FULL STACK DEVELOPMENT WITH MERN

Project Document

1. Introduction

Project Title: Shopez: E-Commerce Application

Team Member:

1. Shardul Rana – Backend Developer
2. Shreya Kumari – Frontend Developer
3. Nishan Chakraborty – Database Administrator
4. Yashasvi Singh – Connectivity and Testing lead

2. Project Overview

Purpose:

The purpose of this project is to design and develop a modern, user-friendly E-commerce platform using the MERN stack (MongoDB, Express.js, React.js, Node.js) that allows users to browse products, register securely, and make purchases efficiently. The platform aims to provide seamless user registration, robust authentication, and efficient order management, with a responsive interface and role-based access for both users and administrators.

This project serves as a learning and implementation opportunity for full-stack development, with a focus on real-world application structure, team collaboration, REST API integration, and containerized development using Docker. It is also intended to simulate industry-standard development workflows and practices using tools like GitHub, VS Code, and Postman.

Features:

1. User-Facing Features

- **User Registration:** Secure registration via email and password. Input validation and feedback.
- **User Login:** Login using email and password. Protected routes with authentication tokens.
- **User Confirmation:** Users are confirmed directly via database (manual or internal process).
- **Product Listing:** Users can view available products with images, names, and prices. Products are fetched dynamically from the database.

- **Product Filtering & Search:** Users can filter products by category or search by name.
- **Cart System:** Users can add products to cart and adjust quantities. Cart stored in state (or database for logged-in users).
- **Checkout:** Users can view total price and place orders. Order data is stored and managed via backend

2. Admin Features

- **Admin Dashboard:** Role-based access to admin panel with login.
- **User Management:** Admin can view and manage registered users.
- **Product Management:** Admin can upload, edit, or delete products.
- **Order Management:** Admin can track and manage customer orders.

3. Technical & Functional Features

- **Full MERN Stack Integration:** React frontend, Express/Node backend, MongoDB database.
- **REST API-Based Architecture:** Clear separation between frontend and backend. API endpoints for all core operations.
- **Authentication & Authorization:** JWT-based token authentication for secure access. Admin and user roles handled separately.
- **Responsive UI:** Built using Bootstrap CSS for fast and modern design. Mobile, tablet, and desktop responsive.
- **Version Control & Team Collaboration:** Code managed using GitHub and GitHub Desktop for collaboration. Branch-based development and commits.

3. Architecture

Frontend Architecture – React.js

- **Component-Based Architecture:** The frontend is built using modular React components for reusability and separation of concerns.
  Examples: LoginForm, RegisterForm, ProductCard, CartPage, AdminDashboard.
  React Router is used to manage navigation between pages.
- **State Management:** Local State using useState, useEffect.
- **Bootstrap CSS:** Provides a utility-first approach for styling, enabling responsive and modern UI designs.
- **Key Pages:** Home (Product list), Product Details, Register/Login, Cart, Checkout, Admin Panel.

Backend Architecture – Node.js & Express.js

Modular Folder Organization:

- routes/: API route handlers for users, products, orders.
- controllers/: Logic for each route (e.g., user registration, product CRUD).
- models/: MongoDB Mongoose schemas.
- middlewares/: Auth checks (JWT verification, role-based access).
- config/: DB connection, environment setup.

User Authentication: Email-password-based signup & login. JWT token issued on login for session handling.

Role-Based Access: Admin vs User privileges for product and order management.

RESTful APIs:

- POST /api/register
- POST /api/login
- GET /api/products
- POST /api/orders
- GET /api/users (admin only)
- POST /api/products (admin only)

Database Schema & MongoDB Interactions

- User Schema
- Product Schema
- Order Schema
- Categories Schema

- Create: New users, orders, products.
- Read: Listing users, fetching products using.
- Update: Admins updating product stock or order status.
- Delete: Admin removing products or orders.

