**Exercise 1: Button Click Counter**

**Task:**

* Create a component with a button.
* Increment a counter each time the button is clicked.

**Solution:**

**Counter.js**

import React, { useState } from "react";

import "./Counter.css";

const Counter = () => {

const [count, setCount] = useState(0);

return (

<div className="counter-container">

<h2>Count: {count}</h2>

<button onClick={() => setCount(count + 1)} className="counter-btn">

Increment

</button>

</div>

);

};

export default Counter;

**App.js**

import React from "react";

import Counter from "./Counter";

const App = () => {

return (

<div>

<h1>Counter Application</h1>

<Counter />

</div>

);

};

export default App;

**Counter.css**

.counter-container {

text-align: center;

font-family: Arial, sans-serif;

}

.counter-btn {

background-color: #4caf50;

color: white;

border: none;

padding: 10px 20px;

margin-top: 10px;

font-size: 16px;

cursor: pointer;

}

.counter-btn:hover {

background-color: #45a049;

}

**Exercise 2: Input Field Event**

**Task:**

* Create a component with an input field.
* Display the input text below the field dynamically as the user types.

**Solution:**

**InputField.js**

import React, { useState } from "react";

import "./InputField.css";

const InputField = () => {

const [text, setText] = useState("");

return (

<div className="input-container">

<input

type="text"

placeholder="Type something..."

value={text}

onChange={(e) => setText(e.target.value)}

className="input-field"

/>

<p className="display-text">{text}</p>

</div>

);

};

export default InputField;

**App.js**

import React from "react";

import Calculator from "./Calculator";

const App = () => {

return (

<div>

<h1>Calculator</h1>

<Calculator />

</div>

);

};

export default App;

**InputField.css**

.input-container {

width: 300px;

margin: 20px auto;

text-align: center;

font-family: Arial, sans-serif;

}

.input-field {

width: 100%;

padding: 10px;

font-size: 14px;

border: 1px solid #ccc;

border-radius: 4px;

}

.display-text {

margin-top: 10px;

font-size: 18px;

color: #333;

}

**Exercise 3: Mouse Hover Effect**

**Task:**

* Create a component with a box.
* Change the color of the box when the mouse hovers over it.

**Solution:**

**HoverBox.js**

import React, { useState } from "react";

import "./HoverBox.css";

const HoverBox = () => {

const [hovered, setHovered] = useState(false);

return (

<div

className={`hover-box ${hovered ? "hovered" : ""}`}

onMouseEnter={() => setHovered(true)}

onMouseLeave={() => setHovered(false)}

>

Hover over me!

</div>

);

};

export default HoverBox;

**App.js**

import React from "react";

import Calculator from "./HoverBox ";

const App = () => {

return (

<div>

<h1>HoverBox </h1>

< HoverBox />

</div>

);

};

export default App;

**HoverBox.css**

.hover-box {

width: 200px;

height: 200px;

background-color: lightblue;

display: flex;

align-items: center;

justify-content: center;

font-size: 16px;

font-family: Arial, sans-serif;

color: white;

transition: background-color 0.3s ease;

}

.hover-box.hovered {

background-color: steelblue;

}

**Exercise 4: Toggle Theme**

**Task:**

* Create a component with a button.
* Toggle between light and dark themes when the button is clicked.

**Solution:**

**ThemeToggle.js**

import React, { useState } from "react";

import "./ThemeToggle.css";

const ThemeToggle = () => {

const [darkMode, setDarkMode] = useState(false);

return (

<div className={`theme-container ${darkMode ? "dark" : "light"}`}>

<h2>{darkMode ? "Dark Mode" : "Light Mode"}</h2>

<button onClick={() => setDarkMode(!darkMode)} className="theme-btn">

Toggle Theme

</button>

</div>

);

};

export default ThemeToggle;

**ThemeToggle.css**

.theme-container {

text-align: center;

padding: 20px;

font-family: Arial, sans-serif;

transition: background-color 0.3s ease, color 0.3s ease;

}

.theme-container.light {

background-color: white;

color: black;

}

.theme-container.dark {

background-color: black;

color: white;

}

.theme-btn {

margin-top: 20px;

padding: 10px 15px;

font-size: 16px;

background-color: #4caf50;

color: white;

border: none;

cursor: pointer;

}

.theme-btn:hover {

background-color: #45a049;

}

**Exercise 5: Click Tracker**

**Task:**

* Create a component that tracks the number of times a specific area is clicked.
* Reset the count when a reset button is clicked.

**Solution:**

**ClickTracker.js**

import React, { useState } from "react";

import "./ClickTracker.css";

const ClickTracker = () => {

const [clicks, setClicks] = useState(0);

return (

<div className="tracker-container">

<div

className="click-area"

onClick={() => setClicks(clicks + 1)}

>

Click Me

</div>

<p>Clicks: {clicks}</p>

<button onClick={() => setClicks(0)} className="reset-btn">

Reset

</button>

</div>

);

};

export default ClickTracker;

**ClickTracker.css**

.tracker-container {

text-align: center;

font-family: Arial, sans-serif;

}

.click-area {

width: 150px;

height: 150px;

background-color: #007bff;

color: white;

display: flex;

align-items: center;

justify-content: center;

margin: 10px auto;

border-radius: 8px;

cursor: pointer;

}

.click-area:hover {

background-color: #0056b3;

}

.reset-btn {

padding: 8px 12px;

font-size: 14px;

background-color: #ff4d4d;

color: white;

border: none;

cursor: pointer;

}

.reset-btn:hover {

background-color: #e60000;

}

**Exercise 6: Todo List Manager**

**Task:**

* Create a Todo app with the following components:
  + **TodoInput**: Input field and button for adding a todo.
  + **TodoItem**: Displays a single todo item with a delete button.
  + **TodoList**: Displays all todo items.

