Workshop 12: Group Practice Problem 1

You have a database of galaxy cards. Use async rendering to ~~show~~ the ~~what~~ code is required to successfully load, pend, or reject states associated w/ rendering this data. You do not need to explicitly create components for the pending and rejected states, just the necessary states, requests, and side effects.

Do not write the answer in this booklet. Use scratch paper to write your answer.

Workshop 12: Group Practice Problem 2

explain how rendering Asynchronous Data in React works in 4 parts

Do not write the answer in this booklet. Use scratch paper to write your answer.

Workshop 12: Group Practice Problem 1

What are some good design principles for the asynchronous states?

- Pending
- Rejected
- Fulfilled

Describe what features each state should have.

Do not write the answer in this booklet. Use scratch paper to write your answer.

Workshop 12: Group Practice Problem 1

Explain when to use a side effect and when to use a handler.

Do not write the answer in this booklet. Use scratch paper to write your answer.

Workshop 12: Group Practice Problem 2

Write the function to get more events for the following component. Assume that the /api/events/more endpoint will return an array of 20 events to be concatenated on to the existing events array.

```
import { useState } from 'react'
import axios from axios
import EventCard from "./EventCard"
export default function EventList ({initEvents}){
  const [events, setEvents] = useState (initEvents)
```

```
return (<>
  <ul>
    { events.map(event, index)=> (
      <EventCard event={event} key={index} />
    )}
  </ul>
  <button onClick={handleClick}> Get More Events</button>
</>)
```

Do not write the answer in this booklet. Use scratch paper to write your answer.

Workshop 12: Group Practice Problem 1

```
import {useState} from "react";

export default function RSVPButton({eventId}) {
  const [status, setStatus] = useState("idle");

    async function handleClick() {

    }

    return (
    <button onClick={handleClick}>

      RSVP
    </button>
    );
}
```

Do not write the answer in this booklet. Use scratch paper to write your answer.

Workshop 12: Group Practice Problem 2

Modify the code below so it sends a GET request to /api/events when the component loads.

```
import {useState} from "react"

export default function EventCount() {
  const [events, setEvents] = useState(null);

  function loadEvents() {
    fetch('/api/events').
    .then(res => res.json())
    .then(data => setEvents(data))
  }

  return (
    <>
      {events ? <p>{events.length} events</p> :
        <p>Loading...</p>
      </>
  )
```

Do not write the answer in this booklet. Use scratch paper to write your answer.

Workshop 12: Group Practice Problem 1

modify code below so when button is clicked, it sends POST request to /api/click. while request is sent, button should be disabled to user can't click it multiple times

```
import {useState} from "react"
export default function ClickButton() {

  return (
    <button onClick={handleClick}>
      Click me
    </button>
  )
}
```

Do not write the answer in this booklet. Use scratch paper to write your answer.

Workshop 12: Group Practice Problem 1

```
import usestate
import {useDebounce} from 'use-Debounce'

export default function Search() {
    const [text, setTer] = useState('')
    const [output, setOutput] = useState('')


return (
    <div>
    <input value={text} onChange={(e) => setText(e.target.value)} />
    <p>{output}</p>
  </div>
)
}
```

The text input is debounced by 300ms

After the user stops typing, set:
output = text.toUpperCase()

Do not write the answer in this booklet. Use scratch paper to write your answer.

Workshop 12: Group Practice Problem 1

(a) Retrieve the `movies` collection from the React client and store the data as `moviesData`.

(b) set default states for when the data `isLoading` and when there is a `dataError`.

(c) Finish the `retry button` code:

```
<button onClick={() => >
</button>
```

Do not write the answer in this booklet. Use scratch paper to write your answer.

Workshop 12: Group Practice Problem 2

Rewrite this code to limit how many HTTP requests are sent to the server.

```
...

const [ searchQuery, setSearchQuery ] = useState('')

useEffect( () => {

    HTTPRequest(searchQuery) }, [searchQuery] )

...
```

Do not write the answer in this booklet. Use scratch paper to write your answer.