Calculator Application

IIHT

Time To Complete: 2 hr

CONTENTS

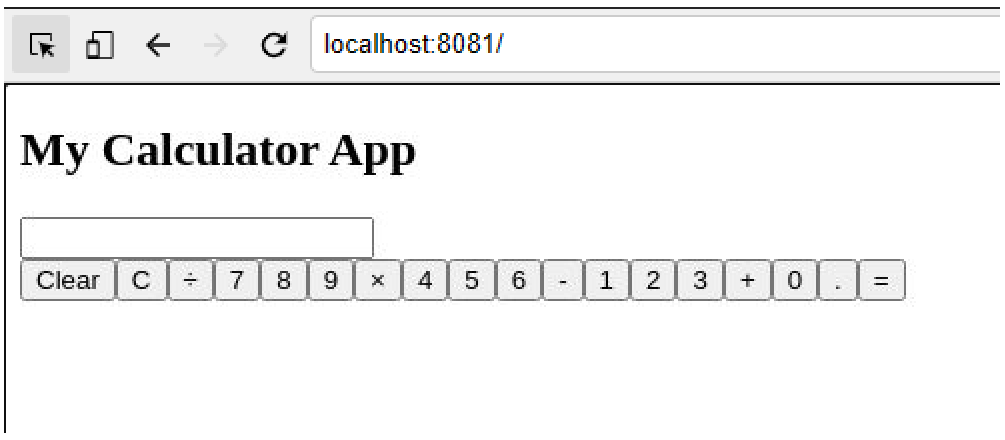
1 Problem Statement

"Calculator Application" is a Single Page Application (SPA) that mimics a regular calculator and has all basic buttons along with regular operations. It also shows the result of any operation.

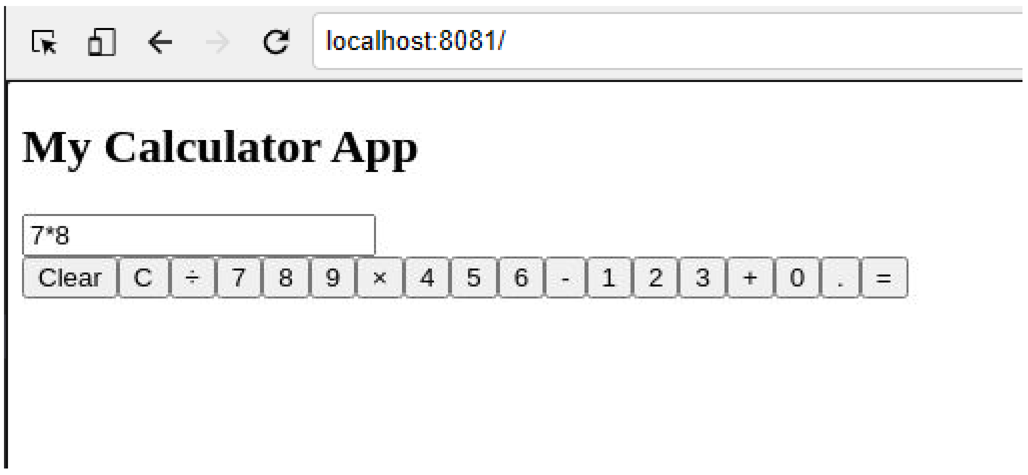
2 Proposed Calculator Application Wireframe

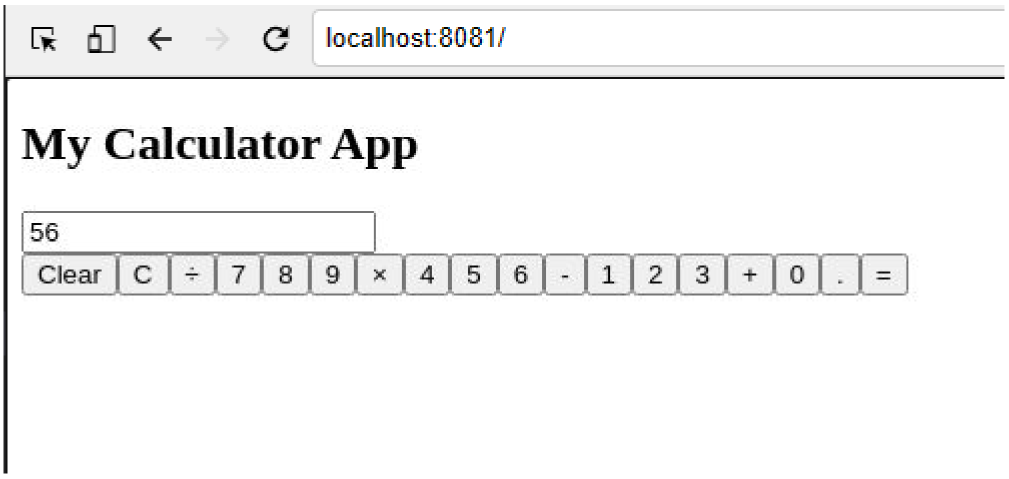
UI needs improvisation and modification as per given use case and to make test cases passed.

