Full Stack Development with MERN

Frontend Development Report

Date	4 July 2024
Team ID	SWTID1720171853
Project Name	Project - Food Ordering System
Maximum Marks	

Project Title: Project - Food Ordering System

Date: 4 July 2024

Prepared by: Kishore Babu U

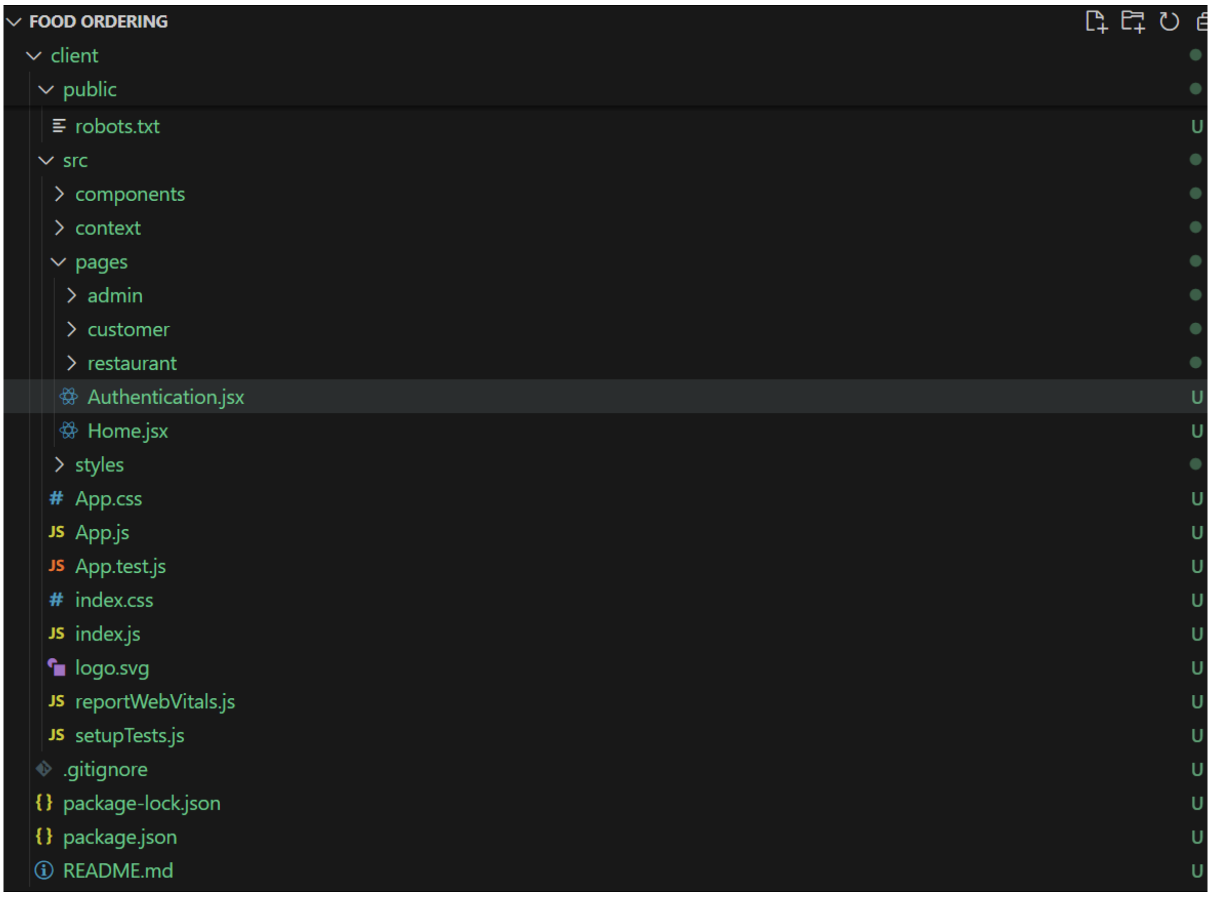
Objective

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.

