

# Full Stack Development with MERN

## Shopez – MERN Full Stack E-Commerce

### 1. Introduction

- **Project Title:** Shopez – MERN Full Stack E-Commerce

Shopez is a full-stack e-commerce application developed using the MERN stack (MongoDB, Express.js, React.js, and Node.js). The project aims to deliver a complete online shopping experience with modern UI/UX, fast performance, and secure backend architecture. The application supports customer shopping activities such as browsing products, adding items to the cart, placing orders, and tracking purchases. Additionally, ShopEZ includes a powerful admin dashboard for managing products, categories, orders, and user accounts. The platform is fully responsive, offering a seamless experience across desktop, tablet, and mobile devices.

### 2. Project Overview

- **Purpose:** The purpose of Shopez is to create an efficient, secure, and user-friendly online marketplace. It aims to simplify product discovery, streamline ordering, and offer a centralized platform for both buyers and sellers. The goal is to provide a fast-loading, responsive platform where users can shop with confidence while sellers manage their business effortlessly.

### 3. Architecture

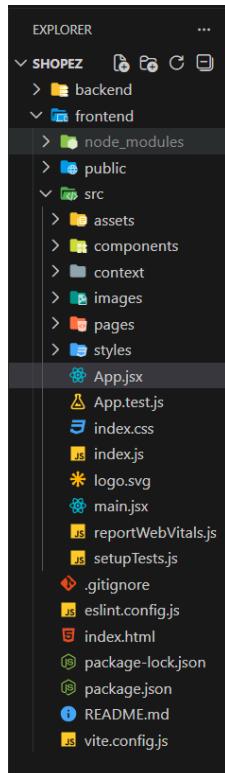
- **Frontend (React.js):** The frontend is built with React.js and follows a component-based architecture for reusability. React Router handles navigation, while Redux or Context API manages global state such as user login, cart data, and orders. The UI is built using clean layouts and TailwindCSS for faster styling. API interactions are handled with Axios.
- **Backend (Node.js + Express.js):** The backend follows RESTful API architecture. Express.js provides route management for authentication, products, cart, orders, and admin functionalities. Middleware handles error control, JWT authentication, and request validation. The backend is modular with controllers, routes, and models separated cleanly.

### 4. Setup Instructions

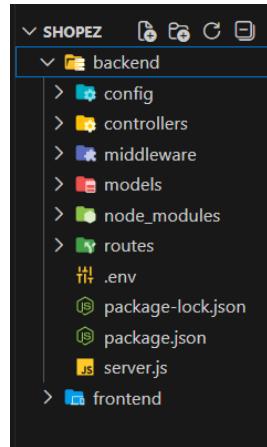
- **Prerequisites:**
  - Node.js (v16+ recommended)
  - npm or yarn
  - MongoDB Atlas or local MongoDB installation
  - Visual Studio Code
  - Postman (for API testing)
  - Git
- **Installation:**
  - cd frontend
  - npm install
  - cd ../backend
  - npm install

### 5. Folder Structure

- **Client:** React frontend.



- **Server:** Node.js backend.



## 6. Running the Application

- **Frontend:** `npm start` in the client directory.
  - `cd frontend`
  - `npm start`
- **Backend:** `npm start` in the server directory.
  - `cd frontend`
  - `npm start`

