

# Full Stack Development with MERN

## Project Documentation

### 1. Introduction

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

**Team Members:**

**Team ID :** LTVIP2026TMIDS79643

**Team Size :** 4

**Team Leader :** Jyothi Nagireddy Palle

**Team member :** Palle Palakolanu Divya

**Team member :** Prathyusha Pradhamakavi

**Team member :** Shaik Mohammad Basha

### 1. Project Overview Purpose:

ShopEZ is a full-stack MERN e-commerce web application designed to provide users with a smooth, fast, and personalized online shopping experience. It allows users to browse products, add items to a cart, place orders, and track purchases. Sellers can manage products and orders through an admin interface.

### Features:

User registration and login system Product listing

with filtering and search Shopping cart functionality

Secure checkout process Order

confirmation system

Admin panel for product and order management Responsive

UI design

MongoDB database integration

## 1. Architecture

### Frontend (React Architecture):

The frontend is built using React.js with a component-based structure. Pages are divided into reusable UI components stored in the components folder. Application state is managed through Context API stored in the context folder. Styling is handled using CSS files inside the styles directory. Routing between pages is managed through React Router.

### Backend (Node.js + Express.js Architecture):

The backend is developed using Node.js and Express.js. The main server entry file is index.js, which handles routing, middleware, and database connection. API endpoints process requests related to users, products, carts, and orders. Environment variables are stored in .env.

### Database (MongoDB):

MongoDB stores application data in collections. Schema definitions are handled in Schema.js, which defines models for users, products, cart items, and orders. CRUD operations are performed through MongoDB queries.

## 1. Setup Instructions

### Prerequisites:

Node.js installed

npm installed

MongoDB installed or cloud MongoDB URI

Git installed

### Installation Steps:

1. Clone project

```
git clone <repository-url>
```

1. Open project folder

```
cd SHOPEZ
```

2. Install frontend dependencies

```
cd client
```

```
npm install
```

#### 4. Install backend dependencies

```
cd ../server
```

```
npm install
```

#### 5. Create .env file inside server folder

```
PORT=5000
```

```
MONGO_URI=your_mongodb_connection_string
```

```
JWT_SECRET=your_secret_key
```

### 5. Folder Structure

Client Folder Structure (React Frontend)

```
client/
```

```
├ node_modules/
```

```
├ public/
```

```
├ src/
```

```
│   ├── components/
```

```
│   ├── context/
```

```
│   ├── images/
```

```
│   ├── pages/
```

```
│   ├── styles/
```

```
│   ├── App.css
```

```
│   ├── App.js
```

```
│   ├── App.test.js
```

```
│   ├── index.css
```

```
│   ├── index.js
```

```
│   ├── logo.svg
```

```
│   └── reportWebVitals.js
```

| └─ setupTests.js

└─ package.json

└─ README.md

Explanation:

components/ reusable UI components

context/ global state management

images/ static assets

pages/ page-level components

styles/ CSS files

App.js root component

index.js entry point

Server Folder Structure (Node Backend)

server/

└─ node\_modules/

└─ .env

└─ index.js

└─ Schema.js

└─ package.json

└─ README.md

Explanation:

index.js main server file

Schema.js database models

.env environment configuration

package.json dependencies

## 6. Running the Application

Start Backend

cd server

npm start

Start Frontend

cd client

npm start

Access Application

<http://localhost:3000>

7. API Documentation

Authentication APIs

POST /api/register

POST /api/login

GET /api/profile

Product APIs

GET /api/products

GET /api/products/:id

POST /api/products

PUT /api/products/:id

DELETE /api/products/:id

Cart APIs

POST /api/cart/add

GET /api/cart/:userId

DELETE /api/cart/remove/:id

Order APIs

POST /api/orders

GET /api/orders/:userId

GET /api/orders/admin

Sample Response

```
{  
  "status": "success",  
  "message": "Order placed successfully"  
}
```

## 1. Authentication

Authentication is handled using JSON Web Tokens (JWT).

Process Flow:

1. User logs in with email and password
2. Server validates credentials
3. JWT token generated
4. Token sent to client
5. Token attached to protected API requests

Authorization:

Middleware checks token validity

Admin routes require admin role

Security Measures:

Password hashing

Token expiration

Protected routes

## 9. User Interface

UI Screens Included:

Login Page

Registration Page

Home Page

Product Listing Page

Product Details Page

Cart Page

Checkout Page

Order Confirmation Page

Admin Dashboard

## **1. Testing**

Testing methods used:

Manual UI testing

API testing using Postman

Form validation testing

Authentication testing

Testing ensures:

Correct API responses

Secure login system

Proper cart functionality

Responsive layout

## **1. Screenshots**

Homepage

Product page

Cart page

Checkout page

Admin dashboard

### **1. Known Issues**

Payment gateway not implemented

Basic recommendation logic only

No real-time notifications

Limited analytics dashboard

### **1. Future Enhancements**

Online payment integration

AI recommendation system

Wishlist feature

Email notifications

Mobile application version

Multi-vendor support