Full Stack Development with MERN

ShopEZ — One Stop Shop for Online Purchases

Team ID: LTVIP2026TMIDS83223

Team Size: 4

Team Members and Roles

Name	Role
Jyotula Venkata Durga Ramkumar	Team Leader / Backend Architect
Gopi Narendra Kondapalli	Member / Frontend Developer
Padmaja Katuri	Member / Full Stack Integration
Veera Venkata Satyanarayana Pragada	Member / Database & Testing

1. Introduction

Project Title: ShopEZ

ShopEZ is a comprehensive e-commerce platform developed to streamline the online shopping experience. Built using the modern MERN stack, the application ensures high performance, scalability, and a seamless user interface for both consumers and store administrators.

2. Project Overview

Purpose

The purpose of ShopEZ is to provide a user-friendly, one-stop solution for diverse online purchases. The project aims to solve the complexity of navigating multiple platforms by offering a consolidated store where users can find products across categories like electronics, apparel, and more. For business owners, it provides a powerful administrative suite to manage digital inventory with zero technical overhead.

Features

- Product Catalog: Dynamic browsing of products with high-quality imagery.
- Smart Filtering: Filter items by category (Electronics, Men, Women) and sort by price/discount.
- Shopping Cart: Real-time cart updates and management.
- Responsive Design: Optimized for seamless usage across Mobiles, Tablets, and Desktops.
- Admin Suite: Full CRUD (Create, Read, Update, Delete) dashboard for products and categories.
- Secure User Flow: Registration and Login functionality for personalized shopping experiences.

3. Architecture

Frontend (React.js)

The frontend is architected as a Single Page Application (SPA) using React. It leverages a modular component design. Key components like the FilterBar, ProductCard, and Navbar are reused throughout the app to maintain UI consistency. Axios is used for handling asynchronous API communications with the Node.js backend.

Backend (Node.js & Express.js)

The backend follows a RESTful API pattern. Built on Node.js, it uses Express to handle routing and middleware functionality. The server acts as a secure layer that validates user requests and coordinates data transactions between the client and the MongoDB database.

Database (MongoDB)

A NoSQL approach is used to store product and category data. This allows for flexible attributes across different product types. Mongoose is utilized as the ODM (Object Data Modeling) tool to enforce data structure and facilitate easy interaction with the database.

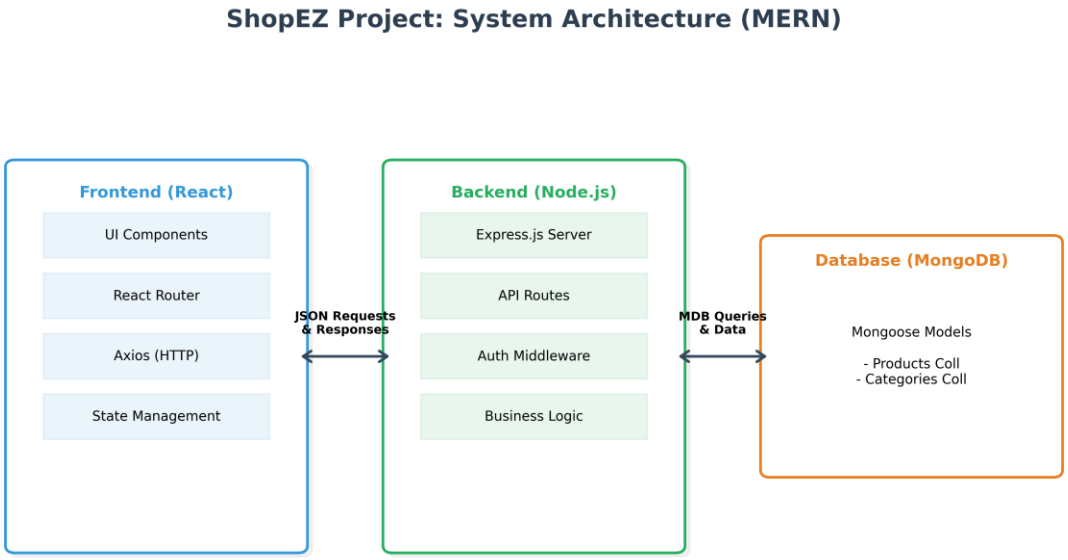

fig: ShopEZ architecture

4. Setup Instructions

Prerequisites

- Ensure the following are installed on your system:
- Node.js (v14 or higher)
- npm (Node Package Manager)
- MongoDB (Local Community Edition or Atlas URI)
- Git for version control

Installation Steps

1. Clone the repository:

   git clone https://github.com/padmajakaturi/shopez-one-stop-shop-for-online-purchases.git

2. Setup Backend:

   cd server
   npm install

   Create a .env file and add your MONGO_URI and PORT (6001).