**Solution:**

**TodoApp.js**

import React, { useState } from "react";

import TodoInput from "./TodoInput";

import TodoList from "./TodoList";

import "./TodoApp.css";

const TodoApp = () => {

const [todos, setTodos] = useState([]);

const addTodo = (newTodo) => {

setTodos([...todos, newTodo]);

};

const deleteTodo = (index) => {

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

setTodos(updatedTodos);

};

return (

<div className="todo-app">

<h1>Todo Manager</h1>

<TodoInput addTodo={addTodo} />

<TodoList todos={todos} deleteTodo={deleteTodo} />

</div>

);

};

export default TodoApp;

**TodoInput.js**

import React, { useState } from "react";

import "./TodoInput.css";

const TodoInput = ({ addTodo }) => {

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

const handleAdd = () => {

if (input.trim()) {

addTodo(input);

setInput("");

}

};

return (

<div className="todo-input">

<input

type="text"

value={input}

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

placeholder="Enter a todo"

/>

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

</div>

);

};

export default TodoInput;

**TodoList.js**

import React from "react";

import TodoItem from "./TodoItem";

import "./TodoList.css";

const TodoList = ({ todos, deleteTodo }) => {

return (

<div className="todo-list">

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

<TodoItem

key={index}

todo={todo}

onDelete={() => deleteTodo(index)}

/>

))}

</div>

);

};

export default TodoList;

**File: TodoItem.js**

import React from "react";

import "./TodoItem.css";

const TodoItem = ({ todo, onDelete }) => {

return (

<div className="todo-item">

<span>{todo}</span>

<button onClick={onDelete}>Delete</button>

</div>

);

};

export default TodoItem;

**TodoApp.css**

.todo-app {

width: 400px;

margin: 20px auto;

text-align: center;

font-family: Arial, sans-serif;

}

h1 {

color: #333;

}

**TodoInput.css**

.todo-input {

display: flex;

justify-content: center;

margin-bottom: 20px;

}

.todo-input input {

width: 70%;

padding: 10px;

font-size: 14px;

}

.todo-input button {

padding: 10px 15px;

margin-left: 10px;

cursor: pointer;

background-color: #4caf50;

color: white;

border: none;

}

**TodoList.css**

.todo-list {

margin-top: 10px;

}

**TodoItem.css**

.todo-item {

display: flex;

justify-content: space-between;

align-items: center;

background-color: #f9f9f9;

padding: 10px;

margin: 5px 0;

border: 1px solid #ddd;

}

.todo-item button {

padding: 5px 10px;

background-color: red;

color: white;

border: none;

cursor: pointer;

}

**Exercise 7: User Form with Preview**

**Task:**

* Create a form to collect user data with the following components:
  + **UserForm**: Collects user data (name, email, age).
  + **UserPreview**: Displays live preview of the entered data.
  + **SubmitButton**: Allows submission of data.

**Solution:**

**UserApp.js**

import React, { useState } from "react";

import UserForm from "./UserForm";

import UserPreview from "./UserPreview";

import SubmitButton from "./SubmitButton";

import "./UserApp.css";

const UserApp = () => {

const [userData, setUserData] = useState({

name: "",

email: "",

age: "",

});

const handleSubmit = () => {

console.log("User Data Submitted:", userData);

};

return (

<div className="user-app">

<h1>User Form</h1>

<UserForm userData={userData} setUserData={setUserData} />

<UserPreview userData={userData} />

<SubmitButton onSubmit={handleSubmit} />

</div>

);

};

export default UserApp;

**UserForm.js**

import React from "react";

import "./UserForm.css";

const UserForm = ({ userData, setUserData }) => {

const handleChange = (e) => {

const { name, value } = e.target;

setUserData({ ...userData, [name]: value });

};

return (

<div className="user-form">

<input

type="text"

name="name"

placeholder="Name"

value={userData.name}

onChange={handleChange}

/>

<input

type="email"

name="email"

placeholder="Email"

value={userData.email}

onChange={handleChange}

/>

<input

type="number"

name="age"

placeholder="Age"

value={userData.age}

onChange={handleChange}

/>

</div>

);

};

export default UserForm;

**UserPreview.js**

import React from "react";

import "./UserPreview.css";

const UserPreview = ({ userData }) => {

return (

<div className="user-preview">

<h3>Preview</h3>

<p>Name: {userData.name || "N/A"}</p>

<p>Email: {userData.email || "N/A"}</p>

<p>Age: {userData.age || "N/A"}</p>

</div>

);

};

export default UserPreview;

**SubmitButton.js**

import React from "react";

import "./SubmitButton.css";

const SubmitButton = ({ onSubmit }) => {

return (

<button className="submit-btn" onClick={onSubmit}>

Submit

</button>

);

};

export default SubmitButton;

**UserApp.css**

.user-app {

text-align: center;

width: 400px;

margin: 20px auto;

}

**UserForm.css**

.user-form input {

display: block;

margin: 10px auto;

padding: 8px;

width: 90%;

}

**UserPreview.css**

.user-preview {

margin-top: 20px;

background-color: #f9f9f9;

padding: 10px;

border: 1px solid #ddd;

}

**SubmitButton.css**

.submit-btn {

padding: 10px 20px;

background-color: #4caf50;

color: white;

border: none;

cursor: pointer;

}

**Exercise 8: Form Validation Application**

**Task**:

* Create a form with fields: Name, Email, and Password.
* Include three components: Form, Field, and ValidationMessage.
* Use state to track inputs and show validation messages for required fields.

