**Full Stack Development**

**Lab 2-2-2026**

Name:Aditya K U

Roll:AV.SC.U4AIE23001

AIE-A

**Experiment 1: Digital Clock**

**Objective: Construct a real-time digital clock application.**

* Use HTML for layout and structure.
* Apply CSS for visual styling.
* Implement JavaScript to update and display the current time dynamically.

**Code:**

**Html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Digital Clock</title>

    <link rel="stylesheet" href="clock.css">

</head>

<body>

    <div class="clock">

        <span id="time">00:00:00</span>

    </div>

    <script src="clock.js"></script>

</body>

</html>

**Css**

body {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    background-color: #121212;

    font-family: Arial, sans-serif;

}

.clock {

    background: #000;

    padding: 20px 40px;

    border-radius: 10px;

    box-shadow: #00ffcc;

    border: 2px solid;

    border-color: aquamarine;

}

#time {

    color: #00ffcc;

    font-size: 48px;

    letter-spacing: 2px;

}

**Js**

function updateClock() {

    const now = new Date();

    let hours = now.getHours();

    let minutes = now.getMinutes();

    let seconds = now.getSeconds();

    hours = hours < 10 ? "0" + hours : hours;

    minutes = minutes < 10 ? "0" + minutes : minutes;

    seconds = seconds < 10 ? "0" + seconds : seconds;

    const currentTime = `${hours}:${minutes}:${seconds}`;

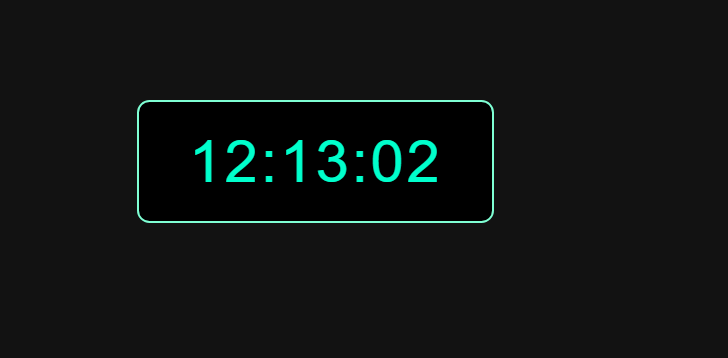
    document.getElementById("time").textContent = currentTime;

}

updateClock();

setInterval(updateClock, 1000);

**Output:**

****

**Experiment 2: Simple Calculator**

**Objective: Develop a basic calculator.**

* Design the user interface with HTML.
* Enhance visual appearance with CSS.
* Execute arithmetic operations using JavaScript logic.

**Code:**

**Html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Simple Calculator</title>

    <link rel="stylesheet" href="calc.css">

</head>

<body>

    <div class="calculator">

        <input type="text" id="display" disabled>

        <div class="buttons">

            <button onclick="clearDisplay()">C</button>

            <button onclick="appendValue('/')">/</button>

            <button onclick="appendValue('\*')">\*</button>

            <button onclick="appendValue('-')">-</button>

            <button onclick="appendValue('7')">7</button>

            <button onclick="appendValue('8')">8</button>

            <button onclick="appendValue('9')">9</button>

            <button onclick="appendValue('+')">+</button>

            <button onclick="appendValue('4')">4</button>

            <button onclick="appendValue('5')">5</button>

            <button onclick="appendValue('6')">6</button>

            <button onclick="calculateResult()">=</button>

            <button onclick="appendValue('1')">1</button>

            <button onclick="appendValue('2')">2</button>

            <button onclick="appendValue('3')">3</button>

            <button onclick="appendValue('0')" class="zero">0</button>

        </div>

    </div>

    <script src="calc.js"></script>

</body>

</html>

**Css**

\* {

    box-sizing: border-box;

}

body {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    background: linear-gradient(135deg, #1e1e1e, #2c2c2c);

    font-family: Arial, sans-serif;

}

.calculator {

    background: #111;

    padding: 25px;

    border-radius: 15px;

    box-shadow: 0 15px 30px rgba(0, 0, 0, 0.7);

}

#display {

    width: 100%;

    height: 55px;

    font-size: 26px;

    margin-bottom: 15px;

    text-align: right;

    padding: 10px;

    border: none;

    border-radius: 8px;

    background-color: #222;

    color: #00ffcc;

}

.buttons {

    display: grid;

    grid-template-columns: repeat(4, 65px);

    gap: 12px;

}

button {

    height: 55px;

    font-size: 18px;

    border: none;

    border-radius: 8px;

    cursor: pointer;

    background-color: #333;

    color: white;

    transition: transform 0.1s ease, background-color 0.2s;

}

button:hover {

    background-color: #444;

    transform: scale(1.05);

}

button:active {

    transform: scale(0.95);

}

.zero {

    grid-column: span 2;

}

button:nth-child(1),

button:nth-child(2),

button:nth-child(3),

button:nth-child(4),

button:nth-child(8),

button:nth-child(12) {

    background-color: #ff9500;

    color: white;

}

button:nth-child(1):hover,

button:nth-child(2):hover,

button:nth-child(3):hover,

button:nth-child(4):hover,

button:nth-child(8):hover,

button:nth-child(12):hover {

    background-color: #e08900;

}

**Js**

const display = document.getElementById("display");

function appendValue(value) {

    display.value += value;

}

function clearDisplay() {

    display.value = "";

}

function calculateResult() {

    try {

        display.value = eval(display.value);

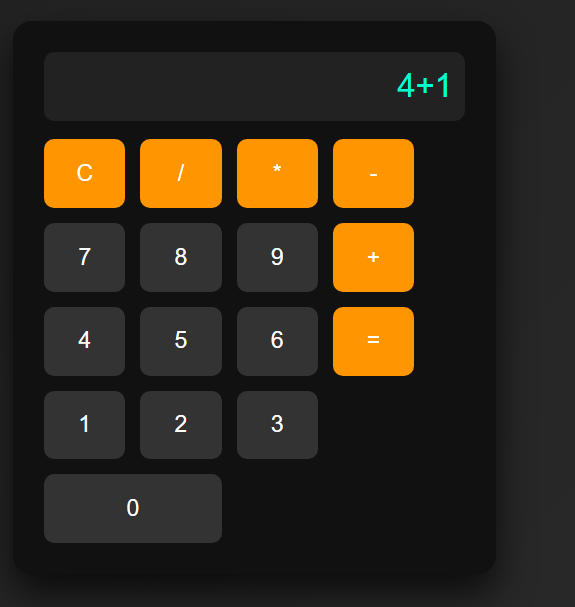
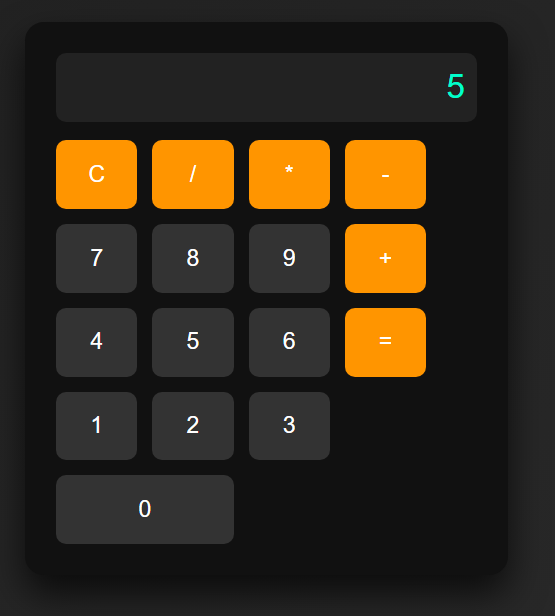
    } catch {