4. Setup Instructions

Prerequisites

Tool	Version	Purpose
Node.js	v18.x or above	JavaScript runtime environment
npm	v9.x or above	Node package manager
MongoDB	v6.x or Atlas	NoSQL database
Git	Latest	To clone repositories
VS Code	Latest	Code editor
Postman	Latest	For testing APIs

Installation

1. Clone the repository

```
git clone https://github.com/your-username/your-project.git
cd your-project
```

2. Setup Frontend

```
cd client
npm install
```

3. Setup Backend

```
cd ../server
npm install
```

4. Setup Environment Variables

Create a .env file inside the server folder and add:

```
PORT=5000

MONGO_URI=your_mongodb_connection_string

JWT_SECRET=your_jwt_secret
```

5. Run the Project

Start Backend

```
cd server

npm run server
```

Start Frontend (in another terminal)

```
cd client

npm start
```

5. Folder Structure

Client –

public/ – Static Public Assets

- Contains images, favicon, index.html, manifest, and other static files.

src/ – Application Source Code

- App.js / App.css – Main React component and styling.
- index.js / index.css – Entry point for the React app.
- .env – Environment variables.
- reportWebVitals.js / setupTests.js – Performance measuring and testing setup.

components/ – Reusable UI Components

- Form/ – Forms like CategoryForm, SearchInput
- Layout/ – Header, Footer, Layout wrappers
- Routes/ – Protected route components like AdminRoute, Private
- Prices.js, Spinner.js – Utility components

context/ – Global State Management

- auth.js, cart.js, search.js – Contexts for authentication, cart, and search.

hooks/ – Custom React Hooks

- useCategory.js – Custom hook to fetch or manage category data.

pages/ – Application Pages (Views)

- Top-level pages like HomePage, Contact, ProductDetails, etc.
- Auth/ – Auth-related pages like Login, Register, ForgotPassword
- Admin/ – Admin dashboard, product and category management, users, etc.
- user/ – User dashboard, orders, and profile pages

styles/ – CSS Files for Pages

- Modular CSS files like AuthStyles.css, Homepage.css, etc.

Project Config and Metadata Files

- .gitignore, package.json, package-lock.json, README.md

Server –

Root-Level Files:

- .env – Environment variables (DB URI, JWT secret, etc.)
- server.js – Entry point; sets up Express server, middleware, and routes.
- package.json / package-lock.json – Project metadata and dependencies.

config/

- db.js – MongoDB connection logic.

controllers/

- Handles business logic for different routes:
- authController.js – Login, register, password management.
- categoryController.js – Create, update, delete, list product categories.
- productController.js – Manage product-related operations.

helpers/

- authHelper.js – Contains helper functions like token generation, hashing, etc.

models/

Mongoose schemas for MongoDB collections:

- userModel.js – User schema.
- productModel.js – Product schema.
- categoryModel.js – Category schema.
- orderModel.js – Order schema.

routes/

Defines API endpoints and connects them to controllers:

- authRoute.js – Routes for login, register, etc.
- categoryRoutes.js – Routes to handle categories.
- productRoutes.js – Routes to manage products.

6. Running the Application

Frontend: npm start in the client directory

Backend: npm run server in the server directory

7. API Documentation

1. Register User

Endpoint: POST /api/v1/auth/register

2. Login User

Endpoint: POST /api/v1/auth/login

3. Forgot Password

Endpoint: POST /api/v1/auth/forgot-password

4. Reset Password

Endpoint: POST /api/v1/auth/reset-password/:token

5. Create Product

Endpoint: POST /api/v1/product/create-product

6. Get All Products

Endpoint: GET /api/v1/product/get-product

7. Get Single Product

Endpoint: GET /api/v1/product/get-product/:slug

8. pdate Product

Endpoint: PUT /api/v1/product/update-product/:pid

9. Delete Product

Endpoint: DELETE /api/v1/product/delete-product/:pid

10. Create Category

Endpoint: POST /api/v1/category/create-category

11. Get All Categories

Endpoint: GET /api/v1/category/get-category

12. Update Category

Endpoint: PUT /api/v1/category/update-category/:id

13. Delete Category

Endpoint: DELETE /api/v1/category/delete-category/:id

8. Authentication

1. User Authentication (Login/Register)

When a user registers or logs in, the backend:

- Validates the user’s credentials.
- Creates a JWT token using the user's _id and role.
- Sends the token back in the response.
- This token is stored client-side (e.g., in localStorage) and sent in Authorization headers for protected routes.

