AIO web app documentation

Objective:

Creating ecommerce web app using MERN stack.

This ecommerce web app containing three pages 1. Food 2. Movies and 3. Sports.

Food: This page will fetch the data from data base and shows data in food page. Admin has access to add or delete the data and user have access to see the data in food page.

Movies: This page will fetch the data from data base and shows data in movies page. Admin has access to add or delete the data and user have access to see the data in movie page. User can also add the ticket price.

Sports: This page will call the crickedata.org website free api and fetch the data which user has searched. That data is displayed in the page

CODE: https://github.com/nithin006/project_aio.git

Code Working Explanation:

1. Server.js

1.1 Dependencies

- **express:** A web application framework for Node.js.
- **cors:** Middleware for handling Cross-Origin Resource Sharing (CORS).
- **mongoose:** An ODM (Object Data Modeling) library for MongoDB and Node.js.

1.2 Middleware

- **express.json():** Middleware for parsing incoming JSON data.
- **cors():** Enables CORS for all routes.

1.3 MongoDB Connection

- Connects to MongoDB Atlas cluster.

1.4 MongoDB Schemas and Models

- **User Schema and Model:** Defines a user schema and model for MongoDB.
- **Food Schema and Model:** Defines a food schema and model for MongoDB.
- **Movie Schema and Model:** Defines a movie schema and model for MongoDB.

1.5 Routes

- **POST /postfood:** Adds a new food item to the database.
- **GET /getfood:** Retrieves all food items from the database.
- **POST /postmovie:** Adds a new movie item to the database.
- **GET /getmovie:** Retrieves all movie items from the database.
- **POST /register:** Registers a new user in the database.

- **POST /login:** Handles user login and redirects on success.
- **DELETE /deletefood/:id:** Deletes a food item by ID.
- **DELETE /deletemovie/:id:** Deletes a movie item by ID.

1.6 Server Start

- Listens on the specified port (8000).

2. Register.js

- A React component for user registration.
- Uses the axios library for making HTTP requests.
- State variables for user input (name, password, email).
- **handleSubmit** function to handle form submission.
- Sends a POST request to the **/register** endpoint on form submission.

3. Login.js

- A React component for user login.
- Uses the axios library for making HTTP requests.
- State variables for user input (name, password).
- **handleSubmit** function to handle form submission.
- Sends a POST request to the **/login** endpoint on form submission.
- Special case: If the user is 'admin', directly redirects to **/admin**.

4. Home.js

- A React component for the home page.
- Uses Bootstrap for styling.
- Displays a navigation bar (Navbar_) and a heading.

5. Navbar_.js

- A React component for the navigation bar.
- Uses Bootstrap components (Navbar, Nav).
- Contains navigation links for 'Food', 'Sport', and 'Movie'.

6.Food.js (Frontend):

1. **Component Structure:**
    - Fetches and displays food items from the backend.
    - Uses the **Navbar_** component for navigation.
2. **State and useEffect Hook:**

- Utilizes **useState** and **useEffect** for managing state and performing side effects.

3. **Fetching Data:**
    - Retrieves food data from the backend using the **fetch** API.
4. **Rendering Data:**
    - Maps over the fetched data to display individual food items.

7.Movie.js (Frontend):

1. **Component Structure:**
    - Fetches and displays movie items from the backend.
    - Utilizes the **Navbar_** component for navigation.
2. **State and useEffect Hook:**
    - Uses **useState** and **useEffect** for state management and side effects.
3. **Fetching and Rendering Data:**
    - Fetches movie data and renders individual movie items.
    - Keeps track of the total amount and allows adding tickets.

8.Sport.js (Frontend):

1. **Component Structure:**
    - etFches and displays sports series information from an external API.
    - Includes a search input to filter series data.
    - Uses the **Navbar_** component for navigation.
2. **State and useEffect Hook:**
    - Utilizes **useState** and **useEffect** for state management and side effects.
3. **Fetching Data:**
    - Uses Axios to fetch data from an external sports API.
4. **Rendering Data:**
    - Displays detailed information about cricket series based on the fetched data.

9.Admin.js:

1. **Component Structure:**
    - Displays the main heading for the Admin page.
    - Imports the **AdminNav** component for navigation.

10.AdminNav.js (AdminNavbar):

1. **Component Structure:**
    - Renders the navigation bar for the Admin section.

- Uses Bootstrap's **Navbar** and **Nav** components.
- Includes links for navigating to different Admin pages (**adminfoodpage**, **adminmoviepage**, **foodAdmin**, **movieAdmin**), and a logout link.

11.FoodAdmin.js:

1. **Component Structure:**
    - Allows the Admin to upload food items.
    - Utilizes the **AdminNav** component for navigation.
    - Contains form elements for entering food details (price, hotel) and uploading an image.
    - Displays a preview of the uploaded image.
    - On submit, sends a POST request to the backend to add a new food item.

12.MovieAdmin.js:

1. **Component Structure:**
    - Allows the Admin to upload movie items.
    - Utilizes the **AdminNav** component for navigation.
    - Contains form elements for entering movie details (price, theatre) and uploading an image.
    - Displays a preview of the uploaded image.
    - On submit, sends a POST request to the backend to add a new movie item.

13.AdminFoodPage.js:

1. **Component Structure:**
    - Fetches and displays a list of food items.
    - Allows the Admin to delete a food item.
    - Uses the **AdminNav** component for navigation.

14.AdminMoviePage.js:

1. **Component Structure:**
    - Fetches and displays a list of movie items.
    - Allows the Admin to delete a movie item.
    - Uses the **AdminNav** component for navigation.

15.App.js:

1. **Component Structure:**
    - Utilizes the **react-router-dom** library for routing.

- Defines routes for different pages such as Login, Register, Home, Food, Sport, Movie, Admin, FoodAdmin, MovieAdmin, AdminFoodPage, and AdminMoviePage.
- Utilizes the **Protected** component to ensure that certain routes are protected and require authentication.
- The **Protected** component checks for the presence of a **token** in **localStorage**. If the token is not present, the user is redirected to the login page; otherwise, the specified component is rendered.

16.Protected.js:

1. **Component Structure:**
    - A simple functional component that serves as a protective wrapper around routes.
    - Checks for the presence of a **token** in localStorage.
    - If a token is not found, it redirects the user to the login page using **Navigate**.
    - If a token is present, it renders the children (the content of the protected route).

RUN CODE:

1. Open two terminals one for React and another for node js.
2. Go to app directory and enter (**npm start**) in one terminal.
3. Go to app directory then to src and enter (**node server.js**) in another terminal.
4. The web app will open in default browser.

REFFER CODE: https://github.com/nithin006/project_aio.git