

| Title: Demonstrate React Hooks |
| --- |

**AIM:** To Implement the React Axios

**Problem Definition:**

### Task: 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.

### Task: Task Manager

1. 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.

### Task: 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.

**Resources used:**

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

**Expected OUTCOME of Experiment:**

**CO 2:**. Illustrate the concepts of various front-end, back-end web application development technologies & frameworks using different web development tools.

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

**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**

* Functional component , Class Component and React Life cycle

Functional Components are simple JavaScript functions that return React elements. They are stateless by default, but with the introduction of Hooks in React 16.8, they can now manage state and side effects. This makes functional components more versatile and often preferred for their simplicity and ease of use.

Example:

import React, { useState } from 'react';

function Counter() {

const [count, setCount] = useState(0);

return (

<div>

<p>You clicked {count} times</p>

<button onClick={() => setCount(count + 1)}>Click me</button>

</div>

);

}

Class Components are ES6 classes that extend React.Component. They are more complex and include built-in support for state management and lifecycle methods. These components are ideal for situations requiring more detailed lifecycle control.

Example:

import React, { Component } from 'react';

class Counter extends Component {

constructor(props) {

super(props);

this.state = { count: 0 };

}

increment = () => {

this.setState({ count: this.state.count + 1 });

}

render() {

return (

<div>

<p>You clicked {this.state.count} times</p>

<button onClick={this.increment}>Click me</button>

</div>

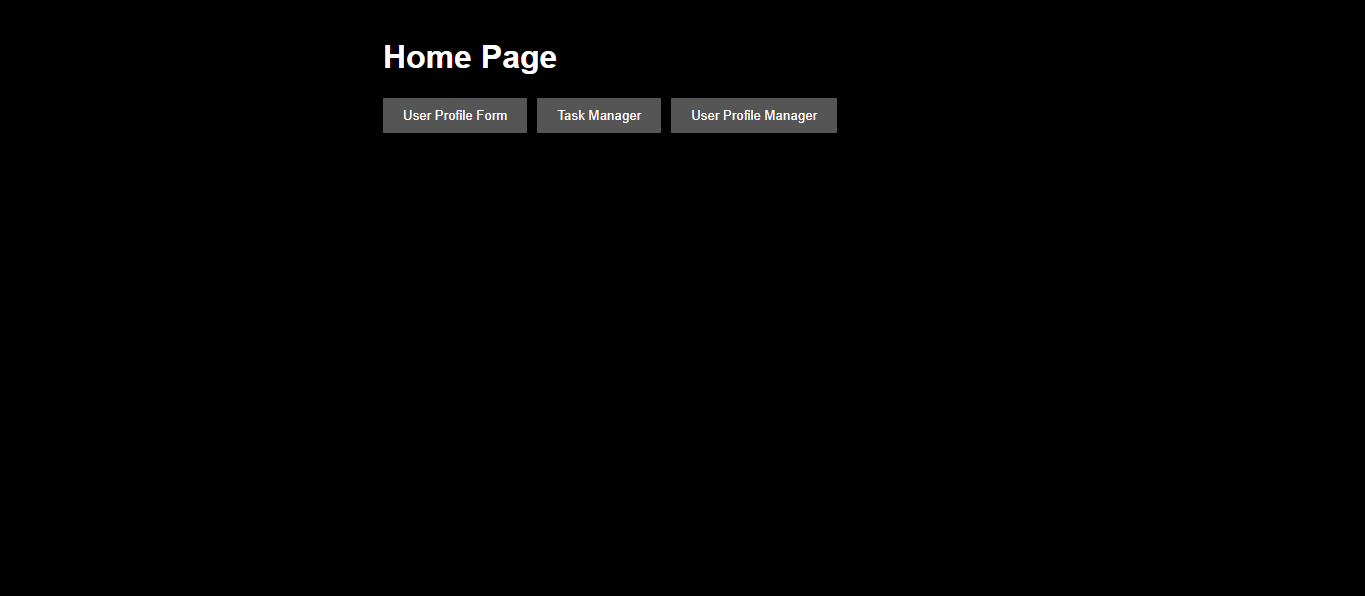
);

}

}

React Lifecycle methods are hooks into different stages of a component’s existence, allowing you to run code at specific points in a component’s life. In Class Components, common lifecycle methods include:

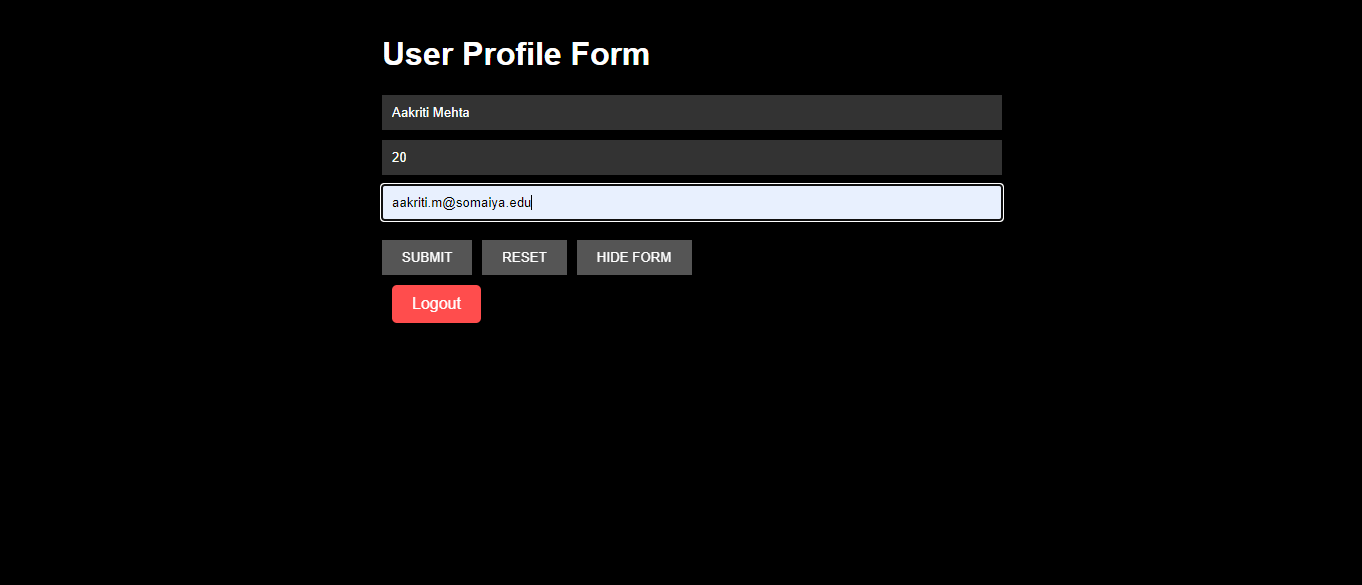
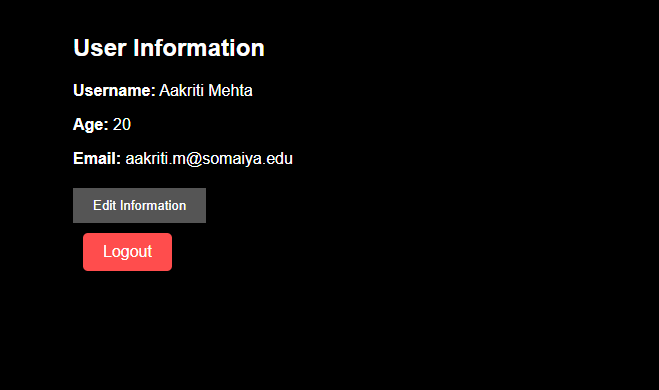
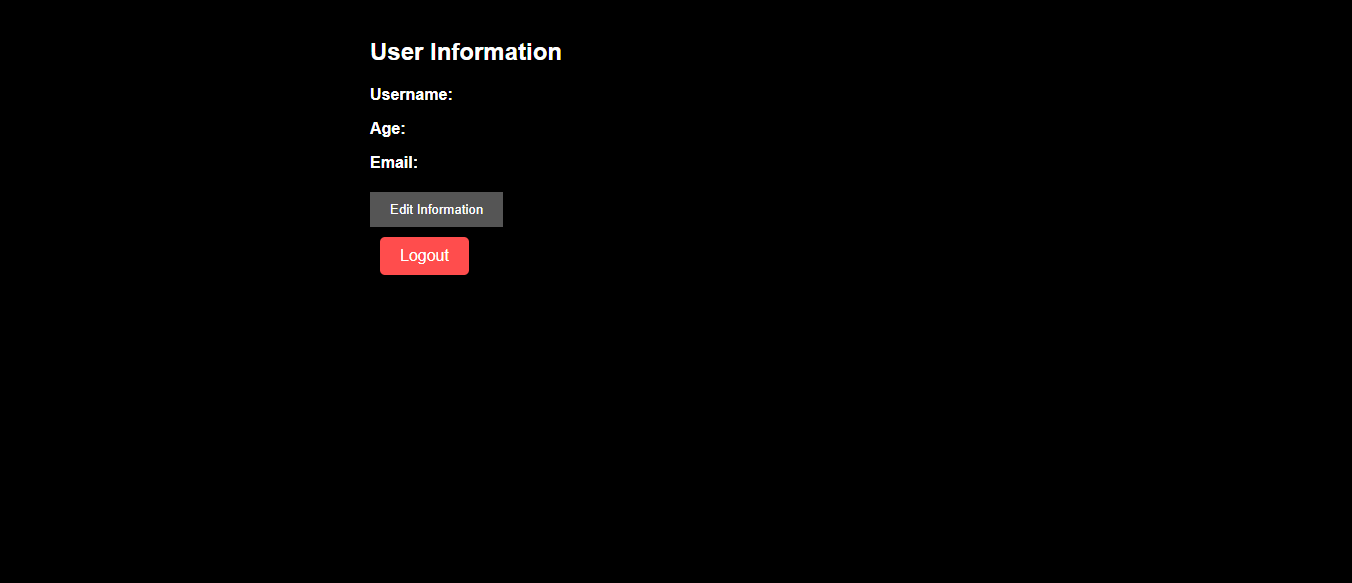
