Full Stack Development with MERN Frontend Development Report

Date	5 July 2024
Team ID	SWTID1720091047
Project Name	Project – E-commerce Platform
Maximum Marks	

Project Title: E-Commerce Platform

Date: 5 July 2024

Prepared by: Hrithik Khanna K B

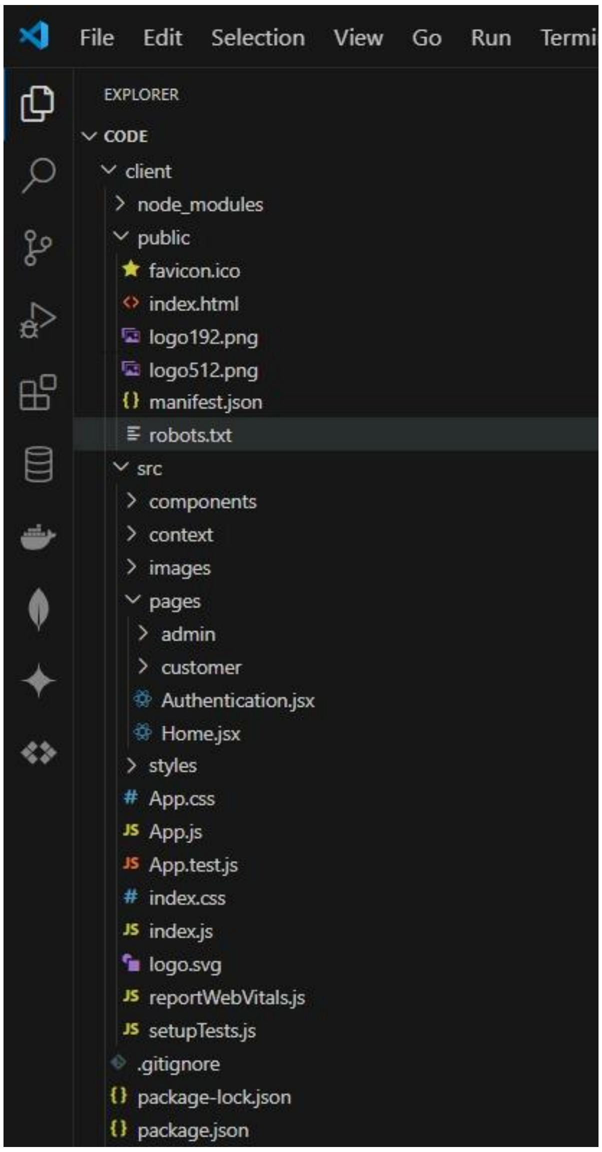
Objective

The objective of this documentation is to provide a comprehensive overview of the frontend development process for the food ordering system project. It aims to detail the progress made in implementing key user interface components, outline the design and development decisions, and highlight the strategies employed to ensure a user-friendly, responsive, and accessible application. This documentation serves as a reference for current and future development efforts, ensuring continuity, consistency, and clarity in the project's evolution.

