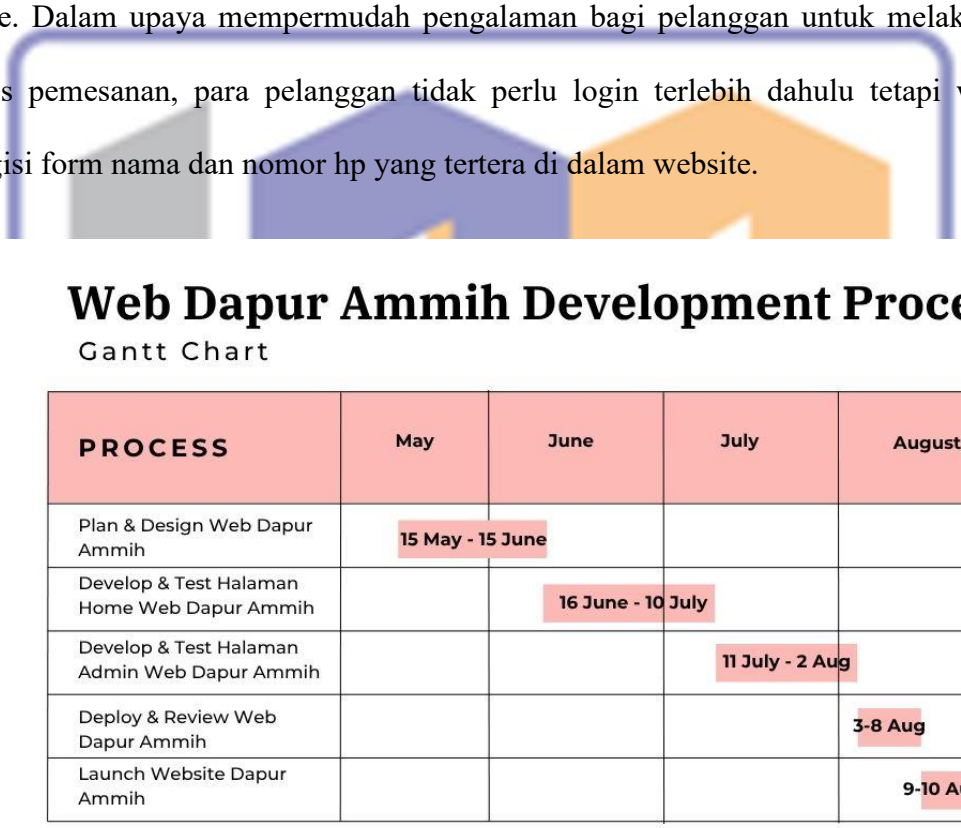
BAB IV
RANCANGAN SISTEM DAN PROGRAM USULAN

4.1 Analisa Kebutuhan Software

A. Tahapan Analisis

Sistem pemesanan makanan adalah sistem yang dirancang untuk mempermudah para pelanggan atau customer mengakses menu digital dan memesan makanan secara online. Dalam upaya mempermudah pengalaman bagi pelanggan untuk melakukan proses pemesanan, para pelanggan tidak perlu login terlebih dahulu tetapi wajib mengisi form nama dan nomor hp yang tertera di dalam website.



Gambar IV.1 Gantt Chart Pengerjaan Web Dapur Ammih

Gambar IV.1 diatas merupakan sebuah timeline proyek website yang penulis kerjakan dalam tugas akhir ini.

Berikut ini spesifikasi kebutuhan (system requirement) dari sistem website yang penulis buat:

Halaman Customer:

A1. Customer dapat mengakses halaman home

A2. Customer dapat memilih menu

A3. Customer dapat memfilter kategori

A4. Customer dapat mengisi form identitas

A5. Customer dapat memesan

A6. Customer dapat melihat invoice

Halaman Admin:

B1. Admin dapat melakukan login

B2. Admin dapat mengelola kategori

B3. Admin dapat mengelola menu

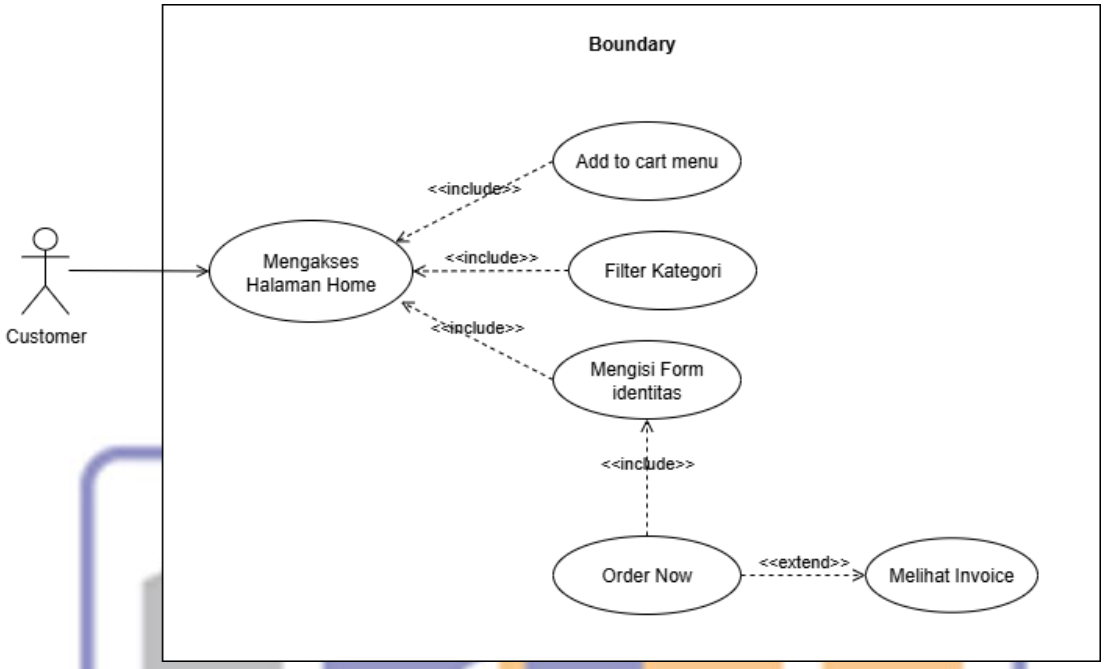
B4. Admin dapat mengelola pesanan customer

B5. Admin dapat mengelola transaksi

B6. Admin dapat edit akun admin

B. Use Case

1. Use case diagram halaman customer.



Sumber:Hasil Penelitian

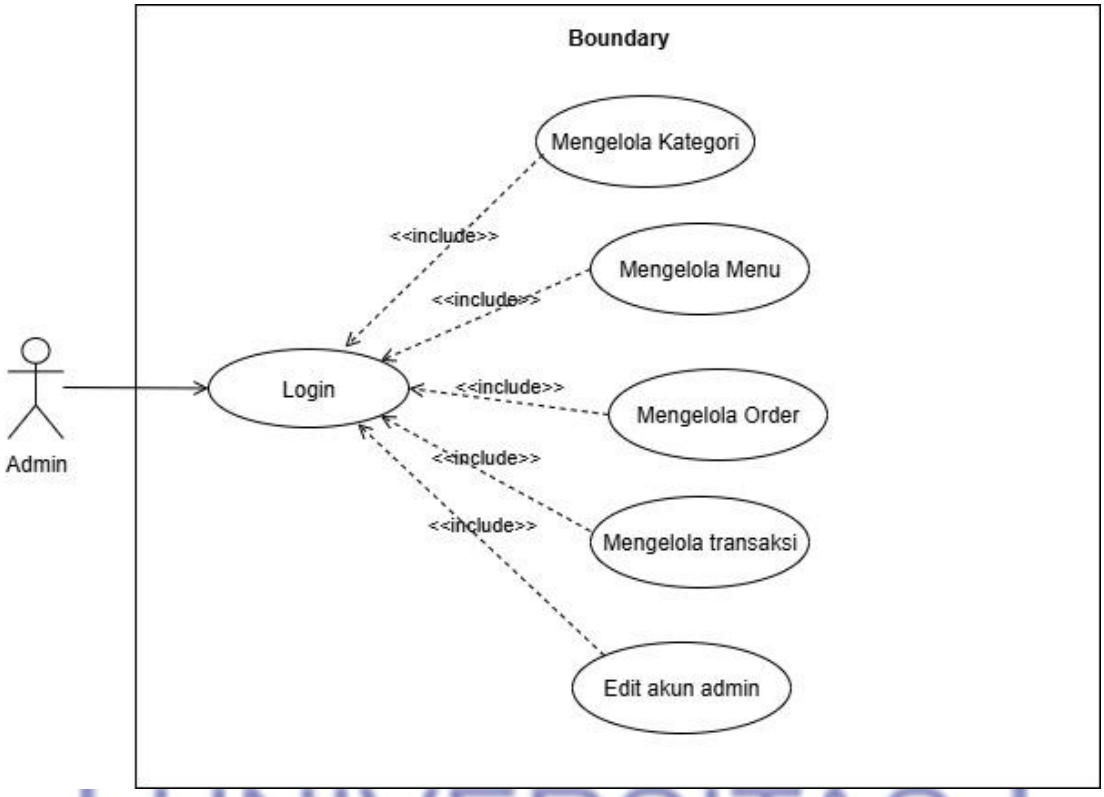
Gambar IV.2 Use Case Diagram Halaman Customer

Tabel IV.1 Deskripsi Use Case Diagram Customer

Use Case Name	Customer
Requirement	A1-A6
Goal	Customer dapat melakukan pemesanan.
Pre-conditions	Customer harus mengisi form data pemesanan agar dapat melakukan pemesanan.
Post-conditions	Customer berhasil melakukan pemesanan setelah mengisi data dan menekan tombol order now.
Failed End conditions	Gagal memesan apabila salah satu form pemesanan kosong.
Primary Actors	Customer
Main Flow / Basic Path	1. Customer dapat mengakses halaman utama 2. Customer memilih menu 3. Customer mengisi data pemesanan

	4. Customer melakukan pemesanan 5. Customer mendapatkan invoice setelah memesan
Invariant	-

2. Use case diagram halaman admin.



Sumber:Hasil Penelitian

Gambar IV.3 Use Case Diagram Halaman Admin

Tabel IV.2 Deskripsi Use Case Diagram Admin Mengelola Kategori

Use Case Name	Admin
Requirement	B2
Goal	Admin dapat mengelola kategori.
Pre-conditions	Admin sudah melakukan login

Post-conditions	Admin dapat menambah, mengubah, dan menghapus kategori.
Failed End conditions	Gagal jika admin lupa mengisi salah satu form.
Primary Actors	Admin
Main Flow / Basic Path	1. Admin menambah, mengubah, atau menghapus kategori
Invariant	-

Tabel IV.3 Deskripsi Use Case Diagram Admin Mengelola Menu

Use Case Name	Admin
Requirement	B3
Goal	Admin dapat mengelola menu.
Pre-conditions	Admin sudah melakukan login
Post-conditions	Admin dapat menambah, mengubah, dan menghapus menu serta menentukan menu apa yang cocok dijadikan best menu.
Failed End conditions	Gagal menambah, mengubah, menghapus menu.
Primary Actors	Admin
Main Flow / Basic Path	1. Admin menambah, mengubah, atau menghapus menu
Invariant	- Stok tidak boleh kosong saat menambah, menghapus, dan mengubah menu

Tabel IV.4 Deskripsi Use Case Diagram Admin Mengelola Order dan Transaksi

Use Case Name	Admin

Requirement	B4-B5
Goal	Admin dapat mengelola pesanan yang dibuat customer dan melihat riwayat pesanan pada halaman transaksi.
Pre-conditions	Admin sudah melakukan login
Post-conditions	Admin dapat melakukan konfirmasi atau cancel pesanan serta dapat melihat semua pesanan yang berhasil pada halaman transaksi.
Failed End conditions	Gagal melakukan perubahan jika sistem mengalami error atau kode pesanan tidak ditemukan.
Primary Actors	Admin
Main Flow / Basic Path	1. Admin melakukan konfirmasi pesanan 2. Tampilan akan dialihkan ke invoice pelanggan 3. Admin dapat melihat semua riwayat pesanan pada halaman transaksi
Invariant	1. Setiap pesanan akan terhubung melalui kode pesanan. 2. Pesanan yang berhasil tidak akan bisa diubah kembali

