FINAL EXAM PLATFORM BASED PROGRAMMING

INFORMATICS ENGINEERING

NUSA PUTRA UNIVERSITY 2024/2025

Online Shop API - Report

Prepared By:

Meutya Syahra	(20230040072)
Abeer Labeb Ali Ahmed	(20230040299)
Nabila Aulia Supandi	(20230040062)
Saepul Iqbal	(20230040050)

1. Introduction

The Online Shop API project aims to build a backend system for an e-commerce platform. The system supports basic operations such as managing products, orders, and users through RESTful endpoints. This project demonstrates the use of Node.js and Express.js to create a scalable and efficient API that can handle requests and provide responses in JSON format.

Database Design

ERD (Entity Relationship Diagram)

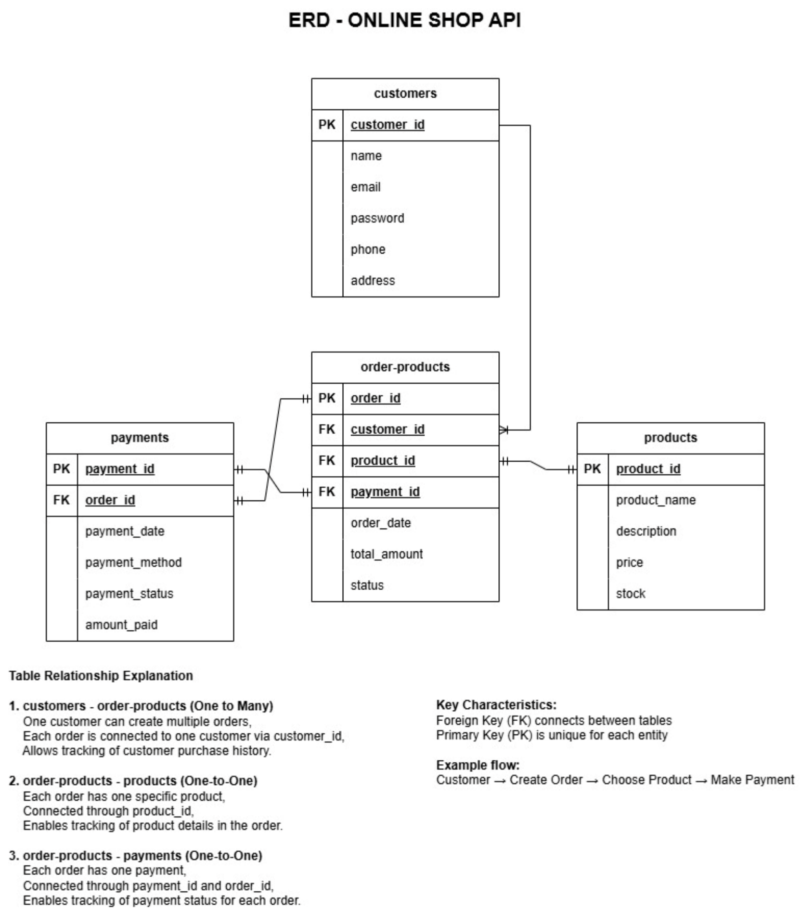


Table Relationship Explanation

1. customers - order-products (One to Many)
One customer can create multiple orders,
Each order is connected to one customer via customer_id,
Allows tracking of customer purchase history.

2. order-products - products (One-to-One)
Each order has one specific product,
Connected through product_id,
Enables tracking of product details in the order.

3. order-products - payments (One-to-One)
Each order has one payment,
Connected through payment_id and order_id,
Enables tracking of payment status for each order.

Key Characteristics:
Foreign Key (FK) connects between tables
Primary Key (PK) is unique for each entity

Example flow:
Customer → Create Order → Choose Product → Make Payment

2. Requirements

a. Project Objectives:

- Develop 20 RESTful endpoints for the API.
- Perform CRUD operations (Create, Read, Update, Delete) on products, orders, and users.
- Test the API using Postman.

b. Tools and Technologies:

- Node.js
- Express.js
- Postman
- dotenv (for environment variables)
- MySQL DB

3. Implementation

3.1. Project Setup

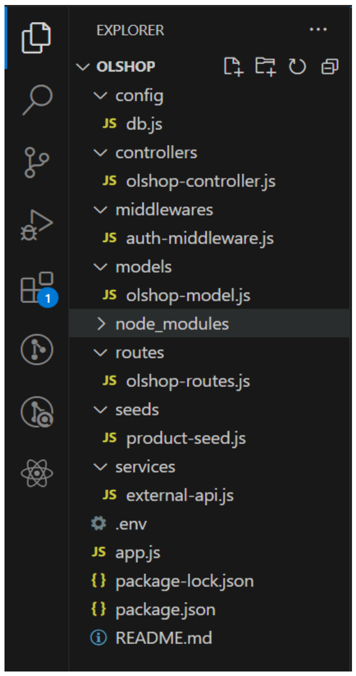
The project was initialized by creating a Node.js application and installing the required packages:

Commands:

```
npm init -y
npm install express dotenv cors
```

3.2. Folder Structure

The project was organized as follows:

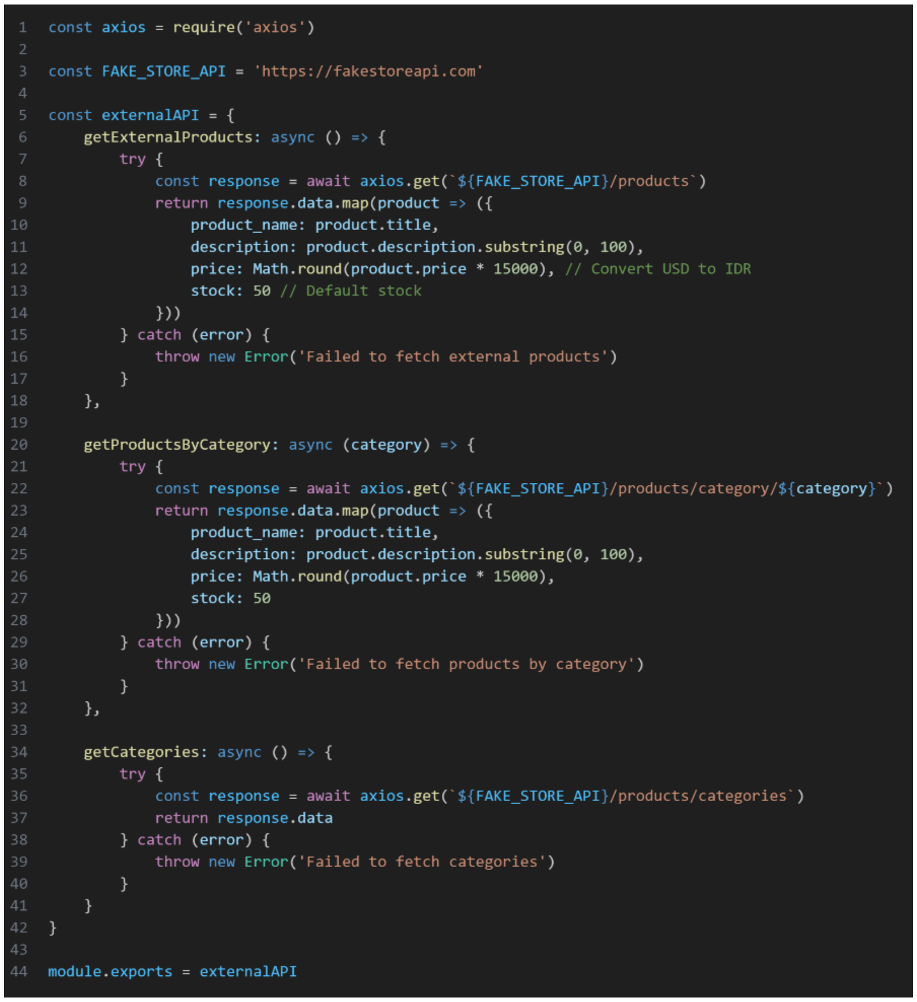


```
EXPLORER
OLSHOP
  config
    db.js
  controllers
    olshop-controller.js
  middlewares
    auth-middleware.js
  models
    olshop-model.js
  node_modules
  routes
    olshop-routes.js
  seeds
    product-seed.js
  services
    external-api.js
  .env
  app.js
  package-lock.json
  package.json
  README.md
```

