React-Understanding and Using Props and State

IIHT

Time To Complete: 10 to 12 hr

CONTENTS

1 Project Abstract

In React development, managing dynamic data and building reusable components are core principles. This project focuses on understanding how to use **props** and **state** to manage and pass data between components. Participants will create a task list application where each task is rendered using a child component and receives its data through props. The main component manages the task list state, reinforcing concepts of data flow and reactivity in React applications.

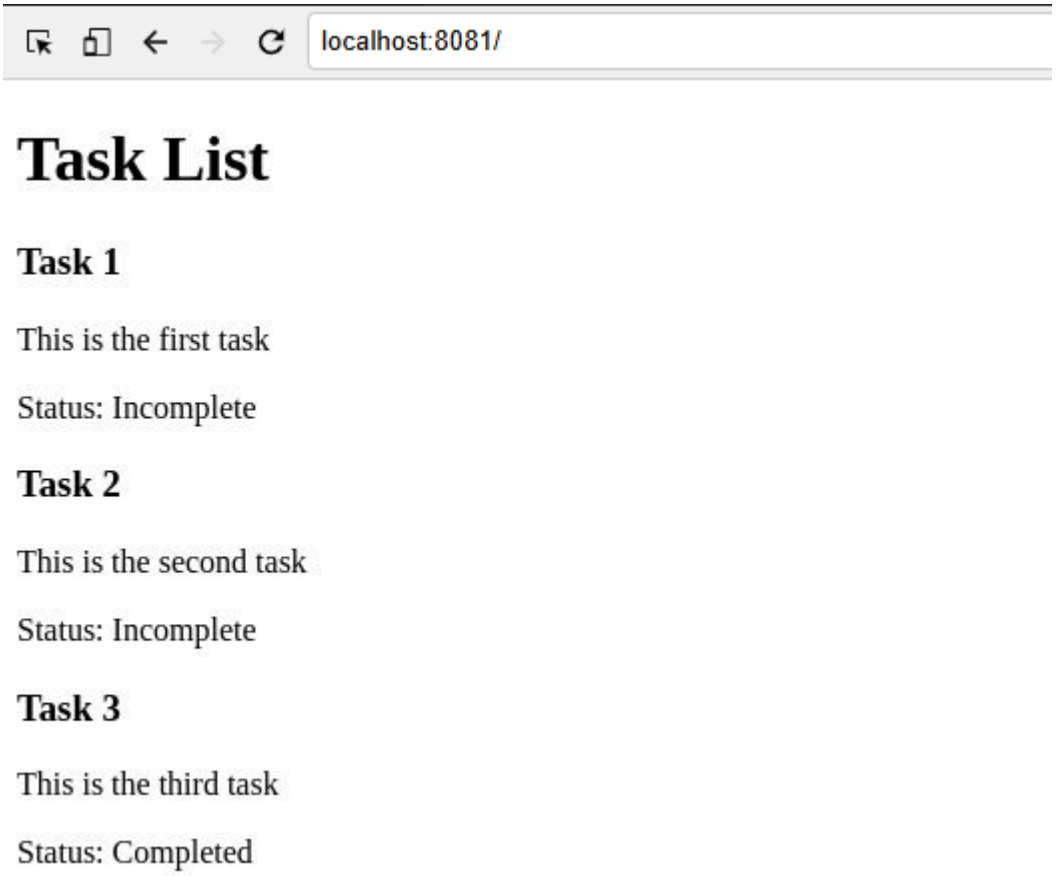
2 Problem Statement

You are required to develop a basic Task List application using React. The application should manage an array of task objects using the useState hook in the main App component. Each task should be displayed through a reusable Task component, receiving individual task data via props. The Task component will be responsible for rendering the task name, description, and its completion status. This project will help reinforce the understanding of **stateful parent components** and **stateless child components** using props.

3 Proposed Understanding and Using Props and State Application Wireframe

UI needs improvisation and modification as per given use case and to make test cases passed.

3.1 ScreenShots



4 BUSINESS-REQUIREMENT:

As an application developer, develop the Understanding and Using Props and State (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Welcome Page	As a user I should be able to visit the welcome page as the default page. Acceptance criteria: 1. Display a heading **"Task List"**. 2. Use state to manage a list of tasks, each with: ● Name ● Description ● Completion status 3. Dynamically render each task using the **Task component**.

		4. In the `Task` component: ● Display task name and description. ● Show **status** as either `"Completed"` or `"Incomplete"` based on the `completed` flag. **State & Props Overview:** **AppComponent:** **State Used:** 1. `tasks`: An array of task objects. ● Each object contains: ➔ `name` (string) ➔ `description` (string) ➔ `completed` (boolean) **Responsibilities:** 2. Holds the list of tasks. 3. Maps through the `tasks` array and renders a `<Task />` component for each. 4. Passes each task as a **prop** to the `Task` component. **TaskComponent:** **Props Received:** 1. `task`: An object containing: ● `name` ● `description` ● `completed` **Responsibilities:** 1. Display the `name` and `description` of the task. 2. Conditionally render the `status`: ● `"Completed"` if `completed` is `true` ● `"Incomplete"` if `false` **HTML Structure:** 1. **App Component**: ● Top-level `<div>` with:

		➔ Heading (<h1>) — “Task List” ➔ A list of rendered <Task /> components. 2. **Task Component:** ● Inside a : ➔ Heading (<h3>) for task name ➔ Paragraph (<p>) for task description ➔ Paragraph (<p>) showing **Status: Completed / Incomplete** **** Kindly refer to the screenshots for any clarifications. ****

5 CONSTRAINTS

1. You should be able to press the “TAB” key and “SHIFT + TAB” to navigate from top field to bottom field and vice-versa.

6 MANDATORY ASSESSMENT GUIDELINES

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.**
3. **This editor Auto Saves the code.**
4. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**
5. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
6. **This is a web-based application, to run the application on a browser, use the**

internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.

Note: The application will not run in the local browser

7. You can follow series of command to setup React environment once you are in your project-name folder:
    a. npm install -> Will install all dependencies -> takes 10 to 15 min.
    b. npm run start -> To compile and deploy the project in browser. You can press the <Ctrl> key while clicking on localhost:4200 to open the project in the browser -> takes 2 to 3 min.
    c. npm run jest -> to run all test cases and see the summary.
    d. npm run test -> to run all test cases and register results on the server. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min.
8. Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on "Submit Assessment" after you are done with code.