Task 8

8. Keeping Components Pure
Tasks:

1. Convert an impure component that uses Math.random() within the render phase to a pure one.

Pure.jsx

```
import { useState,useEffect } from 'react'

export default function Pure() {
    const [randomNumber,setRandomNumber] = useState(0)
    useEffect(() => {
        setRandomNumber(Math.random())
    },[]);
  return (
    <>
        <p>Random Number: {randomNumber}</p>
    </>
  )
}
```

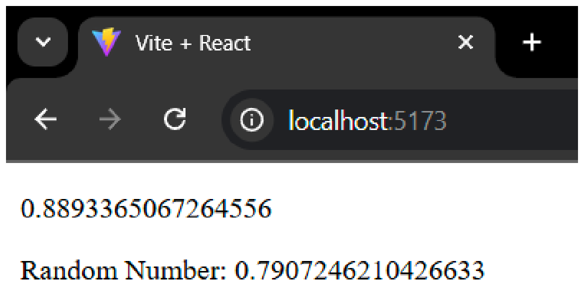
Impure.jsx

```
import React from 'react'

export default function Impure() {
  return (
    <>
        <p>{Math.random()}</p>
    </>
  )
}
```

Main.jsx

```
import Pure from './Pure.jsx'

import Impure from './Impure.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <Impure />
    <Pure />
  </StrictMode>
)
```

Output



```
0.8893365067264556

Random Number: 0.7907246210426633
```

2. Create a pure component Clock that displays the current time and updates every second without causing side-effects during the render phase.

Clock.jsx

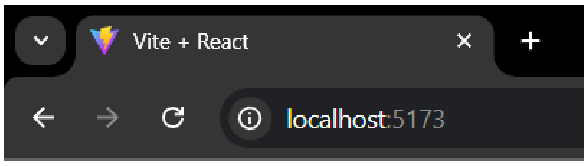
```
import React, { useEffect, useState } from 'react'

export default function Clock() {
    const [time,setTime] = useState(new Date());
    useEffect(()=> {
        const timer = setInterval(()=>setTime(new Date()),1000);
        return () => clearInterval(timer);
    },[]);
  return (
    <>
      <h3>Time: {time.toLocaleTimeString()}</h3>
    </>
  )
}
```

3. Use Strict Mode in an existing application and identify any warnings in the console.

Main.jsx

```
import Clock from './Clock.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <Clock />
  </StrictMode>
)
```

Output



Time: 1:52:08 PM

4. Convert a class-based component with side effects in its lifecycle methods to a pure functional component using hooks.

Class Component

```
import React , {Component} from 'react'

class Counter extends Component {
  constructor(props){
    super(props);
    this.state = {count:0};
  }
  Increment=() => {
    this.setState({count: this.state.count+1})
  }
  render(){
    return(
      <>
        <h2>Count: {this.state.count}</h2>
        <button onClick={this.Increment}>Increase</button>
      </>
    )
  }
}
export default Counter
```

Functional Component

```
import React, { useState } from 'react'

export default function Counter() {
  const [count,setCount] = useState(0);
 return (
  <>
    <h2>Count: {count}</h2>
```

```
        <button onClick={()=>setCount(count+1)}>Increase</button>
    </>
  )
}
```

5. Make a pure ProfilePic component that takes a user ID as a prop and fetches the user's profile picture URL from an array without side-effects during rendering.

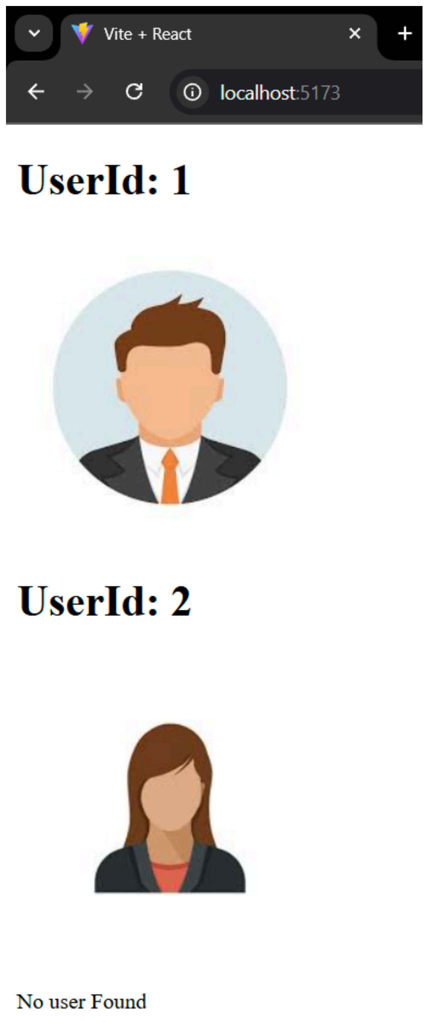
ProfilePic.jsx

```
import React from 'react'

export default function ProfilePic(props) {
    const users = [

{id:1,profile:"https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQyCixyM2urliFC1w0DyNMJpBRMOXFizr3FR8aRIFfcDUGBzEaXcV6mt4gVWRqGAqqu4PI&usqp=CAU"},

{id:2,profile:"https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQQ-Bx4bcOTMKU5bQLVsa5gLWVLWK6blo_r06U9C-ZeJCGkQAwQJ2R1knRcfKrJSO5zpQc&usqp=CAU"}

    ]
    const user = users.find(u => u.id === props.userid)
  return (
    <>
        {user ?(
            <div>
                <h1>UserId: {props.userid}</h1>
                <img src={user.profile}></img>
            </div>
        ):
        (<p>No user Found</p>)}
    </>
  )
}
```

Main.jsx

```
import ProfilePic from './ProfilePic.jsx'
createRoot(document.getElementById('root')).render(
  <StrictMode>
    <ProfilePic userid={1} />
    <ProfilePic userid={2} />
    <ProfilePic userid={3} />
  </StrictMode>
)
```

Output



```
UserId: 1

UserId: 2

No user Found
```