Name- Veerpal Kaur

Student ID- 500224580

**Week 10- Assignment 3**

**GitHub repository link: -**

[Veerpalkaur08/React-Hooks-Assignment](https://github.com/Veerpalkaur08/React-Hooks-Assignment)

Part 1: -**use State: Managing Task State**

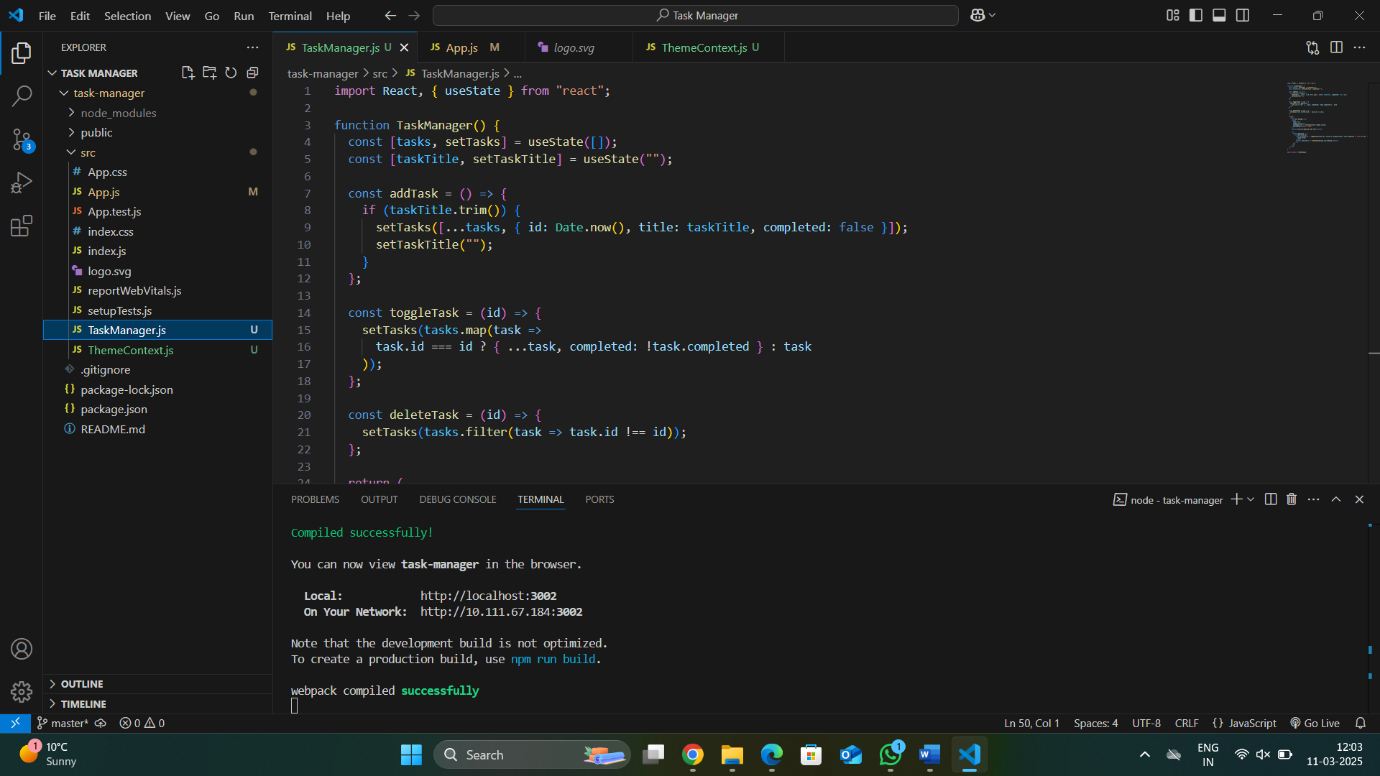
**Task:**

* Implement a form that allows users to add tasks with a **title** and **description**.
* Use useState to manage the list of tasks.

