**Objective:**

The goal of this web application is to make timezone conversions easier for users and to give them access to real-time weather information. Users can retrieve current weather information, such as temperature, wind speed, and humidity, by inputting their location using an easy-to-use interface. Users can also use the app to find the current time in other time zones by entering the capital of a particular nation. Error handling techniques are used to handle server problems or incorrect inputs sympathetically while maintaining a smooth user experience. The web app seeks to improve user engagement and give a useful tool for remaining updated about weather conditions and worldwide time differences by regularly updating the displayed information and incorporating interactive features.

**Code Structure:**

1**. HTML:**

The HTML code that is provided utilizes markup to arrange and structure the content of a webpage, which mostly consists of a weather app and timezone features. The type and version of the HTML document are declared first, and then metadata like character encoding and viewport settings are defined. Elements such as <div>, <input>, <button>, and <img> are frequently utilized in the page to organize the layout and include interactive elements. These components define different areas of the website, such as the time, date, and timezone displays and the input boxes for location search and timezone selection. To improve the page's appearance and usefulness, additional external resources are linked, such as CSS stylesheets and icon libraries.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Weather App</title>

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

    <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>

</head>

<body>

    <div class="clock" id="clock"></div>

    <div class="date" id="date"></div>

    <div class="timezone" id="timezone"></div>

    <div class="container">

        <div class="search-box">

            <i class='bx bx-map'></i>

            <input type="text" placeholder="enter your location">

            <button>Search</button>

        </div>

        <div class="weather-box">

            <div class="box">

                <div class="info-weather">

                    <div class="weather"></div>

                    <img src="image/cloudy.png" alt="Weather icon">

                    <p class="temperature">0<span>°C</span></p>

                    <p class="description"></p>

                </div>

            </div>

        </div>

        <div class="weather-details">

            <div class="humidity">

                <i class='bx bx-water'></i>

                <div class="text">

                    <div class="info-Humidity">

                        <span>0%</span>

                    </div>

                    <p>Humidity</p>

                </div>

            </div>

            <div class="Wind">

                <i class='bx bxl-tailwind-css'></i>

                <div class="text">

                    <div class="info-Wind">

                        <span>0Km/hr</span>

                    </div>

                    <p>Wind Speed</p>

                </div>

            </div>

        </div>

    </div>

    <div class="time-container">

        <div class="time-search-box">

            <input type="text" placeholder="Enter Capital of a Country" />

            <button>Search Timezones</button>

        </div>

        <div id="timezone-select-container" style="display: none;">

            <select id="timezone-select"></select>

            <button id="get-time-button">Get Time</button>

        </div>

        <div id="time-output"></div>

        <div id="time-error" class="error"></div>

    </div>

    <script src="main.js"></script>

</body>

</html>

2. **WebSocket Server**

Real-time bidirectional communication between the client and server is facilitated by WebSocket. The WebSocket.Server class from the ws library is used in this code to create the WebSocket server. A client logs a message identifying the connection when it establishes one with the WebSocket server.

const express = require('express');

const WebSocket = require('ws');

const app = express();

const PORT = process.env.PORT || 8081;

const wss = new WebSocket.Server({ noServer: true });

async function startServer() {

  let fetch;

  try {

    const fetchModule = await import('node-fetch');

    fetch = fetchModule.default;

  } catch (err) {

    console.error('Failed to load node-fetch with dynamic import:', err);

    return;

  }

  wss.on('connection', function connection(ws) {

    console.log('A client connected');

    ws.on('message', function incoming(message) {

      console.log('received: %s', message);

      fetchWeatherData(message)

        .then(data => {

          ws.send(JSON.stringify(data));

        })

        .catch(error => {

          console.error('Error fetching weather data:', error);

          ws.send(JSON.stringify({ error: 'Error fetching weather data' }));

        });

    });

    ws.send('Hello, client! You are connected to the WebSocket server.');

  });

  async function fetchWeatherData(city) {

    const APIKey = 'b9a1454443aaf648f44484f69396de38';

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

    const response = await fetch(url);

    const data = await response.json();

    return data;

  }

  app.server = app.listen(PORT, () => {

    console.log(`Server listening on port ${PORT}`);

  });

  app.server.on('upgrade', (request, socket, head) => {

    wss.handleUpgrade(request, socket, head, (ws) => {

      wss.emit('connection', ws, request);

    });

  });

}

startServer().catch(err => {

  console.error('Error starting server:', err);

});

3. **JavaScript:**

In this case, the JavaScript code acts as a link between the server-side functionality and the client-side UI. It connects to the server using a WebSocket, enabling bidirectional communication for updates in real time. It does things like collect weather data based on user input by listening for signals from the server.

