Assignment 4: useEffect Side Effects
Awareness, Application, Mastery, Influence

Gaurab Wagle

June 16, 2025

Overview of the Assignment

This assignment focuses on the concepts and usage of useEffect hooks to perform side effects such as data fetching, timers, or subscriptions. Major focus will be on using the cleanup functions, understanding of the dependencies and avoiding common pitfalls like infinite loops.

GitHub Repository Link

Get the complete code and project details on my GitHub repository:
`https://github.com/gxaurab/Tangible/tree/main/Assignment/src/components/Part4`
Deployed Site:
`https://tangible-production.up.railway.app/ass4/awareness`

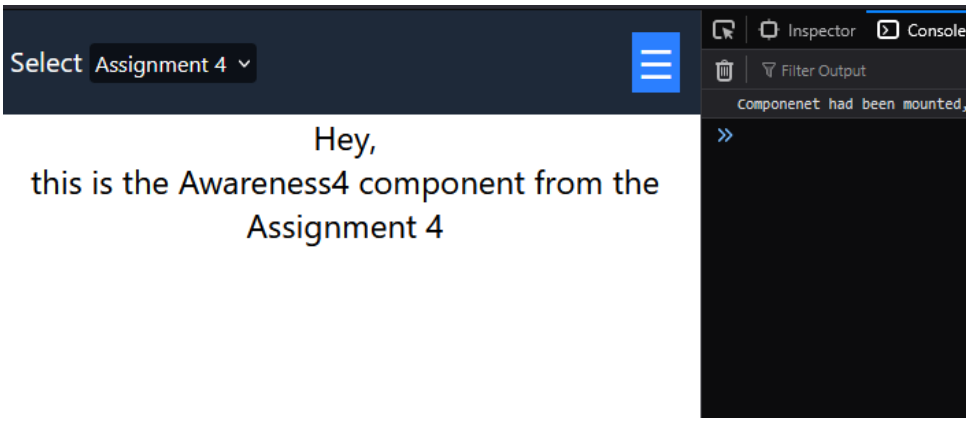
1 Awareness

- **Rubric:** Understand useEffect runs after render and can be used for side effects like logging or simple updates.
- **Proof of Work:** Log a message like "Component mounted" to the console when the component first renders.

1.1 Approach

In this part of the assignment, I learned about the proper usage of useEffect hook. For the awareness part i rendered the message on the screen as well as on console that said Component Mounted during it's first render that is, when it was mounted.

1.2 Outcome



Hey,
this is the Awareness4 component from the Assignment 4

```
Componenet had been mounted,
```

Figure 1: Awareness Implementation

2 Application

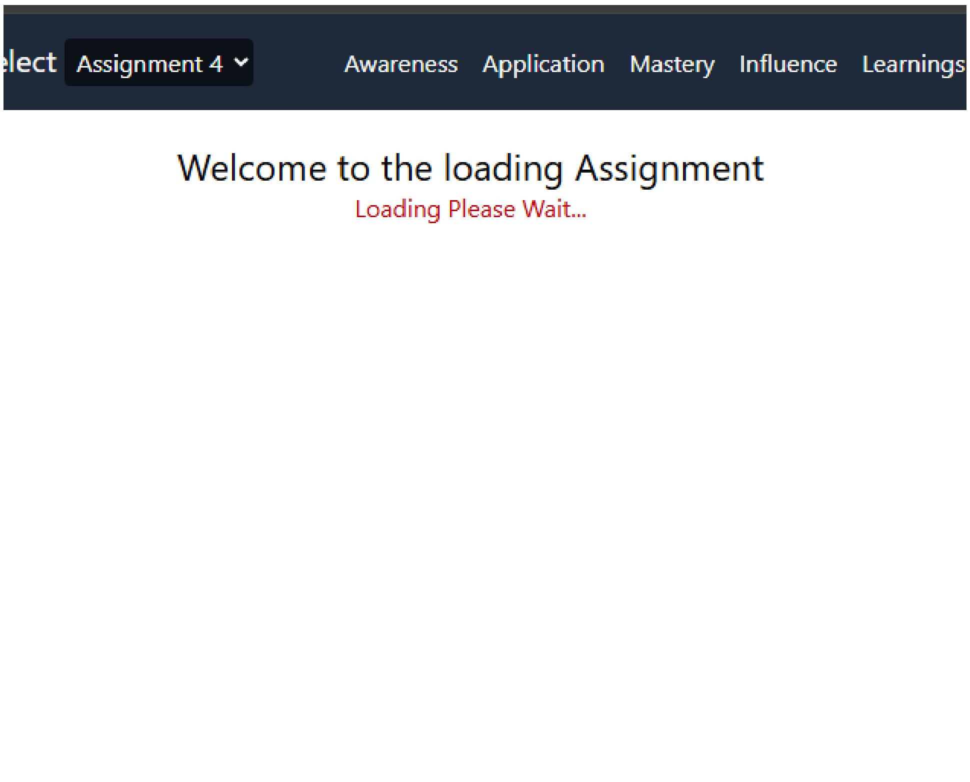
- **Rubric:** Simulate data fetching with setTimeout inside useEffect to show loading state and delayed data.
- **Proof of Work:** Simulate loading a list of items by showing "Loading..." for 2 seconds before displaying the hardcoded list.

