Lab Activity: Building a Todo List App using ReactJS and Redux

Objectives:

- Understand how to use ReactJS to build a web application
- Learn about the Redux architecture for managing application state
- Understand how to deploy a ReactJS and Redux application using CodeSandbox

Instructions:

1. Create a new CodeSandbox project by navigating to https://codesandbox.io/ and clicking on the "Create Sandbox" button.
2. In the "Select a Template" section, choose the "React" template.
3. Rename the default "index.js" file to "App.js".
4. Create a new directory named "actions" in the src folder.
5. Create a new file named "types.js" inside the "actions" directory.
6. Define the following constants in the "types.js" file:

```
export const ADD_TODO = 'ADD_TODO';
export const REMOVE_TODO = 'REMOVE_TODO';
```

7. Create a new file named "actions.js" inside the "actions" directory.
8. Define the following action creators in the "actions.js" file:

```
import { ADD_TODO, REMOVE_TODO } from './types';

export const addTodo = (text) => ({
  type: ADD_TODO,
  payload: { text },
});

export const removeTodo = (id) => ({
  type: REMOVE_TODO,
  payload: { id },
});
```

9. Create a new directory named "reducers" in the src folder.
10. Create a new file named "index.js" inside the "reducers" directory.
11. Define the initial state and the reducer function in the "index.js" file:

```
import { ADD_TODO, REMOVE_TODO } from '../actions/types';

const initialState = {
  todos: [],
};

const rootReducer = (state = initialState, action) => {
  switch (action.type) {
    case ADD_TODO:
      return {
        ...state,
        todos: [...state.todos, { id: Date.now(), text:
action.payload.text }],
```

```
      };
    case REMOVE_TODO:
      return {
        ...state,
        todos: state.todos.filter((todo) => todo.id !==
action.payload.id),
      };
    default:
      return state;
  }
};

export default rootReducer;
```

12. Import the rootReducer function in the "App.js" file and create the Redux store using the createStore function from the redux library:

```
import React from 'react';
import { createStore } from 'redux';
import { Provider } from 'react-redux';
import rootReducer from './reducers';
import TodoList from './components/TodoList';

const store = createStore(rootReducer);

const App = () => {
  return (
    <Provider store={store}>
      <TodoList />
    </Provider>
  );
};

export default App;
```

13. Create a new directory named "components" in the src folder.
14. Create a new file named "TodoList.js" inside the "components" directory.
15. Define the TodoList component in the "TodoList.js" file and connect it to the Redux store using the connect function from the react-redux library:

```
import React, { useState } from 'react';
import { connect } from 'react-redux';
import { addTodo, removeTodo } from '../actions';

const TodoList = ({ todos, addTodo, removeTodo }) => {
  const [text, setText] = useState('');

  const handleSubmit = (event) => {
    event.preventDefault();
    if (text.trim()) {
          addTodo(text);
     setText('');
   }
 };

 return(
  <div>
```

```
      <form onSubmit={handleSubmit}>
        <input type="text" value={text} onChange={(e) =>
setText(e.target.value)} />
        <button type="submit">Add</button>
      </form>
      <ul>
        {todos.map((todo) => (
          <li key={todo.id}>
            {todo.text}{' '}
            <button onClick={() => removeTodo(todo.id)}>Remove</button>
          </li>
        ))}
      </ul>
    </div>
  );
};

const mapStateToProps = (state) => ({
todos: state.todos,
});

export default connect(mapStateToProps, { addTodo, removeTodo })(TodoList);
```

16. Test the application locally by clicking on the "Run" button in CodeSandbox.
17. Once the application is working correctly, deploy it to CodeSandbox by clicking on the "Share" button in the top menu bar and copying the link.
18. Share the link with the students and have them test the application in their own browser.

Additional Tasks:

- Add a new feature to the application, such as a checkbox to mark a task as completed or a search bar to filter the tasks by keyword.
- Use CSS to style the application and make it more visually appealing.

Submit your work here not later than **May 12, 2023**: https://forms.gle/C3n2sSByaU9cpgYM9