

| Title: Implementation of React Hooks. |
| --- |

**AIM:** To Implement the React Hooks

**Problem Definition:**

To demonstrate the working of react hooks based on the following points and Apply this on assigned programming task

* useState
* useEffect
* useContext
* **useReducer**:
* useCallback
* useMemo

\*(Students have to perform the task assigned within group and demonstrate the same).

**Resources used:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO 1:**.Build full stack applications in JavaScript using the MERN technologies.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Shelly Powers Learning Node O’ Reilly 2 nd Edition, 2016.

**Pre Lab/ Prior Concepts:**

**Write details about the following content**

* useState
* useEffect
* useContext
* **useReducer**:
* useCallback
* useMemo

**Implementation Details:**

**Task 1:**

**User Profile Editor**

**1. Create a form with fields for the user's name, age, and email.**

**2. Display the entered information below the form.**

**3. Add a button to toggle the visibility of the form.**

**4. Add a button to reset the form fields to their initial values.**

**Code:-**

import React, { useState } from 'react';

import './UserProfileEditor.css';

function UserProfileEditor() {

const [formData, setFormData] = useState({ name: '', age: '', email: '' });

const [isVisible, setIsVisible] = useState(true);

const handleChange = (e) => {

const { name, value } = e.target;

setFormData({ ...formData, [name]: value });

};

const resetForm = () => {

setFormData({ name: '', age: '', email: '' });

};

return (

<div className="user-profile-editor">

<button className="toggle-button" onClick={() => setIsVisible(!isVisible)}>

{isVisible ? 'Hide Form' : 'Show Form'}

</button>

{isVisible && (

<div className="form-container">

<h2>User Profile Editor</h2>

<input type="text" name="name" value={formData.name} onChange={handleChange} placeholder="Name" />

<input type="number" name="age" value={formData.age} onChange={handleChange} placeholder="Age" />

<input type="email" name="email" value={formData.email} onChange={handleChange} placeholder="Email" />

<button className="reset-button" onClick={resetForm}>Reset</button>

<div className="display-info">

<h3>Entered Information:</h3>

<p>Name: {formData.name}</p>

<p>Age: {formData.age}</p>

<p>Email: {formData.email}</p>

</div>

</div>

)}

</div>

);

}

export default UserProfileEditor;

css:

.user-profile-editor {

max-width: 400px;

margin: auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

background-color: #f9f9f9;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

}

.toggle-button,

.reset-button {

background-color: #007bff;

color: white;

border: none;

padding: 10px;

margin: 10px 0;

cursor: pointer;

border-radius: 5px;

}

.toggle-button:hover,

.reset-button:hover {

background-color: #0056b3;

}

.form-container {

display: flex;

flex-direction: column;

}

input {

padding: 10px;

margin: 5px 0;

border: 1px solid #ccc;

border-radius: 5px;

}

.display-info {

margin-top: 20px;

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

background-color: #e9ecef;

}

**Task 2:- Task Manager**

**Create a task manager with the following features:**

* + **An input field to add a new task.**
  + **A list to display added tasks.**
  + **A button to mark tasks as complete.**
  + **A button to toggle the visibility of completed tasks.**
  + **A button to reset the task list.**