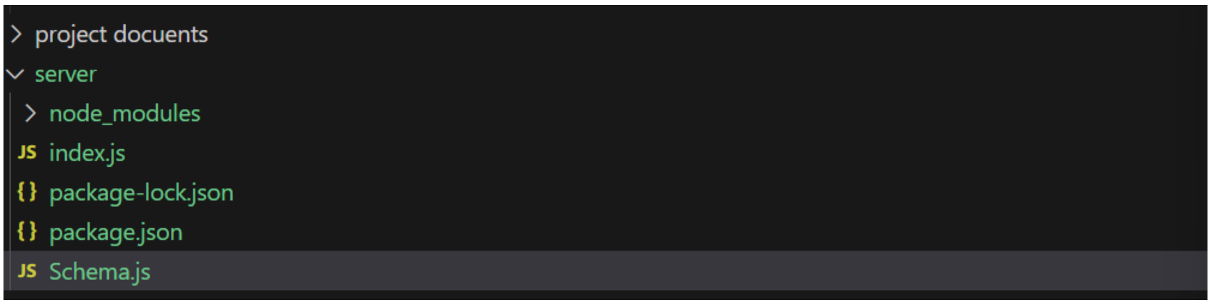
Technologies Used

- **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



```
FOOD ORDERING
  client
    public
      robots.txt                 U
    src
      components
      context
      pages
        admin
        customer
        restaurant
        Authentication.jsx       U
        Home.jsx                 U
      styles
      App.css                    U
      App.js                     U
      App.test.js                U
      index.css                  U
      index.js                   U
      logo.svg                   U
      reportWebVitals.js         U
      setupTests.js              U
    .gitignore                   U
    package-lock.json            U
    package.json                 U
    README.md                    U
```



```
project docuents
server
  node_modules
  index.js
  package-lock.json
  package.json
  Schema.js
```

Public

The frontend structure contains various folders starting from ‘**public**’, which contains the ‘**favicon.ico**’ that shows in the browser's tab top. The main ‘index.html’ file is the starting file of the frontend, containing meta links and links to external sources. It also includes the ‘**manifest.json**’ file, a configuration file for the web app, specifying details like the name, icons, and start URL. Additionally, it contains logo images and the ‘**robots.txt**’ file, which provides instructions for web crawlers about which pages to index or not.

Src

The ‘**src**’ directory contains the source code for the React application. This is where the main logic and components of the application reside. It includes all the JavaScript files and components necessary for the application's functionality, as well as the ‘**index.js**’ file, which serves as the entry point for the React application.

Components

The ‘**components**’ folder houses reusable components used throughout the application. These components include ‘**Footer.jsx**’ for the footer section, ‘**Login.jsx**’ for the login page, ‘**Navbar.jsx**’ for the navigation bar, ‘**PopularRestaurants.jsx**’ for displaying popular restaurants, ‘**Register.jsx**’ for the registration page, and ‘**Restaurants.jsx**’ for displaying a list of restaurants. These components are crucial for building the user interface.

It also contains ‘**Authentication.jsx’** component for authentication (login/register) and also have ‘**Home.jsx’** for main home page component.

Context

The ‘**context**’ directory contains ‘**GeneralContext.js**’ file which provides context providers and consumers for managing global state. This folder is essential for maintaining state across different components of the application, making state management more efficient and organized.

Pages

The ‘**pages**’ directory contains page components, each representing a different route in the application.

Admin

The ‘**admin**’ folder within pages contains components related to the admin section of the application. This includes ‘**Admin.jsx**’ for the main admin dashboard, ‘**AllOrders.jsx**’ for displaying all orders, ‘**AllProducts.jsx**’ for displaying all products, ‘**AllRestaurants.jsx**’ for displaying all restaurants, and ‘**AllUsers.jsx**’ for displaying all users. These components are vital for managing the administrative functions of the application.

Customer

The ‘**customer’** folder within pages contains components related to the customer section of the application. This includes ‘**Cart.jsx’** for the shopping cart, ‘**CategoryProducts.jsx’** for displaying products by category, ‘**IndividualRestaurant.jsx’** for displaying details of a single restaurant, and ‘**Profile.jsx’** for displaying and editing user profiles. These components cater to the user-specific functionalities of the application.

