Batch: **D-2** **Roll No.:** **16010123325**

Experiment / assignment / tutorial No._6___

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Implementation of React Fundamentals.

AIM: To Implement the Basic operations of React js

Problem Definition:

-Demonstrate the

- React Fundamentals
- Function Component
- Styling / Bootstrap
- React JSX
- Expressions in JSX
- React Props

*(Students have to perform the task assigned within group/ Individual and demonstrate the same).

Resources used:

Expected OUTCOME of Experiment:

CO 1:.Build full stack applications in JavaScript using the MERN technologies.
CO5: Deploy MERN applications to cloud platforms like Heroku or AWS and collaborate using GitHub version control.

Books/ Journals/ Websites referred:

1. Shelly Powers Learning Node O' Reilly 2 nd Edition, 2016.

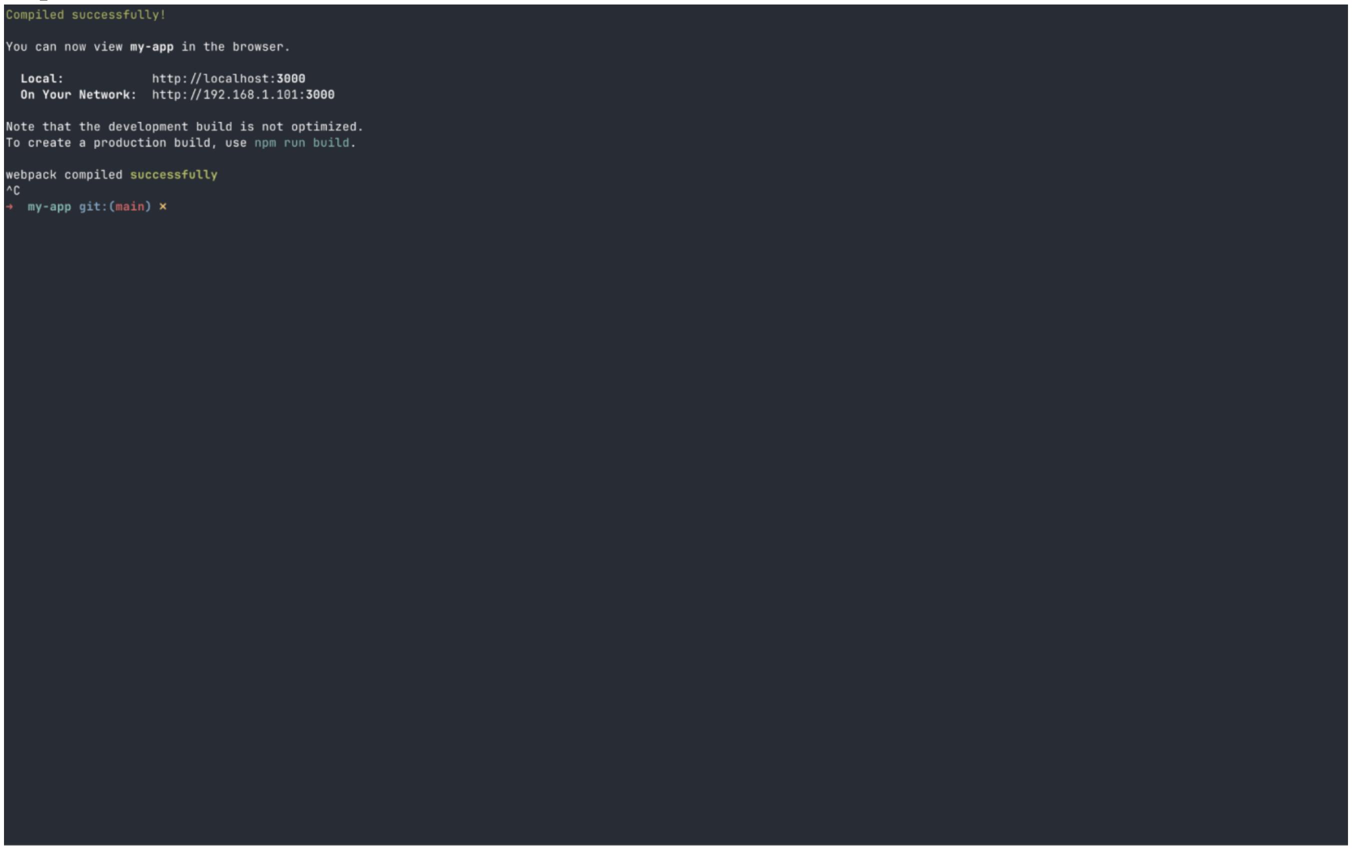
Pre Lab/ Prior Concepts:

React JS:

