**Student Have To Prepare Report In Format**

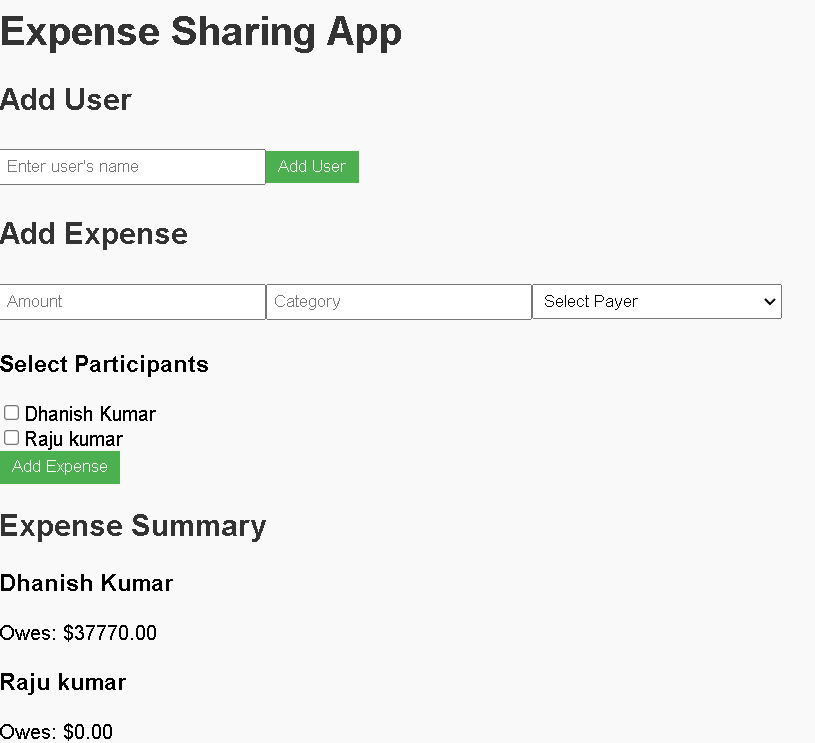
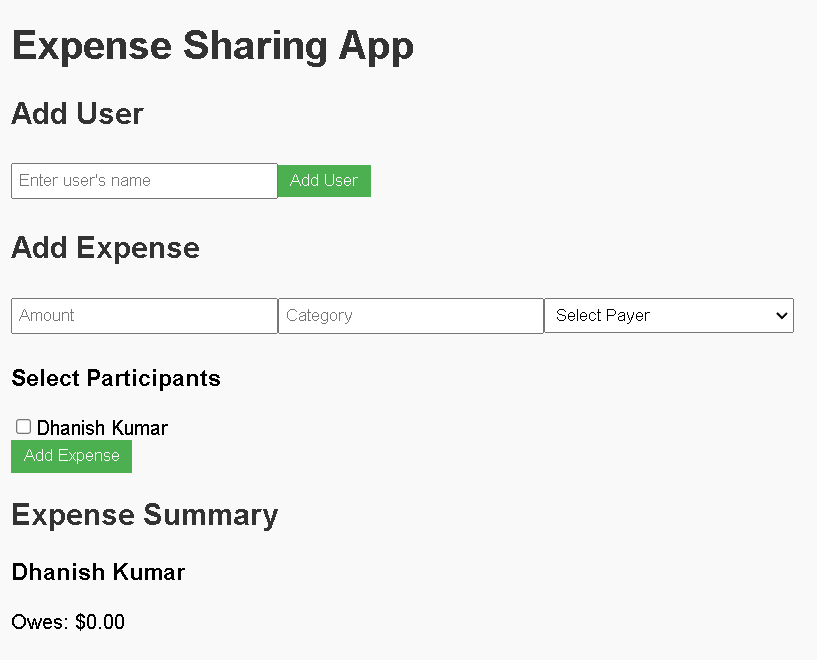
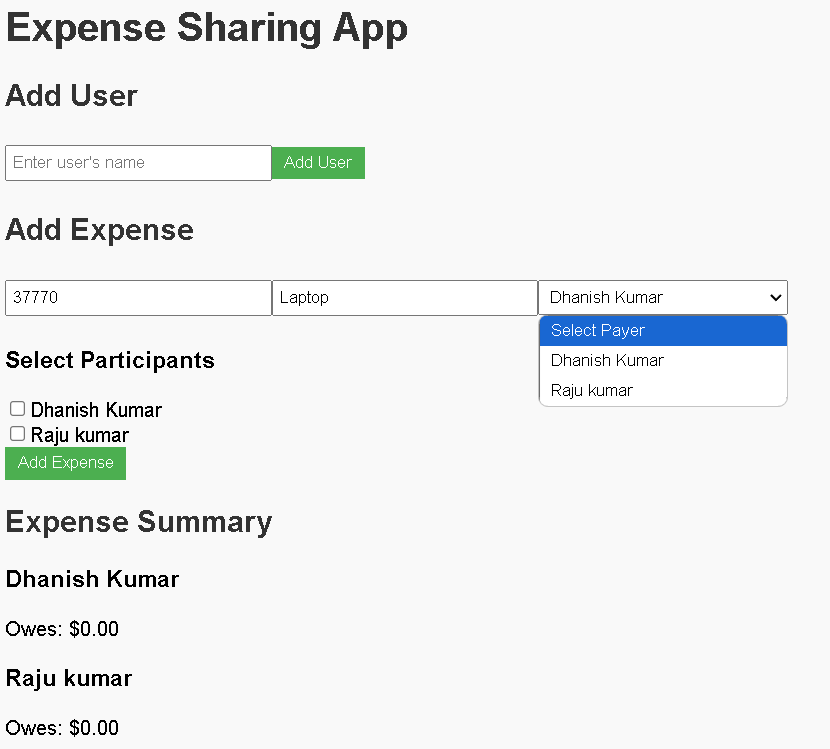
1. Add Task Description
2. Attach Screenshot Of Output.
3. Describe Widget/Algorithm Used In Task
4. Add Report In Your Task Zip File