        display.value = "Error";

    }

}

**Output:**

** **

**Experiment 3: Login Page**

**Objective: Create a user login form.**

* Structure the form using HTML elements.
* Style the page with CSS for improved usability.
* Add input validation features via JavaScript.

**Code:**

**Html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Login Page</title>

    <link rel="stylesheet" href="login.css">

</head>

<body>

    <div class="login-container">

        <h2>Login</h2>

        <form onsubmit="return validateForm()">

            <input type="text" id="username" placeholder="Username">

            <input type="password" id="password" placeholder="Password">

            <button type="submit">Login</button>

            <p id="error"></p>

        </form>

    </div>

    <script src="login.js"></script>

</body>

</html>

**Css**

body {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    background-color: #f2f2f2;

    font-family: Arial, sans-serif;

}

.login-container {

    background: #ffffff;

    padding: 30px;

    border-radius: 10px;

    width: 300px;

    box-shadow: 0 0 10px rgba(0,0,0,0.1);

    text-align: center;

}

input {

    width: 100%;

    padding: 10px;

    margin: 10px 0;

    font-size: 16px;

}

button {

    width: 100%;

    padding: 10px;

    background-color: #007bff;

    border: none;

    color: white;

    font-size: 16px;

    cursor: pointer;

}

button:hover {

    background-color: #0056b3;

}

#error {

    color: red;

    margin-top: 10px;

}

**Js**

function validateForm() {

    const username = document.getElementById("username").value;

    const password = document.getElementById("password").value;

    const error = document.getElementById("error");

    if (username === "" || password === "") {

        error.textContent = "Both fields are required!";

        return false;

    }

    if (password.length < 6) {

        error.textContent = "Password must be at least 6 characters!";

        return false;

    }

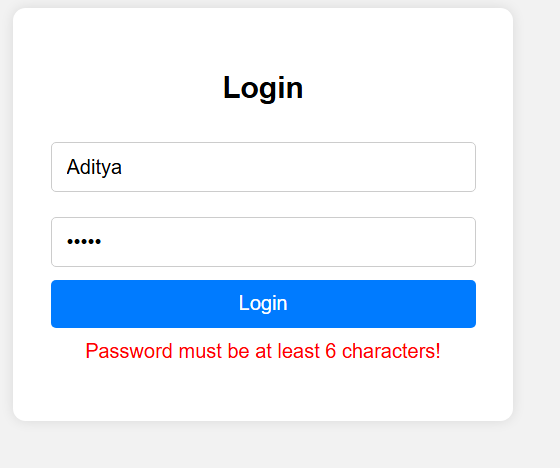
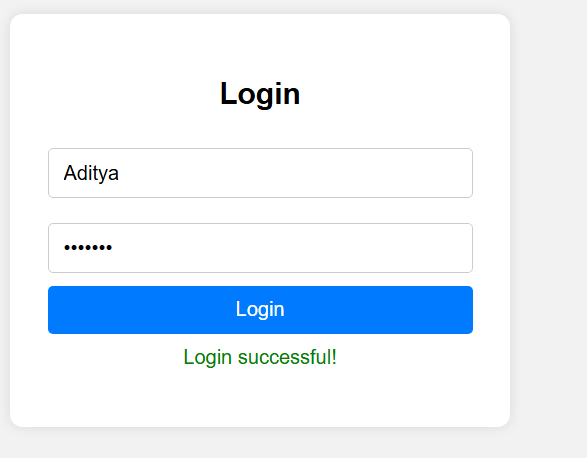
    error.textContent = "Login successful!";

    error.style.color = "green";

    return false;

}

**Output:**

** **

**Experiment 4: To-Do List**

**Objective: Build a task management tool.**

* Establish the task list layout with HTML.
* Style the interface using CSS.
* Manage task addition and removal through JavaScript functions.

**Code:**

**Html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>To-Do List</title>

    <link rel="stylesheet" href="task.css">

</head>

<body>

    <div class="todo-container">

        <h2>My To-Do List</h2>

        <input type="text" id="taskInput" placeholder="Enter a task">

        <button onclick="addTask()">Add Task</button>

        <ul id="taskList"></ul>

    </div>

    <script src="task.js"></script>

</body>

</html>

**Css**

\* {

    box-sizing: border-box;

}

body {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    background-color: #eef2f3;

    font-family: Arial, sans-serif;

}

.todo-container {

    background: #ffffff;

    padding: 25px;

    border-radius: 10px;

    width: 420px;

    box-shadow: 0 0 10px rgba(0,0,0,0.1);

}

input {

    width: 100%;

    padding: 10px;

    margin: 12px 0;

}

button {

    width: 100%;

    padding: 10px;

    background-color: #28a745;

    border: none;

    color: white;

    font-size: 16px;

    cursor: pointer;

}

button:hover {

    background-color: #218838;

}

li {

    list-style: none;

    padding: 10px;

    margin-top: 10px;

    background: #f4f4f4;

    display: flex;

    justify-content: space-between;

    align-items: center;

    border: 2px solid black;

    border-radius: 10px;

}

li button {

    background-color: #dc3545;

    padding: 5px 10px;

    font-size: 14px;

    margin: 10px;

    border-radius: 10px;

}

**Js**

function addTask() {

    const taskInput = document.getElementById("taskInput");

    const taskList = document.getElementById("taskList");

    if (taskInput.value === "") {

        alert("Please enter a task!");

        return;

    }

    const li = document.createElement("li");

    li.textContent = taskInput.value;

    const deleteBtn = document.createElement("button");

    deleteBtn.textContent = "Delete";

    deleteBtn.onclick = function () {

        taskList.removeChild(li);

    };