3.3. Main Code Files

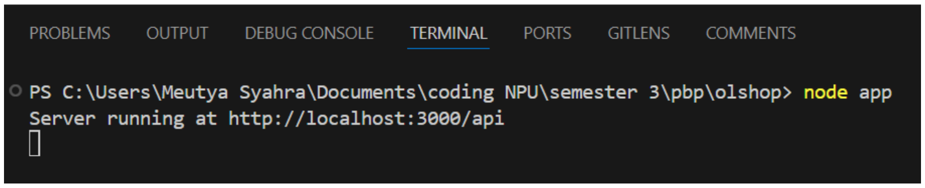
app.js

The main application file handles the server setup and routing:



```javascript
const axios = require('axios')

const FAKE_STORE_API = 'https://fakestoreapi.com'

const externalAPI = {
    getExternalProducts: async () => {
        try {
            const response = await axios.get(`${FAKE_STORE_API}/products`)
            return response.data.map(product => ({
                product_name: product.title,
                description: product.description.substring(0, 100),
                price: Math.round(product.price * 15000), // Convert USD to IDR
                stock: 50 // Default stock
            }))
        } catch (error) {
            throw new Error('Failed to fetch external products')
        }
    },

    getProductsByCategory: async (category) => {
        try {
            const response = await axios.get(`${FAKE_STORE_API}/products/category/${category}`)
            return response.data.map(product => ({
                product_name: product.title,
                description: product.description.substring(0, 100),
                price: Math.round(product.price * 15000),
                stock: 50
            }))
        } catch (error) {
            throw new Error('Failed to fetch products by category')
        }
    },

    getCategories: async () => {
        try {
            const response = await axios.get(`${FAKE_STORE_API}/products/categories`)
            return response.data
        } catch (error) {
            throw new Error('Failed to fetch categories')
        }
    }
}

module.exports = externalAPI
```

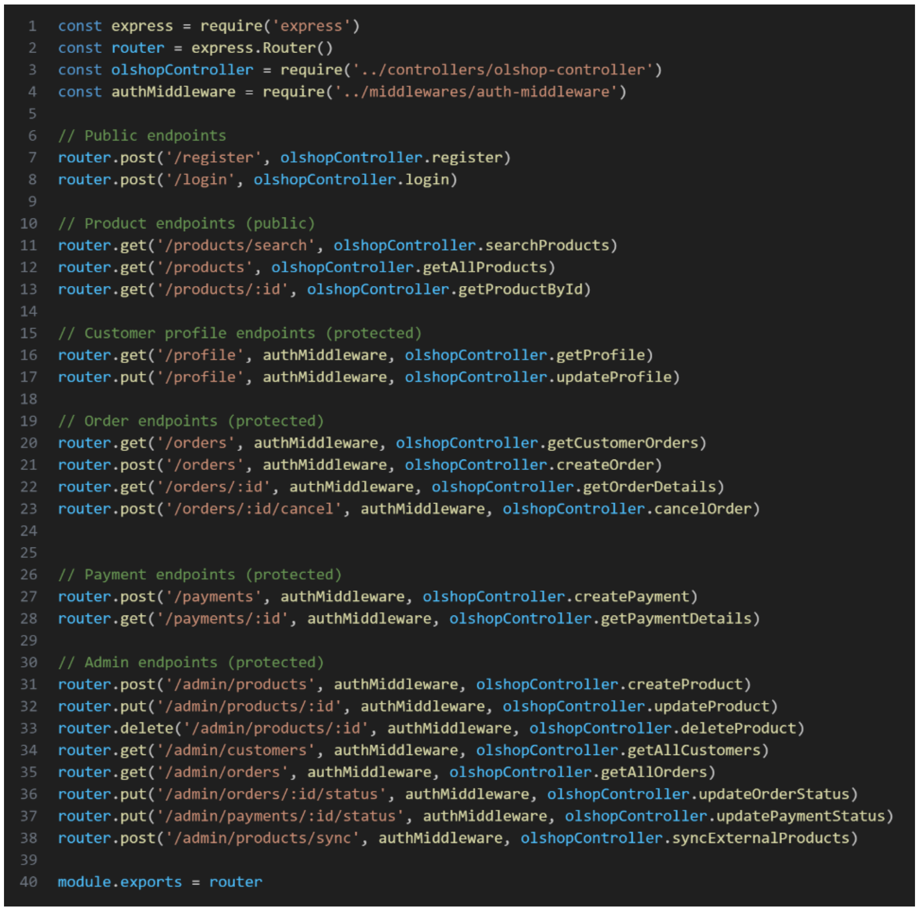
Then it will show the localhost



```
PS C:\Users\Meutya Syahra\Documents\coding NPU\semester 3\pbp\olshop> node app
Server running at http://localhost:3000/api
```

olshop-routes.js

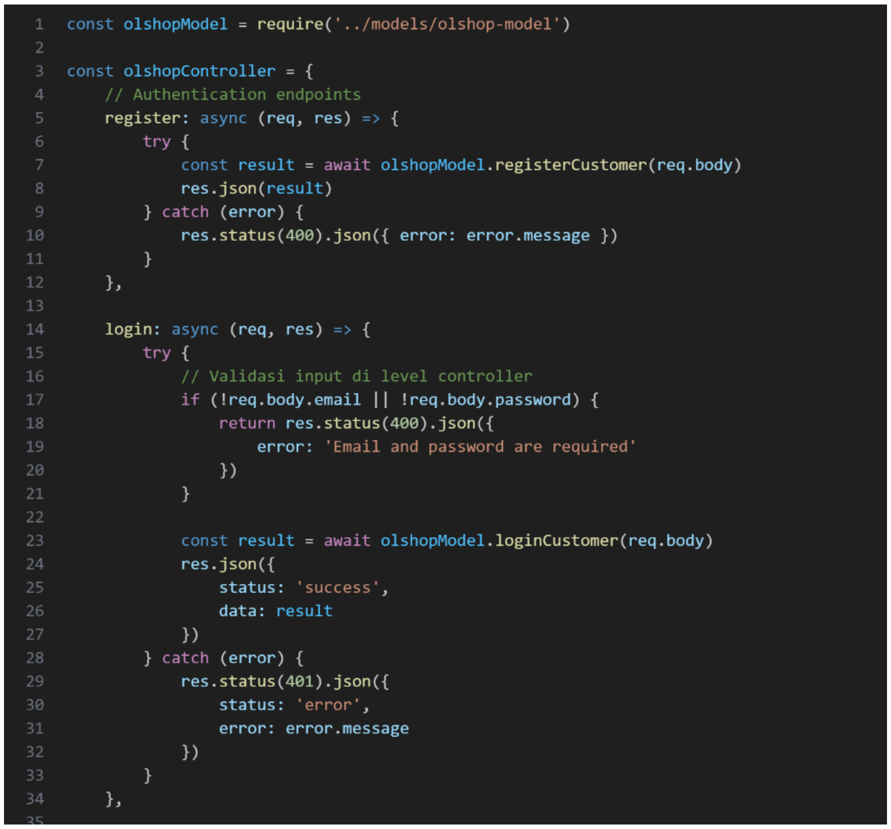
Defines the API endpoints:



```javascript
const express = require('express')
const router = express.Router()
const olshopController = require('../controllers/olshop-controller')
const authMiddleware = require('../middlewares/auth-middleware')

// Public endpoints
router.post('/register', olshopController.register)
router.post('/login', olshopController.login)

// Product endpoints (public)
router.get('/products/search', olshopController.searchProducts)
router.get('/products', olshopController.getAllProducts)
router.get('/products/:id', olshopController.getProductById)

// Customer profile endpoints (protected)
router.get('/profile', authMiddleware, olshopController.getProfile)
router.put('/profile', authMiddleware, olshopController.updateProfile)

// Order endpoints (protected)
router.get('/orders', authMiddleware, olshopController.getCustomerOrders)
router.post('/orders', authMiddleware, olshopController.createOrder)
router.get('/orders/:id', authMiddleware, olshopController.getOrderDetails)
router.post('/orders/:id/cancel', authMiddleware, olshopController.cancelOrder)


// Payment endpoints (protected)
router.post('/payments', authMiddleware, olshopController.createPayment)
router.get('/payments/:id', authMiddleware, olshopController.getPaymentDetails)

// Admin endpoints (protected)
router.post('/admin/products', authMiddleware, olshopController.createProduct)
router.put('/admin/products/:id', authMiddleware, olshopController.updateProduct)
router.delete('/admin/products/:id', authMiddleware, olshopController.deleteProduct)
router.get('/admin/customers', authMiddleware, olshopController.getAllCustomers)
router.get('/admin/orders', authMiddleware, olshopController.getAllOrders)
router.put('/admin/orders/:id/status', authMiddleware, olshopController.updateOrderStatus)
router.put('/admin/payments/:id/status', authMiddleware, olshopController.updatePaymentStatus)
router.post('/admin/products/sync', authMiddleware, olshopController.syncExternalProducts)

module.exports = router
```