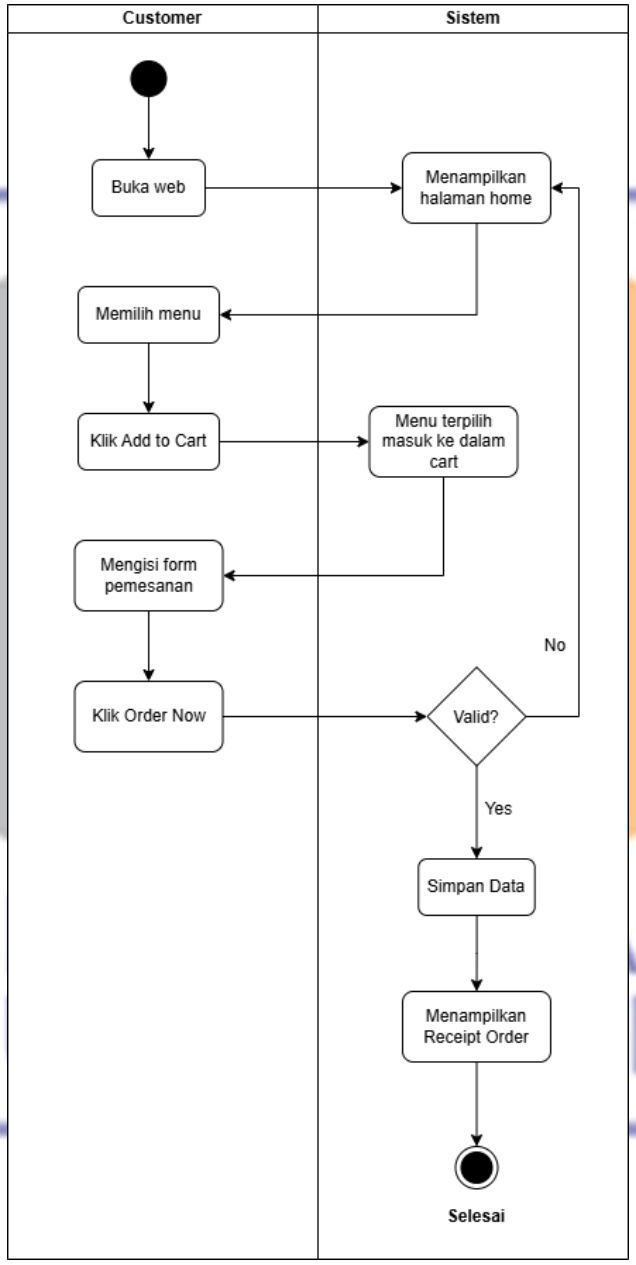
Tabel IV.5 Deskripsi Use Case Diagram Admin Edit Akun Admin

Use Case Name	Admin
Requirement	B6
Goal	Admin dapat mengubah email dan password akun admin.
Pre-conditions	Admin sudah melakukan login
Post-conditions	Admin dapat mengubah email dan password sesuai dengan yang diinginkan.
Failed End conditions	Gagal jika admin lupa mengisi salah satu form.
Primary Actors	Admin
Main Flow / Basic Path	1. Admin mengubah email dan password

Invariant	-

C. Activity Diagram

1. Activity Diagram Memesan

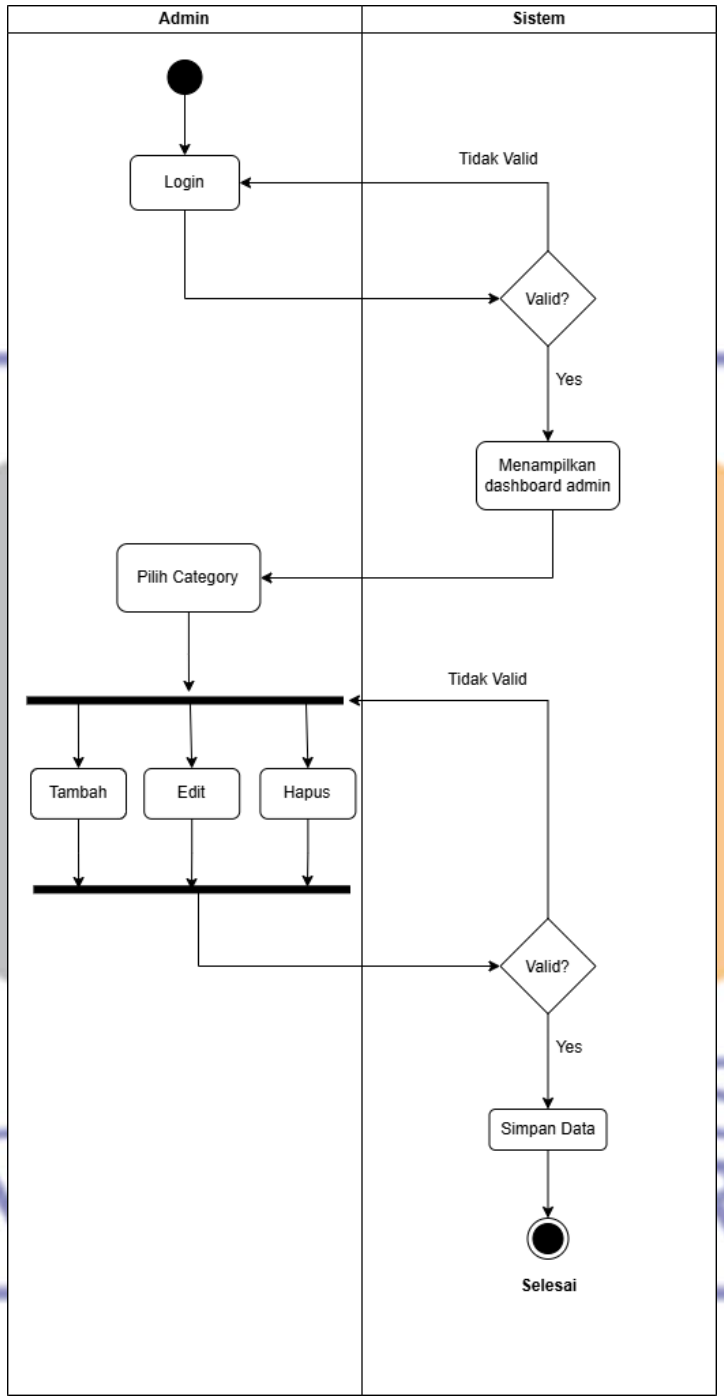


Sumber:Hasil Penelitian

Gambar IV.4 Activity Diagram Memesan

Gambar IV.4 menggambarkan aktivitas customer ketika memesan pada website dan sistem merespon dengan menampilkan halaman sesuai apa yang dilakukan customer.

2. Activity Diagram Admin Mengelola Kategori

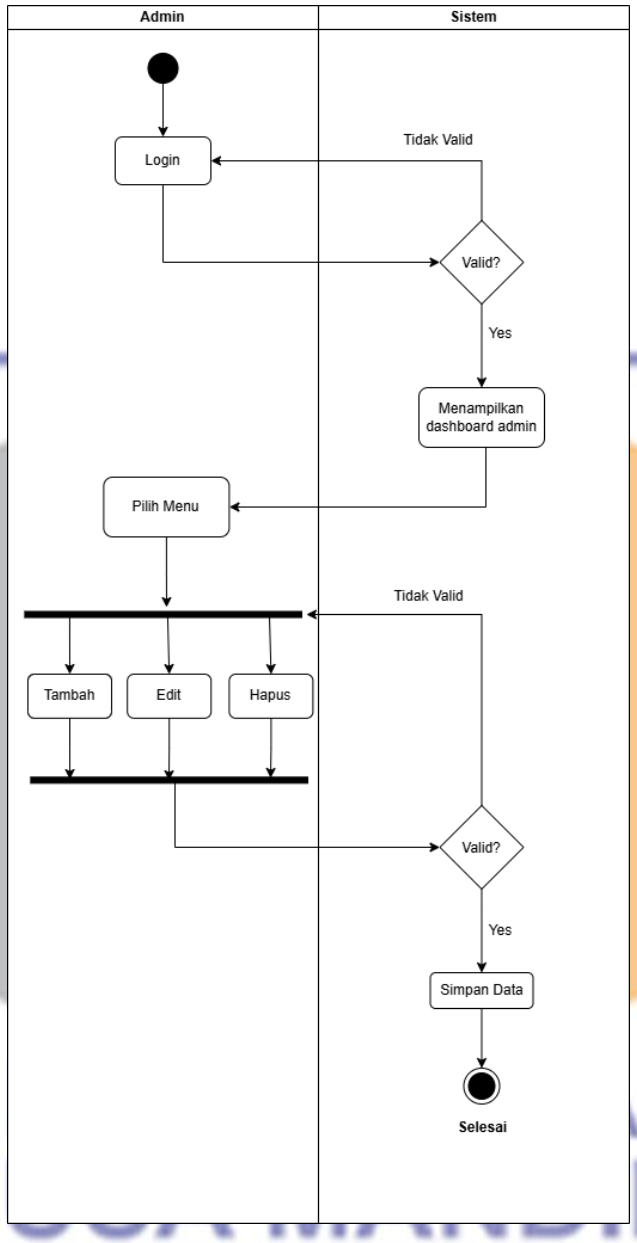


Sumber:Hasil Penelitian

Gambar IV.5 Activity Diagram Admin Mengelola Kategori

Gambar IV.5 menunjukkan aktivitas seorang admin mengelola kategori dengan menambah, mengubah, dan menghapus. Kemudian, sistem akan merespon dengan memvalidasi input tersebut dan jika sesuai maka sistem akan menyimpan data.

3. Activity Diagram Admin Mengelola Menu

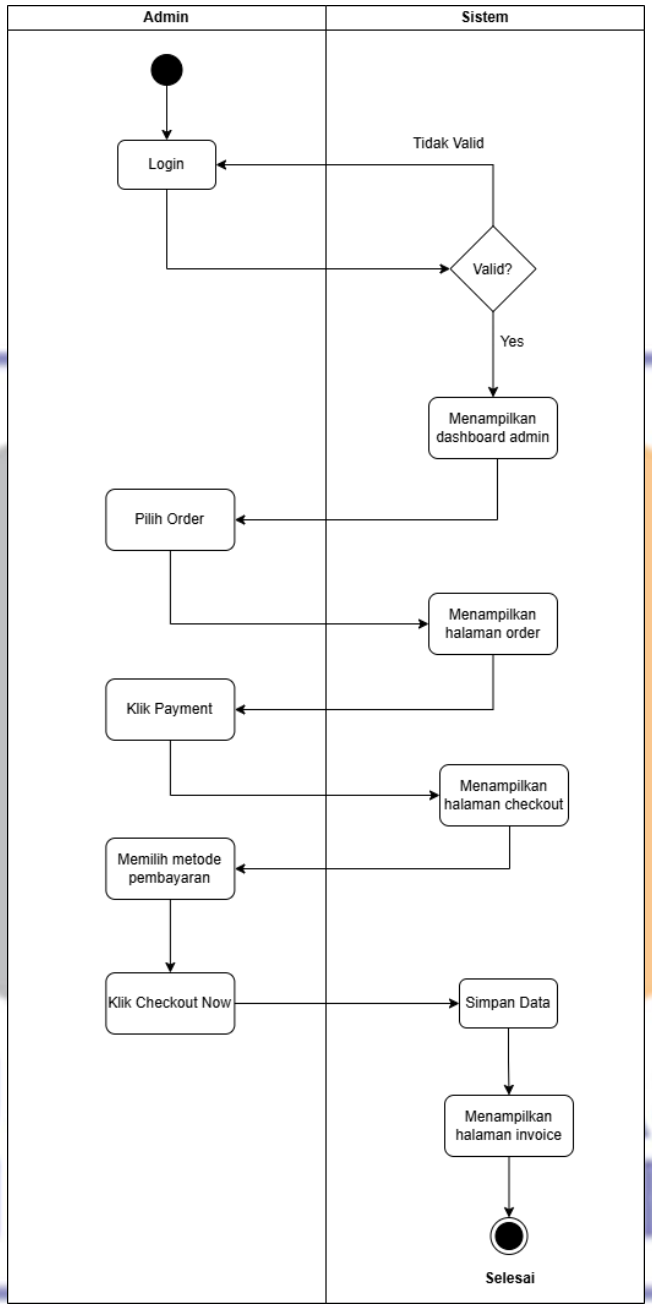


Sumber:Hasil Penelitian

Gambar IV.6 Activity Diagram Admin Mengelola Menu

Gambar IV.6 menggambarkan aktivitas seorang admin mengelola menu dimana inputnya akan divalidasi terlebih dahulu oleh sistem kemudian disimpan datanya apabila sudah sesuai validasi.

