Frontend Assignment Report

Assignment Overview:

The frontend assignment involved creating an interactive Kanban board application using React JS that interacts with an external API. The application should allow users to group and sort tickets based on various criteria and maintain the user's view state even after page reloads.

Requirements:

- Create a Kanban board application using React JS.
- Retrieve data from the provided API: https://api.quicksell.co/v1/internal/frontend-assignment.
- Implement three distinct ways to group data:
    1. by status
    2. by user
    3. by priority.
- Implement two ways to sort data:
    1. by priority
    2. by title.
- Make the application responsive and visually appealing.
- Save the user's view state, including grouping and sorting preferences, even after page reloads.
- Avoid using CSS libraries like Tailwind and Next.js.

Implementations:

1. **React Components:**

The application was structured into React components to ensure modularity and maintainability. The primary components include:

- **Card :** A reusable component used to display individual tickets on the Kanban board.
- **Data.js:** File to define various maps and data structures related to statuses, users' images, priorities, and status images for the project.
- **Loader :** Added a Loader to provide visual feedback to users while data is being fetched or processed.

2. States:

Below are the states created with their functionality:

- **Loading :** Will be false when data is fetched from API else will be used for showing loading animation.
- **IsVisible :** for popup for the display button.
- **Grouping :** will store the way to group the data (default : “userId”)
- **Sorting :** will store the way to sort the data (default : “priority”)
- **Tickets :** will store the tickets data fetched from API
- **Users :** will store the users data fetched from API
- **SortedTickets:** will store the sorted and grouped tickets.
- **ColumnHeaders:** will store the headers data of each column/section based on grouping and sorting status.

3. Data Fetching:

Data was retrieved from the provided API using the fetch API. The fetched data, including ticket information and user details, was stored in state variables within the App component. The Loader will animate till the data is being fetched.

4. Grouping and Sorting:

Created functions for Sorting and Grouping the data.

- **Grouping**: Tickets were grouped based on user preferences, with grouping options including status, user, and priority. A flexible function was implemented to group tickets dynamically.
- **Sorting**: Tickets were sorted based on user preferences, with sorting options including priority and title. A sorting function was applied to arrange tickets in the desired order.

5. Responsive Design:

A responsive design approach was adopted to ensure the application's usability across various devices and screen sizes.

Designed Icons in Figma to match the requirements.

6. Page Reload:

Used **sessionStorage** for saving the user's view state (until tab closes) even after page reloads instead of saving the sorted data.

We can also use **localStorage** for the same to save even after the tab closes.

Tech Stack Used : React JS and CSS

Demo: https://kanban-board1-j2ov.onrender.com/