## 7. API Documentation

```

productRoutes.js
backend > routes > productRoutes.js > ...
1 import express from 'express';
2 import {
3   fetchProducts,
4   fetchProductDetails,
5   addNewProduct,
6   updateProduct,
7   fetchCategories,
8   deleteProductSimple
9 } from '../controllers/productController.js';
10
11 import { protect } from '../middleware/authMiddleware.js'; // import
12 import { buyProduct } from '../controllers/productController.js';
13
14 // Protected route: user must be logged in to buy
15
16 const router = express.Router();
17
18 // Public routes
19 router.get('/fetch-products', fetchProducts);
20 router.get('/fetch-product-details/:id', fetchProductDetails);
21 router.get('/fetch-categories', fetchCategories);
22
23 // Protected routes (require login)
24 router.post('/add-new-product', protect, addNewProduct);
25 router.put('/update-product/:id', protect, updateProduct);
26
27 router.delete('/delete-product/:id', deleteProductSimple);
28
29 export default router;
30
31

Schema.js
backend > models > Schema.js > ...
1 import mongoose from "mongoose";
2
3 const userSchema = new mongoose.Schema({
4   username: {type: String},
5   password: {type: String},
6   email: {type: String},
7   usertype: {type: String}
8 });
9
10 const adminSchema = new mongoose.Schema({
11   banner: {type: String},
12   categories: {type: Array}
13 });
14
15 const productSchema = new mongoose.Schema({
16   title: {type: String},
17   description: {type: String},
18   mainImg: {type: String},
19   carousel: {type: Array},
20   sizes: {type: Array},
21   category: {type: String},
22   gender: {type: String},
23   price: {type: Number},
24   discount: {type: Number}
25 });
26
27 const orderSchema = new mongoose.Schema({
28   userId: {type: String},
29   name: {type: String},
30   email: {type: String},
31   mobile: {type: String},
32   address: {type: String},
33   pincode: {type: String},
34   title: {type: String},
35   description: {type: String},
36   mainImg: {type: String},
37   size: {type: String}
38 });


```

## 8. Authentication

Authentication uses JWT stored in cookies for secure sessions. Authorization ensures only logged-in users can access protected routes while admin routes require an admin role. Passwords are securely hashed using bcrypt. Middleware validates tokens, checks user roles, and prevents unauthorized access.

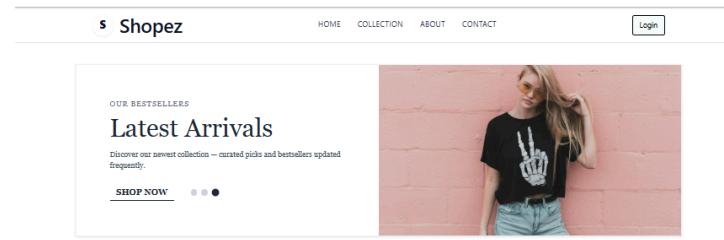
```

authMiddleware.js
backend > middleware > authMiddleware.js > protect
1 import jwt from "jsonwebtoken";
2
3 export const protect = (req, res, next) => {
4   let token;
5
6   if (
7     req.headers.authorization &&
8     req.headers.authorization.startsWith("Bearer")
9   ) {
10     try {
11       token = req.headers.authorization.split(" ")[1];
12
13       // verify token
14       const decoded = jwt.verify(token, process.env.JWT_SECRET);
15
16       req.user = decoded; // { id, role/usertype, ... }
17
18       next();
19     } catch (error) {
20       console.error(error);
21       res.status(401).json({ message: "Not authorized, token invalid" });
22     }
23   }
24
25   if (!token) {
26     res.status(401).json({ message: "Not authorized, no token provided" });
27   }
28 };

```

## 9. User Interface

The UI is designed with React and styled using TailwindCSS for clean, modern visuals. Components include product cards, navigation bars, hero banners, filters, cart UI, checkout forms, and admin dashboards. The design is fully responsive for smooth mobile and desktop use.



## 10. Testing

Testing is done using Postman for backend APIs and browser dev tools for frontend validation. Console logs, error boundaries, and custom middleware help identify issues. Manual testing ensures function reliability and user experience.

The screenshot shows a Postman interface with the following details:

- Request URL:** http://localhost:3000/api/products/fetch-products
- Method:** GET
- Headers:** (empty)
- Body:** (empty)
- Test Results:** 200 OK (84 ms, 5.34 KB)
- JSON Response:**

```

1 [ 
2   {
3     "_id": "69203e15a2bedff1cb3d6cf0",
4     "title": "iPhone 17 Pro Max",
5     "description": "The iPhone 17 Pro Max features a stunning 6.9-inch ProMotion display, A19 Bionic chip, advanced quad-camera system, titanium build, and all-day battery life. Designed for power users, photography lovers, and premium smartphone experience.",
6     "mainimg": "https://www.apple.com/newsroom/images/2025/09/apple-unveils-iphone-17-pro-and-iphone-17-pro-max/article/Apple-iPhone-17-Pro-color-lineup-250909_inline.jpg.large.jpg",
7     "carousel": [
8       "https://fdn2.gsmarena.com/vv/pics/apple/apple-iphone-17-pro-max-1.jpg",
9       "https://images1.kabum.com.br/produtos/fotos/925351/iphone-17-pro-apple-512gb-48mp-tela-6-3-super-retina-xdr-azul-intenso_1757692738_gg.jpg",
10      "https://shop.switch.com.my/cdn/shop/files/iPhone_17_Pro_Max_Silver_PDP_Image_Position_1_Silver_Colour__MY-EN.jpg?v=1757467933"
11    ],
12    "sizes": [],
13    "category": "Mobile",
14    "gender": "Unisex",
15    "price": 159999,
16    "discount": 5,
17    "__v": 0
18  ],
19 ]

