```
import React, { useEffect, useState } from "react";
import "./App.css";


const API_URL = "https://api.quicksell.co/v1/internal/frontend-assignment";


const App = () => {
const [tickets, setTickets] = useState([]);
const [groupedTickets, setGroupedTickets] = useState([]);
const [grouping, setGrouping] = useState("status"); // Default grouping by status
const [sortOption, setSortOption] = useState(""); // Sorting option


// Fetch data from API
useEffect(() => {
  const fetchTickets = async () => {
    try {
      const response = await fetch(API_URL);
      const data = await response.json();
      setTickets(data);
      setGroupedTickets(groupByOption(data, grouping));
    } catch (error) {
      console.error("Error fetching tickets:", error);
    }
  };


  fetchTickets();
}, []);


// Group tickets based on user selection
```

```
const groupByOption = (tickets, option) => {
  switch (option) {
    case "status":
      return groupBy(tickets, "status");
    case "user":
      return groupBy(tickets, "assignedUser");
    case "priority":
      return groupBy(tickets, "priority");
    default:
      return tickets;
  }
};


const groupBy = (array, key) => {
  return array.reduce((result, item) => {
    (result[item[key]] = result[item[key]] || []).push(item);
    return result;
  }, {});
};


// Handle grouping change
const handleGroupingChange = (event) => {
  const selectedOption = event.target.value;
  setGrouping(selectedOption);
  setGroupedTickets(groupByOption(tickets, selectedOption));
};


// Handle sorting
const handleSortChange = (event) => {
  const selectedSort = event.target.value;
```

```
    setSortOption(selectedSort);


    const sortedTickets = [...tickets].sort((a, b) => {
      if (selectedSort === "priority") {
        return b.priority - a.priority; // Descending by priority
      } else if (selectedSort === "title") {
        return a.title.localeCompare(b.title); // Ascending by title
      }
      return 0;
    });


    setGroupedTickets(groupByOption(sortedTickets, grouping));
  };


  return (
    <div className="App">
      <h1>Kanban Board</h1>


      {/* Grouping Options */}
      <div className="controls">
        <label>
          Group by:
          <select value={grouping} onChange={handleGroupingChange}>
            <option value="status">Status</option>
            <option value="user">User</option>
            <option value="priority">Priority</option>
          </select>
        </label>
        {/* Sorting Options */}
        <label>
          Sort by:
          <select value={sortOption} onChange={handleSortChange}>
            <option value="">None</option>
            <option value="priority">Priority</option>
            <option value="title">Title</option>
          </select>
        </label>
      </div>


      {/* Kanban Board */}
      <div className="kanban-board">
        {Object.entries(groupedTickets).map(([group, items]) => (
          <div key={group} className="column">
            <h2>{group}</h2>
            {items.map((ticket) => (
              <div key={ticket.id} className="card">
                <h3>{ticket.title}</h3>
                <p>Priority: {ticket.priority}</p>
                <p>User: {ticket.assignedUser}</p>
                <p>Status: {ticket.status}</p>
              </div>
            ))}
          </div>
        ))}
      </div>
    </div>
  );
  };
```

```
export default App;
```