const container = document.querySelector('.container');

const search = document.querySelector('.search-box button');

const weatherBox = document.querySelector('.weather-box');

const weatherDetails = document.querySelector('.weather-details');

const errorDisplay = document.createElement('p');

errorDisplay.className = 'error';

container.appendChild(errorDisplay);

function updateClock() {

    const now = new Date();

    let hours = now.getHours();

    const minutes = String(now.getMinutes()).padStart(2, '0');

    const seconds = String(now.getSeconds()).padStart(2, '0');

    const ampm = hours >= 12 ? 'PM' : 'AM';

    hours = hours % 12 || 12;

    document.getElementById('clock').textContent = `${hours}:${minutes}:${seconds} ${ampm}`;

    const month = now.toLocaleString('default', { month: 'long' });

    const day = now.getDate();

    const year = now.getFullYear();

    document.getElementById('date').textContent = `${month} ${day}, ${year}`;

    const timezone = Intl.DateTimeFormat().resolvedOptions().timeZone;

    document.getElementById('timezone').textContent = `Timezone: ${timezone}`;

}

updateClock();

setInterval(updateClock, 1000);

const ws = new WebSocket('ws://localhost:8081');

ws.onopen = function () {

    console.log('Connected to WebSocket server');

};

ws.onmessage = function (event) {

    console.log('Message from server:', event.data);

    if (event.data.startsWith('Hello, client!')) {

        console.log('Received welcome message from server:', event.data);

    } else {

        try {

            const weatherData = JSON.parse(event.data);

            handleWeatherData(weatherData);

        } catch (error) {

            console.warn('Received non-JSON message:', event.data);

            errorDisplay.textContent = 'Unknown error occurred';

        }

    }

};

ws.onclose = function () {

    console.log('Disconnected from WebSocket server');

};

function handleWeatherData(json) {

    if (json.cod === '404') {

        errorDisplay.textContent = 'City not found. Please try again.';

        weatherBox.style.display = 'none';

        weatherDetails.style.display = 'none';

        return;

    }

    errorDisplay.textContent = '';

    weatherBox.style.display = 'block';

    weatherDetails.style.display = 'flex';

    const image = document.querySelector('.weather-box img');

    const temperature = document.querySelector('.weather-box .temperature');

    const description = document.querySelector('.weather-box .description');

    const wind = document.querySelector('.weather-details .Wind span');

    const humidity = document.querySelector('.weather-details .humidity span');

    switch (json.weather[0].main) {

        case 'Clear':

            image.src = 'image/clear.png';

            break;

        case 'Clouds':

            image.src = 'image/cloudy.png';

            break;

        case 'Rain':

            image.src = 'image/rain.png';

            break;

        case 'Mist':

            image.src = 'image/mist.png';

            break;

        case 'Haze':

            image.src = 'image/haze.png';

            break;

        case 'Snow':

            image.src = 'image/snow.png';

            break;

        default:

            image.src = 'image/cloudy.png';

    }

    temperature.innerHTML = `${parseInt(json.main.temp)}<span>°C</span>`;

    description.innerHTML = json.weather[0].description;

    wind.innerHTML = `${parseInt(json.wind.speed)} Km/hr`;

    humidity.innerHTML = `${json.main.humidity}%`;

}

search.addEventListener('click', () => {

    const APIKey = 'b9a1454443aaf648f44484f69396de38';

    const city = document.querySelector('.search-box input').value;

    if (city === '') return;

    ws.send(city);

});

const timeSearchInput = document.querySelector('.time-search-box input');

const timeSearchButton = document.querySelector('.time-search-box button');

const timeZoneSelectContainer = document.getElementById('timezone-select-container');

const timeZoneSelect = document.getElementById('timezone-select');

const getTimeButton = document.getElementById('get-time-button');

const timeOutput = document.getElementById('time-output');

const timeError = document.getElementById('time-error');

const worldTimeAPI = 'https://worldtimeapi.org/api/timezone/';

const searchTimeZones = () => {

    const country = timeSearchInput.value;

    if (country === '') return;

    timeOutput.textContent = 'Loading, please wait...';

    timeError.textContent = '';

    fetch(worldTimeAPI)

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

        .then(data => {

            const filteredTimeZones = data.filter(timezone => timezone.toLowerCase().includes(country.toLowerCase()));

            if (filteredTimeZones.length === 0) {

                timeOutput.textContent = '';

                timeError.textContent = 'No timezones found for this country. Please try again.';

                timeZoneSelectContainer.style.display = 'none';

                return;