```

## 11. Screenshots or Demo

## OUR BESTSELLERS

## Latest Arrivals

Discover our newest collection – curated picks and bestsellers updated frequently.

[SHOP NOW](#)

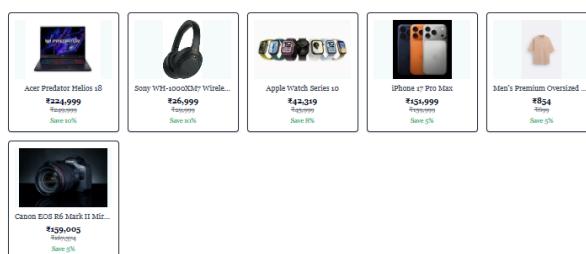
## LATEST COLLECTIONS

These products are loaded from your API.



## BEST SELLER

Products with discounts (higher discount first).



## Filters

[Clear all](#)

## Price Range

₹0 

## Categories

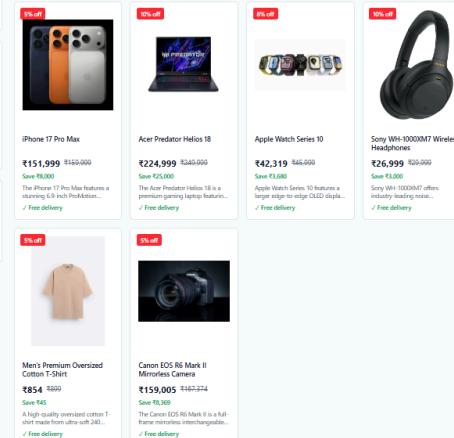
- Mobile
- Laptops
- Smart Watch
- Headphones
- Fashion

## Gender

- Men
- Women
- Unisex

## All Products

6 results

Sort by: [Featured](#)

## Admin Dashboard

Manage your e-commerce platform

[View all users](#)[View all products](#)[View all orders](#)[Add new product](#)[Update Banner](#)

Change the homepage banner image

## Banner Image URL:

[Update Banner](#)

Active Users

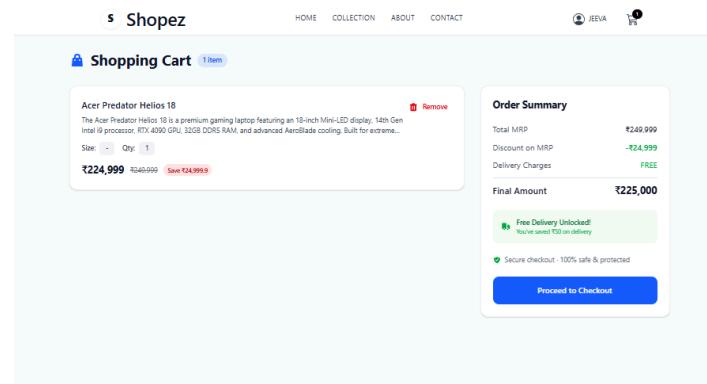
2

Total Products

6

Total Orders

1



## 12. Known Issues

- Payment gateway not integrated yet
- Email notifications pending
- Pagination for admin dashboard can be improved
- UI optimization needed for very small screens

## 13. Future Enhancements

- Add online payment gateway (Razorpay/Stripe)
- Add wishlist feature
- Add delivery tracking
- Add analytics dashboard for admin
- Add email + SMS notifications
- Add product comparison
- Add AI-based product recommendations