4. Activity Diagram Admin Mengelola Order



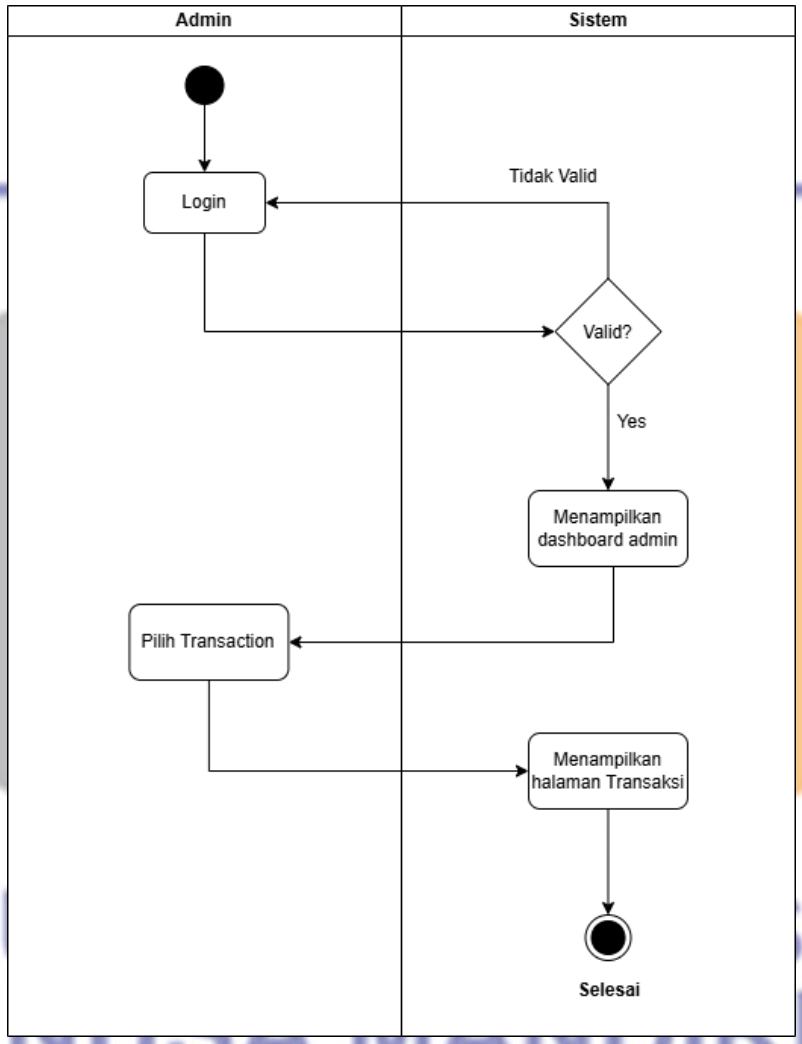
Sumber:Hasil Penelitian

Gambar IV.7 Activity Diagram Admin Mengelola Order

Gambar IV.7 menunjukkan bagaimana seorang admin mengelola order dari customer. Pertama, admin masuk ke halaman order dan klik “payment” pada order yang ada. Kemudian, admin dapat mengkonfirmasi metode pembayaran yang dilakukan oleh

customer dengan klik tombol “check out now”. Sistem akan merespon dengan menampilkan halaman invoice dan data tersimpan.

5. Activity Diagram Admin Mengelola Transaksi

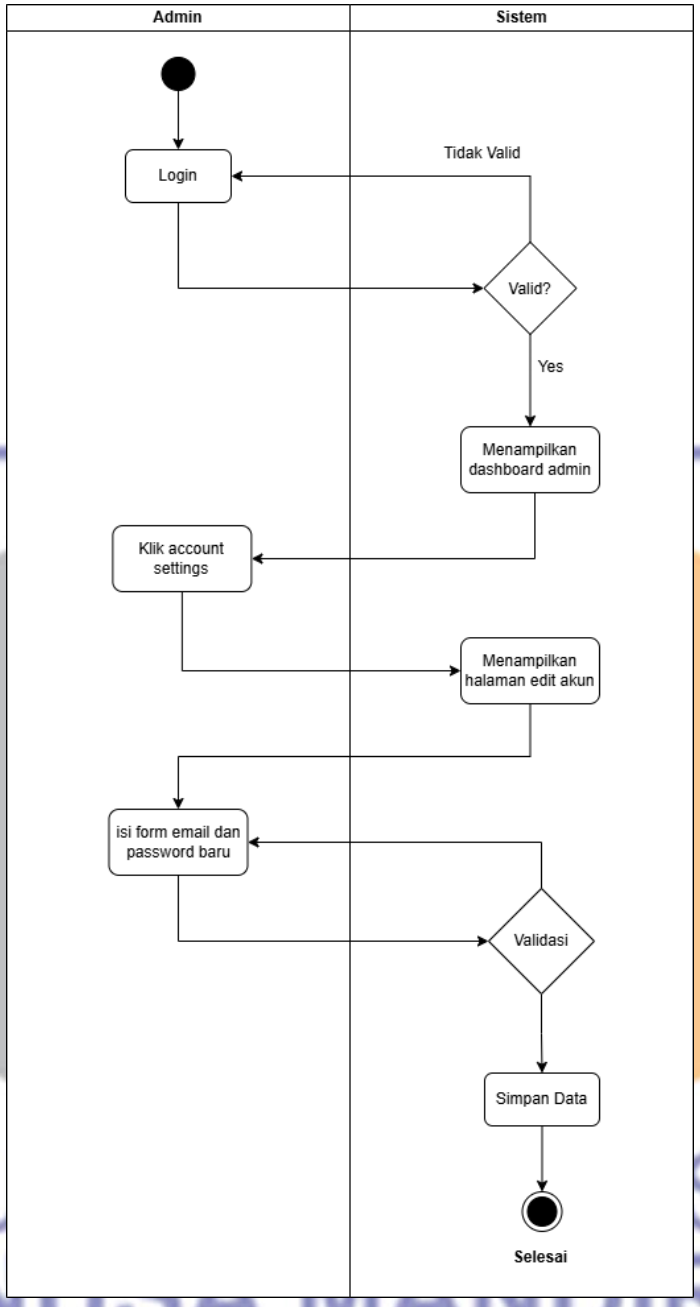


Sumber:Hasil Penelitian

Gambar IV.8 Activity Diagram Admin Mengelola Transaksi

Gambar IV.8 menunjukkan aktivitas admin ketika ingin melakukan pengecekan semua transaksi yang ada dengan melakukan logi terlebih dahulu. Kemudian, sistem merespon dengan menampilkan halaman transaksi.

6. Activity Diagram Admin Mengubah Akun Admin



Sumber:Hasil Penelitian

Gambar IV.9 Activity Diagram Admin Mengubah Akun Admin

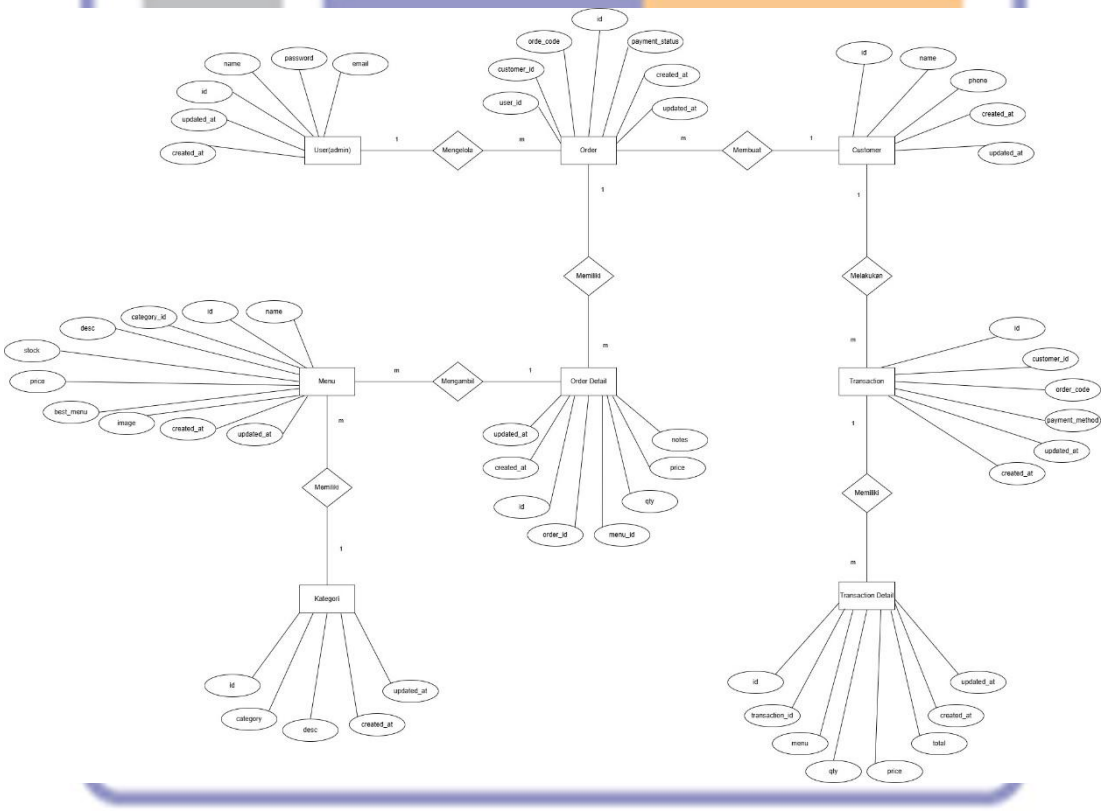
Gambar IV.9 menggambarkan aktivitas admin ketika ingin mengubah email atau password kemudian sistem akan merespon denga validasi data terlebih dahulu dan jika sesuai makan data otomatis tersimpan.