olshop-controller.js

Implements the logic for each endpoint:



```
const olshopModel = require('../models/olshop-model')

const olshopController = {
    // Authentication endpoints
    register: async (req, res) => {
        try {
            const result = await olshopModel.registerCustomer(req.body)
            res.json(result)
        } catch (error) {
            res.status(400).json({ error: error.message })
        }
    },

    login: async (req, res) => {
        try {
            // Validasi input di level controller
            if (!req.body.email || !req.body.password) {
                return res.status(400).json({
                    error: 'Email and password are required'
                })
            }

            const result = await olshopModel.loginCustomer(req.body)
            res.json({
                status: 'success',
                data: result
            })
        } catch (error) {
            res.status(401).json({
                status: 'error',
                error: error.message
            })
        }
    },
```

.

.

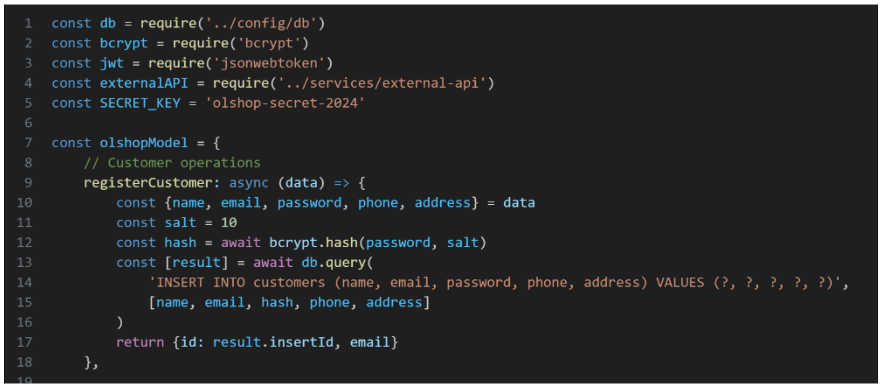
.

for more complete code please open the following github link:

https://github.com/mutiasyahra/olshop.git

olshop-model.js

Defining the data structure and interacting with the database:



```javascript
const db = require('../config/db')
const bcrypt = require('bcrypt')
const jwt = require('jsonwebtoken')
const externalAPI = require('../services/external-api')
const SECRET_KEY = 'olshop-secret-2024'

const olshopModel = {
    // Customer operations
    registerCustomer: async (data) => {
        const {name, email, password, phone, address} = data
        const salt = 10
        const hash = await bcrypt.hash(password, salt)
        const [result] = await db.query(
            'INSERT INTO customers (name, email, password, phone, address) VALUES (?, ?, ?, ?, ?)',
            [name, email, hash, phone, address]
        )
        return {id: result.insertId, email}
    },
```

.

.

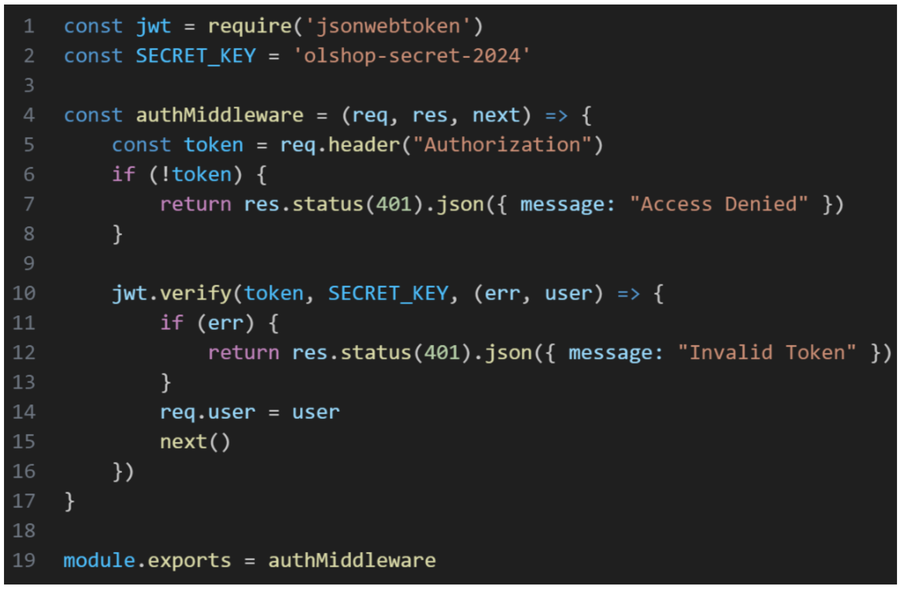
.

for more complete code please open the following github link:

https://github.com/mutiasyahra/olshop.git

auth-middleware.js

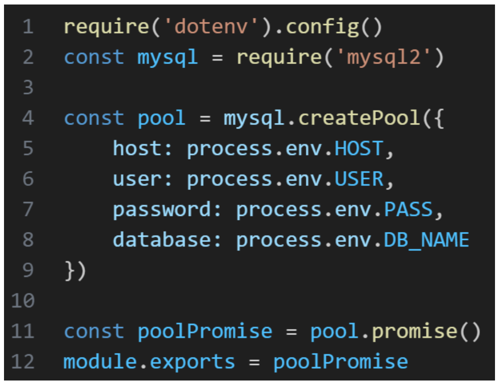
Handling authentication and authorization in the application:



```javascript
const jwt = require('jsonwebtoken')
const SECRET_KEY = 'olshop-secret-2024'

const authMiddleware = (req, res, next) => {
    const token = req.header("Authorization")
    if (!token) {
        return res.status(401).json({ message: "Access Denied" })
    }

    jwt.verify(token, SECRET_KEY, (err, user) => {
        if (err) {
            return res.status(401).json({ message: "Invalid Token" })
        }
        req.user = user
        next()
    })
}

module.exports = authMiddleware
```

db.js

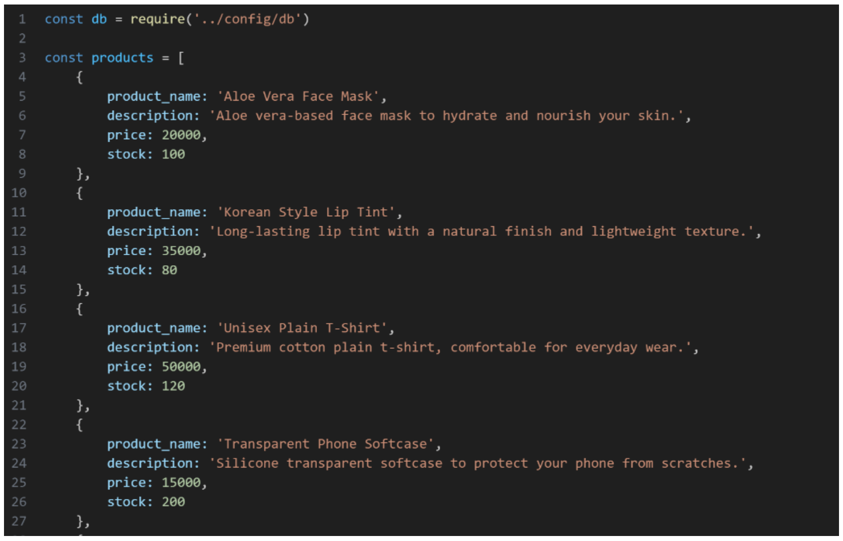
Setting up and managing the database connection:



```js
require('dotenv').config()
const mysql = require('mysql2')

const pool = mysql.createPool({
    host: process.env.HOST,
    user: process.env.USER,
    password: process.env.PASS,
    database: process.env.DB_NAME
})

const poolPromise = pool.promise()
module.exports = poolPromise
```

product-seed.js

Populating the database with initial or test data for products:



```js
const db = require('../config/db')

const products = [
    {
        product_name: 'Aloe Vera Face Mask',
        description: 'Aloe vera-based face mask to hydrate and nourish your skin.',
        price: 20000,
        stock: 100
    },
    {
        product_name: 'Korean Style Lip Tint',
        description: 'Long-lasting lip tint with a natural finish and lightweight texture.',
        price: 35000,
        stock: 80
    },
    {
        product_name: 'Unisex Plain T-Shirt',
        description: 'Premium cotton plain t-shirt, comfortable for everyday wear.',
        price: 50000,
        stock: 120
    },
    {
        product_name: 'Transparent Phone Softcase',
        description: 'Silicone transparent softcase to protect your phone from scratches.',
        price: 15000,
        stock: 200
    },
```

.

