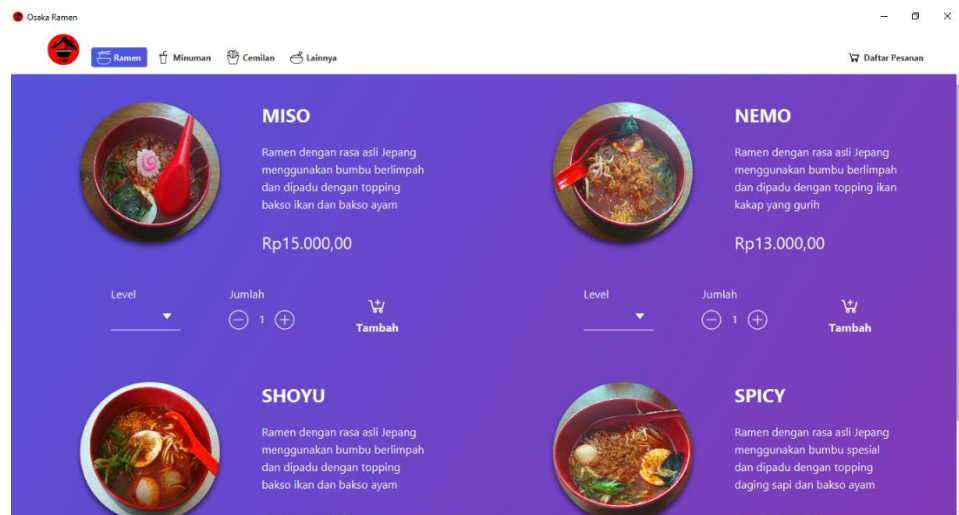
a. Tampilan Navigasi



Gambar 4.43
Tampilan Navigasi
Sumber : Dokumen Pribadi

Pada gambar di atas adalah tampilan navigasi pada aplikasi *client*. Navigasi digunakan sebagai petunjuk arah untuk menuju ke sebuah tampilan seperti tampilan daftar ramen, daftar minuman, daftar cemilan, daftar lainnya, dan daftar pesanan.

b. Tampilan Daftar Ramen

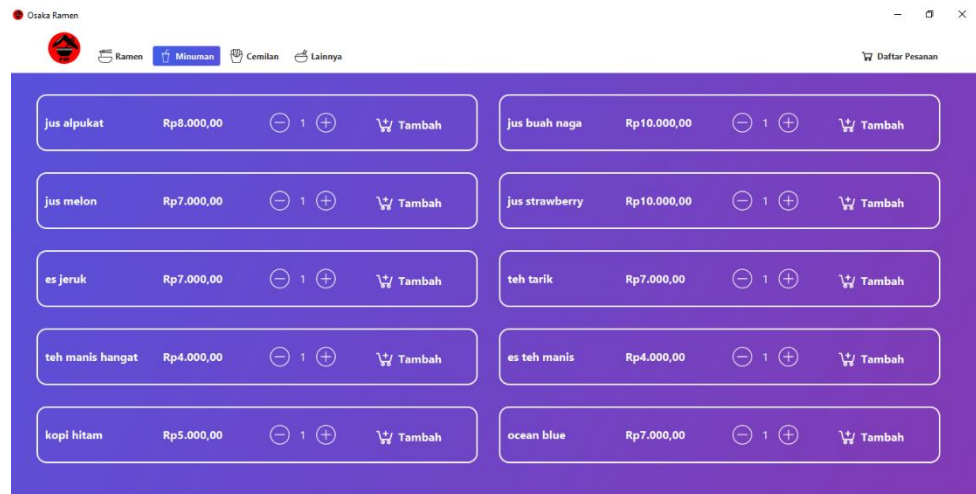


Gambar 4.44
Tampilan Daftar Ramen
Sumber : Dokumen Pribadi

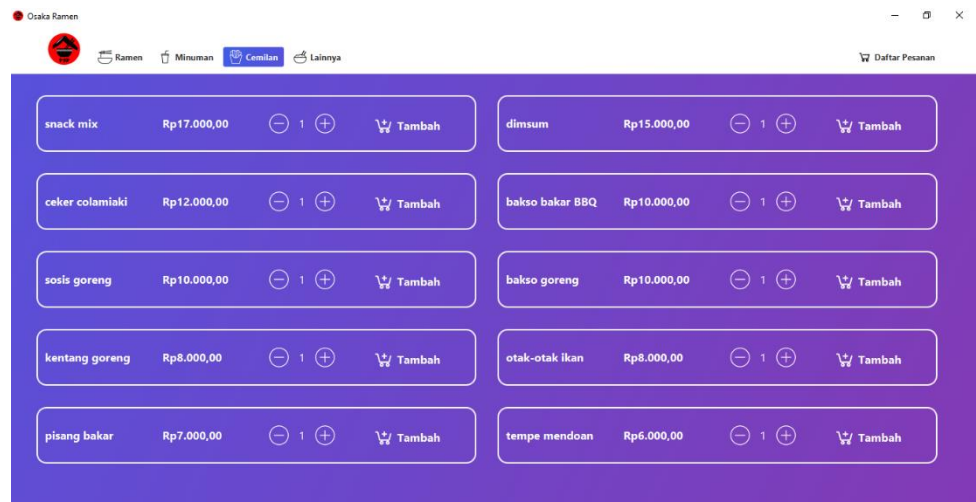
Pada gambar di atas merupakan daftar menu ramen yang ada di Osaka Ramen. Untuk melakukan pesanan, pelanggan dapat memilih

level kepedasan, menentukan jumlah pesanan yang ada di setiap bawah menu. Setelah menentukan level dan jumlah, pelanggan dapat menekan tombol tambah. Pesanan akan disimpan ke dalam daftar pesanan.

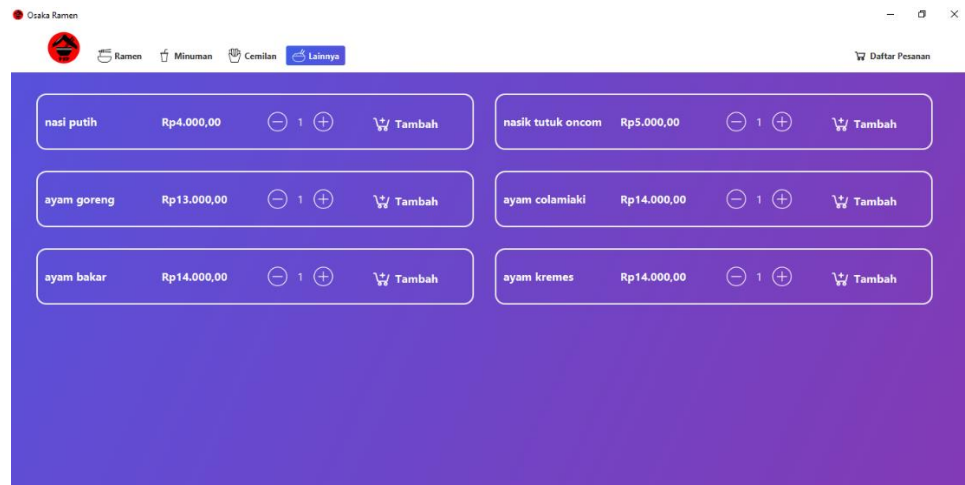
c. Tampilan Daftar Menu Minuman, Cemilan, dan Lainnya



Gambar 4.45
Tampilan Daftar Minuman
Sumber : Dokumen Pribadi



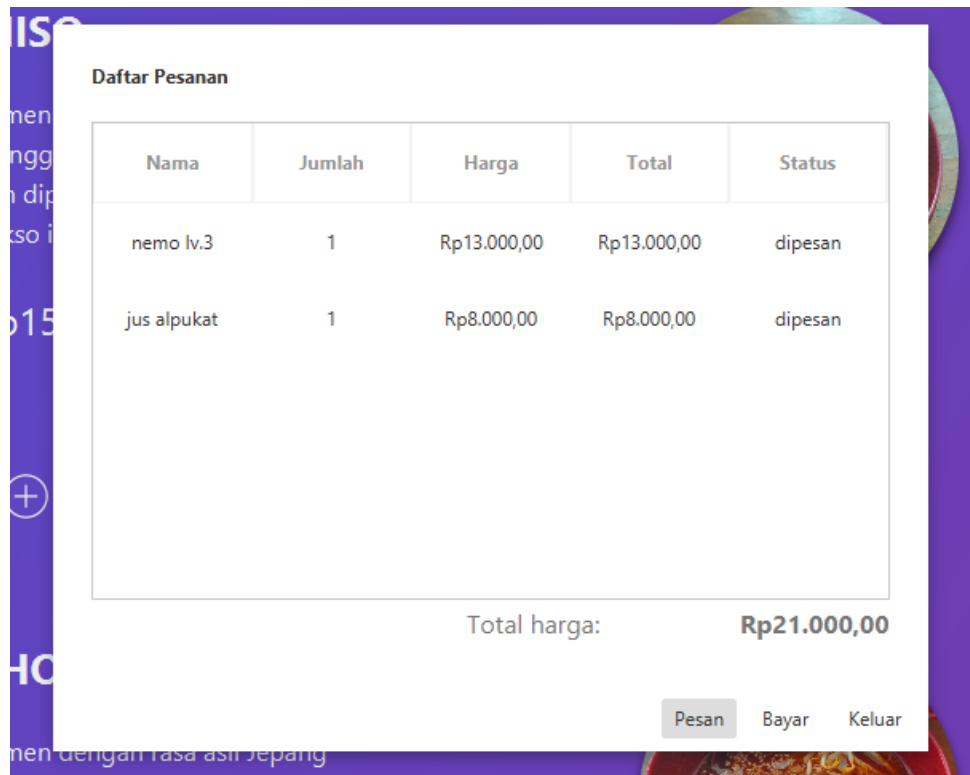
Gambar 4.46
Tampilan Daftar Cemilan
Sumber : Dokumen Pribadi



Gambar 4.47
Tampilan Daftar Lainnya
Sumber : Dokumen Pribadi

Pada tiga gambar di atas merupakan tampilan daftar menu minuman, cemilan, dan lainnya. Untuk melakukan pesanan, pelanggan hanya tinggal menentukan jumlah pesanan dan menekan tombol tambah. Pesanan akan disimpan di dalam daftar pesanan.

d. Tampilan Daftar Pesanan



Daftar Pesanan

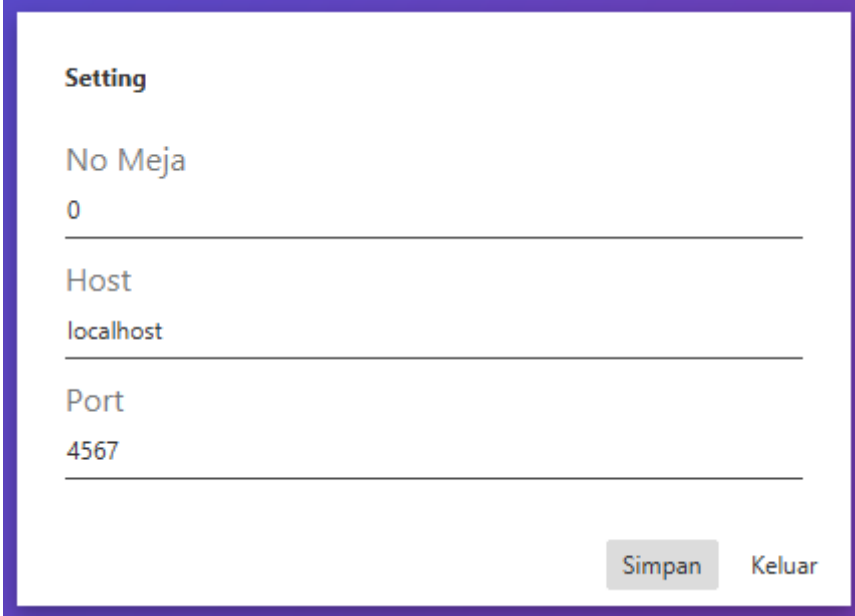
Nama	Jumlah	Harga	Total	Status
nemo lv.3	1	Rp13.000,00	Rp13.000,00	dipesan
jus alpukat	1	Rp8.000,00	Rp8.000,00	dipesan

Total harga: **Rp21.000,00**

Pesan Bayar Keluar

Gambar 4.48
Tampilan Daftar Pesanan
Sumber : Dokumen Pribadi

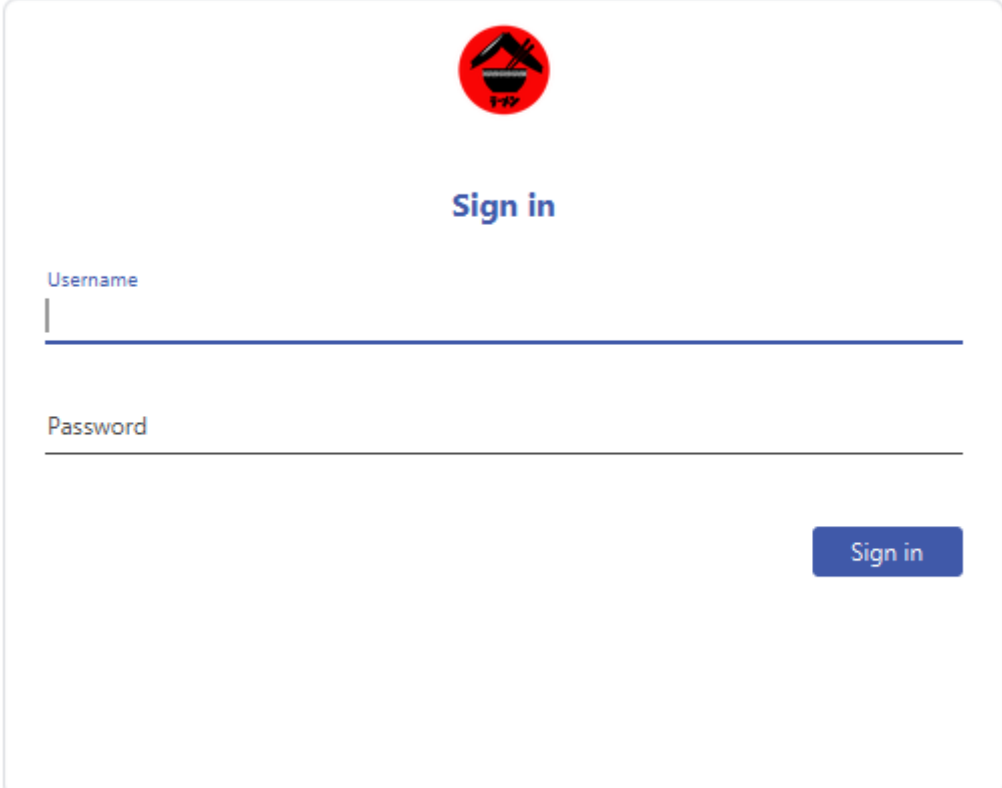
Gambar di atas merupakan tampilan daftar pesanan. Daftar pesanan berfungsi untuk menampilkan daftar pesanan yang sudah ditambahkan. Untuk melakukan pemesanan, pelanggan dapat menekan tombol pesan, kemudian akan muncul dialog konfirmasi untuk memastikan bahwa data pesanan sudah benar. Apabila konfirmasi diterima, pesanan akan dikirim ke komputer *server* dan status pesanan akan berubah menjadi dipesan. Untuk melakukan pembayaran dapat menekan tombol bayar. Pembayaran dapat dilakukan setelah semua pesanan diproses di dapur. Untuk keluar dari daftar pesanan, pelanggan dapat menekan tombol keluar.

e. Tampilan *Setting*

The image shows a web-based 'Setting' form. It has a title 'Setting' at the top left. Below the title are three input fields: 'No Meja' with the value '0', 'Host' with the value 'localhost', and 'Port' with the value '4567'. At the bottom right of the form are two buttons: 'Simpan' (Save) and 'Keluar' (Exit).

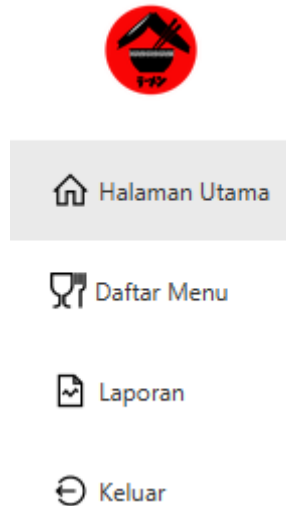
Gambar 4.49
Tampilan *Setting*
Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan dari *form setting* untuk melakukan konfigurasi seperti mengatur nomor meja, mengatur alamat *server*, dan *port server*. Pengaturan tersebut akan disimpan ke sebuah *file* berformat JSON.

f. Tampilan *Sign In*The image shows a 'Sign in' form within a light gray rounded rectangle. At the top center is a red circular logo containing a black silhouette of a ramen bowl with chopsticks. Below the logo, the text 'Sign in' is displayed in a bold, blue font. Underneath, there are two input fields: the first is labeled 'Username' in blue text and has a blue underline; the second is labeled 'Password' in gray text and has a gray underline. In the bottom right corner of the form, there is a blue rectangular button with the text 'Sign in' in white.

