E-commerce Platform

# Introduction

The E-commerce Platform is a web-based solution that enables users to browse, buy, and sell products online. It allows sellers to manage their listings, and customers to search and purchase products seamlessly. The platform integrates various third-party services to handle payments, product reviews, and delivery tracking, ensuring a comprehensive online shopping experience.

# What?

The E-commerce Platform is a full-stack web application that facilitates online transactions between buyers and sellers. It provides users with features such as product search, payment integration, user authentication, shopping cart management, and order tracking.

# Why?

Online shopping has become a significant part of the global economy. The E-commerce Platform allows businesses to reach a larger customer base, and customers to enjoy the convenience of buying from anywhere. This project aims to develop an accessible, scalable, and user-friendly platform for both buyers and sellers.

# List of Supported Features

* User Authentication: Sign-up, login, and password recovery.
* Product Listings: Sellers can add and manage their products with descriptions, prices, and categories.
* Shopping Cart: Customers can add items to a cart and proceed to checkout.
* Payment Gateway Integration: Secure payment processing (e.g., Stripe/PayPal).
* Order Management: Track orders, view order history, and manage shipping information.
* Search Functionality: Users can search for products by category, price range, and keywords.
* Reviews and Ratings: Customers can leave reviews for products they’ve purchased.
* Admin Dashboard: Admins can manage users, products, and monitor transactions.

# List of Not Supported Features

* Real-time Chat Support: No direct chat feature for customer support.
* Wishlist: Customers cannot save items for future purchases.
* Multi-language Support: Only English is supported in the initial version.

# List of Future Planned Features

* Real-time Chat Support: Integrate a chat feature for customer support.
* Wishlist: Allow customers to save products for future purchases.
* AI-Powered Recommendations: Suggest products based on browsing and purchase history.
* Multi-language Support: Offer the platform in multiple languages.
* Loyalty Program: A points system to reward frequent customers.

# How

## High Level Diagram

* 1. **List of Components/Modules**
     + Frontend: Product listing, search, cart, and user management pages.
     + Backend: API endpoints for managing products, users, and orders.
     + Database: Store user profiles, product data, orders, and reviews.
     + Authentication Module: Manages login and registration (JWT or OAuth).
     + Payment Gateway Module: Handles transactions using third-party services (e.g., Stripe).

## Languages to be Used for Each Module

* + - Frontend: HTML, CSS, JavaScript, React.js.
    - Backend: Node.js (Express).
    - Database: MongoDB (NoSQL).
    - Payment Gateway: Integration via Stripe/PayPal SDKs.

## List of 3rd-party/Open Source Modules

* + - Frontend:
      * React.js (MIT License)
      * Axios for API calls (MIT License)
    - Backend:
      * Express.js (MIT License)
      * JWT for authentication (MIT License)
      * Mongoose for MongoDB (MIT License)
      * Stripe/PayPal SDK (BSD-3-Clause License)

## Table of Licenses

* + - React.js: MIT
    - Express.js: MIT
    - Mongoose: MIT
    - Stripe SDK: BSD-3-Clause
    - PayPal SDK: BSD-3-Clause

## List of any 3rd-party Services/APIs

* + - Stripe: Paid (for processing transactions).
    - PayPal: Paid (for processing transactions).
    - SendGrid: Free tier for sending confirmation emails.
    - Google Maps API: Free/Paid for location-based services.

## REST API Endpoints with Payloads

* + - User Authentication:
      * POST /api/auth/register

Payload: { "email": "string", "password": "string" }

* + - * POST /api/auth/login

Payload: { "email": "string", "password": "string" }

* + - Product Management:
      * GET /api/products
      * POST /api/products

Payload: { "name": "string", "price": "number", "description": "string", "category":"string"}

* + - Cart:
      * POST /api/cart

Payload: { "productId": "string", "quantity": "number" }

* + - * GET /api/cart
    - Payment:
      * POST /api/checkout

Payload: { "cartId": "string", "paymentMethod": "string" }

## Build Steps/Scripts

1. Frontend:
   * Navigate to the frontend directory:

```bash cd client

```

* + Install dependencies:

```bash npm install

```

* + Build the production version:

```bash

npm run build

```

1. Backend:
   * Navigate to the backend directory:

```bash cd server

```

* + Install dependencies:

```bash npm install

## Install Steps/Scripts

Frontend:

1. Clone the repository.
2. Run npm install to install all dependencies.
3. Use npm start to launch the development server.

Backend:

1. Clone the repository.
2. Run npm install to install backend dependencies.
3. Run npm run dev to launch the backend server.

# GitHub Information

Repository name: comp5130

Github URL: https://github.com/Dhanush-Amileneni/comp5130

# References

* [MERN Stack [Documentation](https://www.mongodb.com/mern-stack)](http://www.mongodb.com/mern-stack))
* [JWT Authentication](https://jwt.io/)
* [Stripe API Documentation](https://stripe.com/docs/api)
* [React Documentation](https://reactjs.org/docs/getting-started.html)
* [Express Documentation](<https://expressjs.com/>)

## Nothing changes made

All the frontend and backend work is done, and UI/UX improvements is done,

Currently working on the payments process.