3. Setup Frontend:

   cd ../client
   npm install

5. Folder Structure

Client (React)

```
client/
├── public/            # Static files
├── src/
│   ├── components/    # UI Components (Navbar, Cards)
│   ├── pages/         # Specific Views (Home, Shop, Admin)
│   ├── App.js         # Routing and Main Logic
│   └── index.js       # Entry point
└── package.json       # Dependencies
```

Server (Node/Express)

```
server/
├── .env               # Config environment variables
├── Schema.js          # Mongoose Data Models
├── index.js           # Server entry and API routes
└── package.json       # Dependencies
```

6. Running the Application

To start the application locally, run commands in separate terminals:

Frontend

Command: npm start (inside /client directory)

URL: http://localhost:3000

Backend

Command: npm start (inside /server directory)

URL: http://localhost:6001

7. API Documentation

Method	Endpoint	Description
GET	/fetch-categories	Retrieve all categories for filters
POST	/add-category	Admin: Create a new category
DELETE	/delete-category/:cat	Admin: Path-based deletion of category
GET	/fetch-products	Retrieve list of all shop items
POST	/add-product	Admin: Add a new inventory item
PUT	/update-product/:id	Admin: Update details for an existing item
DELETE	/delete-product/:id	Admin: Remove product by DB ID

8. Authentication

8.1 Authentication Mechanism ShopEZ employs a secure, credential-based authentication system to verify user identity.

- **Password Hashing:** To ensure data privacy, user passwords are never stored in plain text. The system utilizes **Bcrypt.js** for salting and hashing passwords before they are committed to the MongoDB database.
- **JWT Implementation:** Upon successful login, the server generates a **JSON Web Token (JWT)**. This token act as a "digital passport," allowing the frontend to make authenticated requests to protected API endpoints without needing to resend credentials.

8.2 Role-Based Access Control (RBAC) The application distinguishes between different user levels to maintain system integrity:

- **User Role:** Can browse products, manage their personal cart, and view their specific order history.
- **Admin Role:** Granted elevated privileges to access the Product Management dashboard, modify inventory, and manage global categories.

8.3 Session Management

- **Persistence:** The user's authentication state is managed using the **React Context API** or **Local Storage**. This ensures that the user remains logged in even after refreshing the browser page.
- **Logout Logic:** On logout, the client-side token is destroyed, and the global user state is cleared, preventing any further access to private routes.

8.4 Backend Security Middleware The Node.js server acts as the final gatekeeper. Every request to an admin-only endpoint (like POST /add-product) passes through a custom middleware function. This middleware:

1. Extracts the token from the request header.
2. Verifies the token's validity.
3. Checks the is Admin boolean within the decoded payload.
4. If unauthorized, it returns a 403 Forbidden or 401 Unauthorized status code.

9. User Interface

The visual identity of ShopEZ is designed to be clean, modern, and high-contrast.

Register Page:

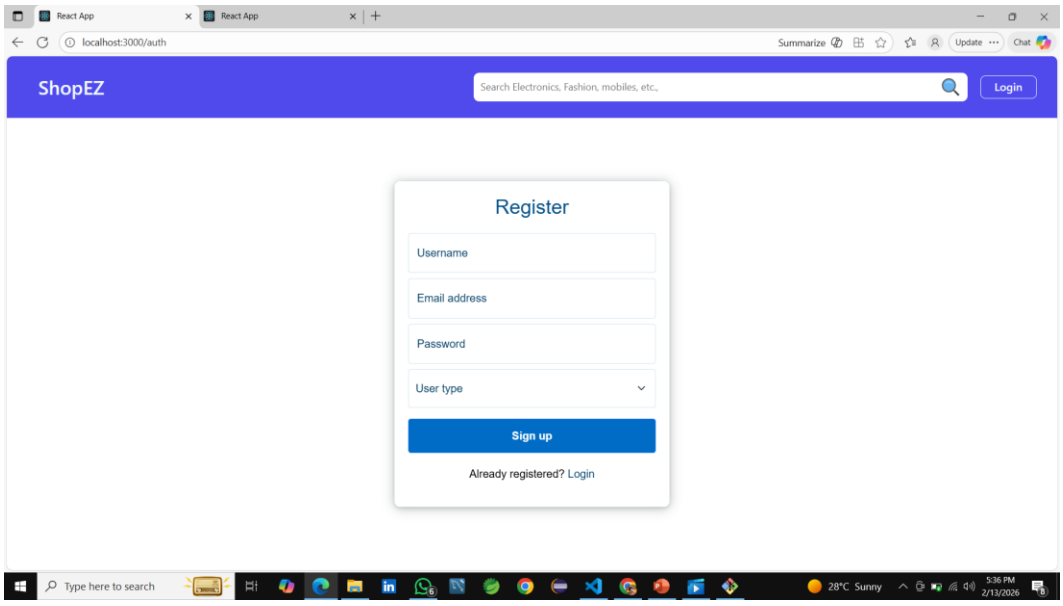

fig: Register user for Credentials

Login Page:

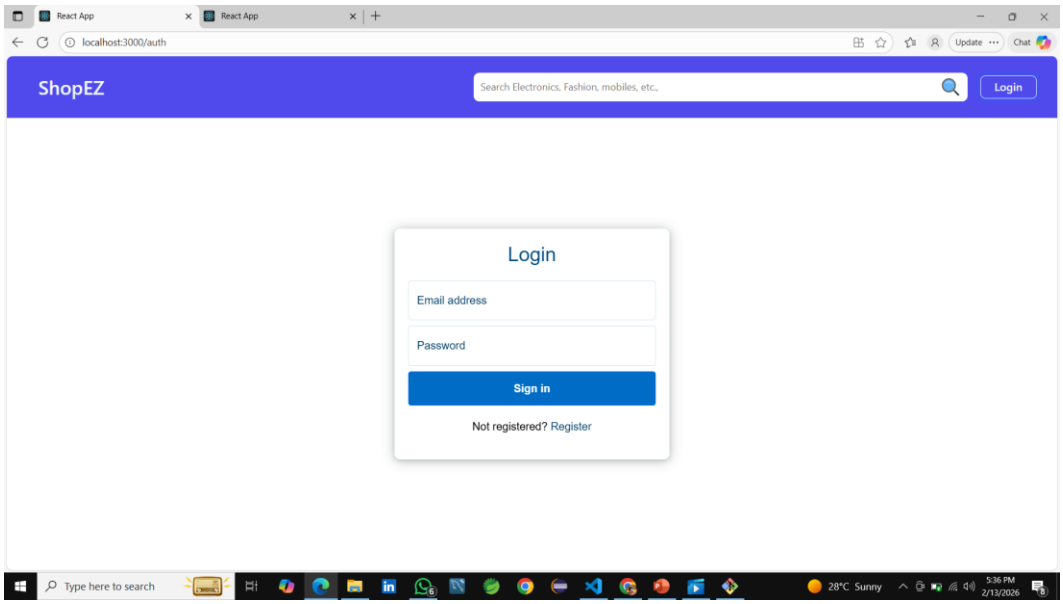

fig: Login with credentials

Product Grid & Filtering:

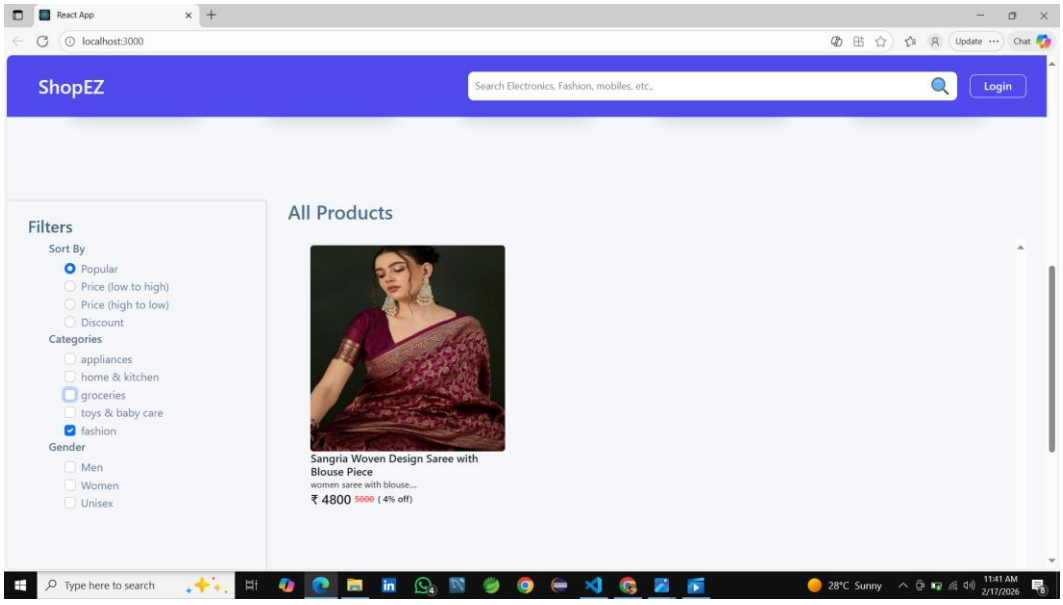

fig: Search products through filters

Admin Dashboard:

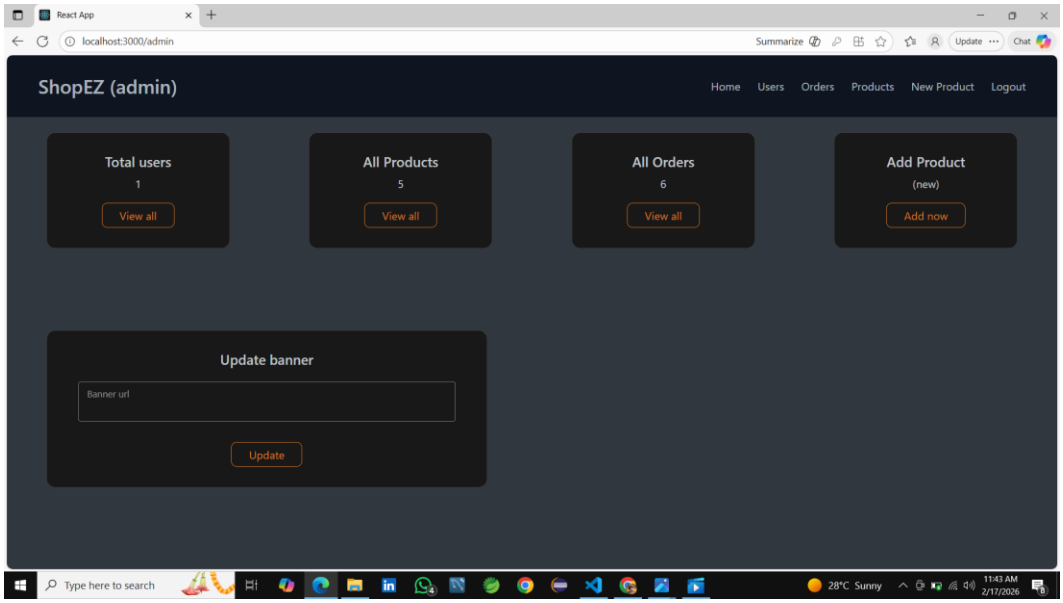

fig: Admin dashboard

Add Product Interface:

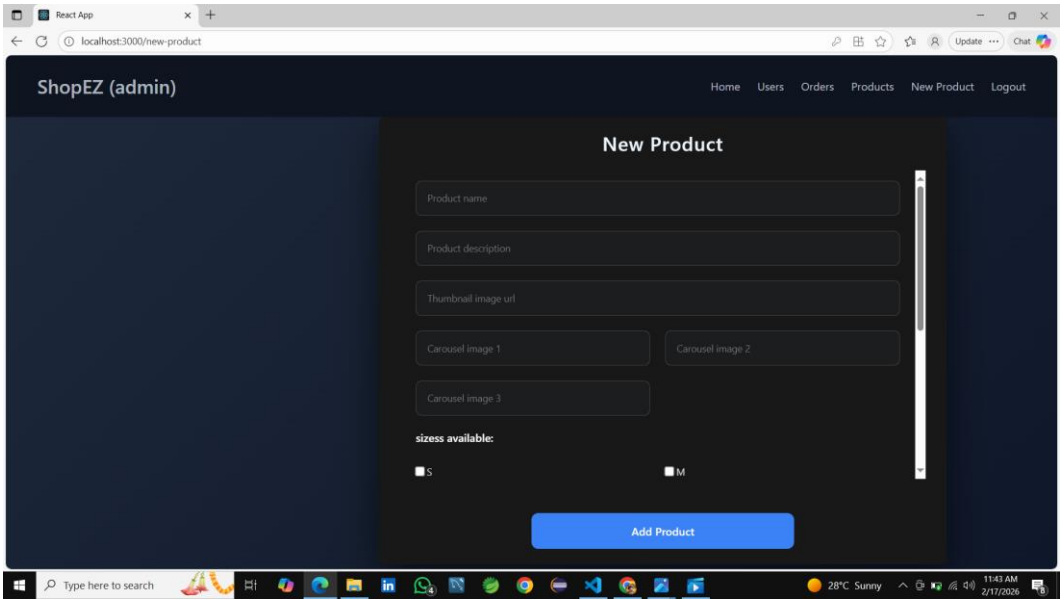

fig: Admin feature to add new products

10. Testing

Functional testing was performed on all core modules. The team used Postman to ensure that the backend endpoints return the correct status codes and JSON structures. Manual UI testing was conducted on different screen sizes to ensure the layout remains responsive.

11. Known Issues

1. Image Caching: Occasionally, newly added product images may take time to refresh in the grid.
2. Pagination: The catalog current loads all products at once; pagination is pending for larger datasets.

12. Future Enhancements

- Integrated Payment Gateway (Stripe/Razorpay)
- Advanced Search with Autocomplete
- User Reviews and Star Ratings
- Push Notifications for Sales