CHAPTER 1

INTRODUCTION

The Age Calculator project is a simple yet effective tool designed to help users calculate their age based on their birthdate. With a clean and user-friendly interface, this web application allows you to input your birthdate, and with the click of a button, it provides you with an accurate breakdown of your age in years, months, and days.

The project's aesthetic appeal is enhanced by its sleek design, featuring a modern black and white color scheme with a hint of orange. The use of the elegant "Poppins" font from Google Fonts ensures a pleasant and readable user experience. The responsive layout ensures that the calculator looks great on both desktop and mobile devices.

Behind the scenes, the JavaScript code powers the age calculation, taking into account leap years and handling various scenarios, such as future birthdates. In addition to its core functionality, this Age Calculator project offers a seamless and error-handling experience. It ensures that users are informed and alerted if they input a birthdate that suggests they haven't been born yet. With this thoughtful approach, the calculator not only provides accurate age results but also ensures a user-friendly and reliable experience, making it a valuable tool for anyone wanting to determine their age or that of others effortlessly.

CHAPTER 2

IMPLEMENTATION

HTML Code

```
<html lang="en">
<head>
    <title>Age Calculator</title>
    <!--Google Font-->
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;500&display=swap"
rel="stylesheet">
    <!--Stylesheet-->
    <link rel="stylesheet" href="deeksha.css">
</head>
<body>
    <div class="container">
        <div class="inputs-wrapper">
            <input type="date" id="date-input">
            <button onclick="ageCalculate()">Calculate</button>
        </div>
        <div class="outputs-wrapper">
            <div>
                <span id="years">
                    -
                </span>
                <p>
                    Years
                </p>
            </div>
            <div>
                <span id="months">
                    -
                </span>
                <p>
                   Months
                </p>
            </div>
            <div>
                <span id="days">
                    -
                </span>
                <p>
                    Days
```

```
                </p>
            </div>
        </div>
    </div>
    <script src="deeksha.js"></script>
</body>
</html>
```

Title and Styling: The web page begins with a title, "Age Calculator," which is displayed in the browser's title bar or tab. It also imports the "Poppins" font from Google Fonts to enhance the typography of the page. The external stylesheet "deeksha.css" is linked to ensure consistent and visually appealing styling throughout the application.

Input and Calculation: Within the main content, there's a container div that holds the entire user interface. Inside this container, the "inputs-wrapper" div contains an input field with the type "date," allowing users to select their birthdate conveniently. Next to the input field, there's a "Calculate" button with an "onclick" attribute, which triggers the "ageCalculate()" function when clicked.

Output Display: Below the input section, the "outputs-wrapper" div is responsible for displaying the calculated age. It consists of three separate divs, each containing a span element with a unique ID ("years," "months," and "days") where the calculated age results will be dynamically updated. These divs are accompanied by descriptive paragraphs to specify the units of the displayed age (Years, Months, and Days).

JavaScript Integration: To enable the age calculation functionality, the web page references an external JavaScript file named "deeksha.js" using the "script" tag. This JavaScript code handles the calculation of the user's age and updates the displayed results in real-time.

CSS Code

```
*,
*:before,
*:after{
    padding: 0;
    margin: 0;
    box-sizing: border-box;
}
body{
    background-color: orange;
}
.container{
    width: 40%;
    min-width: 450px;
    position: absolute;
    transform: translate(-50%,-50%);
    left: 50%;
    top: 50%;
    padding: 50px 30px;
}
.container *{
    font-family: "Poppins",sans-serif;
    border: none;
    outline: none;
}
.inputs-wrapper{
    background-color: #080808;
    padding: 30px 25px;
    border-radius: 8px;
    box-shadow: 0 15px 20px rgba(0,0,0,0.3);
    margin-bottom: 50px;
}
input,
button{
    height: 50px;
    background-color: #ffffff;
    color: #080808;
    font-weight: 500;
    border-radius: 5px;
}
input{
    width: 60%;
    padding: 0 20px;
    font-size: 14px;
}
button{
    width: 30%;
```

```
    float: right;
}
.outputs-wrapper{
    width: 100%;
    display: flex;
    justify-content: space-between;
}
.outputs-wrapper div{
    height: 100px;
    width: 100px;
    background-color: #080808;
    border-radius: 5px;
    color: #ffffff;
    display: grid;
    place-items: center;
    padding: 20px 0;
    box-shadow: 0 15px 20px rgba(0,0,0,0.3);
}
span{
    font-size: 30px;
    font-weight: 500;
}
p{
    font-size: 14px;
    color: #707070;
    font-weight: 400;
}
```

Universal Reset and Background Color: The CSS code begins with a universal reset, targeting all elements, including their pseudo-elements and pseudo-classes. This reset standardizes the padding and margin values while ensuring that the box-sizing property is set to "border-box." Such a reset is crucial for achieving consistent and predictable layouts across different browsers. Additionally, the code sets the background color of the web page's body to a vibrant orange hue. This choice of background color adds a visually pleasing backdrop to the entire web page, creating a visually striking first impression for users.

