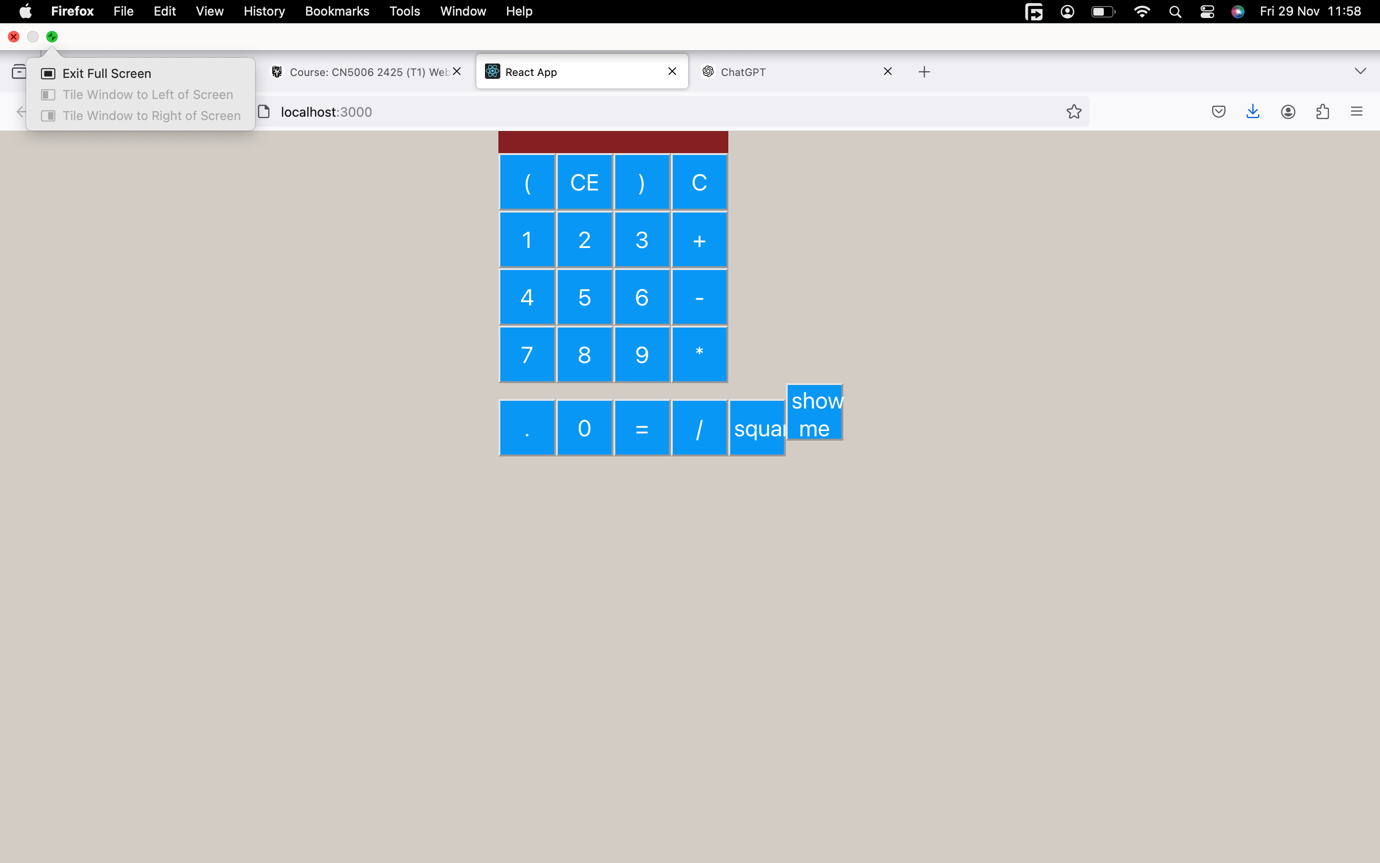
Assignment 2

output



**Task 1: Create a React Application and Integrate Calculator.js and Button.js**

**Steps to Develop the Application:**

1. **Initialize the React Application**:
   * Run npx create-react-app calculator-app to set up a React project.
   * Navigate to the project folder: cd calculator-app.
2. **Add the Provided Files**:
   * Place Calculator.js and Button.js inside the src directory of your React project.
3. **Edit and Import Components**:
   * Import Calculator and Button into App.js and use them to build the calculator interface.
4. **Functional Integration**:
   * Calculator.js serves as the logic manager for handling operations.
   * Button.js is used to define buttons that trigger actions.

**Breakdown of Provided Code:**

**Calculator.js**:

import React, { useState } from 'react';

const Calculator = () => {

const [display, setDisplay] = useState('');

const handleButtonClick = (value) => {

if (value === '=') {

try {

setDisplay(eval(display)); // Avoid using eval in real-world apps; use a parser for safety.

