**E-commerce Project Documentation**

DEPI Graduation Project

By:

Hamada Reda

Ahmed Mahmoud

Hamis Hesham

Iman Mohamed

Sarah Mohsen

**Table of Contents**

1. Project Overview
2. Features
3. Technologies Used
4. Project Structure
5. Installation and Setup
6. Configuration
7. API Endpoints
8. Frontend Functionality
9. Authentication
10. Image Slider Component
11. Search Component
12. Shopping Cart
13. Dark Mode Switcher
14. Product Collection
15. Product Details
16. Footer And Header
17. Deployment
18. Future Enhancements
19. Conclusion
20. Domain
21. Design

**1. Project Overview**

This is a full-fledged **eCommerce website** built using **React** on the frontend and **Node.js/Express** on the backend. It allows users to browse products, search for items, add products to their cart, and complete the checkout process. The project aims to provide a seamless online shopping experience.

**2. Features**

* User authentication (sign-up, login, logout)
* Dark mode toggle
* Fully functional shopping cart with real-time updates
* Product search with autocomplete
* Responsive design
* Product categories and filtering
* Product details page
* Checkout page with billing and shipping details
* Admin dashboard for product and user management
* Image slider for promotions and banners
* API integration for fetching products, user data, etc.

**3. Technologies Used**

* **Frontend**:
  + React
  + React Router
  + Axios (for API requests)
  + PrimeReact (UI components)
  + TailwindCSS (for styling)
  + Framer Motion (for animations)
* **Backend**:
  + Node.js
  + Express.js
  + MongoDB (Database)
  + JWT (JSON Web Tokens) for authentication
* **Dev Tools**:
  + Postman (API testing)
  + ESLint (code linting)

**4. Project Structure**

A screen shot of a computer

Description automatically generated

**5. Installation and Setup**

**Requirements**

* Node.js (v14+)
* MongoDB
* A package manager (npm)

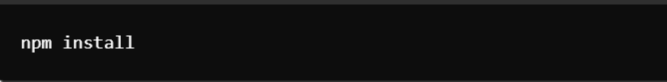
**Backend Setup**

1. Clone the repository:

A black background with white text

Description automatically generated

1. Install dependencies:



1. Configure environment variables:

* Create a .env file in the backend/ folder.
* Add the following variables:

A black screen with white text

Description automatically generated

1. Start the backend server:



**Frontend Setup**

1. Go to the frontend/ directory:

A black screen with a black background

Description automatically generated

1. Install dependencies:



1. Start the React development server:



**6. Configuration**

Ensure that the backend and frontend are configured to communicate correctly:

* Update the API base URL in the frontend at src/utils/api.js:



**7. API Endpoints**

**User Authentication**

* POST /api/v1/users/register: Register a new user
* POST /api/v1/users/login: Login an existing user
* **GET** /api/v1/users/profile: Fetch user profile details (requires authentication)

**Products**

* **GET** /api/v1/products: Fetch all products
* **GET** /api/v1/product/:id: Fetch product details by ID
* **Patch** /api/v1/ product: Edit product
* Delete /api/v1/ products Delete product

**categories**

* **GET** /api/v1/categories: Fetch product categories
* **Patch** /api/v1/categories: Edit product categories
* Delete /api/v1/categories: Delete product categories

**Carousels**

* POST /api/v1/ carousel: Add a carousel to the slider image
* **GET** /api/v1/ carousels: Fetch all carousels
* **Patch** /api/v1/ carousel: Edit carousels
* Delete /api/v1/ carousel: Delete carousels

**Order**

* **GET** /api/v1/Orders: Fetch all Orders
* **GET** /api/v1/Order/:id: Fetch Order details by ID
* **Patch** /api/v1/ Order: Edit Order
* Delete /api/v1/ Order Delete Order

**8. Frontend Functionality**

**Image Slider Component**

The image slider is a responsive component built with Framer Motion. It fetches slides from the backend and allows users to navigate between promotional banners. The buttons can dynamically link to product categories.

**Search Component**

* Allows users to search for products by name.
* The autocomplete feature is built using PrimeReact's AutoComplete component.
* Products are fetched from the backend, and results are shown as the user types.

**Shopping Cart**

* The cart dynamically updates as users add or remove items.
* The cart icon displays the current number of items.
* Users can view an order summary before proceeding to checkout.