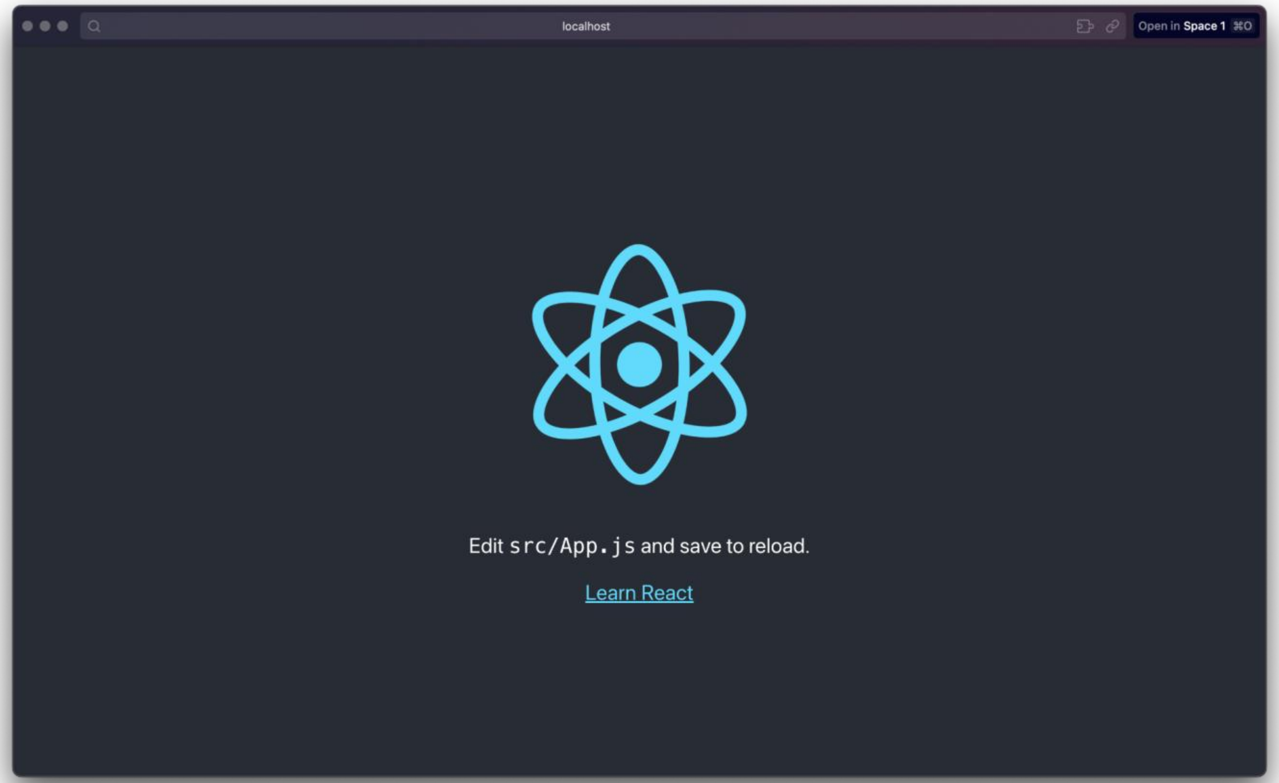
Setup of First Application using React js Steps:

step 1)npx create-react-app my-app
step 2)cd my-app
Setp 3)npm i
Step 4) npm run dev

output



```
Compiled successfully!

You can now view my-app in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.1.101:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
^C
➜  my-app git:(main) ✗
```



Edit `src/App.js` and save to reload.

Learn React

Importance of all Files considering the Folder Structure of First React Application:

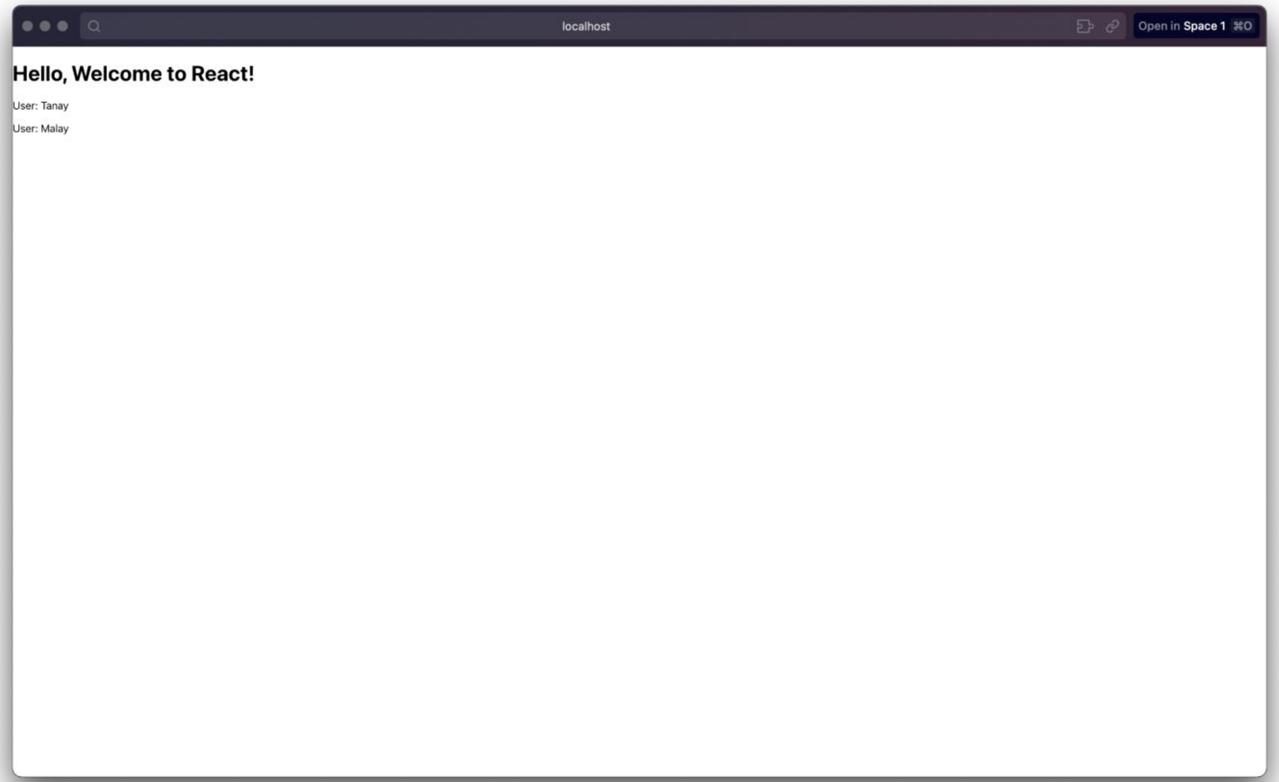
Component in React:

In React, components are reusable building blocks that define parts of a user interface. They can be simple functions or classes that return JSX, allowing developers to create modular, maintainable, and dynamic UIs by combining multiple components together.

Code:

```
import React from "react";

// A simple component
function Welcome() {
 return <h1>Hello, Welcome to React!</h1>;
}

// Another component
function User(props) {
 return <p>User: {props.name}</p>;
}

// Main App component
function App() {
 return (
   <div>
     <Welcome />
     <User name="Tanay" />
     <User name="Malay" />
   </div>
 );
}

export default App;
```

Output:



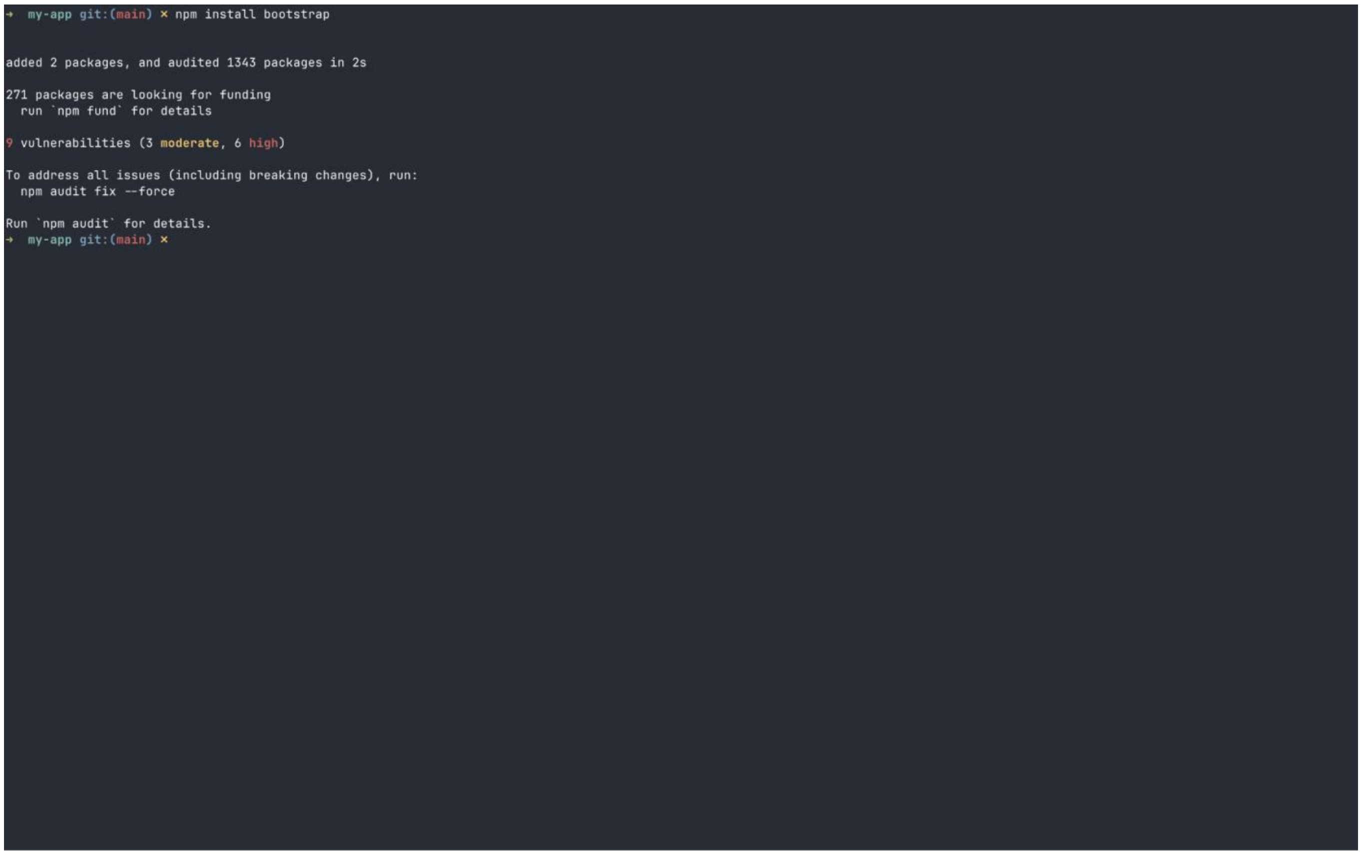
Hello, Welcome to React!