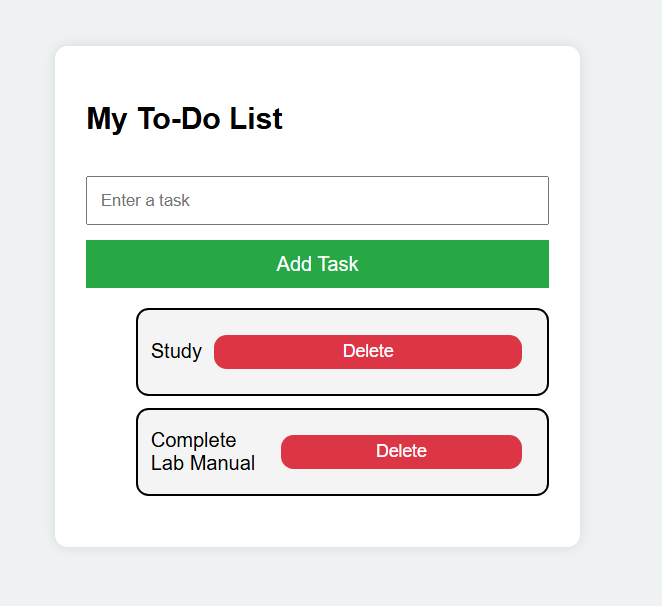
    li.appendChild(deleteBtn);

    taskList.appendChild(li);

    taskInput.value = "";

}

**Output:**

****

**Experiment 5: Hamburger Menu**

**Objective: Implement a responsive navigation menu.**

* Construct menu structure with HTML.
* Use CSS for menu styling and responsiveness.
* Use JavaScript to toggle menu visibility, especially for smaller screen sizes.

**Code:**

**Html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Hamburger Menu</title>

    <link rel="stylesheet" href="burger.css">

</head>

<body>

    <nav class="navbar">

        <div class="logo"> Burger Menu</div>

        <ul class="nav-links" id="navLinks">

            <li><a href="#">Home</a></li>

            <li><a href="#">About</a></li>

            <li><a href="#">Services</a></li>

            <li><a href="#">Contact</a></li>

        </ul>

        <div class="hamburger" onclick="toggleMenu()">

            ☰

        </div>

    </nav>

    <script src="burger.js"></script>

</body>

</html>

**Css**

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    font-family: Arial, sans-serif;

}

.navbar {

    display: flex;

    justify-content: space-between;

    align-items: center;

    background-color: #333;

    padding: 15px 20px;

    color: white;

}

.logo {

    font-size: 20px;

}

.nav-links {

    list-style: none;

    display: flex;

}

.nav-links li {

    margin-left: 20px;

}

.nav-links a {

    color: white;

    text-decoration: none;

}

.hamburger {

    display: none;

    font-size: 26px;

    cursor: pointer;

}

@media (max-width: 768px) {

    .nav-links {

        display: none;

        flex-direction: column;

        background-color: #333;

        position: absolute;

        top: 60px;

        right: 0;

        width: 200px;

    }

    .nav-links li {

        margin: 15px 0;

        text-align: center;

    }

    .hamburger {

        display: block;

    }

}

**Js**

function toggleMenu() {

    const navLinks = document.getElementById("navLinks");

    if (navLinks.style.display === "flex") {

        navLinks.style.display = "none";

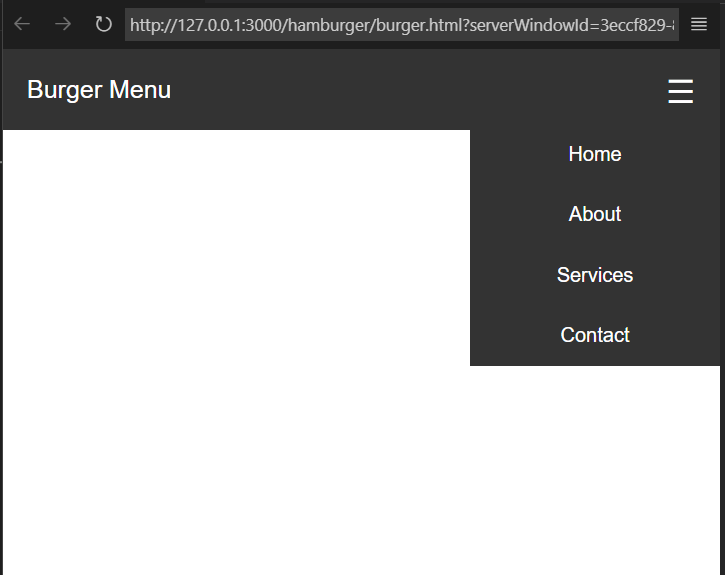
    } else {

        navLinks.style.display = "flex";

    }

}

**Output:**

****

**Experiment 6: Weather App**

**Description: Design a simple weather application that consumes live data from an API.**

**Objectives:**

* Build the user interface with HTML.
* Apply CSS for a clean, appealing look.
* Utilize JavaScript to fetch weather data from an API and display it on the interface.

**Code:**

**Html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Weather App</title>

    <link rel="stylesheet" href="weather.css">

</head>

<body>

    <div class="weather-container">

        <h2>Weather App</h2>

        <input type="text" id="city" placeholder="Enter city name">

        <button onclick="getWeather()">Get Weather</button>

        <div class="weather-result" id="result"></div>

    </div>

    <script src="weather.js"></script>

</body>

</html>

**Css**

\* {

    box-sizing: border-box;

}

body {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    background-color: #e3f2fd;

    font-family: Arial, sans-serif;

}

.weather-container {

    background: #ffffff;

    padding: 25px;

    border-radius: 10px;

    width: 100%;

    max-width: 400px;   /\* controls card size \*/

    text-align: center;

    box-shadow: 0 0 10px rgba(0,0,0,0.1);

}

input {

    width: 100%;

    padding: 12px;

    margin: 12px 0;

    font-size: 16px;

}

button {

    padding: 12px;

    width: 100%;

    background-color: #2196f3;

    color: white;

    border: none;

    font-size: 16px;

    cursor: pointer;

    border-radius: 4px;

}

button:hover {

    background-color: #1976d2;

}

.weather-result {

    margin-top: 15px;

    font-size: 16px;

}

**Js**

function getWeather() {

    const city = document.getElementById("city").value;

    const result = document.getElementById("result");

    const apiKey = "dfad33a86177f71234c101fe0e49a111";

    const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=${apiKey}`;

    fetch(url)

        .then(response => response.json())

        .then(data => {

            if (data.cod === "404") {

                result.innerHTML = "City not found!";

            } else {

                result.innerHTML = `

                    <p><strong>City:</strong> ${data.name}</p>

                    <p><strong>Temperature:</strong> ${data.main.temp} °C</p>

                    <p><strong>Weather:</strong> ${data.weather[0].description}</p>

                `;