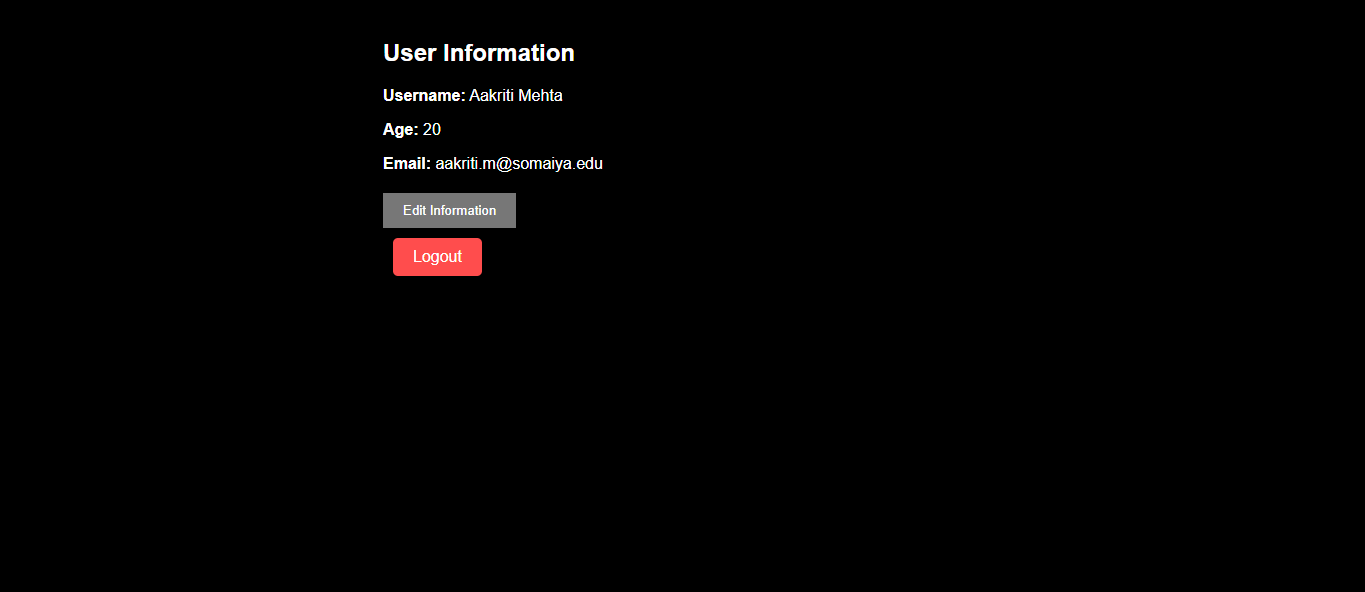
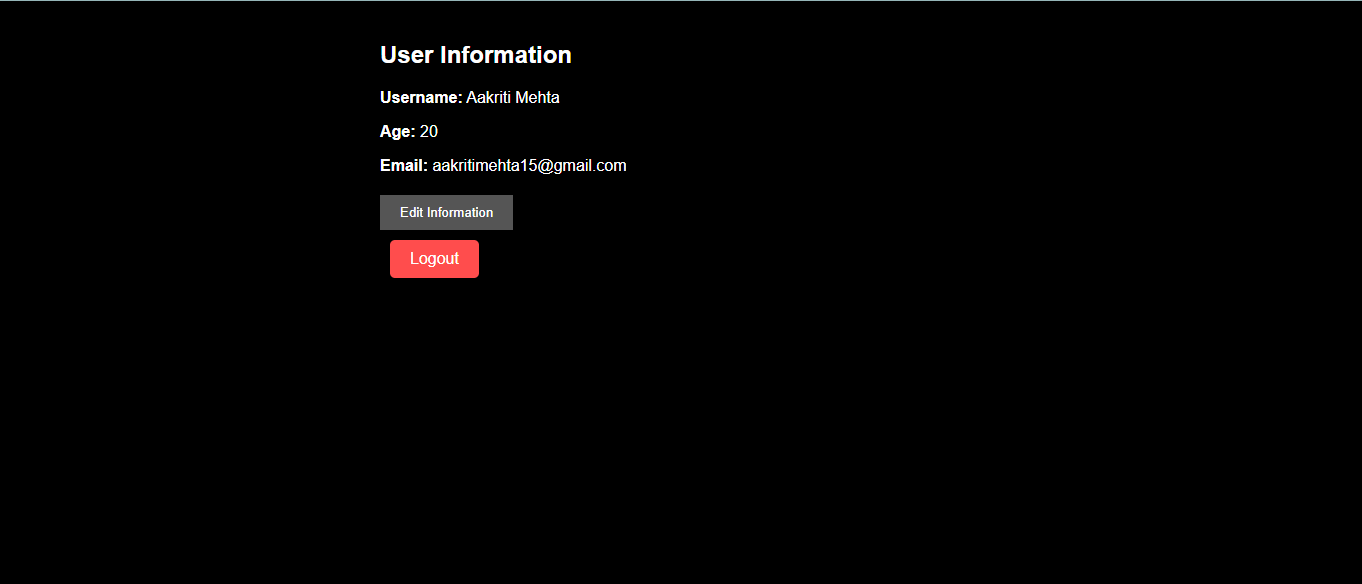
* componentDidMount(): After the component mounts.
* componentDidUpdate(prevProps, prevState): After updates.
* componentWillUnmount(): Before the component unmounts.