Gambar 4.50
Tampilan *Sign in*
Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan dari *form sign in* untuk pegawai Osaka Ramen dan pemilik Osaka Ramen masuk ke aplikasi *server*. Pada tampilan terdapat dua buah *text field* untuk memasukkan *username* dan *password*. *Username* dan *password* tersebut akan diperiksa di *database* apakah data yang dimasukkan sesuai dengan data yang ada di *database*.

g. Tampilan *Side Bar*

Gambar 4.51
Tampilan *Side Bar*
Sumber : Dokumen Pribadi

Pada gambar di atas merupakan *side bar* dari aplikasi *server*. Menu pada *side bar* digunakan untuk mengarahkan pengguna untuk menuju ke tampilan yang diinginkan seperti halaman utama, daftar menu, laporan, dan tombol keluar untuk kembali ke *form sign in*.

h. Tampilan Halaman Utama

Pesanan Masuk					Pembayaran				
No Meja	Nama	Jumlah	Terima	Tolak	No Meja	Total Harga	Bill	Struk	Simpan
1	miso lv3	1	☑ Terima	🗑 Tolak	1	Rp29.000,00	🖨 Cetak	🖨 Cetak	💾 Simpan
1	jeruk hangat	2	☑ Terima	🗑 Tolak					

Gambar 4.52
Tampilan Halaman Utama
Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan dari halaman utama. Pada halaman utama terdapat dua fitur yaitu *monitoring* pesanan masuk dan *monitoring* pembayaran. Apabila terdapat pesanan masuk maka akan

muncul pada tabel pesanan masuk yang berisi data pesanan seperti nomor meja, nama pesanan, jumlah pesanan. Kemudian data tersebut dapat dikonfirmasi apakah pesanan dapat diterima karena stok di dapur tersedia atau dapat ditolak karena tidak ada stok atau alasan lain. Pada tabel pembayaran, data pembayaran atau transaksi akan muncul apabila ada pelanggan yang melakukan permintaan untuk melakukan pembayaran. Data pada tabel pembayaran berupa nomor meja, total pembayaran. Pelayan atau kasir dapat melakukan pencetakan *bill* untuk diantarkan ke meja pelanggan. Setelah pelanggan membayar, pelayan atau kasir dapat memasukkan nominal uang pembayaran dengan menekan terlebih dahulu tombol cetak struk. Setelah transaksi selesai, transaksi dapat disimpan ke *database* dengan menekan tombol simpan.

2. Tampilan Masukan

a. Tampilan Daftar Menu

Daftar Menu

Nama	Tipe	Harga
ayam bakar	lainnya	Rp14.000,00
ayam colamiaki	lainnya	Rp14.000,00
ayam goreng	lainnya	Rp13.000,00
ayam kremes	lainnya	Rp14.000,00
bakso bakar BBQ	cemilan	Rp10.000,00

Tambah Menu

Tipe

Pilih tipe menu...

Harga

Deskripsi

Gambar

Pilih gambar... (max : 2048 KB)

Tambah

Level

Level	Harga
0	Rp0,00
1	Rp0,00
10	Rp2.000,00
2	Rp0,00

Ubah Level

Level

Harga

Ubah

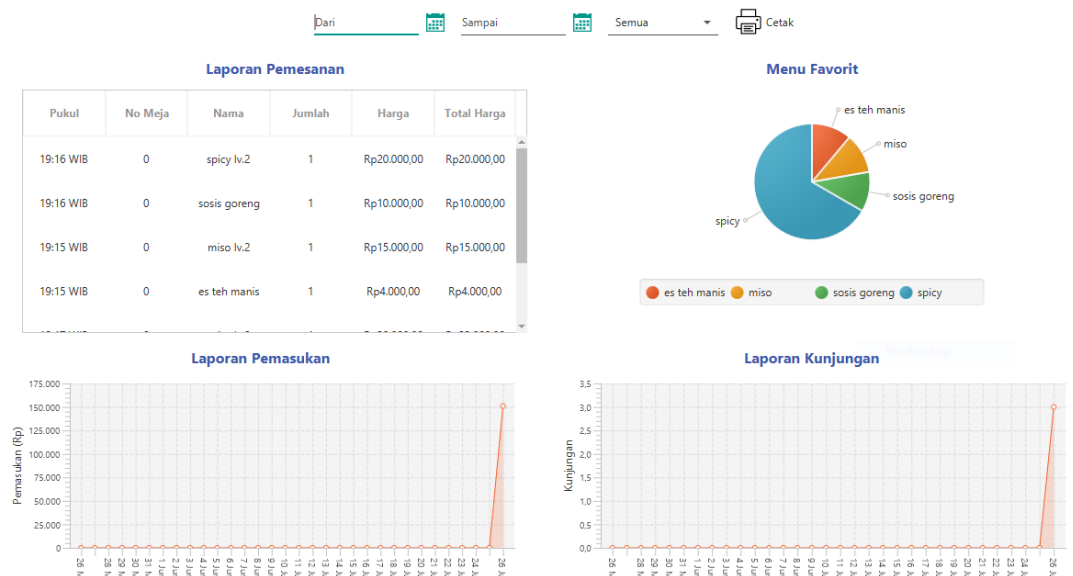
Gambar 4.53
Tampilan Daftar Menu

Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan untuk mengelola daftar menu. Tampilan ini hanya dapat diakses oleh pemilik Osaka Ramen. Tampilan ini digunakan untuk mengelola data menu seperti menambah data menu, mengubah data menu, dan menghapus data menu. Selain mengelola daftar menu, juga terdapat fitur untuk mengelola harga level.

3. Tampilan Keluaran

a. Tampilan Laporan



Gambar 4.54
Tampilan Laporan
Sumber : Dokumen Pribadi

Gambar di atas merupakan tampilan laporan-laporan yaitu laporan pemesanan, laporan menu favorit, laporan pemasukan, dan laporan kunjungan. Tampilan ini hanya dapat diakses oleh pemilik Osaka Ramen. Setiap laporan terdapat tombol untuk menampilkan laporan yang lebih rinci dalam bentuk pdf.

b. Tampilan Laporan Pemesanan

Osaka Ramen
Laporan Pemesanan

Pukul	No Meja	Nama Menu	Jumlah	Harga	Total Harga
8:3 WIB	1	bakso goreng	1	Rp10.000,00	Rp10.000,00
8:3 WIB	1	nasi putih	1	Rp4.000,00	Rp4.000,00
8:3 WIB	1	teh tarik	1	Rp7.000,00	Rp7.000,00

Depok, Senin, 10 Juni 2019

Pemilik
Taufiq

Gambar 4.55
Tampilan Laporan Pemesanan
Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan laporan pemesanan di dalam *file* pdf. Laporan ini dirincikan menampilkan daftar transaksi pemesanan yang dicatat di dalam sistem.

c. Tampilan Laporan Menu Favorit

Osaka Ramen
Laporan Menu Favorit

Nama Menu	Tipe	Harga	Total Dipesan
bakso goreng	cemilan	Rp10.000,00	1
nasi putih	lainnya	Rp4.000,00	1
teh tarik	minuman	Rp7.000,00	1

Depok, Senin, 10 Juni 2019

Pemilik
Taufiq

Gambar 4.56
Tampilan Laporan Menu Favorit
Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan laporan menu favorit di dalam *file* pdf. Laporan ini dirincikan seperti menampilkan daftar menu dan total menu yang dipesan.

d. Tampilan Laporan Pemasukan



Osaka Ramen
Laporan Pemasukan

Tanggal	Total Pemasukan
2019-06-04	Rp21.000,00
2019-06-05	Rp21.000,00
2019-06-06	Rp21.000,00
2019-06-07	Rp21.000,00
2019-06-08	Rp21.000,00
2019-06-09	Rp21.000,00
2019-06-10	Rp21.000,00

Depok, Senin, 10 Juni 2019

Pemilik
Taufiq

Gambar 4.57
Tampilan Laporan Pemasukan
Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan laporan pemasukan di dalam *file* pdf. Laporan ini dirincikan seperti menampilkan daftar pemasukan setiap hari.

e. Tampilan Laporan Kunjungan

Osaka Ramen
Laporan Kunjungan

Tanggal	Total Kunjungan
2019-06-04	1
2019-06-05	1
2019-06-06	1
2019-06-07	1
2019-06-08	1
2019-06-09	1
2019-06-10	1

Depok, Senin, 10 Juni 2019

Pemilik
Taufiq

Gambar 4.58
Tampilan Laporan Kunjungan
Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan laporan kunjungan di dalam *file* pdf. Laporan ini dirincikan seperti menampilkan daftar kunjungan setiap hari.

BAB V

SIMPULAN DAN SARAN

A. Simpulan

Berdasarkan permasalahan dan pembahasan yang dijelaskan di bab-bab sebelumnya, maka dapat disimpulkan bahwa:

1. Proses pencatatan secara komputerisasi dapat memberikan suatu daya tarik yang berbeda kepada pelanggan, mengurangi risiko kesalahan data pesanan, serta mengurangi pekerjaan pelayan karena proses pencatatan pesanan sudah ditangani oleh komputer.
2. Data-data pesanan yang terkumpul langsung diolah dalam bentuk laporan sehingga pelayan Osaka Ramen tidak perlu menghitung semua rangkap bukti pembayaran secara manual. Terdapat laporan menu favorit yang bermanfaat sebagai bahan untuk pengambil keputusan dalam menjalankan bisnis restoran.
3. Sistem pemesanan dapat diimplementasikan menggunakan jaringan *server client*. Data-data menu yang dipesan pelanggan melalui komputer *client* diubah dalam format *JSON* kemudian dikirim melalui *HTTP method* untuk dikirim ke komputer *server*. Kemudian di komputer *server* data-data tersebut diubah kembali ke bentuk *object* dan ditampilkan ke tabel pesanan masuk.

B. Saran

Berdasarkan hasil dari penelitian yang telah dilakukan, saran yang dapat penulis sampaikan untuk pengembangan sistem pemesanan Osaka Ramen berikutnya antara lain:

1. Sistem dapat dibuat lebih kompleks lagi dengan menambahkan fitur penghitungan profit yang didapat dalam setiap transaksi.
2. Aplikasi *client* yang digunakan dapat diubah menjadi aplikasi berbasis *platform mobile* sehingga lebih praktis dan lebih mudah digunakan pelanggan saat mengoperasikannya.
3. Pemesanan tidak hanya dapat dilakukan secara *offline* di Osaka Ramen, tetapi dapat dilakukan secara *online* dan terdapat layanan *delivery* ke tempat tujuan.

DAFTAR PUSTAKA

- Ahmar, A. S. (2013). *Modifikasi Template CMS Lokomedia Cara Cepat dan Mudah Membuat Website Elegan Secara Gratis*. Yogyakarta: Garudhawaca.
- Andi. (2015). *Membangun Sendiri Sistem Jaringan Komputer*. Yogyakarta: MADCOMS.
- Buana, I. K. S. (2014). *Jago Pemrograman PHP*. Jakarta: Dunia Komputer.
- Coronel, C., Morris, S., & Rob, P. (2013). *Database Systems: Design, Implementation and Management. Management*.
- Enterprise, J. (2015). *Pengenalan Pemrograman Komputer*. Jakarta: PT Elex Media Komputindo.
- Faizal, E., & Irnawati. (2015). *Pemrograman Java Web (JSP, JSTL & SERVLET) tentang Pembuatan Sistem Informasi Klinik Diimplementasikan dengan Netbeans IDE 7.2 dan MySQL*. Yogyakarta: Gava Media.
- Hariyanto, B. (2014). *Esensi-Esensi Bahasa Pemrograman Java: Disertai Lebih Dari 100 Contoh Program*. Bandung: Informatika.
- Juansyah Andi. (2015). PEMBANGUNAN APLIKASI CHILD TRACKER BERBASIS ASSISTED – GLOBAL POSITIONING SYSTEM (A-GPS) DENGAN PLATFORM ANDROID. *Jurnal Ilmiah Komputer Dan Informatika (KOMPUTA)*, 1(1), 1–8. Retrieved from elib.unikom.ac.id/download.php?id=300375
- Ladjamudin, A.-B. (2013). *Analisis dan Desain Sistem Informasi*. Yogyakarta: Graha Ilmu.

- Nofriadi. (2015). *Java Fundamental Dengan Netbeans 8.0.2*. Yogyakarta: DeePublish.
- Nugroho, B. (2014). *Dasar Pemrograman Web PHP-MySQL dengan Dreamweaver*. Yogyakarta: Gava Media.
- Rahman, F. (2015). Aplikasi pemesanan undangan online. *Jurnal Sains Dan Informatika*, 1(2), 78–87.
- Rosa, & Shalahuddin. (2013). *Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek)*. Bandung: Informatika.
- Sofana, I. (2013). *Membangun Jaringan Komputer: Mudah Membuat Jaringan Komputer (Wire & Wireless) Untuk Pengguna Windows dan Linux*. Bandung: Informatika.
- Sujarweni, V. W. (2015). *Sistem Akuntansi*. Yogyakarta: Pustaka Baru Press.
- Sukanto, & Shalahuddin. (2014). *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*. Bandung: Informatika.
- Susanto, A. (2013). *Sistem Informasi Akuntansi*. Bandung: Lingga jaya.
- Sutabri, T. (2012). *Analisis Sistem Informasi*. Yogyakarta: Andi.
- Winarno, E., Zaki, A., & SmitDev Community. (2014). *Pemrograman Web Berbasis HTML5, PHP, & JavaScript*. Jakarta: PT Elex Media Komputindo.
- Yakub. (2012). *Pengantar Sistem Informasi*. Yogyakarta: Graha Ilmu.

DAFTAR RIWAYAT HIDUP PENULIS



A. DATA PRIBADI

Nama : Robby Awaldi
Tempat/Tanggal Lahir : Depok, 17 Juli 1997
NPM : 201543501022
Program Studi : Informatika
Jenis Kelamin : Laki-Laki
Agama : Islam
Kebangsaan : WNI
No. Handphone : 0896-5262-9124
Alamat : Jl. Kembang Beji RT 05 RW 03 No.62B
Kelurahan Beji Kecamatan Beji, Depok, Jawa Barat

B. RIWAYAT PENDIDIKAN

2003 - 2009 : SDN Beji 1 Depok
2009 - 2012 : SMPN 5 Depok
2012 - 2015 : SMK Prisma Depok

Demikian riwayat hidup ini saya buat dengan sebenar – benarnya tanpa adanya pemalsuan data dan untuk dapat dipergunakan sebagaimana mestinya.

Jakarta, 02 Juli 2019

Robby Awaldi

LISTING PROGRAM

A. Client.java

```
1. package com.unindra.restoclient;
2. import com.google.gson.ExclusionStrategy;
3. import com.google.gson.FieldAttributes;
4. import com.google.gson.Gson;
5. import com.google.gson.GsonBuilder;
6. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
7. import com.unindra.restoclient.models.StandardResponse;
8. import com.unindra.restoclient.models.StatusResponse;
9. import org.apache.commons.io.IOUtils;
10. import java.io.*;
11. import java.net.HttpURLConnection;
12. import java.net.URL;
13. import java.nio.charset.StandardCharsets;
14. import static com.unindra.restoclient.models.Setting.setting;
15. public class Client {
16.     public static StandardResponse get(String paramUrl) throws IOException {
17.         URL url = new URL(setting().getBaseUrl() + paramUrl);
18.         HttpURLConnection connection = (HttpURLConnection) url.openConnection();
19.         connection.setRequestMethod("GET");
20.         if (connection.getResponseCode() == HttpURLConnection.HTTP_OK) {
21.             BufferedReader bufferedReader = new BufferedReader(new
                InputStreamReader(connection.getInputStream()));
22.             String inputLine;
23.             StringBuilder stringBuilder = new StringBuilder();
24.             while ((inputLine = bufferedReader.readLine()) != null)
                { stringBuilder.append(inputLine); }
25.             bufferedReader.close();
26.             connection.disconnect();
27.             return gson().fromJson(stringBuilder.toString(), StandardResponse.class);
28.         } else return new StandardResponse(StatusResponse.ERROR);
29.     public static StandardResponse send(String paramUrl, String requestMethod, String
        json){
30.         try {
31.             URL url = new URL(setting().getBaseUrl() + paramUrl);
32.             HttpURLConnection connection = (HttpURLConnection) url.openConnection();
33.             connection.setConnectTimeout(5000);
34.             connection.setRequestProperty("Content-Type", "application/json; charset=UTF-8");
35.             connection.setRequestMethod(requestMethod);
36.             connection.setDoInput(true);
37.             connection.setDoOutput(true);
38.             OutputStream outputStream = connection.getOutputStream();
39.             connection.disconnect();
40.             outputStream.write(json.getBytes(StandardCharsets.UTF_8));
41.             outputStream.close();
42.             InputStream inputStream = new BufferedInputStream(connection.getInputStream());
43.             String result = IOUtils.toString(inputStream, StandardCharsets.UTF_8);
44.             return gson().fromJson(result, StandardResponse.class);
45.         } catch (IOException e) {
46.             return new StandardResponse(StatusResponse.ERROR);
47.         }
48.     public static Gson gson() {
49.         return new GsonBuilder().addSerializationExclusionStrategy(new ExclusionStrategy() {
```

```

50. public boolean shouldSkipField(FieldAttributes fieldAttributes) {return
    fieldAttributes.getDeclaringClass().equals(RecursiveTreeObject.class);}
51. @Override
52. public boolean shouldSkipClass(Class<?> aClass) {
    return false;}
53. }).addDeserializationExclusionStrategy(new ExclusionStrategy() {
54. @Override
55. public boolean shouldSkipField(FieldAttributes fieldAttributes) {
    return fieldAttributes.getDeclaringClass().equals(RecursiveTreeObject.class);}
56. @Override
57. public boolean shouldSkipClass(Class<?> aClass) {
    return false;}
58. }).create();}}

