

Tugas-UAS-[Individu]- System Development and Implementation



Disusun Oleh:

Karisma Nabil Santosa (6026242010)

Departemen Sistem Informasi

Pengembangan dan Penerapan Sistem

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember Surabaya

2024/2025

Daftar isi

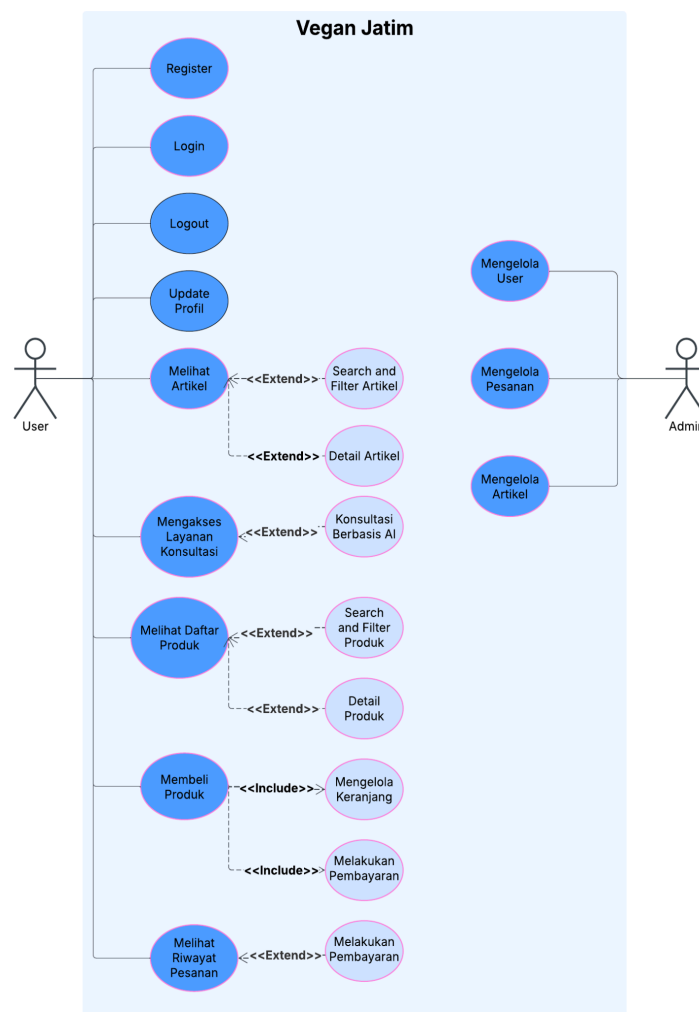
Daftar isi.....	2
1. Give an overview (aka BRIEF) regarding the project as a whole and the part of the project that is part of your responsibility (use case or use case description)...	4
a. Show the division of tasks/work of each group member. Explain the part you are working on. (Tunjukkan pembagian tugas/pekerjaan masing-masing anggota kelompok. Jelaskan bagian yang sedang Anda kerjakan.).....	4
Gambar 1.1 Use Case Diagram Vegan Jatim.....	4
Tabel 1.1 Functional Requirement.....	5
b. Explain the background of this project, the problems that need to be resolved, objectives, etc. The results of previous work or assignments can be attached at the end of this report.....	5
c. Show the mapping from requirements to use cases for which you are responsible	7
Gambar 1.2 Data Flow Diagram Vegan Jatim.....	8
Tabel 1.2 User Melakukan Registrasi.....	8
Tabel 1.3 User atau Admin Melakukan Login.....	9
Tabel 1.4 User Melihat Produk.....	10
Tabel 1.5 User Menambahkan Produk ke Cart.....	10
Tabel 1.6 User Melakukan Checkout.....	11
Tabel 1.7 User Melakukan Pembayaran.....	12
Tabel 1.8 User Update Status Selesai oleh User.....	13
Tabel 1.9 Admin Melakukan Update Pesanan.....	14
2. Based on your own use case description, develop an architecture for SOA.....	16
a. Show and explain the final architecture of the SOA you designed. What is the function of each layer? Are the services on the right layer?.....	16
Gambar 2.1 Service-oriented architecture.....	16
b. Show in tabular form the services to be created and how to access the services provided. Explain how to get the parameters sent to these services.....	17
Table 2.1 Services Table and Access Method.....	17
c. Show and explain how to implement the services related to NodeJS, database, other components, etc. Give a picture/figure to show the relationship.....	19
Gambar 2.2 Relation between front-end, back-end, dan database.....	19
Gambar 2.3 db.js.....	20
Gambar 2.4 server.js.....	21
Gambar 2.5 Database.....	21
Gambar 2.6 Stuktur folder frontend.....	22
d. How do you manage input parameters and return values? Also, explain how to communicate with the client-side if a SUCCESS/FAIL scenario occurs.....	23
Gambar 2.7 Contoh endpoint list product.....	23

Gambar 2.8 Potongan handling di sisi frontend.....	24
Gambar 2.9 Body request frontend.....	25
Gambar 2.10 respon dari sisi server jika sukses.....	25
Gambar 2.11 respon dari sisi server jika gagal.....	26
e. Show and explain the flowchart related to the orchestration with underlying services. Give highlights to the underlying services in the flowchart.....	26
Gambar 1.5 Flowchart Order.....	26
3. Explain in paragraphs and make a video tutorial (20 – 30 minutes) that explores server-side how you implement one of the services that is your responsibility using NodeJS. (Note: The face must be visible when explaining; otherwise, the score will be ZERO. :)).....	27
a. Explain one of the services that will be created (usability, flowchart, design).....	27
Gambar 3.1 Order Service.....	28
Gambar 3.2 UI dari halaman checkout Vegan Jatim.....	28
b. Explain how to implement it with NodeJS. How are input parameters obtained? How will the return value be communicated to a client-side script?.....	29
Gambar 3.3 state pada frontend checkout.....	29
Gambar 3.4 function handle checkout.....	29
c. Give an example of a SUCCESS test scenario and a FAILED scenario for the service. Explain each test scenario. Example test case:.....	31
d. Then, show the testing with Postman (or a similar application). Don't forget to include a screenshot.....	32
4. . Explain in paragraphs and make a video tutorial that discusses how you use the service on a client-side basis using a web application or a mobile phone. Choose one according to your programming skills. (Note: The face must be visible when explaining; otherwise, the score will be ZERO. :)).....	34
a. Explain the scenario that will be used for the demo. Provide at least two scenarios, for example, a SUCCESS and a FAILED scenario.....	34
b. Show and explain the application coding that will be demonstrated.....	34
c. Show that the scenario is as expected.....	34
5. Attachment.....	35

1. Give an overview (aka BRIEF) regarding the project as a whole and the part of the project that is part of your responsibility (use case or use case description).

(Berikan ikhtisar (alias RINGKASAN) mengenai proyek secara keseluruhan dan bagian proyek yang menjadi tanggung jawab Anda (use case atau use case description).

a. Show the division of tasks/work of each group member. Explain the part you are working on. (Tunjukkan pembagian tugas/pekerjaan masing-masing anggota kelompok. Jelaskan bagian yang sedang Anda kerjakan.)



Gambar 1.1 Use Case Diagram Vegan Jatim

Pada Gambar 1.1, dipaparkan bahwa terdapat dua aktor utama, yaitu user sebagai customer dan admin sebagai pengelola. Inti dari use case ini adalah:

1. Admin dapat mengelola user, mengelola pesanan, dan mengelola artikel
2. User dapat registrasi dan login, membeli produk, melihat artikel, dan mengakses layanan konsultasi.

Pada Tabel 1.1 disajikan functional requirement yang dikerjakan oleh penulis. Adapun lingkup pengerjaannya adalah pada bagian **pemesanan**.

Tabel 1.1 Functional Requirement

Use Case	Deskripsi
Melihat Daftar Produk	User dapat melihat daftar produk yang tersedia
Mengelola Keranjang	User dapat melihat, menambahkan, menghapus, produk ke keranjang belanja
Membeli Produk	User dapat membeli produk sesuai pilihan (checkout pesanan)
Melakukan Pembayaran	User dapat melakukan pembayaran dan mengirim bukti pembayaran
Melihat Riwayat Pesanan	User dapat melihat riwayat pesanan dan melacak progress pesanannya.
Mengelola Pesanan	Admin dapat melihat dan memperbarui status pesanan.

b. Explain the background of this project, the problems that need to be resolved, objectives, etc. The results of previous work or assignments can be attached at the end of this report.