4.2 Desain

4.2.1 Database

Hubungan atau relasi antar data dalam sebuah basis data dapat dibentuk melalui objek-objek utama yang saling terhubung satu sama lain. Objek-objek ini, yang disebut entitas, memiliki hubungan atau keterkaitan tertentu yang direpresentasikan dalam bentuk relasi. Adapun hubungan atau relasi dalam ERD (Entity Relationship Diagram) yang terdapat pada sistem ini digambarkan sebagai berikut:

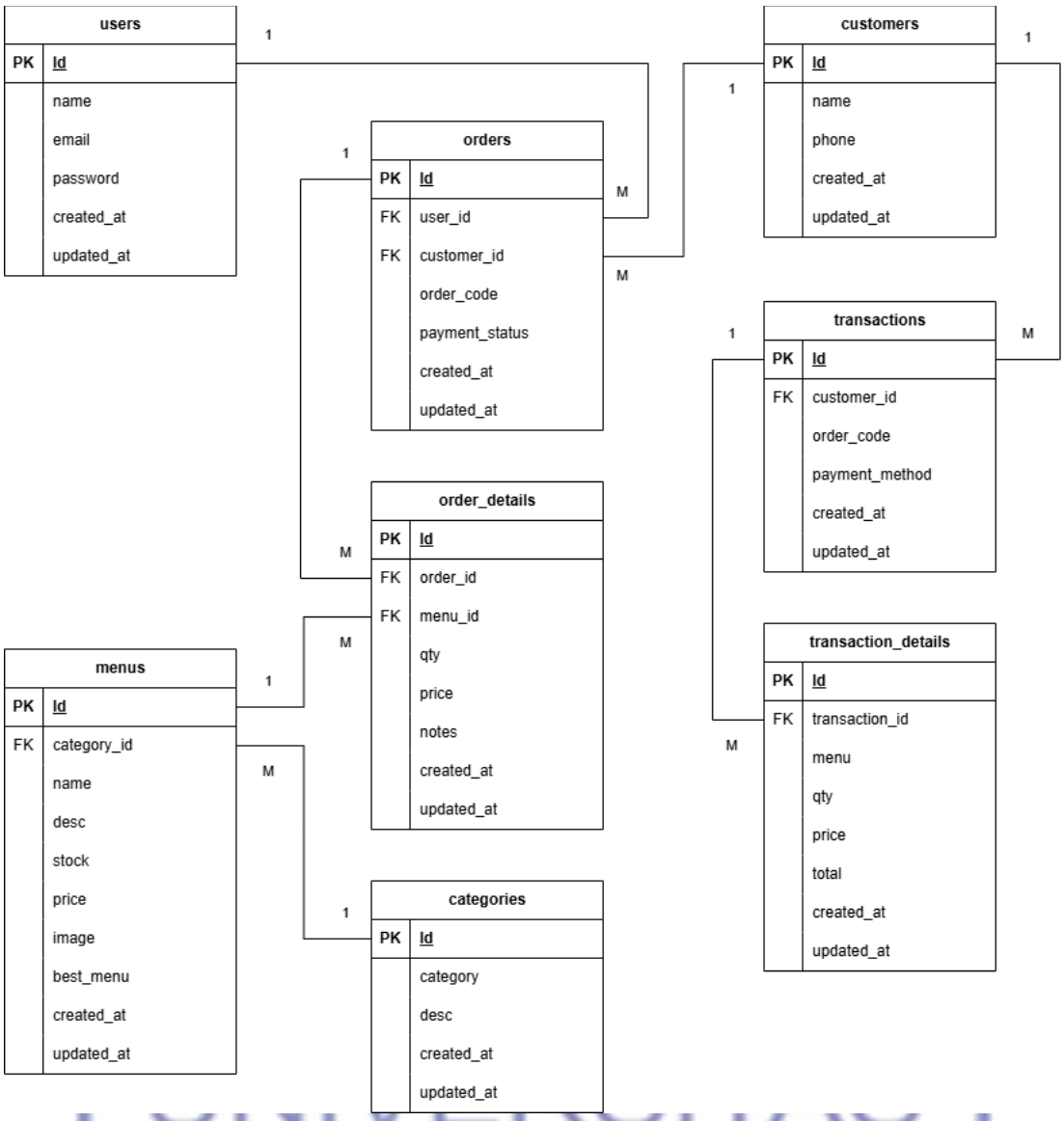
A. Entity Relationship Diagram



Gambar IV.10 Entity Relationship Diagram

Gambar IV.10 menggambarkan rangkaian delapan buah entitas yang saling berhubungan atau memiliki relasi beserta atributnya masing – masing.

B. Logical Record Structure (LRS)



Sumber:Hasil Penelitian

Gambar IV.11 Logical Record Structure

Gambar IV.11 merupakan LRS dari sebuah sistem manajemen restoran yang menunjukkan bagaimana entitas(tabel) dalam database saling berkaitan atau mempunyai relasi.

C. Spesifikasi File

Dalam pembuatan sistem website ini penulis membuat sebuah database bernama "dapurammih" yang memiliki delapan tabel utama yaitu:

1. Spesifikasi File Table Users

Nama Database : dapurammih

Nama File : users

Akronim : users

Tipe File : File Master

Akses File : Random

Panjang Record : 200 Byte

Primary Key : id

Tabel IV.6 Spesifikasi Table Users

No.	Nama Field	Type	Size	Keterangan
1	id	int	20	Primary Key
2	name	varchar	50	
3	email	varchar	30	
4	password	varchar	100	
5	created_at	timestamp	-	
6	updated_at	timestamp	-	

2. Spesifikasi File Table Customers

Nama Database : dapurammih

Nama File : customers

Akronim : customers

Tipe File : File Master

Akses File : Random

Panjang Record : 95 Byte

Primary Key : id

Tabel IV.7 Spesifikasi Table Customers

No.	Nama Field	Type	Size	Keterangan
1	id	int	20	Primary Key
2	name	varchar	50	
3	phone	varchar	25	
4	created_at	timestamp	-	
5	updated_at	timestamp	-	

3. Spesifikasi File Table Orders

Nama Database : dapurammih

Nama File : orders

Akronim : orders

Tipe File : File Master

Akses File : Random

Panjang Record : 210 Byte

Primary Key : id

Tabel IV.8 Spesifikasi Table Orders

No.	Nama Field	Type	Size	Keterangan
1	id	int	20	Primary Key
2	order_code	varchar	150	
3	user_id	int	20	Foreign Key
4	customer_id	int	20	Foreign Key

5	payment_status	enum	-	
6	created_at	timestamp	-	
7	updated_at	timestamp	-	

4. Spesifikasi File Table Order Details

Nama Database : dapurammih

Nama File : order_details

Akronim : order_details

Tipe File : File Master

Akses File : Random

Panjang Record : 300 Byte

Primary Key : id

Tabel IV.9 Spesifikasi Table Order Details

No.	Nama Field	Type	Size	Keterangan
1	id	int	20	Primary Key
2	order_id	int	50	Foreign Key
3	menu_id	int	30	Foreign Key
4	qty	varchar	50	
5	price	varchar	150	
6	notes	text	-	
7	created_at	timestamp	-	
8	updated_at	timestamp	-	

5. Spesifikasi File Table Menus

Nama Database : dapurammih

Nama File : menus

Akronim : menus

Tipe File : File Master

Akses File : Random

Panjang Record : 335 Byte

Primary Key : id

Tabel IV.10 Spesifikasi Table Menus

No.	Nama Field	Type	Size	Keterangan
1	id	int	20	Primary Key
2	name	varchar	50	
3	category_id	int	20	Foreign Key
4	desc	text	-	
5	stock	int	20	
6	price	varchar	100	
7	image	varchar	125	
8	best_menu	enum	-	
9	created_at	timestamp	-	
10	updated_at	timestamp	-	

8. Spesifikasi File Table Categories

Nama Database : dapurammih

Nama File : categories

Akronim : categories

Tipe File : File Master

Akses File : Random

Panjang Record : 120 Byte

Primary Key : id

Tabel IV.11 Spesifikasi Table Categories

No.	Nama Field	Type	Size	Keterangan
1	id	int	20	Primary Key
2	category	varchar	100	
3	desc	text	-	
4	created_at	timestamp	-	
5	updated_at	timestamp	-	

9. Spesifikasi File Table Transactions

Nama Database : dapurammih

Nama File : transactions

Akronim : transactions

Tipe File : File Master

Akses File : Random

Panjang Record : 220 Byte

Primary Key : id

Tabel IV.12 Spesifikasi Table Transactions

No.	Nama Field	Type	Size	Keterangan
1	id	int	20	Primary Key
2	customer_id	varchar	20	Foreign Key
3	order_code	varchar	150	
4	payment_method	varchar	30	
5	created_at	timestamp	-	
6	updated_at	timestamp	-	

10. Spesifikasi File Table Transaction Details

Nama Database : dapurammih

Nama File : trasactions_details

Akronim : transactions_details

Tipe File : File Master

Akses File : Random

Panjang Record : 390 Byte

Primary Key : id

Tabel IV.13 Spesifikasi Table Transaction Details

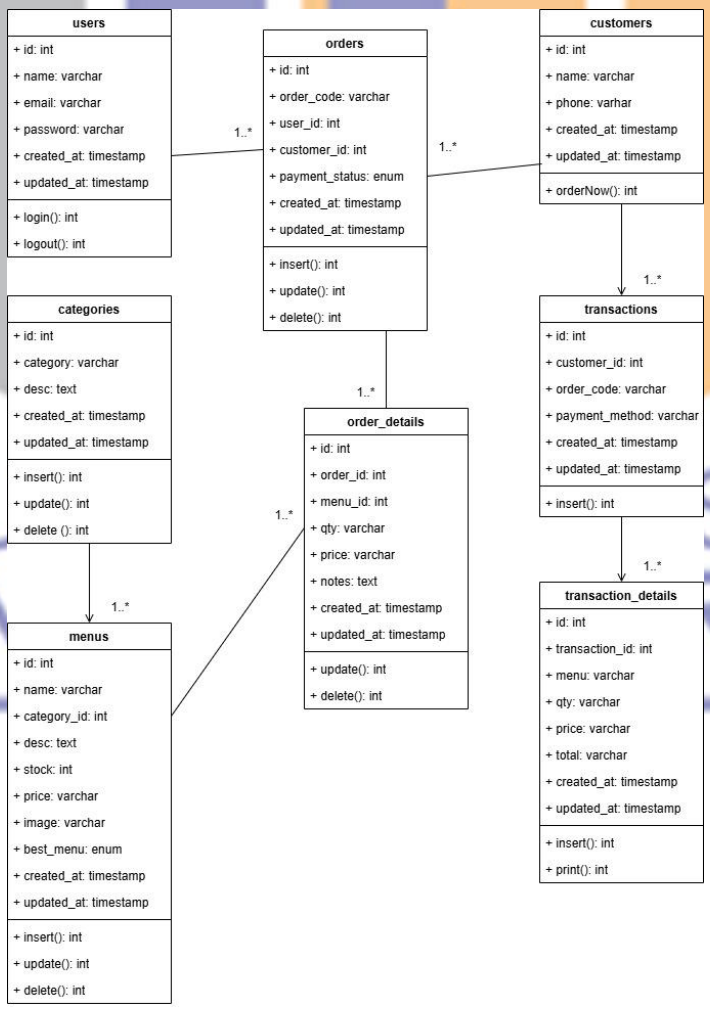
No.	Nama Field	Type	Size	Keterangan
1	id	int	20	Primary Key
2	transaction_details	int	20	Foreign Key
3	menu	varchar	100	
4	qty	varchar	50	
5	price	varchar	100	

6	total	varchar	100	
7	created_at	timestamp	-	
8	updated_at	timestamp	-	

4.2.2 Software Architecture

1. Class Diagram

Class diagram adalah diagram yang menggambarkan struktur sistem dengan menunjukkan kelas-kelas yang ada, atribut dan metodenya, serta hubungan antar kelas seperti asosiasi, generalisasi, dan dependensi.



Sumber:Hasil Penelitian

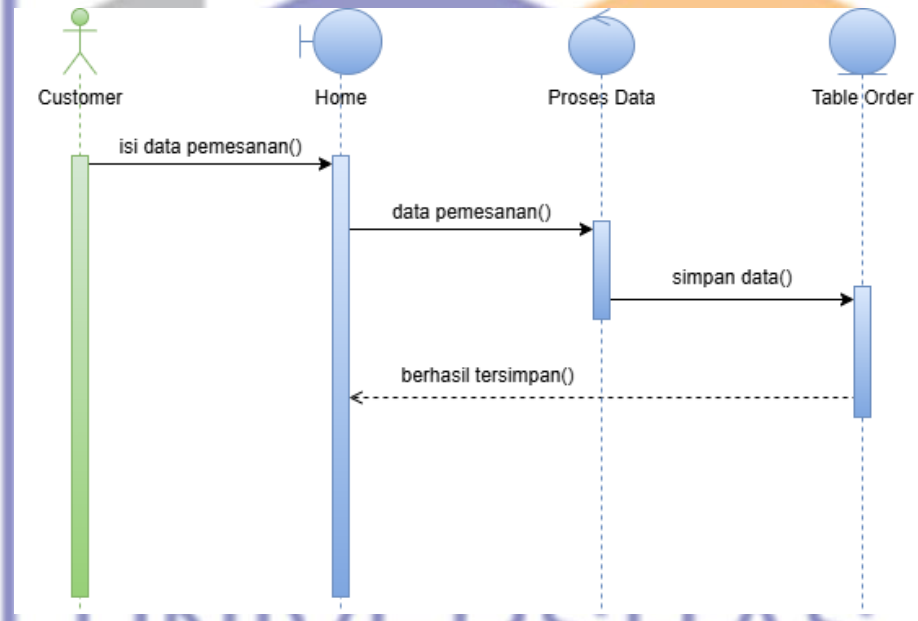
Gambar IV.12 Class Diagram

Gambar IV.12 menunjukkan sebuah class diagram yang digunakan pada sistem manajemen restoran dimana dalam tiap – tiap kelas terdapat field dan juga method().