            }

        })

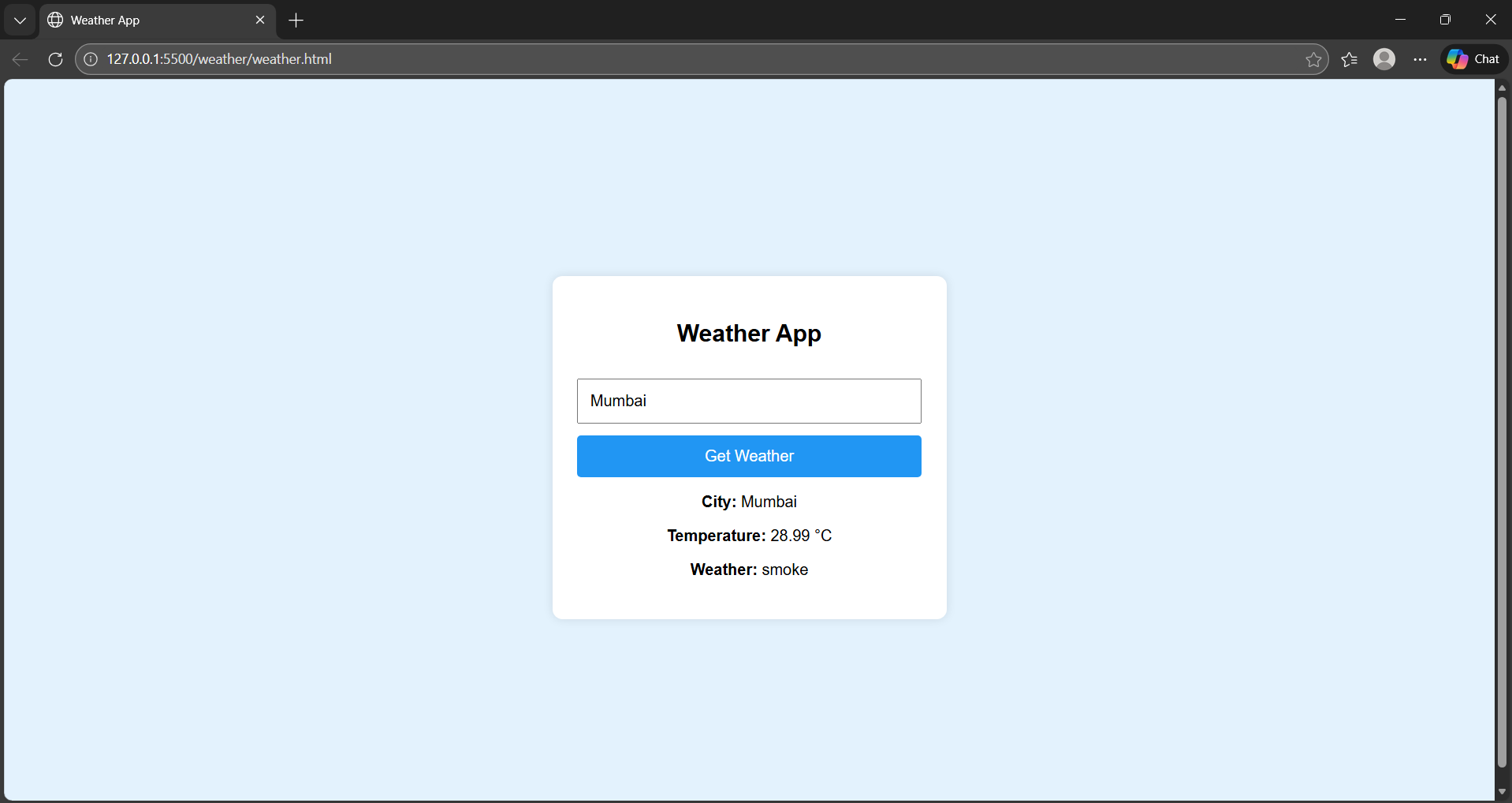
        .catch(() => {

            result.innerHTML = "Error fetching data!";

        });

}

**Output:**



**Experiment 7: Quiz Application**

**Description: Create a quiz application featuring multiple-choice questions and a scoring system.**

**Objectives:**

* Construct the quiz layout with HTML.
* Style the application using CSS for better user experience.
* Implement JavaScript for managing question flow and scoring logic.

**Code:**

**Html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Quiz Application</title>

    <link rel="stylesheet" href="quiz.css">

</head>

<body>

<div class="quiz-container">

    <h2 id="question">Question text</h2>

    <div class="options">

        <button class="option" onclick="checkAnswer(0)"></button>

        <button class="option" onclick="checkAnswer(1)"></button>

        <button class="option" onclick="checkAnswer(2)"></button>

        <button class="option" onclick="checkAnswer(3)"></button>

    </div>

    <button id="nextBtn" onclick="nextQuestion()">Next</button>

    <h3 id="score"></h3>

</div>

<script src="quiz.js"></script>

</body>

</html>

**Css**

\* {

    box-sizing: border-box;

}

body {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    background: linear-gradient(135deg, #667eea, #764ba2);

    font-family: "Segoe UI", Arial, sans-serif;

}

.quiz-container {

    background: #ffffff;

    padding: 30px;

    border-radius: 16px;

    width: 420px;

    text-align: center;

    box-shadow: 0 20px 40px rgba(0,0,0,0.25);

    animation: fadeIn 0.6s ease;

}

#question {

    font-size: 20px;

    font-weight: 600;

    margin-bottom: 20px;

    color: #333;

}

.options {

    margin: 20px 0;

}

.option {

    display: block;

    width: 100%;

    padding: 12px;

    margin: 10px 0;

    font-size: 16px;

    cursor: pointer;

    border: 2px solid #ddd;

    border-radius: 10px;

    background: #8ee3f5;

    transition: all 0.25s ease;

}

.option:hover {

    background-color: #eef3ff;

    border-color: #667eea;

    transform: translateY(-2px);

}

.option:disabled {

    cursor: not-allowed;

    opacity: 0.7;

}

#nextBtn {

    padding: 12px;

    width: 100%;

    background: linear-gradient(135deg, #228f34, #2e8017);

    color: white;

    border: none;

    font-size: 16px;

    cursor: pointer;

    border-radius: 10px;

    transition: transform 0.2s ease, box-shadow 0.2s ease;

}

#nextBtn:hover {

    transform: translateY(-2px);

    box-shadow: 0 8px 15px rgba(0,0,0,0.2);

}

#score {

    margin-top: 20px;

    font-size: 18px;

    font-weight: bold;

    color: #28a745;