(Jelaskan latar belakang proyek ini, permasalahan yang perlu diselesaikan, tujuan, dan lain-lain. Hasil pekerjaan atau tugas sebelumnya dapat dilampirkan di akhir laporan ini.)

Latar Belakang

Perkembangan gaya hidup sehat dan kesadaran terhadap dampak lingkungan telah mendorong meningkatnya minat masyarakat terhadap pola makan berbasis nabati (vegan). Di tengah tren tersebut, kebutuhan akan pilihan kuliner vegan yang tidak hanya sehat tetapi juga tetap mempertahankan cita rasa tradisional lokal menjadi semakin penting. Jawa Timur, sebagai salah satu daerah dengan kekayaan kuliner yang khas dan beragam, memiliki potensi besar untuk menghadirkan makanan vegan yang autentik tanpa harus mengorbankan nilai budaya rasa.

Namun, kenyataannya pilihan makanan vegan khas Jawa Timur masih sangat terbatas. Banyak masyarakat yang tertarik mencoba gaya hidup vegan, tetapi kesulitan menemukan hidangan yang sesuai dengan lidah lokal dan mudah diakses. Di sisi lain, makanan vegan yang beredar di pasaran sering kali tidak menawarkan cita rasa lokal yang kuat atau menggunakan bahan yang bercampur dengan bahan non vegan, sehingga ini menyulitkan bagi para vegan untuk mencari produk vegan yang tepat..

Menjawab kebutuhan tersebut, Vegan Jatim hadir sebagai layanan kuliner yang menyajikan makanan dan minuman khas Jawa Timur dalam versi vegan, tanpa bahan hewani namun tetap mempertahankan keautentikan rasa dan kekayaan rempah. Dengan menggunakan bahan alami berkualitas dan teknik memasak tradisional, Vegan Jatim menawarkan solusi kuliner yang sehat, lezat, dan ramah lingkungan. Tidak hanya itu, layanan ini juga didukung oleh sistem pemesanan yang mudah dan praktis, sehingga konsumen dapat menikmati sajian vegan kapan saja dan di mana saja.

Melalui inisiatif ini, Vegan Jatim tidak hanya ingin menjadi pilihan makanan sehat, tetapi juga menjadi bagian dari gerakan pelestarian budaya kuliner lokal dan gaya hidup berkelanjutan.

Permasalahan

1. Masih terbatasnya ketersediaan makanan vegan yang tetap mempertahankan cita rasa autentik khas Jawa Timur.
2. Kurangnya pilihan kuliner sehat dan ramah lingkungan yang berbasis nabati bagi masyarakat Jawa Timur dan sekitarnya.
3. Minimnya layanan pemesanan makanan vegan yang mudah dan praktis dengan kualitas rasa dan gizi yang terjamin.

Tujuan

1. Menghadirkan layanan kuliner vegan khas Jawa Timur tanpa bahan hewani namun tetap mempertahankan cita rasa tradisional.
2. Menyediakan makanan dan minuman sehat, bergizi, dan ramah lingkungan melalui penggunaan bahan nabati berkualitas.
3. Menyediakan sistem pemesanan yang mudah diakses dan praktis bagi konsumen yang ingin menikmati kuliner vegan.

Manfaat

1. Memberikan alternatif makanan sehat bagi masyarakat yang ingin beralih ke pola makan berbasis nabati.
2. Menjaga dan melestarikan cita rasa khas Jawa Timur dalam versi vegan yang inovatif.

3. Mendukung gaya hidup sehat dan berkelanjutan melalui kuliner yang ramah lingkungan.
4. Meningkatkan kesadaran masyarakat terhadap pentingnya konsumsi makanan tanpa bahan hewani.

Batasan Pengerjaan

1. Produk makanan dan minuman yang ditawarkan hanya berbasis nabati dan tidak mengandung unsur hewani sama sekali.
2. Menu yang disajikan berfokus pada kuliner khas Jawa Timur, bukan dari daerah lain.
3. Layanan berbasis website yang dibangun dengan React JS dan Node JS
4. Pada tugas ini, penulis hanya berfokus ke fitur pemesanan saja.

c. Show the mapping from requirements to use cases for which you are responsible

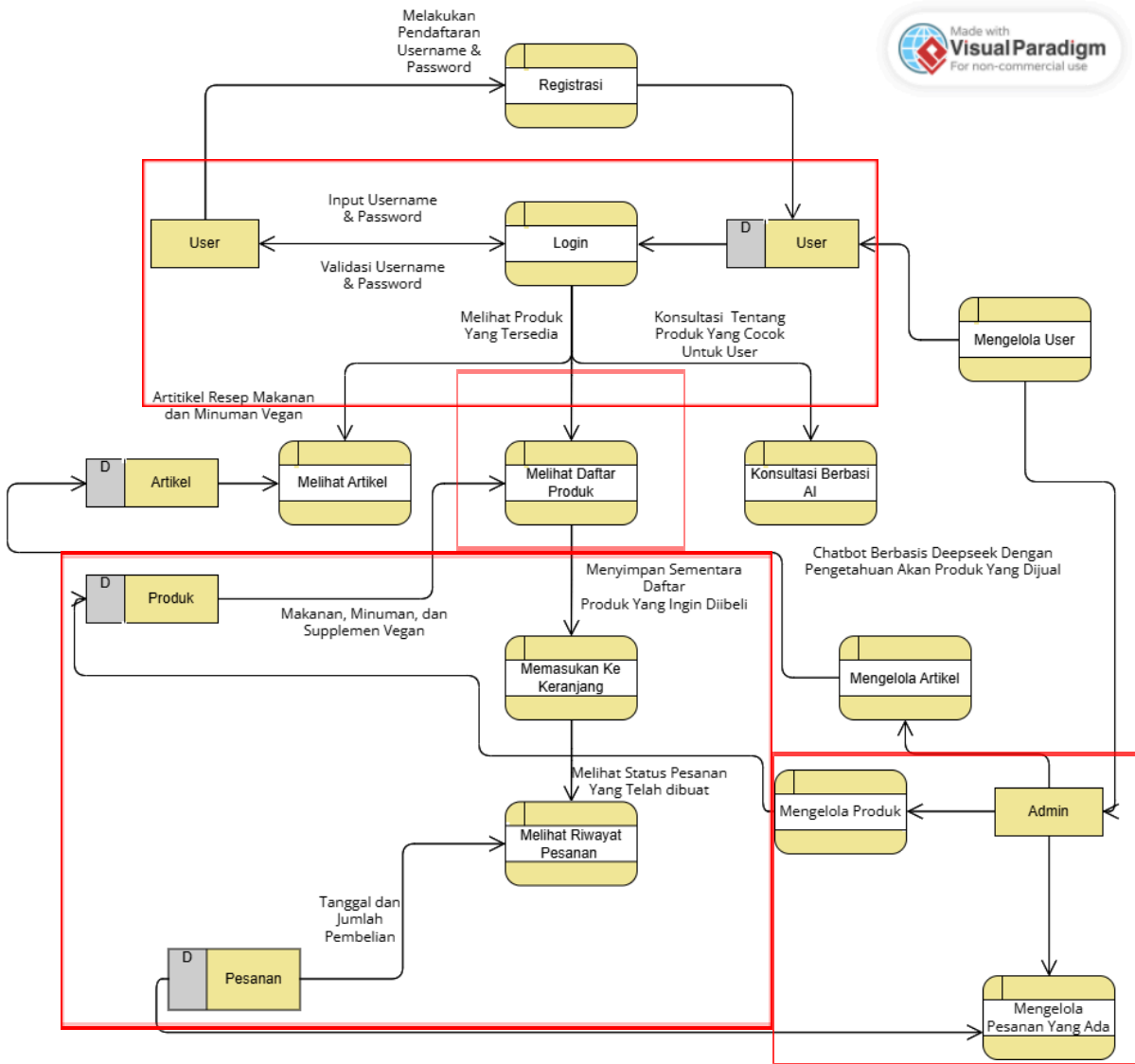
(Tunjukkan pemetaan dari persyaratan ke use cases yang menjadi tanggung jawab Anda)

Adapun requirement yang dikerjakan oleh penulis adalah sebagai berikut::

1. **User Melihat Daftar Produk**
2. **User Mengelola Keranjang**
3. **User Membeli Produk**
4. **User Melakukan Pembayaran**
5. **User Melihat Riwayat Pesanan**
6. **Admin Mengelola Pesanan**

Diperlukan juga sistem login dan registrasi agar dapat menyimpan data pesanan masing-masing user, dan untuk akses ke dashboard admin.