**Product Collection**

* + Displays a grid or list of products in a specific category or collection.
  + Users can filter by category, price, and other product attributes.
  + Pagination for navigating through multiple pages of products.
  + Each product card includes an image, price, title, and link to the product details page.

**Product Details**

* + Displays detailed information about a specific product.
  + **Product Information**: Includes product images, description, price, specifications, and customer reviews.
  + **Add to Cart**: Button for adding the product to the shopping cart.
  + **Quantity Selector**: Allows users to select the number of items before adding to the cart.
  + **Related Products**: Suggests similar or related products based on the current product category.

**Header**

* + Provides site-wide navigation and access to essential features.
  + **Logo and Navigation Links**: Links to home, shop, collections, and contact pages.
  + **Search Bar**: Includes the search functionality with an AutoComplete feature.
  + **User Dropdown**: Displays login/logout options and a link to the user profile or dashboard.
  + **Cart Icon**: A clickable icon showing the number of items in the cart, redirecting to the cart page.

**Footer**

* + Provides site-wide information and links.
  + **Quick Links**: Links to important pages like About, Contact, FAQ, and Terms & Conditions.
  + **Social Media Links**: Icons linking to social media profiles.
  + **Newsletter Subscription**: An input field allowing users to subscribe to a mailing list.

**9. Authentication**

The website uses **JWT** for authentication. Tokens are stored in **HTTP-only cookies** to ensure secure transmission. Logged-in users can access their profiles, view order history, and proceed with purchases.

**10. Image Slider Component**

The slider dynamically displays carousel images, fetched from the backend, promoting products or collections. It also links to product categories via buttons that dynamically update based on the content of each slide.

**11. Search Component**

The SearchComponent uses the PrimeReact AutoComplete component, allowing users to search for products by title. The data is fetched from the API, and suggestions are displayed as the user types. Clicking on a product redirects the user to its details page.

**12. Shopping Cart**

The cart is an essential feature of the e-commerce app:

* Real-time cart updates: Users can see the number of items in their cart in the header.
* The sidebar updates automatically when products are added.
* Items in the cart can be viewed, adjusted, or removed from the checkout page.

**13. Dark Mode Switcher**

The Dark Mode Switcher allows users to toggle between light and dark modes for better accessibility and user experience. It is implemented using React hooks and Tailwind's dark mode class utilities.

**14. Product Collection**

The **Product Collection** feature allows users to browse and filter products by categories, ensuring an organized shopping experience. It is implemented using **React** for dynamic rendering and **Tailwind CSS** for styling.

* **Dynamic Category Filtering**: Users can select categories to view a filtered list of products, ensuring they quickly find what they’re looking for.
* **Backend Integration**: The component fetches products from the backend API based on the selected category. This ensures that only relevant products are displayed.
* **Pagination and Sorting**: The collection page supports pagination for large product inventories, and users can sort products by price, popularity, or relevance.
* **Responsive Design**: The collection layout adjusts automatically across different screen sizes for seamless usability on mobile, tablet, and desktop devices. Tailwind CSS utility classes ensure smooth styling across all breakpoints.

**15. Product Details Page**

The Product Details page provides in-depth information about each product, allowing users to view product descriptions, pricing, and images. It is built to be responsive and includes dynamic features for a seamless shopping experience.

* **Dynamic Data Rendering**: The product details are fetched from the backend using the product's unique ID, ensuring that each page displays accurate and up-to-date information.
* **Add to Cart**: Users can add items to their cart directly from the product details page. The cart automatically updates, and the user is notified of the action.
* **Image Gallery**: The product page includes an image gallery where users can view multiple images of the product. Images are displayed in a carousel format, allowing users to swipe or click through.
* **Responsive Design**: The page is designed to work on all devices, including mobile, tablet, and desktop. Tailwind CSS utilities are used to manage layout adjustments for different screen sizes.
* **Dark Mode Compatibility**: The Product Details page supports both light and dark modes, providing users the option to toggle between themes for a comfortable viewing experience. This feature is implemented using React hooks and Tailwind CSS's dark utilities. When dark mode is enabled, the background, text, and UI elements adjust to offer a visually appealing interface in low-light environments.

**16. Header and Footer**

**Header**

* The header is designed to provide easy navigation throughout the e-commerce platform. It includes a logo, navigation links, a search bar, user authentication links, and a shopping cart icon.
* The header is fully responsive and adapts to various screen sizes.
* The search bar allows users to quickly find products.
* The shopping cart icon dynamically updates to show the number of items in the cart.
* The user authentication section displays login and sign-up links when a user is not authenticated and shows a user dropdown menu when authenticated.
* The header integrates with the Dark Mode Switcher, adjusting its colors and appearance according to the current mode (light or dark).