User: Tanay

User: Malay

Styling with Bootstrap:
Styling with Bootstrap in React is done by installing Bootstrap and importing its CSS file. Developers can then use predefined Bootstrap classes like bt`, container, and text-center in JSX elements to quickly build responsive, modern, and consistent user interfaces.

Initializing

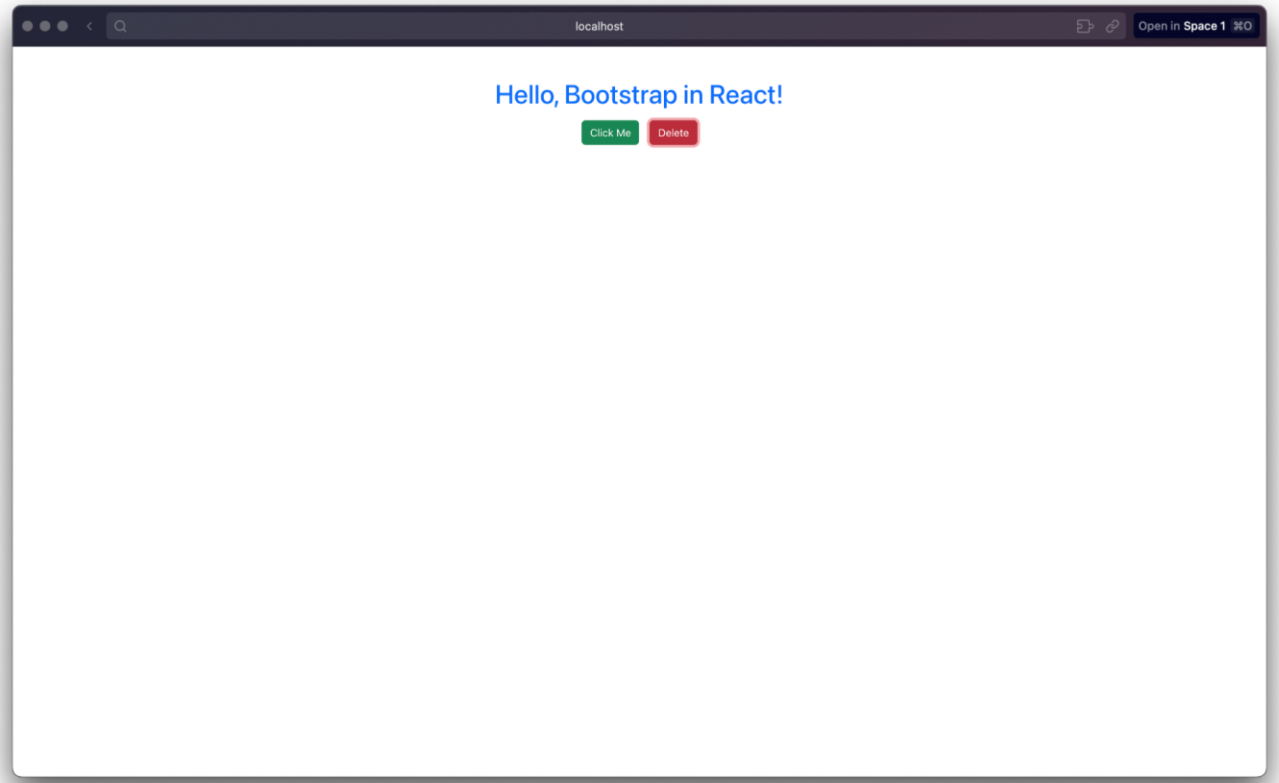


```
→  my-app git:(main) ✗ npm install bootstrap


added 2 packages, and audited 1343 packages in 2s

271 packages are looking for funding
  run `npm fund` for details

9 vulnerabilities (3 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
→  my-app git:(main) ✗
```

Code:

```
import React from "react";
import "bootstrap/dist/css/bootstrap.min.css";

function App() {
 return (
   <div className="container text-center mt-5">
     <h1 className="text-primary">Hello, Bootstrap in React!</h1>
     <button className="btn btn-success m-2">Click Me</button>
     <button className="btn btn-danger m-2">Delete</button>
   </div>
 );
}

export default App;
```

output:



Hello, Bootstrap in React!

