INDUSTRIAL TRAINING PROJECT

eCommerce REST API

BRAINWARE UNIVERSITY

In partial fulfillment of the requirement for the award of degree of Diploma in Computer Science and Engineering

Submitted By

SOURAV SHARMA – BWU/DCS/22/119
RAJANKITA SAHA – BWU/DCS/22/108
SUBHAYAN MANDAL – BWU/DCS/22/131
SWAGATA DAS – BWU/DCS/22/092
KARTIK PAUL – BWU/DCS/22/062

INDEX

1.Introduction

This project focuses on building a REST API for eCommerce functionalities, including user authentication, product management, review management, and order management. The system allows users to register, log in, manage products, create and manage reviews, and place and manage orders. It leverages modern technologies such as Node.js, Express.js, MongoDB, and JWT to provide a robust, scalable, and secure solution. JWT-based authentication ensures secure access to the system, making it ideal for handling sensitive user and transaction data.

2. Technology Stack

The project uses the following technologies:

- **Backend**:
    - **Node.js**: A JavaScript runtime built on Chrome's V8 engine, used to build scalable network applications. It provides an event-driven architecture capable of handling multiple requests simultaneously.
    - **Express.js**: A minimal and flexible Node.js web application framework, which simplifies building robust web APIs. It is used to handle routing, middleware, and HTTP requests in this project.
- **Database**:
    - **MongoDB**: A NoSQL database that stores data in a flexible, JSON-like format, making it ideal for dynamic applications. In this project, MongoDB is used for storing user and application data.
    - **Mongoose**: An ODM (Object Data Modeling) library for MongoDB and Node.js. It provides a schema-based solution to model application data and interacts seamlessly with MongoDB.
- **Authentication**:
    - **JWT (JSON Web Tokens)**: A compact and self-contained way to securely transmit information between parties as a JSON object. JWT is used in this project for authentication purposes, enabling secure API access.
- **Image Upload**:
    - **Cloudinary API**: A cloud-based image and video management platform. Cloudinary is integrated into the project to handle image uploads, storage, and optimization, ensuring high performance and scalability.
- **Environment Management**:
    - **dotenv**: A zero-dependency module for loading environment variables from a .env file into process.env. It is used to manage sensitive configuration values like database credentials and API keys securely.

3. Roadmap

User Authentication

- **Objective**: Implement secure user authentication to allow users to register, log in, and log out.
- **Tasks**:
  - Secure registration with email and password.
  - Implement JWT-based authentication for session management.
  - Provide functionality for secure login and token issuance.
  - Implement logout functionality to invalidate JWT tokens.

Product Management

- **Objective**: Provide CRUD (Create, Read, Update, Delete) operations for managing products.
- **Tasks:**
  - Implement routes to add, update, and delete products.
  - Include product details such as name, price, description, and stock quantity.
  - Integrate image upload functionality for product images via Cloudinary API.
  - Allow users to view products with pagination and search filters.

Review Management

- **Objective**: Enable users to manage product reviews through CRUD operations.
- **Tasks**:
  - Implement routes for creating, updating, and deleting reviews.
  - Allow users to rate products and write reviews.
  - Display reviews for each product with the ability to sort and filter based on ratings.

Order Management

- **Objective:** Implement order creation, updates, and retrieval.
- **Tasks:**
  - Provide functionality to create orders with product details and user information.
  - Enable order updates (status changes, cancellations, etc.).
  - Implement routes to retrieve orders by user or order ID.
  - Implement basic order status tracking (e.g., pending, shipped, delivered).

API Testing

- **Objective**: Ensure robustness, reliability, and security through comprehensive testing.
- **Tasks**:
  - Write unit tests for individual API routes and functionalities.
  - Implement integration tests to ensure smooth interaction between different API components.
  - Perform security tests such as JWT validation, input sanitization, and vulnerability scanning.
  - Use tools like Mocha, Chai, and Supertest for automated testing.

Deployment

- **Objective**: Deploy the API on cloud platforms for production use.
- **Tasks**:
  - Set up deployment pipelines for continuous integration and delivery (CI/CD).
  - Deploy the API to cloud platforms like **Heroku** or **AWS**.
  - Configure environment variables and database connections for production.
  - Ensure deployment includes logging and monitoring for production health.
  -

Future Enhancements

- **Objective**: Add advanced features to further enhance the eCommerce platform.
- **Tasks**:
    - Integrate a **payment gateway** (e.g., Stripe, PayPal) for handling payments securely.
    - Implement **order history** functionality for users to track past orders.
    - Introduce **advanced user roles** (e.g., admin, seller, buyer) with different permissions.
    - Explore adding **real-time features** (e.g., order status updates, notifications).
    - Implement **product recommendations** based on user activity or ratings.

4. Features of the Project

- **User Registration and Authentication:** Users can register, login, and securely authenticate via JWT.
- **Product CRUD:** Create, update, delete, and retrieve product details, including image uploads.
- **Review Management:** Allow users to create, update, delete, and retrieve reviews for products.
- **Order Management:** Users can place, update, and retrieve orders.
- **JWT-based Authentication:** Secure access for users with token-based authentication.

5. Feature Solutions

We provide tailored solutions to meet specific needs, ensuring enhanced functionality, performance, and user experience. Key features include:

- **Customization**: Tailor features to meet unique business or user requirements.
- **Scalability**: Support for future growth and expansion with seamless integration.
- **Security**: Robust security protocols to protect data and ensure safe transactions.
- **User-Friendly Interface**: Intuitive design for improved user engagement.
- **Real-Time Performance**: Fast, efficient solutions for real-time data processing.

6. Test Cases

1. User Authentication Tests

Test Case 1.1: User Registration

Send a POST request to /register with valid user details (e.g., email, password).

Input:

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

Expected Result:

- Response status: 201 (Created)
- Response message: "User successfully registered."
- User added to the database.

Test Case 1.2: User Registration (Missing Fields)

Send a POST request to /register with missing fields (e.g., missing password).

Input:

```
{
  "email": "user@example.com"
}
```

Expected Result:

- Response status: 400 (Bad Request)
- Error message: "Password is required."

Test Case 1.3: User Login

Send a POST request to /login with valid credentials (email and password).

Input:

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

Expected Result:

- Response status: 200 (OK)
- Response includes a JWT token for authentication.

Test Case 1.4: User Login (Invalid Credentials)

Send a POST request to /login with invalid credentials.

Input:

```
{
  "email": "user@example.com",
  "password": "wrongpassword"
}
```

Expected Result:

- Response status: 401 (Unauthorized)
- Error message: "Invalid email or password."

2. Product Management Tests

Test Case 2.1: Add Product (Admin)

Send a POST request to /products to add a new product with valid data (only accessible by admin).

Input:

```
{
  "name": "Laptop",
  "price": 1200,
  "category": "Electronics",
  "description": "High-end gaming laptop"
}
```

Expected Result:

- Response status: 201 (Created)
- Response includes the product details.
- Product added to the database.

Test Case 2.2: Add Product (Without Admin Privileges)

Send a POST request to /products to add a new product without admin privileges.

Input:

```
{
  "name": "Phone",
  "price": 500,
  "category": "Electronics"
}
```

Expected Result:

- Response status: 403 (Forbidden)
- Error message: "Admin privileges required."

Test Case 2.3: Get Product List

Send a GET request to /products to retrieve the list of products.

Input: No input required.

Expected Result:

- Response status: 200 (OK)
- Response includes an array of products with details like name, price, and category.

3. Review Management Tests

Test Case 3.1: Add Product Review

Send a POST request to /reviews to add a review for a product.

Input:

```
{
  "productId": 1,
  "rating": 5,
  "review": "Great product!"
}
```

Expected Result:

- Response status: 201 (Created)
- Response includes the review details.

Test Case 3.2: Get Product Reviews

Send a GET request to /reviews?productId=1 to retrieve all reviews for a product.

Input: No input required.

Expected Result:

- Response status: 200 (OK)
- Response includes a list of reviews for the product.

Error Codes

The following are the possible error codes returned by the API:

- **400** - Bad Request
- **401** - Unauthorized
- **404** - Not Found
- **500** - Internal Server Error

7. Project Review

Overview

This eCommerce REST API is built using Node.js and Express.js to provide essential functionalities for an online store. It focuses on user authentication, product management, review management, and order management. The system leverages JWT-based authentication to ensure secure access and facilitate smooth user interactions. This API is designed to manage users, products, reviews, and orders efficiently, with roles and permissions to secure sensitive operations.

Strengths

- **Scalability:** The REST API is designed to be scalable, making it suitable for handling a growing number of products, users, and orders.
- **Security:** JWT authentication ensures that only authorized users can access and perform actions in the system.
- **User-Centric:** Provides an intuitive user interface for product browsing, reviews, and order management.
- **Efficient Management:** Simplifies product and review management for administrators, providing a secure and structured way to manage data.

Challenges

- **Error Handling:** Improved error handling and clear responses for invalid requests can be implemented for better user experience.
- **Rate Limiting**: To prevent abuse of the system, implementing rate limiting for API requests could enhance security and reduce the risk of DDoS attacks.
- **Extensibility:** Further features like payment gateway integration and shipment tracking could be considered in future updates.

Future Enhancements

- **Payment Integration:** Adding a payment gateway (e.g., PayPal, Stripe) for handling transactions.
- **Inventory Management:** Integrating real-time stock tracking and alerts for low stock levels.
- **Mobile App Integration:** Building a mobile app that interacts with the API to provide a seamless shopping experience on smartphones.
- **Product Search and Filter:** Implementing advanced search and filter functionality for products based on various criteria like price range, category, and ratings.

8. Conclusion

This eCommerce REST API built using Node.js and Express.js offers a secure, scalable, and efficient solution for managing users, products, reviews, and orders. By implementing JWT authentication, it ensures secure access to the system. While the project successfully meets the essential eCommerce requirements, there is room for future enhancements, such as adding payment gateway integrations, inventory management, and mobile app support. Overall, the system lays a solid foundation for building a robust online store, and its flexibility allows for future growth and improvements.

9. Reference

- Express.js Official Documentation
- JWT Official Introduction
- MongoDB Official Website
- JavaScript Modules - MDN Web Docs
- Postman API Testing Tool
- DigitalOcean Tutorials
- W3Schools - Web Development Tutorials