2. Sequence Diagram

Sequence diagram merupakan jenis diagram yang digunakan untuk menggambarkan serta memperlihatkan alur interaksi antara berbagai komponen dalam suatu sistem secara berurutan berdasarkan waktu.

a. *Sequence Diagram* Customer Memesan

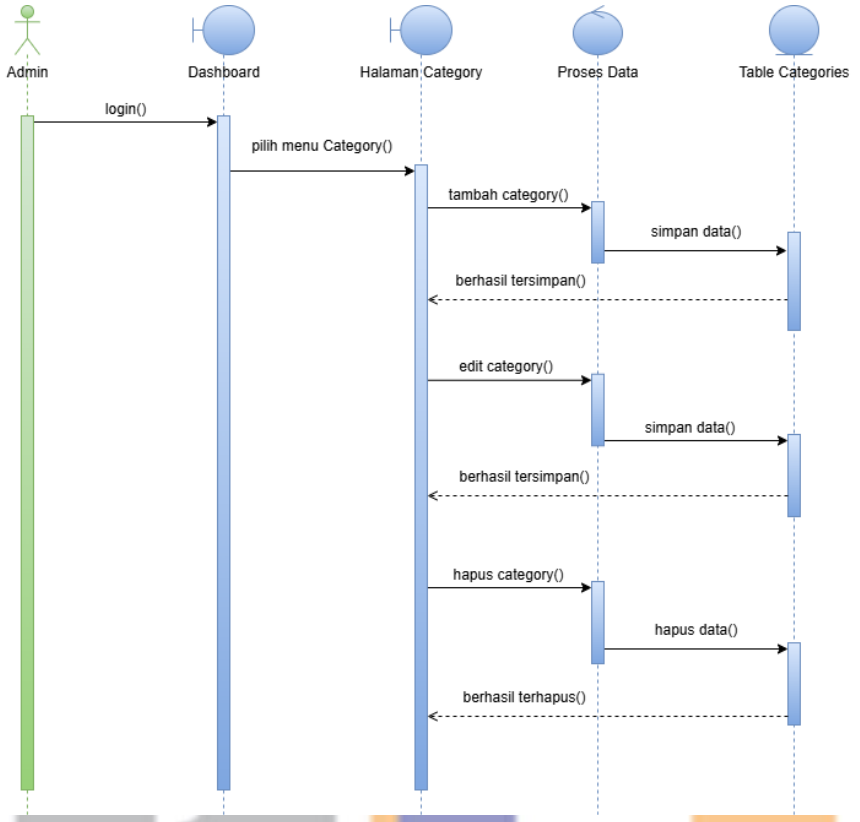


Sumber:Hasil Penelitian

Gambar IV.13 Sequence Diagram Customer

Gambar IV.13 menujukkan sebuah diagram sequence atau alur ketika customer memesan dengan mengisi data pemesanan terlebih dulu. Kemudian, data tersebut tersimpan ke dalam tabel order.

b. *Sequence Diagram* Admin Kelola Kategori

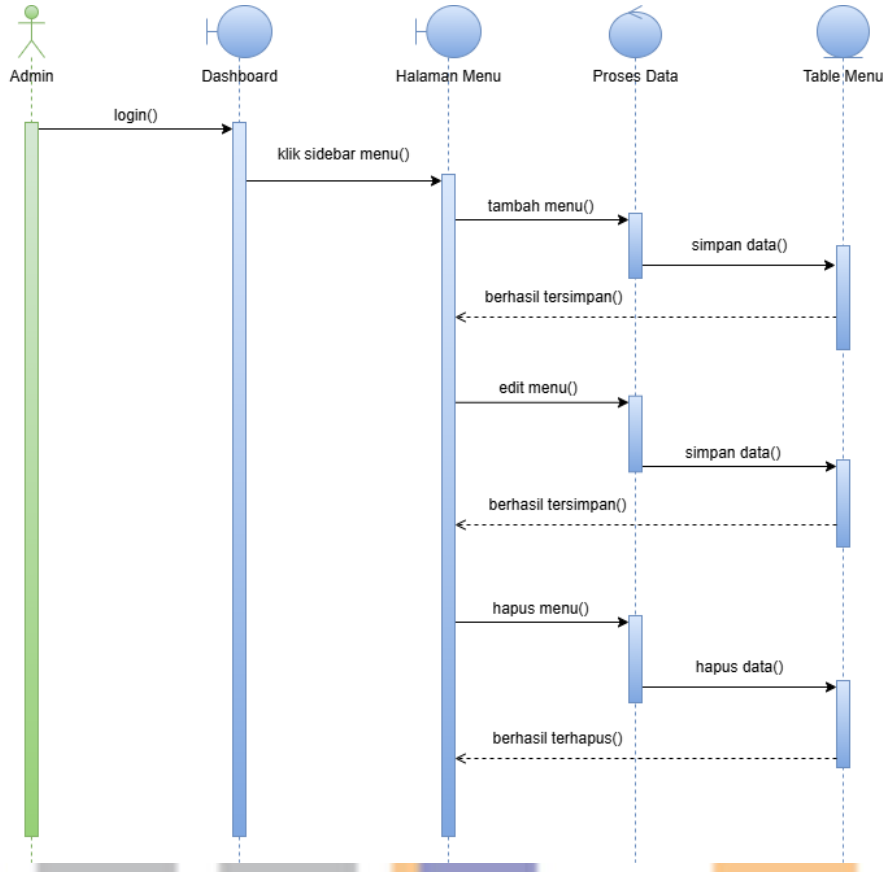


Sumber:Hasil Penelitian

Gambar IV.14 Sequence Diagram Admin Kelola Kategori

Gambar IV.14 menunjukkan urutan seorang admin mengelola kategori dengan melakukan login terlebih dulu. Kemudian, data yang diubah, ditambah, atau dihapus akan disimpan atau dihapus dalam tabel kategori.

c. *Sequence Diagram* Admin Kelola Menu

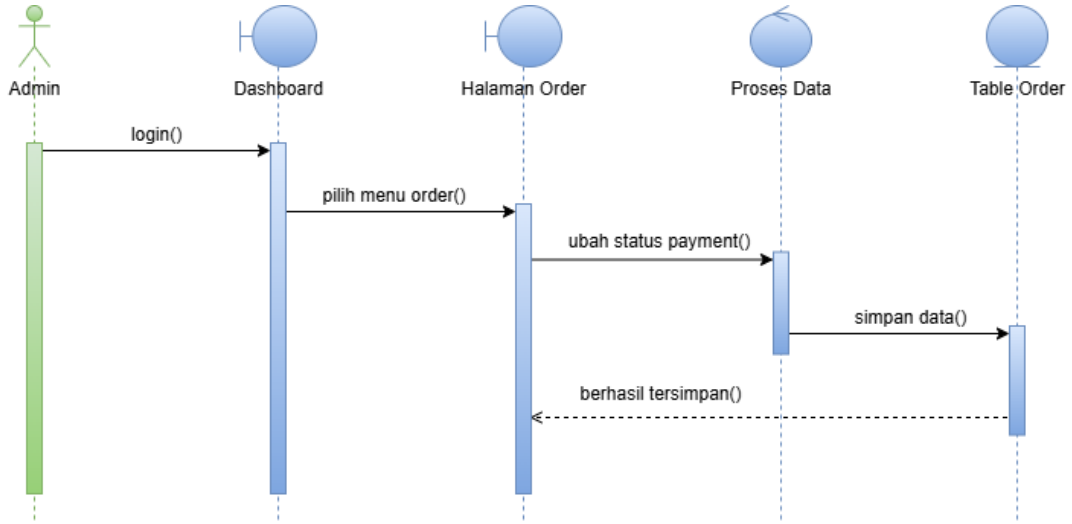


Sumber:Hasil Penelitian

Gambar IV.15 Sequence Diagram Admin Kelola Menu

Gambar IV. 15 menunjukkan diagram urutan ketika admin mengelola menu. Menu yang diubah, ditambah, atau dihapus oleh admin datanya akan direkam pada tabel menu.

d. *Sequence Diagram* Admin Kelola Order

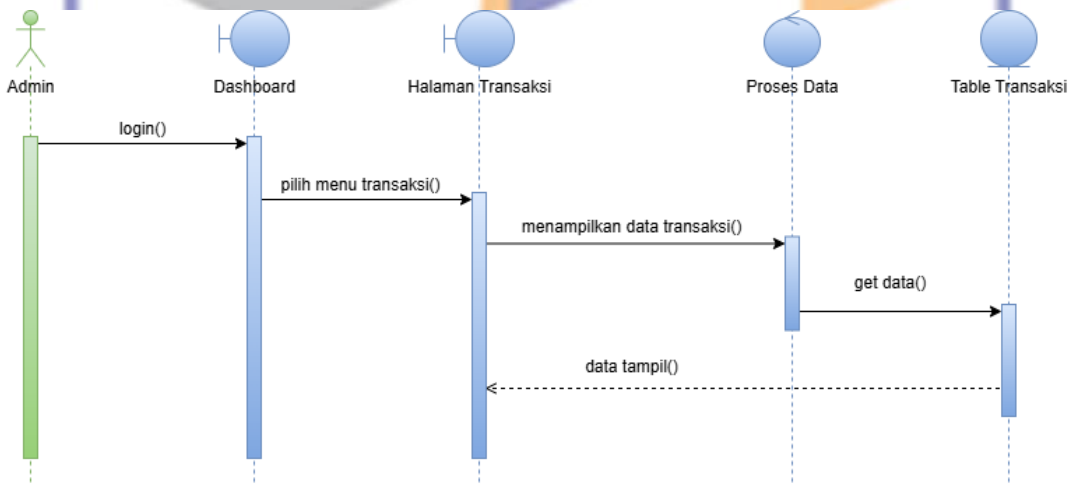


Sumber:Hasil Penelitian

Gambar IV.16 Sequence Diagram Admin Kelola Order

Gambar IV.16 menggambarkan urutan ketika admin megubah status payment untuk kofirmasi pesanan customer. Kemudian, data tersebut berhasil tersimpan dalamm tabel order.

e. *Sequence Diagram* Admin Kelola Transaksi

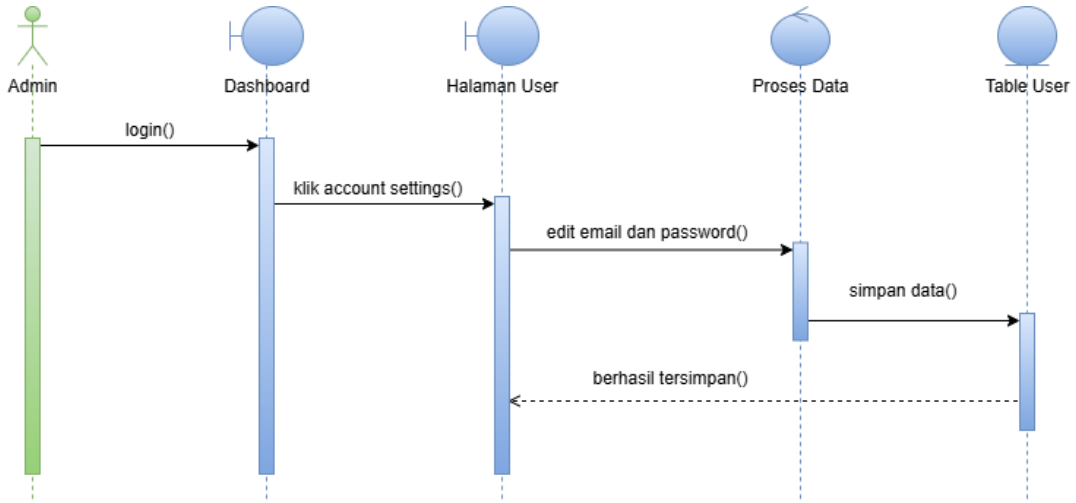


Sumber:Hasil Penelitian

Gambar IV.17 Sequence Diagram Admin Kelola Transaksi

Gambar IV.17 menggambarkan ketika admin ingin melihat semua riwayat transaksi penjualan. Maka, controller proses data akan menghubungkan ke tabel transksi untuk mengambil semua datanya kemudian menampilkan data transaksi ke halaman transaksi.