Dalam memetakan, perlu dibuat terlebih dahulu data flow diagram. Pada Gambar 1.2, dipaparkan data flow diagram yang menggambarkan pergerakan informasi atau data dari sistem Vegan Jatim. Kotak berwarna merah merupakan lingkup yang dikerjakan oleh penulis.



Gambar 1.2 Data Flow Diagram Vegan Jatim

Dibawah ini adalah Tabel 1.2 hingga Tabel 1.9 yang berisi use case description.

Tabel 1.2 User Melakukan Registrasi

Use case name	Melakukan Register
Actors	User
Pre-condition	User mengakses ke website.
Post-condition	User masuk ke halaman login
Business Rules	-

Primary flows		User melakukan registrasi untuk dapat memiliki akun sehingga dapat memesan produk
Actor Actions		
1. User memasukkan nama, email, dan password, lalu simpan		
		2. Website menyimpan data dan mengarahkan ke halaman login
Alternate path	A1	Email sudah pernah ada
Actor Actions		
		2a1. Website menampilkan alert gagal registrasi karena email sudah terdaftar
2a2. Pelanggan menggunakan alamat email lain		

Tabel 1.3 User atau Admin Melakukan Login

Use case name		Melakukan Login
Actors		User, Admin
Pre-condition		User/Admin mengakses ke website.
Post-condition		User/Admin masuk ke halaman home
Business Rules		-
Primary flows		User melakukan login untuk dapat mengakses perubahan pada data
Actor Actions		
1. User memasukkan email, dan password, lalu login		
		2. Website verifikasi data dan mengarahkan ke halaman home
Alternate path	A1	Email atau password salah

Actor Actions	
	2a1. Website menampilkan alert gagal registrasi karena email atau password salah
2a2. Pelanggan mengulangi mengisi email dan password yang benar	

Tabel 1.4 User Melihat Produk

Use case name	Melihat Daftar Produk
Actors	User
Pre-condition	User mengakses website
Post-condition	User melihat-lihat produk
Business Rules	-
Primary flows	User mengakses halaman home untuk melihat daftar produk
Actor Actions	
1. User masuk ke halaman home	
	2. Website menampilkan daftar produk di halaman home

Tabel 1.5 User Menambahkan Produk ke Cart

Use case name	Menambahkan Produk ke Cart
Actors	User
Pre-condition	User sudah login
Post-condition	User melakukan checkout pesanan
Business Rules	Produk yang ditambahkan ke keranjang tidak akan dipesan sampai pelanggan menyelesaikan checkout.

Primary flows		User menambahkan produk ke cart dan meneruskan ke checkout
Actor Actions		
1. User menambahkan produk ke cart dari halaman home		
		2. Website menampilkan alert produk telah ditambahkan
3. User Melakukan chekout		
		4. Website akan meneruskan data cart ke halaman checkout dan redirect ke checkout
Alternate path	A1	Menambahkan atau mengurangi quantity
Actor Actions		
3a1. User ingin menambahkan atau mengurangi jumlah quantity		
		3a1 Website akan menampilkan jumlah quantity dan subtotal harganya dari setiap produk

Tabel 1.6 User Melakukan Checkout

Use case name	Melakukan Checkout
Actors	User
Pre-condition	User berada di halaman checkout
Post-condition	User menyimpan pesanan dan pergi ke halaman pembayaran
Business Rules	-
Primary flows	User melakukan checkout pesanan untuk menyimpan data pesanan
Actor Actions	

1. User memasukkan data alamat, pengiriman daerah, dan metode pembayaran	
	2. Website menampilkan summary subtotal dan total harga
3. User melakukan konfirmasi dan bayar	
	4. Website menyimpan data pesanan dan redirect ke halaman pembayaran

Tabel.1.7 User Melakukan Pembayaran

Use case name	Melakukan Pembayaran
Actors	User
Pre-condition	User sudah masuk ke halaman pembayaran
Post-condition	Status pesanan “waiting for confirmation”
Business Rules	Jika pelanggan tidak menyelesaikan pembayaran dalam waktu tertentu Pesanan akan otomatis dibatalkan. Bukti pembayaran wajib diunggah dengan format jpg, png, jpeg.
Primary flows	User melakukan pembayaran dan mengunggah bukti pembayaran
Actor Actions	
1. User Melakukan pembayaran	
2. User mengunggah bukti pembayaran	
	3. Website akan mengupdate data pesanan dan mengubah status menjadi “waiting for confirmation”
	4. Website akan redirect ke halaman riwayat pesanan

Alternate path	A1	Pelanggan Tidak Mengunggah Bukti Pembayaran dalam Waktu yang Ditentukan
Actor Actions		
		2a1. Website menampilkan peringatan bahwa pelanggan harus mengunggah bukti pembayaran dalam batas waktu tertentu.
		2a2. Jika batas waktu terlewati, website otomatis membatalkan pesanan.
		2a3. Pelanggan dapat memesan ulang memesan produk dari awal.
Alternate path	A2	User Tidak jadi memesan
Actor Actions		
		1a1. User dapat pergi ke riwayat pesanan
		1a2. User mengakses tab status waiting for payment dan membatalkan pesanan
		1a3. Website akan mengubah status menjadi cancelled.
Exception path	E1	Bukti Pembayaran palsu
Actor Actions		
		3a1 Website akan mengembalikan status menjadi payment failed
		3a2. User dapat melakukan order ulang

Tabel 1.8 User Update Status Selesai oleh User

Use case name	Melakukan Update Pesanan
Actors	User

Pre-condition	User menunggu proses barang di halaman riwayat pesanan
Post-condition	User menyelesaikan pesanan
Business Rules	-
Primary flows	User mengubah status menjadi complete
Actor Actions	
1. User mengecek pesanan di status "shipped" karena barang sedang proses krim	
2. User melakukan selesaikan pesanan	
	3. Website menyimpan data status complete
Exception path	E1
Actor Actions	
	1a1. Website menampilkan status pesanan yang tidak berubah
1a2. User dapat komplain ke admin atau tracking no resi di luar sistem	

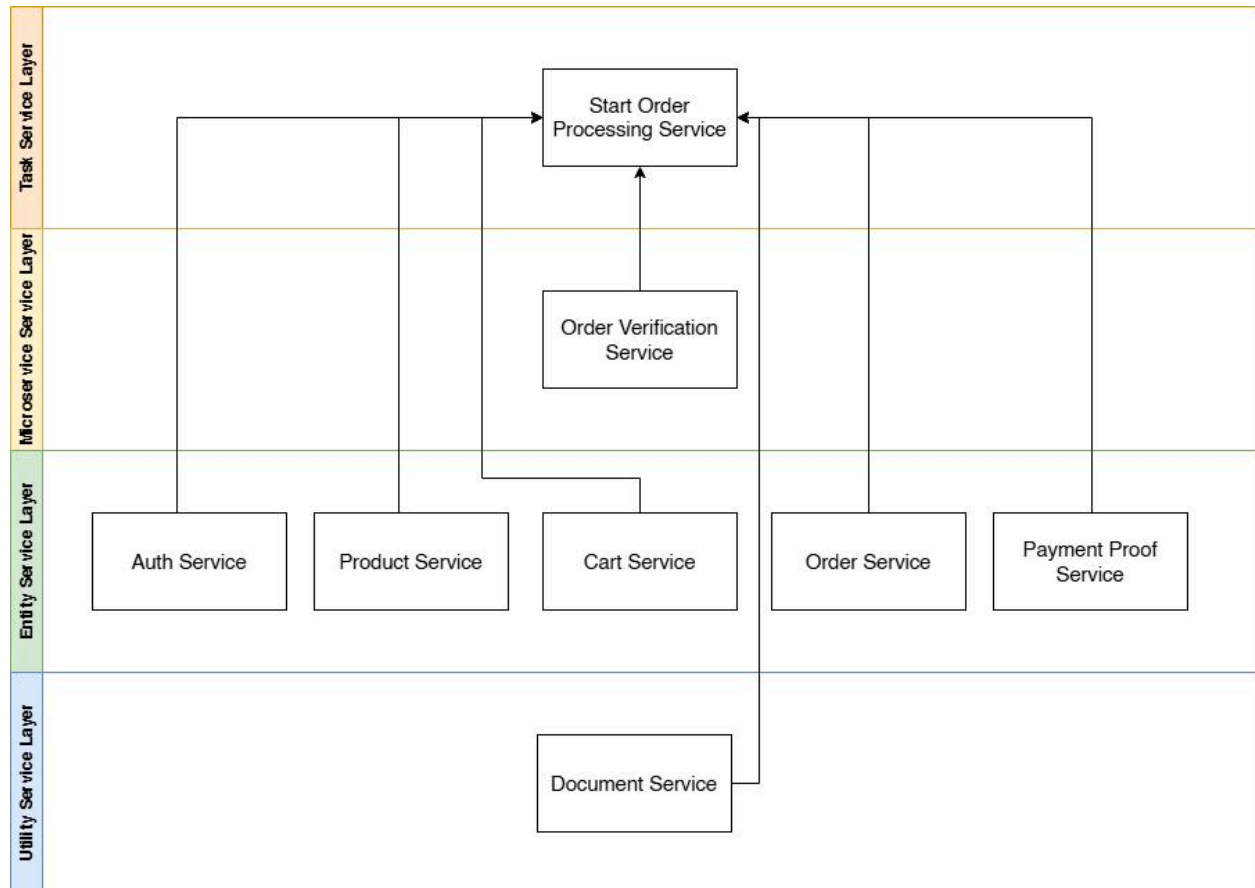
Tabel 1.9 Admin Melakukan Update Pesanan