}

@keyframes fadeIn {

    from {

        opacity: 0;

        transform: scale(0.95);

    }

    to {

        opacity: 1;

        transform: scale(1);

    }

}

**Js**

const quizData = [

    {

        question: "What does HTML stand for?",

        options: [

            "Hyper Text Markup Language",

            "High Text Machine Language",

            "Hyperlinks Text Mark Language",

            "Home Tool Markup Language"

        ],

        answer: 0

    },

    {

        question: "Which language is used for styling web pages?",

        options: ["HTML", "JQuery", "CSS", "XML"],

        answer: 2

    },

    {

        question: "Which is not a JavaScript framework?",

        options: ["React", "Angular", "Django", "Vue"],

        answer: 2

    },

    {

        question: "Which symbol is used for comments in JavaScript?",

        options: ["<!-- -->", "//", "\*\*", "##"],

        answer: 1

    },

    {

        question: "Which HTML tag is used for JavaScript?",

        options: ["<script>", "<js>", "<javascript>", "<code>"],

        answer: 0

    },

    {

        question: "Which company developed JavaScript?",

        options: ["Microsoft", "Netscape", "Google", "Apple"],

        answer: 1

    },

    {

        question: "Which CSS property controls text size?",

        options: ["font-style", "text-size", "font-size", "text-style"],

        answer: 2

    },

    {

        question: "What does DOM stand for?",

        options: [

            "Document Object Model",

            "Display Object Management",

            "Digital Object Model",

            "Document Order Model"

        ],

        answer: 0

    },

    {

        question: "Which keyword is used to declare a variable in JavaScript?",

        options: ["var", "int", "string", "float"],

        answer: 0

    },

    {

        question: "Which method is used to fetch data from an API?",

        options: ["get()", "fetch()", "request()", "api()"],

        answer: 1

    }

];

let currentQuestion = 0;

let score = 0;

const questionEl = document.getElementById("question");

const optionButtons = document.querySelectorAll(".option");

const scoreEl = document.getElementById("score");

const nextBtn = document.getElementById("nextBtn");

function loadQuestion() {

    const q = quizData[currentQuestion];

    questionEl.textContent = q.question;

    optionButtons.forEach((btn, index) => {

        btn.textContent = q.options[index];

        btn.disabled = false;

    });

    nextBtn.style.display = "none";

}

function checkAnswer(selected) {

    if (selected === quizData[currentQuestion].answer) {

        score++;

    }

    optionButtons.forEach(btn => btn.disabled = true);

    nextBtn.style.display = "block";

}

function nextQuestion() {

    currentQuestion++;

    if (currentQuestion < quizData.length) {

        loadQuestion();

    } else {

        showScore();

    }

}

function showScore() {

    questionEl.textContent = "Quiz Completed!";

    document.querySelector(".options").style.display = "none";

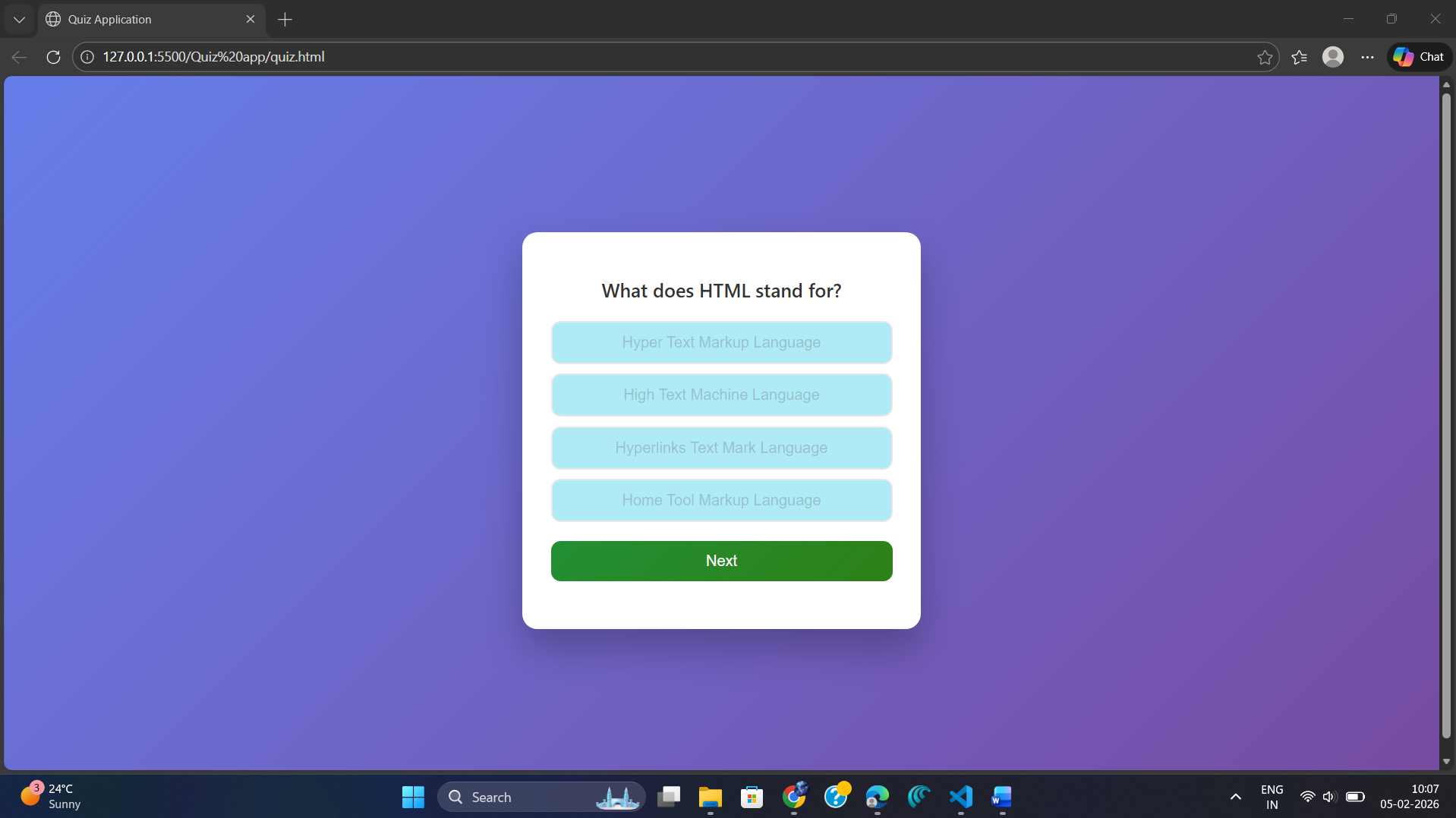
    nextBtn.style.display = "none";