**Implementation Details:   
Task: User Profile Editor**

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

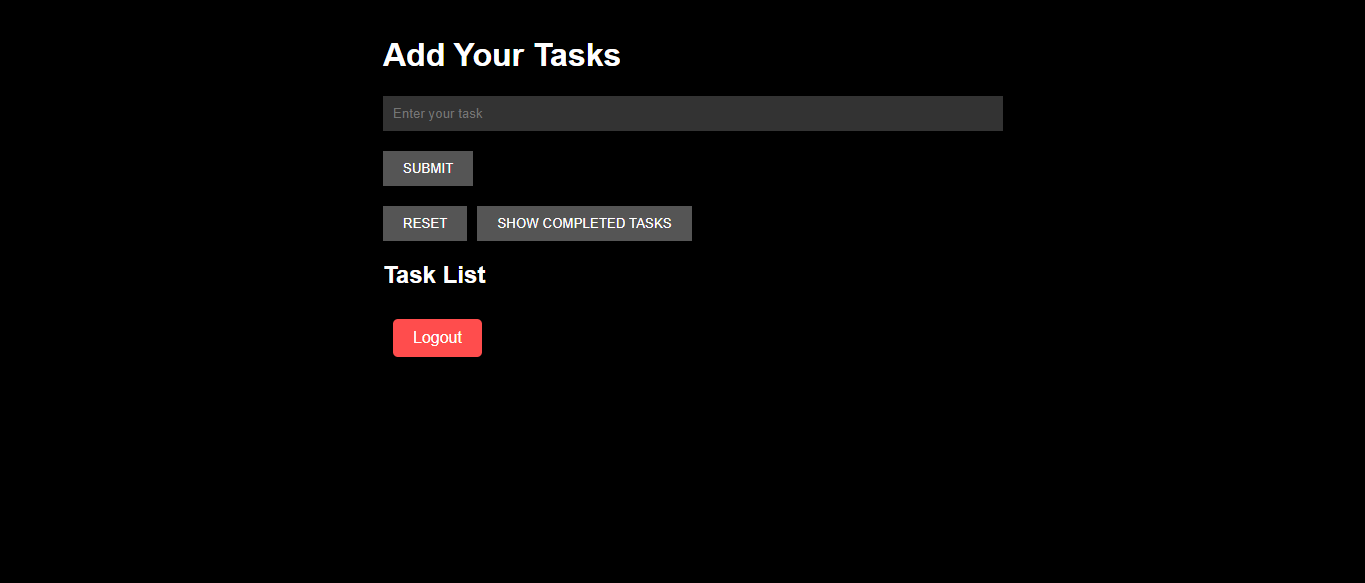
**2. Display the entered information below the form.**