```

B. Main.java

```

1. package com.unindra.restoclient;
2. import javafx.application.Application;
3. import javafx.fxml.FXMLLoader;
4. import javafx.scene.Parent;
5. import javafx.scene.Scene;
6. import javafx.scene.image.Image;
7. import javafx.stage.Stage;
8. public class Main extends Application {
9. @Override
10. public void start(Stage stage) throws Exception {
11. Parent root = FXMLLoader.load(getClass().getResource("/fxml/app.fxml"));
12. stage.setOnCloseRequest(event -> System.exit(0));
13. stage.getIcons().add(new Image("/icons/logo-ramen-bulet-merah-copy20x20.png"));
14. stage.setTitle("Osaka Ramen");
15. stage.setScene(new Scene(root));
16. stage.show();}
17. public static void main(String[] args) {
18. launch(args);}}

```

C. AllMenuController.java

```

1. package com.unindra.restoclient.controllers;
2. import com.jfoenix.controls.JFXButton;
3. import com.unindra.restoclient.Dialog;
4. import com.unindra.restoclient.Rupiah;
5. import com.unindra.restoclient.models.Pesanan;
6. import com.unindra.restoclient.models.Menu;
7. import com.unindra.restoclient.models.StandardResponse;
8. import com.unindra.restoclient.models.StatusResponse;
9. import javafx.scene.control.Label;
10. import javafx.stage.Stage;
11. import java.util.concurrent.atomic.AtomicInteger;
12. import static com.unindra.restoclient.models.Pesanan.getPesananList;
13. public class AllMenuController {
14. public Label namaLabel;
15. public Label hargaLabel;
16. public JFXButton tambahButton;
17. public Label jumlahLabel;
18. private AtomicInteger jumlah = new AtomicInteger(1);
19. void setMenu(Menu menu) {
20. namaLabel.setText(menu.getNama_menu());
21. hargaLabel.setText(Rupiah.rupiah(menu.getHarga_menu()));

```



```

22. tambahButton.setOnAction(event -> {
23. Dialog alert = new Dialog((Stage) tambahButton.getScene().getWindow());
24. if (getPesananList("dibayar").isEmpty()) {
25. Pesanan pesanan = new Pesanan(menu, jumlah.get());
26. StandardResponse standardResponse = pesanan.post();
27. if (standardResponse.getStatus() == StatusResponse.SUCCESS)
28. alert.information("Berhasil", "Pesanan anda disimpan ke daftar pesanan");
29. else alert.information("Gagal", "Pesanan anda gagal diproses");
30. } else alert.information("Gagal", "Proses pembayaran belum selesai");
31. reset();});}
32. private void reset() {
33. jumlah.set(1);
34. jumlahLabel.setText(String.valueOf(jumlah.get()));
35. namaLabel.requestFocus();}
36. public void kurangJmlHandle() {
37. if (jumlah.decrementAndGet() > 0) {
38. jumlahLabel.setText(String.valueOf(jumlah.get()));
39. } else jumlah.incrementAndGet();}
40. public void tambahJmlHandle() {
41. jumlahLabel.setText(String.valueOf(jumlah.incrementAndGet()));} }

```

D. AppController.java

```

1. package com.unindra.restoclient.controllers;
2. import com.jfoenix.controls.JFXButton;
3. import com.unindra.restoclient.Dialog;
4. import com.unindra.restoclient.models.Pesanan;
5. import com.unindra.restoclient.models.Menu;
6. import com.unindra.restoclient.models.Setting;
7. import javafx.application.Platform;
8. import javafx.event.ActionEvent;
9. import javafx.fxml.FXMLLoader;
10. import javafx.fxml.Initializable;
11. import javafx.geometry.Insets;
12. import javafx.scene.Parent;
13. import javafx.scene.control.Label;
14. import javafx.scene.control.ScrollPane;
15. import javafx.scene.input.MouseEvent;
16. import javafx.scene.layout.FlowPane;
17. import javafx.scene.layout.VBox;
18. import javafx.stage.Stage;
19. import java.io.IOException;
20. import java.net.URL;
21. import java.util.List;
22. import java.util.ResourceBundle;
23. import static com.unindra.restoclient.Dialog.getDialogLayout;
24. import static com.unindra.restoclient.models.Pesanan.getPesananList;
25. import static com.unindra.restoclient.models.Setting.setting;
26. public class AppController implements Initializable {
27. public JFXButton ramenButton;
28. public JFXButton minumanButton;
29. public JFXButton cemilanButton;
30. public JFXButton lainnyaButton;
31. public JFXButton pesananButton;
32. public ScrollPane mainPane;
33. private FlowPane ramenPane;
34. private FlowPane minumanPane;

```

```

35. private FlowPane cemilanPane;
36. private FlowPane lainnyaPane;
37. private VBox pesananPane;
38. private Dialog pesananDialog;
39. @Override
40. public void initialize(URL location, ResourceBundle resources) {
41. ramenPane = new FlowPane();
42. minumanPane = new FlowPane();
43. cemilanPane = new FlowPane();
44. lainnyaPane = new FlowPane();
45. setRamenPane(Menu.menus("ramen"));
46. setAllMenuPane(Menu.menus("minuman"), minumanPane);
47. setAllMenuPane(Menu.menus("cemilan"), cemilanPane);
48. setAllMenuPane(Menu.menus("lainnya"), lainnyaPane);
49. mainPane.setContent(ramenPane);
50. try {
51. pesananPane = FXMLLoader.load(getClass().getResource("/fxml/pesanan.fxml"));
52. } catch (IOException e) {
53. e.printStackTrace();
54. Thread thread = new Thread() -> {
55. while (!Thread.interrupted()) {
56. try {
57. Pesanan.updatePesanan();
58. Thread.sleep(1000);
59. } catch (IOException | InterruptedException e) {
60. break;
61. }
61. ramenButton.setDisable(true);
62. minumanButton.setDisable(true);
63. cemilanButton.setDisable(true);
64. lainnyaButton.setDisable(true);
65. pesananButton.setDisable(true);
66. Platform.runLater() ->
67. getDialog().information("Koneksi Terputus", "Buka setting untuk mengubah alamat host
atau port");
68. thread.start();
69. Platform.runLater() -> pesananDialog = getDialog();
70. JFXButton pesanButton = new JFXButton("Pesan");
71. JFXButton bayarButton = new JFXButton("Bayar");
72. JFXButton keluarButton = new JFXButton("Keluar");
73. Platform.runLater() -> pesananDialog.getDialog()
74. .setContent(getDialogLayout(new Label("Daftar Pesanan"),
75. pesananPane, pesanButton, bayarButton, keluarButton));
76. pesanButton.setOnAction(event -> {
77. if (!getPesananList("belum dipesan").isEmpty()) {
78. Dialog dialog = getDialog();
79. dialog.confirmation("Pesanan tidak dapat dibatalkan setelah proses pemesanan
berhasil",
80. e -> {
81. if (Pesanan.pesan()) {
82. dialog.information("Berhasil", "Pesanan anda berhasil! mohon tunggu pesanan
disajikan");
83. bayarButton.setOnAction(event -> {
84. if (getPesananList("diproses").size() == Pesanan.getPesananList().size())
85. if (Pesanan.getPesananList().size() != 0) try {
86. if (Pesanan.bayar())

```

```

87.  getDialog().information("Mohon tunggu","Kasir akan mengantarkan bill ke meja
    anda");
88.  } catch (IOException e) {
89.  e.printStackTrace();}});
90.  keluarButton.setOnAction(event -> pesananDialog.getDialog().hide());}
91.  public void menuHandle(ActionEvent actionEvent) {
92.  ramenButton.getStyleClass().set(2, "ramen");
93.  minumanButton.getStyleClass().set(2, "minuman");
94.  cemilanButton.getStyleClass().set(2, "cemilan");
95.  lainnyaButton.getStyleClass().set(2, "lainnya");
96.  Object source = actionEvent.getSource();
97.  if (ramenButton.equals(source)) {
98.  ramenButton.getStyleClass().set(2, "ramen-pressed");
99.  mainPane.setContent(ramenPane);
100. } else if (minumanButton.equals(source)) {
101. minumanButton.getStyleClass().set(2, "minuman-pressed");
102. mainPane.setContent(minumanPane);
103. } else if (cemilanButton.equals(source)) {
104. cemilanButton.getStyleClass().set(2, "cemilan-pressed");
105. mainPane.setContent(cemilanPane);
106. } else if (lainnyaButton.equals(source)) {
107. lainnyaButton.getStyleClass().set(2, "lainnya-pressed");
108. mainPane.setContent(lainnyaPane);}}
109. public void daftarPesananHandle() {
110. Platform.runLater() -> pesananDialog.getDialog().show();}
111. public void settingHandle(MouseEvent mouseEvent) {
112. if (mouseEvent.getClickCount() == 3) {
113. FXMLLoader fxmllLoader = new FXMLLoader();
114. fxmllLoader.setLocation(getClass().getResource("/fxml/setting.fxml"));
115. JFXButton keluarButton = new JFXButton("Keluar");
116. JFXButton simpanButton = new JFXButton("Simpan");
117. Dialog settingDialog = getDialog();
118. try {
119. settingDialog.getDialog().setContent(
120. getDialogLayout(new
    Label("Setting"),fxmllLoader.load(),simpanButton,keluarButton));
121. } catch (IOException e) {
122. e.printStackTrace();}
123. SettingController settingController = fxmllLoader.getController();
124. simpanButton.setOnAction(event -> {
125. Setting setting = setting();
126. setting.setNo_meja(settingController.mejaField.getText());
127. setting.setHost(settingController.hostField.getText());
128. setting.setPort(settingController.portField.getText());
129. setting.simpan();
130. settingDialog.confirmation("Aplikasi akan dimatikan",
131. e -> System.exit(0));});
132. simpanButton.requestFocus();
133. keluarButton.setOnAction(event -> settingDialog.getDialog().hide());
134. settingDialog.getDialog().show();}}
135. private void setRamenPane(List<Menu> menuList) {
136. ramenPane.setPrefWidth(800);
137. ramenPane.setPrefHeight(500);
138. ramenPane.getStyleClass().add("body-pane");
139. menuList.forEach(menu -> {

```

```

140. try {
141. FXMLLoader fxmLoader = new
142. FXMLLoader(getClass().getResource("/fxml/ramen.fxml"));
143. Parent root = fxmLoader.load();
144. RamenController c = fxmLoader.getController();
145. c.setMenu(menu);
146. ramenPane.getChildren().add(root);
147. } catch (IOException e) {
148. e.printStackTrace(); } }
149. private void setAllMenuPane(List<Menu> menuList, FlowPane allMenuPane) {
150. allMenuPane.setPrefWidth(800);
151. allMenuPane.setPrefHeight(500);
152. allMenuPane.setHgap(30);
153. allMenuPane.setVgap(30);
154. allMenuPane.setPadding(new Insets(30,20,30,36));
155. allMenuPane.getStyleClass().add("body-pane");
156. menuList.forEach(menu -> {
157. try {
158. FXMLLoader fxmLoader = new
159. FXMLLoader(getClass().getResource("/fxml/allmenu.fxml"));
160. Parent root = fxmLoader.load();
161. AllMenuController c = fxmLoader.getController();
162. c.setMenu(menu);
163. allMenuPane.getChildren().add(root);
164. } catch (IOException e) {
165. e.printStackTrace(); } } }
165. private Dialog getDialog() {
166. return new Dialog((Stage) mainPane.getScene().getWindow()); } }

```

E. PesananController.java

```

1. package com.unindra.restoclient.controllers;
2. import com.jfoenix.controls.JFXButton;
3. import com.jfoenix.controls.JFXTreeTableView;
4. import com.jfoenix.controls.RecursiveTreeItem;
5. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
6. import com.unindra.restoclient.Dialog;
7. import com.unindra.restoclient.models.Pesanan;
8. import com.unindra.restoclient.models.StatusResponse;
9. import javafx.application.Platform;
10. import javafx.beans.property.SimpleStringProperty;
11. import javafx.collections.ListChangeListener;
12. import javafx.fxml.Initializable;
13. import javafx.scene.control.Label;
14. import javafx.scene.control.TreeTableCell;
15. import javafx.scene.control.TreeTableColumn;
16. import javafx.scene.control.TreeTableView;
17. import javafx.stage.Stage;
18. import javafx.util.Callback;
19. import java.net.URL;
20. import java.util.ResourceBundle;
21. import static com.unindra.restoclient.Rupiah.rupiah;
22. import static com.unindra.restoclient.models.Menu.menu;
23. public class PesananController implements Initializable {
24. public JFXTreeTableView<Pesanan> pesananTableView;
25. public Label totalLabel;
26. @Override

```

```

27. public void initialize(URL location, ResourceBundle resources) {
28.     TreeTableColumn<Pesanan, String> namaCol = new TreeTableColumn<>("Nama");
29.     TreeTableColumn<Pesanan, Integer> jumlahCol = new
        TreeTableColumn<>("Jumlah");
30.     TreeTableColumn<Pesanan, String> hargaCol = new TreeTableColumn<>("Harga");
31.     TreeTableColumn<Pesanan, String> totalCol = new TreeTableColumn<>("Total");
32.     TreeTableColumn<Pesanan, String> hapusCol = new TreeTableColumn<>("Status");
33.     namaCol.setCellValueFactory(param ->
        menu(param.getValue().getValue()).nama_menuProperty());
34.     jumlahCol.setCellValueFactory(param ->
        param.getValue().getValue().jumlahProperty());
35.     hargaCol.setCellValueFactory(param ->
        menu(param.getValue().getValue()).harga_menuProperty());
36.     totalCol.setCellValueFactory(param -> param.getValue().getValue().totalProperty());
37.     hapusCol.setCellValueFactory(param -> new SimpleStringProperty(""));
38.     namaCol.setCellFactory(new Callback<TreeTableColumn<Pesanan, String>,
        TreeTableCell<Pesanan, String>>() {
39.         @Override
40.         public TreeTableCell<Pesanan, String> call(TreeTableColumn<Pesanan, String>
            param) {
41.             return new TreeTableCell<Pesanan, String>() {
42.                 @Override
43.                 protected void updateItem(String item, boolean empty) {
44.                     super.updateItem(item, empty);
45.                     if (item == null) {
46.                         setText(null);
47.                     } else {
48.                         Pesanan i = Pesanan.getPesananList().get(getIndex());
49.                         if (menu(i).getTipe().equals("ramen"))
50.                             setText(item + " lv." + i.getLevel());
51.                         else setText(item);
52.                     }
53.                 }
54.             };
55.         });
56.     hapusCol.setCellFactory(new Callback<TreeTableColumn<Pesanan, String>,
        TreeTableCell<Pesanan, String>>() {
57.         @Override
58.         public TreeTableCell<Pesanan, String> call(TreeTableColumn<Pesanan, String>
            param) {
59.             return new TreeTableCell<Pesanan, String>() {
60.                 final JFXButton button = new JFXButton("hapus");
61.                 @Override
62.                 protected void updateItem(String item, boolean empty) {
63.                     super.updateItem(item, empty);
64.                     if (item == null) {
65.                         setText(null);
66.                         setGraphic(null);
67.                     } else {
68.                         Pesanan thisPesanan = Pesanan.getPesananList().get(getIndex());
69.                         button.setFocusTraversable(false);
70.                         button.getStyleClass().add("hapus");
71.                         button.setOnAction(event -> {
72.                             Dialog alert = new Dialog((Stage) pesananTableView.getScene().getWindow());
73.                             alert.confirmation(
                                "Anda yakin ingin menghapus pesanan ini?",
                                e -> {
                                    if (thisPesanan.delete().getStatus() == StatusResponse.SUCCESS)
                                        alert.getDialog().hide();
                                });
                            });
                        });
                    }
                }
            };
        });
    }
}

```

```

74. if (!thisPesanan.getStatus_item().equals("belum dipesan")) {
75.     setText(thisPesanan.getStatus_item());
76.     setGraphic(null);
77. } else {
78.     setText(null);
79.     setGraphic(button);}}}}}});
80. Pesanan.getPesananList().addListener((ChangeListener<Pesanan>) c ->
81. Platform.runLater(() -> totalLabel.setText(rupiah(Pesanan.getGrandTotal()))));
82. pesananTableView.setRoot(new RecursiveTreeItem<>(Pesanan.getPesananList(),
RecursiveTreeObject::getChildren));
83. pesananTableView.getColumns().add(namaCol);
84. pesananTableView.getColumns().add(jumlahCol);
85. pesananTableView.getColumns().add(hargaCol);
86. pesananTableView.getColumns().add(totalCol);
87. pesananTableView.getColumns().add(hapusCol);
88. pesananTableView.setColumnResizePolicy(TreeTableView.CONSTRAINED_RESIZE
_POLICY);}}