**Real-world Use Cases:**

1. Handling form inputs dynamically.
2. Managing UI state like modal visibility.
3. Toggling dark/light mode.

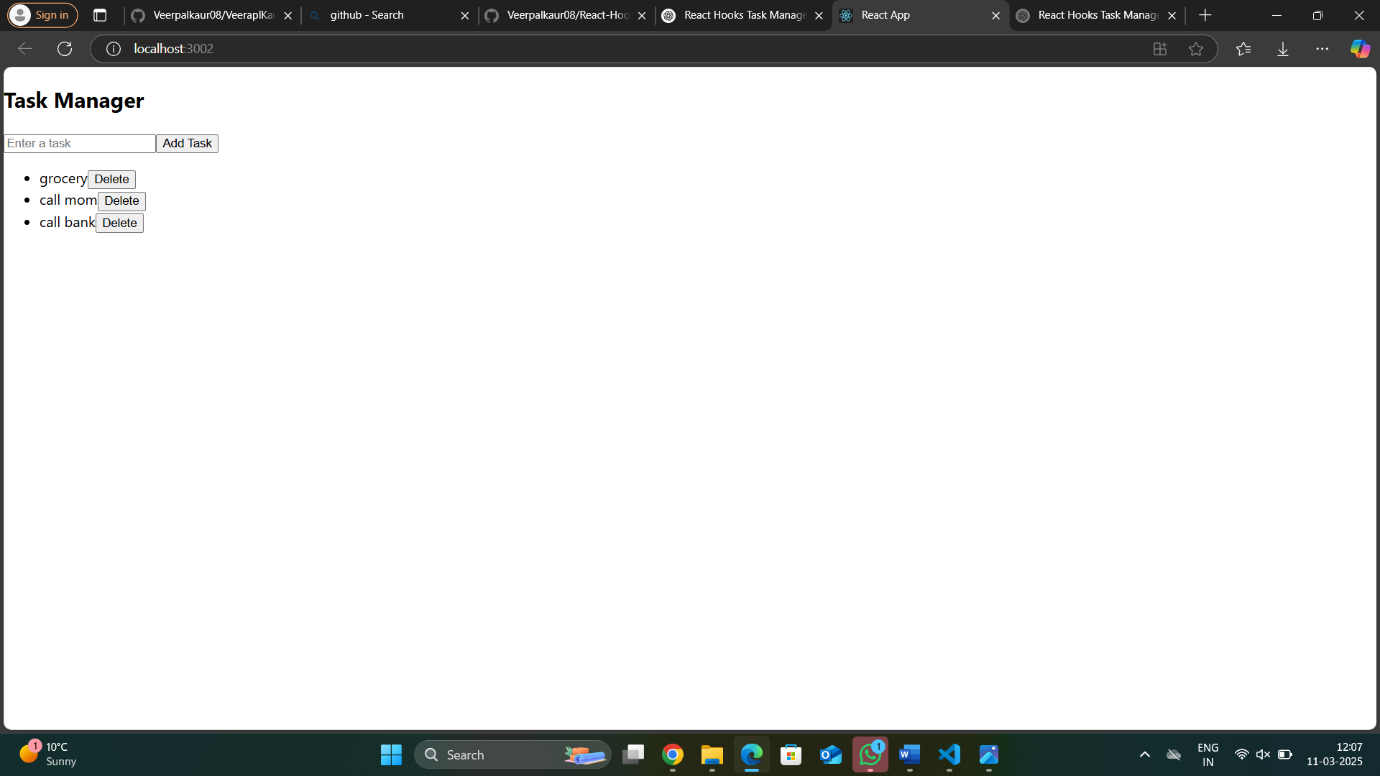
Code



A screenshot of a computer screen

AI-generated content may be incorrect.

Output



Part 2: - **useEffect: Side Effects and Data Persistence**

**Task:**

* Implement an **autosave** feature that saves tasks to localStorage whenever a new task is added or removed.
* Display an alert when the task list is updated.

**Real-world Use Cases:**

1. Fetching data from APIs when a component mounts.
2. Listening to window resize events.
3. Syncing state with local storage or a database.

Code: -

A screen shot of a computer screen

AI-generated content may be incorrect.

Output: -A screenshot of a computer

AI-generated content may be incorrect.