**Components**

**File: Field.js**

import React from "react";

const Field = ({ label, value, onChange, type = "text", error }) => {

return (

<div className="form-field">

<label>{label}</label>

<input type={type} value={value} onChange={onChange} />

{error && <span className="error">{error}</span>}

</div>

);

};

export default Field;

**File: ValidationMessage.js**

import React from "react";

const ValidationMessage = ({ message }) => {

return <p className="validation-message">{message}</p>;

};

export default ValidationMessage;

**File: Form.js**

import React, { useState } from "react";

import Field from "./Field";

import ValidationMessage from "./ValidationMessage";

const Form = () => {

const [name, setName] = useState("");

const [email, setEmail] = useState("");

const [password, setPassword] = useState("");

const [errors, setErrors] = useState({});

const handleSubmit = (e) => {

e.preventDefault();

let formErrors = {};

if (!name) formErrors.name = "Name is required.";

if (!email) formErrors.email = "Email is required.";

if (!password) formErrors.password = "Password is required.";

setErrors(formErrors);

if (Object.keys(formErrors).length === 0) {

alert("Form submitted successfully!");

}

};

return (

<form onSubmit={handleSubmit} className="form">

<Field

label="Name"

value={name}

onChange={(e) => setName(e.target.value)}

error={errors.name}

/>

<Field

label="Email"

value={email}

onChange={(e) => setEmail(e.target.value)}

error={errors.email}

/>

<Field

label="Password"

type="password"

value={password}

onChange={(e) => setPassword(e.target.value)}

error={errors.password}

/>

<button type="submit">Submit</button>

<ValidationMessage message="Please fill out all required fields." />

</form>

);

};

export default Form;

**File: App.js**

import React from "react";

import Form from "./components/Form";

const App = () => {

return (

<div>

<h1>Form Validation Application</h1>

<Form />

</div>

);

};

export default App;

**File: styles.css**

.form {

width: 300px;

margin: 20px auto;

}

.form-field {

margin-bottom: 15px;

}

label {

display: block;

font-weight: bold;

margin-bottom: 5px;

}

input {

width: 100%;

padding: 8px;

box-sizing: border-box;

}

button {

padding: 10px 15px;

background-color: #4caf50;

color: white;

border: none;

cursor: pointer;

}

button:hover {

background-color: #45a049;

}

.error {

color: red;

font-size: 12px;

}

.validation-message {

font-size: 14px;

color: #555;

margin-top: 10px;

}

**Exercise 9: Multi-Step Form Application**

**Task**:  
Create a multi-step form with three steps:

1. **Step 1**: Collect personal information (First Name, Last Name, Age).
2. **Step 2**: Collect account details (Email, Password).
3. **Step 3**: Display confirmation of the entered details.

**Navigation**:

* Use Next and Back buttons to navigate between steps.
* Include validation for required fields before moving to the next step.

**Components**

**File: Step1.js**

import React from "react";

const Step1 = ({ formData, handleChange }) => {

return (

<div>

<h2>Step 1: Personal Info</h2>

<div className="form-field">

<label>First Name:</label>

<input

type="text"

name="firstName"

value={formData.firstName}

onChange={handleChange}

/>

</div>

<div className="form-field">

<label>Last Name:</label>

<input

type="text"

name="lastName"

value={formData.lastName}

onChange={handleChange}

/>

</div>

<div className="form-field">

<label>Age:</label>

<input

type="number"

name="age"

value={formData.age}

onChange={handleChange}

/>

</div>

</div>

);

};

export default Step1;

**File: Step2.js**

import React from "react";

const Step2 = ({ formData, handleChange }) => {

return (

<div>

<h2>Step 2: Account Details</h2>

<div className="form-field">

<label>Email:</label>

<input

type="email"

name="email"

value={formData.email}

onChange={handleChange}

/>

</div>

<div className="form-field">

<label>Password:</label>

<input

type="password"

name="password"

value={formData.password}

onChange={handleChange}

/>

</div>

</div>

);

};

export default Step2;

**File: Step3.js**

import React from "react";

const Step3 = ({ formData }) => {

return (

<div>

<h2>Step 3: Confirmation</h2>

<ul>

<li>First Name: {formData.firstName}</li>

<li>Last Name: {formData.lastName}</li>

<li>Age: {formData.age}</li>

<li>Email: {formData.email}</li>

</ul>

<p>Please confirm your details before submitting.</p>

</div>

);

};

export default Step3;

**File: MultiStepForm.js**

import React, { useState } from "react";

import Step1 from "./Step1";

import Step2 from "./Step2";

import Step3 from "./Step3";

const MultiStepForm = () => {

const [step, setStep] = useState(1);

const [formData, setFormData] = useState({

firstName: "",

lastName: "",

age: "",

email: "",

password: "",

});

const handleChange = (e) => {

setFormData({ ...formData, [e.target.name]: e.target.value });

};

const nextStep = () => {

if (validateStep(step)) setStep(step + 1);

};

const prevStep = () => setStep(step - 1);

const validateStep = (currentStep) => {

if (currentStep === 1) {

if (!formData.firstName || !formData.lastName || !formData.age) {

alert("Please fill all fields in Step 1.");

return false;

}

}

if (currentStep === 2) {

if (!formData.email || !formData.password) {

alert("Please fill all fields in Step 2.");

return false;

}

}

return true;

};

return (

<div className="multi-step-form">

{step === 1 && <Step1 formData={formData} handleChange={handleChange} />}

{step === 2 && <Step2 formData={formData} handleChange={handleChange} />}

{step === 3 && <Step3 formData={formData} />}

<div className="navigation-buttons">

{step > 1 && <button onClick={prevStep}>Back</button>}

{step < 3 && <button onClick={nextStep}>Next</button>}

{step === 3 && <button onClick={() => alert("Form Submitted!")}>Submit</button>}

</div>

</div>

);

};

export default MultiStepForm;

**File: App.js**

import React from "react";

import MultiStepForm from "./components/MultiStepForm";

const App = () => {

return (

<div>

<h1>Multi-Step Form Application</h1>

<MultiStepForm />

</div>

);

};

export default App;

**File: styles.css**

.multi-step-form {

width: 400px;

margin: 20px auto;

border: 1px solid #ccc;

padding: 20px;

border-radius: 5px;

background-color: #f9f9f9;

}

h2 {

color: #333;

margin-bottom: 15px;

}

.form-field {

margin-bottom: 15px;

}

label {

display: block;

font-weight: bold;

margin-bottom: 5px;

}

input {

width: 100%;

padding: 8px;

box-sizing: border-box;

}

.navigation-buttons button {

margin-right: 10px;

padding: 10px 15px;

background-color: #4caf50;

color: white;

border: none;

cursor: pointer;

}

.navigation-buttons button:hover {

background-color: #45a049;

}

**Exercise 10: To-Do List with Filters**

**Task**:  
Create a to-do list application with the following features:

* Add new tasks.
* Mark tasks as completed.
* Filter tasks by:
  + All Tasks.
  + Completed Tasks.
  + Pending Tasks.