2.1 Approach

This part of the assignment required me to simulate data fetching with setTimeout inside useEffect to show loading state and delayed data.

I used the javascript's setTimeout function with delayed time of 2 seconds (2000ms) and cleanup function that cleared the operation when the component has been dismounted.

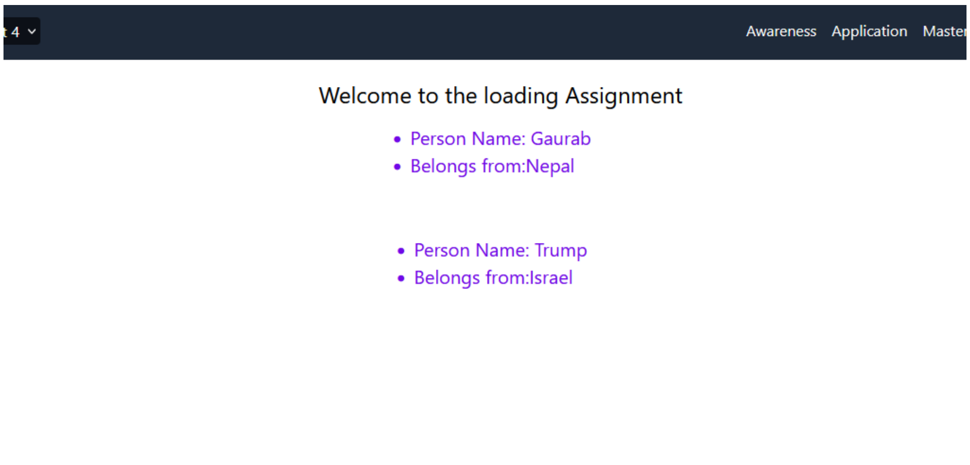
2.2 Outcome



Welcome to the loading Assignment

Loading Please Wait...

Figure 2: Loading State



Welcome to the loading Assignment

- Person Name: Gaurab
- Belongs from:Nepal

- Person Name: Trump
- Belongs from:Israel

Figure 3: Rendering after 2 seconds

3 Mastery

- **Rubric:** Implement interval-based updates with cleanup to avoid memory leaks and stale data.
- **Proof of Work:** Create a digital clock component that updates every second using setInterval and cleans up properly on unmount.

3.1 Approach

I used the javascript's date object to access the clock and utilized different methods to get the desired format. I used the setInterval property which ran the code every 1 second, at each second I would update the time by repeatedly calling the Date object. I used the cleanup function in case of the component unmounting.