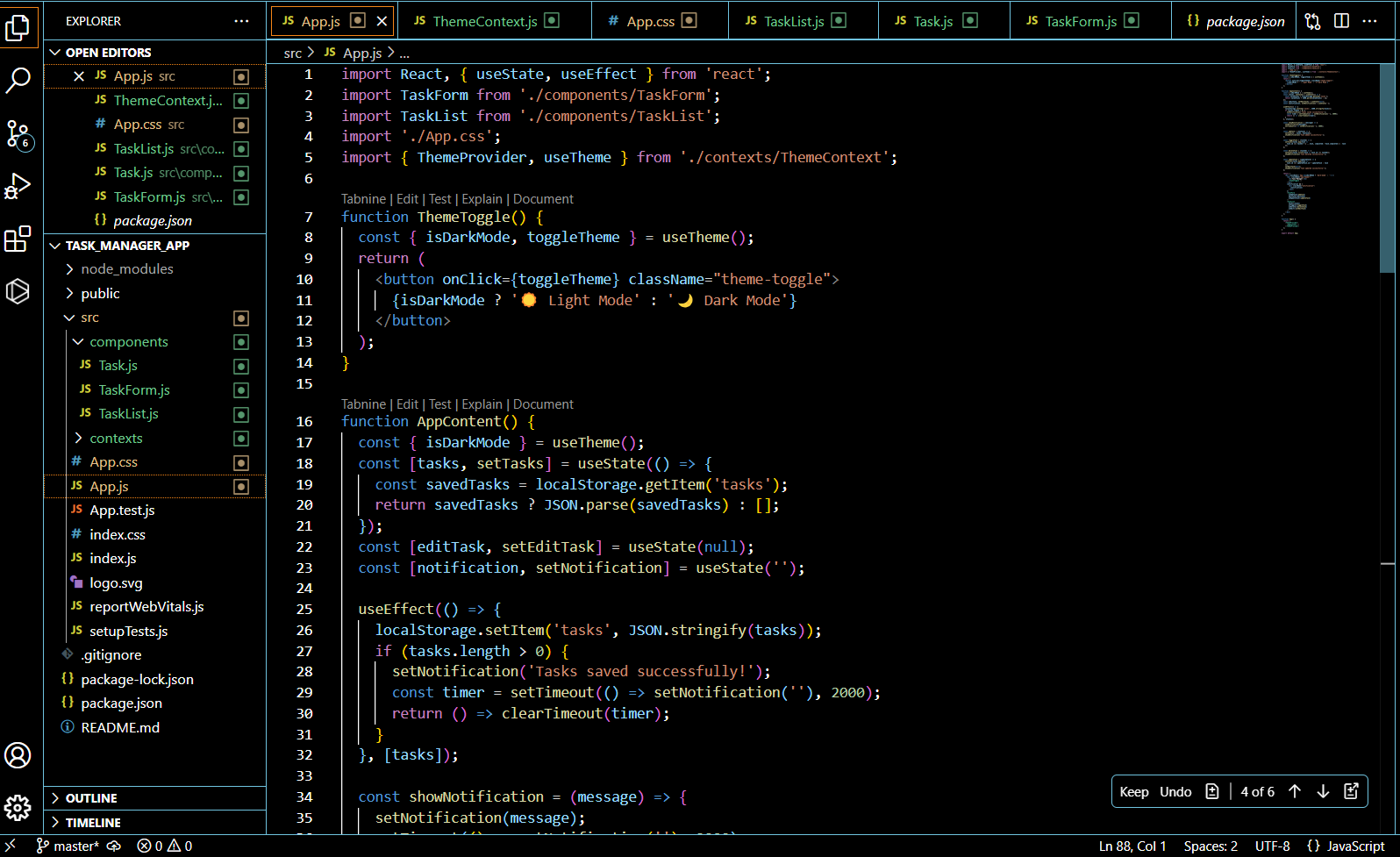
Part 3: -  **useContext: Theme Toggle (Dark Mode)**

**Task:**

* Implement a **dark mode toggle** using useContext.
* The theme should persist across page reloads.

**Real-world Use Cases:**

1. Managing global themes (dark/light mode).
2. Sharing user authentication state across components.
3. Managing language preferences in multi-lingual apps.

Code: 

Output in Light Mode: -

A screenshot of a computer

AI-generated content may be incorrect.

Output in Dark Mode: -

A screenshot of a computer

AI-generated content may be incorrect.