**Code:-**

**import React, { useState } from 'react';**

**import './TaskManager.css';**

**function TaskManager() {**

**const [task, setTask] = useState('');**

**const [tasks, setTasks] = useState([]);**

**const [showCompleted, setShowCompleted] = useState(true);**

**const addTask = () => {**

**if (task) {**

**setTasks([...tasks, { text: task, completed: false }]);**

**setTask('');**

**}**

**};**

**const toggleTaskCompletion = (index) => {**

**const newTasks = [...tasks];**

**newTasks[index].completed = !newTasks[index].completed;**

**setTasks(newTasks);**

**};**

**const resetTasks = () => {**

**setTasks([]);**

**};**

**return (**

**<div className="task-manager">**

**<h2>Task Manager</h2>**

**<input type="text" value={task} onChange={(e) => setTask(e.target.value)} placeholder="Add a new task" />**

**<button onClick={addTask}>Add Task</button>**

**<button onClick={() => setShowCompleted(!showCompleted)}>**

**{showCompleted ? 'Hide Completed Tasks' : 'Show Completed Tasks'}**

**</button>**

**<button onClick={resetTasks}>Reset Tasks</button>**

**<ul>**

**{tasks.map((t, index) => (**

**<li key={index} className={t.completed ? 'completed' : ''}>**

**<span onClick={() => toggleTaskCompletion(index)}>{t.text}</span>**

**</li>**

**))}**

**</ul>**

**</div>**

**);**

**}**

**export default TaskManager;**

**css:**

**.task-manager {**

**max-width: 400px;**

**margin: auto;**

**padding: 20px;**

**border: 1px solid #ccc;**

**border-radius: 5px;**

**background-color: #f9f9f9;**

**box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);**

**}**

**input {**

**padding: 10px;**

**margin: 5px 0;**

**border: 1px solid #ccc;**

**border-radius: 5px;**

**}**

**button {**

**background-color: #007bff;**

**color: white;**

**border: none;**

**padding: 10px;**

**margin: 10px 0;**

**cursor: pointer;**

**border-radius: 5px;**

**}**

**button:hover {**

**background-color: #0056b3;**

**}**

**ul {**

**list-style-type: none;**

**padding: 0;**

**}**

**li {**

**padding: 5px;**

**cursor: pointer;**

**}**

**li.completed {**

**text-decoration: line-through;**

**color: gray;**

**}**

**Task 3:-**

**User Profile Manager**

1. **Create a form with fields for the user's name, age, and email.**
2. **Fetch initial profile data when the component mounts.**
3. **Display the fetched profile data in the form fields.**
4. **Allow the user to update their profile information.**
5. **Display the updated profile information below the form.**
6. **Log a message to the console whenever the profile data is updated.**

