# Full Stack Development with MERN Project Documentation format

## 1. Introduction

- **Project Title:** SHOPEZ
- **Team Members:**

- **Team Leader:** M.Sujitha

- **Team Members:** Lavuluru Joshitha,M.Akash,M.Teja sree

## 2. Project Overview

- **Purpose:**
  SHOPEZ is an e-commerce platform designed to provide users with a seamless online shopping experience. The project aims to integrate a user-friendly interface with robust backend functionalities to manage products, user accounts, and transactions efficiently.
- **Features:**
- User registration and authentication
- Product browsing and search functionality
- Shopping cart and checkout process
- Order history and tracking

## 3. Architecture

- **Frontend:**
  Developed using **React.js**, the frontend offers a dynamic and responsive user interface. Components are structured to facilitate reusability and maintainability.
- **Backend:**
  Built with **Node.js** and **Express.js**, the backend handles API requests, business logic, and server-side operations. It ensures secure and efficient data processing.
- **Database:**
  Utilizes **MongoDB** for data storage, providing a flexible schema to accommodate various data types. Mongoose is used as an ODM to manage data relationships and validations.
-

## 4. Setup Instructions

- **Prerequisites:**

  - Node.js and npm installed

    - MongoDB installed or access to MongoDB Atlas
    - Git installed
- ●
- **Installation:** Step-by-step guide to clone, install dependencies, and set up the environment variables.

## 5. Folder Structure

- **Client:** Describe the structure of the React frontend.
- **Server:** Explain the organization of the Node.js backend.

## 6. Running the Application

**Endpoints:**

- **User Authentication:**
    - POST /api/users/register - Register a new user
    - POST /api/users/login - Authenticate user and return token
- **Products:**
    - GET /api/products - Retrieve all products
    - GET /api/products/:id - Retrieve product by ID
    - POST /api/products - Add a new product (Admin only)
    - PUT /api/products/:id - Update product details (Admin only)
    - DELETE /api/products/:id - Delete a product (Admin only)
- **Orders:**

- ● `POST /api/orders` - Create a new order
- ● `GET /api/orders/:id` - Retrieve order by ID
- ● `GET /api/orders/user/:userId` - Retrieve orders for a specific user.

## 7. API Documentation

- Document all endpoints exposed by the backend.
- Include request methods, parameters, and example responses.

## 8. Authentication

- **Registration and Login:**
- Users register with a username, email, and password.
- Passwords are hashed using bcrypt before storage.

## 9. User Interface

- **Home Page:** Displays featured products and categories.
- **Product Page:** Detailed view of a selected product.
- **Cart:** Shows selected items with quantity and total price.
- **Checkout:** Form for shipping details and payment method.
- **Admin Panel:** Interface for managing products and orders.

## 10. Testing

- T**esting Strategy:**
  - Unit tests for individual components and functions.
  - Integration tests for API endpoints.
- **Tools Used:**
  - Jest for JavaScript testing.
  - Supertest for HTTP assertions.

## 11. Screenshots or Demo

- Provide screenshots or a link to a demo to showcase the application.

## 12. Known Issues

- **Responsive Design:** Some pages may not be fully responsive on smaller screens.
  - **Error Handling:** Need to implement comprehensive error messages for API failures.

## 13. Future Enhancements

- **Payment Gateway Integration:** Incorporate services like Stripe or PayPal for real payments.
- **Product Reviews:** Allow users to leave reviews and ratings for products.
- **Wishlist Feature:** Enable users to save products for future purchases.

- **Enhanced Search:** Implement advanced search filters and sorting options.