```

F. RamenController.java

```

1. package com.unindra.restoclient.controllers;
2. import com.jfoenix.controls.JFXButton;
3. import com.jfoenix.controls.JFXComboBox;
4. import com.unindra.restoclient.Dialog;
5. import com.unindra.restoclient.models.*;
6. import javafx.collections.FXCollections;
7. import javafx.scene.control.Label;
8. import javafx.scene.control.ListCell;
9. import javafx.scene.control.ListView;
10. import javafx.scene.image.Image;
11. import javafx.scene.layout.VBox;
12. import javafx.scene.paint.ImagePattern;
13. import javafx.scene.shape.Circle;
14. import javafx.stage.Stage;
15. import javafx.util.Callback;
16. import java.io.ByteArrayInputStream;
17. import java.io.IOException;
18. import java.util.concurrent.atomic.AtomicInteger;
19. import static com.unindra.restoclient.Rupiah.rupiah;
20. import static com.unindra.restoclient.models.Pesanan.getPesananList;
21. public class RamenController {
22.     public VBox rootPane;
23.     public JFXComboBox<String> levelCombo;
24.     public Label namaLabel;
25.     public Label keteranganLabel;
26.     public Label hargaLabel;
27.     public Circle circle;
28.     public JFXButton tambahButton;
29.     public Label jumlahLabel;
30.     private AtomicInteger jumlah = new AtomicInteger(1);
31.     void setMenu(Menu menu) {
32.         namaLabel.setText(menu.getNama_menu().toUpperCase());
33.         hargaLabel.setText(rupiah(menu.getHarga_menu()));
34.     try {
35.         DetailRamen detailRamen = DetailRamen.detailRamen(menu.getNama_menu());
36.         if (detailRamen != null) {
37.             keteranganLabel.setText(detailRamen.getDeskripsi());

```

```

38. Image image = new Image(new ByteArrayInputStream(detailRamen.getFoto()));
39. circle.setFill(new ImagePattern(image));}
40. } catch (IOException e) {
41. e.printStackTrace();}
42. levelCombo.setItems(
43. FXCollections.observableArrayList("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10"));
44. levelCombo.setCellFactory(new Callback<ListView<String>, ListCell<String>>() {
45. @Override
46. public ListCell<String> call(ListView<String> param) {
47. return new ListCell<String>() {
48. @Override
49. protected void updateItem(String item, boolean empty) {
50. super.updateItem(item, empty);
51. if (item == null || empty) {
52. setGraphic(null);
53. setText(null);
54. } else {
55. try {
56. if (Level.level(Integer.parseInt(item)).getHarga_level() > 0) {
57. setText(item+" ("+"rupiah(Level.level(Integer.parseInt(item)).getHarga_level())+"");
58. } else {
59. setText(item);} } catch (IOException e) {e.printStackTrace();}}}}});
60. tambahButton.setOnAction(event -> {
61. Dialog alert = new Dialog((Stage) rootPane.getScene().getWindow());
62. if (getPesananList("dibayar").isEmpty()) {
63. if (!levelCombo.getSelectionModel().isEmpty()) {
64. Pesanan pesanan = new Pesanan(menu, jumlah.get(),
65. Integer.parseInt(levelCombo.getValue()));
66. StandardResponse standardResponse = pesanan.post();
67. if (standardResponse.getStatus() == StatusResponse.SUCCESS)
68. alert.information("Berhasil", "Pesanan anda disimpan ke daftar pesanan");
69. else alert.information("Gagal", "Pesanan anda gagal diproses");
70. } else alert.information("Gagal", "Level belum dimasukkan");
71. } else alert.information("Gagal", "Proses pembayaran belum selesai");
72. reset();});}
73. private void reset() {
74. jumlah.set(1);
75. jumlahLabel.setText(String.valueOf(jumlah.get()));
76. levelCombo.getSelectionModel().clearSelection();
77. rootPane.requestFocus();}
78. public void kurangJmlHandle() {
79. if (jumlah.decrementAndGet() > 0) {
80. jumlahLabel.setText(String.valueOf(jumlah.get()));
81. } else jumlah.incrementAndGet();}
82. public void tambahJmlHandle() {
83. jumlahLabel.setText(String.valueOf(jumlah.incrementAndGet()));}

```

G. DetailRamen.java

```

1. package com.unindra.restoclient.models;
2. import java.io.IOException;
3. import java.util.Arrays;
4. import static com.unindra.restoclient.Client.get;
5. import static com.unindra.restoclient.Client.gson;
6. public class DetailRamen {
7. private String nama_menu;
8. private byte[] foto;

```

```

9. private String deskripsi;
10. public DetailRamen(String nama_menu, byte[] foto, String deskripsi) {
11. this.nama_menu = nama_menu;
12. this.foto = foto;
13. this.deskripsi = deskripsi;}
14. public static DetailRamen detailRamen(String nama_menu) throws IOException {
15. StandardResponse standardResponse = get("/detail_ramen/"+nama_menu);
16. if (standardResponse.getStatus() == StatusResponse.SUCCESS) {
17. return gson().fromJson(standardResponse.getData(), DetailRamen.class);}
18. return null;}
19. public byte[] getFoto() {
20. return foto;}
21. public String getDeskripsi() {
22. return deskripsi;}
23. @Override
24. public String toString() {
25. return "DetailRamen{" + "nama_menu=" + nama_menu + "\" +", foto=" +
    Arrays.toString(foto) +", deskripsi=" + deskripsi + "\" +'}';}}

```

H. Level.java

```

1. package com.unindra.restoclient.models;
2. import javafx.collections.FXCollections;
3. import java.io.IOException;
4. import java.util.List;
5. import static com.unindra.restoclient.Client.get;
6. import static com.unindra.restoclient.Client.gson;
7. public class Level {
8. private int level;
9. private int harga_level;
10. private static final String paramUrl = "/levels";
11. private Level(int level, int harga_level) {
12. this.level = level;
13. this.harga_level = harga_level;}
14. private static List<Level> levelList() throws IOException {
15. StandardResponse standardResponse = get(paramUrl);
16. if (standardResponse.getStatus() == StatusResponse.SUCCESS) {
17. Level[] levels = gson().fromJson(standardResponse.getData(), Level[].class);}
18. return FXCollections.observableArrayList(levels);}
19. return FXCollections.observableArrayList();}
20. public static Level level(int level) throws IOException {
21. return levelList().stream().filter(l -> l.level == level).findFirst().orElse(null);}
22. public int getHarga_level() {
23. return harga_level;}
24. @Override
25. public String toString() {
26. return "Level{" + "level=" + level +", harga_level=" + harga_level +'}';}}

```

I. Menu.java

```

1. package com.unindra.restoclient.models;
2. import javafx.beans.property.SimpleStringProperty;
3. import javafx.beans.property.StringProperty;
4. import javafx.collections.FXCollections;
5. import java.io.IOException;
6. import java.util.List;
7. import java.util.stream.Collectors;
8. import static com.unindra.restoclient.Client.get;

```



```

9. import static com.unindra.restoclient.Client.gson;
10. import static com.unindra.restoclient.Rupiah.rupiah;
11. public class Menu {
12. private String nama_menu;
13. private int harga_menu;
14. private String tipe;
15. public Menu(String nama_menu, int harga_menu, String tipe) {
16. this.nama_menu = nama_menu;
17. this.harga_menu = harga_menu;
18. this.tipe = tipe;}
19. private static List<Menu> menus() {
20. try {
21. Menu[] daftarMenus = gson().fromJson(get("/menus").getData(), Menu[].class);
22. return FXCollections.observableArrayList(daftarMenus);
23. } catch (IOException e) {
24. return FXCollections.observableArrayList();}}
25. public static List<Menu> menus(String tipe) {
26. return menus().stream().filter(menu ->
    menu.tipe.equals(tipe)).collect(Collectors.toList());}
27. public static Menu menu(Pesanan pesanan) {
28. return menus()
29. .stream()
30. .filter(menu -> menu.nama_menu.equals(pesanan.getNama_menu()))
31. .findFirst()
32. .orElse(null);}
33. public String getNama_menu() {
34. return nama_menu;}
35. public int getHarga_menu() {
36. return harga_menu;}
37. public String getTipe() {
38. return tipe;}
39. public StringProperty nama_menuProperty() {
40. return new SimpleStringProperty(nama_menu);}
41. public StringProperty harga_menuProperty() {
42. return new SimpleStringProperty(rupiah(harga_menu));}
43. @Override
44. public String toString() {
45. return "Menu{" + ", nama_menu=" + nama_menu + "\" +", harga_menu=" + harga_menu
    + "}'";}}

```

J. Pesanan.java

```

1. package com.unindra.restoclient.models;
2. import com.google.gson.annotations.Expose;
3. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
4. import javafx.beans.property.ObjectProperty;
5. import javafx.beans.property.SimpleObjectProperty;
6. import javafx.beans.property.SimpleStringProperty;
7. import javafx.beans.property.StringProperty;
8. import javafx.collections.FXCollections;
9. import javafx.collections.ObservableList;
10. import java.io.IOException;
11. import java.util.List;
12. import java.util.stream.Collectors;
13. import static com.unindra.restoclient.Client.*;
14. import static com.unindra.restoclient.Rupiah.rupiah;
15. import static com.unindra.restoclient.models.Level.level;

```

```

16. import static com.unindra.restoclient.models.Menu.menu;
17. import static com.unindra.restoclient.models.Setting.setting;
18. public class Pesanan extends RecursiveTreeObject<Pesanan> {
19.     private String id_pesanan;
20.     private String nama_menu;
21.     private int jumlah;
22.     private int level;
23.     private String no_meja;
24.     private String status_item;
25.     @Expose
26.     private static final String paramUrl = "/pesanan";
27.     @Expose
28.     private static ObservableList<Pesanan> pesananList =
        FXCollections.observableArrayList();
29.     private Pesanan(String nama_menu, int jumlah, int lvl_item, String no_meja, String
        status_item) {
30.         this.id_pesanan = "";
31.         this.nama_menu = nama_menu;
32.         this.jumlah = jumlah;
33.         this.level = lvl_item;
34.         this.no_meja = no_meja;
35.         this.status_item = status_item; }
36.     public Pesanan(Menu menu, int jumlah) {
37.         this(menu.getNama_menu(), jumlah, 0, setting().getNo_meja(), "belum dipesan"); }
38.     public Pesanan(Menu menu, int jumlah, int lvl_item) {
39.         this(menu.getNama_menu(), jumlah, lvl_item, setting().getNo_meja(), "belum
        dipesan"); }
40.     public static void updatePesanan() throws IOException {
41.         StandardResponse standardResponse = get(paramUrl + "/" + setting().getNo_meja());
42.         if (standardResponse.getStatus() == StatusResponse.SUCCESS) {
43.             Pesanan[] pesananArrays = gson().fromJson(standardResponse.getData(),
                Pesanan[].class);
44.             for (Pesanan pesanan : pesananArrays)
                pesanan.setChildren(FXCollections.observableArrayList());
45.             Pesanan.pesananList.setAll(pesananArrays); } }
46.     public static boolean pesan() {
47.         getPesananList("belum dipesan").forEach(item -> item.status_item = "dipesan");
48.         return getPesananList("dipesan")
49.             .stream()
50.             .map(item -> item.put().getStatus() == StatusResponse.SUCCESS)
51.             .reduce(true, (a, b) -> a && b); }
52.     public static boolean bayar() throws IOException {
53.         getPesananList("diproses").forEach(item -> item.status_item = "dibayar");
54.         boolean success = getPesananList("dibayar").stream()
55.             .map(item -> item.put().getStatus() == StatusResponse.SUCCESS)
56.             .reduce(true, (a, b) -> a && b);
57.         StandardResponse standardResponse = get("/bayar/" + setting().getNo_meja());
58.         return success && standardResponse.getStatus() == StatusResponse.SUCCESS; }
59.     public StandardResponse post() {
60.         return send(paramUrl, "POST", gson().toJson(this)); }
61.     private StandardResponse put() {
62.         return send(paramUrl, "PUT", gson().toJson(this)); }
63.     public StandardResponse delete() {
64.         return send(paramUrl, "DELETE", gson().toJson(this)); }
65.     public static ObservableList<Pesanan> getPesananList() {

```

```

66. return pesananList;}
67. public static List<Pesanan> getPesananList(String status_item) {
68. return pesananList
69. .stream()
70. .filter(pesanan -> pesanan.getStatus_item().equals(status_item))
71. .collect(Collectors.toList());}
72. private int getTotal() {
73. try {
74. return (menu(this)).getHarga_menu() + level(level).getHarga_level() * jumlah;
75. } catch (IOException e) {
76. return 0;}}
77. public static int getGrandTotal() {
78. return getPesananList().stream().mapToInt(Pesanan::getTotal).sum();}
79. String getNama_menu() {
80. return nama_menu;}
81. public int getLevel() {
82. return level;}
83. public String getStatus_item() {
84. return status_item;}
85. public ObjectProperty<Integer> jumlahProperty() {
86. return new SimpleObjectProperty<>(jumlah);}
87. public StringProperty totalProperty() {
88. return new SimpleStringProperty(rupiah(getTotal()));}
89. @Override
90. public String toString() {
91. return "Pesanan{" + "id_pesanan=" + id_pesanan + ", nama_menu=" + nama_menu + ",
    jumlah=" + jumlah + ", level=" + level + ", no_meja=" + no_meja + ", status_item=" +
    status_item + "\" + '}'";}}

```

K. allmenu.fxml

```

1. <?xml version="1.0" encoding="UTF-8"?>
2. <?import com.jfoenix.controls.JFXButton?>
3. <?import java.lang.String?>
4. <?import javafx.geometry.Insets?>
5. <?import javafx.scene.control.Label?>
6. <?import javafx.scene.image.ImageView?>
7. <?import javafx.scene.layout.ColumnConstraints?>
8. <?import javafx.scene.layout.GridPane?>
9. <?import javafx.scene.layout.HBox?>
10. <?import javafx.scene.layout.RowConstraints?>
11. <GridPane alignment="CENTER" hgap="10.0" prefHeight="80.0" styleClass="border"
    styleSheets="@../css/style.css" xmlns="http://javafx.com/javafx/8.0.172-ea"
    xmlns:fx="http://javafx.com/fxml/1"
    fx:controller="com.unindra.restoclient.controllers.AllMenuController">
12. <columnConstraints>
13. <ColumnConstraints hgrow="NEVER" maxWidth="173.0" minWidth="10.0"
    prefWidth="156.0" />
14. <ColumnConstraints hgrow="NEVER" maxWidth="298.0" minWidth="10.0"
    prefWidth="129.0" />
15. <ColumnConstraints hgrow="NEVER" maxWidth="298.0" minWidth="10.0"
    prefWidth="116.0" />
16. <ColumnConstraints hgrow="NEVER" maxWidth="177.0" minWidth="10.0"
    prefWidth="177.0" />
17. </columnConstraints>
18. <rowConstraints>
19. <RowConstraints minHeight="10.0" vgrow="SOMETIMES" />

```

```

20. </rowConstraints>
21. <padding>
22. <Insets left="10.0" />
23. </padding>
24. <Label fx:id="namaLabel" prefHeight="17.0" prefWidth="163.0" text="Nama Menu">
25. <styleClass>
26. <String fx:value="text-body" />
27. <String fx:value="text-white" />
28. <String fx:value="text-bold" />
29. </styleClass>
30. </Label>
31. <Label fx:id="hargaLabel" prefHeight="17.0" prefWidth="128.0" text="Harga"
    GridPane.columnIndex="1">
32. <styleClass>
33. <String fx:value="text-body" />
34. <String fx:value="text-white" />
35. <String fx:value="text-bold" />
36. </styleClass>
37. </Label>
38. <HBox alignment="CENTER" prefHeight="19.0" prefWidth="100.0" spacing="5.0"
    GridPane.columnIndex="2">
39. <JFXButton focusTraversable="false" mnemonicParsing="false"
    onAction="#kurangJmlHandle" styleClass="kurang" />
40. <Label fx:id="jumlahLabel" text="1">
41. <styleClass>
42. <String fx:value="text-body" />
43. <String fx:value="text-white" />
44. </styleClass>
45. </Label>
46. <JFXButton focusTraversable="false" layoutX="10.0" layoutY="10.0"
    mnemonicParsing="false" onAction="#tambahJmlHandle" styleClass="tambah" />
47. </HBox>
48. <JFXButton fx:id="tambahButton" focusTraversable="false" mnemonicParsing="false"
    prefHeight="25.0" prefWidth="160.0" styleClass="tambah-pesanan-button"
    text="Tambah" GridPane.columnIndex="3">
49. <graphic>
50. <ImageView fitHeight="30.0" fitWidth="30.0" pickOnBounds="true"
    preserveRatio="true" />
51. </graphic>
52. <padding>
53. <Insets bottom="-5.0" />
54. </padding>
55. </JFXButton>
56. </GridPane>