**Code:-**

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

**import './UserProfileManager.css';**

**function UserProfileManager() {**

**const [profile, setProfile] = useState({ name: '', age: '', email: '' });**

**useEffect(() => {**

**const fetchProfileData = () => {**

**const savedProfile = { name: 'John Doe', age: 30, email: 'john.doe@example.com' };**

**setProfile(savedProfile);**

**};**

**fetchProfileData();**

**}, []);**

**const handleChange = (e) => {**

**const { name, value } = e.target;**

**setProfile({ ...profile, [name]: value });**

**};**

**const handleUpdate = () => {**

**console.log('Profile updated:', profile);**

**};**

**return (**

**<div className="user-profile-manager">**

**<h2>User Profile Manager</h2>**

**<input type="text" name="name" value={profile.name} onChange={handleChange} placeholder="Name" />**

**<input type="number" name="age" value={profile.age} onChange={handleChange} placeholder="Age" />**

**<input type="email" name="email" value={profile.email} onChange={handleChange} placeholder="Email" />**

**<button onClick={handleUpdate}>Update Profile</button>**

**<h3>Updated Profile</h3>**

**<div className="profile-info">**

**<p>Name: {profile.name}</p>**

**<p>Age: {profile.age}</p>**

**<p>Email: {profile.email}</p>**

**</div>**

**</div>**

**);**

**}**

**export default UserProfileManager;**

**css:**

**.user-profile-manager {**

**max-width: 400px;**

**margin: auto;**

**padding: 20px;**

**border: 1px solid #ccc;**

**border-radius: 5px;**

**background-color: #f9f9f9;**

**box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);**

**}**

**input {**

**padding: 10px;**

**margin: 5px 0;**

**border: 1px solid #ccc;**

**border-radius: 5px;**

**}**

**button {**

**background-color: #007bff;**

**color: white;**

**border: none;**

**padding: 10px;**

**margin: 10px 0;**

**cursor: pointer;**

**border-radius: 5px;**

**}**

**button:hover {**

**background-color: #0056b3;**

**}**

**.profile-info {**

**margin-top: 20px;**

**padding: 10px;**

**border: 1px solid #ccc;**

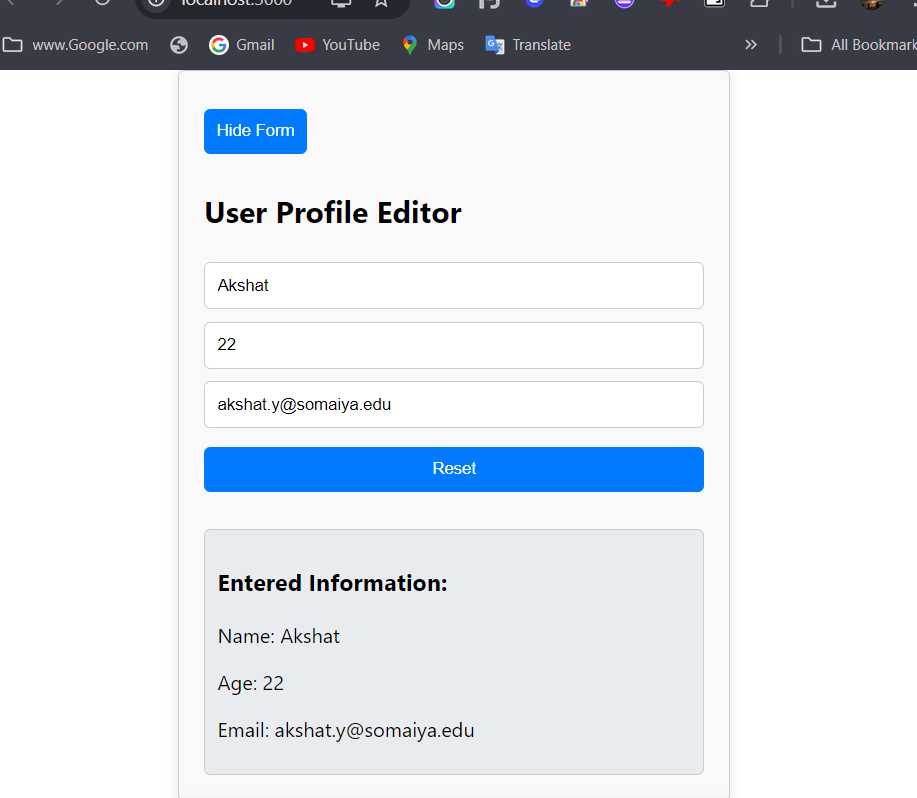
**border-radius: 5px;**

**background-color: #e9ecef;**

**}**

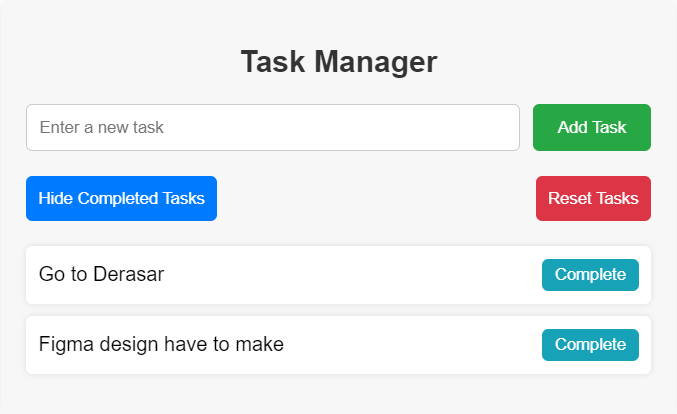
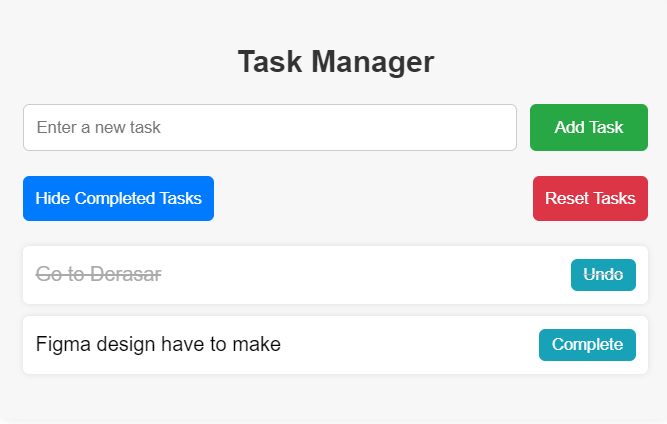
**Outputs:-**

