Full Stack Development with MERN

Project Documentation format

1. Introduction

- **Project Title:** ShopEZ: One-Stop Shop for Online Purchases

- **Team Members:**
  - Frontend developer - Kotikalapudi Sri Devi Manasa
  - Backend Developer- Kamireddy Naga Satya Sri Vaishnavi
  - Database Developer - Rayapalli Durga Venkata Jaya Surya Sri
  - Deployment Engineer - kalla Yamuna Maha Lakshmi

2. Project Overview

- **Purpose:** The purpose of the ShopEZ project is to develop a user-friendly e-commerce platform that simplifies online shopping by enabling effortless product discovery, personalized recommendations, and secure checkout. The project also aims to help sellers efficiently manage orders and analyze business performance through a dedicated dashboard.

- **Features:**

- Effortless product browsing and filtering
- Personalized product recommendations
- Secure and seamless checkout process
- Instant order confirmation
- Seller dashboard for order management and analytics

3. Architecture

- **Frontend: React.js**

The frontend of ShopEZ is developed using **React.js** and follows a component-based architecture. It includes reusable UI components such as User Authentication, Product Listing, Cart, Profile, and Admin Dashboard. React Router is used for navigation, and API integration is handled using HTTP requests to communicate with the backend. This architecture ensures a responsive, interactive, and user-friendly interface.

- **Backend: Node.js & Express.js**

The backend is built using **Node.js** with the **Express.js** framework. It

follows a RESTful architecture and provides API endpoints for user authentication, product management, cart operations, and order processing. Middleware is used for request handling, authentication, and error management. The backend acts as a bridge between the frontend and the database.

- **Database: MongoDB**

ShopEZ uses **MongoDB**, a NoSQL database, to store application data. Collections include Users, Products, Cart, Orders, and Admin. Mongoose is used as an Object Data Modeling (ODM) tool to define schemas and perform CRUD operations. The database ensures efficient data storage, retrieval, and scalability.

4. Setup Instructions

- **Prerequisites:**

To develop and run the ShopEZ e-commerce application, the following software and tools are required:

- **Node.js & npm – To run server-side JavaScript and manage packages**
- **MongoDB – For storing users, products, cart, and orders data (local or MongoDB Atlas)**
- **Express.js – Backend framework for handling APIs and server logic**
- **React.js – Frontend library for building user interfaces**
- **HTML, CSS, JavaScript – For UI structure, styling, and interactivity**
- **Git – For version control and repository management**
- **Code Editor – Visual Studio Code / WebStorm / Sublime Text**

Installation:

1. **Clone the Project Repository**
   Open terminal or command prompt and run:

https://github.com/KamireddyVaishnavi/ShopEZ-One-Stop-Shop-for-Online-Purchases

2. **Navigate to Project Directory**

Code : cd ShopEZ--e-commerce-MERN

3. **Install Dependencies**
   Install all required frontend and backend packages:

Code : npm install

4. **Set Up Environment Variables**

Create a .env file and configure:

- **MongoDB connection URL**
- **Port number**
- **Any required secret keys**

5. **Start the Development Server**

Code : npm run dev

6. **Access the Application**
   Open a web browser and navigate to:

Code : http://localhost:3000

Successful Setup Confirmation

If the ShopEZ homepage loads successfully, the application has been installed and configured correctly. You can now proceed with testing, customization, and further development.

5. Folder Structure

· Client: React Frontend

The client folder contains the frontend of the ShopEZ application built using React.js. It follows a modular structure where the src directory holds reusable UI components, pages, routing logic, and API service calls. Components such as Login, Register, Home, Product Listing, Cart, Profile, and Admin Dashboard are organized for better maintainability and scalability. Styling and assets are also managed within this structure to ensure a responsive user interface.

· Server: Node.js Backend

The server folder contains the backend implementation developed using Node.js and

Express.js. It is organized into folders such as routes, controllers, models, and configuration files. Routes handle API requests, controllers contain business logic, models define MongoDB schemas using Mongoose, and configuration files manage database connections and environment variables. This separation ensures clean code structure and easy maintenance.

6. Running the Application

- **Frontend:** `npm start` in the client directory.

To run the ShopEZ application locally, follow the steps below:

Frontend (Client Side)

1. Open a terminal and navigate to the **client** directory:

cd client

2. Start the frontend server:

npm start

3. The frontend will run at:

http://localhost:3000

- **Backend:** `npm start` in the server directory.

1. Open a new terminal and navigate to the **server** directory:

cd server

2. Start the backend server:

npm start

The backend API will run on the configured server port (as defined in the .env file).

7. API Documentation

The backend of ShopEZ exposes RESTful APIs to handle user authentication, product management, cart operations, and order processing. These APIs enable smooth communication between the frontend and backend.