f. *Sequence Diagram* Admin Ubah Akun



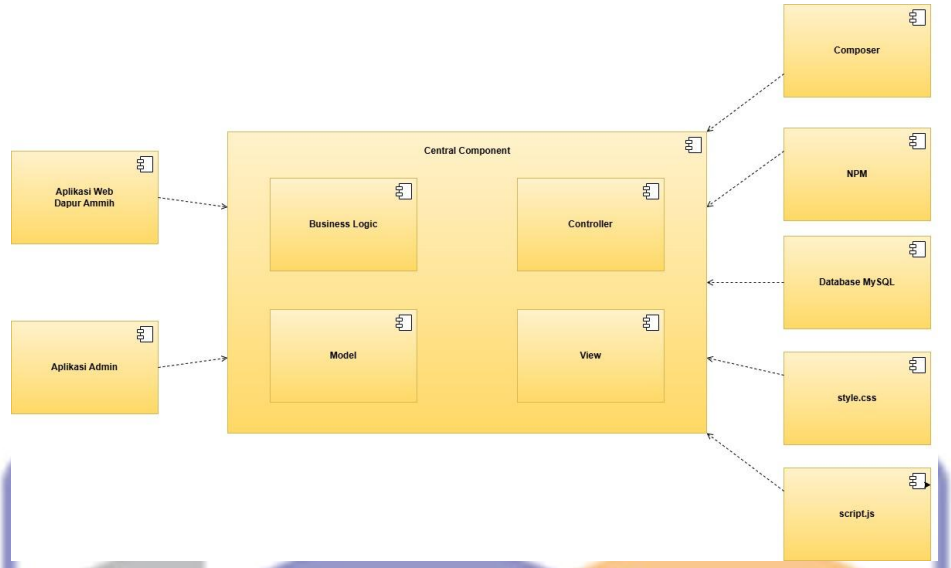
Sumber:Hasil Penelitian

Gambar IV.18 Sequence Diagram Admin Ubah Akun

Gambar IV.18 menggambarkan proses admin mengubah email atau password kemudian controller proses data akan menghubungkan dan data berhasil tersimpan ke tabel user.

3. Component Diagram

Component diagram menunjukkan susunan dan keterkaitan antara berbagai komponen dalam perangkat lunak, termasuk hubungan ketergantungan di antara komponen-komponen tersebut. Diagram ini juga dapat menggambarkan interface, yaitu sekumpulan layanan yang disediakan oleh suatu komponen untuk digunakan oleh komponen lainnya.



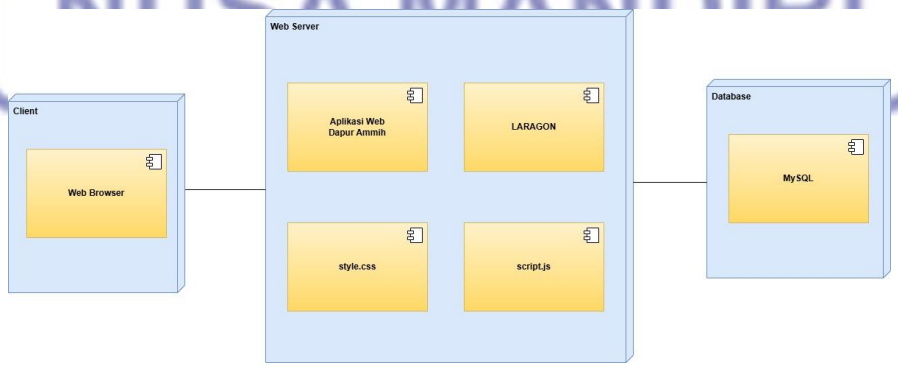
Sumber:Hasil Penelitian

Gambar IV.19 Component Diagram

Gambar IV.19 menunjukkan banyak komponen yang terhubung ke satu central component dimana pada component tersebut sebuah website dapat berjalan sebagai satu kesatuan.

4. Deployment Diagram

Menggambarkan susunan fisik dari sistem, termasuk perangkat lunak yang dijalankan pada perangkat keras, serta menjelaskan bagaimana setiap komponen perangkat keras saling terhubung dan berinteraksi dalam penerapan sistem tersebut.



Sumber:Hasil Penelitian

Gambar IV.20 Deployment Diagram

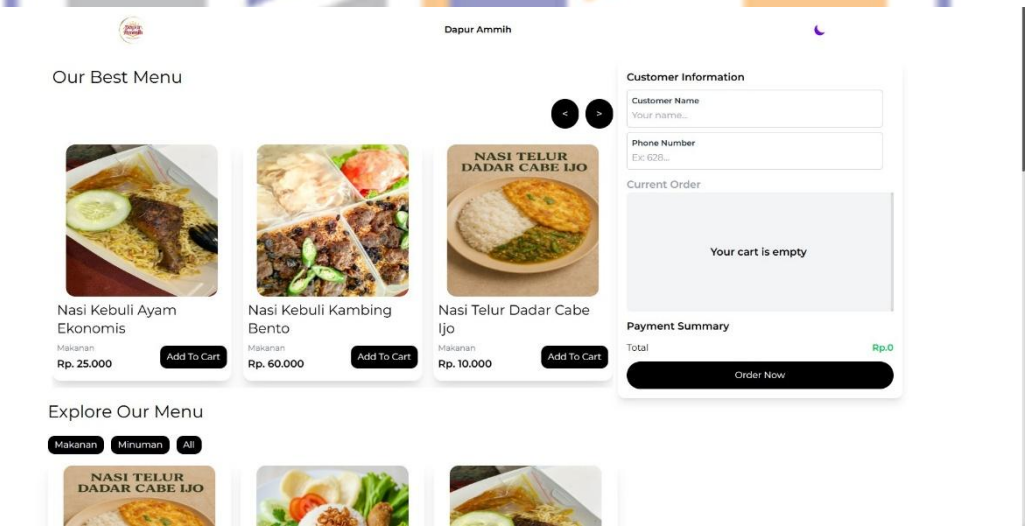
Gambar IV.20 menunjukkan diagram arsitektur dimana terdapat tiga buah lapisan dari sisi client menjalakan web browser untuk mengakses aplikasi web tersebut, web server yang merespon client dengan menampilkan halaman web dapur ammih, dan database yang akan menyimpan semua transaksi yang terjadi dalam website tersebut.

4.2.3 User Interface

Adapun berikut ini merupakan *user interface* dari website sistem pemesanan makanan Dapur Ammih.

a. Tampilan Home (Dapur Ammih Menu)

Adalah sebuah halaman utama dimana customer dapat melihat daftar menu makanan yang tersedia.



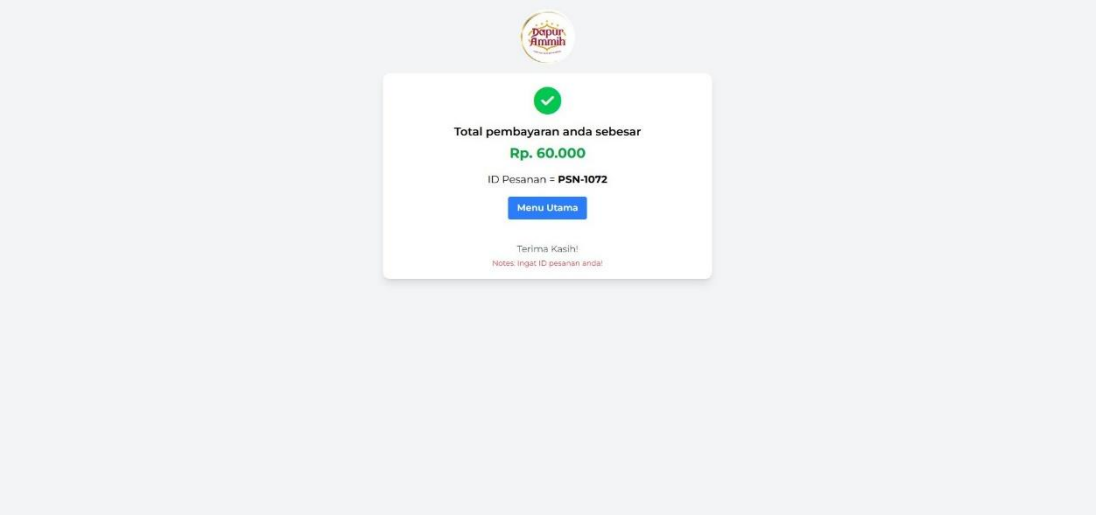
Sumber:Hasil Penelitian

Gambar IV.21 Tampilan Home

Gambar IV.21 merupakan tampilan utama pada website dapur ammih, customer dapat menambahkan menu sesuai keinginan dengan klik tombol “add to cart”, mengisi data pemesanan pada customer information dan klik tombol “order now” untuk memesan.

b. Tampilan Receipt Order (Struk Digital)

Setelah memesan customer diingatkan berapa total biaya yang harus dibayar serta wajib menyebutkan ID pesanannya kepada kasir untuk konfirmasi.



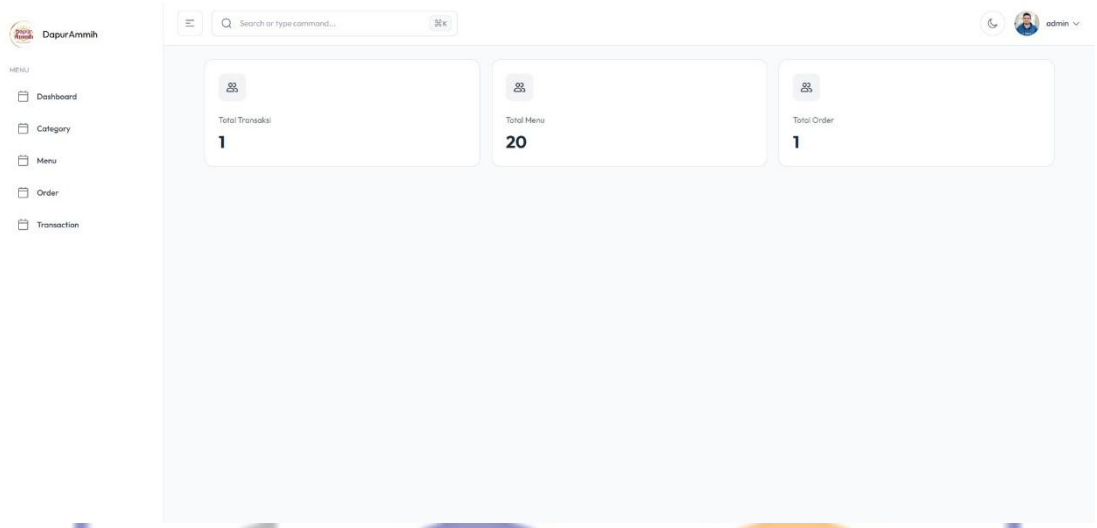
Sumber:Hasil Penelitian

Gambar IV.22 Tampilan Receipt Order

Gambar IV.22 menunjukkan tampilan struk ketika customer berhasil memesan pada tampilan utama.

c. Tampilan Dashboard Admin

Berikut adalah tampilan dimana seorang admin website Dapur Ammih dapat mengelola apapun yang perlu ditampilkan dan menyelesaikan pesanan yang masuk ke dalam sistem website tersebut. Pada halaman ini admin juga dapat meninjau kembali semua transaksi yang sudah berstatus "*done*"



Sumber:Hasil Penelitian

Gambar IV.23 Tampilan Dashboard Admin

Gambar IV.23 menunjukkan tampilan dashboard dimana seorang admin dapat mengelola kategori, menu, order, atau melihat riwayat transaksi dengan mengklik salah satunya pada sidebar menu.