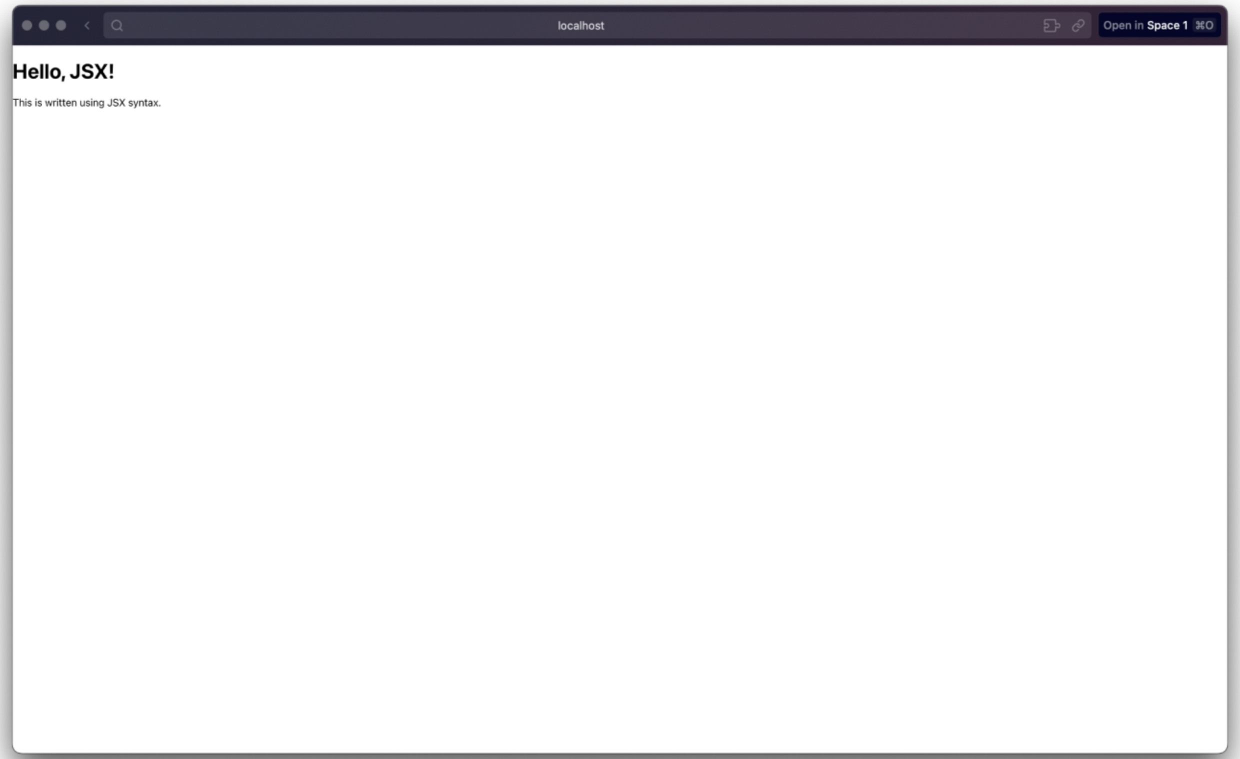
React JSX:

Code:

```
import React from "react";

function App() {
 return (
   <div>
     <h1>Hello, JSX!</h1>
     <p>This is written using JSX syntax.</p>
   </div>
 );
}
```

```
export default App;
```

output:



Hello, JSX!

This is written using JSX syntax.