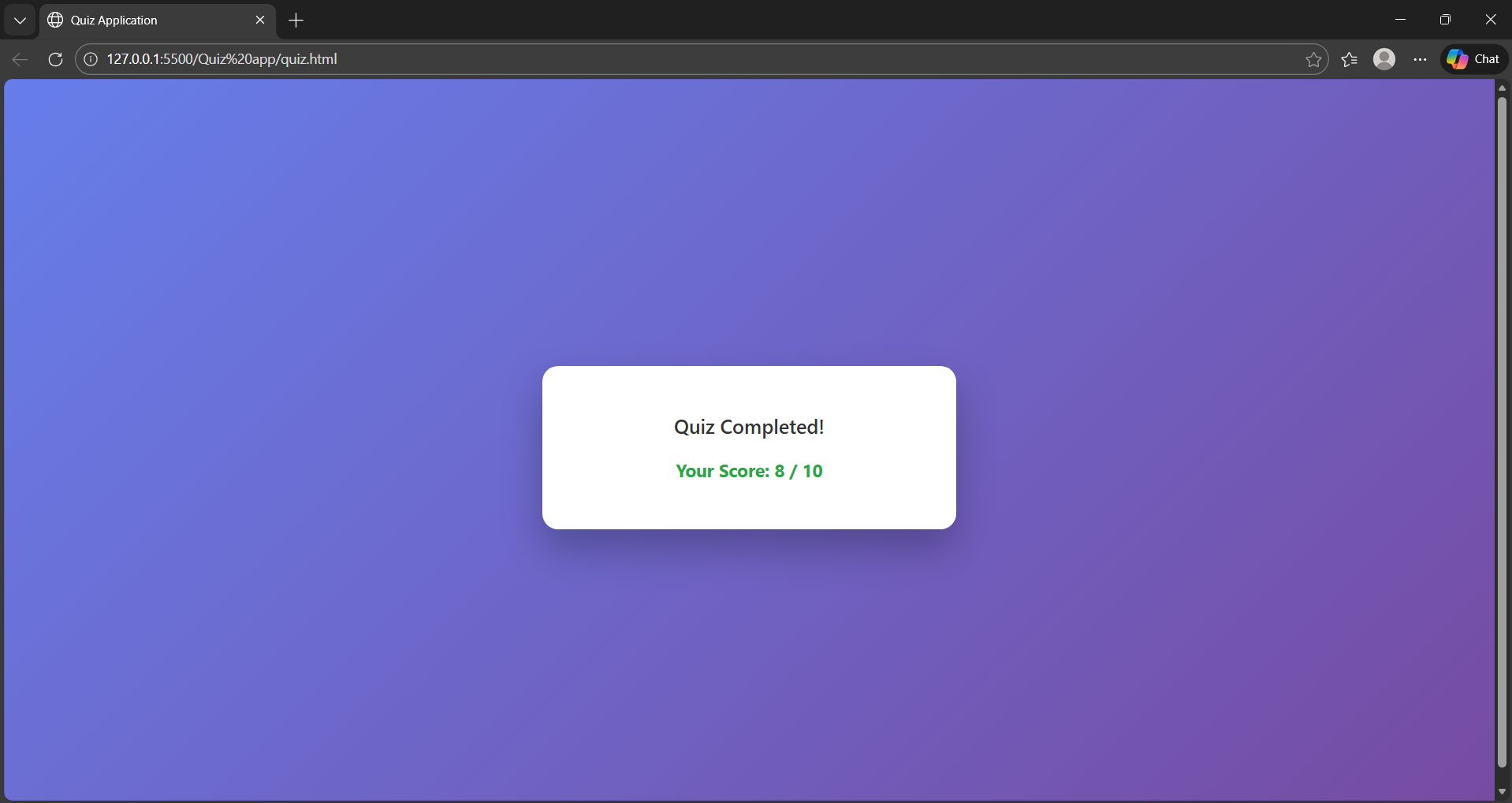
    scoreEl.textContent = `Your Score: ${score} / ${quizData.length}`;

}

loadQuestion();

**Output:**





**Experiment 8: Age Calculator**

**Objective: Create an application for calculating a user's current age.**

* Allow users to input their birthdate using HTML input elements.
* Style the interface with CSS for clarity and appeal.
* Use JavaScript to calculate and display the user’s age in years, months, and days.

**Code:**

**Html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Age Calculator</title>

    <link rel="stylesheet" href="age.css">

</head>

<body>

    <div class="container">

        <h1>Age Calculator</h1>

        <label for="birthdate">Enter Your Birthdate:</label>

        <input type="date" id="birthdate">

        <button onclick="calculateAge()">Calculate Age</button>

        <div id="result"></div>

    </div>

    <script src="age.js"></script>

</body>

</html>

**Css**

body {

    font-family: Arial, sans-serif;

    background: linear-gradient(135deg, #4e73df, #1cc88a);

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    margin: 0;

}

.container {

    background: white;

    padding: 30px;

    border-radius: 10px;

    box-shadow: 0 8px 20px rgba(0, 0, 0, 0.2);

    text-align: center;

    width: 350px;

}

h1 {

    margin-bottom: 20px;

}

input {

    padding: 10px;

    width: 100%;

    margin: 10px 0;

    font-size: 16px;

}

button {

    padding: 10px 20px;

    font-size: 16px;

    border: none;

    border-radius: 5px;

    background-color: #4e73df;

    color: white;

    cursor: pointer;

}

button:hover {

    background-color: #2e59d9;

}

#result {

    margin-top: 20px;

    font-size: 18px;

    font-weight: bold;

    color: #333;

}

**Js**

function calculateAge() {

    const birthdate = document.getElementById("birthdate").value;

    if (!birthdate) {

        document.getElementById("result").innerHTML = "Please select your birthdate.";

        return;

    }

    const birthDateObj = new Date(birthdate);

    const today = new Date();

    let years = today.getFullYear() - birthDateObj.getFullYear();

    let months = today.getMonth() - birthDateObj.getMonth();

    let days = today.getDate() - birthDateObj.getDate();

    if (days < 0) {

        months--;

        const previousMonth = new Date(today.getFullYear(), today.getMonth(), 0);

        days += previousMonth.getDate();

    }

    if (months < 0) {

        years--;

        months += 12;