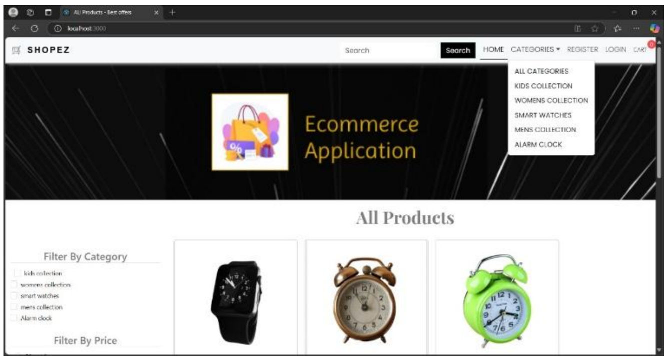
2. Authorization Middleware

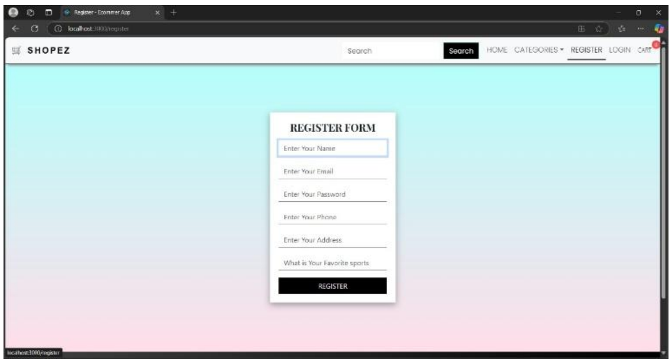
- Defined in authMiddleware.js.

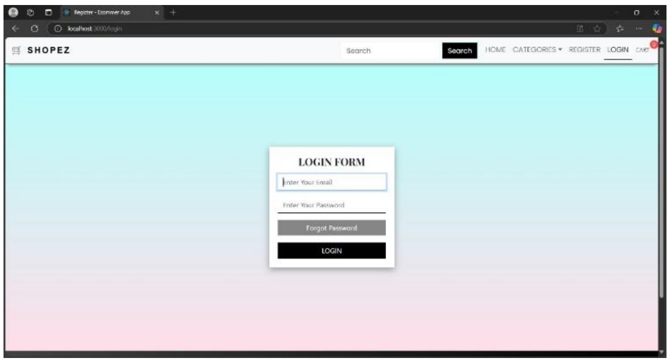
requireSignIn

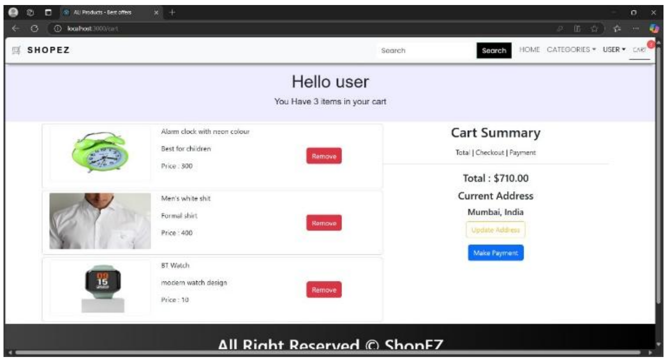
- Verifies that the JWT token is valid.
- Used to protect private routes.
- Checks if the logged-in user has admin privileges.

