**Assignment 10: Building a Multi-Component App**

**Combine the concepts learned to build a simple multi-component React application.**

**Ensure navigation between different components works correctly.**

Let's build a simple multi-component React application that includes global state management with the Context API and navigation using React Router. In this example, we'll create a To-Do List application with different components for adding, listing, and managing tasks.

Step 1: Set up React Router and Global State

Follow the steps from Assignment 8 to set up React Router and create a global state using the Context API.

Step 2: Create Components

Create the necessary components for your To-Do List application. For simplicity, let's create AddTask.js, TaskList.js, and Task.js

| // AddTask.js  import React, { useState } from 'react';  import { useAppContext } from './AppContext';  const AddTask = () => {  const { dispatch } = useAppContext();  const [task, setTask] = useState('');  const addTask = () => {  if (task.trim() !== '') {  dispatch({ type: 'ADD\_TASK', payload: task });  setTask('');  }  };  return (  <div>  <input type="text" value={task} onChange={(e) => setTask(e.target.value)} />  <button onClick={addTask}>Add Task</button>  </div>  );  };  export default AddTask; |
| --- |

| // TaskList.js  import React from 'react';  import Task from './Task';  import { useAppContext } from './AppContext';  const TaskList = () => {  const { state } = useAppContext();  return (  <div>  <h2>Task List</h2>  {state.tasks.map((task) => (  <Task key={task.id} task={task} />  ))}  </div>  );  };  export default TaskList; |
| --- |

| // Task.js  import React from 'react';  const Task = ({ task }) => {  return <div>{task.text}</div>;  };  export default Task; |
| --- |

Step 3: Update AppRouter

Update the AppRouter.js file to include routes for your components:

| // AppRouter.js  import React from 'react';  import { BrowserRouter as Router, Route, Switch, Link } from 'react-router-dom';  import AddTask from './AddTask';  import TaskList from './TaskList';  const AppRouter = () => {  return (  <Router>  <div>  <nav>  <ul>  <li>  <Link to="/">Add Task</Link>  </li>  <li>  <Link to="/tasks">Task List</Link>  </li>  </ul>  </nav>  <Switch>  <Route path="/tasks">  <TaskList />  </Route>  <Route path="/">  <AddTask />  </Route>  </Switch>  </div>  </Router>  );  };  export default AppRouter; |
| --- |

Step 4: Update Reducer in AppContext.js

Update the reducer in the AppContext.js file to handle adding tasks:

| // AppContext.js  // ... (previous code)  const AppProvider = ({ children }) => {  const reducer = (state, action) => {  switch (action.type) {  case 'INCREMENT':  return { ...state, count: state.count + 1 };  case 'DECREMENT':  return { ...state, count: state.count - 1 };  case 'ADD\_TASK':  return {  ...state,  tasks: [...state.tasks, { id: Date.now(), text: action.payload }],  };  default:  return state;  }  };  // ... (remaining code)  };  // ... (remaining code) |
| --- |

Step 5: Integrate in App.js

Finally, integrate everything in your App.js file:

| // App.js  import React from 'react';  import { AppProvider } from './AppContext';  import AppRouter from './AppRouter';  const App = () => {  return (  <AppProvider>  <div>  <h1>React Multi-Component App</h1>  <AppRouter />  </div>  </AppProvider>  );  };  export default App; |
| --- |

Step 6: Run the App

Start your development server:

npm start

Visit http://localhost:3000 in your browser. You should see your To-Do List application with the ability to add tasks and view the task list. The navigation between different components should work correctly.

Feel free to enhance this example based on your needs and add more features to your To-Do List application. This example illustrates the combination of React Router for navigation, the Context API for global state management, and multiple components working together.