**Components**:

1. **TodoInput**: For adding new tasks.
2. **TodoList**: For displaying the list of tasks.
3. **TodoFilters**: For filtering tasks.

**File: TodoInput.js**

import React, { useState } from "react";

const TodoInput = ({ addTask }) => {

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

const handleAddTask = () => {

if (task.trim() !== "") {

addTask(task);

setTask("");

} else {

alert("Task cannot be empty!");

}

};

return (

<div className="todo-input">

<input

type="text"

placeholder="Add a new task"

value={task}

onChange={(e) => setTask(e.target.value)}

/>

<button onClick={handleAddTask}>Add Task</button>

</div>

);

};

export default TodoInput;

**File: TodoList.js**

import React from "react";

const TodoList = ({ tasks, toggleTask }) => {

return (

<ul className="todo-list">

{tasks.map((task) => (

<li

key={task.id}

onClick={() => toggleTask(task.id)}

className={task.completed ? "completed" : ""}

>

{task.text}

</li>

))}

</ul>

);

};

export default TodoList;

**File: TodoFilters.js**

import React from "react";

const TodoFilters = ({ filter, setFilter }) => {

return (

<div className="todo-filters">

<button

className={filter === "all" ? "active" : ""}

onClick={() => setFilter("all")}

>

All

</button>

<button

className={filter === "completed" ? "active" : ""}

onClick={() => setFilter("completed")}

>

Completed

</button>

<button

className={filter === "pending" ? "active" : ""}

onClick={() => setFilter("pending")}

>

Pending

</button>

</div>

);

};

export default TodoFilters;

**File: TodoApp.js**

import React, { useState } from "react";

import TodoInput from "./TodoInput";

import TodoList from "./TodoList";

import TodoFilters from "./TodoFilters";

const TodoApp = () => {

const [tasks, setTasks] = useState([]);

const [filter, setFilter] = useState("all");

const addTask = (text) => {

setTasks([...tasks, { id: Date.now(), text, completed: false }]);

};

const toggleTask = (id) => {

setTasks(

tasks.map((task) =>

task.id === id ? { ...task, completed: !task.completed } : task

)

);

};

const getFilteredTasks = () => {

if (filter === "completed") {

return tasks.filter((task) => task.completed);

}

if (filter === "pending") {

return tasks.filter((task) => !task.completed);

}

return tasks;

};

return (

<div className="todo-app">

<h2>To-Do List</h2>

<TodoInput addTask={addTask} />

<TodoFilters filter={filter} setFilter={setFilter} />

<TodoList tasks={getFilteredTasks()} toggleTask={toggleTask} />

</div>

);

};

export default TodoApp;

**File: App.js**

import React from "react";

import TodoApp from "./components/TodoApp";

const App = () => {

return (

<div>

<h1>React To-Do Application</h1>

<TodoApp />

</div>

);

};

export default App;

**File: styles.css**

.todo-app {

width: 400px;

margin: 20px auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

background-color: #f9f9f9;

}

h2 {

text-align: center;

color: #333;

}

.todo-input {

display: flex;

margin-bottom: 15px;

}

.todo-input input {

flex: 1;

padding: 10px;

border: 1px solid #ccc;

border-radius: 4px;

}

.todo-input button {

padding: 10px;

border: none;

background-color: #4caf50;

color: white;

cursor: pointer;

}

.todo-input button:hover {

background-color: #45a049;

}

.todo-list {

list-style: none;

padding: 0;

}

.todo-list li {

padding: 10px;

border-bottom: 1px solid #ddd;

cursor: pointer;

}

.todo-list li.completed {

text-decoration: line-through;

color: #888;

}

.todo-filters {

display: flex;

justify-content: space-around;

margin: 10px 0;

}

.todo-filters button {

padding: 10px;

border: none;

background-color: #ddd;

cursor: pointer;

}

.todo-filters button.active {

background-color: #4caf50;

color: white;

}

**Exercise 11: Recipe Manager**

**Task**:  
Build a Recipe Manager app that allows users to:

* Add new recipes with a title and ingredients.
* Display a list of recipes.
* View detailed information about a selected recipe.

**Components**:

1. **RecipeInput**: For adding new recipes.
2. **RecipeList**: Displays a list of recipes.
3. **RecipeDetails**: Shows the details of the selected recipe.

**File: RecipeInput.js**

import React, { useState } from "react";

const RecipeInput = ({ addRecipe }) => {

const [title, setTitle] = useState("");

const [ingredients, setIngredients] = useState("");

const handleAddRecipe = () => {

if (title.trim() && ingredients.trim()) {

addRecipe({ title, ingredients });

setTitle("");

setIngredients("");

} else {

alert("Please fill in all fields!");

}

};

return (

<div className="recipe-input">

<input

type="text"

placeholder="Recipe Title"

value={title}

onChange={(e) => setTitle(e.target.value)}

/>

<textarea

placeholder="Ingredients (comma separated)"

value={ingredients}

onChange={(e) => setIngredients(e.target.value)}

></textarea>

<button onClick={handleAddRecipe}>Add Recipe</button>

</div>

);

};

export default RecipeInput;

**File: RecipeList.js**

import React from "react";

const RecipeList = ({ recipes, selectRecipe }) => {

return (

<div className="recipe-list">

{recipes.map((recipe, index) => (

<div

key={index}

className="recipe-item"

onClick={() => selectRecipe(index)}

>

{recipe.title}

</div>

))}

</div>

);

};

export default RecipeList;

**File: RecipeDetails.js**

import React from "react";

const RecipeDetails = ({ recipe }) => {

if (!recipe) {

return <div className="recipe-details">Select a recipe to view details</div>;

}

return (

<div className="recipe-details">

<h3>{recipe.title}</h3>

<p>Ingredients:</p>

<ul>

{recipe.ingredients.split(",").map((ingredient, index) => (

<li key={index}>{ingredient.trim()}</li>

))}

</ul>

</div>

);

};

export default RecipeDetails;