```

L. app.fxml

```

1. <?xml version="1.0" encoding="UTF-8"?>
2. <?import com.jfoenix.controls.JFXButton?>
3. <?import javafx.geometry.Insets?>
4. <?import javafx.scene.control.ScrollPane?>
5. <?import javafx.scene.image.Image?>
6. <?import javafx.scene.image.ImageView?>
7. <?import javafx.scene.layout.HBox?>
8. <?import javafx.scene.layout.Pane?>
9. <?import javafx.scene.layout.VBox?>
10. <?import javafx.scene.text.Font?>

```

11. <VBox prefHeight="470.0" prefWidth="690.0" stylesheets="@../css/style.css" xmlns="http://javafx.com/javafx/8.0.172-ea" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.unindra.restoclient.controllers.AppController">
12. <HBox alignment="BOTTOM_LEFT" prefHeight="69.0" prefWidth="800.0" spacing="5.0" styleClass="body-white">
13. <ImageView fitHeight="50.0" fitWidth="50.0" onMousePressed="#settingHandle" pickOnBounds="true" preserveRatio="true">
14. <HBox.margin>
15. <Insets right="10.0" />
16. </HBox.margin>
17. <Image url="@../icons/logo-ramen-bulet-merah-copy50x50.png" />
18. </ImageView>
19. <JFXButton fx:id="ramenButton" focusTraversable="false" mnemonicParsing="false" onAction="#menuHandle" ripplerFill="WHITE" styleClass="ramen-pressed" text="Ramen" textOverrun="CLIP" HBox.hgrow="ALWAYS">
20.
21. <graphic>
22. <ImageView fitHeight="20.0" fitWidth="20.0" pickOnBounds="true" preserveRatio="true" />
23. </graphic>
24. </JFXButton>
25. <JFXButton fx:id="minumanButton" focusTraversable="false" layoutX="75.0" layoutY="45.0" mnemonicParsing="false" onAction="#menuHandle" ripplerFill="WHITE" styleClass="minuman" text="Minuman" textOverrun="CLIP" HBox.hgrow="ALWAYS">
26.
27.
28.
29. <graphic>
30. <ImageView fitHeight="20.0" fitWidth="20.0" pickOnBounds="true" preserveRatio="true" />
31. </graphic>
32. </JFXButton>
33. <JFXButton fx:id="cemilanButton" focusTraversable="false" mnemonicParsing="false" onAction="#menuHandle" ripplerFill="WHITE" styleClass="cemilan" text="Cemilan" textOverrun="CLIP" HBox.hgrow="ALWAYS">
34.
35.
36.
37. <graphic>
38. <ImageView fitHeight="20.0" fitWidth="20.0" pickOnBounds="true" preserveRatio="true" />
39. </graphic>
40. </JFXButton>
41. <JFXButton fx:id="lainnyaButton" focusTraversable="false" mnemonicParsing="false" onAction="#menuHandle" ripplerFill="WHITE" styleClass="lainnya" text="Lainnya" textOverrun="CLIP" HBox.hgrow="ALWAYS">
42.
43.
44.
45. <graphic>
46. <ImageView fitHeight="20.0" fitWidth="20.0" pickOnBounds="true" preserveRatio="true" />
47. </graphic>
48. </JFXButton>

```

49. <Pane HBox.hgrow="SOMETIMES" />
50. <JFXButton fx:id="pesananButton" alignment="TOP_CENTER" ellipsisString=""
    focusTraversable="false" mnemonicParsing="false" onAction="#daftarPesananHandle"
    prefHeight="28.0" prefWidth="128.0" styleClass="pesanan-button" text="Daftar
    Pesanan" textAlignment="CENTER" textOverrun="CLIP">
51. <graphic>
52. <ImageView fitHeight="20.0" fitWidth="20.0" pickOnBounds="true"
    preserveRatio="true" />
53. </graphic>
54. <font>
55. <Font name="System Bold" size="12.0" />
56. </font>
57. </JFXButton>
58. <padding>
59. <Insets bottom="10.0" left="50.0" right="50.0" top="10.0" />
60. </padding>
61. <VBox.margin>
62. <Insets />
63. </VBox.margin>
64. </HBox>
65. <ScrollPane fx:id="mainPane" fitToHeight="true" fitToWidth="true"
    stylesheets="@../css/scroll.css" VBox.vgrow="ALWAYS" />
66. </VBox>

```

M. DB.java

```

1. package com.unindra.restoserver;
2. import org.sql2o.Sql2o;
3. public class DB {
4.     public static Sql2o sql2o;
5.     static {
6.         sql2o = new Sql2o("jdbc:mysql://localhost/osaka", "root", "zxcasdqwe123");
7.         try {
8.             sql2o.open();
9.         } catch (Exception e) {
10.            sql2o = new Sql2o("jdbc:mysql://localhost/osaka", "root", "");
11.        }
12.    }
13. }

```

N. Laporan.java

```

1. package com.unindra.restoserver;
2. import com.itextpdf.io.image.ImageDataFactory;
3. import com.itextpdf.kernel.font.PdfFont;
4. import com.itextpdf.kernel.font.PdfFontFactory;
5. import com.itextpdf.kernel.geom.PageSize;
6. import com.itextpdf.kernel.geom.Rectangle;
7. import com.itextpdf.kernel.pdf.PdfDocument;
8. import com.itextpdf.kernel.pdf.PdfWriter;
9. import com.itextpdf.layout.Document;
10. import com.itextpdf.layout.borders.Border;
11. import com.itextpdf.layout.element.Image;
12. import com.itextpdf.layout.element.*;
13. import com.itextpdf.layout.property.HorizontalAlignment;
14. import com.itextpdf.layout.property.TextAlignment;
15. import com.itextpdf.layout.property.UnitValue;
16. import com.itextpdf.layout.property.VerticalAlignment;
17. import com.unindra.restoserver.models.Menu;
18. import com.unindra.restoserver.models.Pesanan;
19. import com.unindra.restoserver.models.PesananService;

```

```

20. import com.unindra.restoserver.models.Transaksi;
21. import javafx.collections.FXCollections;
22. import org.joda.time.LocalDate;
23. import org.joda.time.LocalTime;
24. import javax.imageio.ImageIO;
25. import java.awt.*;
26. import java.awt.image.BufferedImage;
27. import java.io.ByteArrayOutputStream;
28. import java.io.File;
29. import java.io.IOException;
30. import java.util.Arrays;
31. import java.util.Date;
32. import java.util.List;
33. import java.util.concurrent.atomic.AtomicInteger;
34. import static com.unindra.restoserver.Rupiah.rupiah;
35. import static com.unindra.restoserver.models.Menu.getMenus;
36. import static com.unindra.restoserver.models.Menu.menu;
37. import static com.unindra.restoserver.models.Pesanan.getPesanan;
38. import static com.unindra.restoserver.models.Transaksi.getTotalBayar;
39. import static com.unindra.restoserver.models.Transaksi.getTransaksiList;
40. public class Laporan {
41.     private static final String bold = "fonts/OpenSans-Bold.ttf";
42.     private static Table kop_surat(String judul) throws IOException {
43.         BufferedImage image = ImageIO.read(
44.             Laporan.class.getResourceAsStream("/icons/logo-ramen-bulet-merah-copy50x50.png"));
45.         ByteArrayOutputStream baos = new ByteArrayOutputStream();
46.         ImageIO.write(image, "png", baos);
47.         baos.flush();
48.         byte[] imageInByte = baos.toByteArray();
49.         baos.close();
50.         Image img = new Image(ImageDataFactory.create(imageInByte));
51.         return new Table(new UnitValue[][] {
52.             new UnitValue(UnitValue.PERCENT, 10),
53.             new UnitValue(UnitValue.PERCENT, 90)}, true)
54.             .setFontSize(12)
55.             .addCell(cellNoBorder(img.setAutoScale(true)))
56.             .addCell(cellNoBorder("Osaka Ramen\n" + judul)
57.                 .setTextAlignment(TextAlignment.CENTER)
58.                 .setHorizontalAlignment(HorizontalAlignment.CENTER)
59.                 .setVerticalAlignment(VerticalAlignment.MIDDLE));}
60.         private static Table signature(LocalDate tgl) {
61.             return new Table(1)
62.                 .setFontSize(10)
63.                 .setWidth(130)
64.                 .setHeight(80)
65.                 .setMarginTop(10)
66.                 .setHorizontalAlignment(HorizontalAlignment.RIGHT)
67.                 .addCell(cellNoBorder("Depok" + ", " +
68.                     hari().get(tgl.getDayOfWeek()) + ", " +
69.                     tgl.getDayOfMonth() + " " +
70.                     bulan().get(tgl.getMonthOfYear()) + " " +
71.                     tgl.getYear()).setTextAlignment(TextAlignment.CENTER))
72.                 .addCell(cellNoBorder("Pemilik\nTaufik Hidayat")
73.                     .setTextAlignment(TextAlignment.CENTER))

```

```

74. .setVerticalAlignment(VerticalAlignment.BOTTOM));}
75. private static Cell cellNoBorder(String text) {
76.     return new Cell()
77.     .setBorder(Border.NO_BORDER)
78.     .add(new Paragraph(text));}
79. private static Cell cellNoBorder(Image image) {
80.     return new Cell()
81.     .setBorder(Border.NO_BORDER)
82.     .add(image);}
83. private static Cell cell(String text) {
84.     return new Cell().add(new Paragraph(text));}
85. private static List<String> hari() {
86.     return Arrays.asList("", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu");}
87. private static List<String> bulan() {
88.     return
        Arrays.asList("", "Januari", "Februari", "Maret", "April", "Mei", "Juni", "Juli", "Agustus", "S
        eptember", "Oktober", "November", "Desember");}
89. private static void showReport(String fileName) {
90.     File file = new File(fileName);
91.     Desktop desktop = Desktop.getDesktop();
92.     try {
93.         desktop.open(file);
94.     } catch (IOException e) {
95.         e.printStackTrace();}
96.     private static boolean makeDir() {
97.         File file = new File(System.getProperty("user.home") + "\\Documents\\LaporanResto");
98.         if (!file.exists()) return file.mkdir();
99.         else return true;}
100. public static void pemesanan(LocalDate dari, LocalDate sampai) throws IOException {
101.     PdfFont boldFont = PdfFontFactory.createFont(bold, true);
102.     LocalDate localDate = new LocalDate(new Date());
103.     String fileName = String.format("%s\\Documents\\LaporanResto\\laporan-pemesanan-
        %s.pdf", System.getProperty("user.home"), localDate.toString());
104.     if (makeDir()) {
105.         PdfWriter writer = new PdfWriter(fileName);
106.         PdfDocument pdf = new PdfDocument(writer);
107.         Document document = new Document(pdf, PageSize.A4);
108.         document.add(kop_surat("Laporan Pemesanan"));
109.         Table table = new Table(6)
110.         .setWidth(520)
111.         .setMarginTop(0)
112.         .setFontSize(10)
113.         .addHeaderCell(cell("Pukul").setFont(boldFont))
114.         .addHeaderCell(cell("No Meja").setFont(boldFont))
115.         .addHeaderCell(cell("Nama Menu").setFont(boldFont))
116.         .addHeaderCell(cell("Jumlah").setFont(boldFont))
117.         .addHeaderCell(cell("Harga").setFont(boldFont))
118.         .addHeaderCell(cell("Total Harga").setFont(boldFont));
119.         while (dari.isBefore(sampai.plusDays(1))) {
120.             getTransaksiList(dari).forEach(transaksi ->
                Pesanan.getPesanan(transaksi).forEach(pesanan -> {
121.                 LocalTime t = new LocalTime(transaksi.getTanggal());
122.                 table.addCell(cell(String.format("%d:%d WIB", t.getHourOfDay(),
123.                 t.getMinuteOfHour())));
124.                 table.addCell(cell(transaksi.getNo_meja()));

```



```

125. table.addCell(cell(pesanan.getNama_menu()));
126. table.addCell(cell(String.valueOf(pesanan.getJumlah())));
127. table.addCell(cell(rupiah(menu(pesanan).getHarga_menu())));
128. table.addCell(cell(rupiah(pesanan.getTotal())));});
129. dari = dari.plusDays(1);}
130. document.add(table);
131. document.add(signature(localDate));
132. document.close();
133. showReport(fileName);} }
134. public static void pemasukan(LocalDate dari, LocalDate sampai) throws IOException {
135. PdfFont boldFont = PdfFontFactory.createFont(bold, true);
136. LocalDate localDate = new LocalDate(new Date());
137. String fileName = String.format(
138. "%s\\Documents\\LaporanResto\\laporan-pemasukan-%s.pdf",
139. System.getProperty("user.home"),
140. localDate.toString());
141. if (mkdir()) {
142. PdfWriter writer = new PdfWriter(fileName);
143. PdfDocument pdf = new PdfDocument(writer);
144. Document document = new Document(pdf, PageSize.A4);
145. document.add(kop_surat("Laporan Pemasukan"));
146. Table table = new Table(2)
147. .setWidth(520)
148. .setMarginTop(0)
149. .setFontSize(10)
150. .addHeaderCell(cell("Tanggal").setFont(boldFont))
151. .addHeaderCell(cell("Total Pemasukan").setFont(boldFont));
152. while (dari.isBefore(sampai.plusDays(1))) {
153. table.addCell(cell(dari.toString()));
154. table.addCell(cell(rupiah(getTotalBayar(dari.getYear(), dari.getMonthOfYear()))));
155. dari = dari.plusDays(1);}
156. document.add(table);
157. document.add(signature(localDate));
158. document.close();
159. showReport(fileName);} }
160. public static void menuFavorit(LocalDate dari, LocalDate sampai) throws
    IOException{
161. PdfFont boldFont = PdfFontFactory.createFont(bold, true);
162. LocalDate localDate = new LocalDate(new Date());
163. String fileName = String.format(
164. "%s\\Documents\\LaporanResto\\laporan-menu-favorit-%s.pdf",
165. System.getProperty("user.home"),
166. localDate.toString());
167. if (mkdir()) {
168. PdfWriter writer = new PdfWriter(fileName);
169. PdfDocument pdf = new PdfDocument(writer);
170. Document document = new Document(pdf, PageSize.A4);
171. document.add(kop_surat("Laporan Menu Favorit"));
172. Table table = new Table(4)
173. .setWidth(520)
174. .setMarginTop(0)
175. .setFontSize(10)
176. .addHeaderCell(cell("Nama Menu").setFont(boldFont))
177. .addHeaderCell(cell("Tipe").setFont(boldFont))
178. .addHeaderCell(cell("Harga").setFont(boldFont))

```

```

179. .addHeaderCell(cell("Total Dipesan").setFont(boldFont));
180. List<Menu> menus = FXCollections.observableArrayList(getMenus());
181. menus.sort((menu1, menu2) -> {
182. List<Pesanan> items1 = getPesanan(menu1);
183. List<Pesanan> items2 = getPesanan(menu2);
184. return items2.size() - items1.size();});
185. for (Menu menu : getMenus()) {
186. AtomicInteger jumlahMenu = new AtomicInteger();
187. LocalDate tgl = dari;
188. while (tgl.isBefore(sampai.plusDays(1))) {
189. for (Transaksi transaksi : getTransaksiList(tgl)) {
190. jumlahMenu.addAndGet(getPesanan(transaksi,
    menu).stream().mapToInt(Pesanan::getJumlah).sum());}
191. tgl = tgl.plusDays(1);}
192. if (jumlahMenu.get() > 0) {
193. table.addCell(cell(menu.getNama_menu()));
194. table.addCell(cell(menu.getTipe()));
195. table.addCell(cell(rupiah(menu.getHarga_menu())));
196. table.addCell(cell(String.valueOf(getPesanan(menu).size())));} }
197. document.add(table);
198. document.add(signature(localDate));
199. document.close();
200. showReport(fileName);} }
201. public static void kunjungan(LocalDate dari, LocalDate sampai) throws IOException {
202. PdfFont boldFont = PdfFontFactory.createFont(bold, true);
203. LocalDate localDate = new LocalDate(new Date());
204. String fileName = String.format("%s\\Documents\\LaporanResto\\laporan-kunjungan-
    %s.pdf",
205. System.getProperty("user.home"),
206. localDate.toString());
207. if (mkdir()) {
208. PdfWriter writer = new PdfWriter(fileName);
209. PdfDocument pdf = new PdfDocument(writer);
210. Document document = new Document(pdf, PageSize.A4);
211. document.add(kop_surat("Laporan Kunjungan"));
212. Table table = new Table(2)
213. .setWidth(520)
214. .setMarginTop(0)
215. .setFontSize(10)
216. .addHeaderCell(cell("Tanggal").setFont(boldFont))
217. .addHeaderCell(cell("Total Kunjungan").setFont(boldFont));
218. while (dari.isBefore(sampai.plusDays(1))) {
219. int totalKunjungan = getTransaksiList(dari.getYear(), dari.getMonthOfYear()).size();
220. table.addCell(cell(dari.toString()));
221. table.addCell(cell(String.valueOf(totalKunjungan)));
222. dari = dari.plusDays(1);}
223. document.add(table);
224. document.add(signature(localDate));
225. document.close();
226. showReport(fileName);} }
227. public static void bill(Transaksi transaksi) throws IOException {
228. List<Pesanan> pesanans = PesananService.getItems(transaksi);
229. String fileName = String.format("%s\\Documents\\LaporanResto\\bill.pdf",
    System.getProperty("user.home"));
230. PdfFont boldFont = PdfFontFactory.createFont(bold, true);