Restaurant

The ‘**restaurant’** folder within pages contains components related to the restaurant section of the application. This includes ‘**EditProduct.jsx’** for editing a product, **NewProduct.jsx** for adding a new product, ‘**RestaurantHome.jsx’** for the restaurant home page, ‘**RestaurantMenu.jsx’** for displaying the restaurant menu, and ‘**RestaurantOrders.jsx’** for displaying orders related to the restaurant. These components are designed to facilitate restaurant-specific operations within the application.

Styles

This directory contains the CSS style sheets for all the components and also contain file for global styles for the entire application.

JavaScript Files

The JavaScript files are located in the ‘**src**’ directory and include ‘**App.js**’ for main application component, ‘**App.test.js**’ for the tests of the component, ‘**index.js**’ for the entry point for the React Application, ‘**reportWebVitals.js**’ used for measuring performance metrics and ‘**setupTests.js**’ for configuration for setting up tests.

Root Files

The root files of the project include configuration and metadata files like ‘.gitignore’ file which specifies files and directories to be ignored by Git

and other packages JSON files to list the dependencies needed.

Key Components

1. **App.js**
   - Responsible for routing and main application layout.
2. **/components**
   - Contains reusable UI components used across the application like Navbar,Login,Footer,body and Register components are present in this directory.
3. **/pages**
   - Includes different section of directory for the different roles like Admin,Customer and also the Restaurant page separated for each roles.
   - The Admin directory Contains the allorders,allproducts,allusers allrestaurent access for the admin to use.
   - The customer directory has the cart page,product category page separate restaurants page for the customer and the customer profile pages.

- The Restaurent Directory has the new item adding ,editing existing item, Home,order and menu pages for the restaurant.
- It also Contains pages files for authentication and home page

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 />’.
- **‘/restaurant/:id’**: Dynamic route for individual restaurant pages, rendering ‘<IndividualRestaurant />’.
- **‘/category/: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-restaurants’: Route for viewing all restaurants (admin), rendering ‘<AllRestaurants />’.
- ‘/all-users’: Route for viewing all users (admin), rendering ‘<AllUsers />’.
- ‘/all-orders:’ Route for viewing all orders (admin), rendering ‘<AllOrders />’.

Restaurant Routes:

- **‘/restaurant’**: Route for the restaurant dashboard or home page, rendering ‘<RestaurantHome />’.
- **‘/restaurant-orders’**: Route for managing restaurant orders, rendering ‘<RestaurantOrders />’.
- **‘/restaurant-menu**’: Route for managing the restaurant menu, rendering ‘<RestaurantMenu />’.
- **‘/new-product’**: Route for adding a new product to the restaurant menu, rendering ‘<NewProduct />’.
- **‘/update-product/’**:id: Dynamic route for updating a specific product in the restaurant menu, rendering ‘<EditProduct />’.

Integration with Backend

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

- **GET /api/data** - Retrieves data for display.
- **POST /api/user/login** - Handles user authentication.
- **POST /api/user/login -** Authenticates a user and returns a token.
- **POST /api/approve-user -** Approves a restaurant user.
- **GET /api/fetch-restaurants** - Fetches all restaurants.
- **PUT /api/update-order-status -** Updates the status of an order.
- **POST /api/add-new-product -** Adds a new product (food item).
- **GET /api/fetch-cart -** Fetches all cart items.
- **POST /api/add-to-cart** - Adds an item to the cart.
- **PUT /api/remove-item** - Removes an item from the cart.
- **GET /api/fetch-user-details/-** Fetches details of a user by ID.
- **GET /api/fetch-users -** Fetches all users.

User Interface (UI) Design

- The UI design follows
    - Intuitive Navigation.
    - Responsive and Accessible Design.
    - Visual Hierarchy and Clarity.
    - Efficient Search and Filtering.
    - Engaging Imagery and Presentation.
    - Performance Optimization.
- Implemented using UI Framework (Bootstrap).