3.2 Outcome

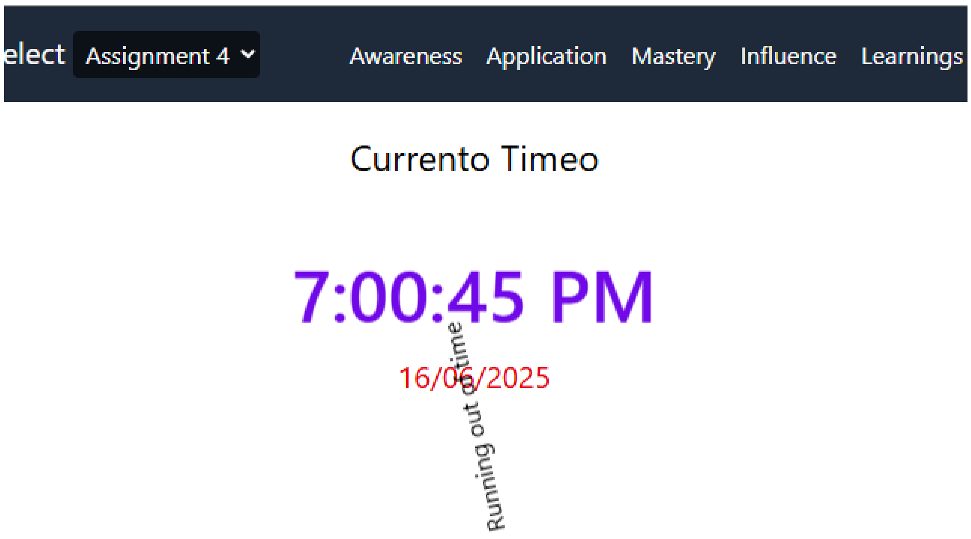


Figure 4: Clock

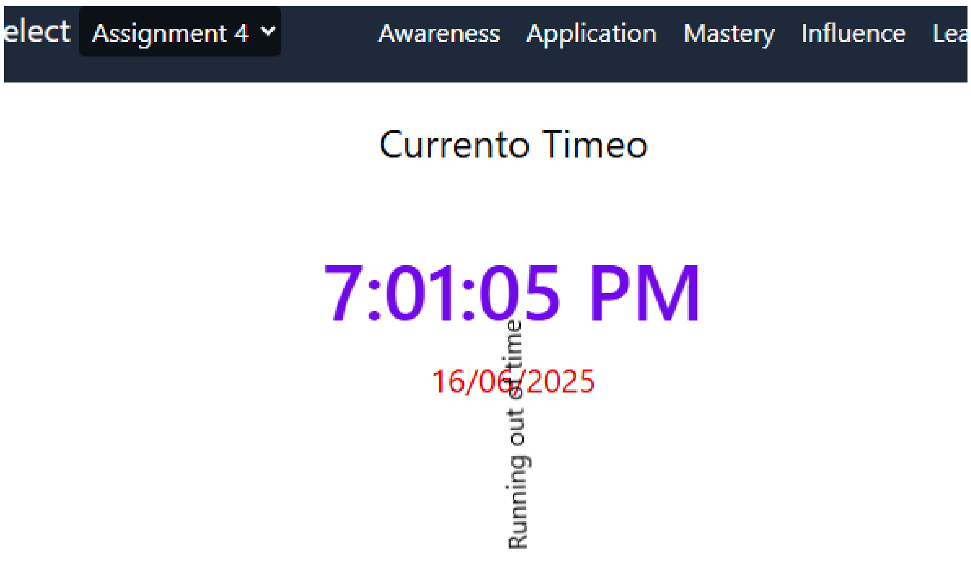


Figure 5: ClockUpdate: (there is a funny spin i animated)

4 Influence

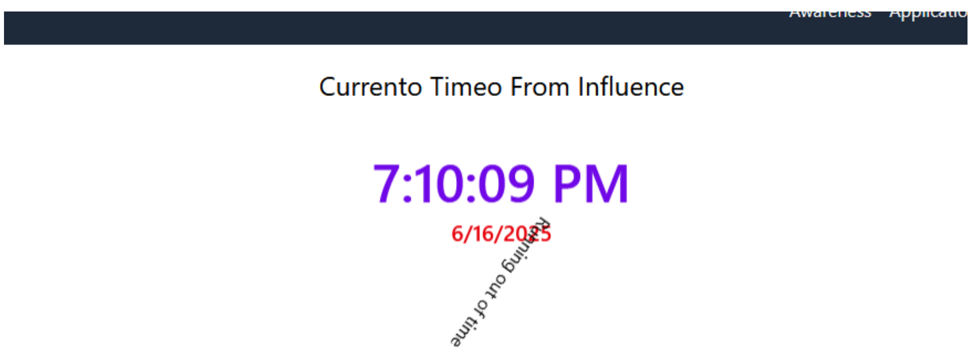
- **Rubric:** Build a reusable custom hook to handle intervals or repeated side effects with proper cleanup.
- **Proof of Work:** Create a useInterval hook that accepts a callback and delay, and use it to build the clock component or auto-refresh UI.

4.1 Approach

For this assignment, I created a custom hook named ”useInterval” as stated which accepted a callback function and delay, number.

In the main component, I created a function named callback (for easiness), then i called the setInterval hook that goes like this: setInterval(callback, 1000). On the other side, my custom hook used this callback function and delay inside it's setInterval method. I also utilized the useRef, which basically stored the most recent function in it's current property. My project didn't actually required useRef property but while reviewing the code by AI, I was encouraged to follow this practice to make my code always store and use the previous property.

4.2 Outcome



Currento Timeo From Influence

7:10:09 PM

6/16/2025

Running out of time

Figure 6: UseInterval Implementation

4.3 Conclusion:

As a conclusion, I am confident in the useEffect hook, custom hook and the cleanup property.