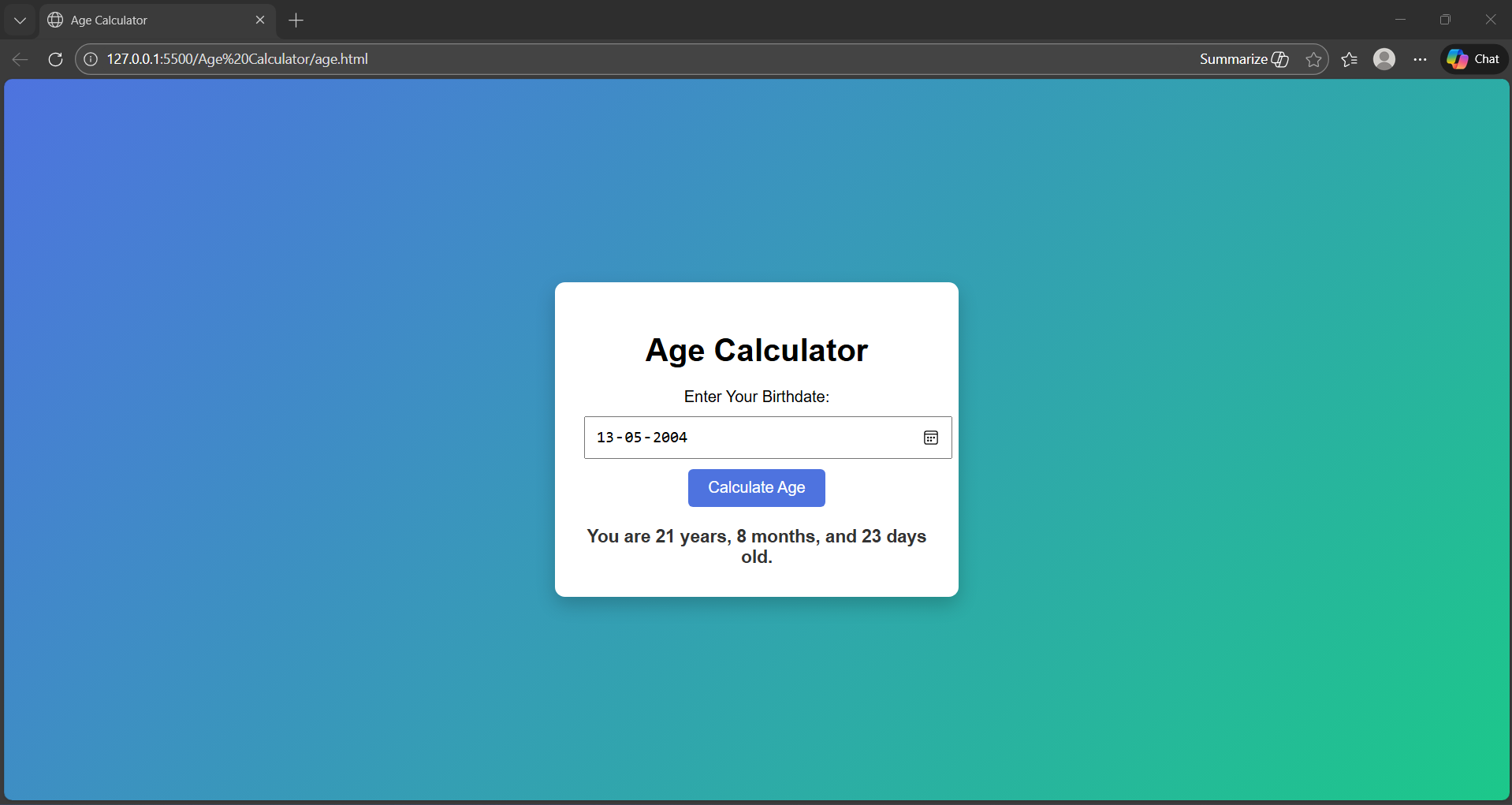
    }

    document.getElementById("result").innerHTML =

        `You are ${years} years, ${months} months, and ${days} days old.`;

}

**Output:**



**Experiment 9: Dice Roll Simulator**

**Objective: Develop a virtual dice rolling simulator.**

* Provide a user interface with HTML, including a button for rolling the dice.
* Use CSS to visually represent the dice and results.
* Employ JavaScript to generate random dice outcomes and display them.
* Maintain and display a history of previous dice rolls.

**Code:**

**Html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Dice Roll Simulator</title>

    <link rel="stylesheet" href="dice.css">

</head>

<body>

    <div class="container">

        <h1>🎲 Dice Roll Simulator</h1>

        <div class="dice" id="dice">1</div>

        <button onclick="rollDice()">Roll Dice</button>

        <h3>Roll History:</h3>

        <ul id="history"></ul>

    </div>

    <script src="dice.js"></script>

</body>

</html>

**Css**

body {

    font-family: Arial, sans-serif;

    background: linear-gradient(135deg, #ff9a9e, #fad0c4);

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    margin: 0;

}

.container {

    background: white;

    padding: 30px;

    border-radius: 10px;

    box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);

    text-align: center;

    width: 350px;

}

.dice {

    width: 100px;

    height: 100px;

    margin: 20px auto;

    background-color: #ffffff;

    border: 4px solid #333;

    border-radius: 15px;

    font-size: 40px;

    font-weight: bold;

    display: flex;

    justify-content: center;

    align-items: center;

}

button {

    padding: 10px 20px;

    font-size: 16px;

    border: none;

    border-radius: 5px;

    background-color: #ff6f61;

    color: white;

    cursor: pointer;

}

button:hover {

    background-color: #e85c50;

}

ul {

    list-style-type: none;

    padding: 0;

    max-height: 150px;

    overflow-y: auto;

    margin-top: 10px;

}

li {

    padding: 5px;

    border-bottom: 1px solid #ddd;

}

**Js**

let historyList = [];

function rollDice() {

    const dice = document.getElementById("dice");

    const history = document.getElementById("history");

    const randomNumber = Math.floor(Math.random() \* 6) + 1;

    dice.textContent = randomNumber;

    historyList.unshift(randomNumber);

    if (historyList.length > 10) {

        historyList.pop();

    }

    history.innerHTML = "";

    historyList.forEach((roll, index) => {

        const li = document.createElement("li");