Expressions in JSX

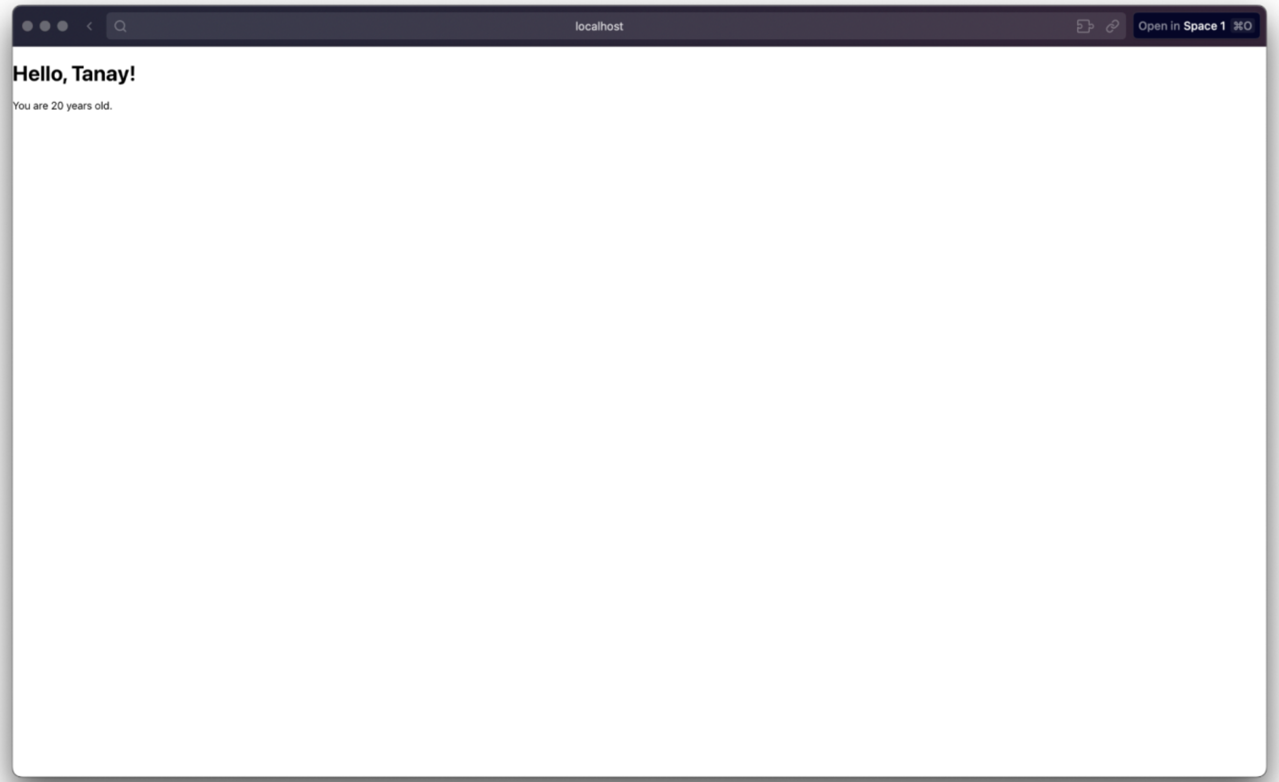
Expressions in JSX allow embedding JavaScript code inside curly braces { }. They can include variables, function calls, calculations, and conditional logic using the ternary operator, making React components dynamic and interactive without separating logic from the user interface.

1) Variables in JSX

Code

```
import React from "react";

function App() {
 const name = "Tanay";
 const age = 20;

 return (
   <div>
     <h1>Hello, {name}!</h1>
     <p>You are {age} years old.</p>
   </div>
 );
}

export default App;
```

Output



Hello, Tanay!

You are 20 years old.

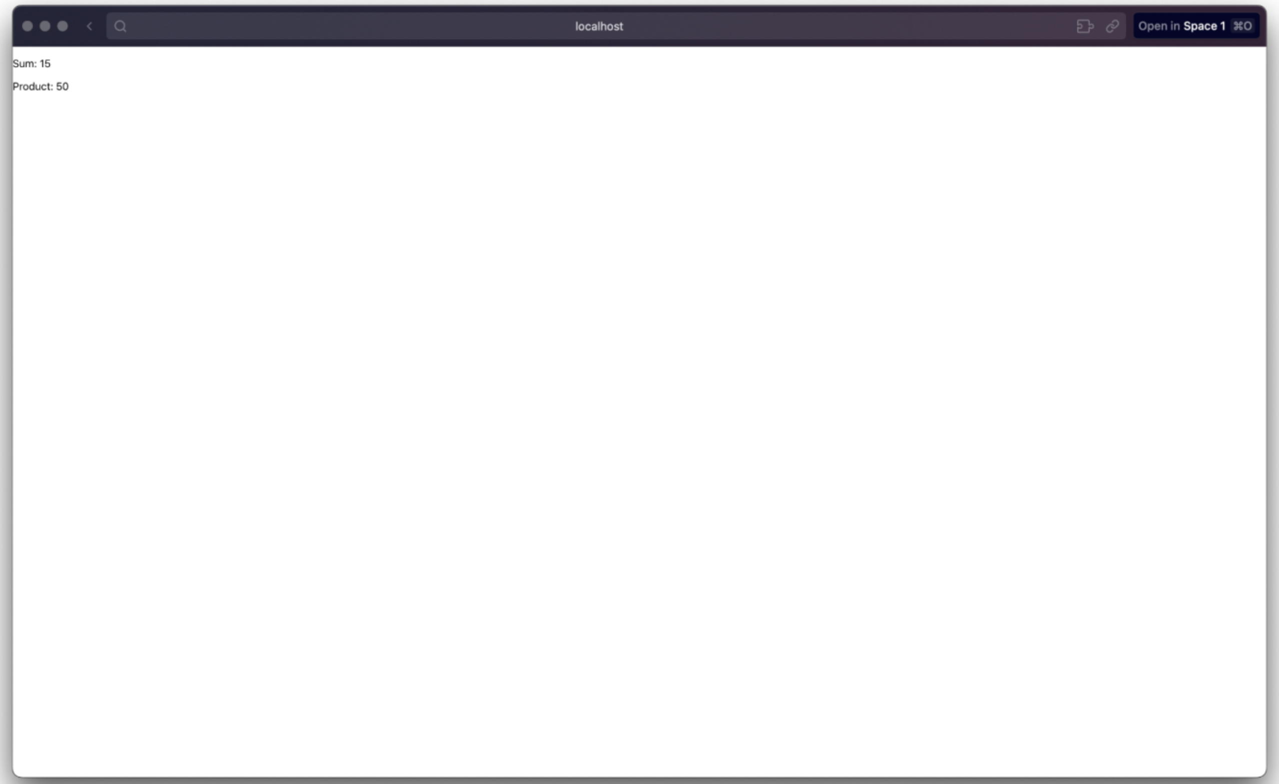
2) Mathematical Expressions

Mathematical expressions in JSX are written inside curly braces { }. You can perform operations like addition, subtraction, multiplication, or division directly within JSX, making it easy to display calculated results dynamically as part of the user interface.

