MERN STACK

Task 2: Movie Website Development

Project Objective

Build a feature-rich movie application using ReactJS as the frontend framework, integrated with a suitable movie-related API (e.g., TMDb API, OMDb API). The app will showcase dynamic movie data with interactive features such as searching, filtering, and detailed movie information**.**

Project Steps

1. Application Development

- **Frontend Framework: Use ReactJS for a modular and reusable frontend.**
- **Interface Design:**
  - Develop a user-friendly layout with clear sections for movie lists, search, and filters.
  - Include responsive design for seamless usage across devices**.**

---

2. API Integration

- **API Selection:**
  - Use TMDb API (preferred) or OMDb API for reliable movie-related data.
  - Obtain API keys by signing up for the respective API platform.
- **Dynamic Data to Fetch:**
  - Movie Titles
  - Posters/Images
  - Release Dates
  - Ratings
  - Genres
  - Summaries/Overviews
- **Endpoints Example (TMDb API):**
  - Popular Movies: /movie/popular
  - Search Movies: /search/movie
  - Movie Details: /movie/{movie_id}
  - Genres List: /genre/movie/list

---

3. Key Features Implementation

1. **Movie List Display:**
    - Create a grid or list to showcase trending/popular movies.
    - Fetch data dynamically from API endpoints.
2. **Search Functionality:**
    - Add a search bar for users to search movies by title.
    - Display search results in real-time or on submit.
3. **Movie Details Page:**
    - Allow users to click on a movie to view more details on a separate page or modal.
    - Include movie descriptions, ratings, cast, trailers, and more.
4. **Filter and Sort Options:**
    - **Provide dropdowns or buttons to filter movies by:**
        - Genre (e.g., Action, Comedy, Drama)
        - Year
    - **Sort by:**
        - Popularity
        - Release Date
        - Rating

---

4. ReactJS Best Practices

- **Component-based Structure:**
    - Use reusable React components for modularity (e.g., MovieCard, SearchBar, FilterBar, etc.).
- **State Management:**
    - Use React Hooks (e.g., useState, useEffect) for state management.
    - Consider Redux or Context API for larger or more complex apps.
- **Data Fetching:**
    - Use fetch or Axios for API calls.
    - Implement loading and error handling states.

---

5. Styling the Application

- **CSS Frameworks:**
    - Use Tailwind CSS or Bootstrap for a visually appealing layout.
    - Combine with custom CSS for branding and style adjustments.
- **Responsive Design:**
    - Ensure the app is mobile-friendly using media queries or responsive utility classes in frameworks.

Main Flow Services and Technologies Pvt. Ltd.
Contact Us. +91 9389641586, +91 97736 99074
Email-Add. contact.mainflow@gmail.com
www.mainflow.in

6. Testing and Debugging

- **Test the app for:**
  - API integration issues (invalid keys, missing data).
  - UI responsiveness (desktop, tablet, mobile).
  - Functional errors (e.g., broken links, incorrect data).
- **Debug using browser developer tools and React debugging tools.**

---

7. Documentation and Deployment

1. **Documentation:**
   - **Provide a clear setup guide, including steps to:**
     - Install dependencies.
     - Configure API keys.
     - Run the development server**.**
2. **Deployment Platforms:**
   - Use Vercel, Netlify, or GitHub Pages for deployment.
   - Ensure deployment settings handle environment variables for API keys.

Deliverables

1. **Fully Functional Movie Application**
   - With all required features implemented and tested.
2. **Documentation**
   - Includes setup instructions, app features, and API usage.
3. **Deployment**
   - A live app accessible to users for testing and feedback.

Key Outcomes

- Hands-on experience in building a ReactJS application.
- Exposure to integrating external APIs and handling dynamic data.
- Familiarity with deploying React apps on modern platforms.

Main Flow Services and Technologies Pvt. Ltd.
Contact Us. +91 9389641586, +91 97736 99074
Email-Add. contact.mainflow@gmail.com
www.mainflow.in