Part 4: -

**useReducer: Complex State Management**

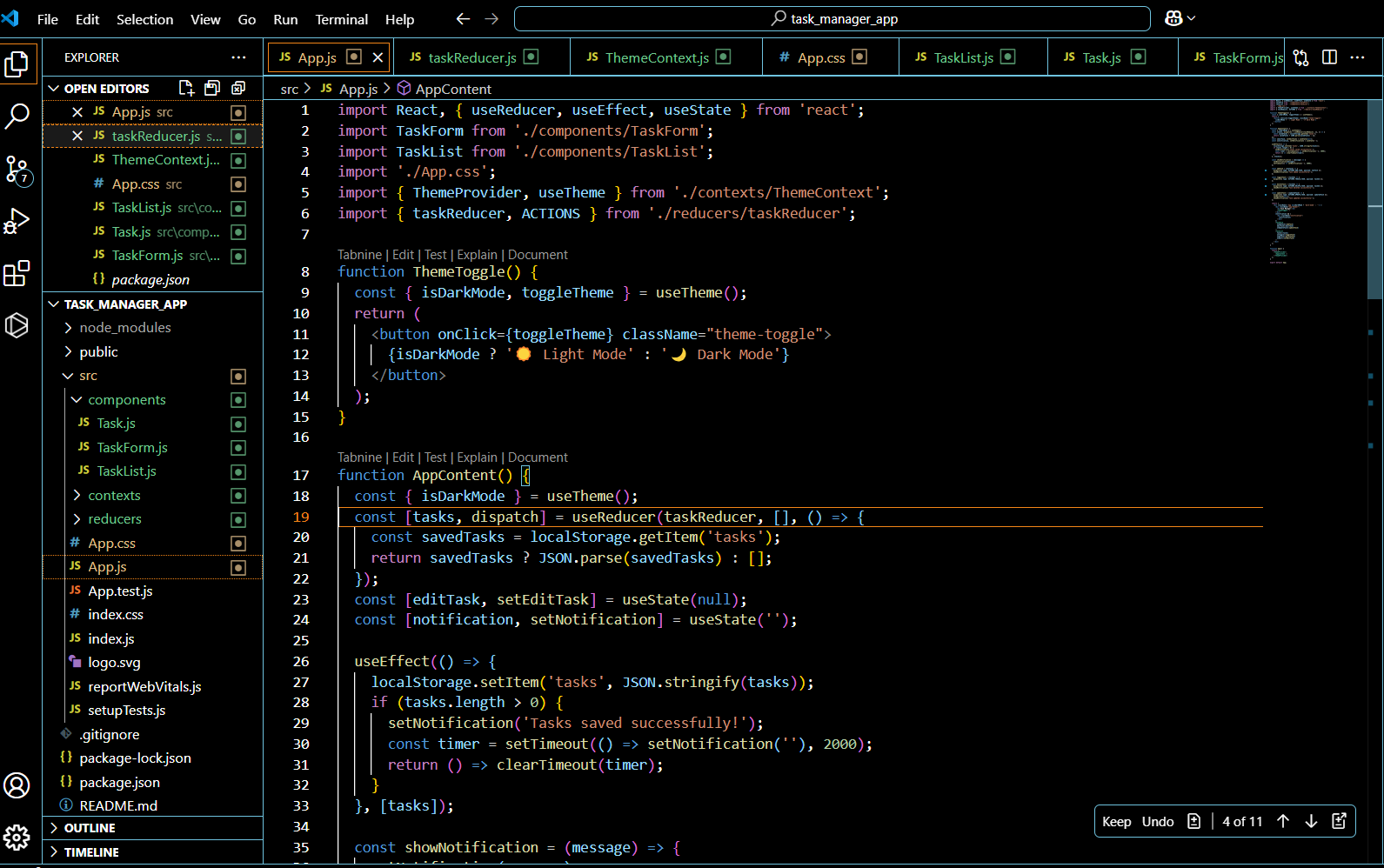
**Task:**

* Replace useState with useReducer to manage tasks.
* Implement actions for adding, updating, and deleting tasks.

**Real-world Use Cases:**

1. Managing complex state logic (shopping cart, forms with multiple fields).
2. Handling undo/redo functionality.
3. Managing authentication state.

Code: -



Part 5: -

**useRef: Managing References and Performance Optimization**

**Task:**

* Use useRef to store a reference to the task input field.
* Auto-focus the input field when the component mounts.

**Real-world Use Cases:**

1. Storing references to DOM elements (focus management, input fields).
2. Tracking previous values without triggering re-renders.

Controlling animations.

Code: -

A screen shot of a computer program

AI-generated content may be incorrect.

Part 6: -

**useMemo: Performance Optimization**

**Task:**

* Optimize task filtering performance using useMemo.
* Prevent unnecessary re-renders when filtering tasks.

**Real-world Use Cases:**

1. Optimizing expensive computations (filtering, sorting large datasets).
2. Preventing unnecessary re-renders in performance-critical components.
3. Caching API response transformations.

Code: -

A screen shot of a computer program

AI-generated content may be incorrect.

Output: -

A screenshot of a computer

AI-generated content may be incorrect.

Part 7: -

**useCallback: Memoizing Functions**

**Task:**

* Use useCallback to optimize the onClick event handlers for updating and deleting tasks.
* Prevent functions from being re-created on every render.

**Real-world Use Cases:**

1. Preventing unnecessary re-renders in child components.
2. Optimizing event handlers in large applications.
3. Reducing memory usage in frequently re-rendered components.

Code: -