Use case name	Melakukan Update Pesanan
Actors	Admin
Pre-condition	Admin berhasil login as admin
Post-condition	Admin mengubah status hingga shipped
Business Rules	Admin tidak boleh mengubah status menjadi waiting for payment, cancelled, dan selesaikan pesanan

Primary flows		Admin mengubah status hingga shipped
Actor Actions		
1. Admin mengecek tabel pesanan di status "waiting confirmation"		
		2. Website menampilkan filter status "waiting confirmation"
3. Admin mengubah tabel pesanan di status "waiting confirmation" menjadi "processing"		
		4. Website menyimpan perubahan status.
5. Admin mengubah tabel pesanan di status "processing" menjadi "shipped"		
6. Admin mengisi nomor resi dan menyimpan perubahan		
		7. Website menyimpan perubahan dan menampilkan status shipped dan nomor resinya.
Alternation path	E1	Bukti tidak valid
Actor Actions		
1a1. Admin mengubah status menjadi "payment failed"		
		1a2. Website akan menyimpan status menjadi payment failed

2. Based on your own use case description, develop an architecture for SOA

a. Show and explain the final architecture of the SOA you designed. What is the function of each layer? Are the services on the right layer?



Gambar 2.1 Service-oriented architecture

Pada Gambar 2.1, dipaparkan Service-oriented architecture dari sistem Vegan Jatim meliputi:

1. **Task Service Layer** berisi keseluruhan layanan yang tujuannya untuk memproses layanan,
2. **Microservice Layer** berisi order verification, untuk otomatisasi verifikasi order, yang mana kedepannya dapat ditambahkan ke dalam proses karena sistem saat ini masih semi otomasi dengan validasi oleh admin.
3. **Entity Service Layer**, berisi 5 layanan utama, antara lain; Auth service, Product Service, Cart Service, Order Service, Payment Proof Service. Kelima layanan ini yang menjadi proses terhadap jalannya sistem Vegan Jatim
4. **Utility Service**, berisi Document Service yang berfungsi untuk upload file berupa dokumen gambar bukti pembayaran.

b. Show in tabular form the services to be created and how to access the services provided. Explain how to get the parameters sent to these services

Terdapat 5 aktivitas utama pada Tabel 2.1 yang diambil dari entity service layer, meliputi:

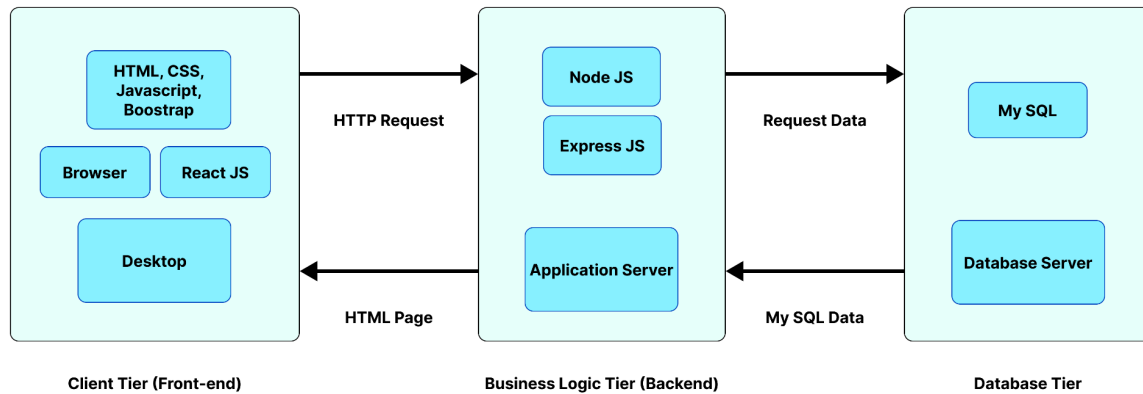
1. Terdapat **Auth service** yang berisi path /login dan /register untuk urusan autentikasi, yang mana membutuhkan email dan password untuk dapat melakukan pesanan. Kemudian Product Service
2. **Product Service**, hanya menampilkan produknya saja, yang mana user baik sudah login maupun belum, dapat mengaksesnya.
3. **Cart Service** menunjukkan user dapat menambahkan produk ke keranjang yang membutuhkan user id, dan quantity agar setiap user dapat memiliki cartnya masing-masing.
4. **Order Service**, adalah layanan yang memungkinkan user bisa memesan produk yang sudah ditambah dari Cart, dengan parameter user_id, shipping_address, dan payment_method. Selain itu dari sisi admin, akan mampu untuk mengubah status pesanan untuk mengontrol jalannya proses pesanan.
5. **Payment Proof Service**, berisi instruksi pembayaran, dan fitur upload file bukti pembayaran, sebagai validasi ke admin.

Table 2.1 Services Table and Access Method

Method	Path	Description	Input Parameters	Return Values
POST	/login	Mengirim kredensial untuk login	email, password	Success: Login successful Failed: Invalid credentials
POST	/register	Mengirim data user baru untuk registrasi	email, password	Success: User registered successfully Failed: Error registering user
GET	/	Menampilkan semua produk		Success: List of products Failed: Data empty
GET	/cart/:userId	Melihat isi keranjang	userId (path param)	Success: List of cart items Failed: Data Empty
POST	/cart	Menambahkan produk ke keranjang	user_id, product_id, quantity	Success: Product Add to cart Failed:

POST	/checkout	Checkout pesanan	user_id, shipping_address, payment_method	Success: Go to Payment page (Order created successfully) Failed:
GET	/payment/:orderId/	Cek pembayaran	user_id, orderId	Success: Order Summary, status, instruction payment Failed:
POST	/orders/:orderId/upload	Unggah bukti pembayaran	order_id, payment_proof (file)	Success: Bukti pembayaran berhasil diunggah! Failed: Gagal menyimpan bukti pembayaran
GET	/orders/:userId	Lihat riwayat pesanan	userId	Success: List of orders and statuses Failed: No orders found for this user.
PUT	/orders/:orderId	Admin ubah status pesanan	order_id, status	Success: Order status updated successfully Failed:

c. Show and explain how to implement the services related to NodeJS, database, other components, etc. Give a picture/figure to show the relationship



Gambar 2.2 Relation between front-end, back-end, dan database

Dalam membangun Vegan Jatim berbasis website, penulis menggunakan React JS, Node JS, Express JS, dan MySQL. Pada Gambar 2.2, dipaparkan relasi antara masing masing tier.

Client tier (Front-end): Merupakan antarmuka yang berinteraksi langsung dengan pengguna (user). Mengirim permintaan (HTTP Request) ke server dan menerima respons (HTML Page/data).

