TP2:UseState && routing

1. Initialisation du projet

Crée un nouveau projet React en utilisant **`create-react-app`** :

```
npx create-react-app todo-app
cd todo-app
npm install react-router-dom
npm start
```

2. Structure du projet

Tu peux organiser ton projet comme suit :

```
src/
|-- components/
|   |-- Home.js
|   |-- TodoList.js
|-- App.js
|-- index.js
```

3. Configurer `react-router-dom`

Dans le fichier **`App.js`**, ajoute les routes pour naviguer entre les pages.

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link } from
'react-router-dom';
import Home from './components/Home';
import TodoList from './components/TodoList';

function App() {
  return (
    <Router>
      <div>
        <nav>
          <ul>
            <li>
              <Link to="/">Home</Link>
            </li>
```

```
            <li>
              <Link to="/todos">Todo List</Link>
            </li>
          </ul>
        </nav>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/todos" element={<TodoList />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

4. Créer la page Home

Dans le fichier **`components/Home.js`**, ajoute une simple page d'accueil.

```
import React from 'react';

function Home() {
  return (
    <div>
      <h1>Bienvenue sur l'application de gestion des tâches</h1>
    </div>
  );
}

export default Home;
```

5. Utilisation de useState dans la gestion des tâches

Dans le fichier `components/TodoList.js`, tu vas créer la logique pour gérer la liste des tâches avec le hook **`useState`.**

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

function TodoList() {
```

```
  const [task, setTask] = useState('');
  const [tasks, setTasks] = useState([]);

  const handleAddTask = () => {
    if (task.trim()) {
      setTasks([...tasks, task]);
      setTask(''); // Clear the input field
    }
  };

  const handleDeleteTask = (index) => {
    const newTasks = tasks.filter((_, i) => i !== index);
    setTasks(newTasks);
  };

  return (
    <div>
      <h1>Liste des tâches</h1>
      <input
        type="text"
        value={task}
        onChange={(e) => setTask(e.target.value)}
        placeholder="Ajouter une tâche"
      />
      <button onClick={handleAddTask}>Ajouter</button>
      <ul>
        {tasks.map((task, index) => (
          <li key={index}>
            {task}
            <button onClick={() =>
handleDeleteTask(index)}>Supprimer</button>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default TodoList;
```

6. Explication du Code

- **`useState` : Utilisé pour gérer l'état des tâches (`task` pour la tâche actuelle et `tasks` pour la liste des tâches).**
    - **`task` est utilisé pour stocker la valeur de l'input.**
    - **`tasks` contient toutes les tâches ajoutées.**
- **Ajout d'une tâche : La fonction `handleAddTask` vérifie que la tâche n'est pas vide avant de l'ajouter à la liste `tasks`.**
- **Suppression d'une tâche : Chaque tâche a un bouton "Supprimer" qui utilise la fonction `handleDeleteTask` pour retirer une tâche de la liste.**