**Task 1 :-**

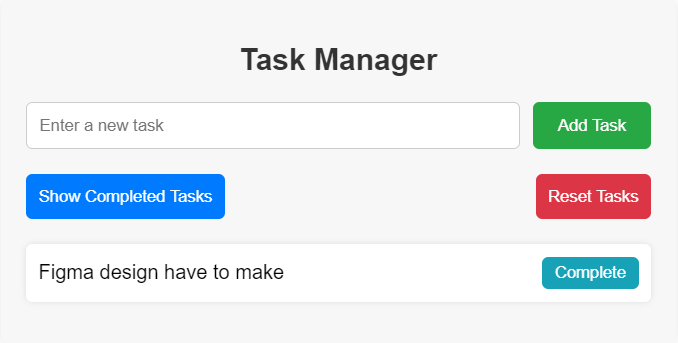
****

**Task 2:-**

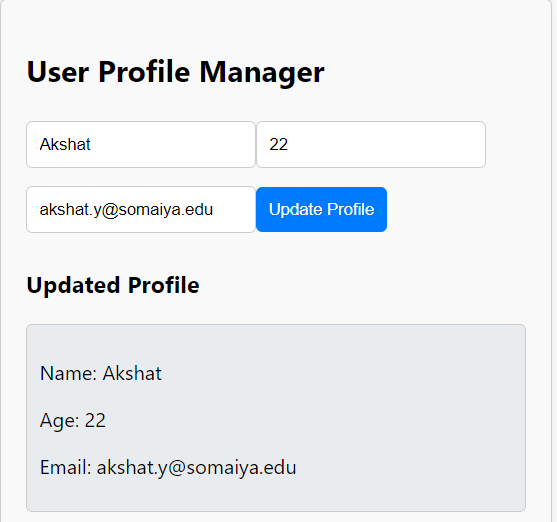
**After adding task :- After Completion :-**

****

**After hiding Completed task:-**

****

**Task 3 :-**

****

**Steps for execution:**

**For Task 1 :-**

* **Add the Component:** In the src folder of your project, create a new file for the UserProfileEditor component and paste the provided code into it.
* **Modify the Main Application:** Update the main application file to import and render the UserProfileEditor component, ensuring it is displayed when the app runs.
* **Run the Application:** Start the React development server using npm. This will automatically open the application in your default web browser.

**For Task 2 :-**

* **Add the Component:** In the src folder of your project, create a new file for the TaskManager component and paste the provided code into it.
* **Update the Main Application:** Modify the main application file to import and render the TaskManager component to ensure it is displayed when the app runs.
* **Run the Application:** Start the React development server using npm. This will launch the application in your default web browser.

**For Task 3 :-**

* **Add the UserProfileManager Component:** Inside the src folder of your project, create a new file and paste the provided code for the UserProfileManager component.
* **Modify the Main Application File:** Update the main application file to import and render the UserProfileManager component, ensuring it is displayed when the app runs.
* **Run the React Application:** Start the development server using npm, which will open the application in your default web browser.

**Conclusion:**

Hence we are able to implement React Fundamentals Hooks in three area in building User Profile Editor ,

Task Manager and User Profile Manager