Container Styling and Element Customization: The CSS further proceeds to style the main container of the Age Calculator application. The container's width is set to a flexible 40%, with a minimum width of 450 pixels, ensuring that the calculator maintains a responsive and user-friendly layout. Positioned absolutely at 50% from both the left and top, the container is centered on the screen, providing a visually balanced and centered appearance. Inside the container, the "Poppins" font from Google Fonts is applied universally to all child elements, enhancing readability and aesthetics. Additionally, elements within the container, including input fields and buttons, are customized with a consistent design. They feature a white background, contrasting text color, a noticeable border-radius, and a subtle box-shadow, creating a modern and cohesive visual identity.

Output Styling and Visual Consistency: The CSS code continues by styling the output section of the Age Calculator. The "outputs-wrapper" div employs flexbox to ensure that the age results are evenly spaced within the container. Each individual result div is given a fixed size, a dark background color, rounded corners, and a visually pleasing box-shadow effect. The text within these result divs is styled for clarity and legibility, with distinct font sizes, weights, and colors. The use of the "Poppins" font throughout the container contributes to a harmonious and visually consistent design. Overall, this CSS code meticulously shapes the visual elements of the Age Calculator, providing a polished and user-friendly interface for age calculations.

Javascript Code

```
const months = [31,28,31,30,31,30,31,31,30,31,30,31];

function ageCalculate(){
    let today = new Date();
    let inputDate = new Date(document.getElementById("date-input").value);
    let birthMonth,birthDate,birthYear;
    let birthDetails = {
        date:inputDate.getDate(),
        month:inputDate.getMonth()+1,
        year:inputDate.getFullYear()
    }
    let currentYear = today.getFullYear();
    let currentMonth = today.getMonth()+1;
    let currentDate = today.getDate();

    leapChecker(currentYear);

    if(
        birthDetails.year > currentYear ||
        ( birthDetails.month > currentMonth && birthDetails.year == currentYear)
||
        (birthDetails.date > currentDate && birthDetails.month == currentMonth &&
birthDetails.year == currentYear)
    ){
        alert("Not Born Yet");
        displayResult("-","-","-");
        return;
    }

    birthYear = currentYear - birthDetails.year;

    if(currentMonth >= birthDetails.month){
        birthMonth = currentMonth - birthDetails.month;
    }
    else{
        birthYear--;
        birthMonth = 12 + currentMonth - birthDetails.month;
    }

    if(currentDate >= birthDetails.date){
        birthDate = currentDate - birthDetails.date;
    }
    else{
        birthMonth--;
        let days = months[currentMonth - 2];
```

```
        birthDate = days + currentDate - birthDetails.date;
        if(birthMonth < 0){
            birthMonth = 11;
            birthYear--;
        }
    }
    displayResult(birthDate,birthMonth,birthYear);
}

function displayResult(bDate,bMonth,bYear){
    document.getElementById("years").textContent = bYear;
    document.getElementById("months").textContent = bMonth;
    document.getElementById("days").textContent = bDate;
}

function leapChecker(year){
    if(year % 4 == 0 || (year % 100 == 0 && year % 400 == 0)){
        months[1] = 29;
    }
    else{
        months[1] = 28;
    }
}
```

Months Array and Leap Year Checker: The code starts with the declaration of a constant array named **months**, containing the number of days in each respective month. This array serves as the foundation for calculating age in months and days. Additionally, the **leapChecker** function is defined to adjust the days in February depending on whether the current year is a leap year or not. This ensures accurate calculations for individuals born on or around February 29th.

ageCalculate Function: The **ageCalculate** function is responsible for calculating the age based on the inputted birthdate. It begins by retrieving the current date using the **Date** object. It then extracts the user-inputted birthdate from the HTML input field. The birth details (day, month, and year) are stored in the **birthDetails** object for easy access.

The current year, month, and date are also obtained and stored in variables. The **leapChecker** function is called to ensure that February has the correct number of days.

The function then proceeds to check if the birthdate is valid (i.e., not in the future). If the birthdate is invalid, an alert is shown, and the display is set to show dashes ("-") to indicate no valid age has been calculated.

Next, the function calculates the years, months, and days difference between the birthdate and the current date. This includes handling scenarios where the current month or day is before the birth month or day respectively.

Finally, the **displayResult** function is called to update the HTML elements with the calculated age.

displayResult Function: The **displayResult** function is responsible for updating the HTML elements with the calculated age values. It takes the calculated birthdate, birth month, and birth year as arguments and sets the corresponding elements' text content accordingly.

Leap Year Checker Function: The **leapChecker** function determines if a given year is a leap year. It adjusts the number of days in February (index 1 of the **months** array) accordingly, ensuring the accurate calculation of age for individuals born in a leap year.

CHAPTER 3

OUTPUT