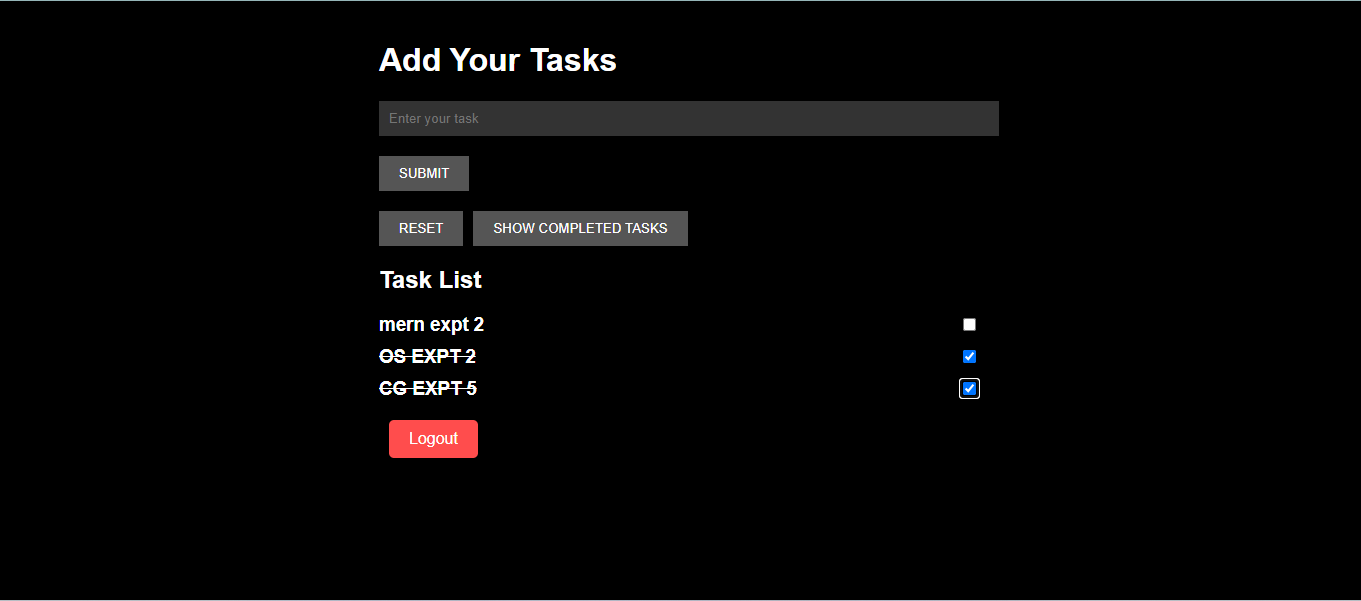
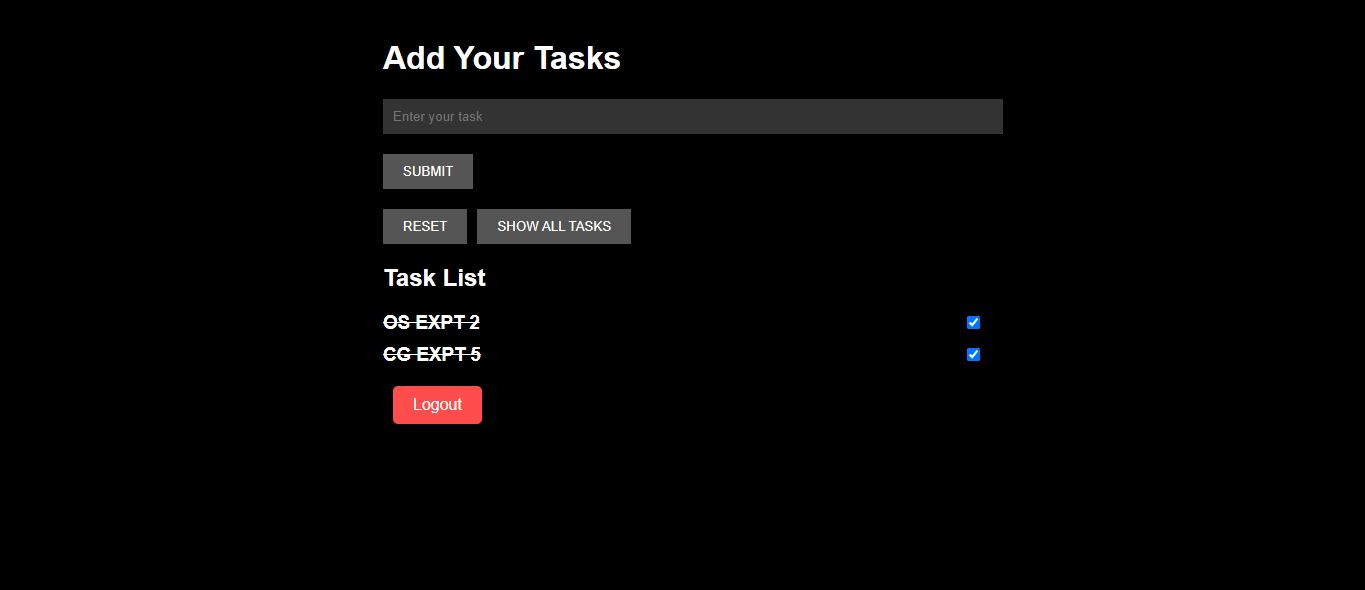
**3. Add a button to toggle the visibility of the form.**

