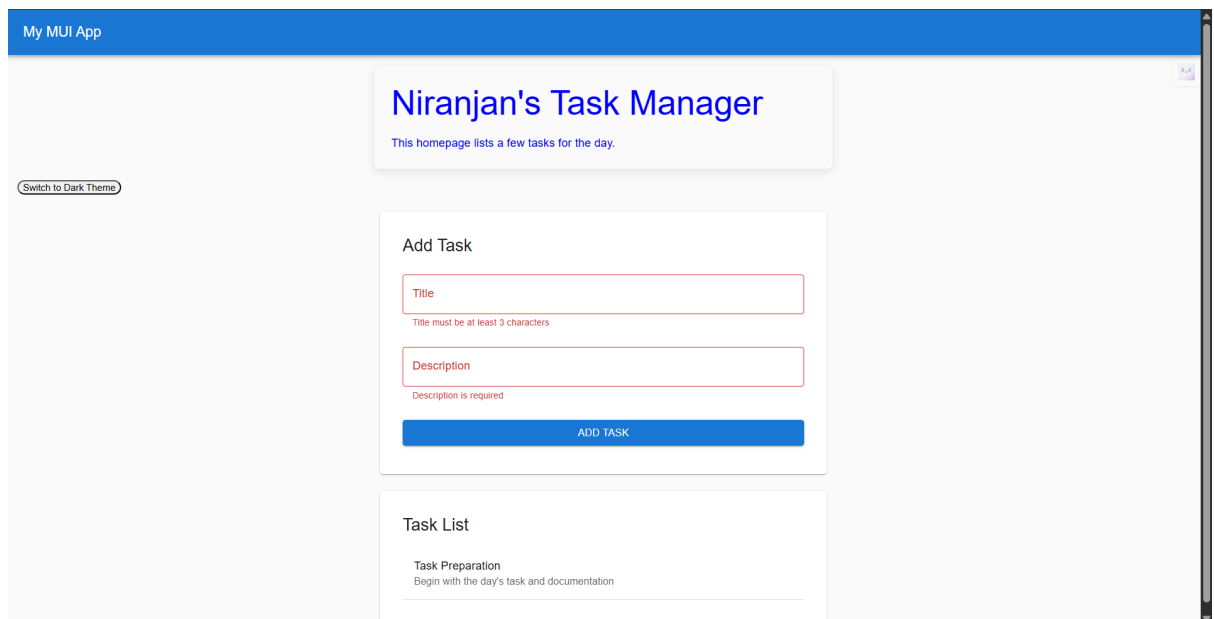


AeroAspire -SDE Intern Training

NIRANJAN C N
Week2-Day 3-Oct1

The image displays two screenshots of a web application titled "Niranjan's Task Manager". The top screenshot shows the application in a dark theme, with a blue header bar labeled "My MUI App". The main content area has a dark background. A white card at the top contains the title "Niranjan's Task Manager" and a subtitle "This homepage lists a few tasks for the day.". Below this is a "Switch to Light Theme" button. The "Add Task" section features two input fields: "Title" (containing "Task Preparation") and "Description" (containing "Begin with the day's task and documentation"), followed by a blue "ADD TASK" button. The "Task List" section below it shows "No tasks yet". The bottom screenshot shows the same application in a light theme, with a light gray background. The "Switch to Dark Theme" button is visible. The "Add Task" and "Task List" sections maintain the same structure and content as the dark theme version.

- Created a new page TaskForm.jsx for the section to add tasks
- Removed the demo values
- The fields check for the criteria for the task input
- Used onChange and onSubmit to check for input criteria
- Prevented reload using preventDefault()
- Added a border radius to make the button a bit rounded
- Further I am thinking to add a pop for errors instead of just displaying them in the input field itself.



Questions

1. Walk through flow: user types → state updates → component re-renders → effect runs (if any).
 - a. When an input is received from the user a event is triggered
 - b. It updates the component using useState
 - c. If there is any changes or updates react re-renders the dev server
 - d. In case there is any useEffect related to the changed states, the effect runs after the re-render
2. How useEffect works: dependencies, cleanup, initial render.
 - a. The useEffect works once the re-render has taken place .
 - b. It happens like this after every render
 - c. We can pass a dependency array after it re-runs for controlling
 - d. Meaning If no array: run after every render, if empty array, run only once after first render
 - e. When with dependencies, run after every change
 - f. If effect returns a function React calls it before re-running the effect or **when the component unmounts.**
3. What pitfalls exist (e.g. stale closures, infinite loops)?
 - a. Stale closures-
 - i. If the useEffect uses old values from props we might get old reference of the effect due to the JavaScript closures
 - ii. We can overcome this by including all used props in the dependency array
 - b. Infinite loops-
 - i. If the effect updates a state that is also in its dependency array, the effect will re render for each state change
 - ii. We can overcome this by updating state only when required
4. What is a controlled vs uncontrolled component?

- a. Controlled- The inputs in forms are linked to states, and updated by events and data received from props
 - b. Uncontrolled- The data is merged with the DOM and accessed from refs.
- 5. Describe event handling in forms (onChange, onSubmit).
 - a. onChange- Interprets the input received on each stage
 - b. onSubmit- This function is called only when the form is submitted.
 - c. This difference between both functions allows validation and control of data at all stages
- 6. How does MUI help: theming, form helpers, error display?
 - a. MUI helps in a uniform theme all around the webpage
 - b. All the components share the same css
 - c. Form layouts are easier using MUI's box and grid
 - d. The TextField provides errors and the helperText helps with the validation of errors part
 - e. MUI helps in handling the errors in inputs