```

```

231. LocalDate localDate = new LocalDate(new Date());
232. if (makeDir()) {
233. PdfWriter writer = new PdfWriter(fileName);
234. PdfDocument pdf = new PdfDocument(writer);
235. Document document = new Document(pdf, new PageSize(new Rectangle(226.8f,
    600f)));
236. document.add(
237. new Paragraph()
238. .setTextAlignment(TextAlignment.CENTER)
239. .setFontSize(5)
240. .add(new Text("OSAKA RAMEN").setFont(boldFont))
241. .add("\n-----\n")
242. .add("No Meja:")
243. .add(pesanans.get(0).getNo_meja()).add("\tTanggal:")
244. .add(localDate + " " + new LocalTime().toString().substring(0, 8))
245. .add("\n-----\n"));
246. Table table = new Table(new UnitValue[]{
247. new UnitValue(UnitValue.PERCENT, 30),
248. new UnitValue(UnitValue.PERCENT, 20),
249. new UnitValue(UnitValue.PERCENT, 50),}, true)
250. .setFontSize(6)
251. .setTextAlignment(TextAlignment.CENTER);
252. pesanans.forEach(item -> {
253. table.addCell(cellNoBorder(menu(item).getNama_menu()));
254. table.addCell(cellNoBorder(item.getJumlah() + "x"));
255. table.addCell(cellNoBorder(rupiah(item.getTotal())));});
256. document.add(table);
257. document.add(
258. new Paragraph()
259. .setTextAlignment(TextAlignment.CENTER)
260. .setFontSize(5)
261. .add("-----"));
262. Table footerTable = new Table(new UnitValue[]{
263. new UnitValue(UnitValue.PERCENT, 50),
264. new UnitValue(UnitValue.PERCENT, 50),}, true)
265. .setTextAlignment(TextAlignment.RIGHT)
266. .addCell(cellNoBorder("Total").setFontSize(6))
267. .addCell(cellNoBorder(rupiah(transaksi.getTotalBayarFromService())).setFontSize(6));
268. document.add(footerTable);
269. document.close();
270. showReport(fileName);}}
271. public static void struk(Transaksi transaksi, int tunai) throws IOException {
272. List<Pesanan> pesanans = PesananService.getItems(transaksi);
273. String fileName = String.format("%s\\Documents\\LaporanResto\\struk.pdf",
    System.getProperty("user.home"));
274. PdfFont boldFont = PdfFontFactory.createFont(bold, true);
275. LocalDate localDate = new LocalDate(new Date());
276. if (makeDir()) {
277. PdfWriter writer = new PdfWriter(fileName);
278. PdfDocument pdf = new PdfDocument(writer);
279. Document document = new Document(pdf, new PageSize(new Rectangle(226.8f,
    600f)));
280. document.add(
281. new Paragraph()
282. .setTextAlignment(TextAlignment.CENTER)

```

```

283. .setFontSize(5)
284. .add(new Text("OSAKA RAMEN").setFont(boldFont))
285. .add("\nJl. Keadilan No. 23G, Rangkapan Jaya Baru, Pancoran Mas, Kota Depok Jawa Barat")
286. .add("\n-----\n")
287. .add("No Meja:")
288. .add(pesanans.get(0).getNo_meja())
289. .add("\tTanggal:")
290. .add(LocalDate + " " + new LocalTime().toString().substring(0, 8))
291. .add("\n-----\n"));
292. Table table = new Table(new UnitValue[]{
293.     new UnitValue(UnitValue.PERCENT, 30),
294.     new UnitValue(UnitValue.PERCENT, 20),
295.     new UnitValue(UnitValue.PERCENT, 50),}, true)
296. .setFontSize(6)
297. .setTextAlignment(TextAlignment.CENTER);
298. pesanans.forEach(item -> {
299.     table.addCell(cellNoBorder(menu(item).getNama_menu()));
300.     table.addCell(cellNoBorder(item.getJumlah() + "x"));
301.     table.addCell(cellNoBorder(rupiah(item.getTotal()))); });
302. document.add(table);
303. document.add(
304.     new Paragraph()
305.     .setTextAlignment(TextAlignment.CENTER)
306.     .setFontSize(5)
307.     .add("-----"));
308. Table footerTable = new Table(new UnitValue[]{
309.     new UnitValue(UnitValue.PERCENT, 50),
310.     new UnitValue(UnitValue.PERCENT, 50),}, true)
311. .setTextAlignment(TextAlignment.RIGHT)
312. .addCell(cellNoBorder("Total").setFontSize(6))
313. .addCell(cellNoBorder(rupiah(transaksi.getTotalBayarFromService())).setFontSize(6))
314. .addCell(cellNoBorder("Tunai").setFontSize(6))
315. .addCell(cellNoBorder(rupiah(tunai).setFontSize(6))
316. .addCell(cellNoBorder("Kembali").setFontSize(6))
317. .addCell(cellNoBorder(rupiah(tunai -
318.     transaksi.getTotalBayarFromService())).setFontSize(6));
319. document.add(footerTable);
320. document.close();
321. showReport(fileName);}}

```

O. Router.java

```

1. package com.unindra.restoserver;
2. import com.google.gson.*;
3. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
4. import com.unindra.restoserver.models.*;
5. import javafx.collections.FXCollections;
6. import static com.unindra.restoserver.models.Level.getLevels;
7. import static com.unindra.restoserver.models.Menu.getMenus;
8. import static com.unindra.restoserver.models.PesananService.getItems;
9. import static spark.Spark.*;
10. class Router {
11.     private static Gson gson;
12.     static {
13.         gson = new GsonBuilder().addSerializationExclusionStrategy(new ExclusionStrategy(){
14.             @Override

```

```

15. public boolean shouldSkipField(FieldAttributes fieldAttributes) {
    return fieldAttributes.getDeclaringClass().equals(RecursiveTreeObject.class);}
16. @Override
17. public boolean shouldSkipClass(Class<?> aClass) {
    return false;}
18. }).addDeserializationExclusionStrategy(new ExclusionStrategy() {
19. @Override
20. public boolean shouldSkipField(FieldAttributes fieldAttributes) {
    return fieldAttributes.getDeclaringClass().equals(RecursiveTreeObject.class);}
21. @Override
22. public boolean shouldSkipClass(Class<?> aClass) {
    return false;
23. }).create();
24. post("/pesanan", (request, response) -> {
25. response.type("application/json");
26. Pesanan pesanan = gson.fromJson(request.body(), Pesanan.class);
27. PesananService.add(pesanan);
28. return gson.toJson(new StandardResponse(StatusResponse.SUCCESS));});
29. get("/pesanan/:no_meja", (request, response) -> {
30. response.type("application/json");
31. JsonElement jsonElement = gson.toJsonTree(getItems(request.params(":no_meja")));
32. return gson.toJson(new StandardResponse(StatusResponse.SUCCESS, jsonElement));});
33. put("/pesanan", (request, response) -> {
34. response.type("application/json");
35. Pesanan pesanan = gson.fromJson(request.body(), Pesanan.class);
36. pesanan.setChildren(FXCollections.observableArrayList());
37. if (PesananService.update(pesanan))
38. return gson.toJson(new StandardResponse(StatusResponse.SUCCESS, "Pesanan
    diedit"));
39. return gson.toJson(new StandardResponse(StatusResponse.ERROR, "Pesanan gagal
    dihapus"));});
40. delete("/pesanan", (request, response) -> {
41. response.type("application/json");
42. Pesanan pesanan = gson.fromJson(request.body(), Pesanan.class);
43. if (PesananService.delete(pesanan))
44. return gson.toJson(new StandardResponse(StatusResponse.SUCCESS, "Pesanan
    dihapus"));
45. return gson.toJson(new StandardResponse(StatusResponse.ERROR, "Pesanan gagal
    dihapus"));});
46. get("/menus", (request, response) -> {
47. response.type("application/json");
48. return gson.toJson(new StandardResponse(
49. StatusResponse.SUCCESS,
50. gson.toJsonTree(getMenus())));});
51. get("/levels", (request, response) -> {
52. response.type("application/json");
53. return gson.toJson(new StandardResponse(
54. StatusResponse.SUCCESS,
55. gson.toJsonTree(getLevels())));});
56. get("/bayar/:no_meja", (request, response) -> {
57. response.type("application/json");
58. Transaksi transaksi = new Transaksi(request.params(":no_meja"));
59. TransaksiService.add(transaksi);
60. return gson.toJson(new StandardResponse(StatusResponse.SUCCESS));});
61. get("/detail_ramen/:nama_menu", (request, response) -> {

```

```

62. response.type("application/json");
63. return gson.toJson(new StandardResponse(StatusResponse.SUCCESS,
64. gson.toJsonTree(DetailRamen.detailRamen(request.params(":nama_menu"))));));});}}

```

P. DaftarMenuController.java

```

1.  package com.unindra.restoserver.controllers;
2.  import com.jfoenix.controls.*;
3.  import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
4.  import com.unindra.restoserver.Dialog;
5.  import com.unindra.restoserver.models.DetailRamen;
6.  import com.unindra.restoserver.models.Level;
7.  import com.unindra.restoserver.models.Menu;
8.  import javafx.collections.FXCollections;
9.  import javafx.collections.ObservableList;
10. import javafx.fxml.Initializable;
11. import javafx.scene.control.Label;
12. import javafx.scene.control.TreeItem;
13. import javafx.scene.control.TreeTableColumn;
14. import javafx.scene.control.TreeTableView;
15. import javafx.scene.input.MouseEvent;
16. import javafx.scene.layout.HBox;
17. import javafx.stage.FileChooser;
18. import javafx.stage.Stage;
19. import java.io.File;
20. import java.io.IOException;
21. import java.net.URL;
22. import java.nio.file.Files;
23. import java.util.ResourceBundle;
24. import static com.unindra.restoserver.models.Level.getLevels;
25. import static com.unindra.restoserver.models.Menu.getMenus;
26. public class DaftarMenuController implements Initializable {
27.     public JFXTreeTableView<Menu> menuTableView;
28.     public JFXButton actionButton;
29.     public JFXButton hapusButton;
30.     public JFXTextField namaField;
31.     public JFXTextField hargaField;
32.     public JFXComboBox<String> tipeComboBox;
33.     public JFXTextArea deskArea;
34.     public Label titleLabel;
35.     public JFXTreeTableView<Level> levelTableView;
36.     public JFXTextField hargaLevelField;
37.     public JFXTextField levelField;
38.     public HBox formForRamenPane;
39.     public JFXButton pilihGambarButton;
40.     private ObservableList<String> tipeList;
41.     private Menu menu;
42.     private Level level;
43.     private DetailRamen detailRamen;
44.     @Override
45.     public void initialize(URL location, ResourceBundle resources) {
46.         TreeTableColumn<Menu, String> namaCol = new TreeTableColumn<>("Nama");
47.         TreeTableColumn<Menu, String> tipeCol = new TreeTableColumn<>("Tipe");
48.         TreeTableColumn<Menu, String> hargaCol = new TreeTableColumn<>("Harga");
49.         namaCol.setCellValueFactory(param ->
50.             param.getValue().getValue().nama_menuProperty());
51.         tipeCol.setCellValueFactory(param -> param.getValue().getValue().tipeProperty());

```

```

51. hargaCol.setCellValueFactory(param ->
    param.getValue().getValue().harga_menuProperty());
52. TreeItem<Menu> rootMenu = new RecursiveTreeItem<>(getMenus(),
    RecursiveTreeObject::getChildren);
53. menuTableView.setRoot(rootMenu);
54. menuTableView.getColumns().add(namaCol);
55. menuTableView.getColumns().add(tipeCol);
56. menuTableView.getColumns().add(hargaCol);
57. menuTableView.setColumnResizePolicy(TreeTableView.CONSTRAINED_RESIZE_P
    OLICY);
58. tipeList = FXCollections.observableArrayList("ramen", "minuman", "cemilan",
    "lainnya");
59. tipeComboBox.setItems(tipeList);
60. TreeTableColumn<Level, Integer> levelCol = new TreeTableColumn<>("Level");
61. TreeTableColumn<Level, String> hargaLevelCol = new
    TreeTableColumn<>("Harga");
62. levelCol.setCellValueFactory(param -> param.getValue().getValue().levelProperty());
63. hargaLevelCol.setCellValueFactory(param ->
    param.getValue().getValue().harga_levelProperty());
64. TreeItem<Level> rootLevel = new RecursiveTreeItem<>(getLevels(),
    RecursiveTreeObject::getChildren);
65. levelTableView.setRoot(rootLevel);
66. levelTableView.getColumns().add(levelCol);
67. levelTableView.getColumns().add(hargaLevelCol);
68. levelTableView.setColumnResizePolicy(TreeTableView.CONSTRAINED_RESIZE_P
    OLICY);
69. hargaField.textProperty().addListener((observable, oldValue, newValue) -> {
70.     if (!newValue.matches("\\d*")) {hargaField.setText(newValue.replaceAll("[^\\d]",
        ""));}});
71. hargaLevelField.textProperty().addListener((observable, oldValue, newValue) -> {
72.     if (!newValue.matches("\\d*")) {hargaLevelField.setText(newValue.replaceAll("[^\\d]",
        ""));}});
73. public void actionHandle() {
74.     if (actionButton.getText().equals("Tambah")) {
75.         menu = new Menu(
76.             namaField.getText(),
77.             tipeComboBox.getSelectionModel().getSelectedItem(),
78.             Integer.valueOf(hargaField.getText()));
79.         if (menu.getTipe().equals("ramen")) {
80.             if (detailRamen != null) {
81.                 detailRamen.setNama_menu(namaField.getText());
82.                 detailRamen.setDeskripsi(deskArea.getText());
83.                 if (menu.add() && detailRamen.add()) {
84.                     getDialog().information("Berhasil!", "Menu berhasil ditambahkan");
85.                     reset();
86.                 } else getDialog().information("Gagal", "Menu gagal ditambahkan");
87.             } else getDialog().information("Gagal", "Gambar belum dipilih");
88.             } else {
89.                 if (menu.add()) {
90.                     getDialog().information("Berhasil!", "Menu berhasil ditambahkan");
91.                     reset();
92.                 } else getDialog().information("Gagal", "Menu gagal ditambahkan");}
93.             } else { // Ubah
94.                 menu.setNama_menu(namaField.getText());
95.                 menu.setTipe(tipeComboBox.getSelectionModel().getSelectedItem());

```

```

96. menu.setHarga_menu(Integer.valueOf(hargaField.getText()));
97. if (menu.getTipe().equals("ramen")) {
98. detailRamen.setNama_menu(menu.getNama_menu());
99. detailRamen.setDeskripsi(deskArea.getText());
100. if (menu.update() && detailRamen.update()) {
101. getDialog().information("Berhasil!", "Menu berhasil diubah");
102. reset();
103. } else getDialog().information("Gagal", "Menu gagal diubah");
104. } else {
105. if (menu.update()) {
106. getDialog().information("Berhasil!", "Menu berhasil diubah");
107. reset();
108. } else getDialog().information("Gagal", "Menu gagal diubah");}}
109. public void ubahLevelHandle() {
110. level.setHarga_level(Integer.parseInt(hargaLevelField.getText()));
111. if (level.update()) {
112. getDialog().information("Berhasil!", "Level berhasil diubah");
113. reset();}}
114. public void pilihHandle(MouseEvent mouseEvent) {
115. if (!menuTableView.getSelectionModel().isEmpty()) {
116. menu = menuTableView.getSelectionModel().getSelectedItem().getValue();
117. detailRamen = DetailRamen.detailRamen(menu);
118. namaField.setText(menu.getNama_menu());
119. int index = tipeList.indexOf(menu.getTipe());
120. tipeComboBox.getSelectionModel().clearAndSelect(index);
121. hargaField.setText(String.valueOf(menu.getHarga_menu()));
122. if (detailRamen != null) deskArea.setText(detailRamen.getDeskripsi());
123. titleLabel.setText("UBAH MENU");
124. namaField.setDisable(true);
125. hapusButton.setVisible(true);
126. actionButton.setText("Ubah");
127. actionButton.getStyleClass().set(2, "ubah");}
128. if (mouseEvent.getClickCount() == 2) reset();}
129. public void pilihLevelHandle(MouseEvent mouseEvent) {
130. if (!levelTableView.getSelectionModel().isEmpty()) {
131. level = levelTableView.getSelectionModel().getSelectedItem().getValue();
132. levelField.setText(String.valueOf(level.getLevel()));
133. hargaLevelField.setText(String.valueOf(level.getHarga_level()));
134. hargaLevelField.setDisable(false);}
135. if (mouseEvent.getClickCount() == 2) reset();}
136. public void hapusHandle() {
137. Dialog alert = getDialog();
138. alert.confirmation("Anda yakin ingin menghapus menu ini?",
139. event -> {
140. Menu menu = menuTableView.getSelectionModel().getSelectedItem().getValue();
141. if (menu.getTipe().equals("ramen")) {
142. if (menu.delete() && detailRamen.delete()) {
143. alert.information("Berhasil!", "Menu berhasil dihapus");
144. reset();
145. } else alert.information("Gagal", "Menu gagal dihapus");
146. } else {
147. if (menu.delete()) {alert.information("Berhasil!", "Menu berhasil dihapus");
148. reset();
149. } else alert.information("Gagal", "Menu gagal dihapus");}}});}
150. private void reset() {