**Footer**

* The footer provides essential links for users, including categories, contact information, and other important site pages.
* It is designed to be responsive, with the layout adjusting for different screen sizes.
* It also follows the dark mode theme, ensuring a seamless visual experience whether the user is in light or dark mode.

Both the header and footer make use of Tailwind's dark mode utilities and React hooks to ensure a consistent and adaptive design across the platform.

**17. Deployment**

To deploy the application:

* **Backend**: Deploy the Node.js/Express app on platforms like Heroku, AWS, or Digital Ocean.
* **Frontend**: Deploy the React app using services like Netlify or Vercel.
* Ensure to update the environment variables with production values.

**18. Future Enhancements**

* **Wishlist functionality**: Allow users to save products for later.
* **Advanced filtering**: Implement more advanced product filtering by price, rating, and availability.
* **Payment integration**: Add payment gateway integration such as Stripe or PayPal.
* **Order history**: Allow users to view their past orders.

**19. Conclusion**

This eCommerce project demonstrates a complete end-to-end web application, from user registration to product browsing and checkout. The project is scalable and customizable, with plenty of room for future enhancements and integrations.

**20.Domain**

* Domain URL For Server: <server-esw.up.railway.app>
* Domain URL For Client: <https://client-esw.vercel.app/>

**21. Design**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated **Admin Pages**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