9. User Interface









Hello user

You Have 3 items in your cart

Alarm clock with neon colour
Best for children
Price : 300

Men's white shirt
Formal shirt
Price : 400

BT Watch
modern watch design
Price : 10

Cart Summary

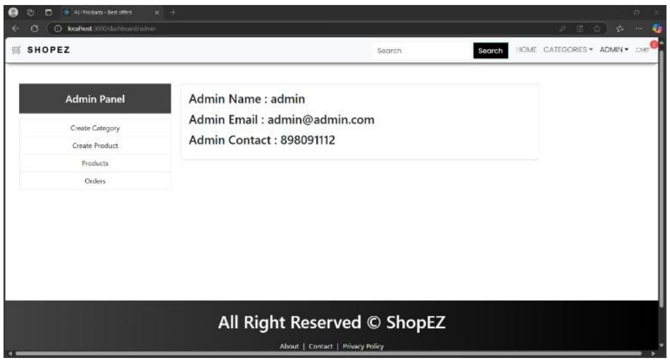
Total | Checkout | Payment

Total : $710.00

Current Address

Mumbai, India

All Right Reserved © ShopEZ

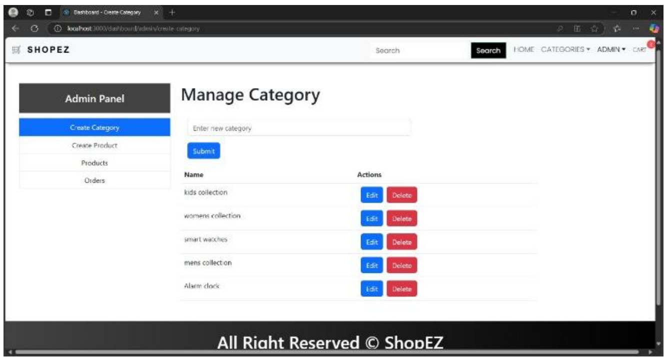


Admin Name : admin

Admin Email : admin@admin.com

Admin Contact : 898091112

All Right Reserved © ShopEZ



Manage Category

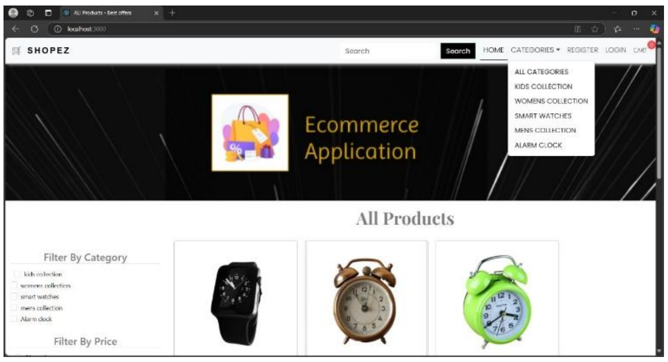
Name	Actions
kids collection	Edit Delete
womens collection	Edit Delete
smart watches	Edit Delete
mens collection	Edit Delete
Alarm clock	Edit Delete

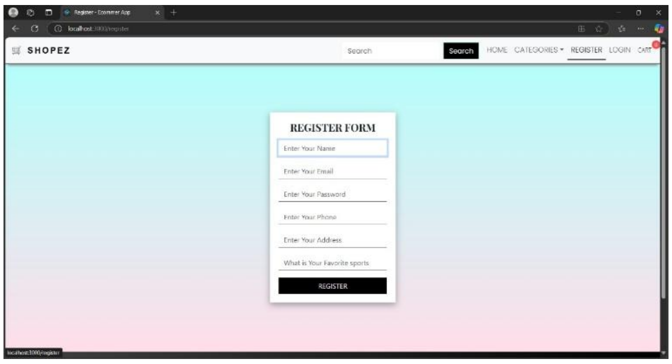
All Right Reserved © ShopEZ

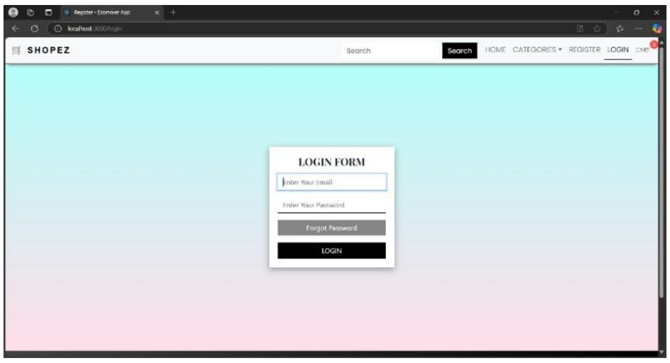
10. Testing

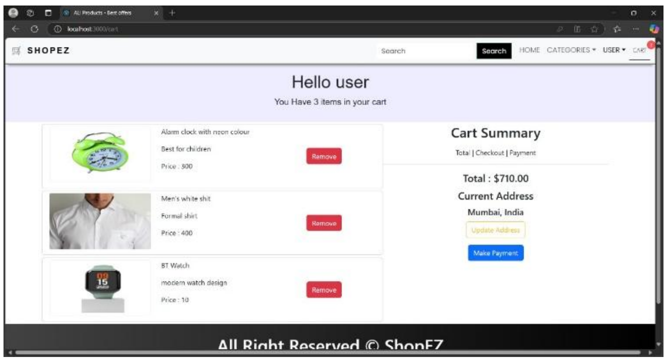
API Type	API Endpoint	Status Code	Response Time (ms)
Category	/api/v1/category/get-category	304	51.753
Product	/api/v1/product/product-list/1	304	46.899
Product Img	/api/v1/product/product-photo/67f93a1537dabe8278f60a34	304	45.206
Payment	/api/v1/product/braintree/token	200	1355.800
Auth	/api/v1/auth/user-auth	304	25.447

