Software Development Engineer (SDE) Intern - Frontend Development Assignment

Assignment Overview

In this task, you will develop a **"Personalized Content Dashboard"** for users to track and interact with data in an engaging, dynamic interface. The dashboard will present content such as news, recommendations, and social posts from multiple sources, and it will allow users to interact with and customize their dashboard experience. You will be challenged to use **React**, **Next.js**, **TypeScript**, **Redux Toolkit**, **API Integration**, and **Testing** to create an interactive, user-centric platform.

This project will assess your understanding of **frontend architecture, state management, API handling, complex UI designs**, and **testing best practices**.

Duration: 48 hours from assignment receipt.

Submission: Provide a GitHub repository link with a clear README file, including instructions on how to run the project. Optionally, include a live demo link.

Project Requirements

1. Core Features:

Personalized Content Feed:

- **User Preferences:** The user can configure their content preferences (e.g., favorite categories such as technology, sports, finance) from a **settings panel**. Use **local storage** or **Redux** to persist these settings.
- **Data Fetching:** Fetch personalized content from two APIs:
    - **News API (e.g., NewsAPI):** Fetch the latest news based on user preferences (categories).
    - **Recommendations API (e.g., TMDB API or Spotify API):** Show personalized movie or music recommendations based on user history or preferences.
    - **Social Media API (e.g., Twitter, Instagram):** Display posts based on specific hashtags or user profiles (this can be a mock API if necessary).

- **Interactive Content Cards:**
    - Display **cards for each content piece** (e.g., news article, recommendation) with images, headlines, brief descriptions, and call-to-action buttons like "Read More" or "Play Now".
    - Implement **infinite scrolling** or **pagination** for efficient display of content.

2. User Dashboard Layout:

- **Main Layout:** Create a **responsive dashboard layout** with a **sidebar** for navigation and a **top header** with a search bar, user settings, and account info.
- **Sections:**
    - **Personalized Feed Section:** Display the news, recommendations, and social posts in one unified feed.
    - **Trending Section:** Display top trending items based on categories (e.g., trending news, movies, posts).
    - **Favorite Section:** Users can **mark** content as favorite, and the dashboard will display it under a personalized “Favorites” section.

3. Search Functionality:

- **Search Bar:** Allow users to search for content across different categories (e.g., search for a movie, news, or social media post).
- **Debounced Search:** Implement a **debounced search functionality** to optimize performance while the user types.

4. Advanced UI/UX Features:

- **Drag-and-Drop for Content Organization:** Allow users to drag and reorder the content cards in their feed. Use **React DnD** or **Framer Motion** for smooth, interactive drag-and-drop functionality.
- **Dark Mode:** Implement **dark mode** toggle using **CSS custom properties** and **Tailwind CSS**.

- **Animations:** Include **smooth transitions** between sections, loading spinners, and card hover effects using **Framer Motion** or **CSS animations**.

5. State Management & Logic Handling:

- **Redux Toolkit:** Use **Redux Toolkit** for managing the global state (e.g., user preferences, content data).
- **Async Logic:** Use **Redux Thunks** or **RTK Query** to handle asynchronous data fetching from the APIs.
- **Local Storage:** Save the user's preferences (e.g., favorite categories, dark mode preference) in **localStorage** or **Redux Persist** to maintain session data across page reloads.

6. Testing:

- **Unit Testing:** Write **unit tests** for the main components, focusing on edge cases and logic flows (e.g., user preferences, API responses).
- **Integration Testing:** Ensure that content is rendered properly when it is fetched and handle edge cases like no content, empty states, and errors.
- **E2E Testing:** Use **Cypress** or **Playwright** to test critical user flows, such as:
    - Search functionality.
    - Drag-and-drop reordering.
    - User authentication (if applicable).

7. Bonus Features (Optional):

- **Authentication:** Implement user login/signup using **NextAuth.js** or mock authentication. Allow users to customize their profile.
- **Real-time Data:** Implement a **real-time feed** where content (e.g., social media posts or news) gets updated periodically via WebSockets or **Server-Sent Events**.
- **Multi-language Support:** Implement language switching using **react-i18next** for internationalization.

Evaluation Criteria:

- **Functionality:** Does the application meet the requirements and provide the intended user experience?
- **Code Quality:** Clean, modular, well-documented code. Proper use of React/Redux conventions.
- **UI/UX Design:** Is the design intuitive, responsive, and aesthetically pleasing? Is the UI accessible (WCAG compliance)?
- **State Management:** Is Redux used effectively for managing state across different sections? Is asynchronous logic handled well?
- **Performance:** Is the app optimized for performance? How is the data fetching optimized (debouncing, pagination)?
- **Testing:** Does the application have sufficient coverage for unit tests, integration tests, and E2E tests?
- **Creativity:** Bonus points for innovative features or designs that go above and beyond the requirements.
- **Security:** Proper handling of sensitive data (e.g., API keys, user authentication).

Submission Guidelines:

- **GitHub Repository:** Push your code to a **public GitHub repository**. Ensure the repository includes all the required files, with clear commit history. Include a detailed **README** with project setup and user flow
- **Demo Video:** Video showcasing the whole Application [**Bonus:** Explanation of the Application]
- **Live Link:** Hosted link of the application for testing and Usage

---

Additional Resources:

- React Documentation
- Next.js Documentation
- Redux Toolkit Documentation
- React Testing Library Documentation
- Tailwind CSS Documentation
- Framer Motion Documentation
- Cypress Documentation
- React-i18next Documentation

This assignment emphasizes a **complete frontend application** with an **interactive and customizable dashboard**, testing a candidate's skills in **UI/UX design**, **state management**, **API handling**, **logic handling**, and **testing**. The project also provides room for **creativity** and **problem-solving** while ensuring the candidate can work within the constraints of modern web development practices.