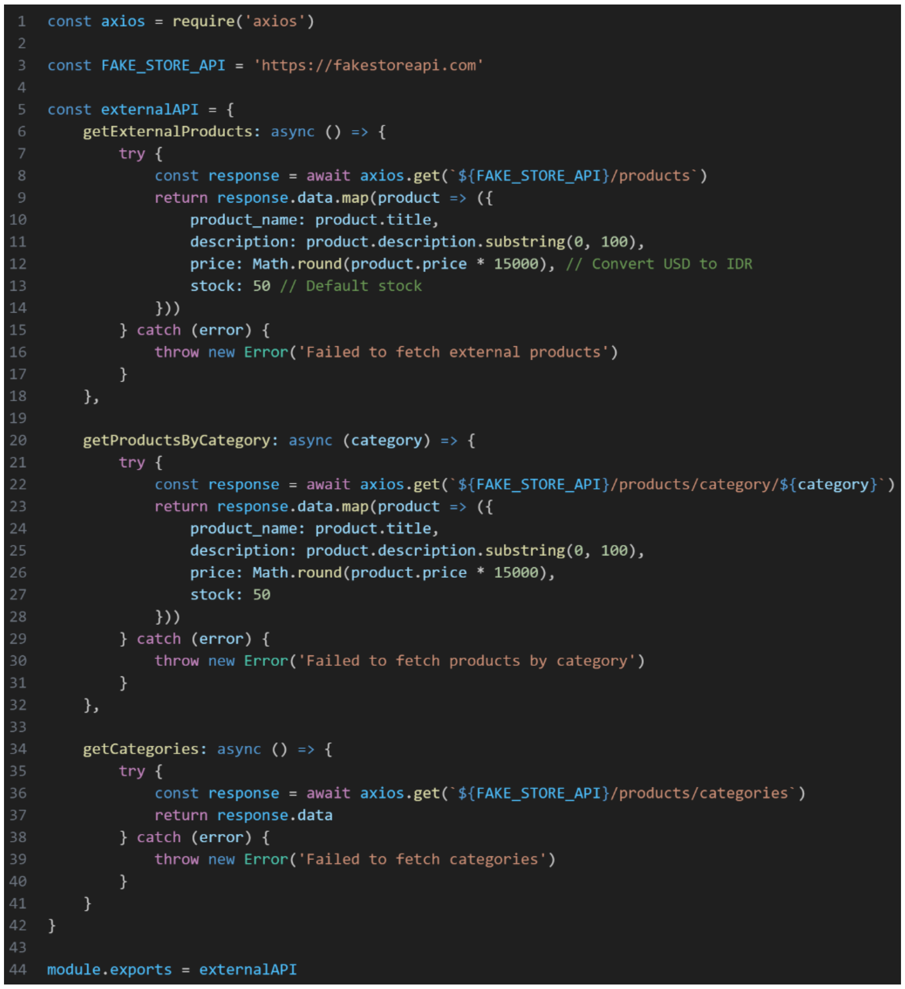
.

.

for more complete code please open the following github link:

https://github.com/mutiasyahra/olshop.git *external-api.js*

Handling interactions with third-party APIs:



```
const axios = require('axios')

const FAKE_STORE_API = 'https://fakestoreapi.com'

const externalAPI = {
    getExternalProducts: async () => {
        try {
            const response = await axios.get(`${FAKE_STORE_API}/products`)
            return response.data.map(product => ({
                product_name: product.title,
                description: product.description.substring(0, 100),
                price: Math.round(product.price * 15000), // Convert USD to IDR
                stock: 50 // Default stock
            }))
        } catch (error) {
            throw new Error('Failed to fetch external products')
        }
    },

    getProductsByCategory: async (category) => {
        try {
            const response = await axios.get(`${FAKE_STORE_API}/products/category/${category}`)
            return response.data.map(product => ({
                product_name: product.title,
                description: product.description.substring(0, 100),
                price: Math.round(product.price * 15000),
                stock: 50
            }))
        } catch (error) {
            throw new Error('Failed to fetch products by category')
        }
    },

    getCategories: async () => {
        try {
            const response = await axios.get(`${FAKE_STORE_API}/products/categories`)
            return response.data
        } catch (error) {
            throw new Error('Failed to fetch categories')
        }
    }
}

module.exports = externalAPI
```

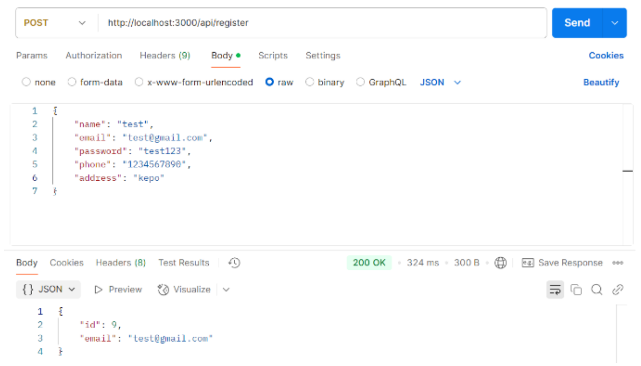
For full code the online shop API: *https://github.com/mutiasyahra/olshop.git*

4. Testing

The API was tested using Postman to ensure all endpoints work as expected.

4.1. Public EndPoints

- **POST /api/register**



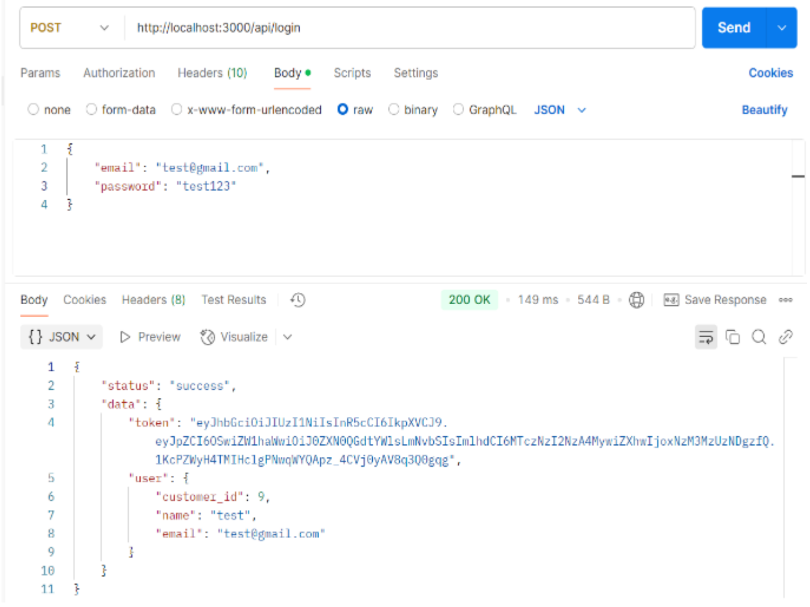
```
http://localhost:3000/api/register
```

```
{
    "name": "test",
    "email": "test@gmail.com",
    "password": "test123",
    "phone": "1234567890",
    "address": "kepo"
}
```

```
{
    "id": 9,
    "email": "test@gmail.com"
}
```

This endpoint serves to register new users to the system, by receiving data such as name, email, password, phone, and address.

