Calculator

- Project Name: EL's CALC

- Version: 1.0

- Date: [26/06/2024]

- Author: [OBEN BRANDON]

1. **Planning**

- Purpose: To develop a user-friendly calculator application, EL's CALC, for basic arithmetic operations.

- Scope: The calculator will support addition, subtraction, multiplication, and division.

- Objectives: Create a reliable, efficient, and visually appealing calculator app.

- Stakeholders: [OBEN BRANDON], End Users

---

**2. Analysis**

- Functional Requirements:

  - Perform basic arithmetic operations: addition, subtraction, multiplication, and division.

  - Display inputs and results on the screen.

  - Clear input and results.

- Non-Functional Requirements:

  - The app should be responsive and work on various screen sizes.

  - The UI should be intuitive and visually appealing.

- Constraints:

  - Must be developed using HTML, CSS, and JavaScript.

  - Must be compatible with modern web browsers.

- Assumptions:

  - Users have basic knowledge of how to use a calculator.

  - The app will be used on devices with internet access.

**3. Design**

- System Architecture:

  - Overview: The calculator app will be built as a web application using HTML, CSS, and JavaScript.

  - Components:

    - User Interface: HTML and CSS for layout and styling.

  - Logic: JavaScript for handling calculations and interactions.

- User Interface Design:

  - Mockups:

    - Display Screen: A rectangular display area to show inputs and results.

    - Buttons: Numeric buttons (0-9), operational buttons (+, -, \*, /), and a clear button.

  - User Flows:

    1. User enters numbers and operations via buttons.

    2. The display shows the entered values.

    3. Upon pressing the "=" button, the result is calculated and displayed.

---

**4. Implementation**

- Tools and Technologies:

  - HTML for structure.

  - CSS for styling.

  - JavaScript for functionality.

- Setup Instructions:

  1. Install a code editor (e.g., Visual Studio Code).

  2. Set up a project directory.

  3. Create HTML, CSS, and JavaScript files.

- Code Implementation:

  - HTML: Structure the calculator layout with a display area and buttons.

  - CSS: Style the buttons, display area, and overall layout.

  - JavaScript: Implement the logic for handling button clicks and performing calculations.

---

**5. Testing**

- Test Plan:

  - Unit Tests: Verify each arithmetic operation.

  - Integration Tests: Ensure the UI correctly displays the input and results.

  - User Acceptance Tests: Confirm the app meets user expectations.

- Test Cases:

  - Addition of two numbers.

  - Subtraction of two numbers.

  - Multiplication of two numbers.

  - Division of two numbers.

  - Clearing the input.

- Testing Tools: Browser developer tools, JavaScript testing frameworks (e.g., Jest).

---

**6. Deployment**

- Deployment Instructions:

  1. Host the files on a web server.

  2. Ensure the app is accessible via a web URL.

- Hosting Environment: GitHub Pages, Netlify, or any web hosting service.

- Deployment Steps:

  1. Push the code to a GitHub repository.

  2. Use GitHub Pages or another service to host the site.

---

**7. Maintenance**

- Maintenance Plan:

  - Regular updates for compatibility and performance improvements.

  - Bug fixes based on user feedback.

- Future Enhancements:

  - Advanced mathematical functions (e.g., square root, exponentiation).

  - Memory functions (e.g., M+, M-, MR).

  - Monitor the app for issues.

  - Provide support through user feedback channels.