} catch (error) {

setDisplay('Error');

}

} else if (value === 'C') {

setDisplay('');

} else {

setDisplay(display + value);

}

};

return (

<div>

<input type="text" value={display} readOnly />

<div>

{['1', '2', '3', '+', '4', '5', '6', '-', '7', '8', '9', '\*', '0', 'C', '=', '/'].map((btn) => (

<Button key={btn} text={btn} onClick={() => handleButtonClick(btn)} />

))}

</div>

</div>

);

};

export default Calculator;

import React from 'react';

const Button = ({ text, onClick }) => {

return (

<button onClick={onClick}>

{text}

</button>

);

};

export default Button;

import React from 'react';

import Calculator from './Calculator';

const App = () => {

return (

<div>

<h1>React Calculator</h1>

<Calculator />

</div>

);

};

export default App;

### ****Task 2: Add a Button to Display a Picture****

#### Modify Button.js:

import React from 'react';

const Button = ({ text, onClick }) => {

return (

<button onClick={onClick}>

{text}

</button>

);

};

export default Button;

#### Add Button Logic in Calculator.js:

const Calculator = () => {

const [display, setDisplay] = useState('');

const [showImage, setShowImage] = useState(false);

const handleButtonClick = (value) => {

if (value === '=') {

try {

setDisplay(eval(display));

} catch (error) {

setDisplay('Error');

}

} else if (value === 'C') {

setDisplay('');

} else if (value === 'show me') {

setShowImage(!showImage);

} else {

setDisplay(display + value);

}

};

return (

<div>

<input type="text" value={display} readOnly />

<div>

{['1', '2', '3', '+', '4', '5', '6', '-', '7', '8', '9', '\*', '0', 'C', '=', '/', 'show me'].map((btn) => (

<Button key={btn} text={btn} onClick={() => handleButtonClick(btn)} />

))}

</div>

{showImage && <img src="your-image-url.jpg" alt="Your Image" />}

</div>

);

};

### ****Task 3: Add a Button to Display the Square of a Number****

#### Update Calculator.js:

const handleButtonClick = (value) => {

if (value === '=') {

try {

setDisplay(eval(display));

} catch (error) {

setDisplay('Error');

}

} else if (value === 'C') {

setDisplay('');

} else if (value === 'square') {

try {

setDisplay(String(Math.pow(Number(display), 2)));

} catch (error) {

setDisplay('Error');

}

} else {

setDisplay(display + value);

}

};

return (

<div>

<input type="text" value={display} readOnly />

<div>

{['1', '2', '3', '+', '4', '5', '6', '-', '7', '8', '9', '\*', '0', 'C', '=', '/', 'show me', 'square'].map

**Task 1: React Calculator Application**

**Implementation Overview:**

1. **Project Setup**:
   * A React app was created using create-react-app, and the provided files Calculator.js and Button.js were added to the src folder.
   * These files form the backbone of the calculator: Calculator.js handles the logic, and Button.js provides reusable button components.
2. **Integration**:
   * The Calculator.js file manages the state and logic for the calculator's display and operations using React's useState hook.
   * Buttons representing digits and operators were dynamically generated by mapping over an array of button labels.
   * Each button triggers the handleButtonClick function, which updates the display or calculates results (e.g., using eval to evaluate expressions).

**Output**:

1. A functional calculator interface where users can input numbers, perform calculations, and clear the display.

**Task 2: Add a Button to Display a Picture**

**Implementation Overview:**

1. **Additional State for the Image**:
   * A new state variable (showImage) was added using useState to track whether the image should be displayed or not.
2. **New Button**:
   * A button labeled show me was added to the calculator interface using the Button.js component.
3. **Image Toggle Logic**:
   * When the show me button is clicked, the handleButtonClick function toggles the showImage state.
   * If showImage is true, the image is rendered below the calculator. Otherwise, it remains hidden.
4. **Outcome**:
   * Clicking the show me button displays or hides an image, demonstrating dynamic rendering based on state.

**Task 3: Add a Button to Calculate the Square of a Number**

**Implementation Overview:**

1. **Additional Button for "Square"**:
   * A button labeled square was added to the calculator using the existing Button.js component.
2. **Square Calculation Logic**:
   * The handleButtonClick function was updated to handle the square operation.
   * When the square button is clicked, the number currently displayed in the calculator is squared using Math.pow.
3. **Enhanced Functionality**:
   * This extends the calculator’s capabilities by enabling users to compute the square of a number with a single button press.

**Functional Component Integration**

All tasks were implemented as a functional component (Calculator.js) using React hooks like useState to manage states for:

* Display updates for the calculator.
* Image visibility for the show me button.
* Square calculations for the square button.

This modular and reusable approach ensures clarity and efficiency in managing each feature while adhering to React's best practices.