**File: RecipeApp.js**

import React, { useState } from "react";

import RecipeInput from "./RecipeInput";

import RecipeList from "./RecipeList";

import RecipeDetails from "./RecipeDetails";

const RecipeApp = () => {

const [recipes, setRecipes] = useState([]);

const [selectedRecipe, setSelectedRecipe] = useState(null);

const addRecipe = (recipe) => {

setRecipes([...recipes, recipe]);

};

return (

<div className="recipe-app">

<h2>Recipe Manager</h2>

<RecipeInput addRecipe={addRecipe} />

<div className="recipe-container">

<RecipeList recipes={recipes} selectRecipe={(index) => setSelectedRecipe(recipes[index])} />

<RecipeDetails recipe={selectedRecipe} />

</div>

</div>

);

};

export default RecipeApp;

**File: App.js**

import React from "react";

import RecipeApp from "./components/RecipeApp";

const App = () => {

return (

<div>

<h1>React Recipe Application</h1>

<RecipeApp />

</div>

);

};

export default App;

**File: styles.css**

.recipe-app {

width: 600px;

margin: 20px auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

background-color: #f9f9f9;

}

.recipe-input {

margin-bottom: 20px;

}

.recipe-input input,

.recipe-input textarea {

width: 100%;

margin: 5px 0;

padding: 10px;

border: 1px solid #ccc;

border-radius: 4px;

}

.recipe-input button {

display: block;

width: 100%;

padding: 10px;

border: none;

background-color: #4caf50;

color: white;

cursor: pointer;

}

.recipe-input button:hover {

background-color: #45a049;

}

.recipe-container {

display: flex;

justify-content: space-between;

}

.recipe-list {

width: 40%;

}

.recipe-item {

padding: 10px;

border: 1px solid #ddd;

margin-bottom: 5px;

cursor: pointer;

background-color: #f1f1f1;

}

.recipe-item:hover {

background-color: #e0e0e0;

}

.recipe-details {

width: 55%;

padding: 10px;

border: 1px solid #ddd;

background-color: #f1f1f1;

}

**Exercise 12: Library Management System**

**Task**:  
Create a Library Management System that allows users to:

* View a list of available books.
* Borrow a book (mark it as borrowed).
* Add a new book to the library.

**Components**:

1. **BookList**: Displays the list of books with a "Borrow" button for each.
2. **AddBookForm**: Allows adding a new book to the list.
3. **BorrowedBooks**: Displays a list of borrowed books.

**File: BookList.js**

import React from "react";

const BookList = ({ books, borrowBook }) => {

return (

<div className="book-list">

<h3>Available Books</h3>

<ul>

{books.map((book, index) => (

<li key={index} className={book.isBorrowed ? "borrowed" : ""}>

{book.title} by {book.author}

{!book.isBorrowed && (

<button onClick={() => borrowBook(index)}>Borrow</button>

)}

</li>

))}

</ul>

</div>

);

};

export default BookList;

**File: AddBookForm.js**

import React, { useState } from "react";

const AddBookForm = ({ addBook }) => {

const [title, setTitle] = useState("");

const [author, setAuthor] = useState("");

const handleAddBook = () => {

if (title.trim() && author.trim()) {

addBook({ title, author, isBorrowed: false });

setTitle("");

setAuthor("");

} else {

alert("Please fill in both fields!");

}

};

return (

<div className="add-book-form">

<h3>Add a New Book</h3>

<input

type="text"

placeholder="Book Title"

value={title}

onChange={(e) => setTitle(e.target.value)}

/>

<input

type="text"

placeholder="Author"

value={author}

onChange={(e) => setAuthor(e.target.value)}

/>

<button onClick={handleAddBook}>Add Book</button>

</div>

);

};

export default AddBookForm;

**File: BorrowedBooks.js**

import React from "react";

const BorrowedBooks = ({ books }) => {

const borrowedBooks = books.filter((book) => book.isBorrowed);

return (

<div className="borrowed-books">

<h3>Borrowed Books</h3>

<ul>

{borrowedBooks.length > 0 ? (

borrowedBooks.map((book, index) => (

<li key={index}>

{book.title} by {book.author}

</li>

))

) : (

<p>No books borrowed yet.</p>

)}

</ul>

</div>

);

};

export default BorrowedBooks;

**File: LibraryApp.js**

import React, { useState } from "react";

import BookList from "./BookList";

import AddBookForm from "./AddBookForm";

import BorrowedBooks from "./BorrowedBooks";

const LibraryApp = () => {

const [books, setBooks] = useState([

{ title: "1984", author: "George Orwell", isBorrowed: false },

{ title: "To Kill a Mockingbird", author: "Harper Lee", isBorrowed: false },

{ title: "Moby-Dick", author: "Herman Melville", isBorrowed: false },

]);

const addBook = (book) => {

setBooks([...books, book]);

};

const borrowBook = (index) => {

const updatedBooks = [...books];

updatedBooks[index].isBorrowed = true;

setBooks(updatedBooks);

};

return (

<div className="library-app">

<h2>Library Management System</h2>

<AddBookForm addBook={addBook} />

<div className="library-container">

<BookList books={books} borrowBook={borrowBook} />

<BorrowedBooks books={books} />

</div>

</div>

);

};

export default LibraryApp;

**File: App.js**

import React from "react";

import LibraryApp from "./components/LibraryApp";

const App = () => {

return (

<div>

<h1>React Library Application</h1>

<LibraryApp />

</div>

);

};

export default App;

**File: styles.css**

.library-app {

width: 700px;

margin: 20px auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

background-color: #fdfdfd;

}

.add-book-form input {

margin: 5px 0;

padding: 8px;

width: calc(50% - 10px);

border: 1px solid #ccc;

border-radius: 4px;

}

.add-book-form button {

margin-top: 10px;

padding: 10px;

border: none;

background-color: #007bff;

color: white;

cursor: pointer;

}

.add-book-form button:hover {

background-color: #0056b3;

}

.library-container {

display: flex;

justify-content: space-between;

}

.book-list ul,

.borrowed-books ul {

list-style: none;

padding: 0;

}

.book-list li,

.borrowed-books li {

padding: 10px;

border-bottom: 1px solid #ddd;

}

.borrowed {

text-decoration: line-through;

color: #888;

}

**Exercise 13: Task Management Application**

**Task**:  
Create a Task Management Application with the following features:

* Add tasks with a description and priority level.
* Mark tasks as completed.
* Filter tasks based on their priority.

