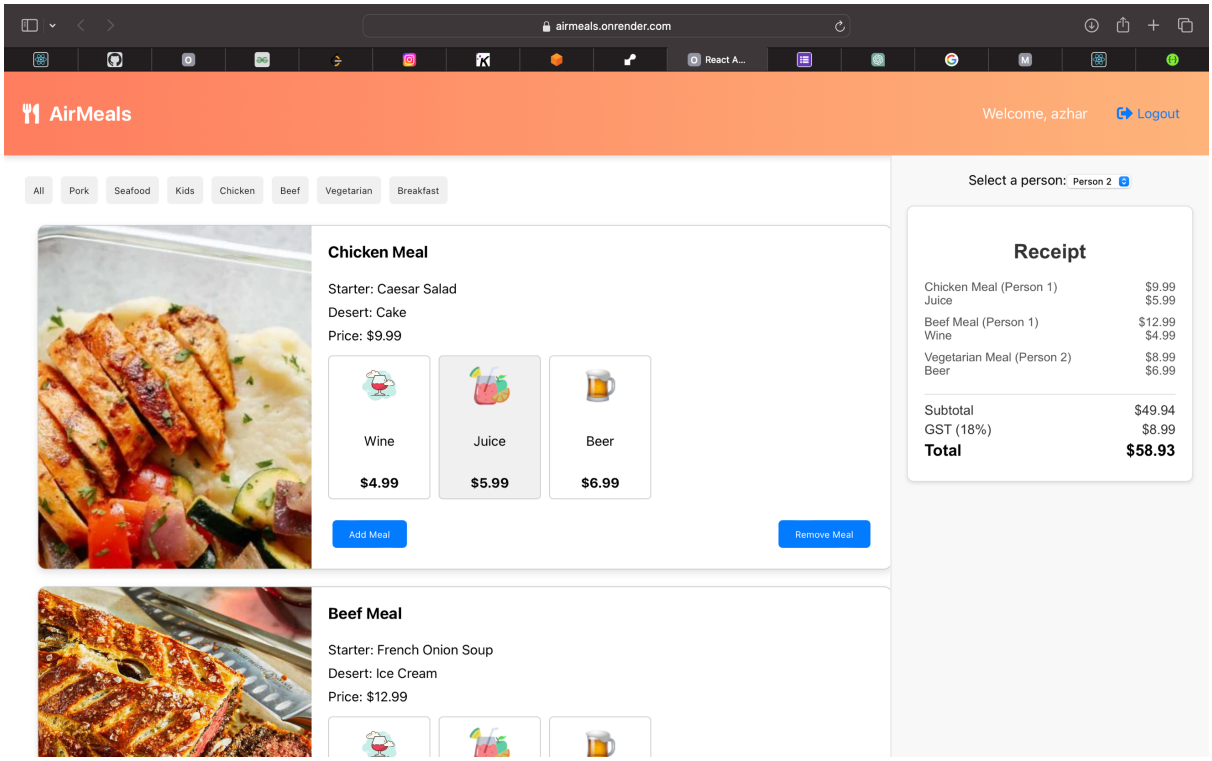
AirMeals : Food Ordering App Documentation