**Sample Example :**

1. **Task Description**

The **Expense Sharing App** is built with a combination of modern front-end technologies, utilizing a variety of tools and libraries to manage the user interface, manage state, and enhance the user experience. Here's a breakdown of the key technologies used:

1. **Task Output Screenshot**



1. **Widget/Algorithm Used In Task**

**React (Library)**

**Purpose**: React is used as the core library for building the user interface (UI) of the app.

* **Component-based architecture**: React is based on components, which allow for reusability and easier management of the app's UI. Components like App, AddUser, AddExpense, and ExpenseSummary are each responsible for handling different pieces of functionality in the app.
* **Declarative UI**: React enables the declarative approach to building UIs, meaning you describe the UI in terms of components and their state. React takes care of efficiently updating and rendering the UI whenever the underlying state changes.
* **JSX**: React uses JSX (JavaScript XML) syntax, which allows you to write HTML-like code in JavaScript. This simplifies rendering UI elements based on dynamic data (like user inputs and expenses).

**2. CSS (Cascading Style Sheets)**

**Purpose**: CSS is used to style the user interface of the app.

* **Responsive design**: Basic styles are applied to ensure the app looks good on various screen sizes.
* **Component-level styling**: Styling is applied locally to each component (such as input fields, buttons, and expense lists) to create a clean and organized UI.

**Technologies Used in the Expense Sharing App**

The **Expense Sharing App** is built with a combination of modern front-end technologies, utilizing a variety of tools and libraries to manage the user interface, manage state, and enhance the user experience. Here's a breakdown of the key technologies used:

**1. React (Library)**

**Purpose**: React is used as the core library for building the user interface (UI) of the app.

* **Component-based architecture**: React is based on components, which allow for reusability and easier management of the app's UI. Components like App, AddUser, AddExpense, and ExpenseSummary are each responsible for handling different pieces of functionality in the app.
* **Declarative UI**: React enables the declarative approach to building UIs, meaning you describe the UI in terms of components and their state. React takes care of efficiently updating and rendering the UI whenever the underlying state changes.
* **JSX**: React uses JSX (JavaScript XML) syntax, which allows you to write HTML-like code in JavaScript. This simplifies rendering UI elements based on dynamic data (like user inputs and expenses).

**Key React Features Used**:

* **State Management** (useState): Used to manage and update the app's state for users, expenses, and balances.
* **Event Handling**: React handles form submissions and user interactions, such as adding users and expenses.

**2. CSS (Cascading Style Sheets)**

**Purpose**: CSS is used to style the user interface of the app.

* **Responsive design**: Basic styles are applied to ensure the app looks good on various screen sizes.
* **Component-level styling**: Styling is applied locally to each component (such as input fields, buttons, and expense lists) to create a clean and organized UI.

**Key Features**:

* **Flexbox**: CSS Flexbox is used to create flexible layouts that adapt to the screen size.
* **Hover effects**: Buttons and inputs have hover effects to improve user interaction.

**3. JavaScript (ES6+)**

**Purpose**: JavaScript is used for the logic and functionality of the app.

* **Arrow functions**: ES6 arrow functions are used to define concise functions, such as handling events (e.g., adding a user or expense).
* **Template literals**: Template literals are used to construct dynamic strings, such as error messages or dynamic component rendering.
* **Array methods**: Methods like .map(), .filter(), and .reduce() are used to iterate over users and expenses for rendering and calculations.

**4. useState (React Hook)**

**Purpose**: useState is a built-in React hook used to manage the state of the app. It is central to handling dynamic changes in the app, like adding users, expenses, and updating balances.

* **User State**: The app keeps track of a list of users in state using useState.
* **Expense State**: The app tracks expenses and stores them in state using useState.
* **Balance Calculation**: The balance for each user is calculated based on the expenses, and the result is stored and updated in the state.

**5. Helper Function (calculateBalances.js)**

**Purpose**: A utility function is used to calculate the balances (how much each user owes or is owed) after an expense is added.

* **Balance calculation**: The function takes in the list of users and expenses and computes how much each user should pay or receive based on the expenses.