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


### Task: Task Manager

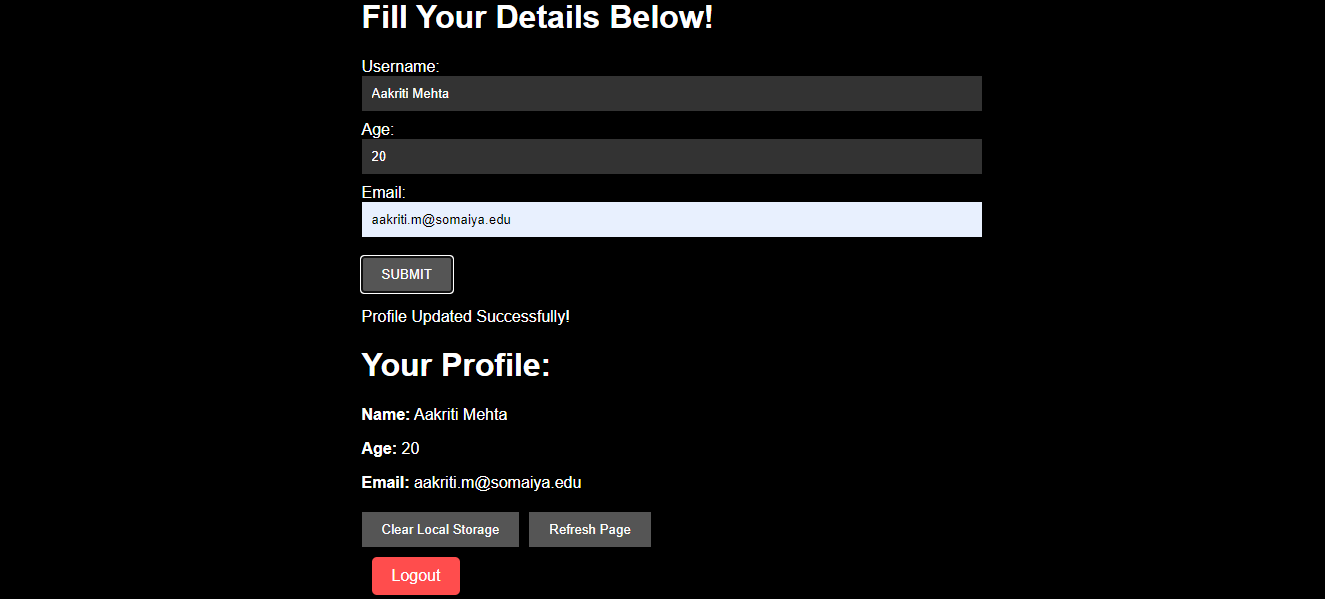
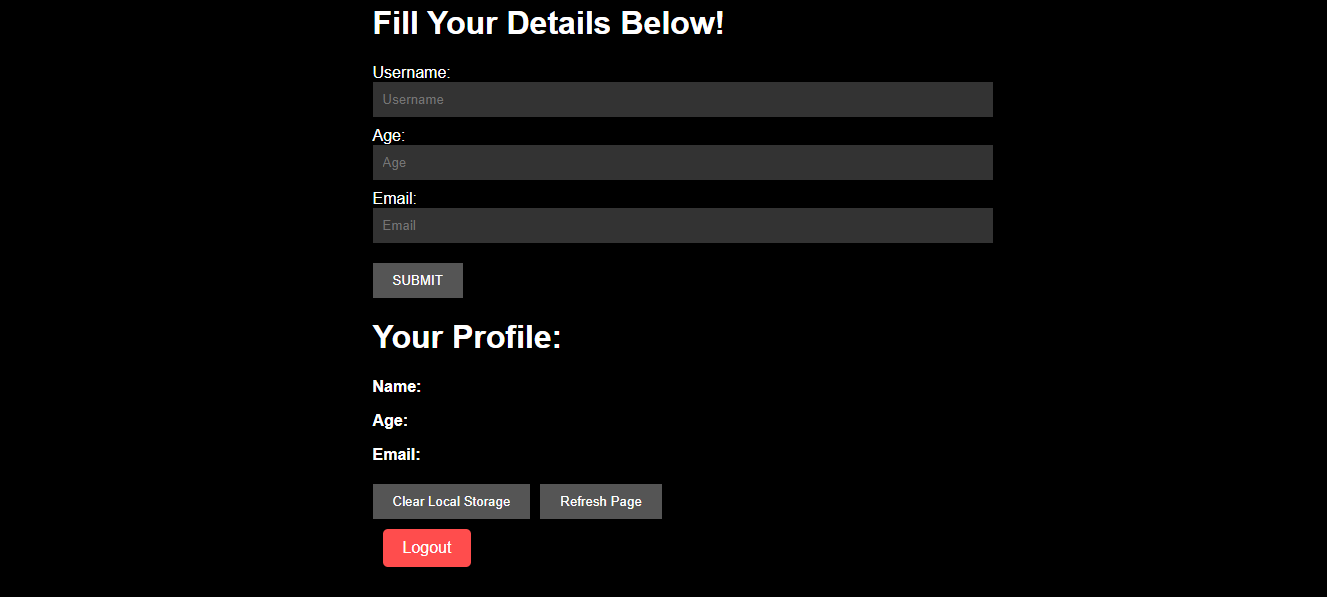
1. **Create a task manager with the following features:**
   * **An input field to add a new task.**
   * **A list to display added tasks.**
   * **A button to mark tasks as complete.**
   * **A button to toggle the visibility of completed tasks.**
   * **A button to reset the task list.**

****

****

### Task: 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.**

**  
  
**

**Conclusion:**

Learned basic react commands to build MERN application.

**Postlab questions:**

1) Different type of hooks

* Basic Hooks:
* useState: Manages state in functional components.

Example:  
const [count, setCount] = useState(0);

* useEffect: Handles side effects (e.g., data fetching).

Example:  
  
useEffect(() => { /\* side effect \*/ }, [dependencies]);

* useContext: Accesses context values.

Example:  
const value = useContext(MyContext);

* Additional Hooks:
* useReducer: Manages complex state logic.

Example:  
const [state, dispatch] = useReducer(reducer, initialState);

* useCallback: Memoizes callback functions.

Example:  
const memoizedCallback = useCallback(() => { /\* callback \*/ }, [dependencies]);

* useMemo: Memoizes expensive calculations.

Example:  
const memoizedValue = useMemo(() => computeExpensiveValue(a, b), [a, b]);

* useRef: Accesses and persists mutable values.

Example:  
const ref = useRef(null);

* Custom Hooks