Full Stack Development with MERN

Frontend Development Report

Date	15th July 2024
Team ID	SWTID1720076571
Project Name	Food Ordering System
Maximum Marks	

Project Title: Food Ordering System

Date: 15th July 2024

Prepared by:

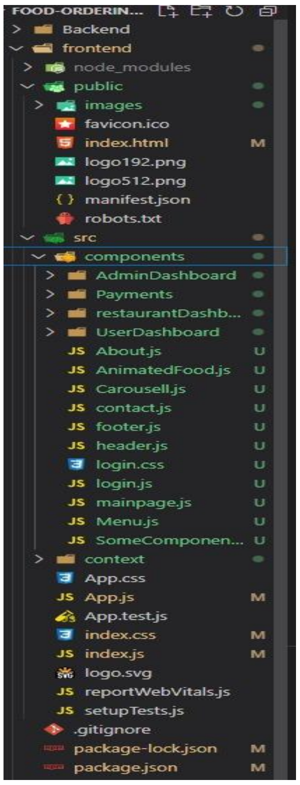
Objective

The objective of this report is to document the frontend development progress and key aspects of the user interface implementation for the SB Food Ordering System project.

Technologies Used

- **Frontend Framework:** React.js
- **State Management:**none
- **UI Framework/Libraries:**None
- **API Libraries:** None

Project Structure



```
FOOD-ORDERIN...
> Backend
v frontend
  > node_modules
  v public
    > images
      favicon.ico
      index.html                 M
      logo192.png
      logo512.png
      manifest.json
      robots.txt
  v src
    v components
      > AdminDashboard
      > Payments
      > restaurantDashb...
      > UserDashboard
        About.js                 U
        AnimatedFood.js          U
        Carousell.js             U
        contact.js               U
        footer.js                U
        header.js                U
        login.css                U
        login.js                 U
        mainpage.js              U
        Menu.js                  U
        SomeComponen...          U
    > context
      App.css
      App.js                     M
      App.test.js
      index.css                  M
      index.js                   M
      logo.svg
      reportWebVitals.js
      setupTests.js
    .gitignore
    package-lock.json            M
    package.json                 M
```

Overview

The React frontend project for the food ordering app is structured to maintain a clean separation of concerns, making it scalable and maintainable. The main components of the project include the `public` directory for static files and the `src` directory for the React application logic.

Directory Structure

1. **frontend**
   - The main directory for the React frontend application.
   - **node_modules**
     - Contains all the npm packages required for the project. Automatically generated and updated based on `package.json`.
   - **public**
     - **images**
       - Stores image assets like icons and logos used across the application.
     - `favicon.ico`
       - The icon displayed in the browser tab.
     - `index.html`
       - The main HTML file that serves as the entry point for the React application.
     - `logo192.png`, `logo512.png`
       - Logo images for different screen resolutions.

        - `manifest.json`
            - Contains metadata for the web app, useful for Progressive Web App (PWA) capabilities.
        - `robots.txt`
            - Instructions for web crawlers on how to index the site.
    - **src**
        - The main source directory for the React application.
        - **components**
            - Contains reusable React components.
            - **AdminDashboard**
                - Components related to the admin dashboard functionality.
            - **Payments**
                - Components handling the payment process.
            - **restaurantDashboard**
                - Components for the restaurant's interface.
            - **UserDashboard**
                - Components for the user dashboard interface.
        - `About.js`
            - Component displaying information about the application.
        - `AnimatedFood.js`
            - Component for animated food items, possibly for visual effects or dynamic displays.
        - `Carousell.js`
            - Component for implementing a carousel/slider functionality.
        - `contact.js`
            - Component for the contact page.
        - `footer.js`
            - Component for the footer section of the application.
        - `header.js`
            - Component for the header/navigation bar.
        - `login.css`
            - CSS file for styling the login page.
        - `login.js`
            - Component for the login functionality.
        - `mainpage.js`
            - Component for the main landing page.
        - `Menu.js`
            - Component for displaying the menu items.
        - `SomeComponent.js`
            - Placeholder name, should be replaced with the actual functionality description.
        - **context**
            - Context API files to manage global state across the application.
        - `App.css`
            - Main CSS file for the App component.
        - `App.js`

            - Main App component that serves as the root of the React application.
        - `App.test.js`
            - Test file for the App component.
        - `index.css`
            - Global CSS file for styling.
        - `index.js`
            - Entry point for the React application, responsible for rendering the App component into the DOM.
        - `logo.svg`
            - SVG logo used in the application.
        - `reportWebVitals.js`
            - Utility for measuring performance metrics.
        - `setupTests.js`
            - Configuration file for setting up tests.
    - `.gitignore`
        - Specifies which files and directories to ignore in version control.
    - `package-lock.json`
        - Automatically generated file that locks the versions of dependencies for consistency.
    - `package.json`
        - Lists the project dependencies and scripts.