```



```

151. titleLabel.setText("TAMBAH MENU");
152. hapusButton.setVisible(false);
153. actionButton.setText("Tambah");
154. actionButton.getStyleClass().set(2, "tambah");
155. menuTableView.getSelectionModel().clearSelection();
156. namaField.setDisable(false);
157. namaField.setText("");
158. tipeComboBox.getSelectionModel().clearSelection();
159. hargaField.setText("");
160. deskArea.setText("");
161. levelField.setText("");
162. hargaLevelField.setText("");
163. hargaLevelField.setDisable(true);
164. levelTableView.getSelectionModel().clearSelection();
165. namaField.requestFocus();
166. pilihGambarButton.setText("Pilih gambar... (max : 2048 KB)");
167. public void tipeHandle() {
168. if (tipeComboBox.getSelectionModel().getSelectedItem() != null) {
169. if (tipeComboBox.getSelectionModel().getSelectedItem().equals("ramen"))
        formForRamenPane.setDisable(false);
170. else {
171. formForRamenPane.setDisable(true);
172. detailRamen = null;
173. deskArea.setText("");
174. pilihGambarButton.setText("Pilih gambar... (max : 2048 KB)");}}
175. public void pilihGambarHandle() throws IOException {
176. FileChooser fileChooser = new FileChooser();
177. fileChooser.setTitle("Open Resource File");
178. File file = fileChooser.showOpenDialog(actionButton.getScene().getWindow());
179. if (file != null) {
180. if (file.length() <= 2048000) {
181. pilihGambarButton.setText(file.getName());
182. detailRamen = new DetailRamen(Files.readAllBytes(file.toPath()));
183. } else {
184. getDialog().information("Gagal", "Ukuran foto terlalu besar");}}
185. private Dialog getDialog() {
186. return new Dialog((Stage) actionButton.getScene().getWindow());}

```

Q. LaporanController.java

```

1. package com.unindra.restoserver.controllers;
2. import com.jfoenix.controls.JFXComboBox;
3. import com.jfoenix.controls.JFXDatePicker;
4. import com.jfoenix.controls.JFXTreeTableView;
5. import com.jfoenix.controls.RecursiveTreeItem;
6. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
7. import com.unindra.restoserver.Laporan;
8. import com.unindra.restoserver.models.Menu;
9. import com.unindra.restoserver.models.Pesanan;
10. import com.unindra.restoserver.models.Transaksi;
11. import javafx.application.Platform;
12. import javafx.collections.FXCollections;
13. import javafx.collections.ListChangeListener;
14. import javafx.collections.ObservableList;
15. import javafx.fxml.Initializable;
16. import javafx.scene.chart.AreaChart;
17. import javafx.scene.chart.PieChart;

```

```

18. import javafx.scene.chart.XYChart;
19. import javafx.scene.control.TreeItem;
20. import javafx.scene.control.TreeTableCell;
21. import javafx.scene.control.TreeTableColumn;
22. import javafx.scene.control.TreeTableView;
23. import javafx.util.Callback;
24. import org.joda.time.LocalDate;
25. import java.io.IOException;
26. import java.net.URL;
27. import java.time.ZoneId;
28. import java.util.Date;
29. import java.util.ResourceBundle;
30. import java.util.concurrent.atomic.AtomicInteger;
31. import static com.unindra.restoserver.models.Menu.getMenus;
32. import static com.unindra.restoserver.models.Menu.menu;
33. import static com.unindra.restoserver.models.Pesanan.getPesanan;
34. import static com.unindra.restoserver.models.Transaksi.getTransaksi;
35. import static com.unindra.restoserver.models.Transaksi.getTransaksiList;
36. public class LaporanController implements Initializable {
37.     public AreaChart pemasukanChart;
38.     public PieChart menuFavChart;
39.     public AreaChart kunjunganChart;
40.     public JFXComboBox<String> pilihLaporanCombo;
41.     public JFXDatePicker dariDatePicker;
42.     public JFXDatePicker sampaiDatePicker;
43.     public JFXTreeTableView<Pesanan> pemesananTableView;
44.     @Override
45.     public void initialize(URL location, ResourceBundle resources) {
46.         TreeTableColumn<Pesanan, String> pukulCol = new TreeTableColumn<>("Pukul");
47.         TreeTableColumn<Pesanan, String> mejaCol = new TreeTableColumn<>("No Meja");
48.         TreeTableColumn<Pesanan, String> namaCol = new TreeTableColumn<>("Nama");
49.         TreeTableColumn<Pesanan, Integer> jumlahCol = new
            TreeTableColumn<>("Jumlah");
50.         TreeTableColumn<Pesanan, String> hargaCol = new TreeTableColumn<>("Harga");
51.         TreeTableColumn<Pesanan, String> totalHargaCol = new TreeTableColumn<>("Total
            Harga");
52.         pukulCol.setCellValueFactory(param ->
            getTransaksi(param.getValue().getValue()).pukulProperty());
53.         pukulCol.setCellValueFactory(param ->
            getTransaksi(param.getValue().getValue()).pukulProperty());
54.         mejaCol.setCellValueFactory(param ->
            getTransaksi(param.getValue().getValue()).no_mejaProperty());
55.         namaCol.setCellValueFactory(param ->
            menu(param.getValue().getValue()).nama_menuProperty());
56.         jumlahCol.setCellValueFactory(param ->
            param.getValue().getValue().jumlahProperty());
57.         hargaCol.setCellValueFactory(param ->
            menu(param.getValue().getValue()).harga_menuProperty());
58.         totalHargaCol.setCellValueFactory(param ->
            param.getValue().getValue().totalHargaProperty());
59.         namaCol.setCellFactory(new Callback<TreeTableColumn<Pesanan, String>,
            TreeTableCell<Pesanan, String>>() {
60.             @Override
61.             public TreeTableCell<Pesanan, String> call(TreeTableColumn<Pesanan, String>
                param) {

```

```

62. return new TreeTableCell<Pesanan, String>() {
63.     @Override
64.     protected void updateItem(String item, boolean empty) {
65.         super.updateItem(item, empty);
66.         if (item == null) {
67.             setText(null);
68.         } else {
69.             Pesanan i = Pesanan.getPesananList().get(getIndex());
70.             if (menu(i).getTipe().equals("ramen"))
71.                 setText(item + " lv." + i.getLevel());
72.             else setText(item);
73.         }
74.     };
75.     TreeItem<Pesanan> rootItem = new RecursiveTreeItem<>(Pesanan.getPesananList(),
76.         RecursiveTreeObject::getChildren);
77.     pemesananTableView.setRoot(rootItem);
78.     pemesananTableView.getColumns().add(pukulCol);
79.     pemesananTableView.getColumns().add(mejaCol);
80.     pemesananTableView.getColumns().add(namaCol);
81.     pemesananTableView.getColumns().add(jumlahCol);
82.     pemesananTableView.getColumns().add(hargaCol);
83.     pemesananTableView.getColumns().add(totalHargaCol);
84.     pemesananTableView.setColumnResizePolicy(TreeTableView.CONSTRAINED_RESIZE_POLICY);
85.     Platform.runLater(() -> {
86.         pemasukanChart.getYAxis().setLabel("Pemasukan (Rp)");
87.         menuFavChart.setStartAngle(90);
88.         kunjunganChart.getYAxis().setLabel("Kunjungan");
89.     });
90.     ObservableList<String> pilihLaporanObservableList =
91.         FXCollections.observableArrayList("Semua", "Pemesanan", "Menu
92.         Favorit", "Pemasukan", "Kunjungan");
93.     pilihLaporanCombo.setItems(pilihLaporanObservableList);
94.     pilihLaporanCombo.getSelectionModel().select(0);
95.     getTransaksiList().addListener(transaksiListChangeListener());
96.     private void pemesanan(LocalDate dari, LocalDate sampai) {
97.         ObservableList<Pesanan> pesananList = FXCollections.observableArrayList();
98.         while (dari.isBefore(sampai.plusDays(1))) {
99.             getTransaksiList(dari).forEach(transaksi -> pesananList.addAll(getPesanan(transaksi)));
100.            dari = dari.plusDays(1);
101.        }
102.        Pesanan.getPesananList().setAll(pesananList);
103.        @SuppressWarnings("unchecked")
104.        private void pemasukan(LocalDate dari, LocalDate sampai) {
105.            XYChart.Series data = new XYChart.Series();
106.            while (dari.isBefore(sampai.plusDays(1))) {
107.                int totalPemasukan =
108.                    getTransaksiList(dari).stream().mapToInt(Transaksi::getTotalBayar).sum();
109.                LocalDate tgl = dari;
110.                Platform.runLater(() -> data.getData().add(
111.                    new XYChart.Data(tgl.getDayOfMonth() + " " + tgl.getMonthOfYear().getAsText(),
112.                        totalPemasukan)));
113.                dari = dari.plusDays(1);
114.            }
115.            Platform.runLater(() -> pemasukanChart.getData().setAll(data));
116.        }
117.        private void menuFavorit(LocalDate dari, LocalDate sampai) {
118.            ObservableList<PieChart.Data> menuFavData = FXCollections.observableArrayList();
119.            ObservableList<Menu> menus = getMenuList();
120.            for (Menu menu : menus) {

```

```

111. AtomicInteger jumlahMenu = new AtomicInteger();
112. LocalDate tgl = dari;
113. while (tgl.isBefore(sampai.plusDays(1))) {
114. for (Transaksi transaksi : getTransaksiList(tgl)) {
115. jumlahMenu.addAndGet(getPesanan(transaksi,
    menu).stream().mapToInt(Pesanan::getJumlah).sum());}
116. tgl = tgl.plusDays(1);}
117. if (jumlahMenu.get() > 0)
118. Platform.runLater() -> menuFavData.add(new PieChart.Data(menu.getNama_menu(),
    jumlahMenu.get()));}
119. Platform.runLater() -> menuFavChart.setData(menuFavData));}
120. @SuppressWarnings("unchecked")
121. private void kunjungan(LocalDate dari, LocalDate sampai) {
122. XYChart.Series data = new XYChart.Series();
123. while (dari.isBefore(sampai.plusDays(1))) {
124. int totalKunjungan = getTransaksiList(dari).size();
125. LocalDate tgl = dari;
126. Platform.runLater() -> data.getData().add(
127. new XYChart.Data(tgl.getDayOfMonth() + " " + tgl.monthOfYear().getAsText(),
    totalKunjungan));
128. dari = dari.plusDays(1);}
129. Platform.runLater() -> kunjunganChart.getData().setAll(data));}
130. private ListChangeListener<Transaksi> transaksiListChangeListener() {
131. return c -> {
132. pemesanan(getDariDate(), getSampaiDate());
133. pemasukan(getDariDate(), getSampaiDate());
134. menuFavorit(getDariDate(), getSampaiDate());
135. kunjungan(getDariDate(), getSampaiDate());};}
136. private LocalDate getDariDate() {
137. if (dariDatePicker.getValue() == null) return new LocalDate().minusMonths(1);
138. else {
139. Date dari =
    Date.from(dariDatePicker.getValue().atStartOfDay(ZoneId.systemDefault()).toInstant()
    );
140. return new LocalDate(dari);}}
141. private LocalDate getSampaiDate() {
142. if (sampaiDatePicker.getValue() == null) return new LocalDate();
143. else {
144. Date sampai =
    Date.from(sampaiDatePicker.getValue().atStartOfDay(ZoneId.systemDefault()).toInstant()
    );
145. return new LocalDate(sampai);}}
146. public void cetakHarianHandle() {
147. Thread thread = new Thread() -> {
148. try {
149. switch (pilihLaporanCombo.getSelectionModel().getSelectedItem()) {
150. case "Semua":
151. Laporan.pemesanan(getDariDate(), getSampaiDate());
152. Laporan.pemasukan(getDariDate(), getSampaiDate());
153. Laporan.menuFavorit(getDariDate(), getSampaiDate());
154. Laporan.kunjungan(getDariDate(), getSampaiDate());
155. break;
156. case "Pemesanan":
157. Laporan.pemesanan(getDariDate(), getSampaiDate());
158. break;

```

```

159. case "Menu Favorit":
160. Laporan.menuFavorit(getDariDate(), getSampaiDate());
161. break;
162. case "Pemasukan":
163. Laporan.pemasukan(getDariDate(), getSampaiDate());
164. break;
165. case "Kunjungan":
166. Laporan.kunjungan(getDariDate(), getSampaiDate());
167. break;}
168. } catch (IOException e) {
169. e.printStackTrace();}});
170. thread.start();}}

```

R. SignInController.java

```

1. package com.unindra.restoserver.controllers;
2. import com.jfoenix.controls.JFXPasswordField;
3. import com.jfoenix.controls.JFXTextField;
4. import com.unindra.restoserver.Dialog;
5. import com.unindra.restoserver.models.User;
6. import javafx.fxml.FXMLLoader;
7. import javafx.scene.Parent;
8. import javafx.scene.Scene;
9. import javafx.stage.Stage;
10. import java.io.IOException;
11. public class SignInController {
12. public JFXTextField usernameField;
13. public JFXPasswordField passwordField;
14. public void signInAction() throws IOException {
15. User user = User.user(usernameField.getText());
16. if (user != null) {
17. if (user.getPassword().equals(passwordField.getText())) {
18. FXMLLoader fxmlLoader = new
        FXMLLoader(getClass().getResource("/fxml/app.fxml"));
19. Parent parent = fxmlLoader.load();
20. ((AppController) fxmlLoader.getController()).setUser(user);
21. getStage().setScene(new Scene(parent));
22. } else {
23. getDialog().information("Error", "Password salah");
24. passwordField.requestFocus();}
25. } else {
26. getDialog().information("Error", "Username tidak ditemukan");
27. usernameField.requestFocus();}}
28. private Stage getStage() {
29. return (Stage) usernameField.getScene().getWindow();}
30. private Dialog getDialog() {
31. return new Dialog(getStage());}}

```

S. UtamaController.java

```

1. package com.unindra.restoserver.controllers;
2. import com.jfoenix.controls.JFXButton;
3. import com.jfoenix.controls.JFXTextField;
4. import com.jfoenix.controls.JFXTreeTableView;
5. import com.jfoenix.controls.RecursiveTreeItem;
6. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
7. import com.unindra.restoserver.Dialog;
8. import com.unindra.restoserver.Laporan;

```

```

9.  import com.unindra.restoserver.models.Pesanan;
10. import com.unindra.restoserver.models.Transaksi;
11. import javafx.application.Platform;
12. import javafx.beans.property.SimpleStringProperty;
13. import javafx.collections.ListChangeListener;
14. import javafx.collections.transformation.FilteredList;
15. import javafx.fxml.Initializable;
16. import javafx.scene.control.TreeItem;
17. import javafx.scene.control.TreeTableCell;
18. import javafx.scene.control.TreeTableColumn;
19. import javafx.scene.control.TreeTableView;
20. import javafx.stage.Stage;
21. import javafx.util.Callback;
22. import java.io.IOException;
23. import java.net.URL;
24. import java.util.ResourceBundle;
25. import java.util.function.Predicate;
26. import static com.unindra.restoserver.models.PesananService.*;
27. import static com.unindra.restoserver.models.Menu.menu;
28. import static com.unindra.restoserver.models.TransaksiService.getTransaksiList;
29. public class UtamaController implements Initializable {
30.     public JFXTreeTableView<Pesanan> pesananTableView;
31.     public JFXTreeTableView<Transaksi> pembayaranTableView;
32.     @Override
33.     public void initialize(URL location, ResourceBundle resources) {
34.         TreeTableColumn<Pesanan, String> mejaCol = new TreeTableColumn<>("No Meja");
35.         TreeTableColumn<Pesanan, String> namaCol = new TreeTableColumn<>("Nama");
36.         TreeTableColumn<Pesanan, Integer> jumlahCol = new
            TreeTableColumn<>("Jumlah");
37.         TreeTableColumn<Pesanan, String> terimaCol = new TreeTableColumn<>("Terima");
38.         TreeTableColumn<Pesanan, String> tolakCol = new TreeTableColumn<>("Tolak");
39.         mejaCol.setCellValueFactory(param ->
            param.getValue().getValue().no_mejaProperty());
40.         namaCol.setCellValueFactory(param ->
            menu(param.getValue().getValue()).nama_menuProperty());
41.         jumlahCol.setCellValueFactory(param ->
            param.getValue().getValue().jumlahProperty());
42.         terimaCol.setCellValueFactory(param -> new SimpleStringProperty(""));
43.         tolakCol.setCellValueFactory(param -> new SimpleStringProperty(""));
44.         namaCol.setCellFactory(new Callback<TreeTableColumn<Pesanan, String>,
            TreeTableCell<Pesanan, String>>() {
45.             @Override
46.             public TreeTableCell<Pesanan, String> call(TreeTableColumn<Pesanan, String>
                param) {
47.                 return new TreeTableCell<Pesanan, String>() {
48.                     @Override
49.                     protected void updateItem(String item, boolean empty) {
50.                         super.updateItem(item, empty);
51.                         if (item == null) {
52.                             setText(null);
53.                         } else {
54.                             Pesanan i = getPesananList().get(getIndex());
55.                             if (menu(i).getTipe().equals("ramen"))
56.                                 setText(item + " lv." + i.getLevel());
57.                             else setText(item);
58.                         }
59.                     }
60.                 };
61.             }
62.         });
63.     }
64. }