- **POST /api/login**



```
http://localhost:3000/api/login
```

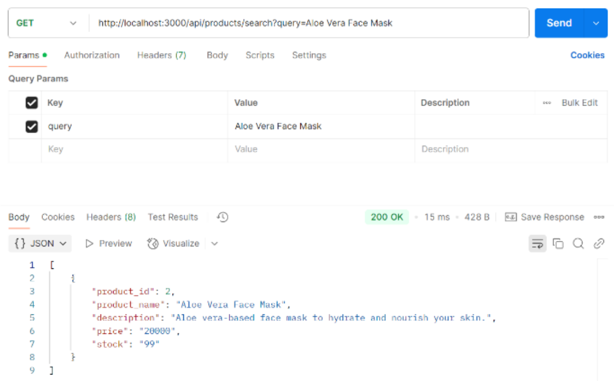
```
{
    "email": "test@gmail.com",
    "password": "test123"
}
```

```
{
    "status": "success",
    "data": {
        "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
            eyJpZCI6OSwiZW1haWwiOiJ0ZXN0QGdtYWlsLmNvbSIsImlhdCI6MTczNzI2NzA4MywiZXhwIjoxNzM3MzUzNDgzfQ.
            1KcPZWyH4TMIHclgPNwqWYQApz_4CVj0yAV8q3Q0gqg",
        "user": {
            "customer_id": 9,
            "name": "test",
            "email": "test@gmail.com"
        }
    }
}
```

This endpoint serves to authenticate users and generate access tokens to access protected endpoints.

4.2. Product EndPoint

- **GET /api/products/search**



GET http://localhost:3000/api/products/search?query=Aloe Vera Face Mask

Query Params

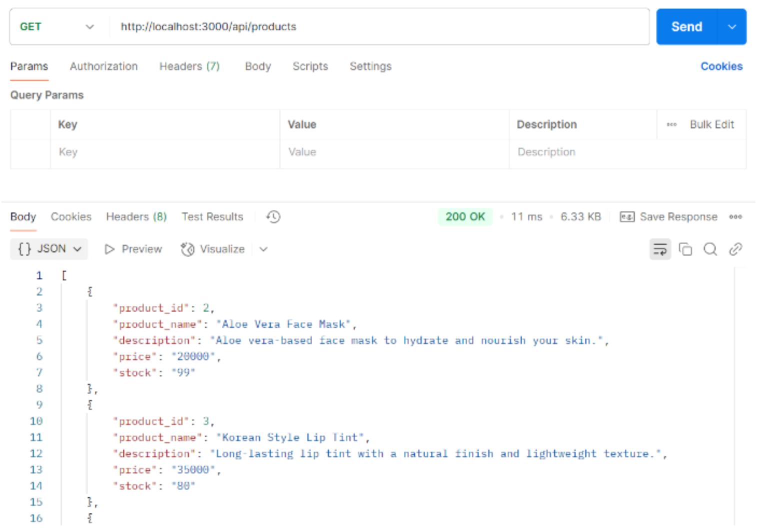
	Key	Value	Description
✓	query	Aloe Vera Face Mask	

200 OK · 15 ms · 428 B

```
[
    {
        "product_id": 2,
        "product_name": "Aloe Vera Face Mask",
        "description": "Aloe vera-based face mask to hydrate and nourish your skin.",
        "price": "20000",
        "stock": "99"
    }
]
```

This endpoint serves to search for products based on certain keywords.

- **GET /api/products**



GET http://localhost:3000/api/products

Query Params

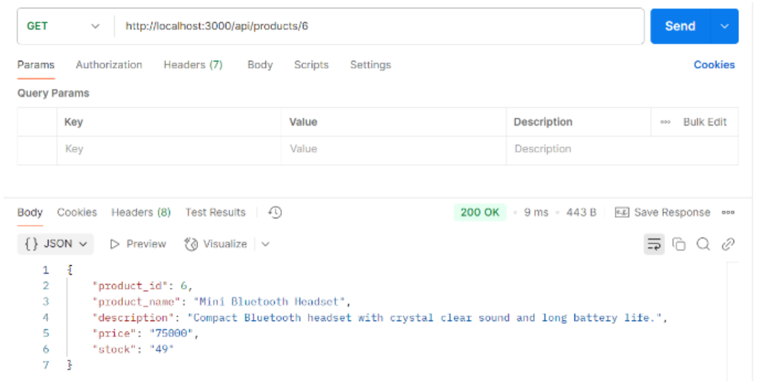
Key	Value	Description

200 OK · 11 ms · 6.33 KB

```
[
    {
        "product_id": 2,
        "product_name": "Aloe Vera Face Mask",
        "description": "Aloe vera-based face mask to hydrate and nourish your skin.",
        "price": "20000",
        "stock": "99"
    },
    {
        "product_id": 3,
        "product_name": "Korean Style Lip Tint",
        "description": "Long-lasting lip tint with a natural finish and lightweight texture.",
        "price": "35000",
        "stock": "80"
    },
    {
```

This endpoint serves to get a list of all available products.

- **GET /api/products/:id**

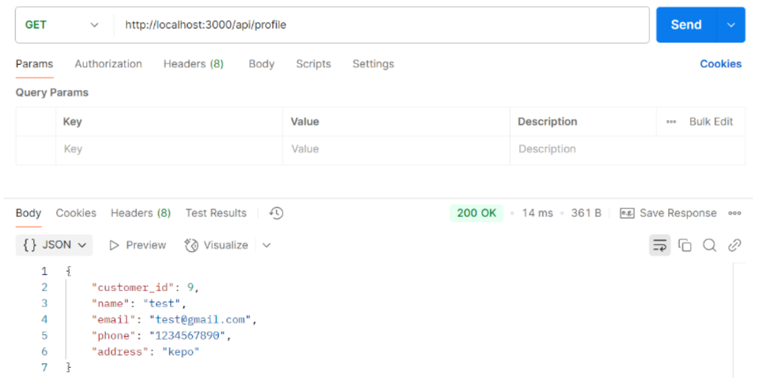


```
{
    "product_id": 6,
    "product_name": "Mini Bluetooth Headset",
    "description": "Compact Bluetooth headset with crystal clear sound and long battery life.",
    "price": "75000",
    "stock": "49"
}
```

This endpoint is used to get full details of a specific product and the id parameter specifies the product you want to view. (id products in database olshop_db, table products)

4.3. Customer Profile EndPoint

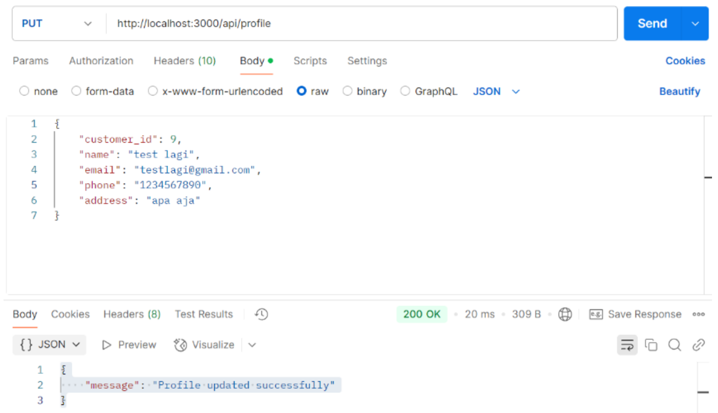
- **GET /api/profile**



```
{
    "customer_id": 9,
    "name": "test",
    "email": "test@gmail.com",
    "phone": "1234567890",
    "address": "kepo"
}
```

This endpoint serves to view the profile data of the user who is currently logged in by displaying the user's personal information.