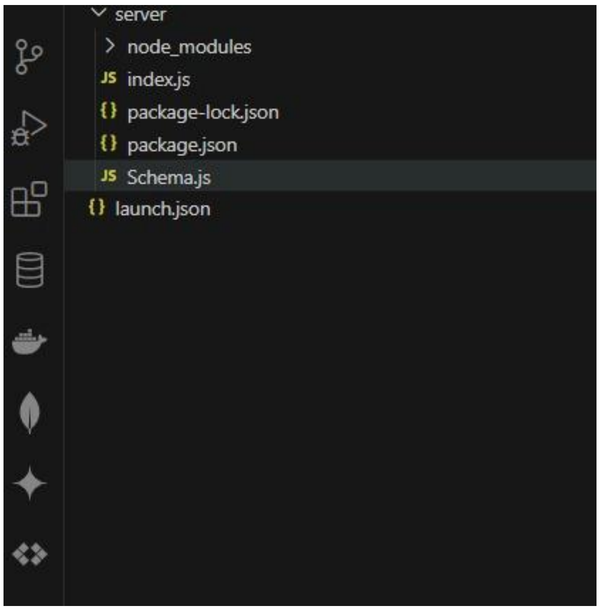
Technologies Used

- **Frontend Framework:** React.js
- **State Management:** Context API
- **UI Framework/Libraries:** Bootstrap
- API Libraries: Axios
- **Testing Libraries:** @testing-library/jest-dom, @testing-library/react, @testing-library/user-event
- **Routing:** React Router (react-router-dom)
- **Icons:** React Icons (react-icons)
- **Performance Monitoring:** web-vitals
- **Build and Development Tools:** react-scripts

Project Structure



```
EXPLORER
∨ CODE
  ∨ client
    > node_modules
    ∨ public
      favicon.ico
      index.html
      logo192.png
      logo512.png
      manifest.json
      robots.txt
    ∨ src
      > components
      > context
      > images
      ∨ pages
        > admin
        > customer
        Authentication.jsx
        Home.jsx
      > styles
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
    .gitignore
    package-lock.json
    package.json
```



```
  ∨ server
    > node_modules
    index.js
    package-lock.json
    package.json
    Schema.js
  launch.json
```

Public

The public folder contains resources that are publicly accessible and served directly by the browser. It includes the favicon.ico, which is the icon displayed in the browser tab, and the index.html file, which is the main entry point for the application, containing essential meta tags and links to external resources. The manifest.json file provides configuration for the web app, detailing aspects such as the name, icons, and starting URL. Additionally, it includes logo192.png and logo512.png for different icon resolutions and the robots.txt file, which instructs web crawlers on which pages to index or ignore.

Src

The src directory is the heart of the React application, housing the core logic and components. The index.js file serves as the entry point, bootstrapping the React application. The App.js file contains the main application component, while App.test.js includes tests for this component. reportWebVitals.js is used for measuring performance metrics, and setupTests.js configures the testing environment. This directory encapsulates the structure and functionality of the application, ensuring organized and maintainable code.

Components

The components folder holds reusable UI components that are utilized throughout the application. These include Footer.jsx for the footer section, Login.jsx for the login page, Navbar.jsx for the navigation bar, PopularRestaurants.jsx for showcasing popular restaurants, Register.jsx for the registration page, and Restaurants.jsx for listing restaurants. The Authentication.jsx component handles authentication processes, while Home.jsx serves as the main home page component. This directory promotes reusability and consistency in the UI.

Context

The context directory is dedicated to managing global state within the application. It contains the GeneralContext.js file, which provides context providers and consumers, facilitating state management across different components. This setup enhances efficiency and organization by centralizing the state, making it easier to share data and functions between components without prop drilling.

Pages

The pages directory contains components that represent different routes or views within the application. It is subdivided into sections based on user roles.

Admin

The admin folder within pages includes components for the administrative section of the application. These components, such as Admin.jsx for the main dashboard, AllOrders.jsx for displaying all orders, AllProducts.jsx for managing products, and AllUsers.jsx for user management, are crucial for performing various administrative functions and maintaining control over the application's content and user base.

Customer

The customer folder within pages is focused on user-related components. It includes Cart.jsx for the shopping cart, CategoryProducts.jsx for displaying products by category, IndividualRestaurant.jsx for showing details of a specific restaurant, and Profile.jsx for user profile management. These components provide essential functionality for the customer experience, enabling users to browse, select, and purchase products, as well as manage their personal information.

Styles

The styles directory contains CSS files for styling the components and pages within the application. It includes individual stylesheets for specific components, ensuring a consistent look and feel across the application. Additionally, it has global styles that apply universally, maintaining visual coherence and enhancing user experience.