1. User APIs :

Register User

Endpoint: /api/users/register

Method: POST

Parameters: username, email, password

Response: User registered successfully

Login User

Endpoint: /api/users/login

Method: POST

Parameters: email, password

Response: Authentication success with user details

2. Product APIs :

Get All Products

Endpoint: /api/products

Method: GET

Parameters: None

Response: List of all products

Add New Product (Admin)

Endpoint: /api/products/add

Method: POST

Parameters: productName, price, category, description

Response: Product added successfully

3. Cart APIs :

Add Product to Cart

Endpoint: /api/cart/add

Method: POST

Parameters: userId, productId, quantity

Response: Product added to cart

View Cart

Endpoint: /api/cart/:userId

Method: GET

Parameters: userId

Response: Cart details of the user

4. Order APIs :

Place Order

Endpoint: /api/orders/place

Method: POST

Parameters: userId, address, paymentMethod

Response: Order placed successfully

View User Orders

Endpoint: /api/orders/:userId

Method: GET

Parameters: userId

Response: List of user orders

5. Admin APIs :

Admin Login

Endpoint: /api/admin/login

Method: POST

Parameters: email, password

Response: Admin authentication success

View All Orders

Endpoint: /api/admin/orders

Method: GET

Parameters: None

Response: All orders in the system

Example API Response :

{

"status": "success",

"message": "Order placed successfully",

"orderId": "ORD12345"

}

8. Authentication

Authentication and authorization in the ShopEZ project are handled using a secure, token-based mechanism.

User and admin authentication is implemented through **login and registration APIs** in the backend using **Node.js and Express.js**. User credentials are verified against the MongoDB database, and passwords are securely stored using encryption techniques.

Upon successful login, the server generates a **JSON Web Token (JWT)**, which is sent to the frontend. This token is stored on the client side and included in API requests to authenticate users. Protected routes are accessible only when a valid token is present.

Authorization is managed using middleware, which checks user roles (User or Admin) before granting access to restricted features such as the Admin Dashboard, product management, and order controls.

This approach ensures secure access, data protection, and role-based functionality within the application.

9. User Interface

This section showcases the main user interface screens of the ShopEZ application, highlighting key features available to users and administrators. Screenshots and GIFs are provided to demonstrate the usability and functionality of the system.

1. Landing Page / Home Page
Description:
The landing page displays the ShopEZ brand, navigation menu, and featured products. It provides easy access to product categories and search functionality.
Screenshot/GIF: Landing Page UI

2. User Authentication Page (Login & Register)
Description:
This screen allows users to register with valid details and log in securely using their credentials.
Screenshot/GIF: Login Page
Screenshot/GIF: Registration Page

3. Product Listing Page
Description:
Displays all available products with images, price, category, and filters. Users can browse, search, and filter products based on preferences.
Screenshot/GIF: Products Page

4. Product Details / Order Details Page
Description:
Shows detailed information about a selected product, including description, price, and purchase options such as “Add to Cart” or “Buy Now”.
Screenshot/GIF: Product Details Page

5. Cart Page
Description:
Displays products added to the cart along with quantity and total price. Users can proceed to checkout from this page.
Screenshot/GIF: Cart Page

6. Checkout Page
Description:
Allows users to enter shipping address and select a payment method. Ensures a secure and seamless checkout experience.
Screenshot/GIF: Checkout Page

7. Order Confirmation Page
Description:
After successful checkout, users receive order confirmation details including order ID and summary.
Screenshot/GIF: Order Confirmation Page

8. User Profile & Order History
Description:
Users can view their profile information and track previous orders placed through ShopEZ.
Screenshot/GIF: User Profile Page
Screenshot/GIF: Order History Page

9. Admin Dashboard
Description:
The admin dashboard allows administrators to manage products, view all orders, and monitor platform activity.
Screenshot/GIF: Admin Dashboard
Screenshot/GIF: Add New Product Page
Screenshot/GIF: All Orders page

10. Testing

· **Testing**

The ShopEZ project follows a systematic testing strategy to ensure functionality, security, and performance.
Manual testing is performed to validate user flows such as registration, login, product browsing, cart operations, checkout, and order confirmation.
API testing is carried out using tools like **Postman** to verify backend endpoints for users, products, cart, and orders.
Frontend testing ensures proper rendering of React components and smooth navigation across pages.
Database testing is done to confirm accurate data storage and retrieval from MongoDB.

Tools Used

- Manual Testing
- Postman (API Testing)
- Browser Developer Tools
- MongoDB Compass

11. Screenshots or Demo

https://drive.google.com/file/d/1fkZ2leM8Z8FWukBqgIJuaP5O_RQgT5OG/view?u

sp=drivesdk

12. Known Issues

• Document any known bugs or issues that users or developers should be aware of.

13. Future Enhancements

- Integration of AI-based product recommendations for better personalization
- Development of a mobile application for Android and iOS users
- Addition of multiple payment gateways including UPI and wallets
- Implementation of real-time order tracking and notifications
- Expansion to a multi-vendor marketplace with advanced seller features