# Full Stack Development with MERN

# Project Documentation format

## 1. Introduction

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

| Name | Role |
|---|---|
| **Lakshmi Sathvika Devanaboina** | **Full Stack Developer** |
| **Member 2** | **Frontend Developer** |
| **Member 3** | **Backend Developer** |
| **Member 4** | **Database & Testing** |

## 2. Project Overview

ShopEZ is a MERN-based e-commerce web application designed to simplify online shopping. The goal is to provide a user-friendly, secure, and scalable platform that allows customers to browse products, manage carts, complete secure payments, and track orders efficiently.

**Key Features**

- User Registration & Login (Email + Social Login)
- Secure Authentication
- Product Browsing & Search
- Add to Cart Functionality
- Smooth Checkout Process
- Order Tracking
- Admin Product Management
- Responsive Design (Mobile + Web)

## 3. Architecture

**Frontend (React)**

- Built using **React.js**
- Component-based architecture
- React Router for navigation
- Axios for API communication
- State management using React Hooks
- Responsive UI using CSS / Bootstrap / Tailwind

Flow:
User → React UI → API Calls → Backend

**Backend (Node.js + Express.js)**
- RESTful API architecture
- Express.js server
- Middleware for authentication
- JWT for token-based authentication
- Routes for:
  - User
  - Product
  - Cart
  - Order

Flow:
Frontend → Express Routes → Controllers → MongoDB

**Database (MongoDB)**

Collections:
- Users
- Products
- Orders
- Cart

Example Schema:

**User Schema**
- name
- email
- password (hashed)
- role

**Product Schema**
- title
- description
- price
- category
- stock

**Order Schema**
- userId
- products
- totalAmount
- orderStatus

## 4. Setup Instructions

**Prerequisites**

- Node.js

- MongoDB
- npm
- Git

**Installation**

**Step 1: Clone Repository**

Clone project from GitHub.

**Step 2: Install Dependencies**

Inside client folder:
Install frontend dependencies.

Inside server folder:
Install backend dependencies.

**Step 3: Configure Environment Variables**

Create .env file in server folder:

- MONGO_URI
- JWT_SECRET
- PORT

## 5. Folder Structure

**Prerequisites**

- Node.js
- MongoDB
- npm
- Git

**Installation**

**Step 1: Clone Repository**

Clone project from GitHub.

**Step 2: Install Dependencies**

Inside client folder:
Install frontend dependencies.
Inside server folder:
Install backend dependencies.

**Step 3: Configure Environment Variables**

Create .env file in server folder:

- MONGO_URI
- JWT_SECRET
- PORT

## 6. Running the Application

**Frontend**

**Run inside client directory.**

**Backend**

**Run inside server directory.**

**Application runs at:**
**Frontend → http://localhost:3000**
**Backend → http://localhost:5000**

## 7. API Documentation

| Method | Endpoint | Description |
|---|---|---|
| POST | /api/register | Register user |
| POST | /api/login | Login user |
| GET | /api/profile | Get user profile |

**Product Routes**

| Method | Endpoint | Description |
|---|---|---|
| GET | /api/products | Get all products |
| GET | /api/products/:id | Get single product |
| POST | /api/products | Add product (Admin) |

**Order Routes**

| Method | Endpoint | Description |
|---|---|---|
| POST | /api/orders | Create order |
| GET | /api/orders/:id | Get order details |

### 8. Authentication

- JWT-based authentication
- Password hashing using bcrypt
- Role-based access (Admin / Customer)
- Protected routes using middleware

Authentication Flow:

User Login → Server generates JWT → Token stored → Token sent in header → Protected API accessed

## 9. User Interface

### Screens to include:

- Homepage
- Login Page
- Registration Page
- Product Listing Page
- Cart Page
- Checkout Page
- Admin Dashboard

## 10. Testing

- Manual testing of user flows
- API testing using Postman
- Validation testing for forms

- Functional testing for cart and checkout

## 11. Screenshots or Demo

**https://drive.google.com/file/d/1X1wk1Fyk1rtGAHJP396i22lw_DUeqbdO/view?usp=sharing**

## 12. Known Issues

- Social login may require API configuration
- Payment gateway integration may require production keys
- Performance may vary with large product data

## 13. Future Enhancements

- AI-based product recommendations
- Wishlist feature
- Review & rating system
- Real-time notifications
- Advanced analytics dashboard
- Multi-vendor support