Components

```
import React, { useState, useEffect } from 'react';

import axios from 'axios';


// KanbanBoard component

const KanbanBoard = () => {

  const [tickets, setTickets] = useState([]);

  const [grouping, setGrouping] = useState('status');

  const [sorting, setSorting] = useState('none');


  useEffect(() => {

    const fetchData = async () => {

      try {

        const response = await axios.get('https://api.quicksell.co/v1/internal/frontend-assignment');

        setTickets(response.data);

      } catch (error) {

        console.error('Error fetching data:', error);

      }

    };


    fetchData();

  }, []);


  const handleGroupingChange = (e) => {

    setGrouping(e.target.value);
  };

  const handleSortingChange = (e) => {
    setSorting(e.target.value);
  };

  const groupedTickets = groupTickets(tickets, grouping);
  const sortedTickets = sortTickets(groupedTickets, sorting);

  return (
    <div className="kanban-board">
      <DisplayOptions
        grouping={grouping}
        sorting={sorting}
        onGroupingChange={handleGroupingChange}
        onSortingChange={handleSortingChange}
      />
      {sortedTickets.map((group) => (
        <Column key={group.key} group={group} />
      ))}
    </div>
  );
};

// Column component
```

```
const Column = ({ group }) => {
  return (
    <div className="column">
      <h3>{group.key}</h3>
      {group.tickets.map((ticket) => (
        <Ticket key={ticket.id} ticket={ticket} />
      ))}
    </div>
  );
};

// Ticket component
const Ticket = ({ ticket }) => {
  return (
    <div className="ticket">
      <h4>{ticket.title}</h4>
      <p>{ticket.description}</p>
      {/* Add more ticket details as needed */}
    </div>
  );
};

// DisplayOptions component
const DisplayOptions = ({ grouping, sorting, onGroupingChange, onSortingChange }) => {
  return (
    <div className="display-options">
      <select value={grouping} onChange={onGroupingChange}>
        <option value="status">Status</option>
        <option value="user">User</option>
        <option value="priority">Priority</option>
      </select>
      <select value={sorting} onChange={onSortingChange}>
        <option value="none">None</option>
        <option value="priority-desc">Priority (Descending)</option>
        <option value="title-asc">Title (Ascending)</option>
      </select>
    </div>
  );
};

export default KanbanBoard;
```

css

```
.kanban-board {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
}

.column {
  border: 1px solid #ccc;
  padding: 10px;
  width: 300px;
  min-height: 300px;
}

.ticket {
  border: 1px solid #ccc;
  padding: 10px;
  margin-bottom: 10px;
}

.display-options {
  display: flex;
  gap: 10px;
  margin-bottom: 20px;
}
const groupTickets = (tickets, grouping) => {
  if (grouping === 'status') {
    return groupBy(tickets, 'status');
  } else if (grouping === 'user') {
    return groupBy(tickets, 'assignedUser');
  } else if (grouping === 'priority') {
    return groupBy(tickets, 'priority');
  }
  return tickets;
};

const sortTickets = (tickets, sorting) => {
  if (sorting === 'priority-desc') {
    return tickets.sort((a, b) => b.priority - a.priority);
  } else if (sorting === 'title-asc') {
    return tickets.sort((a, b) => a.title.localeCompare(b.title));
  }
  return tickets;
};

const groupBy = (array, key) => {
  return array.reduce((groups, item) => {
    const groupKey = item[key];
    if (!groups[groupKey]) {
      groups[groupKey] = [];
    }
    groups[groupKey].push(item);
    return groups;
  }, {});
};
```

JavaScript

```
const groupTickets = (tickets, grouping) => {
  if (grouping === 'status') {
    return groupBy(tickets, 'status');
  } else if (grouping === 'user') {
    return groupBy(tickets, 'assignedUser');
  } else if (grouping === 'priority') {
    return groupBy(tickets, 'priority');
  }
  return tickets;
};

const sortTickets = (tickets, sorting) => {
  if (sorting === 'priority-desc') {
    return tickets.sort((a, b) => b.priority - a.priority);
  } else if (sorting === 'title-asc') {
    return tickets.sort((a, b) => a.title.localeCompare(b.title));
  }
  return tickets;
};

const groupBy = (array, key) => {
  return array.reduce((groups, item) => {
    const groupKey = item[key];
    if (!groups[groupKey]) {
      groups[groupKey] = [];
    }
    groups[groupKey].push(item);
    return groups;
  }, {});
};
```