2.1 Welcome Page



2.2 Screen Shots





localhost:8081/

My Calculator App

56

3 Business-Requirement:

As an application developer, develop the Calculator Application (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Welcome Page	As a user I should be able to visit the welcome page as default page. Acceptance Criteria: **AppComponent:** 1. Should render a heading **"My Calculator App"** inside an `<h2>` tag. 2. Should render the `Calculator` component inside the `<main>` section. **Calculator Component:** **State Variables:** 1. result: ● **Type:** string ● **Purpose:** Stores the current expression or calculated result displayed in the input field. ● **Initial Value:** `''` (empty string) **Calculator Functionality:** 1. Input Handling: ● Clicking number/operator buttons appends their value to the `result` state. ● Triggered by `handleClick(e)` function. 2. Clear Button: ● Clears the entire `result`. ● Triggered by `clear()` function. 3. Backspace Button: ● Removes the last character from the `result`. ● Triggered by `backspace()` function. 4. Equal Button: ● Evaluates the current expression and updates the `result`. ● Triggered by `calculate()` function using `eval`. ● If evaluation fails, show `"Error"`. **HTML Structure & Behavior:** 1. Input Field: ● Add an input element of **type** `"text"`.

- Bind its **value** to the `result` state so it displays the current expression or result.

2. Keypad Buttons: Include buttons for:
   - **Numbers**: 0 through 9
   - **Operators**: Addition (`+`), Subtraction (`-`), Multiplication (`*`), Division (`/`), and Decimal (`.`)
   - **Actions**:
     - ➔ **Clear**: Resets the input.
     - ➔ **Backspace** (labeled `"C"`): Removes the last character.
     - ➔ **Equals** (labeled `"="`): Calculates the result.
3. Button Layout:
   - Place all buttons inside a div container with the class `"keypad"`.
   - Each button should have an `onClick` event:
     - ➔ Number/operator buttons should update the result using `handleClick`.
     - ➔ Action buttons should call their respective functions (`clear`, `backspace`, or `calculate`).

Submission Logic:

1. **handleClick** ⟶ Concatenates clicked button's `name` to `result`.
2. **clear** ⟶ Resets `result` to `''`.
3. **backspace** ⟶ Removes the last character from the `result`.
4. **calculate** ⟶ Tries evaluating `result`:
   - If valid ⟶ updates with output.
   - If invalid ⟶ updates with `"Error"`.
5. Users should be able to see a non editable text field which shows all operands, operators and the result of operation also.
6. Users should be able to see the list of all buttons (showed in screenshot).
7. It must clear the fields once we click on the Clear button.
8. Result must be shown only on clicking = button.

**** Kindly refer to the screenshots for any clarifications. ****

4 Validations

- All buttons must be present in it.
- All buttons must be clickable.
- Once any numeric is clicked, it must be represented in a non-editable text field.

5 Constraints

- You should be able to press the “TAB” key and “SHIFT + TAB” to navigate from top field to bottom field and vice-versa.

6 Mandatory Assessment Guidelines

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.**
3. **This editor Auto Saves the code.**
4. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**
5. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
6. **This is a web-based application, to run the application on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**
   Note: The application will not run in the local browser

7. You can follow series of command to setup React environment once you are in your project-name folder:
    a. npm install -> Will install all dependencies -> takes 10 to 15 min
    b. npm run start -> To compile and deploy the project in browser. You can press the <Ctrl> key while clicking on localhost:8080/8081 to open the project in the browser -> takes 2 to 3 min.
    c. npm run jest -> to run all test cases and see the summary. It takes 5 to 6 min to run.
    d. npm run test -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min.
8. You may also run “npm run jest” while developing the solution to refactor the code to pass the test-cases.
9. Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on “Submit Assessment” after you are done with code.
10. You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.