Step-by-Step Guide for Building a Task Management System

1. Define Your Goal

Understand the core functionalities of the application:

- Add a task to the "Pending" list.
- Move tasks between categories: "Pending," "In Progress," and "Completed."
- Remove tasks from any category.
- Render tasks dynamically in their respective categories.

---

2. Plan the Structure

Break the application into smaller components:

1. **Data Storage**: Use an object to categorize tasks (e.g., `pending`, `inProgress`, `completed`).
2. **Methods**: Define functions for adding, moving, removing, and rendering tasks.
3. **Event Handling**: Handle user interactions like form submissions and button clicks.

---

3. Start Writing Code

Begin with the basics and build incrementally:

A. Set Up a Task Manager Object

Create a JavaScript object (`taskManager`) to manage tasks and their associated actions:

- Properties: `tasks` to store tasks in categories.
- Methods:
    - `addTask()`
    - `moveTask()`
    - `removeTask()`
    - `renderTasks()`

B. Implement Core Methods

Write the methods in a modular way:

- `addTask`: Add a new task to the "Pending" category with details like name, ID, and date.

- `moveTask`: Move tasks between categories and update their `completed` status if moved to "Completed."
- `removeTask`: Remove a task from a specific category by filtering it out.
- `renderTasks`: Dynamically update the DOM to display tasks in their respective categories.

C. Attach Event Listeners

Handle the interactions:

- **Form Submission**: Use `addEventListener` on the form to call `addTask` and re-render tasks.
- **Move and Remove Buttons**: Dynamically generate buttons for these actions and attach event handlers.

D. Test and Debug

- Test each method individually in the browser console.
- Validate inputs for edge cases (e.g., empty task names).

---

4. Order of Implementation

Follow this sequence:

Initialize the Task Manager Object:

```
const taskManager = { tasks: { pending: [], inProgress: [], completed: [] } };
```

1.
2. **Add Core Methods to the Object**:
    - `addTask()`
    - `moveTask()`
    - `removeTask()`
    - `renderTasks()`
3. **Create Event Handlers**:
    - Write `addTask` and attach it to the form's `submit` event.
    - Write `moveTask` and `removeTask` functionalities, and attach them to buttons dynamically generated for each task.
4. **DOM Manipulation**:
    - Use `document.createElement` and `appendChild` to render tasks.

    - Update the content dynamically for actions like moving or removing tasks.
5. **Test the Flow**:
    - Add tasks.
    - Move tasks between categories.
    - Remove tasks.
    - Ensure the DOM updates correctly.

---

5. Add Features Incrementally

Once the basics work, enhance the application:

- **Task Details**: Include additional fields like priority or due date for each task.
- **Styling**: Apply CSS classes for better UI and user experience.
- **Validation**: Ensure task names are not empty before adding.

---

6. Checklist for Completion

- Tasks are added to the "Pending" category correctly with unique IDs.
- Tasks can be moved between categories.
- Tasks can be removed from any category.
- The application handles empty inputs gracefully.
- Tasks render dynamically, including updates for all actions.

By following this roadmap, you will systematically build the Task Management System with clarity and focus.