```

```

58. terimaCol.setCellFactory(new Callback<TreeTableColumn<Pesanan, String>,
    TreeTableCell<Pesanan, String>>() {
59.     @Override
60.     public TreeTableCell<Pesanan, String> call(TreeTableColumn<Pesanan, String>
        param) {
61.         return new TreeTableCell<Pesanan, String>() {
62.             final JFXButton button = new JFXButton("Terima");
63.             @Override
64.             protected void updateItem(String item, boolean empty) {
65.                 super.updateItem(item, empty);
66.                 if (item == null) {
67.                     setGraphic(null);
68.                     setText(null);
69.                 } else {
70.                     button.getStyleClass().add("terima");
71.                     button.setOnAction(event -> {
72.                         Pesanan i = pesananTableView.getRoot().getChildren().get(getIndex()).getValue();
73.                         i.terima();
74.                         update(i);});
75.                     setGraphic(button);
76.                     setText(null);});});});
77. tolakCol.setCellFactory(new Callback<TreeTableColumn<Pesanan, String>,
    TreeTableCell<Pesanan, String>>() {
78.     @Override
79.     public TreeTableCell<Pesanan, String> call(TreeTableColumn<Pesanan, String>
        param) {
80.         return new TreeTableCell<Pesanan, String>() {
81.             final JFXButton button = new JFXButton("Tolak");
82.             @Override
83.             protected void updateItem(String item, boolean empty) {
84.                 super.updateItem(item, empty);
85.                 if (item == null) {
86.                     setGraphic(null);
87.                     setText(null);
88.                 } else {
89.                     button.getStyleClass().add("tolak");
90.                     button.setOnAction(event -> {
91.                         Dialog alert = new Dialog((Stage) pesananTableView.getScene().getWindow());
92.                         Pesanan i = pesananTableView.getRoot().getChildren().get(getIndex()).getValue();
93.                         alert.confirmation("Anda yakin ingin menolak pesanan ini?",
94.                             e -> {
95.                                 delete(i);
96.                                 alert.getDialog().hide();});});
97.                     setGraphic(button);
98.                     setText(null);});});});
99. Predicate<Pesanan> predicate = item -> item.getStatus_item().equals("dipesan");
100. FilteredList<Pesanan> filteredList = new FilteredList<>(getPesananList(), predicate);
101. getPesananList().addListener((ListChangeListener<Pesanan>) c ->
    filteredList.setPredicate(predicate));
102. TreeItem<Pesanan> rootItem = new RecursiveTreeItem<>(filteredList,
    RecursiveTreeObject::getChildren);
103. pesananTableView.setRoot(rootItem);
104. pesananTableView.getColumns().add(mejaCol);
105. pesananTableView.getColumns().add(namaCol);
106. pesananTableView.getColumns().add(jumlahCol);

```

```

107. pesananTableView.getColumns().add(terimaCol);
108. pesananTableView.getColumns().add(tolakCol);
109. pesananTableView.setColumnResizePolicy(TreeTableView.CONSTRAINED_RESIZE
    _POLICY);
110. TreeTableColumn<Transaksi, String> mejaTransaksiCol = new
    TreeTableColumn<>("No Meja");
111. TreeTableColumn<Transaksi, String> totalCol = new TreeTableColumn<>("Total
    Harga");
112. TreeTableColumn<Transaksi, String> billCol = new TreeTableColumn<>("Bill");
113. TreeTableColumn<Transaksi, String> strukCol = new TreeTableColumn<>("Struk");
114. TreeTableColumn<Transaksi, String> simpanCol = new
    TreeTableColumn<>("Simpan");
115. mejaTransaksiCol.setCellValueFactory(param ->
    param.getValue().getValue().no_mejaProperty());
116. totalCol.setCellValueFactory(param -> param.getValue().getValue().totalProperty());
117. billCol.setCellValueFactory(param -> new SimpleStringProperty(""));
118. strukCol.setCellValueFactory(param -> new SimpleStringProperty(""));
119. simpanCol.setCellValueFactory(param -> new SimpleStringProperty(""));
120. billCol.setCellFactory(new Callback<TreeTableColumn<Transaksi, String>,
    TreeTableCell<Transaksi, String>>() {
121. @Override
122. public TreeTableCell<Transaksi, String> call(TreeTableColumn<Transaksi, String>
    param) {
123. return new TreeTableCell<Transaksi, String>() {
124. final JFXButton button = new JFXButton("Cetak");
125. @Override
126. protected void updateItem(String item, boolean empty) {
127. super.updateItem(item, empty);
128. if (item == null) {
129. setGraphic(null);
130. setText(null);
131. } else {
132. button.getStyleClass().add("print-20");
133. button.setOnAction(event -> {
134. Thread thread = new Thread() -> {
135. Transaksi transaksi = getTransaksiList().get(getIndex());
136. try {
137. Laporan.bill(transaksi);
138. } catch (IOException e) {
139. e.printStackTrace();});
140. thread.start();});
141. setGraphic(button);
142. setText(null);}}});
143. strukCol.setCellFactory(new Callback<TreeTableColumn<Transaksi, String>,
    TreeTableCell<Transaksi, String>>() {
144. @Override
145. public TreeTableCell<Transaksi, String> call(TreeTableColumn<Transaksi, String>
    param) {
146. return new TreeTableCell<Transaksi, String>() {
147. final JFXButton button = new JFXButton("Cetak");
148. @Override
149. protected void updateItem(String item, boolean empty) {
150. super.updateItem(item, empty);
151. if (item == null) {
152. setGraphic(null);

```



```

153. setText(null);
154. } else {
155. button.getStyleClass().add("print-20");
156. button.setOnAction(event -> {
157. Dialog jumlahTunaiDialog = new Dialog((Stage)
    pesananTableView.getScene().getWindow());
158. JFXTextField tunaiField = new JFXTextField();
159. tunaiField.textProperty().addListener((observable, oldValue, newValue) -> {
160. if (!newValue.matches("\\d*")) {
161. tunaiField.setText(newValue.replaceAll("[^\\d]", ""));}});
162. jumlahTunaiDialog.input(
163. tunaiField,
164. e -> {
165. Thread thread = new Thread() -> {
166. Transaksi transaksi = getTransaksiList().get(getIndex());
167. try {
168. int tunai = Integer.parseInt(tunaiField.getText());
169. if (tunai >= transaksi.getTotalBayarFromService())
170. Laporan.struk(transaksi, tunai);
171. else {
172. Platform.runLater() -> {
173. Dialog dialog = new Dialog(
174. (Stage) pesananTableView
175. .getScene().getWindow());
176. dialog.information("Error", "Jumlah tunai tidak mencukupi total pembayaran");}});
177. } catch (IOException ex) {
178. ex.printStackTrace();}});
179. thread.start();
180. jumlahTunaiDialog.getDialog().hide();}});
181. setGraphic(button);
182. setText(null);}}}});
183. simpanCol.setCellFactory(new Callback<TreeTableColumn<Transaksi, String>,
    TreeTableCell<Transaksi, String>>() {
184. @Override
185. public TreeTableCell<Transaksi, String> call(TreeTableColumn<Transaksi, String>
    param) {
186. return new TreeTableCell<Transaksi, String>() {
187. final JFXButton button = new JFXButton("Simpan");
188. @Override
189. protected void updateItem(String item, boolean empty) {
190. super.updateItem(item, empty);
191. if (item == null) {
192. setGraphic(null);
193. setText(null);
194. } else {
195. button.getStyleClass().add("simpan");
196. button.setOnAction(event -> {
197. Dialog confirmDialog = new Dialog((Stage)
    pesananTableView.getScene().getWindow());
198. confirmDialog.confirmation("Transaksi sudah selesai?",
199. e -> {
200. Transaksi transaksi = getTransaksiList().get(getIndex());
201. transaksi.simpan();
202. confirmDialog.getDialog().hide();}}));
203. setGraphic(button);

```

```

204. setText(null);}}}});
205. TreeItem<Transaksi> rootTrans = new RecursiveTreeItem<>(getTransaksiList(),
    RecursiveTreeObject::getChildren);
206. pembayaranTableView.setRoot(rootTrans);
207. pembayaranTableView.getColumns().add(mejaTransaksiCol);
208. pembayaranTableView.getColumns().add(totalCol);
209. pembayaranTableView.getColumns().add(billCol);
210. pembayaranTableView.getColumns().add(strukCol);
211. pembayaranTableView.getColumns().add(simpanCol);
212. pembayaranTableView.setColumnResizePolicy(TreeTableView.CONSTRAINED_RESIZE_POLICY);}}

```

T. DetailRamen.java

```

1. package com.unindra.restoserver.models;
2. import com.unindra.restoserver.DB;
3. import org.sql2o.Connection;
4. import java.util.Arrays;
5. import java.util.List;
6. public class DetailRamen {
7.     private String nama_menu;
8.     private byte[] foto;
9.     private String deskripsi;
10. private DetailRamen(String nama_menu, byte[] foto, String deskripsi) {
11. this.nama_menu = nama_menu;
12. this.foto = foto;
13. this.deskripsi = deskripsi;
14. public DetailRamen(byte[] foto) {
15. this("", foto, "");
16. private static List<DetailRamen> detailRamen() {
17. try (Connection connection = DB.sql2o.open()) {
18. final String query = "SELECT * FROM `detail_ramen`";
19. return connection.createQuery(query).executeAndFetch(DetailRamen.class);
20. public static DetailRamen detailRamen(String nama_menu) {
21. return detailRamen()
22. .stream()
23. .filter(detailRamen -> detailRamen.getNama_menu().equals(nama_menu))
24. .findFirst()
25. .orElse(null);
26. public static DetailRamen detailRamen(Menu menu) {
27. return detailRamen()
28. .stream()
29. .filter(detailRamen -> detailRamen.getNama_menu().equals(menu.getNama_menu()))
30. .findFirst()
31. .orElse(null);
32. public boolean add() {
33. try (Connection connection = DB.sql2o.open()) {
34. final String query =
35. "INSERT INTO `detail_ramen` (`nama_menu`, `foto`, `deskripsi`) " +
36. "VALUES (:nama_menu, :foto, :deskripsi)";
37. connection.createQuery(query).bind(this).executeUpdate();
38. return connection.getResult() > 0;
39. public boolean update() {
40. try (Connection connection = DB.sql2o.open()) {
41. final String query =
42. "UPDATE `detail_ramen` SET `foto` = :foto, `deskripsi` = :deskripsi " +
43. "WHERE `nama_menu` = :nama_menu";

```

```

44. connection.createQuery(query).bind(this).executeUpdate();
45. return connection.getResult() > 0; }}
46. public boolean delete() {
47. try (Connection connection = DB.sql2o.open()) {
48. final String query = "DELETE FROM `detail_ramen` WHERE `nama_menu` =
    :nama_menu";
49. connection.createQuery(query).bind(this).executeUpdate();
50. return connection.getResult() > 0; }}
51. @SuppressWarnings("WeakerAccess")
52. public String getNama_menu() {
53. return nama_menu; }
54. @SuppressWarnings("unused")
55. public byte[] getFoto() {
56. return foto; }
57. @SuppressWarnings("unused")
58. public String getDeskripsi() {
59. return deskripsi; }
60. public void setNama_menu(String nama_menu) {
61. this.nama_menu = nama_menu; }
62. public void setDeskripsi(String deskripsi) {
63. this.deskripsi = deskripsi; }
64. @Override
65. public String toString() {
66. return "DetailRamen{ " + "nama_menu=" + nama_menu + "\" + ", foto="
    + Arrays.toString(foto) + ", deskripsi=" + deskripsi + "\" + '}'"; } }

```

U. Transaksi.java

```

1. package com.unindra.restoserver.models;
2. import com.google.gson.annotations.Expose;
3. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
4. import com.unindra.restoserver.DB;
5. import javafx.beans.property.SimpleStringProperty;
6. import javafx.beans.property.StringProperty;
7. import javafx.collections.FXCollections;
8. import javafx.collections.ObservableList;
9. import org.joda.time.*;
10. import org.sql2o.Connection;
11. import java.util.Date;
12. import java.util.List;
13. import java.util.stream.Collectors;
14. import static com.unindra.restoserver.Rupiah.rupiah;
15. public class Transaksi extends RecursiveTreeObject<Transaksi> {
16. private String id_transaksi;
17. private String no_meja;
18. private Date tanggal;
19. @Expose
20. private static ObservableList<Transaksi> transaksiList =
    FXCollections.observableArrayList();
21. public Transaksi(String no_meja) {
22. this.id_transaksi = Id.getIdByDateTime(new LocalDateTime());
23. this.no_meja = no_meja;
24. this.tanggal = new Date(); }
25. static {
26. Thread thread = new Thread(() -> {
27. while (!Thread.interrupted()) {
28. try {

```

```

29. updateTransaksi();
30. Thread.sleep(5000);
31. } catch (InterruptedException e) {
32. e.printStackTrace();} });
33. thread.start();}
34. public void simpan() {
35. try (Connection connection = DB.sql2o.open()) {
36. final String query =
37. "INSERT INTO `transaksi` (`id_transaksi`,`no_meja`,`tanggal`) " +
38. "VALUES (:id_transaksi,:no_meja,:tanggal)";
39. connection.createQuery(query).bind(this).executeUpdate();
40. if (connection.getResult() > 0) {
41. List<Pesanan> pesanans = PesananService.getItems(this);
42. pesanans.forEach(pesanan -> pesanan.simpan(this));
43. pesanans.forEach(PesananService::delete);
44. TransaksiService.delete(this);} }
45. private static void updateTransaksi() {
46. try (Connection connection = DB.sql2o.open()) {
47. final String query = "SELECT * FROM `transaksi`";
48. transaksiList.setAll(connection.createQuery(query).executeAndFetch(Transaksi.class));}
49. public static ObservableList<Transaksi> getTransaksiList() {
50. return transaksiList;}
51. public static List<Transaksi> getTransaksiList(LocalDate tanggal) {
52. return getTransaksiList()
53. .stream()
54. .filter(transaksi -> new LocalDate(transaksi.getTanggal()).equals(tanggal))
55. .collect(Collectors.toList());}
56. public static List<Transaksi> getTransaksiList(int tahun, int bulan) {
57. return getTransaksiList()
58. .stream()
59. .filter(transaksi -> {
60. LocalDate localDate = new LocalDate(transaksi.getTanggal());
61. return localDate.getYear() == tahun && localDate.getMonthOfYear() == bulan;}
62. .collect(Collectors.toList());}
63. public static Transaksi getTransaksi(Pesanan pesanan) {
64. return getTransaksiList()
65. .stream()
66. .filter(transaksi -> transaksi.getId_transaksi().equals(pesanan.getId_transaksi()))
67. .findFirst()
68. .orElse(null);}
69. public static int getTotalBayar(int tahun, int bulan) {
70. return getTransaksiList(tahun, bulan)
71. .stream()
72. .mapToInt(Transaksi::getTotalBayar)
73. .sum();}
74. public int getTotalBayar() {
75. return Pesanan.getPesanan(this).stream().mapToInt(Pesanan::getTotal).sum();}
76. public int getTotalBayarFromService() {
77. return PesananService.getPesananList().stream()
78. .filter(item -> item.getNo_meja().equals(no_meja))
79. .collect(Collectors.toList()).stream()
80. .mapToInt(Pesanan::getTotal)
81. .sum();}
82. String getId_transaksi() {
83. return id_transaksi;}

```

```

84. public String getNo_meja() {
85. return no_meja; }
86. public Date getTanggal() {
87. return tanggal; }
88. public StringProperty no_mejaProperty() {
89. return new SimpleStringProperty(no_meja); }
90. public StringProperty totalProperty() {
91. return new SimpleStringProperty(rupiah(getTotalBayarFromService())); }
92. public StringProperty pukulProperty() {
93. LocalDateTime t = new LocalDateTime(tanggal);
94. return new SimpleStringProperty(String.format("%d:%d WIB", t.getHourOfDay(),
    t.getMinuteOfHour())); }
95. @Override
96. public String toString() {
97. return "Transaksi{" + "id_transaksi=" + id_transaksi + ", no_meja=" + no_meja + "\" + ",
    tanggal=" + tanggal + '}'"; } }

```

V. User.java

```

1. package com.unindra.restoserver.models;
2. import com.unindra.restoserver.DB;
3. import org.sql2o.Connection;
4. public class User {
5. private String username;
6. private String password;
7. public User(String username, String password) {
8. this.username = username;
9. this.password = password; }
10. public static User user(String username) {
11. try (Connection connection = DB.sql2o.open()) {
12. final String query = "SELECT * FROM `user` WHERE `username` = :username";
13. return connection
14. .createQuery(query)
15. .addParameter("username", username)
16. .executeAndFetchFirst(User.class); } }
17. public String getUsername() {
18. return username; }
19. public String getPassword() {
20. return password; }
21. @Override
22. public String toString() {
23. return "User{" +
24. "username=" + username + "\" +
25. ", password=" + password + "\" +
26. '}'; } }

```