Task 1:

• Create a simple counter application using Redux for state management. Implement

actions to increment and decrement the counter.

**counterSlice.js:**

import { createSlice } from "@reduxjs/toolkit";

const counterSlice = createSlice({

name: "counter",

initialState: { value: 0 },

reducers: {

increment: (state) => {

state.value += 1;

},

decrement: (state) => {

state.value -= 1;

},

},

});

export const { increment, decrement } = counterSlice.actions;

export default counterSlice.reducer;

**store.js:**

import { configureStore } from "@reduxjs/toolkit";

import counterReducer from "./counterSlice";

const store = configureStore({

reducer: {

counter: counterReducer,

},

});

export default store;

**App.js:**

import React from "react";

import { useSelector, useDispatch } from "react-redux";

import { increment, decrement } from "./counterSlice";

function App() {

const count = useSelector((state) => state.counter.value);

const dispatch = useDispatch();

return (

<div>

<h1>Counter: {count}</h1>

<button onClick={() => dispatch(increment())}>+</button>

<button onClick={() => dispatch(decrement())}>-</button>

</div>

);

}

export default App;

**index.js:**

import React from "react";

import ReactDOM from "react-dom";

import App from "./App";

import { Provider } from "react-redux";

import store from "./store";

ReactDOM.render(

<Provider store={store}>

<App />

</Provider>,

document.getElementById("root")

);

Task 2:

• Build a todo list application using Recoil for state management. Allow users to add,

remove, and mark tasks as complete.

**atoms.js:**

import { atom } from "recoil";

export const todoListState = atom({

key: "todoListState",

default: [],

});

**App.js:**

import React, { useState } from "react";

import { RecoilRoot, useRecoilState } from "recoil";

import { todoListState } from "./atoms";

function TodoApp() {

const [todos, setTodos] = useRecoilState(todoListState);

const [task, setTask] = useState("");

const addTask = () => {

setTodos([...todos, { text: task, completed: false }]);

setTask("");

};

const toggleTask = (index) => {

const newTodos = [...todos];

newTodos[index].completed = !newTodos[index].completed;

setTodos(newTodos);

};

const removeTask = (index) => {

const newTodos = todos.filter((\_, i) => i !== index);

setTodos(newTodos);

};

return (

<div>

<h2>Todo List</h2>

<input value={task} onChange={(e) => setTask(e.target.value)} />

<button onClick={addTask}>Add</button>

<ul>

{todos.map((todo, index) => (

<li key={index} style={{ textDecoration: todo.completed ? "line-through" : "none" }}>

{todo.text}

<button onClick={() => toggleTask(index)}>✔</button>

<button onClick={() => removeTask(index)}>✘</button>

</li>

))}

</ul>

</div>

);

}

export default function App() {

return (

<RecoilRoot>

<TodoApp />

</RecoilRoot>

);

}

Task 3:

• Build a crud application using Redux-Toolkit for state management. Allow users to add,

remove, delete and update.

**crudSlice.js:**

import { createSlice } from "@reduxjs/toolkit";

const crudSlice = createSlice({

name: "crud",

initialState: [],

reducers: {

addItem: (state, action) => {

state.push({ id: Date.now(), text: action.payload });

},

deleteItem: (state, action) => {

return state.filter(item => item.id !== action.payload);

},

updateItem: (state, action) => {

const index = state.findIndex(item => item.id === action.payload.id);

if (index !== -1) {

state[index].text = action.payload.text;

}

},

},

});

export const { addItem, deleteItem, updateItem } = crudSlice.actions;

export default crudSlice.reducer;

**store.js:**

import { configureStore } from "@reduxjs/toolkit";

import crudReducer from "./crudSlice";

const store = configureStore({

reducer: {

crud: crudReducer,

},

});

export default store;

**App.js:**

import React, { useState } from "react";

import { useSelector, useDispatch } from "react-redux";

import { addItem, deleteItem, updateItem } from "./crudSlice";

function App() {

const items = useSelector(state => state.crud);

const dispatch = useDispatch();

const [input, setInput] = useState("");

const [editId, setEditId] = useState(null);

const handleAdd = () => {

if (editId) {

dispatch(updateItem({ id: editId, text: input }));

setEditId(null);

} else {

dispatch(addItem(input));

}

setInput("");

};

return (

<div>

<h2>CRUD App</h2>

<input value={input} onChange={(e) => setInput(e.target.value)} />

<button onClick={handleAdd}>{editId ? "Update" : "Add"}</button>

<ul>

{items.map(item => (

<li key={item.id}>

{item.text}

<button onClick={() => { setEditId(item.id); setInput(item.text); }}>Edit</button>

<button onClick={() => dispatch(deleteItem(item.id))}>Delete</button>

</li>

))}

</ul>

</div>

);

}

export default App**;**

**index.js:**

import React from "react";

import ReactDOM from "react-dom";

import App from "./App";

import { Provider } from "react-redux";

import store from "./store";

ReactDOM.render(

<Provider store={store}>

<App />

</Provider>,

document.getElementById("root")

);