4.3 Code Generation

A. Form Home (Customer memesan)

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@yield('title', "Dapur Ammih")</title>
    @vite('resources/css/app.css')
```

```
    @yield('css')
</head>

<body class="dark:bg-gray-800">
    @include('template.navbar')

    <div class="container mx-auto">
    <div class="md:flex md:h-screen">
      <div class="left flex-1 md:w-2/3">
        @if (count($bestMenus) > 0)
        <section id="best-seller" class="py-3">
          <div class="mx-2">
            <h1 class="text-2xl dark:text-darkMode py-3 md:text-3xl">Our Best Menu</h1>
            <div class="swiper">
              <!-- If we need navigation buttons -->
              <div class="flex justify-end gap-3 py-2">
                <button
                  class="card-button-prev rounded-full bg-black text-white hover:bg-white hover:text-black hover:border dark:hover:border-0 dark:hover:bg-gray-500 dark:bg-gray-700 cursor-pointer p-3 dark:text-darkMode w-12 h-w-12 text-center text-lg">&lt</button>
                <button
                  class="card-button-next rounded-full bg-black text-white hover:bg-white hover:text-black hover:border dark:hover:border-0 dark:hover:bg-
```

```
gray-500 dark:bg-gray-700 cursor-pointer p-3 dark:text-darkMode w-12 h-w-12 text-
center text-lg">&gt</button>
                </div>
                <div class="swiper-wrapper py-3">
                  @foreach ($bestMenus as $bestMenu)
                  <div
                    class="card h-[390px] swiper-slide rounded-2xl overflow-hidden
p-2 dark:bg-[#1E2939] shadow-xl max-w-sm">
                    <img src="{{asset('storage/'.$bestMenu->image)}}" alt="menu"
                      class="rounded-2xl md:w-[266px] md:h-[266px] w-[164px] h-
[164px] m-auto object-fill">
                    <div class="card-body py-2">
                      <div class="card-title md:text-2xl text-[15px] dark:text-
darkMode mb-2">
                        {{$bestMenu->name}}</div>
                      <div class="flex justify-between md:items-center md:flex-row
flex-col gap-2">
                        <div class="flex flex-col gap-1">
                          <div class="dark:text-darkMode md:text-sm text-[12px]
text-gray-500">
                            {{$bestMenu->category->category}}
                          </div>
                          <div class="price dark:text-darkMode md:text-lg text-
[12px] font-semibold">
                            @rupiah($bestMenu->price)
                          </div>
```

```
                    </div>
                    <button
                        class="order bg-black md:text-base text-white md:py-2 md:px-3 md:rounded-xl rounded text-[12px] w-full md:w-auto py-2"
                        onclick="handleOrderNow(this)" data-menu-id="{{$bestMenu->id}}">Add To Cart</button>
                </div>
            </div>
        </div>
        @endforeach
    </div>
</div>
</section>
@endif

<section id="menus">
    <h1 class="text-2xl dark:text-darkMode py-3 md:text-3xl">Explore Our Menu</h1>

    <div class="category py-3 w-full overflow-x-auto flex gap-3">
        @foreach ($categories as $category)
        <a href="{{route('homepage', $category->id)}}" class="bg-black cursor-pointer text-white py-1 px-3 rounded-xl w-fit">{{$category->category}}
        </a>
```

```
            @endforeach
            <a href="/" class="bg-black cursor-pointer text-white py-1 px-3 rounded-xl w-fit">All
            </a>
        </div>

        <div class="grid grid-cols-2 md:grid-cols-3 gap-4">
            @foreach ($menus as $menu)
            <div class="card swiper-slide rounded-2xl overflow-hidden h-[390px] p-2 dark:bg-[#1E2939] shadow-xl max-w-sm">
                <img src="{{asset('storage/'.$menu->image)}}" alt="menu"
                    class="rounded-2xl md:w-[266px] md:h-[266px] w-[164px] h-[164px] m-auto object-fill">
                <div class="card-body py-2">
                    <div class="card-title md:text-2xl text-[15px] dark:text-darkMode mb-2">{{$menu->name}}
                    </div>
                    <div class="flex justify-between md:items-center md:flex-row flex-col gap-2">
                        <div class="flex flex-col gap-1">
                            <div class="dark:text-darkMode md:text-sm text-[12px] text-gray-500">
                                {{$menu->category->category}}
                            </div>
                            <div class="price dark:text-darkMode md:text-lg text-[12px] font-semibold">
```

```
                    @rupiah($menu->price)
                </div>
            </div>
            <button
                class="order bg-black md:text-base text-white md:py-2 md:px-3 md:rounded-xl rounded text-[12px] w-full md:w-auto py-2"
                onclick="handleOrderNow(this)" data-menu-id="{{$menu->id}}">Add To Cart</button>
            </div>
        </div>
    </div>
    @endforeach
</div>
</section>
</div>

<div id="myCart"
    class="w-1/3 my-3 relative bg-white shadow-lg p-4 rounded-xl overflow-y-auto hidden lg:block dark:bg-[#1E2939] h-fit">
    <h2 class="text-lg font-semibold dark:text-darkMode">Customer Information</h2>
    @if ($errors->any())
    <div class="bg-red-500 text-white rounded-lg w-fit px-3 py-1">
        <ul>
            @foreach ($errors->all() as $error)
```

```
                <li>{{ $error }}</li>
                @endforeach
            </ul>
        </div>
        @endif
        <form action="{{route('home.cart.store')}}" method="post">
            @csrf
            <div class="border border-gray-300 rounded-md p-2 w-full max-w-md my-
2">
                <label class="block text-sm font-semibold text-gray-800 mb-1 dark:text-
darkMode">Customer
                    Name</label>
                <input type="text" name="cust_name" placeholder="Your name..."
                    class="w-full text-gray-400 placeholder:text-gray-400 outline-none
bg-transparent" />
            </div>
            <div class="border border-gray-300 rounded-md p-2 w-full max-w-md mb-
2">
                <label class="block text-sm font-semibold text-gray-800 mb-1 dark:text-
darkMode">Phone
                    Number</label>
                <input type="text" name="phone" placeholder="Ex: 628..."
                    class="w-full text-gray-400 placeholder:text-gray-400 outline-none
bg-transparent" />
            </div>
```

```
<input type="hidden" name="cart" id="input_cart">


<div class="mt-3">
  <h2 class="text-lg font-semibold text-darkMode">Current Order</h2>
  <div class="h-52 bg-gray-100 p-3 gap-2 current-order rounded-lg overflow-y-scroll dark:bg-[#242d3c]"
    id="card_cart">
    {{-- <div class="card h-28 bg-white dark:bg-[#1f2939] shadow-xl p-3 rounded-xl my-2">
      <div class="flex w-full h-full relative gap-2">
        <img src="https://images.unsplash.com/photo-1546069901-ba9599a7e63c?q=80&w=1480&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"
          alt="order-thumb" class="rounded-lg object-cover w-1/3">

        <div class="dark:text-darkMode">
          <h3 class="text-lg">Salad</h3>
          <div class="price">Rp.1.300.000</div>
          <div class="qty flex">
            <button
              class="minus inline-flex items-center justify-center w-10 h-10 bg-white rounded-full">-</button>
            <input type="number" value="1" name="qty" min="0" class="text-center w-1/2">
            <button
```

```
                    class="minus inline-flex items-center justify-center w-10
h-10 bg-white rounded-full">+</button>
                </div>
            </div>
            <div
                class="absolute top-0 right-0 bg-red-500 py-1 px-4 cursor-
pointer text-white rounded-lg">
                <i class="fa-solid fa-trash"></i>
            </div>
        </div>
    </div> --}}

    {{-- CARD SHOW HERE --}}
</div>

<div class="dark:text-darkMode my-3">
    <h2 class="text-lg font-semibold mb-3">Payment Summary</h2>
    <div class="flex justify-between">
        <div class="total">Total</div>
        <div class="font-semibold text-green-500"
id="cart_total">Rp.0</div>
    </div>
</div>
```

```
                <button class="w-full bg-black p-3 dark:text-darkMode rounded-full text-white cursor-pointer" onclick="localStorage.removeItem('cartItems')">Order Now</button>
            </div>
        </form>
        <div id="closeButton" onclick="showCart()" class="top-0 right-0 absolute p-3 sm:hidden">
            &#10006;
        </div>
    </div>
  </div>
</div>

{{-- MODAL BOX --}}
<div id="cart-modal" tabindex="-1"
  class="hidden overflow-y-auto overflow-x-hidden fixed top-0 right-0 left-0 z-50 justify-center items-center w-full md:inset-0 h-[calc(100%-1rem)] max-h-full">
  <div class="relative p-4 w-full max-w-2xl max-h-full">
    <!-- Modal content -->
    <div class="relative bg-white rounded-lg shadow-sm dark:bg-gray-700">
      <!-- Modal header -->
      <div
        class="flex items-center justify-between p-4 md:p-5 border-b rounded-t dark:border-gray-600 border-gray-200">
        <button type="button" id="cart-modal-close"
```

```
class="text-gray-400 bg-transparent hover:bg-gray-200 hover:text-gray-900 rounded-lg text-sm w-8 h-8 ms-auto inline-flex justify-center items-center dark:hover:bg-gray-600 dark:hover:text-white">
    <svg class="w-3 h-3" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 14 14">
        <path stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2"
            d="m1 1 6 6m0 0 6 6M7 7l6-6M7 7l-6 6" />
    </svg>
    <span class="sr-only">Close modal</span>
  </button>
</div>
<!-- Modal body -->
<div class="p-4 md:p-5 space-y-4 dark:text-white">
  <input type="hidden" value="123" id="menu_input" readonly>
  <div class="input-group">
    <div class="flex justify-between items-center">
      <label class="py-2">Notes:</label>
      <small>*Optional</small>
    </div>
    <textarea name="desc" rows="5" id="notes_input"
      class="dark:bg-dark-900 shadow-theme-xs focus:border-brand-300 focus:ring-brand-500/10 dark:focus:border-brand-800 rounded-lg border border-gray-200 bg-transparent py-2.5 pr-14 pl-3 text-sm text-gray-800 placeholder:text-gray-400 focus:ring-3 focus:outline-hidden dark:border-gray-800 dark:bg-white/[0.03] dark:text-white/90 dark:placeholder:text-white/30 w-full"></textarea>
```

```
            </div>
            <!-- Modal footer -->
            <div class="flex items-center p-4 md:p-5 border-t border-gray-200 rounded-b dark:border-gray-600">
                <button type="button" id="addToCartButton"
                    class="text-white bg-blue-700 hover:bg-blue-800 focus:ring-4 focus:outline-none focus:ring-blue-300 font-medium rounded-lg text-sm px-5 py-2.5 text-center dark:bg-blue-600 dark:hover:bg-blue-700 dark:focus:ring-blue-800">Add
                    To Cart</button>
            </div>
        </div>
    </div>
</div>
{{-- END OF MODAL BOX --}}

<div onclick="showCart()"
    class="fixed bottom-5 right-5 z-10 w-12 h-12 rounded-full inline-flex items-center justify-center dark:bg-[#101828] shadow-2xl sm:hidden">
    <i class="fa-solid fa-basket-shopping dark:text-white"></i>
</div>
@endsection