Root Files

The root of the project contains essential configuration and metadata files. The .gitignore file specifies which files and directories should be ignored by Git, preventing unnecessary files from being tracked. The package.json and package-lock.json files list the project's dependencies and their versions, ensuring a consistent development environment and facilitating dependency management. These files are fundamental for setting up, configuring, and maintaining the project's environment.

Key Components

1. **App.js**
   - **Responsible for routing and main application layout.**
   - This file serves as the central hub of the application, managing the routes and ensuring the correct components are rendered based on the user's navigation.
2. **/components**
   - **Contains reusable UI components used across the application.**
   - This directory includes components like Navbar, Login, Footer, and Register. These components are designed to be used in various parts of the application to ensure a consistent user interface and to promote reusability.
3. **/pages**
   - **Includes different sections of the directory for different roles like Admin and Customer, separated for each role.**
   - **Admin Directory**
     - Contains components for administrative tasks, allowing the admin to manage various aspects of the application:
       - AllOrders.jsx: Displays all orders in the system.
       - AllProducts.jsx: Manages and displays all products available.
       - AllUsers.jsx: Manages and displays all users.
   - **Customer Directory**
     - Contains components that cater to the customer's shopping experience:
       - Cart.jsx: Manages the shopping cart.
       - CategoryProducts.jsx: Displays products by category.
       - IndividualProduct.jsx: Shows details of a specific product.
       - Profile.jsx: Allows customers to view and edit their profile.
   - **Common Pages**
     - Contains components used universally across the application:
       - Authentication.jsx: Handles user authentication for login and registration.
       - Home.jsx: The main landing page of the application.

Routing

Routing is managed using React Router. Here are the routes:

General Routes:

- **‘/auth’**: Route for authentication pages, rendering <Authentication />.
- **‘/’**: Exact path for the home page, rendering <Home />.
- **‘/cart’**: Route for the cart page, rendering <Cart />.
- **‘/product/’**: Dynamic route for individual product pages, rendering <IndividualProduct />.
- **‘/category/’**: Dynamic route for category-specific product pages, rendering <CategoryProducts />.
- **‘/profile’**: Route for user profile management, rendering <Profile />.

Admin Routes:

- **‘/admin’**: Route for the admin dashboard, rendering <Admin />.
- **‘/all-products’**: Route for viewing all products (admin), rendering <AllProducts />.
- **‘/all-users’**: Route for viewing all users (admin), rendering <AllUsers />.
- **‘/all-orders’**: Route for viewing all orders (admin), rendering <AllOrders />.

Route Descriptions:

1. **General Routes:**
    - **‘/auth’**: This route directs users to the authentication pages where they can log in or register, handled by the <Authentication /> component.
    - **‘/’**: The root path that takes users to the home page of the e-commerce site, rendered by the <Home /> component.
    - **‘/cart’**: This route shows the shopping cart where users can view items they intend to purchase, managed by the <Cart /> component.
    - **‘/product/**

      ’: A dynamic route that leads to individual product pages, allowing users to view detailed information about a specific product, managed by the <IndividualProduct /> component.

    - **‘/category/**

': Another dynamic route that displays products filtered by category, rendered by the <CategoryProducts /> component.

  - **'/profile'**: This route allows users to manage their profiles, including viewing and updating their personal information, handled by the <Profile /> component.

2. **Admin Routes:**
   - **'/admin'**: This route takes the user to the admin dashboard where administrative activities are performed, managed by the <Admin /> component.
   - **'/all-products'**: This route allows the admin to view and manage all the products in the system, rendered by the <AllProducts /> component.
   - **'/all-users'**: This route lets the admin view and manage all registered users, handled by the <AllUsers /> component.
   - **'/all-orders'**: This route displays all orders placed in the system, allowing the admin to manage them, rendered by the <AllOrders /> component.