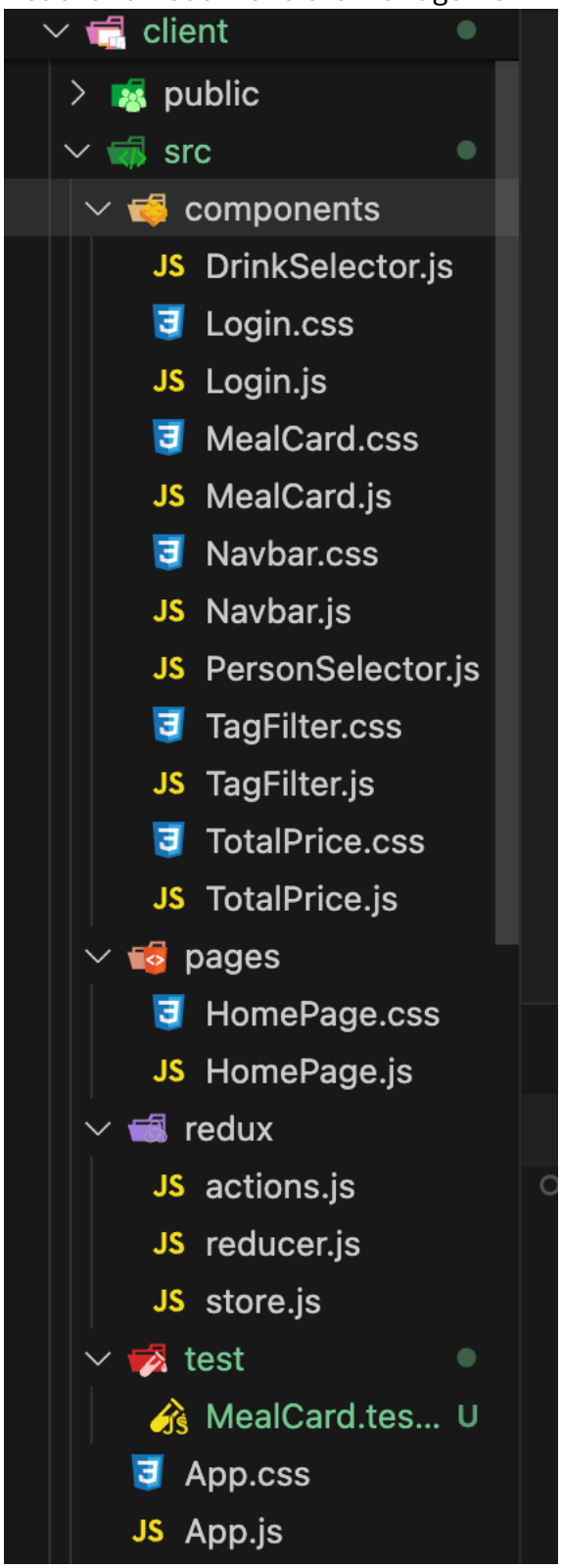
Links

- **Owned By GitHub: https://github.com/bunnysayzz**
- **GitHub Repository**: https://github.com/bunnysayzz/flightmenu.git
- **Hosted Application**: https://airmeals.onrender.com
- **Swagger API Documentation**: https://flightmenubackend.onrender.com/api-docs

Overview

This project is a meal selection application for an airline service. Users can log in, select meals and drinks, and view the total price including GST. The application is built using

React and Redux for state management.



Components

`DrinkSelector.js`

- **Purpose**: Allows users to select a drink for a meal.
- **Props**:
- `mealId`: ID of the meal.

- `drinks`: List of available drinks.
- `onSelectDrink`: Callback function when a drink is selected.
- `reset`: Boolean to reset the selected drink.
- **State**:
- `selectedDrinkId`: ID of the selected drink.
- **Key Functions**:
- `handleDrinkClick`: Handles drink selection and triggers `onSelectDrink`.

`Login.js`

- **Purpose**: Provides a login interface for users.
- **State**:
- `email`: User's email.
- `password`: User's password.
- **Key Functions**:
- `handleLogin`: Handles user login.
- `handleGuestLogin`: Fills in guest login credentials.

`MealCard.js`

- **Purpose**: Displays meal details and allows adding/removing meals.
- **Props**:
- `meal`: Meal object containing details.
- **State**:
- `selectedDrink`: Selected drink for the meal.
- `showPopup`: Boolean to show/hide the popup.
- `resetDrinkSelector`: Boolean to reset the drink selector.
- **Key Functions**:
- `handleAddMeal`: Adds the meal to the selected meals list.
- `handleRemoveMeal`: Removes the meal from the selected meals list.
- `handleSelectDrink`: Sets the selected drink.

`Navbar.js`

- **Purpose**: Navigation bar with login/logout functionality.
- **Key Functions**:
- `handleLogout`: Logs out the user and redirects to the login page.

`PersonSelector.js`

- **Purpose**: Allows users to select a person for whom the meal is being selected.
- **State**:
- `selectedPerson`: Currently selected person.
- **Key Functions**:
- `handlePersonChange`: Dispatches action to set the selected person.

`TagFilter.js`

- **Purpose**: Filters meals based on tags.
- **State**:
- `labels`: List of available labels.
- `tagFilter`: Currently selected tag filter.
- **Key Functions**:
- `handleTagClick`: Dispatches action to set the tag filter.

TotalPrice.js

- **Purpose**: Displays the total price of selected meals including GST.
- **State**:
- `selectedMeals`: List of selected meals.
- **Key Functions**:
- Calculates total price and GST.

Pages

HomePage.js

- **Purpose**: Main page displaying meals, person selector, and total price.
- **State**:
- `currentPage`: Current page number for pagination.
- **Key Functions**:
- `handlePageChange`: Changes the current page.
- Fetches meals on component mount.

Redux

actions.js

- **Action Types**: Defines action types like `ADD_MEAL`, `REMOVE_MEAL`, etc.
- **Action Creators**: Functions to create actions,
e.g., `addMeal`, `removeMeal`, `setPerson`, etc.
- **Async Actions**: `fetchMeals` to fetch meals from the backend.

reducer.js

- **Initial State**: Defines the initial state of the application.
- **Reducers**: Handles actions to update the state,
e.g., `ADD_MEAL`, `REMOVE_MEAL`, `SET_PERSON`, etc.

store.js

- **Purpose**: Configures the Redux store with reducers and middleware.
- **Middleware**: Uses `redux-thunk` for async actions.

Server Documentation

Overview

This server is built using Node.js and Express. It provides APIs for meal data and user authentication. Swagger is integrated for API documentation, making it easy to test and understand the available endpoints.

Dependencies

- **express**: Web framework for Node.js.
- **cors**: Middleware to enable Cross-Origin Resource Sharing.
- **swagger-jsdoc**: Generates Swagger specification from JSDoc comments.