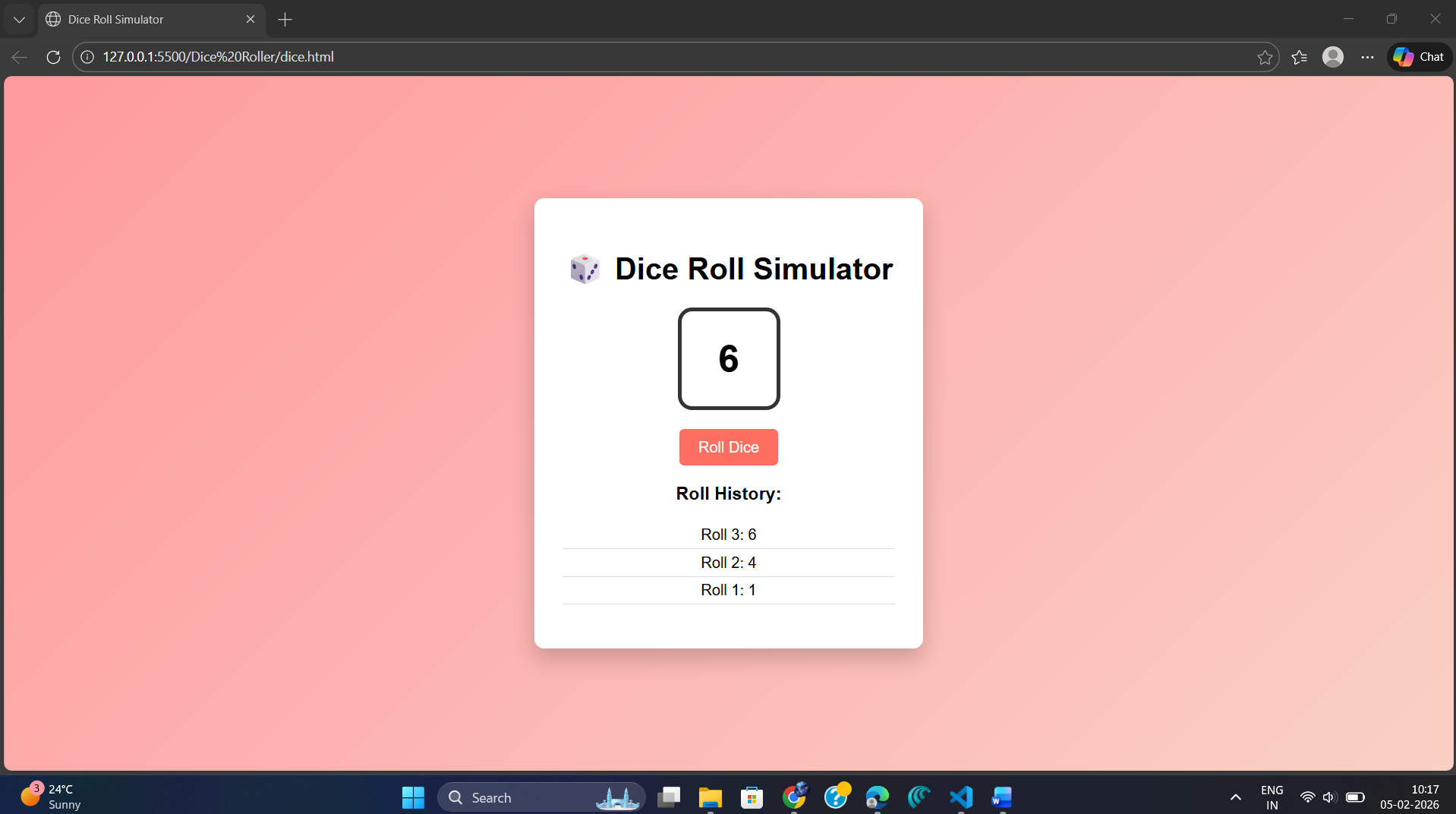
        li.textContent = `Roll ${historyList.length - index}: ${roll}`;

        history.appendChild(li);

    });

}

**Output:**



**Experiment 10: Whack-a-Mole Game**

**Objective: Build an interactive whack-a-mole game.**

* Design the game grid and layout using HTML.
* Apply CSS for visually engaging and responsive game elements.
* Utilize JavaScript to randomly display moles, detect user clicks, and update the score based on the user’s performance.

**Code:**

**Html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Whack-a-Mole Game</title>

    <link rel="stylesheet" href="mole.css">

</head>

<body>

    <div class="container">

        <h1>🎯 Whack-a-Mole</h1>

        <div class="info">

            <p>Score: <span id="score">0</span></p>

            <p>Time Left: <span id="time">30</span>s</p>

        </div>

        <div class="grid" id="grid">

            <!-- 9 squares -->

            <div class="square"></div>

            <div class="square"></div>

            <div class="square"></div>

            <div class="square"></div>

            <div class="square"></div>

            <div class="square"></div>

            <div class="square"></div>

            <div class="square"></div>

            <div class="square"></div>

        </div>

        <button onclick="startGame()">Start Game</button>

    </div>

    <script src="mole.js"></script>

</body>

</html>

**Css**

body {

    font-family: Arial, sans-serif;

    background: linear-gradient(135deg, #84fab0, #8fd3f4);

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    margin: 0;

}

.container {

    text-align: center;

    background: white;

    padding: 25px;

    border-radius: 12px;

    box-shadow: 0 8px 20px rgba(0, 0, 0, 0.2);

}

h1 {

    margin-bottom: 10px;

}

.info {

    display: flex;

    justify-content: space-around;

    margin-bottom: 15px;

    font-weight: bold;

}

.grid {

    display: grid;

    grid-template-columns: repeat(3, 100px);

    grid-gap: 10px;

    justify-content: center;

    margin-bottom: 15px;

}

.square {

    width: 100px;

    height: 100px;

    background-color: #ddd;

    border-radius: 10px;

    cursor: pointer;

    transition: 0.2s;

}

.square.mole {

    background-color: #8b4513;

}

button {

    padding: 8px 15px;

    border: none;

    border-radius: 5px;

    background-color: #ff6f61;

    color: white;

    cursor: pointer;

}

button:hover {

    background-color: #e85c50;

}

**Js**

const squares = document.querySelectorAll(".square");

const scoreDisplay = document.getElementById("score");

const timeDisplay = document.getElementById("time");

let score = 0;

let currentMole = null;

let gameTimer = null;

let moleTimer = null;

let timeLeft = 30;

function randomSquare() {

    squares.forEach(square => {

        square.classList.remove("mole");

    });

    const randomIndex = Math.floor(Math.random() \* squares.length);

    currentMole = squares[randomIndex];

    currentMole.classList.add("mole");

}

function startGame() {

    score = 0;

    timeLeft = 30;

    scoreDisplay.textContent = score;

    timeDisplay.textContent = timeLeft;

    // Show mole every 800ms

    moleTimer = setInterval(randomSquare, 800);

    // Countdown timer

    gameTimer = setInterval(() => {

        timeLeft--;

        timeDisplay.textContent = timeLeft;

        if (timeLeft <= 0) {

            clearInterval(gameTimer);

            clearInterval(moleTimer);

            alert("Game Over! Your final score is: " + score);

        }

    }, 1000);

}

// Detect clicks

squares.forEach(square => {

    square.addEventListener("click", () => {

        if (square === currentMole) {

            score++;

            scoreDisplay.textContent = score;

            square.classList.remove("mole");

            currentMole = null;

        }

    });

});

**Output:**