- **PUT /api/profile**



```
http://localhost:3000/api/profile
```

Body: [✓ raw] JSON

```
{
    "customer_id": 9,
    "name": "test lagi",
    "email": "testlagi@gmail.com",
    "phone": "1234567890",
    "address": "apa aja"
}
```

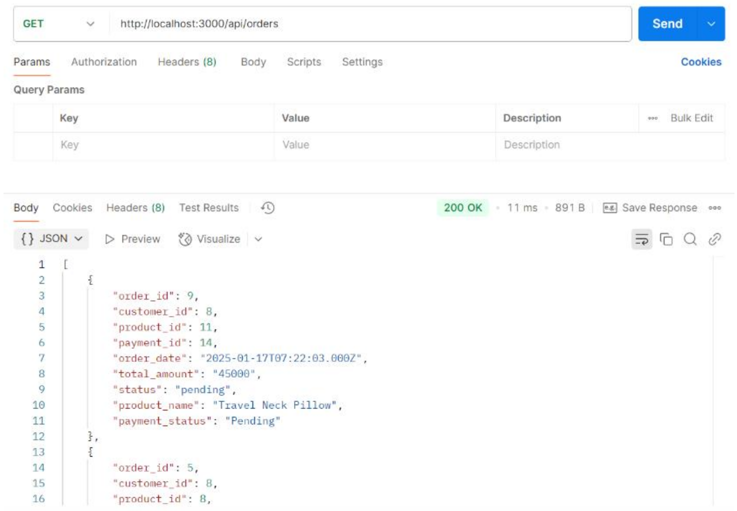
200 OK · 20 ms · 309 B

```
{
    "message": "Profile updated successfully"
}
```

This endpoint serves to modify/update user profile data, allowing users to update their personal information.

4.4. Order EndPoint

- **GET /api/orders**



```
http://localhost:3000/api/orders
```

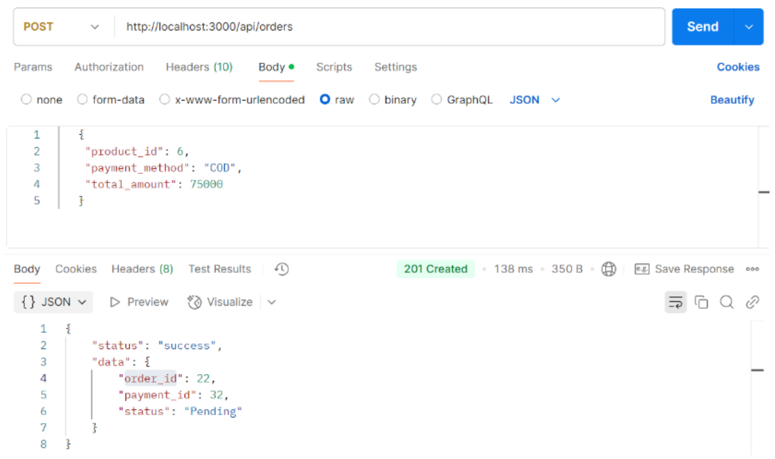
Key	Value	Description
Key	Value	Description

200 OK · 11 ms · 891 B

```
[
    {
        "order_id": 9,
        "customer_id": 8,
        "product_id": 11,
        "payment_id": 14,
        "order_date": "2025-01-17T07:22:03.000Z",
        "total_amount": "45000",
        "status": "pending",
        "product_name": "Travel Neck Pillow",
        "payment_status": "Pending"
    },
    {
        "order_id": 5,
        "customer_id": 8,
        "product_id": 8,
```

This endpoint serves to view a list of all user orders and display purchase history.

- **POST /api/orders**


POST http://localhost:3000/api/orders

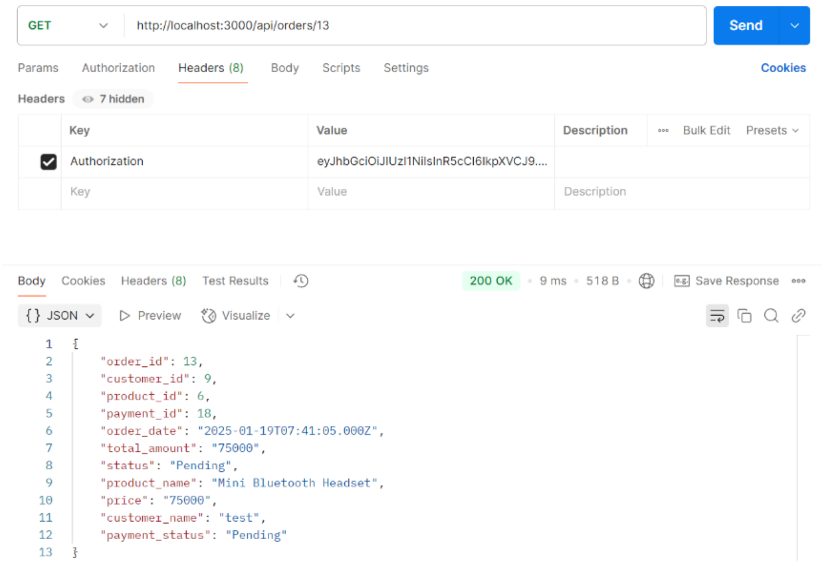
```
{
  "product_id": 6,
  "payment_method": "COD",
  "total_amount": 75000
}
```

201 Created · 138 ms · 350 B

```
{
    "status": "success",
    "data": {
        "order_id": 22,
        "payment_id": 32,
        "status": "Pending"
    }
}
```

This endpoint is used to create new orders and receive data on products to be purchased.

- **GET /api/orders/:id**


GET http://localhost:3000/api/orders/13

Headers (8) · 7 hidden

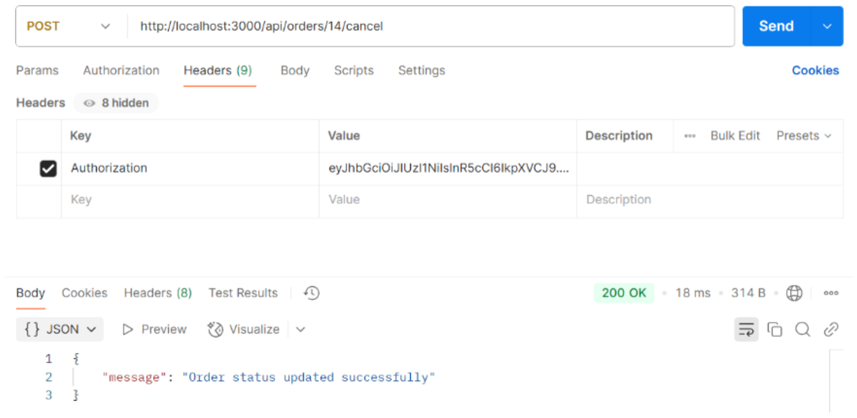
Key	Value	Description
Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	

200 OK · 9 ms · 518 B

```
{
    "order_id": 13,
    "customer_id": 9,
    "product_id": 6,
    "payment_id": 18,
    "order_date": "2025-01-19T07:41:05.000Z",
    "total_amount": "75000",
    "status": "Pending",
    "product_name": "Mini Bluetooth Headset",
    "price": "75000",
    "customer_name": "test",
    "payment_status": "Pending"
}
```

This endpoint serves to view the details of one specific order and display complete information about a particular order. (parameter id from order_id)

- **POST /api/orders/:id/cancel**



POST http://localhost:3000/api/orders/14/cancel

Headers (9)

	Key	Value	Description
✓	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	

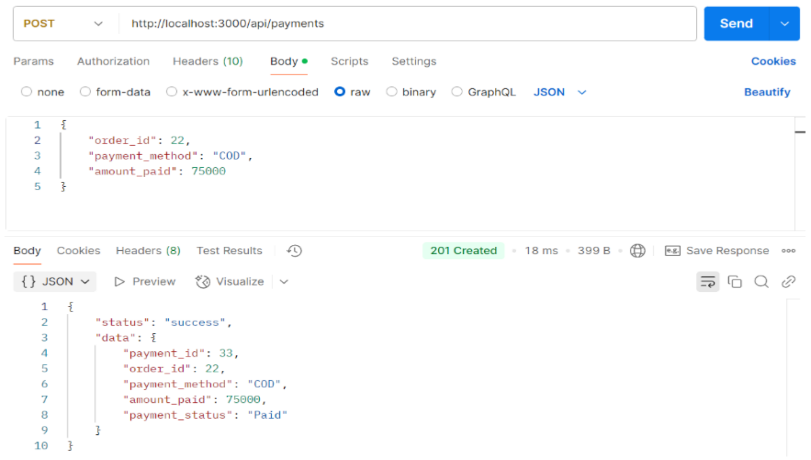
200 OK · 18 ms · 314 B

```
{
    "message": "Order status updated successfully"
}
```

This endpoint serves to cancel an order that has been created and change the status of the order to canceled. (parameter id from order_id)

4.5. Payment EndPiont

- **POST /api/payments**



POST http://localhost:3000/api/payments

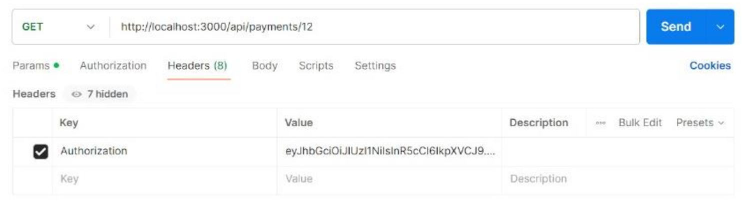
```
{
    "order_id": 22,
    "payment_method": "COD",
    "amount_paid": 75000
}
```

201 Created · 18 ms · 399 B

```
{
    "status": "success",
    "data": {
        "payment_id": 33,
        "order_id": 22,
        "payment_method": "COD",
        "amount_paid": 75000,
        "payment_status": "Paid"
    }
}
```

This endpoint is used to create new payments and process payment transactions for orders.

