DOCUMENTATION

Home Page

Purpose

The Home Page is the entry point for users visiting the Food Delivery App. It provides an overview of the available food items, allows users to browse through the menu, and gives quick access to other key features of the app such as user authentication and the shopping cart.

Key Features

- Food Item Display: Showcases all available food items in a visually appealing manner using card components.
- Search Functionality: Allows users to search for specific food items using keywords.
- Filter Options: Enables users to filter food items based on categories, such as cuisines, dietary preferences, or price ranges.
- Add to Cart: Users can add food items to their cart directly from the Home Page.
- Navigation Bar: Provides easy navigation to other parts of the app such as Login, Register, Cart, and Admin Dashboard (if the user is an admin).
- Responsive Design: Ensures a seamless user experience across different devices and screen sizes.

Implementation Details

- Components:
    - Header.js: Contains the navigation bar with links to various pages.
    - FoodList.js: Displays a list of food items using the `FoodCard` component.
    - FoodCard.js: Represents individual food items with details like image, name, price, and an 'Add to Cart' button.
    - SearchBar.js: A component for searching food items.
    - Filter.js: A component for filtering food items based on selected criteria.

- State Management:
    - Uses React's `useState` and `useEffect` hooks to manage the state of food items, search queries, and filters.
    - Fetches food items from the backend API (`/api/foods`) and updates the state accordingly.
- Styling:
    - Utilizes Material-UI (MUI) for consistent and responsive styling.
    - Custom CSS for additional styling needs.

User Interaction

1. Browsing: Users can scroll through the list of food items.
2. Searching: Users can enter keywords in the search bar to find specific items.
3. Filtering: Users can apply filters to narrow down the displayed food items.
4. Adding to Cart: Users can add desired food items to their cart with a single click.

Cart Page

Purpose

The Cart Page allows users to view the items they have added to their shopping cart, modify quantities, and proceed to checkout. It provides a summary of the selected items and the total cost, enabling users to review their order before placing it.

Key Features

- Cart Item Display: Shows a list of all food items added to the cart, including their names, prices, quantities, and total cost.
- Quantity Modification: Allows users to increase or decrease the quantity of each item in the cart.

- Remove Items: Users can remove items from the cart if they change their mind.
- Order Summary: Displays the total number of items and the total cost of the order.
- Checkout Button: Provides a button for users to proceed to the checkout and place their order.

Implementation Details

- Components:
  - Cart.js: The main component for the Cart Page, displaying the list of cart items and the order summary.
  - CartItem.js: Represents individual items in the cart with options to modify quantity or remove the item.
  - OrderSummary.js: Displays a summary of the total items and cost, along with the checkout button.
- State Management:
  - Uses React's `useState` and `useContext` hooks to manage the state of the cart.
  - Cart items are stored in the context to provide a global state that can be accessed across the application.
  - Updates the cart state based on user interactions (e.g., modifying quantities or removing items).
- Styling:
  - Utilizes Material-UI (MUI) for styling the cart items and the order summary.
  - Custom CSS for additional styling needs.

User Interaction

1. Viewing Cart: Users can see all the items they have added to their cart.
2. Modifying Quantities: Users can increase or decrease the quantity of each item using buttons provided.
3. Removing Items: Users can remove items from the cart by clicking the remove button.
4. Proceeding to Checkout: Users can click the checkout button to proceed with placing their order.

Admin Dashboard Page

Purpose

The Admin Dashboard Page is designed for administrators to manage the app's content and users. It provides an overview of key metrics and quick access to management functions such as adding new food items, viewing orders, and managing users.

Key Features

- Overview Section: Displays key metrics such as the total number of orders, total revenue, and the number of active users.
- Manage Food Items: Admins can add, update, or delete food items from the menu.
- View Orders: Admins can view all orders placed by users, including order details and statuses.
- Manage Users: Admins can view user profiles and manage user accounts.

Implementation Details

- Components:
  - AdminDashboard.js: The main component for the admin dashboard.
  - Overview.js: Displays key metrics.
  - ManageFoods.js: Provides functionality to manage food items.
  - ManageOrders.js: Allows admins to view and manage orders.
  - ManageUsers.js: Enables user management.
- State Management:
  - Uses React's `useState` and `useEffect` hooks to manage state.
  - Fetches data from the backend API endpoints such as `/api/admin/overview`, `/api/admin/foods`, `/api/admin/orders`, and `/api/admin/users`.
  - Uses context for global state management where necessary.
- Styling:
  - Utilizes Material-UI (MUI) for a consistent and responsive design.
  - Custom CSS for additional styling needs.

User Interaction

1. Viewing Metrics: Admins can see an overview of key metrics.
2. Managing Food Items: Admins can add new food items, update existing items, or delete items.
3. Viewing Orders: Admins can view details of all orders placed by users.
4. Managing Users: Admins can view and manage user profiles and accounts.

Login Page

Purpose

The Login Page allows users to authenticate themselves by providing their email and password. It ensures secure access to the app's features and personalized user experience.

Key Features

- Login Form: A form for users to enter their email and password.
- Error Handling: Displays error messages for incorrect login credentials.
- Redirects: Redirects users to their respective home pages based on their roles (admin or regular user) upon successful login.
- Link to Registration Page: Provides a link for new users to register.

Implementation Details

- Components:
    - Login.js: The main component for the login page, containing the login form and handling user authentication.
- State Management:
    - Uses React's `useState` and `useContext` hooks to manage the form state and authentication status.

    - On form submission, sends a POST request to the backend API endpoint `/api/auth/login`.
    - Stores the JWT token in local storage and updates the global authentication state upon successful login.
- Styling:
    - Utilizes Material-UI (MUI) for form elements and layout.
    - Custom CSS for additional styling needs.

User Interaction

1. Entering Credentials: Users enter their email and password.
2. Submitting the Form: Users click the login button to submit the form.
3. Error Handling: If the credentials are incorrect, an error message is displayed.
4. Successful Login: Upon successful login, users are redirected to their respective home pages.

Order History Page

Purpose

The Order History Page allows users to view a record of their past orders. It provides details such as order dates, items ordered, total cost, and order status.

Key Features

- Order List: Displays a list of past orders with summary information.
- Order Details: Provides detailed information about each order, including items, quantities, prices, and total cost.
- Order Status: Shows the current status of each order (e.g., pending, completed, cancelled).

Implementation Details

- Components:
    - OrderHistory.js: The main component for the order history page.
    - OrderCard.js: Represents individual orders with summary information.
    - OrderDetails.js: Displays detailed information about a specific order.
- State Management:
    - Uses React's `useState` and `useEffect` hooks to manage the state of the order history.
    - Fetches order data from the backend API endpoint `/api/orders/:userId` where `:userId` is the ID of the logged-in user.
- Styling:
    - Utilizes Material-UI (MUI) for styling the order list and details.
    - Custom CSS for additional styling needs.

User Interaction

1. Viewing Order List: Users can see a list of their past orders.
2. Viewing Order Details: Users can click on an order to view detailed information about it.
3. Checking Order Status: Users can check the status of their orders.