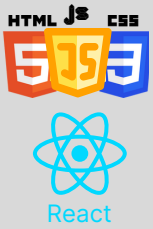
Assignment 3: ReactJS

Deadline
27th June, 23:59

Task **190 + 50 (Bonus)**

Designing an Expense Tracker WebApp



You have to create an expense tracker web app with HTML, CSS, JavaScript & ReactJS

Introduction

Ever felt like your wallet (or I must say e-wallet) was mysteriously lighter than expected? We've all been there, staring at an empty account balance and wondering where all the money went. It's a common challenge for college students like us, juggling expenses while trying to make ends meet.
That's why we're introducing **the Expense Tracker app** – your solution to easily stay on top of your finances. Design an expense tracker app to simplify expense management for yourself and your friends. Stay organized, make informed decisions, and achieve your financial goals with the Expense Tracker website(just kidding ;)). The website will have the following components (components, huh? What are they?):

1. <Navbar /> 20

a. The navbar component will contain navigation links between different sections of your app. (e.g., Home, Expenses List, Add Expense, etc.). **5**

b. Make the navbar responsive to different screen sizes (mobile, tablet, desktop). Show the hamburger menu on a mobile screen. **10**

c. Keep the navbar's position fixed at the top of the viewport. You can use 'position: fixed'(may cause some shifts in other components, so be careful) **5**

2. <HeroSection /> 30

The Hero Section in a web application is the first visible section that users see when they land on the website. Its purpose is to provide a captivating introduction, highlight key features or benefits, or set the tone for the rest of the website.

a. A typical hero section includes some text on the left part and some images/graphics on the right side(you are free to be creative with this section). 10

b. Ensure that the Hero Section is responsive. 10

c. Consider adding some simple animations or effects to elements within the Hero Section for a dynamic and engaging presentation(do not use any animation library). 10

3. <AddExpenseForm /> 50 + 20 (Bonus)

The Expense Form component in your Expense Tracker app allows users to add new expenses. The data is stored as a state variable. (see useState hook)

a. Include input fields for capturing expense details(text for description, numeric for amount), and a submit button. 20

b. Also, include a dropdown for the category. Examples of categories are stationary, food, etc.(include at least three). 10

c. Use Date() to get the time of adding expense. Store it along with expense details. 10

d. Disable the button if the form is invalid to prevent incomplete data submission. 5

e. Clear the form after successful submission. 5

<BONUS marks={20} >

The data gets lost when you reload the page. To resolve this, store this data in localStorage, and retrieve it when the page is loaded.

</BONUS>

4. <ExpenseList /> 90 + 30 (Bonus)

The Expense List component in your Expense Tracker app displays a list of all expenses added by the user.

a. Each expense item should include details such as description, amount, date, and category (or anything else that you have stored). 25

b. Provide an option for users to delete expenses from the list. Implement a delete button or icon next to each expense item for easy deletion. 15

c. Consider adding functionality to edit expense details. Include an edit button that allows users to modify the description or amount (One way would be to put data of selected expense items in the input form fields and delete the item). 15

d. Calculate and display the total expense amount at the bottom of the list. Update the total amount dynamically as users add or delete expenses. 10

e. Never forget responsiveness (like I forgot to add its score in 3rd component :|). 10

f. If you have stored a category of expense, how about a filter to get all expenses of a particular category? 15

<BONUS marks={30} >

Currently, all sections are on the same page. We can make different pages and add links from one to another. ReactJS offers a very nice way of implementing this. Use *react-router-dom* to implement the following routes. [See resource #1]
"/" or "/home" : Navbar + Hero section
"/track" : Navbar + Add Expense + Expense List
Any other route : Custom 404 Page

</BONUS>

<Resources>

- Youtube Playlist for video lovers (up to 15 videos is enough).
- Project from React Doc for text lovers (this project will give you the required knowledge of React).

</Resources>

<WordsOfAdvice>

- Start by learning the basics of React like Create-React-App (or vite), folder structure, modular styling(not necessary but a nice thing to explore), and functional components.
- This knowledge along with previous projects will help you to build a nice Navbar and Hero Section component.
- Now start learning about hooks(useState and useEffect hooks are enough for this project). Maintain an ‘expense’ state carrying an array of all the expense objects.
- Now to make the Expense List component, you need to know how to render lists. Also, you need to perform some actions when clicking on the delete/update buttons.
- You can use CSS frameworks if you want (but we don't prefer it, worry not no marks will be deducted).

</WordsOfAdvice>