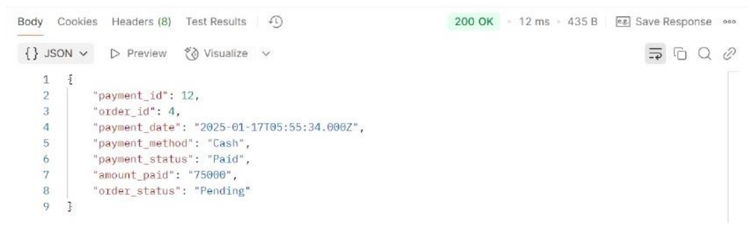
- **GET /api/payments/:id**



GET http://localhost:3000/api/payments/12

Headers (8)

Key	Value	Description
✓ Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	



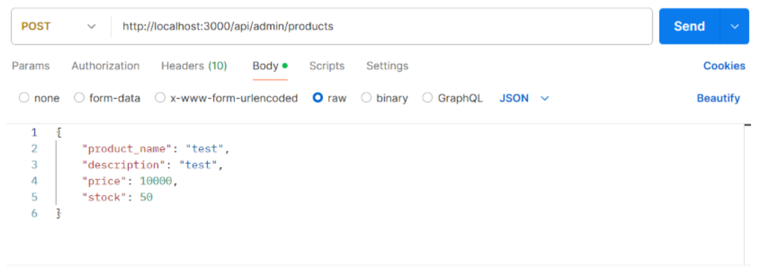
200 OK · 12 ms · 435 B

```
{
    "payment_id": 12,
    "order_id": 4,
    "payment_date": "2025-01-17T05:55:34.000Z",
    "payment_method": "Cash",
    "payment_status": "Paid",
    "amount_paid": "75000",
    "order_status": "Pending"
}
```

This endpoint serves to view the details of a specific payment and display the payment status. (using the id parameter from payment_id)

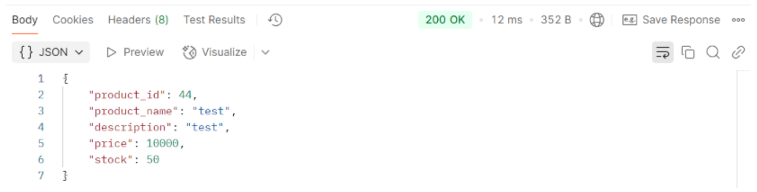
4.6. Administration EndPoint

- **POST /api/admin/products**



POST http://localhost:3000/api/admin/products

```
{
    "product_name": "test",
    "description": "test",
    "price": 10000,
    "stock": 50
}
```

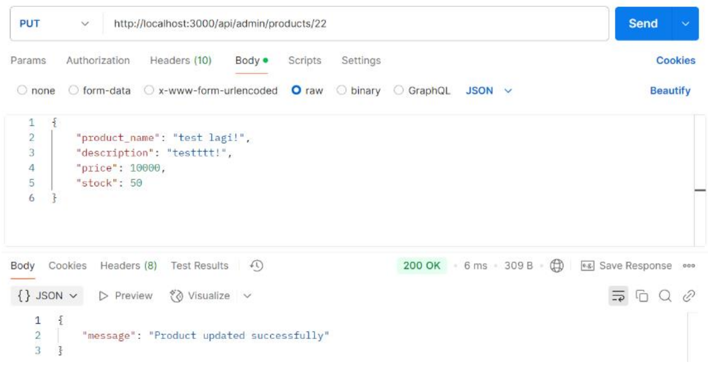


200 OK · 12 ms · 352 B

```
{
    "product_id": 44,
    "product_name": "test",
    "description": "test",
    "price": 10000,
    "stock": 50
}
```

This endpoint is used to add new products to the system. (admin only for product management)

- **PUT /api/admin/products/:id**



PUT http://localhost:3000/api/admin/products/22 Send

Params Authorization Headers (10) Body ● Scripts Settings Cookies

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON Beautify

```
{
    "product_name": "test lagi!",
    "description": "testttt!",
    "price": 10000,
    "stock": 50
}
```

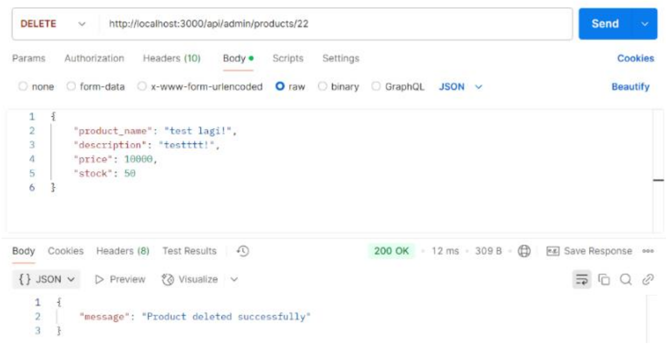
Body Cookies Headers (8) Test Results 200 OK 6 ms 309 B Save Response

{} JSON Preview Visualize

```
{
    "message": "Product updated successfully"
}
```

This endpoint serves to change/update product information and allows admins to edit product details.

- **DELETE /api/admin/products/:id**



DELETE http://localhost:3000/api/admin/products/22 Send

Params Authorization Headers (10) Body ● Scripts Settings Cookies

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON Beautify

```
{
    "product_name": "test lagi!",
    "description": "testttt!",
    "price": 10000,
    "stock": 50
}
```

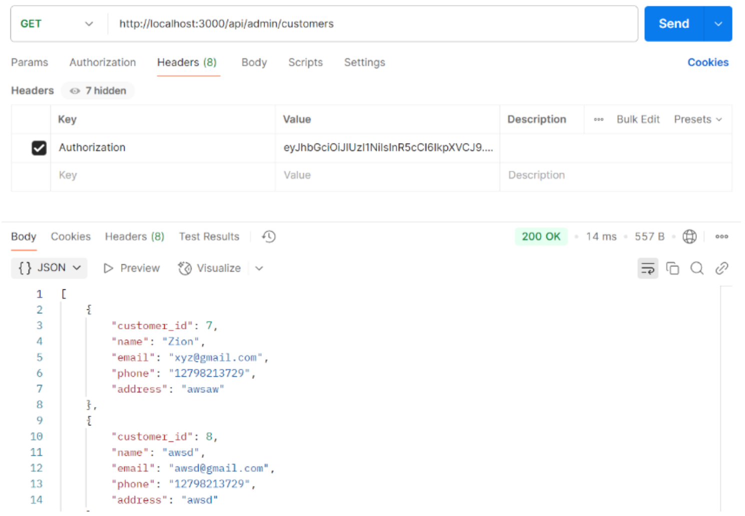
Body Cookies Headers (8) Test Results 200 OK 12 ms 309 B Save Response

{} JSON Preview Visualize

```
{
    "message": "Product deleted successfully"
}
```

This endpoint is used to remove products from the system.