**Components**:

1. **TaskList**: Displays the list of tasks with options to mark them as completed.
2. **AddTaskForm**: Allows adding a new task with a priority level.
3. **TaskFilter**: Allows filtering tasks based on priority level.

**File: TaskList.js**

import React from "react";

const TaskList = ({ tasks, markComplete }) => {

return (

<div className="task-list">

<h3>Task List</h3>

<ul>

{tasks.map((task, index) => (

<li

key={index}

className={task.isCompleted ? "completed" : ""}

>

<span>

{task.description} - <b>{task.priority}</b>

</span>

{!task.isCompleted && (

<button onClick={() => markComplete(index)}>Complete</button>

)}

</li>

))}

</ul>

</div>

);

};

export default TaskList;

**File: AddTaskForm.js**

import React, { useState } from "react";

const AddTaskForm = ({ addTask }) => {

const [description, setDescription] = useState("");

const [priority, setPriority] = useState("Low");

const handleAddTask = () => {

if (description.trim()) {

addTask({ description, priority, isCompleted: false });

setDescription("");

setPriority("Low");

} else {

alert("Please enter a task description!");

}

};

return (

<div className="add-task-form">

<h3>Add a Task</h3>

<input

type="text"

placeholder="Task Description"

value={description}

onChange={(e) => setDescription(e.target.value)}

/>

<select

value={priority}

onChange={(e) => setPriority(e.target.value)}

>

<option value="Low">Low</option>

<option value="Medium">Medium</option>

<option value="High">High</option>

</select>

<button onClick={handleAddTask}>Add Task</button>

</div>

);

};

export default AddTaskForm;

**File: TaskFilter.js**

import React from "react";

const TaskFilter = ({ setFilterPriority }) => {

return (

<div className="task-filter">

<h3>Filter Tasks</h3>

<select onChange={(e) => setFilterPriority(e.target.value)}>

<option value="All">All</option>

<option value="Low">Low</option>

<option value="Medium">Medium</option>

<option value="High">High</option>

</select>

</div>

);

};

export default TaskFilter;

**File: TaskManager.js**

import React, { useState } from "react";

import TaskList from "./TaskList";

import AddTaskForm from "./AddTaskForm";

import TaskFilter from "./TaskFilter";

const TaskManager = () => {

const [tasks, setTasks] = useState([]);

const [filterPriority, setFilterPriority] = useState("All");

const addTask = (task) => {

setTasks([...tasks, task]);

};

const markComplete = (index) => {

const updatedTasks = [...tasks];

updatedTasks[index].isCompleted = true;

setTasks(updatedTasks);

};

const filteredTasks =

filterPriority === "All"

? tasks

: tasks.filter((task) => task.priority === filterPriority);

return (

<div className="task-manager">

<h2>Task Management Application</h2>

<AddTaskForm addTask={addTask} />

<TaskFilter setFilterPriority={setFilterPriority} />

<TaskList tasks={filteredTasks} markComplete={markComplete} />

</div>

);

};

export default TaskManager;

**File: App.js**

import React from "react";

import TaskManager from "./components/TaskManager";

const App = () => {

return (

<div>

<h1>React Task Management</h1>

<TaskManager />

</div>

);

};

export default App;

**File: styles.css**

.task-manager {

width: 600px;

margin: 20px auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

background-color: #f9f9f9;

}

.add-task-form input,

.add-task-form select {

margin: 5px 0;

padding: 8px;

width: calc(50% - 10px);

border: 1px solid #ccc;

border-radius: 4px;

}

.add-task-form button {

margin-top: 10px;

padding: 10px;

border: none;

background-color: #28a745;

color: white;

cursor: pointer;

}

.add-task-form button:hover {

background-color: #218838;

}

.task-list ul {

list-style: none;

padding: 0;

}

.task-list li {

padding: 10px;

border-bottom: 1px solid #ddd;

display: flex;

justify-content: space-between;

}

.task-list li.completed {

text-decoration: line-through;

color: #888;

}

.task-filter select {

padding: 8px;

border: 1px solid #ccc;

border-radius: 4px;

}

**Exercise 14: Expense Tracker Application**

**Task**:  
Create an Expense Tracker Application with the following features:

* Add new expenses with a category and amount.
* Display a summary of expenses by category.
* Allow filtering of expenses by category.

**Components**:

1. **ExpenseList**: Displays the list of expenses and their categories.
2. **AddExpenseForm**: Allows adding a new expense with its category and amount.
3. **ExpenseSummary**: Displays the total expenses for each category.

**File: ExpenseList.js**

import React from "react";

const ExpenseList = ({ expenses, filterCategory }) => {

const filteredExpenses =

filterCategory === "All"

? expenses

: expenses.filter((expense) => expense.category === filterCategory);

return (

<div className="expense-list">

<h3>Expenses</h3>

<ul>

{filteredExpenses.map((expense, index) => (

<li key={index}>

<span>{expense.description}</span> - <b>${expense.amount}</b> (<i>{expense.category}</i>)

</li>

))}

</ul>

</div>

);

};

export default ExpenseList;

**File: AddExpenseForm.js**

import React, { useState } from "react";

const AddExpenseForm = ({ addExpense }) => {

const [description, setDescription] = useState("");

const [amount, setAmount] = useState("");

const [category, setCategory] = useState("Food");

const handleAddExpense = () => {

if (description.trim() && amount > 0) {

addExpense({ description, amount: parseFloat(amount), category });

setDescription("");

setAmount("");

setCategory("Food");

} else {

alert("Please enter a valid description and amount!");