Code:

```
import React from "react";

function App() {
 const a = 10;
 const b = 5;

 return (
   <div>
     <p>Sum: {a + b}</p>
     <p>Product: {a * b}</p>
   </div>
 );
}

export default App;
```

Output:



```
Sum: 15
Product: 50
```

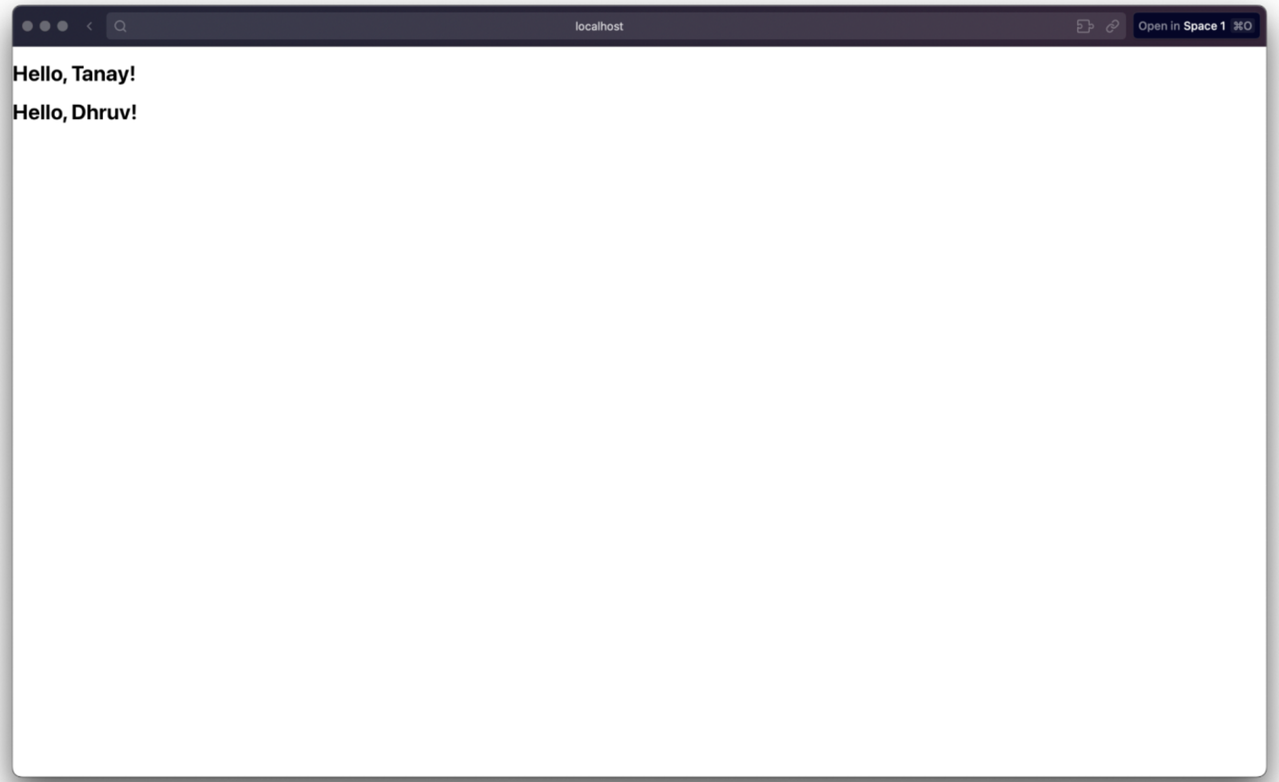
3) Function Calls in JSX

Function calls in JSX are written inside curly braces { } to dynamically display results. You can call functions that return values, process data, or format content, allowing components to be more interactive, reusable, and responsive to different inputs.

Code:

```
import React from "react";

function App() {
 const greet = (name) => `Hello, ${name}!`;

 return (
   <div>
     <h1>{greet("Tanay")}</h1>
     <h1>{greet("Dhruv")}</h1>
   </div>
 );
}

export default App;
```

Output:



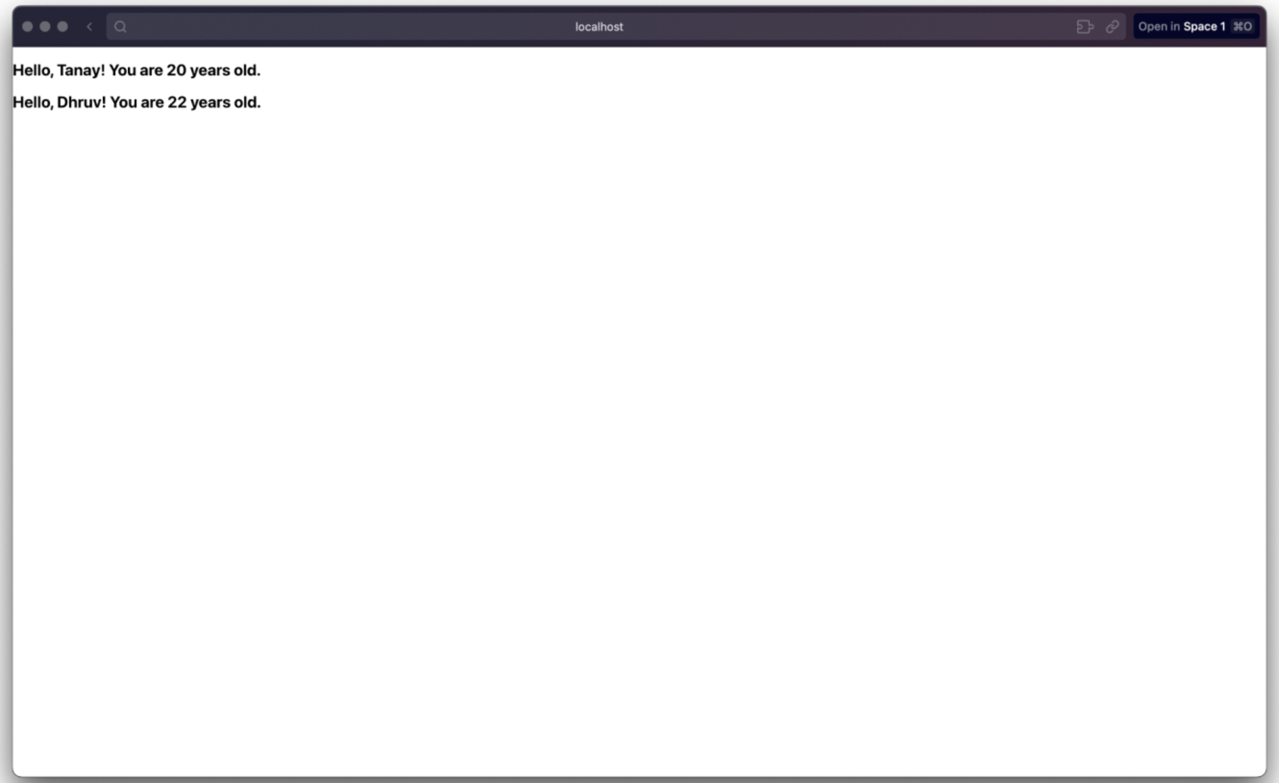
```
Hello, Tanay!
Hello, Dhruv!
```

React props:
In React, props (properties) are used to pass data from a parent component to a child component. They make components reusable and dynamic. Props are read-only, meaning child components can use them but cannot modify their values.

code:

```
import React from "react";

// Child Component
function User(props) {
 return <h2>Hello, {props.name}! You are {props.age} years old.</h2>;
}

// Parent Component
function App() {
 return (
   <div>
     <User name="Tanay" age={20} />
     <User name="Dhruv" age={22} />
   </div>
 );
}

export default App;
```

Output:



```
Hello, Tanay! You are 20 years old.
Hello, Dhruv! You are 22 years old.
```

Conclusion

This experiment provided a comprehensive and practical introduction to the foundational principles of React, successfully demonstrating how to build dynamic user interfaces from the ground up. By creating functional components, we learned to break down the UI into manageable and reusable pieces, which is key to developing scalable applications. The implementation of JSX was a major takeaway, highlighting its power to seamlessly blend HTML structure with JavaScript logic. We saw firsthand how embedding expressions, variables, and function calls directly within the markup makes the UI more declarative and easier to reason about.Furthermore, the concept of props proved essential for component communication, enabling the one-way data flow from parent to child that is central to React's architecture. This makes components highly configurable and reusable. Finally, integrating Bootstrap for styling illustrated how quickly a modern, responsive design can be applied, allowing us to focus more on application logic.

Postlab questions:

1. **Explain the Concept of SPA.**

A Single Page Application (SPA) is a web app that loads a single HTML page and dynamically updates content without reloading. It uses JavaScript and routing to provide a fast, seamless, app-like user experience within the browser.

2. **Describe Props**

Props in React are inputs passed from a parent component to a child component, similar to function parameters. They help make components dynamic and reusable by allowing data transfer. Props are immutable, meaning child components cannot change their values.

3. **Comparison with Functional and Class Component with example.**

Functional components are simple JavaScript functions that return JSX and use hooks like `useState` for state management. Class components are ES6 classes that extend `React.Component`, using `this.state` and lifecycle methods. Functional components are preferred for their simplicity and readability. Example: a functional component uses `function Hello() { return <h1>Hello</h1>; }`, while a class component uses `class Hello extends React.Component { render() { return <h1>Hello</h1>; } }`.