Frontend Assessment

This task is designed to evaluate your proficiency in React development, including your ability to set up a project with Webpack, use modern React features (hooks), and integrate popular UI libraries. The goal is to create a fully functional application with a well-structured architecture, mock API integration, and clean, maintainable code.

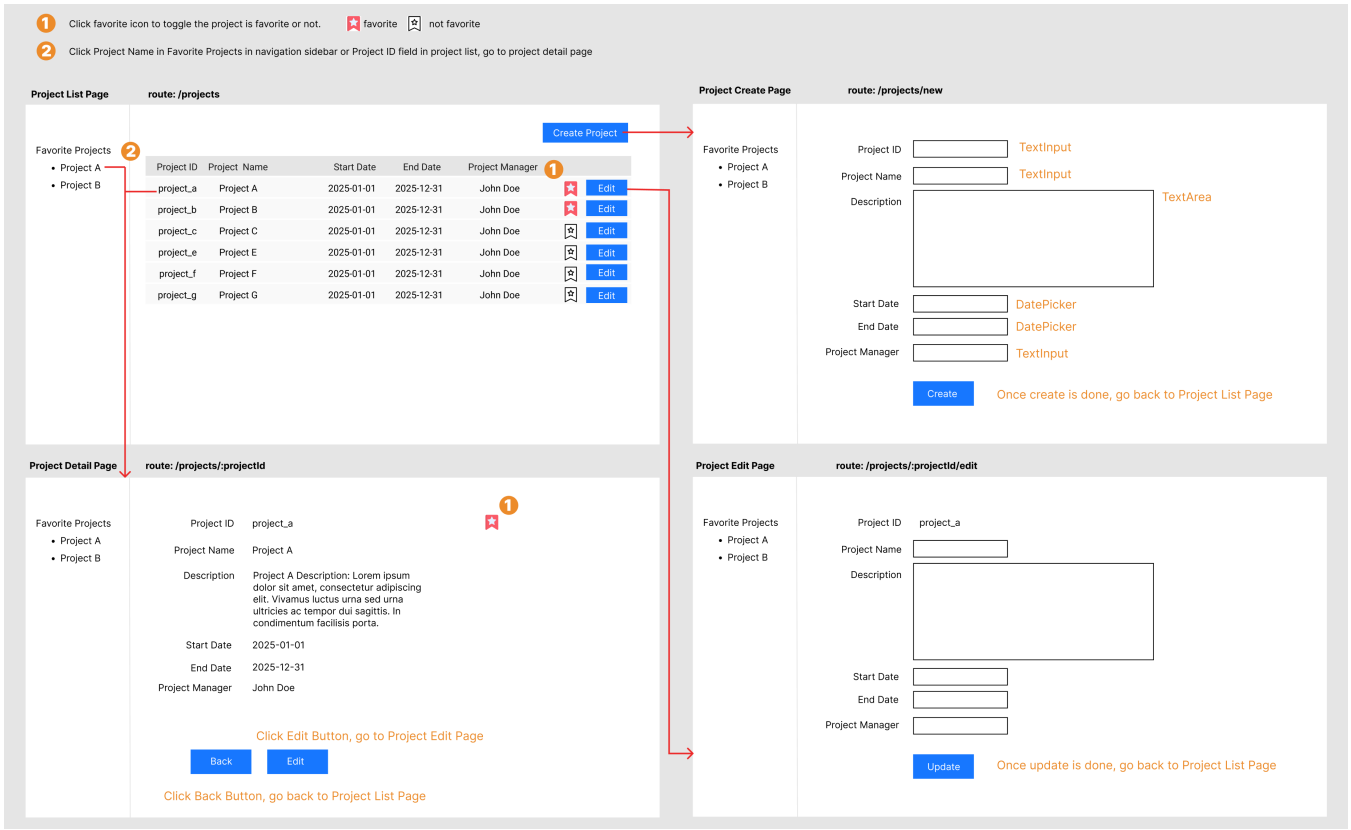
Requirements

Project Setup

1. Use **Webpack** to build the project. Use next.js even better.
2. React version: **18+**.
3. Use either **Material UI** or **Ant Design** for UI components.
4. Must implement css by using **CSS-in-JS** either **styled-components** or **Emotion**.
5. (Optional, nice-to-have) Use **Tailwind CSS** for additional styling.
6. Structure the project to allow easy scalability and maintainability.

Functional Requirements

UI Wireframe



- There are four pages: Project List, Project Create, Project Edit, Project Detail
- Global Layout
  - Left section: navigation sidebar with a menu item labeled “Favorite Projects”.
  - Right section: page content
- User interactions are described as of the wireframe
- The application is single page application.

There no test apis for this project. All of Apis required shall be mocked locally in the code. It may be done by running a simple Express server.

- When update project name of a favorite project, the project name in "Favorite Projects" shall be updated as well.
- When toggle favorite icon on Project List Page or Project Detail Page, the project should be added or removed from Favorite Projects in navigation sidebar.

It is expected to implement state synchronization of favorite and project name with a elegant solution.

Deliverable

- Screen recording of walk through all of requirements, the deliverable could be video or gif, send either video file or a public link to us
- Code is delivered via a new Github repository. Share the link to use.

Additional Expectations

- Error Handling: show appropriate error messages for API failure by randomly injecting errors in mocked api
- Show loading state on UI by increasing response time of mocked api
- Responsive Design: Ensure the app is usable on different screen sizes (mobile and desktop).
- Write unit test code for functionality and UI
- Code Quality:
  - Follow best practices, including modular component structure and meaningful variable names.
  - Ensure the project is well-documented with a README.md explaining how to run the project.
  - Clean and clear commit history and commit message