}

};

return (

<div className="add-expense-form">

<h3>Add an Expense</h3>

<input

type="text"

placeholder="Expense Description"

value={description}

onChange={(e) => setDescription(e.target.value)}

/>

<input

type="number"

placeholder="Amount"

value={amount}

onChange={(e) => setAmount(e.target.value)}

/>

<select

value={category}

onChange={(e) => setCategory(e.target.value)}

>

<option value="Food">Food</option>

<option value="Travel">Travel</option>

<option value="Shopping">Shopping</option>

<option value="Bills">Bills</option>

</select>

<button onClick={handleAddExpense}>Add Expense</button>

</div>

);

};

export default AddExpenseForm;

**File: ExpenseSummary.js**

import React from "react";

const ExpenseSummary = ({ expenses }) => {

const summary = expenses.reduce((acc, expense) => {

acc[expense.category] = (acc[expense.category] || 0) + expense.amount;

return acc;

}, {});

return (

<div className="expense-summary">

<h3>Expense Summary</h3>

<ul>

{Object.keys(summary).map((category) => (

<li key={category}>

{category}: <b>${summary[category].toFixed(2)}</b>

</li>

))}

</ul>

</div>

);

};

export default ExpenseSummary;

**File: ExpenseTracker.js**

import React, { useState } from "react";

import ExpenseList from "./ExpenseList";

import AddExpenseForm from "./AddExpenseForm";

import ExpenseSummary from "./ExpenseSummary";

const ExpenseTracker = () => {

const [expenses, setExpenses] = useState([]);

const [filterCategory, setFilterCategory] = useState("All");

const addExpense = (expense) => {

setExpenses([...expenses, expense]);

};

return (

<div className="expense-tracker">

<h2>Expense Tracker Application</h2>

<AddExpenseForm addExpense={addExpense} />

<ExpenseSummary expenses={expenses} />

<div className="filter-expenses">

<h4>Filter Expenses</h4>

<select

onChange={(e) => setFilterCategory(e.target.value)}

>

<option value="All">All</option>

<option value="Food">Food</option>

<option value="Travel">Travel</option>

<option value="Shopping">Shopping</option>

<option value="Bills">Bills</option>

</select>

</div>

<ExpenseList

expenses={expenses}

filterCategory={filterCategory}

/>

</div>

);

};

export default ExpenseTracker;

**File: App.js**

import React from "react";

import ExpenseTracker from "./components/ExpenseTracker";

const App = () => {

return (

<div>

<h1>React Expense Tracker</h1>

<ExpenseTracker />

</div>

);

};

export default App;

**File: styles.css**

.expense-tracker {

width: 600px;

margin: 20px auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

background-color: #f9f9f9;

}

.add-expense-form input,

.add-expense-form select {

margin: 5px 0;

padding: 8px;

width: calc(50% - 10px);

border: 1px solid #ccc;

border-radius: 4px;

}

.add-expense-form button {

margin-top: 10px;

padding: 10px;

border: none;

background-color: #007bff;

color: white;

cursor: pointer;

}

.add-expense-form button:hover {

background-color: #0056b3;

}

.expense-list ul {

list-style: none;

padding: 0;

}

.expense-list li {

padding: 10px;

border-bottom: 1px solid #ddd;

display: flex;

justify-content: space-between;

}

.expense-summary ul {

list-style: none;

padding: 0;

}

**Exercise 15: Task Management Application**

**Task**:  
Create a Task Management Application with the following features:

* Users can add tasks with a title, description, and due date.
* Tasks can be marked as completed.
* Tasks can be edited or deleted.
* Tasks are categorized (e.g., "Work", "Personal").
* Filter tasks based on category and completion status.
* Display a summary of total tasks and completed tasks.

**Components**:

1. **TaskList**: Displays the list of tasks.
2. **TaskItem**: Represents each task in the list.
3. **AddTaskForm**: Form for adding new tasks.
4. **TaskSummary**: Displays the count of total tasks and completed tasks.
5. **TaskFilter**: Filters tasks based on category and completion status.
6. **CategoryFilter**: Allows filtering tasks by category.
7. **CompletionFilter**: Allows filtering tasks by completion status.
8. **TaskEditForm**: Form to edit an existing task.
9. **DeleteTaskButton**: Button to delete a task.
10. **CompleteTaskButton**: Button to mark a task as completed.

**File: TaskItem.js**

import React from "react";

import CompleteTaskButton from "./CompleteTaskButton";

import DeleteTaskButton from "./DeleteTaskButton";

const TaskItem = ({ task, onComplete, onDelete, onEdit }) => {

return (

<div className="task-item">

<div>

<h4>{task.title}</h4>

<p>{task.description}</p>

<p><i>{task.dueDate}</i></p>

</div>

<div>

<CompleteTaskButton task={task} onComplete={onComplete} />

<button onClick={() => onEdit(task)}>Edit</button>

<DeleteTaskButton task={task} onDelete={onDelete} />

</div>

</div>

);

};

export default TaskItem;

**File: CompleteTaskButton.js**

import React from "react";

const CompleteTaskButton = ({ task, onComplete }) => {

return (

<button

style={{ backgroundColor: task.completed ? "green" : "orange" }}

onClick={() => onComplete(task.id)}

>

{task.completed ? "Completed" : "Mark as Completed"}

</button>

);

};

export default CompleteTaskButton;

**File: DeleteTaskButton.js**

import React from "react";

const DeleteTaskButton = ({ task, onDelete }) => {

return (

<button

style={{ backgroundColor: "red" }}

onClick={() => onDelete(task.id)}

>

Delete

</button>

);

};

export default DeleteTaskButton;

**File: TaskList.js**

import React from "react";

import TaskItem from "./TaskItem";

const TaskList = ({ tasks, onComplete, onDelete, onEdit }) => {

return (

<div className="task-list">

{tasks.map((task) => (

<TaskItem

key={task.id}

task={task}

onComplete={onComplete}

onDelete={onDelete}

onEdit={onEdit}

/>

))}

</div>

);

};

export default TaskList;

**File: AddTaskForm.js**

import React, { useState } from "react";

const AddTaskForm = ({ addTask }) => {

const [title, setTitle] = useState("");

const [description, setDescription] = useState("");

const [dueDate, setDueDate] = useState("");

const [category, setCategory] = useState("Work");

const handleSubmit = () => {

if (title && description && dueDate) {

addTask({ title, description, dueDate, category, completed: false });

setTitle("");

setDescription("");

setDueDate("");

setCategory("Work");

