Full Stack Development with MERN

Project Documentation format

1. Introduction

Project Title:

ShopEZ – One-Stop Online Shopping Platform

Team Members:

- Member 1 – Frontend Developer (React.js UI Development)
- Member 2 – Backend Developer (Node.js & Express APIs)
- Member 3 – Database Manager (MongoDB & Data Modeling)
- Member 4 – Full Stack & Integration (Payment & Deployment)

2. Project Overview

Purpose

ShopEZ is an e-commerce web application developed using the MERN stack. The goal of the project is to provide a secure, user-friendly, and scalable online shopping platform where customers can browse products, add them to cart, and complete purchases securely.

Features

- User Registration & Login (JWT Authentication)
- Social Login (Gmail)
- Product Listing & Categories
- Search & Filter Functionality
- Cart Management
- Secure Checkout & Payment Integration (Razorpay/Stripe)
- Order Management
- Admin Dashboard
- Email Notifications

3. Architecture

ShopEZ follows a **3-Tier Architecture**:

Frontend (React.js)

- Developed using React.js
- Uses Components for UI modularity
- Axios for API communication
- React Router for navigation
- Bootstrap for responsive design

Backend (Node.js & Express.js)

- RESTful API development
- JWT-based authentication
- Middleware for validation & security

- Payment gateway integration
- Role-Based Access Control (Admin/User)

Database (MongoDB)

- Cloud-based MongoDB Atlas
- Collections:
    - Users
    - Products
    - Categories
    - Cart
    - Orders
    - Admin
- Mongoose ODM for schema modeling

4. Setup Instructions

Prerequisites

- Node.js
- MongoDB (Local or MongoDB Atlas)
- npm
- Git

Installation Steps

1. Clone the repository:

git clone <repository-link>

2. Navigate to project folder:

cd shopez

3. Install frontend dependencies:

cd client
npm install

4. Install backend dependencies:

cd server
npm install

5. Create .env file in server folder:

PORT=5000
MONGO_URI=your_mongodb_connection_string
JWT_SECRET=your_secret_key
RAZORPAY_KEY=your_key

5. Folder Structure

Client (React Frontend)

client/
├── public/
├── src/
│ ├── components/
│ ├── pages/
│ ├── services/
│ ├── App.js
│ └── index.js

Server (Node.js Backend)

server/
├── config/
├── controllers/
├── models/
├── routes/
├── middleware/
├── server.js

6. Running the Application

Frontend

cd client

npm start

Backend

cd server

npm start

Frontend runs on: http://localhost:3000
Backend runs on: http://localhost:5000

7. API Documentation

Authentication APIs

Method	Endpoint	Description
POST	/api/register	Register new user
POST	/api/login	Login user

Example Request:

POST /api/login

```
{
  "email": "user@gmail.com",
  "password": "123456"
}
```

Example Response:

```
{
  "token": "jwt_token_here",
  "user": {
    "id": "123",
    "name": "John"
  }
}
```

\

Product APIs

Method	Endpoint	Description
GET	/api/products	Get all products
POST	/api/products	Add new product (Admin)
PUT	/api/products/:id	Update product
DELETE	/api/products/:id	Delete product

Order APIs

Method	Endpoint	Description
POST	/api/orders	Create new order
GET	/api/orders	Get user orders

8. Authentication

- Passwords are hashed using bcrypt.
- JWT tokens are generated upon login.
- Token stored in local storage.
- Middleware verifies token for protected routes.
- Role-Based Authorization for Admin routes.
- HTTPS used for secure communication.

9. User Interface

The application includes:

- Home Page (Product Listing)
- Product Details Page
- Cart Page
- Checkout Page
- Login & Registration Page
- Admin Dashboard

(Screenshots to be attached in final submission)

10. Testing

Testing Strategy:

- Manual testing for UI & API
- Postman used for API testing
- Unit testing for backend routes
- Validation testing for form inputs

11. Screenshots / Demo

- Home Page Screenshot
- Login Page Screenshot
- Cart Page Screenshot
- Admin Dashboard Screenshot
- Payment Success Page

(Demo Link: Add deployed URL if available)

12. Known Issues

- Social login limited to Gmail only
- No product recommendation system
- Basic search functionality
- Payment gateway test mode only

13. Future Enhancements

- AI-based product recommendations
- Wishlist feature
- Mobile App version
- Multi-language support
- Advanced analytics dashboard
- Real-time order tracking
- Integration with delivery APIs

Conclusion

ShopEZ demonstrates full-stack development using the MERN stack with secure authentication, structured architecture, and scalable design. The project effectively combines frontend, backend, and database technologies to deliver a complete e-commerce solution.