Q1. Assignment- ReactJS:

To build a Kanban board in ReactJS that meets the requirements, you'll need to implement several features, including task management, drag-and-drop functionality, search, and adding new tasks. Below is a basic outline of how to implement this using ReactJS, along with some key libraries:

1. Project Setup

First, ensure you have Node.js and npm installed. Then, create a new React project:

```bash
npx create-react-app kanban-board
cd kanban-board
```

2. Install Required Packages

Install the necessary packages for drag-and-drop and state management:

```bash
npm install react-beautiful-dnd uuid
```

react-beautiful-dnd: For drag-and-drop functionality.

uuid: To generate unique IDs for tasks.

3. Create the Kanban Board Layout

Start by setting up the Kanban board with four columns and basic structure:

```jsx
// src/App.js
```

```
import React, { useState } from 'react';
import { DragDropContext, Droppable, Draggable } from 'react-beautiful-dnd';
import { v4 as uuidv4 } from 'uuid';

const initialTasks = {
  'to-do': {
    name: 'To Do',
    items: [],
  },
  'in-progress': {
    name: 'In Progress',
    items: [],
  },
  'peer-review': {
    name: 'Peer Review',
    items: [],
  },
  'done': {
    name: 'Done',
    items: [],
  },
};

function App() {
  const [tasks, setTasks] = useState(initialTasks);
  const [taskTitle, setTaskTitle] = useState('');
  const [taskDescription, setTaskDescription] = useState('');
  const [searchTerm, setSearchTerm] = useState('');
```

```
const onDragEnd = (result) => {
  if (!result.destination) return;

  const { source, destination } = result;
  const sourceColumn = tasks[source.droppableId];
  const destColumn = tasks[destination.droppableId];
  const sourceItems = [...sourceColumn.items];
  const destItems = [...destColumn.items];

  const [removed] = sourceItems.splice(source.index, 1);
  destItems.splice(destination.index, 0, removed);

  setTasks({
    ...tasks,
    [source.droppableId]: {
      ...sourceColumn,
      items: sourceItems,
    },
    [destination.droppableId]: {
      ...destColumn,
      items: destItems,
    },
  });
};

const addTask = () => {
  if (!taskTitle.trim()) return;

  const newTask = {
```

```
    id: uuidv4(),
    title: taskTitle,
    description: taskDescription,
  };

  setTasks((prev) => ({
    ...prev,
    'to-do': {
      ...prev['to-do'],
      items: [newTask, ...prev['to-do'].items],
    },
  }));

  setTaskTitle('');
  setTaskDescription('');
};

const filteredTasks = (items) => {
  return items.filter((item) =>
    item.title.toLowerCase().includes(searchTerm.toLowerCase())
  );
};

return (
  <div style={{ padding: '50px' }}>
    <h2>Kanban Board</h2>
    <div>
      <input
        type="text"
```

```
    placeholder="Search tasks..."
    value={searchTerm}
    onChange={(e) => setSearchTerm(e.target.value)}
   />
  </div>
  <br />
  <div>
   <input
    type="text"
    placeholder="Task Title"
    value={taskTitle}
    onChange={(e) => setTaskTitle(e.target.value)}
   />
   <input
    type="text"
    placeholder="Task Description"
    value={taskDescription}
    onChange={(e) => setTaskDescription(e.target.value)}
   />
   <button onClick={addTask}>Add Task</button>
  </div>
  <br />
  <div style={{ display: 'flex', justifyContent: 'space-between' }}>
   <DragDropContext onDragEnd={onDragEnd}>
    {Object.entries(tasks).map(([columnId, column], index) => (
     <Droppable key={columnId} droppableId={columnId}>
      {(provided, snapshot) => (
       <div
        {...provided.droppableProps}
```

```
 ref={provided.innerRef}
 style={{
  border: '1px solid lightgrey',
  borderRadius: '4px',
  width: '20%',
  minHeight: '500px',
  padding: '10px',
  backgroundColor: snapshot.isDraggingOver ? 'lightblue' : 'lightgrey',
 }}
>
 <h3>{column.name}</h3>
 {filteredTasks(column.items).map((item, index) => (
  <Draggable key={item.id} draggableId={item.id} index={index}>
   {(provided, snapshot) => (
    <div
     ref={provided.innerRef}
     {...provided.draggableProps}
     {...provided.dragHandleProps}
     style={{
      userSelect: 'none',
      padding: '16px',
      margin: '0 0 8px 0',
      minHeight: '50px',
      backgroundColor: snapshot.isDragging ? '#263B4A' : '#456C86',
      color: 'white',
      ...provided.draggableProps.style,
     }}
    >
     <h4>{item.title}</h4>
```

```
                    <p>{item.description.substring(0, 50)}</p>
                  </div>
                )}
              </Draggable>
            ))}
            {provided.placeholder}
          </div>
        )}
      </Droppable>
    ))}
  </DragDropContext>
 </div>
</div>
);
}

export default App;
```

4. Explanation

State Management: `useState` is used to manage the state of the tasks, the input fields for new tasks, and the search term.

Drag-and-Drop: `react-beautiful-dnd` is used to implement drag-and-drop functionality. The `onDragEnd` function is responsible for handling the drop logic.

Adding New Tasks: A new task can be added to the "To Do" column by entering a title and description, then clicking the "Add Task" button.

Search Functionality: The search input filters tasks across all columns based on the title.

5. Styling and Responsiveness

For simplicity, inline styles have been used. However, for a more scalable and maintainable project, you might want to use a CSS framework like `Material-UI` or `styled-components` for better styling and responsiveness.

6. Final Touches

State Persistence: To persist tasks between page reloads, consider using `localStorage` or integrating with an API backend.

Responsive Design: Ensure the Kanban board is responsive and adapts to different screen sizes.

7. Running the Project

To run the project, use the following command:

```bash
npm start
```