} else {

alert("Please fill in all fields.");

}

};

return (

<div className="add-task-form">

<h3>Add New Task</h3>

<input

type="text"

placeholder="Task Title"

value={title}

onChange={(e) => setTitle(e.target.value)}

/>

<textarea

placeholder="Task Description"

value={description}

onChange={(e) => setDescription(e.target.value)}

/>

<input

type="date"

value={dueDate}

onChange={(e) => setDueDate(e.target.value)}

/>

<select value={category} onChange={(e) => setCategory(e.target.value)}>

<option value="Work">Work</option>

<option value="Personal">Personal</option>

</select>

<button onClick={handleSubmit}>Add Task</button>

</div>

);

};

export default AddTaskForm;

**File: TaskFilter.js**

import React from "react";

import CategoryFilter from "./CategoryFilter";

import CompletionFilter from "./CompletionFilter";

const TaskFilter = ({ setCategoryFilter, setCompletionFilter }) => {

return (

<div className="task-filter">

<CategoryFilter setCategoryFilter={setCategoryFilter} />

<CompletionFilter setCompletionFilter={setCompletionFilter} />

</div>

);

};

export default TaskFilter;

**File: CategoryFilter.js**

import React from "react";

const CategoryFilter = ({ setCategoryFilter }) => {

return (

<div className="category-filter">

<h4>Filter by Category</h4>

<select onChange={(e) => setCategoryFilter(e.target.value)}>

<option value="All">All</option>

<option value="Work">Work</option>

<option value="Personal">Personal</option>

</select>

</div>

);

};

export default CategoryFilter;

**File: CompletionFilter.js**

import React from "react";

const CompletionFilter = ({ setCompletionFilter }) => {

return (

<div className="completion-filter">

<h4>Filter by Completion</h4>

<select onChange={(e) => setCompletionFilter(e.target.value)}>

<option value="All">All</option>

<option value="Completed">Completed</option>

<option value="Pending">Pending</option>

</select>

</div>

);

};

export default CompletionFilter;

**File: TaskEditForm.js**

import React, { useState } from "react";

const TaskEditForm = ({ task, updateTask }) => {

const [title, setTitle] = useState(task.title);

const [description, setDescription] = useState(task.description);

const [dueDate, setDueDate] = useState(task.dueDate);

const [category, setCategory] = useState(task.category);

const handleSubmit = () => {

updateTask({ ...task, title, description, dueDate, category });

};

return (

<div className="edit-task-form">

<h3>Edit Task</h3>

<input

type="text"

value={title}

onChange={(e) => setTitle(e.target.value)}

/>

<textarea

value={description}

onChange={(e) => setDescription(e.target.value)}

/>

<input

type="date"

value={dueDate}

onChange={(e) => setDueDate(e.target.value)}

/>

<select value={category} onChange={(e) => setCategory(e.target.value)}>

<option value="Work">Work</option>

<option value="Personal">Personal</option>

</select>

<button onClick={handleSubmit}>Update Task</button>

</div>

);

};

export default TaskEditForm;

**File: TaskSummary.js**

import React from "react";

const TaskSummary = ({ tasks }) => {

const totalTasks = tasks.length;

const completedTasks = tasks.filter((task) => task.completed).length;

return (

<div className="task-summary">

<h3>Task Summary</h3>

<p>Total Tasks: {totalTasks}</p>

<p>Completed Tasks: {completedTasks}</p>

</div>

);

};

export default TaskSummary;

**File: App.js**

import React, { useState } from "react";

import AddTaskForm from "./components/AddTaskForm";

import TaskList from "./components/TaskList";

import TaskFilter from "./components/TaskFilter";

import TaskSummary from "./components/TaskSummary";

const App = () => {

const [tasks, setTasks] = useState([]);

const [categoryFilter, setCategoryFilter] = useState("All");

const [completionFilter, setCompletionFilter] = useState("All");

const addTask = (task) => {

setTasks([...tasks, { id: Date.now(), ...task }]);

};

const deleteTask = (id) => {

setTasks(tasks.filter((task) => task.id !== id));

};

const updateTask = (updatedTask) => {

setTasks(

tasks.map((task) =>

task.id === updatedTask.id ? updatedTask : task

)

);

};

const completeTask = (id) => {

setTasks(

tasks.map((task) =>

task.id === id ? { ...task, completed: !task.completed } : task

)

);

};

const filteredTasks = tasks.filter((task) => {

const matchesCategory =

categoryFilter === "All" || task.category === categoryFilter;

const matchesCompletion =

completionFilter === "All" ||

(completionFilter === "Completed" && task.completed) ||

(completionFilter === "Pending" && !task.completed);

return matchesCategory && matchesCompletion;

});

return (

<div className="task-manager">

<h1>Task Management App</h1>

<AddTaskForm addTask={addTask} />

<TaskSummary tasks={tasks} />

<TaskFilter

setCategoryFilter={setCategoryFilter}

setCompletionFilter={setCompletionFilter}

/>

<TaskList

tasks={filteredTasks}

onComplete={completeTask}

onDelete={deleteTask}

onEdit={updateTask}

/>

</div>

);

};

export default App;

**File: styles.css**

.task-manager {

width: 600px;

margin: 20px auto;

padding: 20px;

background-color: #f9f9f9;

border: 1px solid #ccc;

border-radius: 8px;

}

.add-task-form input,

.add-task-form textarea,

.add-task-form select {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 4px;

}

.add-task-form button {

padding: 10px;

background-color: #007bff;

color: white;

border: none;

cursor: pointer;

}

.add-task-form button:hover {

background-color: #0056b3;

}

.task-list {

margin-top: 20px;

}

.task-item {

display: flex;

justify-content: space-between;

padding: 10px;

border-bottom: 1px solid #ddd;

}

.task-item button {

margin-left: 5px;

}

.task-summary {

margin-top: 20px;

}

.task-summary p {

font-size: 1.1em;

}