- **HTML/CSS3, JavaScript:** Teknologi dasar untuk struktur, tampilan, dan interaktivitas halaman web.
- **Bootstrap:** digunakan untuk framework styling agar lebih mudah
- **ReactJS:** Library JavaScript modern untuk membangun antarmuka pengguna yang dinamis.
- **Chrome Browser:** Tempat menjalankan aplikasi oleh user.
- **Desktop:** Perangkat user tempat aplikasi diakses.

Business Logic Tier (Back-end): Memproses logika bisnis aplikasi. Menerima permintaan dari client, memprosesnya, berkomunikasi dengan database, lalu mengembalikan hasil ke client.

- **Node.js:** Environment JavaScript untuk menjalankan kode di sisi server.
- **Express.js:** Framework berbasis Node.js yang menyederhanakan pembuatan API dan routing.
- **Application Server:** Server yang menjalankan kode backend dan mengelola komunikasi antara client dan database.

Database Tier: Menyimpan dan mengelola data secara permanen. Menyediakan data saat diminta oleh backend.

- **MySQL:** Sistem manajemen basis data relasional (RDBMS) yang menyimpan data aplikasi.
- **Database Server:** Mesin/server yang menjalankan MySQL dan menangani permintaan data.

```
1 import mysql from "mysql2";
2 import dotenv from "dotenv";
3
4 dotenv.config();
5
6 const db = mysql.createConnection({
7   host: process.env.DB_HOST,
8   user: process.env.DB_USER,
9   password: process.env.DB_PASSWORD,
10  database: process.env.DB_NAME,
11 });
12
13 db.connect((err) => {
14   if (err) {
15     console.error("Database connection failed: " + err.stack);
16     return;
17   }
18   console.log("Connected to database");
19 });
20
21 export default db;
22
```

Gambar 2.3 db.js

Gambar 2.3 db.js menunjukkan koding yang menghubungkan ke database mysql. Lalu pada Gambar 2.4 menunjukkan koding yang berisi controller terhadap layanan, dan menghubungkan front-end ke database dengan api dan query tabel pada database. Adapun databasenya dapat dilihat pada Gambar 2.5. Terdapat 4 tabel, yaitu user, orders, order_items, dan products.

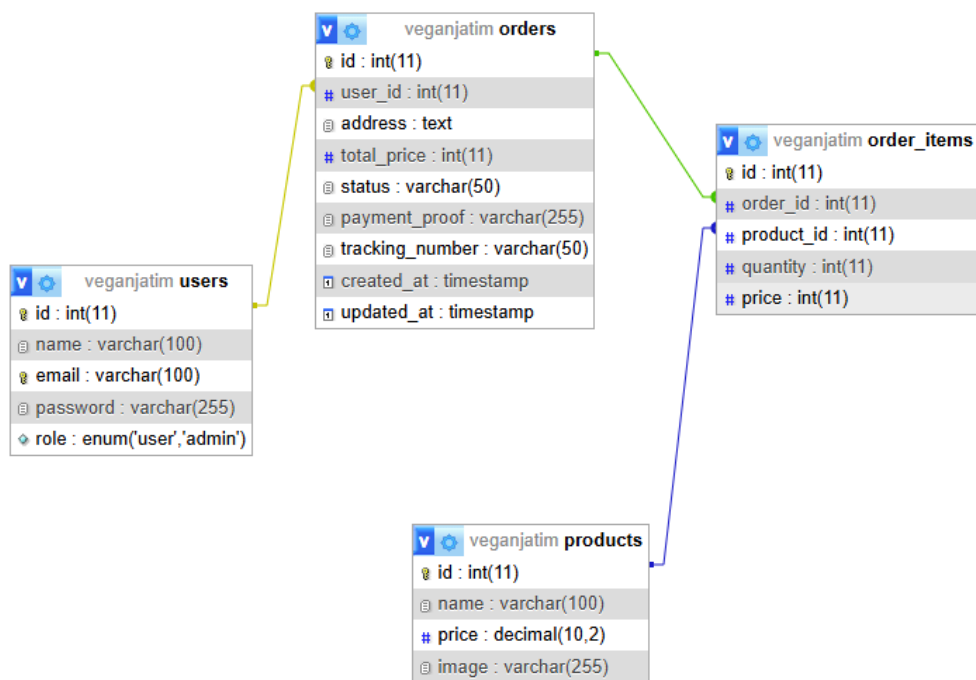
- **users** adalah data pengguna baik itu admin dan user biasa (customer)
- **orders** adalah data yang menampung pesanan dari customer, orders berelasi dengan user_id
- **order-items** adalah data batch atau kumpulan produk yang dipesan kedalam orders, yang mana ini berelasi ke order_id dan product_id
- **products** adalah data makanan dan minuman vegan jatim

```

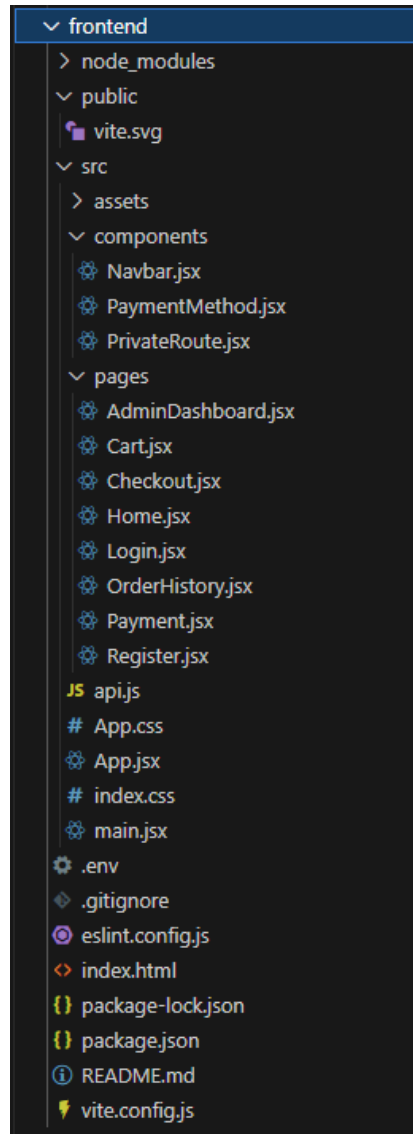
1 import express from "express";
2 import cors from "cors";
3 import bodyParser from "body-parser";
4 import dotenv from "dotenv";
5 import db from "../db.js";
6 import bcrypt from "bcryptjs";
7 import jwt from "jsonwebtoken";
8 import multer from "multer";
9 import path from "path";
10 import fs from "fs";
11 import { fileURLToPath } from "url";
12
13 const __filename = fileURLToPath(import.meta.url);
14 const __dirname = path.dirname(__filename);
15 dotenv.config();
16
17 const app = express();
18 app.use(cors({
19   origin: "http://localhost:5173", // URL frontend
20   credentials: true, // Izinkan cookies
21   methods: ['GET', 'POST', 'PUT', 'DELETE'],
22   allowedHeaders: ['Content-Type', 'Authorization'],
23 }));
24 app.use(bodyParser.json());
25 app.use(express.json());
26 app.use(express.urlencoded({ extended: true }));
27 app.use("/uploads", express.static(path.join(__dirname, "uploads")));
28
29 const PORT = process.env.PORT || 5000;

```

Gambar 2.4 server.js



Gambar.2.5 Database



Gambar 2.6 Struktur folder frontend

Pada struktur folder frontend yang bisa dilihat pada Gambar 2.6, disinilah tempat membuat file react, html, css, dan menyimpan assets. Pages adalah folder yang mengandung halaman-halaman yang menampilkan data ke client, dan komponen adalah file yang akan digunakan di dalam pages. Semua terhubung ke index.html sebagai main file.

d. How do you manage input parameters and return values? Also, explain how to communicate with the client-side if a SUCCESS/FAIL scenario occurs.