- **swagger-ui-express**: Serves Swagger UI for API documentation.

Server Setup

`server.js`

This is the main server file that sets up the Express application, configures middleware, defines API routes, and integrates Swagger for API documentation.

`swagger.js`

This file configures Swagger to generate API documentation from JSDoc comments and serves the Swagger UI.

API Endpoints

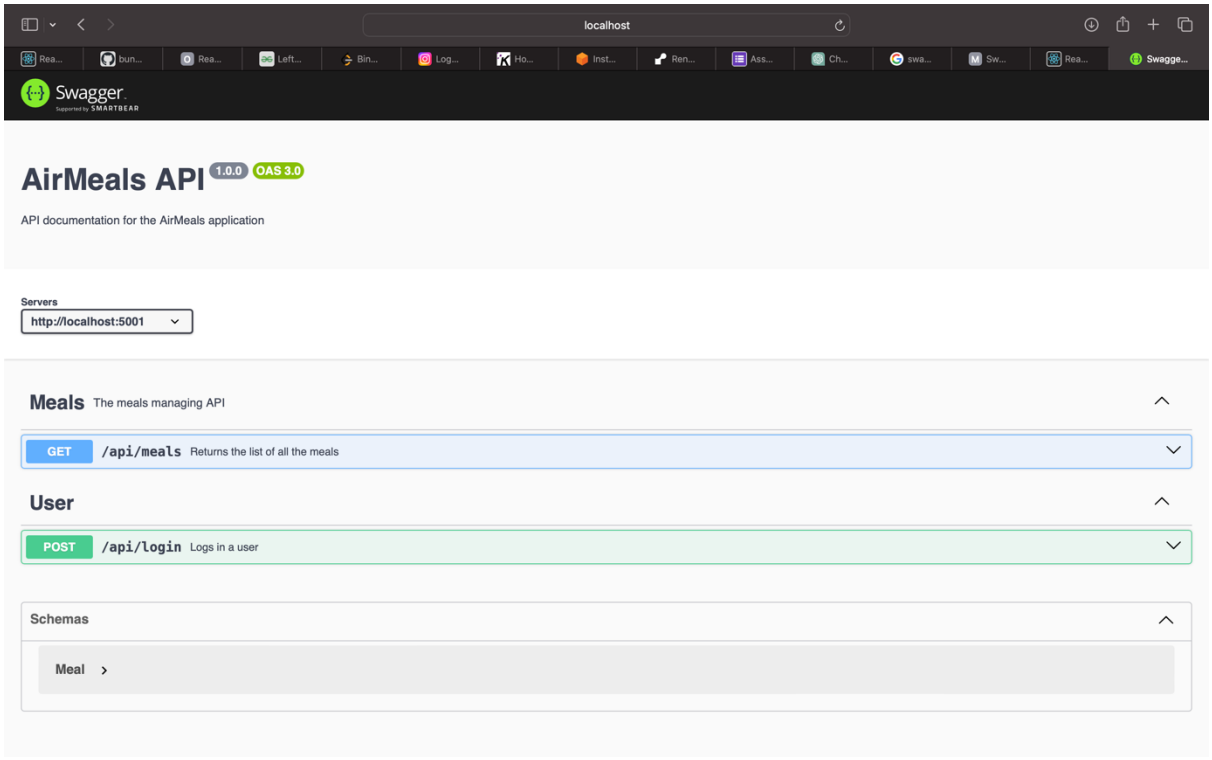
Get Meals

- **Endpoint**: `/api/meals`
- **Method**: GET
- **Description**: Returns the list of all meals.
- **Response**: JSON array of meal objects.
- **Example**: http://localhost:5001/api/meals

User Login

- **Endpoint**: `/api/login`
- **Method**: POST
- **Description**: Logs in a user with email and password.
- **Request Body**: JSON object with `email` and `password`.
- **Response**: JSON object with user details if successful, or an error message if authentication fails.
- **Example**: http://localhost:5001/api/login

Swagger API Documentation



- **Endpoint**: `/api-docs`
- **Description**: Provides a web interface to view and test the API endpoints.
- **Example**: http://localhost:5001/api-docs

How the Server Works

- **Setup**: The server is set up using Express. CORS is enabled to allow cross-origin requests, and JSON bodies are parsed using `express.json()`.
- **Sample User**: A sample user is defined for login purposes. In a real application, user data should be securely stored and managed.
- **API Routes**:
- **GET /api/meals**: Returns a list of meals from `mealsData.json`.
- **POST /api/login**: Authenticates a user based on the provided email and password.
- **Swagger Integration**: Swagger is configured to generate API documentation from JSDoc comments and is served at `/api-docs`.

5. **Server Listening**: The server listens on port 5001 and logs a message when it starts.

Testing Tools

- **Swagger UI**: For interactive API documentation and testing.
- **Postman**: For manual API testing.
- **Jest**: For automated testing