@push('scripts')
<script src="https://cdn.jsdelivr.net/npm/swiper@11/swiper-bundle.min.js"></script>
```

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/js/all.min.js"
    integrity="sha512-
b+nQTCdtTBIRIbraqNEwsjB6UvL3UEMkXnhzd8awtCYh0Kcsjl9uEgwVFVbhoj3
uu1DO1ZMacNvLoyJJiNfcvg=="
    crossorigin="anonymous" referrerpolicy="no-referrer"></script>
{{-- <script src="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.7.2/js/conflict-detection.min.js" integrity="sha512-
K5+wFlOsuophh3a/Im5Xc/i3w5YnOJmfTS74GUNHUH0UPe2Y55Y8iLkFkLjYy6
aJy999ZIoKjsmFJMhjq3MlHQ==" crossorigin="anonymous" referrerpolicy="no-
referrer"></script> --}}
<script>
    document.addEventListener("DOMContentLoaded", function () {
        renderCart();
    });

</script>
<script>
    const swiper = new Swiper('.swiper', {
        loop: true,
        spaceBetween: 20,
        navigation: {
            nextEl: '.card-button-next',
            prevEl: '.card-button-prev',
        },

        breakpoints: {
```

```
            0: {
                slidesPerView: 2
            },
            768: {
                slidesPerView: 2
            },
            1024: {
                slidesPerView: 3
            },
        }
    })

</script>

<script>
    function showCart() {
        const myCart = document.querySelector('#myCart');
        // const closeCart = document.querySelector('#closeButton')
        myCart.classList.toggle('active')
    }

    let mouseDown = false;
    let startX, scrollLeft;
    const slider = document.querySelector('.category');
```

```
const startDragging = (e) => {
    mouseDown = true;
    startX = e.pageX - slider.offsetLeft;
    scrollLeft = slider.scrollLeft;
}

const stopDragging = (e) => {
    mouseDown = false;
}

const move = (e) => {
    e.preventDefault();
    if (!mouseDown) {
        return;
    }
    const x = e.pageX - slider.offsetLeft;
    const scroll = x - startX;
    slider.scrollLeft = scrollLeft - scroll;
}

// Add the event listeners
slider.addEventListener('mousemove', move, false);
slider.addEventListener('mousedown', startDragging, false);
```

```
slider.addEventListener('mouseup', stopDragging, false);
slider.addEventListener('mouseleave', stopDragging, false);

// function closeCart(){
//     const myCart = document.querySelector('#myCart');
//     // const closeCart = document.querySelector('#closeButton')
//     myCart.classList.toggle('active')
// }

async function handleOrderNow(button) {
    const menuId = button.getAttribute('data-menu-id');
    const menu_input = document.querySelector('#menu_input');
    const addToCartButton = document.querySelector('#addToCartButton');
    const modal = new Modal(document.getElementById('cart-modal'));

    menu_input.value = menuId;
    const {
        data
    } = await getData(menuId);

    addToCartButton.onclick = () => handleAddToCart(data, modal);

    const closeButton = document.querySelector('#cart-modal-close').addEventListener('click', function () {
```

```
        modal.hide();
    });

    modal.show();
}

async function getData(id) {
    let url = "{{ route('home.getMenu', ['id' => ':id']) }}".replace(':id', id);
    const menu = await fetch(url).then(res => {
        return res.json();
    });
    return menu;
}

function handleAddToCart(data, modal) {
    // console.log(data)
    const notesInput = document.querySelector('#notes_input');
    data['notes'] = notesInput.value;
    // localStorage.setItem('cartItems', JSON.stringify([data]));
    let cart = JSON.parse(localStorage.getItem('cartItems')) || [];

    const existingIndex = cart.findIndex(item => item.id === data.id);
    if (existingIndex !== -1) {
        cart[existingIndex].quantity += 1;
```

```
    } else {
        cart.push({
            ...data,
            quantity: 1
        });
    }

    // Save updated cart
    localStorage.setItem('cartItems', JSON.stringify(cart));
    renderCart();
    notesInput.value = '';
    modal.hide();
}

function setupDeleteButtons() {
    const deleteButtons = document.querySelectorAll('#delete-btn');
    deleteButtons.forEach(button => {
        button.addEventListener('click', function () {
            const id = this.getAttribute('data-id');
            deleteCartItem(id);
        });
    });
}
```

```
function deleteCartItem(id) {
    let cart = JSON.parse(localStorage.getItem('cartItems')) || [];
    cart = cart.filter(item => item.id != id);
    localStorage.setItem('cartItems', JSON.stringify(cart));
    renderCart();
}

function renderCart() {
    const cardCart = document.querySelector('#card_cart');
    const inputCart = document.querySelector('#input_cart');
    const storedData = localStorage.getItem("cartItems");
    const items = storedData ? JSON.parse(storedData) : [];
    const ASSET_URL = "{{ asset('storage') }}";
    cardCart.innerHTML = '';

    inputCart.value = JSON.stringify(items);

    if (items.length === 0) {
        cardCart.innerHTML = `
            <h4 class="dark:text-white md:text-lg font-semibold w-full h-full flex justify-center items-center">Your cart is empty</h4>
        `;
        updateTotal(0);
        return;
```

```
}

let total = 0;
items.forEach(item => {
    total += item.price * item.quantity;

    const card = document.createElement("div");
    card.className = "card h-28 bg-white dark:bg-[#1f2939] shadow-xl p-3 rounded-xl mb-2";

    card.innerHTML = `
        <div class="flex w-full h-full relative gap-2">
            <img src="${ASSET_URL}/${item.image}" alt="order-thumb" class="rounded-lg object-cover w-1/3">

            <div class="dark:text-darkMode">
                <h3 class="text-lg">${item.name}</h3>
                <div class="price">Rp.${item.price.toLocaleString('id-ID')}</div>
                <div class="qty flex">
                    <button class="minus inline-flex items-center justify-center w-10 h-10 bg-white rounded-full" id="qty_button" data-action="decrease" data-id="${item.id}">-</button>
                    <input type="number" value="${item.quantity}" min="0" class="text-center w-1/2">
```

```
                    <button class="plus inline-flex items-center justify-center w-10 h-10 bg-white rounded-full" id="qty_button" data-action="increase" data-id="${item.id}">+</button>
                </div>
            </div>

            <div class="absolute top-0 right-0 bg-red-500 py-1 px-4 cursor-pointer text-white rounded-lg" id="delete-btn" data-id="${item.id}">
                <i class="fa-solid fa-trash"></i>
            </div>
        </div>
    `;

    cardCart.appendChild(card);
});

setupDeleteButtons();
setQtyButton();
updateTotal(total)
}

function setQtyButton() {
    const qtyButton = document.querySelectorAll('#qty_button');
    qtyButton.forEach(button => {
```

```
        button.addEventListener('click', function () {
            const id = this.getAttribute('data-id');
            const action = this.getAttribute('data-action');
            updateCartQty(id, action);
        })
    })
}

function updateCartQty(id, action) {
    let cart = JSON.parse(localStorage.getItem('cartItems'));
    const itemIndex = cart.findIndex(item => item.id == id);
    if (itemIndex !== -1) {
        if (action === "increase") {
            cart[itemIndex].quantity += 1;
        } else if (action === "decrease") {
            cart[itemIndex].quantity -= 1;
            if (cart[itemIndex].quantity <= 0) {
                cart.splice(itemIndex, 1); // remove item if qty is 0
            }
        }

        localStorage.setItem('cartItems', JSON.stringify(cart));
        renderCart();
    }
```

```
    }

    function updateTotal(total) {
        const totalDisplay = document.getElementById('cart_total');
        if (totalDisplay) {
            totalDisplay.textContent = `Rp.${total.toLocaleString('id-ID')}`;
        }
    }

</script>
@endpush


    @stack('scripts')

    <script>
        const userPref = localStorage.getItem('theme');
        const systemPrefersDark = window.matchMedia('(prefers-color-scheme:
dark)').matches;

        if (userPref === 'dark' || (!userPref && systemPrefersDark)) {
            document.documentElement.classList.add('dark');
        }
```

```
        if (userPref === 'light') {
            document.documentElement.classList.remove('dark');
        }

    </script>
    <script>
        function toggleTheme() {
            const html = document.documentElement;
            const isDark = html.classList.toggle('dark');
            localStorage.setItem('theme', isDark ? 'dark' : 'light');
        }

    </script>
</body>
</html>
```


B. Form Receipt Order

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Receipt Order</title>
```

```
    <!-- Tailwind CSS CDN (or use your own build process) -->
    {{-- <script src="https://cdn.tailwindcss.com"></script> --}}

    <!-- Font Awesome -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css" />

    <!-- Google Font -->
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback" />

    <!-- IonIcons -->
    <link rel="stylesheet" href="https://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css" />

    <!-- Your custom CSS (if needed) -->
    {{-- <link rel="stylesheet" href="{{asset('assets/customer/css/cekout.css')}}"> --}}
    @vite('resources/css/app.css')
</head>

<body class="bg-gray-100 font-sans">

    <div class="container mx-auto px-4">
```

```
<div class="flex justify-center mt-6">
    <img src="{{asset('assets/images/logo/logo.png')}}" alt="No Picture Added" class="rounded-full w-24 h-24 object-cover" />
</div>

<div class="flex justify-center items-center mt-4">
    <div class="w-full max-w-xl bg-white shadow-lg rounded-xl">
        <div class="p-6">
            <div class="mb-4">
                <div class="flex justify-center text-green-500 text-5xl">
                    <i class="fa-solid fa-circle-check"></i>
                </div>
                <div class="text-center mt-4">
                    <p class="text-xl font-semibold">Total pembayaran anda sebesar</p>
                    <div class="text-2xl font-bold text-green-600 mt-2">
                        @rupiah($total)
                    </div>
                </div>
                <div class="text-center mt-4">
                    <p class="text-lg">ID Pesanan = <span class="font-bold">{{$customer->order->order_code}}</span></p>
                    <div class="flex justify-center gap-4 mt-4">
                        <a href="{{route('homepage')}}" class="bg-blue-500 hover:bg-blue-600 text-white font-semibold py-2 px-4 rounded">Menu Utama</a>
```

```
                </div>
            </div>
        </div>
    </div>
    <div class="text-center pb-4">
        <p class="text-gray-600">Terima Kasih!</p>
        <small class="text-red-500">Notes: Ingat ID pesanan anda!</small>
    </div>
</div>
</div>
</div>

</body>
</html>
```

4.4 Testing

Pada tahap ini, penulis melakukan pengujian menggunakan metode *Black Box Testing* untuk mengevaluasi apakah fungsionalitas website berjalan sesuai dengan harapan dan spesifikasi yang telah ditetapkan. Dalam tugas akhir ini, pengujian dilakukan menggunakan aplikasi web browser seperti Google Chrome.

A. Form Home (Customer Memesan)

Tabel IV.14 Hasil Pengujian Black Box Testing Form Home

No.	Skenario Pengujian	Test Case	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Mengosongkan semua isian data	Cart: (kosong) Customer name: (kosong) Phone: (kosong)	Sistem akan menolak dan memberikan pesan "*The field is required.*".	Sesuai Harapan	Valid
2	Mengisi hanya salah satu data saja.	Customer name: (diisi) Phone: (kosong) Cart: (diisi)	Sistem akan menolak dan memberikan pesan "*The phone field is required*"	Sesuai Harapan	Valid

3	Mengisi kedua form namun cart dikosongkan.	Cart: (kosong)	Sistem akan menolak dan mengembalikan ke halaman home dengan memberikan pesan error, “The cart cannot be empty.”	Sesuai Harapan	Valid
4	Memesan satu menu yang melebihi stok	Cart: Item(999)	Sistem akan menolak dan mengembalikan ke halaman home dengan memberikan pesan error, “The stock isn’t enough”	Sesuai Harapan	Valid

5	Mengisi Semua form sesuai atura dan klik order now	Semua form diisi termasuk cart	Sistem akan menampilkan sebuah modal box untuk pelanggan kembali mengkonfirmasi lagi pesanannya.	Sesuai Harapan	Valid

4.5 Support

4.5.1 Publikasi Web

Setelah proses pembangunan website selesai, langkah selanjutnya agar pengunjung dapat mengaksesnya adalah dengan mempublikasikan website melalui layanan hosting. Penulis menggunakan layanan dari tencentcloud.com, dan berikut adalah domain atau alamat web yang akan diupload: dapurammih.me

4.5.2 Kebutuhan Hardware dan Software

Tabel IV.15 Kebutuhan Hardware Server

Item Server	Kebutuhan Item Server
Disk Space	5GB
Storage	SSD

RAM	1GB
Bandwith	Unlimited

Tabel IV.16 Kebutuhan Software Server

Framework	Laravel Command (PHP Artisan)
Interpreter	PHP Interpreter
Sistem Manajemen Database	MySQL
Perangkat Administrasi Database	phpMyAdmin
Bahasa Script	PHP 8

4.6 Spesifikasi Dokumen Sistem Usulan

Nama Dokumen : Daftar Menu

Fungsi : Sebagai sarana informasi bagi customer

Sumber : Kasir

Tujuan : Customer

Media : Tampilan

Frekuensi : Setiap melakukan pemesanan

Format : Lampiran B-1

Nama Dokumen : Riwayat Transaksi Penjualan

Fungsi : Sebagai bukti pendapatan

Sumber : Kasir

Tujuan : Resto

Media : Tampilan

Frekuensi	:	Setiap kasir menyelesaikan orderan
Format	:	Lampiran B-2