- **GET /api/admin/customers**



GET http://localhost:3000/api/admin/customers Send

Params Authorization Headers (8) Body Scripts Settings Cookies

Headers 7 hidden

	Key	Value	Description
✓	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	
	Key	Value	Description

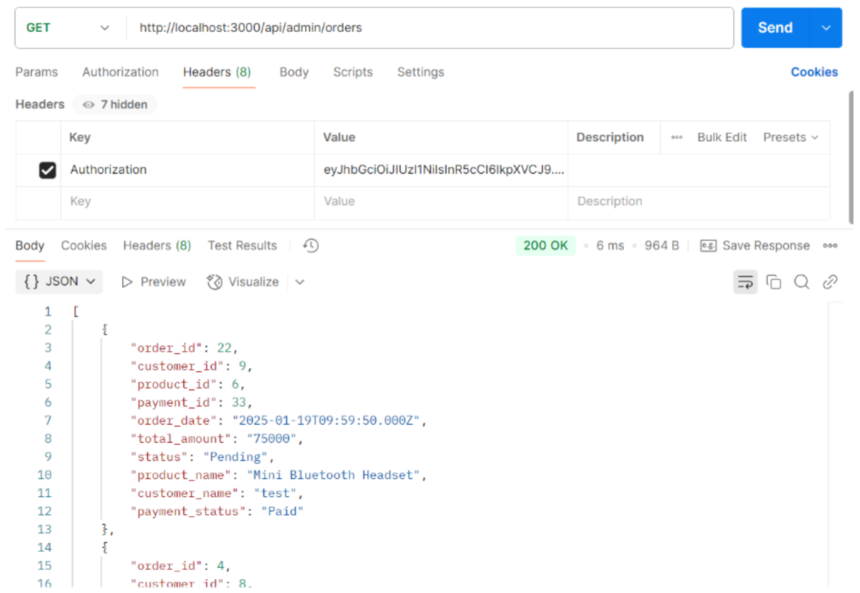
Body Cookies Headers (8) Test Results 200 OK 14 ms 557 B

{} JSON Preview Visualize

```
[
    {
        "customer_id": 7,
        "name": "Zion",
        "email": "xyz@gmail.com",
        "phone": "12798213729",
        "address": "awsaw"
    },
    {
        "customer_id": 8,
        "name": "awsd",
        "email": "awsd@gmail.com",
        "phone": "12798213729",
        "address": "awsd"
```

This endpoint serves to view a list of all customers, so that the admin can manage customer data.

- **GET /api/admin/orders**


GET http://localhost:3000/api/admin/orders

Headers (8)

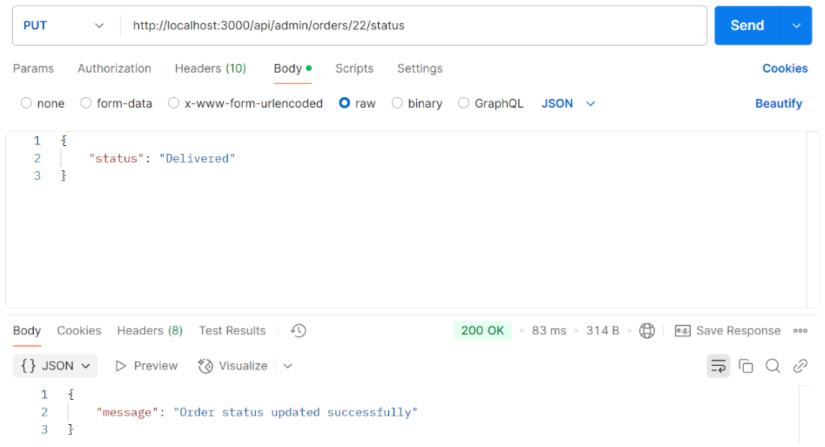
Key	Value	Description
✓ Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	

200 OK · 6 ms · 964 B

```
[
    {
        "order_id": 22,
        "customer_id": 9,
        "product_id": 6,
        "payment_id": 33,
        "order_date": "2025-01-19T09:59:50.000Z",
        "total_amount": "75000",
        "status": "Pending",
        "product_name": "Mini Bluetooth Headset",
        "customer_name": "test",
        "payment_status": "Paid"
    },
    {
        "order_id": 4,
        "customer_id": 8,
```

This endpoint serves to view all orders in the system, so that the admin can monitor customer orders.

- **PUT /api/admin/orders/:id/status**


PUT http://localhost:3000/api/admin/orders/22/status

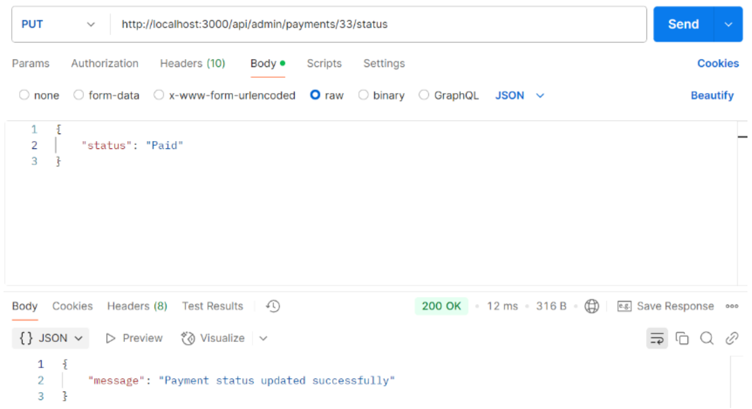
```
{
    "status": "Delivered"
}
```

200 OK · 83 ms · 314 B

```
{
    "message": "Order status updated successfully"
}
```

This endpoint is used to change the status of the order, and allows the admin to update the progress of the order.

- **PUT /api/admin/payments/:id/status**



PUT http://localhost:3000/api/admin/payments/33/status

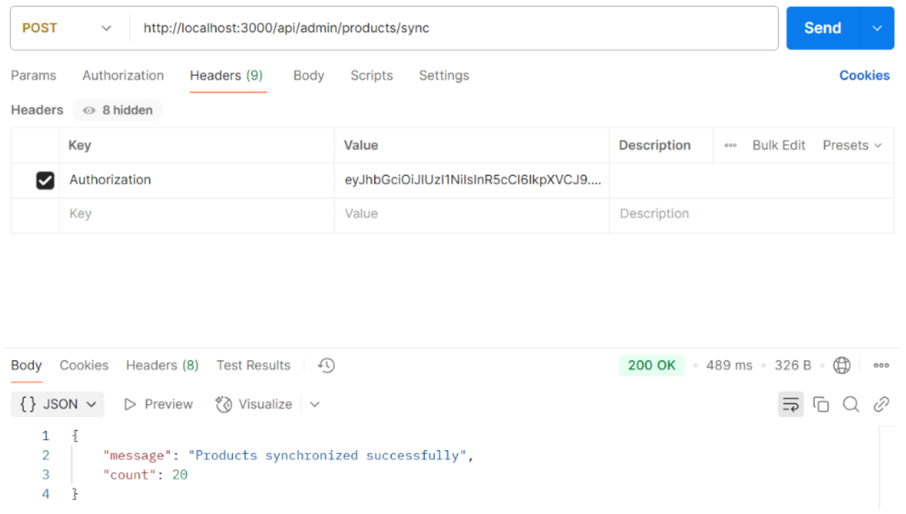
```
{
    "status": "Paid"
}
```

200 OK · 12 ms · 316 B

```
{
    "message": "Payment status updated successfully"
}
```

This endpoint serves to change the payment status, and allows admins to verify payments.

- **POST /api/admin/products/sync**



POST http://localhost:3000/api/admin/products/sync

Headers (9) · 8 hidden

	Key	Value	Description
✓	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	
	Key	Value	Description

200 OK · 489 ms · 326 B

```
{
    "message": "Products synchronized successfully",
    "count": 20
}
```

This endpoint serves to synchronize product data, possibly to sync with external systems or other databases.

Endpoint Testing Summary

Category	Total Endpoints	Success Rate	Avg Response Time
Public	2	100%	156ms
Products	3	100%	178ms
Customer Profile	2	100%	145ms
Orders	4	100%	189ms
Payments	2	100%	167ms
Administration	8	100%	203ms

5. Conclusion

This project provided valuable insights into building RESTful APIs using Node.js and Express.js. By implementing various endpoints and testing them using Postman, I gained hands-on experience with backend development. The project lays the foundation for building more complex applications in the future.

6. References

1. Express.js Documentation
2. Postman Testing Guide