            }

            timeZoneSelect.innerHTML = filteredTimeZones.map(timezone => `<option value="${timezone}">${timezone}</option>`).join('');

            timeZoneSelectContainer.style.display = 'block';

            timeOutput.textContent = '';

        })

        .catch(() => {

            timeOutput.textContent = '';

            timeError.textContent = 'Something went wrong. Please try again later.';

            timeZoneSelectContainer.style.display = 'none';

        });

};

const getTimeForTimeZone = () => {

    const selectedTimeZone = timeZoneSelect.value;

    if (!selectedTimeZone) return;

    timeOutput.textContent = 'Loading, please wait...';

    timeError.textContent = '';

    fetch(`${worldTimeAPI}${selectedTimeZone}`)

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

        .then(data => {

            if (!data.datetime) {

                timeOutput.textContent = '';

                timeError.textContent = 'Could not retrieve time. Please try again.';

                return;

            }

            const timeString = data.datetime.slice(11, 16);

            const hours = parseInt(timeString.slice(0, 2));

            const minutes = timeString.slice(3);

            const ampm = hours >= 12 ? 'PM' : 'AM';

            const formattedHours = hours % 12 || 12;

            timeOutput.textContent = `Time in ${selectedTimeZone}: ${formattedHours}:${minutes} ${ampm}`;

            timeOutput.style.fontSize = '2rem';

            const containerHeight = container.offsetHeight;

            const outputHeight = timeOutput.offsetHeight;

            const topPosition = (containerHeight - outputHeight) / 2.5;

            timeOutput.style.position = 'absolute';

            timeOutput.style.right = '28%';

            timeOutput.style.top = `${topPosition}px`;

        })

        .catch(() => {

            timeOutput.textContent = '';

            timeError.textContent = 'Something went wrong. Please try again later.';

        });

};

timeSearchButton.addEventListener('click', searchTimeZones);

getTimeButton.addEventListener('click', getTimeForTimeZone);

4. **CSS styles:** The purpose of the CSS stylesheet is to improve and stylize the HTML parts of your web application's visual presentation.

@import url('https://fonts.googleapis.com/css2?family=Oswald:wght@200..700&display=swap');

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

    font-family: Oswald;

}

.clock {

    font-size: 3rem;

    position: absolute;

    top: 10px;

    left: 10px;

}

.date {

    font-size: 1.5rem;

    margin-top: 10px;

    position: absolute;

    top: 60px;

    left: 10px;

}

.timezone {

    font-size: 1rem;

    position: absolute;

    top: 101px;

    left: 10px;

}

body {

    display: flex;

    justify-content: center;

    align-items: center;

    min-height: 100vh;

    background: url(image/bg.jpg);

    background-size: cover;

    background-position: center;

}

button {

    padding: 0.8em 1.8em;

    border: 2px solid #ffffff;

    position: absolute;

    right: 0;

    width: 80px;

    height: 100%;

    overflow: hidden;

    background-color: transparent;

    text-align: center;

    text-transform: uppercase;

    font-size: 12px;

    transition: .3s;

    z-index: 1;

    font-family: inherit;

    color: #ffffff;

    outline: none;

    border-radius: 10px;

}

button::before {

    content: '';

    width: 0;

    height: 300%;

    position: absolute;

    top: 50%;

    left: 50%;

    transform: translate(-50%, -50%) rotate(45deg);

    background: #17C3B2;

    transition: .5s ease;

    display: block;

    z-index: -1;

}

button:hover::before {

    width: 105%;

}

button:hover {

    color: #111;

}

.button-2 {

    padding: 0.8em 1.8em;

    border: 2px solid #ffffff;

    width: 100px;

    height: 35%;

    overflow: hidden;

    background-color: transparent;

    text-align: center;

    text-transform: uppercase;

    font-size: 12px;

    transition: .3s;

    z-index: 1;

    font-family: inherit;

    color: #ffffff;

    outline: none;

    border-radius: 10px;

}

.button-2::before {

    content: '';

    width: 0;

    height: 300%;

    position: absolute;

    top: 50%;

    left: 50%;

    transform: translate(-50%, -50%) rotate(45deg);

    background: #17C3B2;

    transition: .5s ease;

    display: block;

    z-index: -1;

}

.button-2:hover::before {

    width: 105%;

}

.button-2:hover {

    color: #111;

}

.container {

    position: relative;

    width: 400px;

    height: 600px;

    background: rgba(182, 150, 81, .7);

    border-radius: 18px;

    padding: 20px;

    color: #ffffff;

}

.search-box {

    position: relative;

    width: 100%;

    height: 55px;

    display: flex;

    align-items: center;