Key Components and Functionality

1. **AdminDashboard, Payments, RestaurantDashboard, UserDashboard**
    - These directories contain components specific to different user roles within the application (admin, restaurant, user) and manage their respective functionalities.
2. **Common Components**
    - `About.js`, `AnimatedFood.js`, `Carousell.js`, `contact.js`, `footer.js`, `header.js`, `login.js`, `mainpage.js`, `Menu.js`
        - These components handle common pages and functionalities such as about us, contact, header/footer, login, main page, and displaying the menu.
3. **Global State Management**
    - The `context` directory is used to manage global state using React's Context API, ensuring state is accessible throughout the app.
4. **Styling**
    - CSS files such as `login.css`, `App.css`, `index.css` are used to style different components and maintain a consistent look and feel across the application.
5. **Entry Point**
    - `index.js` initializes and renders the React application, while `App.js` serves as the root component that encapsulates the entire app's structure and routing.

Routing

Main Routes

1. **Home Page**
   - **Path:** `/`
   - **Component:** `MainPage`
   - **Description:** The landing page of the application, typically showing an overview of the app and featured items.
2. **About Page**
   - **Path:** `/about`
   - **Component:** `About`
   - **Description:** A page providing information about the application and the team behind it.
3. **Animated Food**
   - **Path:** `/animated-food`
   - **Component:** `AnimatedFood`
   - **Description:** Page showcasing animated food items, possibly for promotions or special effects.
4. **Carousell**
   - **Path:** `/carousell`
   - **Component:** `Carousell`
   - **Description:** A page featuring a carousel/slider of images or items, often used for highlighting specials or new arrivals.
5. **Contact Page**
   - **Path:** `/contact`
   - **Component:** `Contact`
   - **Description:** Page containing contact information and possibly a form for users to reach out to the support team.
6. **Login Page**
   - **Path:** `/login`
   - **Component:** `Login`
   - **Description:** The login page for users to authenticate themselves.
7. **Menu Page**
   - **Path:** `/menu`
   - **Component:** `Menu`
   - **Description:** Page displaying the menu items available for ordering.
8. **Admin Dashboard**
   - **Path:** `/admin-dashboard`
   - **Component:** `AdminDashboard`
   - **Description:** Dashboard for admin users to manage the application, such as viewing orders, managing users, etc.
9. **User Dashboard**
   - **Path:** `/user-dashboard`
   - **Component:** `UserDashboard`
   - **Description:** Dashboard for regular users to view their orders, account details, etc.

10. **Restaurant Dashboard**
    - **Path:** `/restaurant-dashboard`
    - **Component:** `RestaurantDashboard`
    - **Description:** Dashboard for restaurant owners or staff to manage their restaurant's orders, menu items, etc.

Integration with Backend

The frontend of the Food Ordering App communicates with the backend APIs hosted on [backend URL]. This integration ensures smooth data exchange and functionality across different modules of the application.

Key Endpoints

1. **User Authentication**
    - **Endpoint:** `POST /api/user/login`
    - **Description:** Handles user login by validating credentials and returning a token for authenticated sessions.
2. **User Registration**
    - **Endpoint:** `POST /api/user/register`
    - **Description:** Registers a new user by saving user details to the database.
3. **Retrieve Orders**
    - **Endpoint:** `GET /api/orders`
    - **Description:** Fetches a list of orders for the authenticated user or admin.
4. **Create Order**
    - **Endpoint:** `POST /api/orders`
    - **Description:** Allows users to create a new order by submitting order details.
5. **Admin Dashboard**
    - **Endpoint:** `GET /api/admin/dashboard`
    - **Description:** Retrieves data and metrics for the admin dashboard.
6. **Restaurant Dashboard**
    - **Endpoint:** `GET /api/restaurant/dashboard`
    - **Description:** Retrieves data and metrics for the restaurant dashboard.