11. Screenshots








Hello user

You Have 3 items in your cart

Alarm clock with neon colour
Best for children
Price : 300

Men's white shit
Formal shirt
Price : 400

BT Watch
modern watch design
Price : 10

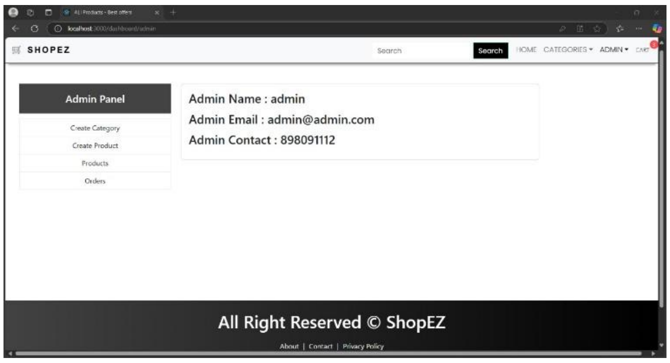
Cart Summary

Total | Checkout | Payment

Total : $710.00

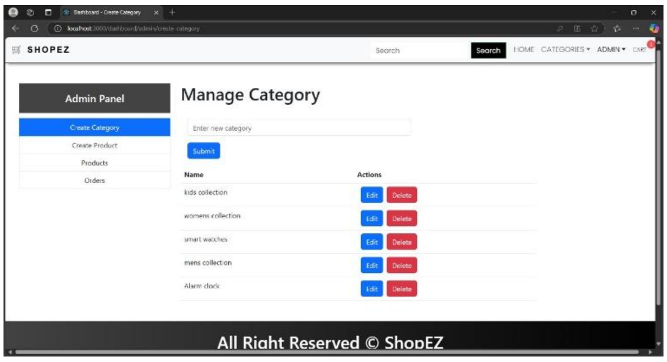
Current Address

Mumbai, India


Admin Name : admin

Admin Email : admin@admin.com

Admin Contact : 898091112


Manage Category

Name	Actions
kids collection	Edit Delete
womens collection	Edit Delete
smart watches	Edit Delete
mens collection	Edit Delete
Alarm clock	Edit Delete

12. Known Issues

1. **No Password Hashing During Registration:** Passwords are stored in plain text. This is a serious security flaw.

2. **JWT Secret Key Hardcoding (or missing):** If the JWT secret is hardcoded or not retrieved securely via .env, it's a security vulnerability.

3. **No Input Validation or Sanitization:** No validation using libraries like express-validator or Joi.

4. **Lack of Error Handling Middleware:** Errors are handled manually inside controllers instead of a global error handler.

5. **No Role-Based Access Control (RBAC):** Currently, only general user authentication exists.

6. **Token Expiry Not Handled Properly:** Token creation works, but no token refresh strategy is in place.

7. **Sensitive Data Exposure in Responses:** Entire user object may be returned, including password or metadata.

8. **No Rate Limiting or Brute-Force Protection:** Repeated login attempts are not limited.

13. Future Enhancements

User Experience Enhancements

- Product Filtering and Sorting: Allow users to filter products by category, price, and popularity.
- Search Autocomplete: Live search suggestions while typing product names.
- Wishlist Functionality: Let users save products to purchase later.
- Product Reviews & Ratings: Enable users to leave feedback and view average ratings.
- Dark Mode Toggle: For a modern and accessible user interface.

Payment & Checkout Improvements

- Multiple Payment Gateways: Add support for PayPal, Stripe, or UPI along with existing Braintree.
- Coupon & Discount System: Admin-defined discounts or personalized codes for users.
- Order Tracking System: Let users track their order status in real-time (e.g., shipped, delivered).
- EMI/Instalment Option: Display products available for EMI and let users choose payment plans.

Admin Panel Enhancements

- Data Analytics Dashboard: Show sales trends, revenue graphs, user behaviour, etc.
- Bulk Product Upload via CSV/Excel: Save time by importing multiple products at once.
- User Activity Logs: Track actions like login attempts, order edits, etc.
- Stock Alerts & Auto Restocking Suggestions: Notify admin when a product is running low.

Security & Access Control

- Email Verification: Add confirmation emails for newly registered users.
- Two-Factor Authentication (2FA): Add extra login security for admins or users.
- Forgot Password Email Reset: Enable password recovery via email with secure links.

Architecture & Codebase Improvements

- Implement Redux for Global State: Especially useful for managing cart, auth, and product filters.
- Introduce Unit & Integration Testing: Using Jest, Supertest, and React Testing Library.
- CI/CD Pipeline Integration: Automate deployments via GitHub Actions or Vercel.
- Deploy to Production Platforms: Use platforms like Render, Railway, or DigitalOcean.

Other Enhancements

- Multi-language Support: Offer the platform in different languages.
- Mobile App Integration (React Native): Create a cross-platform app that connects to the same backend.
- Notification System: Email/SMS/push notifications for order status and updates.