}

.search-box input {

    position: absolute;

    width: 100%;

    height: 100%;

    background: transparent;

    border: 2px solid rgb(238, 234, 234);

    outline: none;

    border-radius: 10px;

    font-size: 22px;

    color: beige;

    font-weight: 500;

    text-transform: uppercase;

    padding: 0 48px 0 42px;

}

.search-box input::placeholder {

    color: aquamarine;

    text-transform: capitalize;

}

.search-box i {

    position: absolute;

    left: 10px;

    font-size: 28px;

}

.weather-box {

    text-align: center;

    margin: 40px 0;

}

.weather-box img {

    width: 60%;

}

.weather-box .temperature {

    position: relative;

    font-size: 64px;

    line-height: 1;

    font-weight: 700;

    margin: 20px 0 6px -30px;

}

.weather-box .temperature span {

    position: absolute;

    font-size: 24px;

    margin-left: 4px;

}

.weather-box .description {

    font-size: 22px;

    font-weight: 500;

    text-transform: capitalize;

}

.weather-details {

    position: absolute;

    bottom: 40px;

    left: 0;

    width: 100%;

    padding: 0 20px;

    display: flex;

}

.weather-details .humidity {

    display: flex;

    align-items: center;

    width: 50%;

}

.weather-details .humidity {

    padding-left: 20px;

    justify-content: flex-start;

}

.weather-details .Wind {

    display: flex;

    align-items: center;

    width: 50%;

}

.weather-details .Wind {

    padding-right: 20px;

    justify-content: flex-start;

}

.weather-details i {

    font-size: 56px;

    margin-right: 10px;

}

.weather-details span {

    display: inline-block;

    font-size: 22px;

    font-weight: 500;

}

.weather-details p {

    font-size: 14px;

    font-weight: 500;

}

.error {

    position: absolute;

    top: 50%;

    left: 50%;

    transform: translate(-50%, -50%);

    color: rgb(255, 255, 255);

    font-size: 18px;

    text-align: center;

}

.time-container {

    position: relative;

    width: 600px;

    height: 500px;

    background: rgba(182, 150, 81, .7);

    border-radius: 18px;

    padding: 20px;

    color: #ffffff;

    margin-left: 30px;

}

.time-search-box input {

    position: absolute;

    width: 50%;

    height: 10%;

    background: transparent;

    border: 2px solid rgb(238, 234, 234);

    outline: none;

    border-radius: 10px;

    font-size: 22px;

    color: white;

    font-weight: 500;

    text-transform: uppercase;

    padding: 0 48px 0 42px;

}

.time-search-box input::placeholder {

    color: white;

    font-size: 15px;

}

.time-search-box button {

    padding: 0.8em 1.8em;

    border: 2px solid #ffffff;

    width: 150px;

    height: 45px;

    overflow: hidden;

    background-color: transparent;

    text-align: center;

    text-transform: uppercase;

    font-size: 12px;

    transition: .3s;

    z-index: 1;

    font-family: inherit;

    color: #ffffff;

    outline: none;

    border-radius: 10px;

    margin-right: 40px;

    margin-top: 5px;

}

.time-search-box button::before {

    content: '';

    width: 0;

    height: 300%;

    position: absolute;

    top: 50%;

    left: 50%;

    transform: translate(-50%, -50%) rotate(45deg);

    background: #17C3B2;

    transition: .5s ease;

    display: block;

    z-index: -1;

}

.time-search-box button:hover::before {

    width: 105%;

}

.time-search-box button:hover {

    color: #111;

}

#timezone-select-container button {

    padding: 0.8em 1.8em;

    border: 2px solid #ffffff;

    width: 150px;

    height: 45px;

    overflow: hidden;

    background-color: transparent;

    text-align: center;

    text-transform: uppercase;

    font-size: 12px;

    transition: .3s;

    z-index: 1;

    font-family: inherit;

    color: #ffffff;

    outline: none;

    border-radius: 10px;

    margin-top: 60px;

    margin-right: 40px;

}

.timezone-select-container button::before {

    content: '';

    width: 0;

    height: 300%;

    position: absolute;

    top: 50%;

    left: 50%;

    transform: translate(-50%, -50%) rotate(45deg);

    background: #17C3B2;

    transition: .5s ease;

    display: block;

    z-index: -1;

}

.timezone-select-container button:hover::before {

    width: 105%;

}

.timezone-select-container button:hover {

    color: #111;

}

#timezone-select-container {

    display: flex;

    flex-direction: column;

}

#timezone-select-container select {

    margin-top: 60px;

}