**PRACTICAL: 6**

**AIM:**

Build a To-do List using React Hooks involves creating a user

interface that allows users to add, remove, and delete tasks. This can

be achieved using React's state management capabilities through

hooks like useState and event handling for managing tasks.

**THEORY:**

HTML (HyperText Markup Language)

HTML is used to create the structure of the voting system. It defines the layout of the poll, including the question, voting buttons, and the section where results are displayed. Each button is associated with a voting option, and result areas are set up to show the current vote counts.

2. CSS (Cascading Style Sheets)

CSS is used to style the poll interface, making it visually appealing and user-friendly. It controls the layout, colors, spacing, and overall appearance of the poll and results sections.

3. JavaScript

JavaScript provides the interactivity and real-time functionality of the voting system. The main concepts used are:

Event Handling: JavaScript listens for button clicks to register votes.

DOM Manipulation: The script updates the displayed vote counts dynamically by changing the content of HTML elements.

State Management: A JavaScript object (e.g., votes) keeps track of the current vote counts for each option.

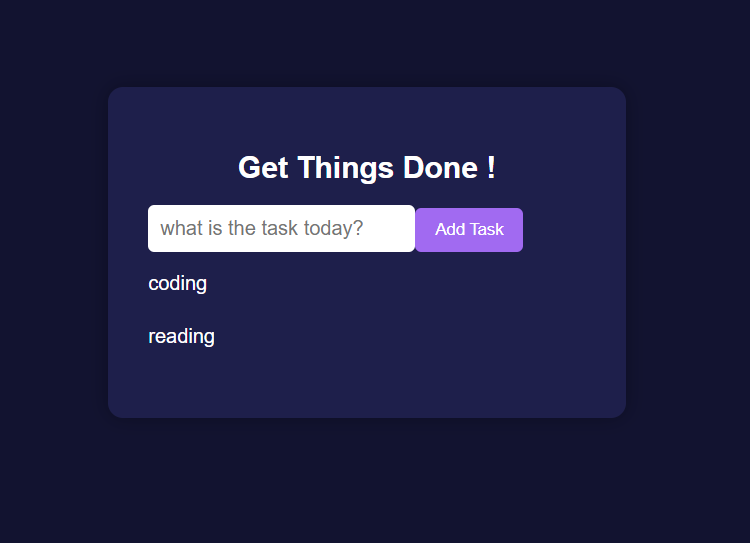
Functions: Functions like vote() and updateVotes() encapsulate the logic for voting and updating the UI.

setInterval: This function is used to simulate real-time voting by periodically (every 2 seconds) incrementing random vote counts, mimicking votes from other users.

**CODE:**

|  |
| --- |
| **APP.JS**  import React, { useState } from 'react';  import './App.css'  function App() {    const [task, setTask] = useState('');    const [tasks, setTasks] = useState([]);    const [editIndex, setEditIndex] = useState(null);    const handleAddTask = () => {      if(task.trim() === '') return;      if(editIndex !== null) {        const updatedTasks = tasks.map((t, index) =>        index === editIndex ? task : t        );        setTasks(updatedTasks);        setEditIndex(null);      } else {        setTasks([...tasks, task]);      }      setTask('');    };    const handleEdit = (index) => {      setTask(tasks[index]);      setEditIndex(index);    };      const handleDelete = (index) => {      const updatedTasks = tasks.filter((\_, i) => i !== index);      setTasks(updatedTasks);      if (editIndex === index) {        setEditIndex(null);        setTask('');      }    };  return (    <><div className='container'>    <h2>Get Things Done ! </h2>    <div className='input=section'><input      type='text'      placeholder='what is the task today?'      value={task}      onChange={(e) => setTask(e.target.value)}      />      <button onClick={handleAddTask}>        {editIndex !== null ? 'update task' : 'Add Task'}      </button>      </div>      <div className='task-list'>        {tasks.map((t, index) => (          <div key={index} className='tasks'>            <span>{t}</span>            <div className='icons'>              <button onClick={() => handleEdit(index)}></button>              <button onClick={() => handleDelete(index)}></button>            </div>          </div>        ))}      </div>    </div>   </>  );  }  export default App;  **APP.CSS**  body {    font-family: Arial, sans-serif;    background-color: #121330;    display: flex;    justify-content: center;    align-items: center;    height: 100vh;  }  .container {    background: #1e1f4b;    padding: 2rem;    border-radius: 12px;    box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);    width: 350px;    color: white;  }  h2 {    text-align: center;    margin-bottom: 1rem;  }  .input-section {    display: flex;    gap: 10px;  }  input {    flex: 1;    padding: 0.6rem;    border: none;    border-radius: 5px;    font-size: 1rem;  }  button {    background: #a16af1;    color: white;    border: none;    padding: 0.6rem 1rem;    border-radius: 5px;    cursor: pointer;  }  .task-list {    margin-top: 1rem;  }  .task {    background: #8757f2;    margin-bottom: 0.5rem;    padding: 0.8rem;    border-radius: 5px;    display: flex;    justify-content: space-between;    align-items: center;  }  .icons button {    background: transparent;    border: none;    color: white;    font-size: 1rem;    margin-left: 0.5rem;    cursor: pointer;  } |

**OUTPUT:**

****

**LATEST APPLICATIONS:**

In todays time html css and javascript are great stepping stones to get into the world of learning web development, a lot of simple websites use this tech stack to reduce complexity and cost of hostings.

**LEARNING OUTCOME:**

By performing this practical I got a thorough understanding of basic web technologies and terms. And how the frontend and backend connect in a real website

**REFERENCES:**

1. React: react.dev
2. Html: <https://www.freecodecamp.org/learn/2022/responsive-web-design/learn-html-by-building-a-cat-photo-app/step-66>
3. Css: <https://www.w3schools.com/css/>