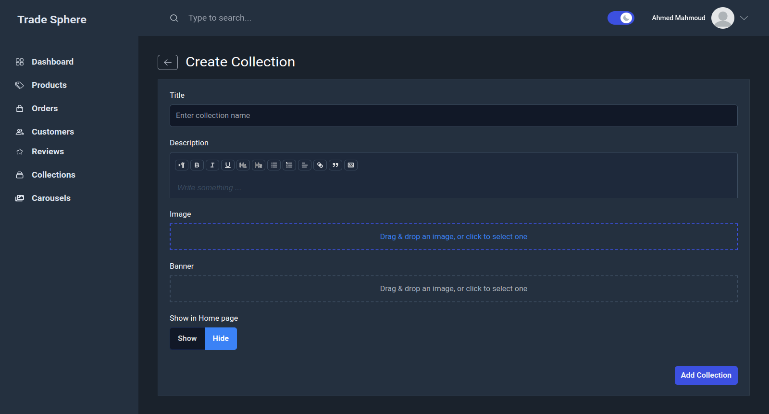
Description automatically generated

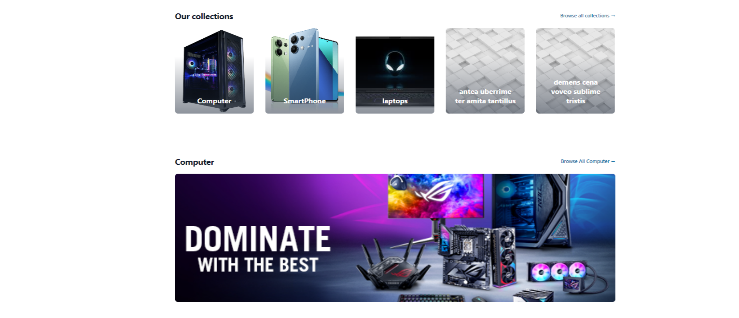
A screenshot of a computer

Description automatically generatedA screenshot of a computer

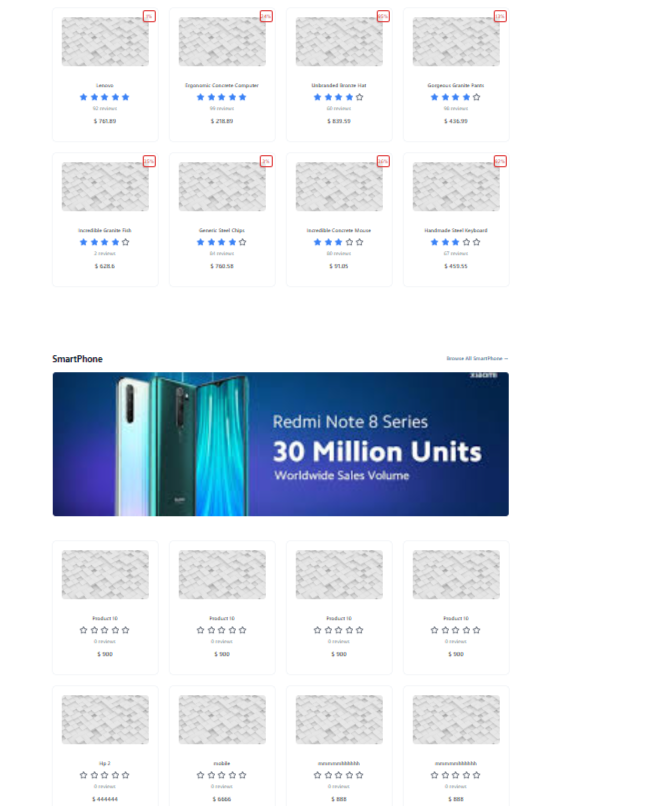
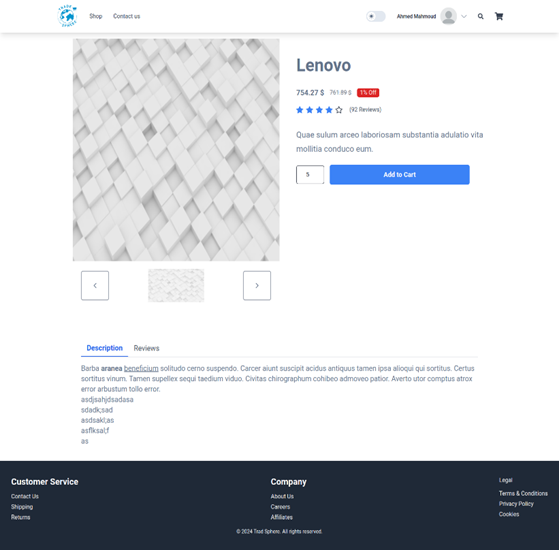
Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

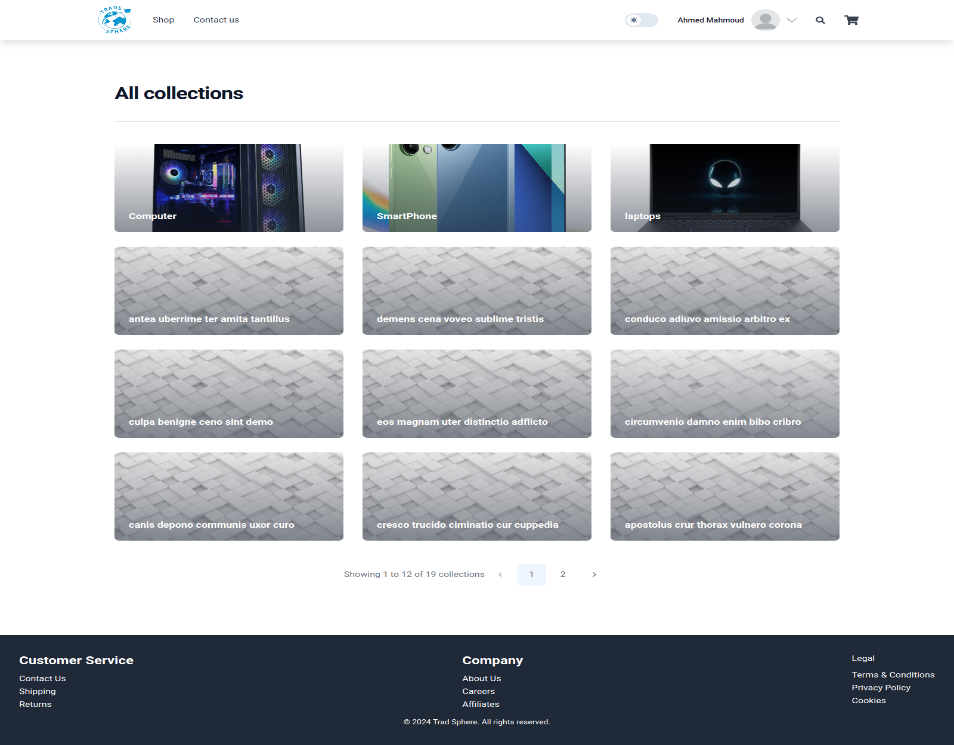
**A screenshot of a computer

Description automatically generatedUser Pages**



A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated