**E-Commerce API**

This project is a Laravel-based RESTful API for managing a simple e-commerce system. The API provides endpoints for user registration and authentication, CRUD operations for products, and order management. This README.md provides an overview of the API, database setup, code review, and documentation.

**Table of Contents**

1. [Project Overview](#project-overview)
2. [Installation and Setup](#installation-and-setup)
3. [Database Setup](#database-setup)
4. [API Endpoints](#api-endpoints)
5. [Code Review Report](#code-review-report)
6. [API Documentation](#api-documentation)

**Project Overview**

The e-commerce API includes the following features:

* **User Registration and Authentication**: Secure user registration and JWT-based authentication.
* **Product Management**: CRUD operations for products.
* **Order Management**: Ability to place orders with one or multiple products.

**Installation and Setup**

Follow these steps to set up and run the project locally:

1. **Clone the Repository:**

bash

Copy code

git clone https://github.com/your-username/e-commerce-api.git

cd e-commerce-api

1. **Install Dependencies:**

bash

Copy code

composer install

1. **Set Up Environment File:**

Copy the example environment file and update the configuration:

bash

Copy code

cp .env.example .env

Generate the application key:

bash

Copy code

php artisan key:generate

1. **Set Up JWT Secret:**

Generate the JWT secret key:

bash

Copy code

php artisan jwt:secret

1. **Run Migrations and Seed the Database:**

bash

Copy code

php artisan migrate

php artisan db:seed

1. **Start the Laravel Development Server:**

bash

Copy code

php artisan serve

The API will be accessible at http://localhost:8000/api.

**Database Setup**

To create the database tables, use the provided SQL commands:

sql

Copy code

CREATE TABLE users (

id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

email\_verified\_at TIMESTAMP NULL,

password VARCHAR(255) NOT NULL,

remember\_token VARCHAR(100) NULL,

created\_at TIMESTAMP NULL,

updated\_at TIMESTAMP NULL

);

CREATE TABLE products (

id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT NOT NULL,

price DECIMAL(8, 2) NOT NULL,

quantity INT NOT NULL,

created\_at TIMESTAMP NULL,

updated\_at TIMESTAMP NULL

);

CREATE TABLE orders (

id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

user\_id BIGINT UNSIGNED,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE,

created\_at TIMESTAMP NULL,

updated\_at TIMESTAMP NULL

);

CREATE TABLE order\_product (

id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

order\_id BIGINT UNSIGNED,

product\_id BIGINT UNSIGNED,

quantity INT NOT NULL,

FOREIGN KEY (order\_id) REFERENCES orders(id) ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES products(id) ON DELETE CASCADE,

created\_at TIMESTAMP NULL,

updated\_at TIMESTAMP NULL

);

**API Endpoints**

**User Registration and Authentication**

* **Register User**
  + POST /api/register
  + Request Body: { "name": "John Doe", "email": "john@example.com", "password": "password" }
  + Response: 200 OK with user details.
* **Login**
  + POST /api/login
  + Request Body: { "email": "john@example.com", "password": "password" }
  + Response: 200 OK with JWT token.
* **Get User Details**
  + GET /api/user
  + Headers: Authorization: Bearer {token}
  + Response: 200 OK with user details.
* **Logout**
  + POST /api/logout
  + Headers: Authorization: Bearer {token}
  + Response: 200 OK with success message.

**Product Management**

* **List Products**
  + GET /api/products
  + Response: 200 OK with list of products.
* **Create Product**
  + POST /api/products
  + Request Body: { "name": "Product Name", "description": "Product Description", "price": 100.00, "quantity": 10 }
  + Response: 201 Created with product details.
* **Get Product Details**
  + GET /api/products/{id}
  + Response: 200 OK with product details.
* **Update Product**
  + PUT /api/products/{id}
  + Request Body: { "name": "Updated Name", "description": "Updated Description", "price": 150.00, "quantity": 20 }
  + Response: 200 OK with updated product details.
* **Delete Product**
  + DELETE /api/products/{id}
  + Response: 204 No Content.

**Order Management**

* **Place Order**
  + POST /api/orders
  + Request Body: { "products": [{ "id": 1, "quantity": 2 }, { "id": 2, "quantity": 1 }] }
  + Headers: Authorization: Bearer {token}
  + Response: 201 Created with order details.

**Code Review Report**

The code review identified several issues with explanations and recommendations:

1. **Hardcoded Secrets:**
   * **Issue:** Hardcoded JWT secret keys.
   * **Recommendation:** Use environment variables to manage secrets securely.
2. **Missing Validation Rules:**
   * **Issue:** Insufficient validation for product creation and update.
   * **Recommendation:** Implement comprehensive validation rules in the controller.
3. **Inefficient Database Queries:**
   * **Issue:** N+1 query problem.
   * **Recommendation:** Use eager loading to optimize database queries.
4. **Inconsistent Error Handling:**
   * **Issue:** Inconsistent error handling for authentication failures.
   * **Recommendation:** Standardize error responses.
5. **Lack of Unit Tests:**
   * **Issue:** Missing unit tests for key functionalities.
   * **Recommendation:** Add unit tests to cover authentication and order management.

**API Documentation**

The API documentation is available in OpenAPI format. It describes all available endpoints, request/response formats, and authentication details. You can find it in the docs folder under openapi.yaml.