Integration with Backend

The frontend communicates with the backend APIs hosted on http://localhost:6001/. Key endpoints include:

General Endpoints:

- **GET /api/data**: Retrieves data for display purposes across the application.
- **POST /api/user/login**: Handles user authentication and returns a token.
- **POST /api/user/register**: Registers a new user in the system.

Product Endpoints:

- **GET /api/products**: Fetches all products available in the store.
- **GET /api/products/**: Fetches details of a specific product by its ID.
- **POST /api/products**: Adds a new product to the store (admin).
- **PUT /api/products/**: Updates details of an existing product by its ID (admin).
- **DELETE /api/products/**: Removes a product from the store by its ID (admin).

Cart Endpoints:

- **GET /api/cart**: Fetches all items in the user's cart.
- **POST /api/cart**: Adds an item to the user's cart.
- **PUT /api/cart/**: Updates the quantity or details of an item in the cart.
- **DELETE /api/cart/**: Removes an item from the cart.

Order Endpoints:

- **POST /api/orders**: Places a new order.
- **GET /api/orders**: Fetches all orders (admin).
- **GET /api/orders/**: Fetches details of a specific order by its ID.
- **PUT /api/orders/**: Updates the status of an existing order (admin).

User Endpoints:

- **GET /api/users/**: Fetches details of a user by their ID.
- **GET /api/users**: Fetches all users (admin).
- **PUT /api/users/**: Updates user details (admin).

User Interface (UI) Design

Design Principles

The UI design for the e-commerce system follows these core design principles:

1. **Consistency**: Ensuring uniformity across all pages by using a consistent color scheme, typography, and component styling.
2. **Simplicity**: Keeping the design simple and intuitive to make navigation and interaction easy for users.
3. **Responsiveness**: Creating a responsive design that works seamlessly across various devices and screen sizes.
4. **Accessibility**: Designing for inclusivity, ensuring that the application is accessible to users with different abilities.
5. **Visual Hierarchy**: Using visual cues like size, color, and spacing to indicate the importance of elements and guide users' attention.
6. **Feedback**: Providing immediate and clear feedback to user actions through visual and auditory cues.

UI Framework/Library

- **Implemented using**: React for the component-based structure, styled-components for CSS-in-JS, and Material-UI for a cohesive, responsive, and modern design system.

Key UI Components

1. **Navbar**
   - **Description**: A top navigation bar that includes links to Home, Products, Categories, Cart, and Profile.
   - **Features**: Dropdown menus for categories, a search bar, and user account management options.
2. **Home Page**
   - **Description**: The landing page showcasing featured products, promotional banners, and links to popular categories.
   - **Features**: Carousel for promotional banners, product grids for featured items, and quick links to different sections.
3. **Product Listing Page**
   - **Description**: Displays products in a grid or list format based on the selected category or search results.
   - **Features**: Filtering options by price, brand, and rating; sorting options; pagination.
4. **Product Detail Page**
   - **Description**: Detailed view of a single product, including images, description, price, reviews, and related products.
   - **Features**: Image gallery, add-to-cart button, review section, and related product suggestions.
5. **Cart Page**
   - **Description**: Displays items added to the cart with options to update quantities or remove items.
   - **Features**: Summary of total cost, proceed to checkout button, and estimated shipping costs.
6. **Checkout Page**
   - **Description**: Step-by-step process for finalizing the purchase, including entering shipping details and payment information.
   - **Features**: Form validation, order summary, and confirmation page after successful order placement.
7. **Profile Page**
   - **Description**: User account management page where users can view and edit their personal information, view order history, and manage addresses.
   - **Features**: Editable forms for personal details, order history list, and address management.

8. **Admin Dashboard**
    - **Description**: Admin-specific interface for managing the store’s products, users, and orders.
    - **Features**: Overview statistics, product management forms, user list with management options, and order tracking.
9. **Authentication Pages**
    - **Description**: Login and registration pages for user authentication.
    - **Features**: Forms for entering login credentials or registration details, password reset option.