Berikut adalah contoh yang penulis ambil dari fitur **checkout** yang dapat dilihat pada Gambar 2.7

```
1 app.post("/checkout", async (req, res) => {
2   const { user_id, address, total_price, items } = req.body;
3
4   if (!user_id || !address || !total_price || !Array.isArray(items) || items.length === 0) {
5     return res.status(400).json({status:"error", message: "Invalid request data" });
6   }
7
8   try {
9     // Insert order
10    const orderResult = await new Promise((resolve, reject) => {
11      const sqlOrder = `
12      INSERT INTO orders (user_id, address, total_price, status, created_at, updated_at)
13      VALUES (?, ?, ?, 'waiting for payment', NOW(), NOW())
14      `;
15      db.query(sqlOrder, [user_id, address, total_price], (err, result) => {
16        if (err) return reject(err);
17        resolve(result);
18      });
19    });
20
21    const orderId = orderResult.insertId;
22
23    // Insert order items
24    const sqlItems = `INSERT INTO order_items (order_id, product_id, quantity, price) VALUES ?`;
25    const itemValues = items.map(item => [orderId, item.product_id, item.quantity, item.product_price * item.quantity]);
26
27    await new Promise((resolve, reject) => {
28      db.query(sqlItems, [itemValues], (err) => {
29        if (err) return reject(err);
30        resolve();
31      });
32    });
33
34    res.status(200).json({status:"success", message:"Order created successfully", data: orderId });
35  } catch (err) {
36    res.status(500).json({ status:"error", message: "Failed to create order", data: err });
37  }
38 });
```

Gambar 2.7 Contoh endpoint list product



```

1  try { // Mengirim data checkout ke server
2      const response = await fetch("http://localhost:5000/checkout", {
3          method: "POST",
4          headers: {
5              "Content-Type": "application/json",
6          },
7          body: JSON.stringify({
8              user_id: userId,
9              address,
10             total_price: TOTAL,
11             items: cart,
12         }),
13     });
14
15     setLoading(false); // Set loading state to false after response
16
17     if (response.ok) {
18         const data = await response.json();
19         localStorage.removeItem("cart");
20         if (data.data) { // Cek apakah data.data ada
21             navigate(`/payment/${data.data}`); // Navigasi ke halaman payment dengan order ID
22         } else {
23             alert("Order ID not found in response. Please contact support.");
24         }
25     } else {
26         const errorText = await response.text();
27         console.error("Checkout failed:", errorText);
28         alert("There was a problem processing your order. Please try again.");
29     }
30 } catch (error) {
31     setLoading(false);
32     console.error("Checkout error:", error);
33     alert("There was a problem processing your order. Please try again.");
34 }

```

Gambar 2.8 Potongan handling di sisi frontend

Client-side (React)

Saat user klik “Confirm and Pay” → Data **cart** + **alamat** + **userid** dikirim lewat **fetch()** POST ke **/checkout**.

Response dari server diproses:

- Jika sukses: ambil data.data (orderId), dan redirect ke **/payment/{orderId}**.
- Jika gagal: tampilkan alert error.

Lihat lebih detail pada Gambar 2.8.


```

1  {
2    "user_id": 1,
3    "address": "Jl. Mawar No. 123, Jember",
4    "total_price": 20000,
5    "items": [
6      {
7        "product_id": 1,
8        "quantity": 1,
9        "product_pric": 10000
10     },
11     {
12       "product_id": 2,
13       "quantity": 1,
14       "product_price": 10000
15     }
16   ]
17 }

```

Gambar 2.9 Body request frontend

Frontend kirim (POST ke **/checkout**) dengan body request yang dapat dilihat pada Gambar 2.9.

Server-side (Node.js/Express)

Endpoint **/checkout** menerima data dari frontend: **user_id**, **address**, **total_price**, dan **items**.

Jika data valid, maka:

1. Menyimpan order ke tabel **orders**
2. Menyimpan detail barang ke tabel **order_items**
3. Mengembalikan response sukses dengan orderId yang bisa dilihat pada Gambar 2.10:

```

1  {
2    "status": "success",
3    "message": "Order created successfully",
4    "data": 56
5  }

```

Gambar 2.10 respon dari sisi server jika sukses

Jika gagal, maka response error dapat dilihat pada Gambar 2.11:

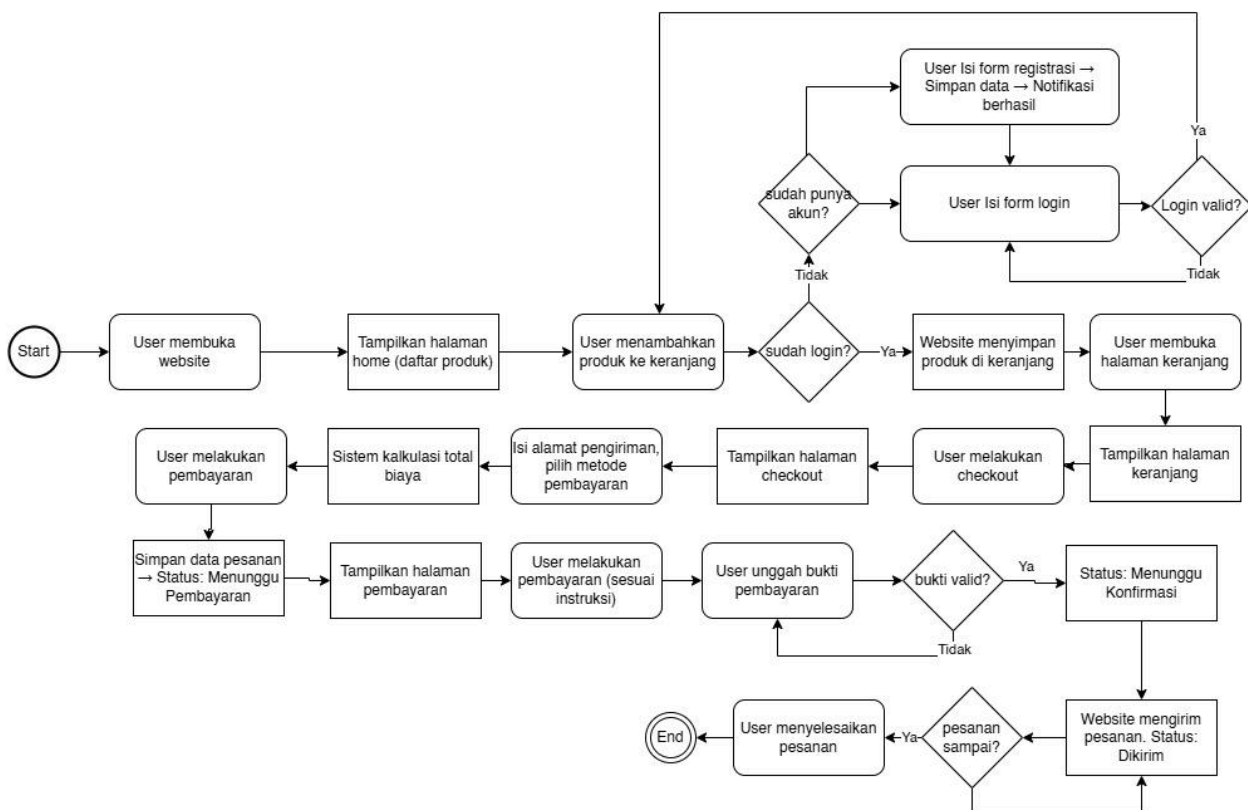
```

1 {
2   "status": "error",
3   "message": "Failed to create order",
4   "data": {
5     "code": "ER_BAD_FIELD_ERROR",
6     "errno": 1054,
7     "sqlState": "42S22",
8     "sqlMessage": "Unknown column 'NaN' in 'field list'",
9     "sql": "INSERT INTO order_items (order_id, product_id, quantity, price) VALUES (57, 1, 1, NaN), (57, 2, 1, 10000)"
10  }
11 }

```

Gambar 2.11 respon dari sisi server jika gagal

e. Show and explain the flowchart related to the orchestration with underlying services. Give highlights to the underlying services in the flowchart



Gambar 1.5 Flowchart Order

Pada Gambar 1.5, ditampilkan flow sistem vegan jatim dari awal hingga akhir.

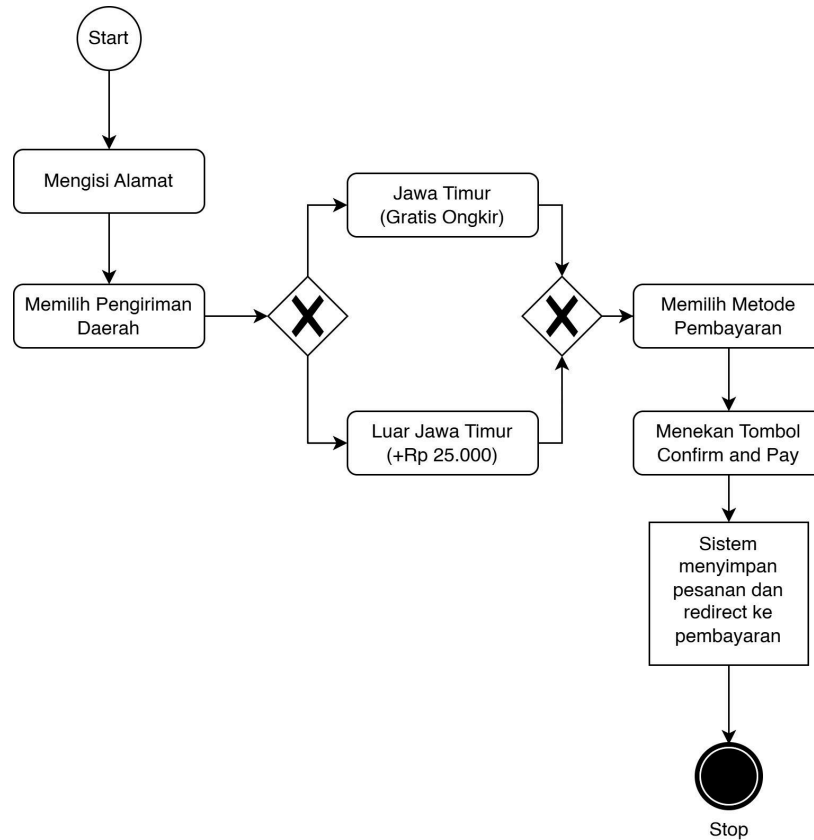
1. **Auth Service** : User dapat melakukan login dan registrasi, bagi user yang belum login dapat melihat halaman home yang berisi daftar produk.
2. **Product Service**: User bisa melihat daftar produk di home.
3. **Cart Service** : User yang sudah login dapat menambahkan produk ke cart, tetapi masih bisa melihat halaman home (produk). Jika sudah login maka bisa menambahkan produk ke cart.

4. **Order (Checkout) Service:** User bisa membuat pesanan dan mengisi data yang diperlukan seperti alamat dan metode pembayaran.
5. **Payment Proof Service:** User dapat melakukan pembayaran dan mengupload bukti pembayaran.
6. Pada akhirnya user dapat menyelesaikan pesanan dengan update status pesanannya jika barang sudah ditangan pembeli.

3. Explain in paragraphs and make a video tutorial (20 – 30 minutes) that explores server-side how you implement one of the services that is your responsibility using NodeJS. (Note: The face must be visible when explaining; otherwise, the score will be ZERO. :))

a. Explain one of the services that will be created (usability, flowchart, design).

Layanan "Order Service" digunakan ketika user checkout dari keranjang. User telah melakukan proses checkout dari halaman keranjang, mengisi Alamat, Pengiriman Daerah, dan Metode Pembayaran. Sistem yang meneruskan item dari cart akan menjumlahkan nilai total harga pesanan ditambah dengan ongkos kirim. Lalu saat user klik Confirm and Pay maka user akan menyimpan pesanan dan meneruskan ke halaman pembayaran. Pada Gambar 3.1 ditunjukkan flowchart pada sistem order / checkout dan Gambar 3.2 menampilkan User Interface dari halaman checkout



Gambar 3.1 Order Service

[Home](#)
[Riwayat Pesanan](#)
[Cart 2](#)
[Logout](#)

Checkout

Alamat

Pengiriman Daerah

Jawa Timur (Gratis Ongkir)

Metode Pembayaran

BCA

Subtotal **Rp15000**

Ongkos Kirim **Rp0**

Total Rp15,000

Confirm and Pay

Gambar 3.2 UI dari halaman checkout Vegan Jatim

b. Explain how to implement it with NodeJS. How are input parameters obtained? How will the return value be communicated to a client-side script?

```
const Checkout = () => {
  const [address, setAddress] = useState(""); // State untuk menyimpan alamat
  const [region, setRegion] = useState("jatim"); // Default: Jawa Timur
  const [paymentMethod, setPaymentMethod] = useState(""); // Menyimpan metode pembayaran
  const shippingFee = region === "jatim" ? 0 : 25000; // Gratis ongkir di Jatim, Rp 25.000 luar Jatim
  const [loading, setLoading] = useState(false); // State untuk menampilkan loading saat checkout
  const userId = localStorage.getItem("userId"); // Mengambil userId dari localStorage
  const navigate = useNavigate(); // Menggunakan useNavigate untuk navigasi
  const location = useLocation(); // Menggunakan useLocation untuk mendapatkan state dari navigasi sebelumnya
  const totalPrice = location.state?.totalPrice || 0; // Mengambil total harga dari state
  const TOTAL = totalPrice + shippingFee; // Mengambil total harga dari state
}
```

Gambar 3.3 state pada frontend checkout

```
const handleCheckout = async () => {
  // Mengambil cart dari localStorage, jika tidak ada, set ke array kosong
  const cart = JSON.parse(localStorage.getItem("cart")) || [];
  if (cart.length === 0) {
    alert("Your cart is empty. Please add items to cart before checking out.");
    return;
  }

  if (!address) { // Cek apakah alamat sudah diisi
    alert("Please enter your address before proceeding.");
    return;
  }

  if (!paymentMethod) { // Cek apakah metode pembayaran sudah dipilih
    alert("Please select a payment method.");
    return;
  }

  localStorage.setItem("paymentMethod", paymentMethod); // Simpan metode pembayaran ke localStorage
  setLoading(true); // Set loading state to true
}
```

Gambar 3.4 function handle checkout

Selanjutnya dapat dilihat kembali pada Gambar 2.8 Potongan handling di sisi frontend yang melakukan proses pengiriman ke <http://localhost:5000/checkout> yang mana akan mengarah ke endpoint checkout di sisi server.

```

1  app.post("/checkout", async (req, res) => {
2    const { user_id, address, total_price, items } = req.body;
3
4    if (!user_id || !address || !total_price || !Array.isArray(items) || items.length === 0) {
5      return res.status(400).json({status:"error", message: "Invalid request data" });
6    }
7
8    try {
9      // Insert order
10     const orderResult = await new Promise((resolve, reject) => {
11       const sqlOrder = `
12         INSERT INTO orders (user_id, address, total_price, status, created_at, updated_at)
13         VALUES (?, ?, ?, 'waiting for payment', NOW(), NOW())
14       `;
15       db.query(sqlOrder, [user_id, address, total_price], (err, result) => {
16         if (err) return reject(err);
17         resolve(result);
18       });
19     });
20
21     const orderId = orderResult.insertId;
22
23     // Insert order items
24     const sqlItems = `INSERT INTO order_items (order_id, product_id, quantity, price) VALUES ?`;
25     const itemValues = items.map(item => [orderId, item.product_id, item.quantity, item.product_price * item.quantity]);
26
27     await new Promise((resolve, reject) => {
28       db.query(sqlItems, [itemValues], (err) => {
29         if (err) return reject(err);
30         resolve();
31       });
32     });
33
34     res.status(200).json({status:"success", message:"Order created successfully", data: orderId });
35   } catch (err) {
36     res.status(500).json({ status:"error", message: "Failed to create order", data: err });
37   }
38 });

```

Pada sisi server, endpoint akan menerima request dari frontend, antara lain:

- user_id,
- address,
- total_price,
- items

(Penulis **tidak membuat metode pembayaran tersimpan di database**, melainkan di local storage yang mana nantinya akan diteruskan ke halaman pembayaran).

Jika tidak sesuai, maka akan mengirim response 400, invalid request data.

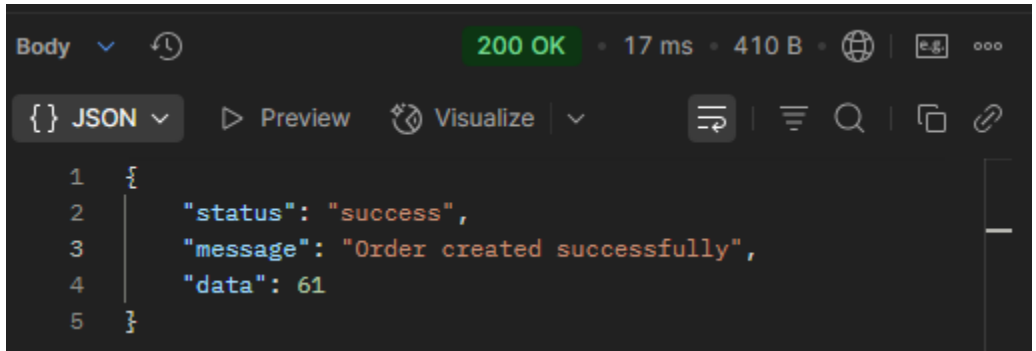
Kemudian, pada blok try, dilakukan proses menambahkan data ke sql. Terdapat dua tabel yaitu orders : pesanan keseluruhan, dan order_items : daftar list produk. Lalu jika berhasil maka akan mengirim response 200, Order created successfully. Namun jika gagal, maka akan mengirim status 500, Failed to create order.

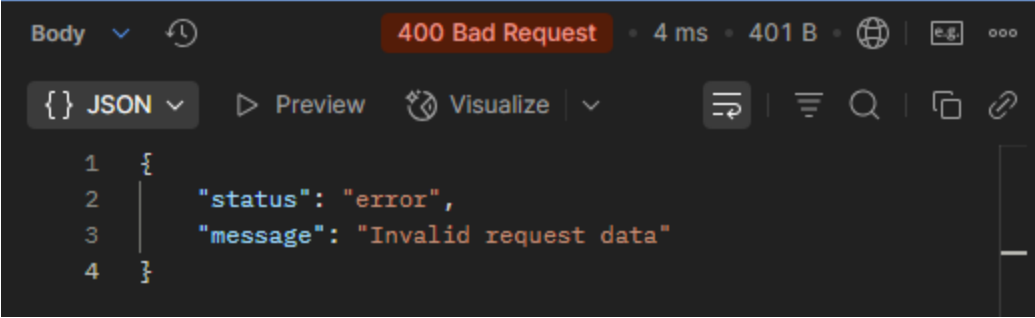
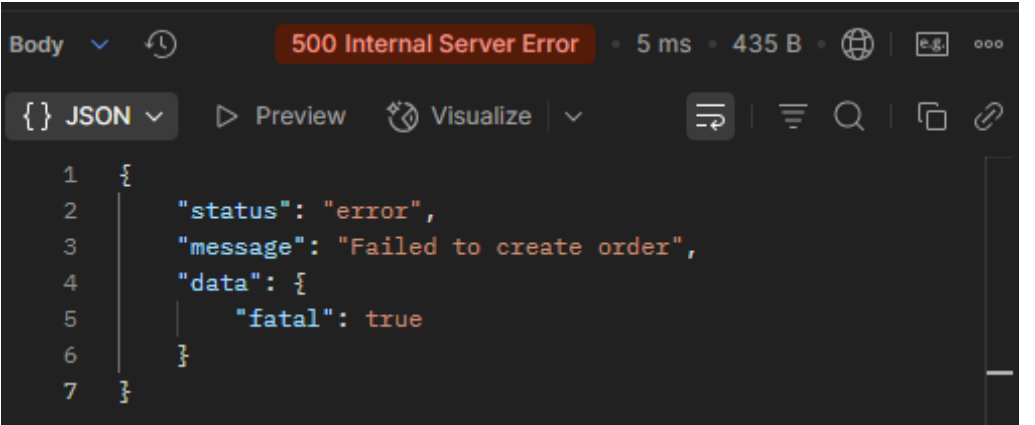
c. Give an example of a **SUCCESS** test scenario and a **FAILED** scenario for the service. Explain each test scenario. Example test case:

Test Case	Input Data	Expected Result
Positif test case	<pre> 1 { 2 "user_id": 1, 3 "address": "Jl. Mawar No. 123, Jember", 4 "total_price": 20000, 5 "items": [6 { 7 "product_id": 1, 8 "quantity": 1, 9 "product_price": 10000 10 }, 11 { 12 "product_id": 2, 13 "quantity": 1, 14 "product_price": 10000 15 } 16] 17 }</pre>	<p>Response Code: 200 OK</p> <p>Response Body:</p> <pre>{ "status": "success", "message": "Order created successfully", "data": {order id} }</pre>
Ada kesalahan server	<pre> 1 { 2 "user_id": 1, 3 "address": "Jl. Mawar No. 123, Jember", 4 "total_price": 20000, 5 "items": [6 { 7 "product_id": 1, 8 "quantity": 1, 9 "product_price": 10000 10 }, 11 { 12 "product_id": 2, 13 "quantity": 1, 14 "product_price": 10000 15 } 16] 17 }</pre> <p>Service Mysql dimatikan</p>	<p>Response Code: 500 Internal Server Error</p> <p>Response Body:</p> <pre>{ "status": "error", "message": "Failed to create order", "data": error message sesuai yang diberikan }</pre>

Request tidak valid	<pre> { "user_id": 0, "address": "Jl. Mawar No. 123, Jember", "total_price": 20000, "items": [{ "product_id": 1, "quantity": 1, "product_price": 10000 }, { "product_id": 2, "quantity": 1, "product_price": 10000 }] } </pre>	Response Code: 400 Bad Request Response Body: <pre> { "status": "error", "message": "Invalid request data" } </pre>
---------------------	--	---

d. Then, show the testing with Postman (or a similar application). Don't forget to include a screenshot.

Test Case	Expected Result
Positif test case	 <p>The screenshot shows the Postman interface with a successful response. The status bar at the top indicates '200 OK' in green, with a response time of 17 ms and a body size of 410 B. The response body is displayed in JSON format:</p> <pre> 1 { 2 "status": "success", 3 "message": "Order created successfully", 4 "data": 61 5 } </pre>

Ada kesalahan server	 <pre> Body ▾ ↺ 400 Bad Request • 4 ms • 401 B • 🌐 📄 ⋮ {} JSON ▾ ▶ Preview 🔗 Visualize ▾ ⌵ ⌵ 🔍 📄 🔗 1 { 2 "status": "error", 3 "message": "Invalid request data" 4 } </pre>
Request tidak valid	 <pre> Body ▾ ↺ 500 Internal Server Error • 5 ms • 435 B • 🌐 📄 ⋮ {} JSON ▾ ▶ Preview 🔗 Visualize ▾ ⌵ ⌵ 🔍 📄 🔗 1 { 2 "status": "error", 3 "message": "Failed to create order", 4 "data": { 5 "fatal": true 6 } 7 } </pre>

4. . Explain in paragraphs and make a video tutorial that discusses how you use the service on a client-side basis using a web application or a mobile phone. Choose one according to your programming skills. (Note: The face must be visible when explaining; otherwise, the score will be ZERO. :))

a. Explain the scenario that will be used for the demo. Provide at least two scenarios, for example, a SUCCESS and a FAILED scenario.

Skenario 1 (Positif Case):

1. Melakukan Register
2. Melakukan Login
3. Menambahkan produk ke Cart
4. Menambahkan / Mengurangi Quantity Item
5. Melanjutkan ke Checkout
6. Mengisi data kelengkapan pesanan dan menyimpan pemesanan
7. Melakukan pembayaran dan upload bukti pembayaran
8. Logout dari user
9. Login dengan role admin
10. Memvalidasi pesanan dengan mengubah status pesanan ke processing
11. Melanjutkan ke shipped dengan mengisi Nomor Resi dan update status
12. Logout dari admin
13. Login sebagai user
14. Cek pesanan di riwayat pesanan, jika barang sudah sampai, ubah status jadi complete.

Skenario 2 (Negatif Case) - Tidak melakukan Pembayaran

1. Melakukan Login
2. Menambahkan produk ke Cart
3. Menambahkan / Mengurangi Quantity Item
4. Melanjutkan ke Checkout
5. Mengisi data kelengkapan pesanan dan menyimpan pemesanan
6. Tidak melakukan pembayaran sampai waktu pembayaran habis
7. Cek status pesanan payment failed

b. Show and explain the application coding that will be demonstrated

Penjelasan akan diterangkan lebih lanjut melalui video

c. Show that the scenario is as expected

Hasil akan ditampilkan lebih lanjut melalui video

5. Attachment

No	Description	Link
a.	Video tutorial on YouTube	https://youtu.be/ma_D8NGWq0c
b.	Source code on GitHub	https://github.com/karismanabil/veganjatim.git