Middleware Technologies

CSIWZG524

Assignment 1.1

Name: HARI SHANKAR
Student ID : 2021WA86905

🌤️ Weather Forecast Application

A sleek, responsive weather app that provides real-time weather updates using the **OpenWeatherMap API**.

- **GitHub Repository Link:-**

https://github.com/2021wa86905/Weather-Forecast-Application.git

- **Application Link:-**

https://2021wa86905.github.io/Weather-Forecast-Application/

✨ Features

- 🔍 **Search for Any City**: Get instant weather details by entering a city name.
- 🌈 **Dynamic Weather Icons**: Displays weather-specific icons (e.g., 🌧️ Rain, 🌞 Clear, 🌫️ Mist).
- 🚫 **Error Handling**: Informs users when an invalid city name is entered.
- 📱 **Responsive Design**: Works beautifully across devices.

🛠️ Technologies Used

- **Frontend**: HTML5, CSS3
- **Styling**: Modern CSS techniques, including gradients and responsiveness
- **Programming**: JavaScript (ES6+ features)
- **API**: OpenWeatherMap API

🚀 How to Run the Project

Follow these steps to get the app up and running on your local machine:

1️⃣ Clone the Repository

2️⃣ Open the Project

- Navigate to the project folder.

- Open the index.html file in your browser.

3️⃣ Configure the API Key

- Get your free API key from OpenWeatherMap.
- Replace the placeholder api key in the index.html file:

🌟 How It Works

1. Type a city name in the input field.
2. Click the **Search** button to fetch the weather data.
3. View the following details:
    - 🌡️ **Temperature**: Current temperature in °C
    - 💧 **Humidity**: Atmospheric humidity in %
    - 💨 **Wind Speed**: Wind velocity in km/h
    - ☁️ **Weather Icon**: Represents the current weather condition
4. Invalid city names display a friendly error message.

📂 Project Structure

```
📁 weather-app/
├── index.html      # Main structure and JavaScript functionality
├── style.css       # UI styling
└── images/         # Weather condition icons
```

📖 Learning Points

- **Asynchronous JavaScript**: Used async/await for API calls.
- **Error Handling**: Gracefully handled incorrect inputs with conditionals.
- **Dynamic DOM Updates**: Utilized JavaScript to update elements dynamically.

- **CSS Styling**: Experimented with gradients and flexible layouts.

🔮 Future Enhancements

- 🗓️ Add a **7-day weather forecast** feature.
- 📍 Integrate **geolocation** to fetch weather data automatically.
- 💾 Allow users to **save their favorite cities**.

🌐 Web Services Architecture Overview

The application uses a **client-server architecture** where:

- The **frontend** (HTML/CSS/JavaScript) runs in the browser.
- The **OpenWeatherMap API** acts as the backend service providing real-time weather data.

📈 Scalability Considerations

Scalability refers to the system's ability to handle increased load without performance degradation.

🔹 Current Architecture

- The app is **client-heavy** with minimal backend logic.
- All weather data is fetched directly from the **OpenWeatherMap API**.

🔹 Challenges

- **API Rate Limits**: OpenWeatherMap imposes limits on free-tier API usage.
- **No Backend Caching**: Every request hits the API, which can become costly and slow under high usage.

🔹 Improvements

- **Introduce a Backend Layer**:
  - Use Node.js/Express or Python Flask to act as a proxy.
  - Implement **caching** (e.g., Redis) to store frequent queries.

- **Use CDN for Static Assets**:
  - Host HTML/CSS/JS/images on a CDN to reduce load times globally.
- **Horizontal Scaling**:
  - Deploy the backend on scalable platforms like AWS Lambda, Azure Functions, or Kubernetes.

🛡️ Fault Tolerance Considerations

Fault tolerance ensures the app continues to function even when parts of the system fail.

🔹 Current Architecture

- Minimal error handling is implemented (e.g., invalid city name).
- No retry mechanism or fallback service.

🔹 Challenges

- **API Downtime**: If OpenWeatherMap is down, the app fails.
- **Network Failures**: No retry logic for transient network issues.

🔹 Improvements

- **Retry Logic**:
  - Implement exponential backoff for failed API calls.
- **Fallback Mechanism**:
  - Use a secondary weather API (e.g., WeatherAPI, AccuWeather) as a backup.
- **Graceful Degradation**:
  - Show cached or last-known data if the API is unreachable.
- **Monitoring & Alerts**:
  - Use tools like Sentry or LogRocket to monitor frontend errors.

⚡ Performance Considerations

Performance ensures the app responds quickly and efficiently.

🔹 Current Architecture

- Lightweight frontend with async API calls.
- No backend processing or optimization.

🔹 **Challenges**

- **Cold Start**: First-time API calls may be slow.
- **Unoptimized DOM Updates**: All DOM elements are updated even if only one value changes.

🔹 **Improvements**

- **Lazy Loading Icons**:
  - Load weather icons only when needed.
- **Minify & Bundle Assets**:
  - Use tools like Webpack or Parcel to reduce file sizes.
- **Efficient DOM Manipulation**:
  - Use frameworks like React or Vue for better state management.
- **Service Workers**:
  - Cache static assets and even API responses for offline support.

🧠 **Summary Table**

Aspect	Current State	Suggested Enhancements
Scalability	Direct API calls, no caching	Add backend with caching, use CDN, horizontal scaling
Fault Tolerance	Basic error handling only	Retry logic, fallback APIs, graceful degradation
Performance	Async JS, no optimization	Lazy loading, minification, service workers

Application working snips:-

Delhi
26°c
Delhi
94%
humidity
1.54 km/h
wind speed

Jammu
25°c
Jammu
81%
humidity
2.57 km/h
wind speed

Bangalore
21°c
Bengaluru
88%
humidity
3.09 km/h
wind speed

INDEX code –[HTML]

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Weather app - Hari Shankar</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="card">
    <div class="search">
      <input type="text" placeholder="enter city name" spellcheck="false">
      <button><img src="images/search.png"></button>
    </div>
    <div class="error">
      <p>invalid city name</p>
    </div>
    <div class="weather">
      <img src="images/rain.png" class="weather-icon">
      <h1 class="temp">22°c</h1>
      <h2 class="city">new york</h2>
      <div class="details">
        <div class="col">
          <img src="images/humidity.png">
          <div>
            <p class="humidity">50%</p>
```

```
            <p>humidity</p>
          </div>
        </div>
        <div class="col">
          <img src="images/wind.png">
          <div>
            <p class="wind">15km/h</p>
            <p>wind speed</p>
          </div>
        </div>
      </div>
    </div>
  </div>

  <script>
    const apikey = "17979f41718e81f87596320b69c4ea01";
    const apiurl = "https://api.openweathermap.org/data/2.5/weather?units=metric&q=";

    const searchBox = document.querySelector(".search input");
    const searchBtn = document.querySelector(".search button");
    const weatherIcon = document.querySelector(".weather-icon");

    async function checkweather(city){
      const response = await fetch(apiurl + city + `&appid=${apikey}` );

      if(response.status == 404){
        document.querySelector(".error").style.display = "block";
```

```
        document.querySelector(".weather").style.display = "none";
    }else{
        var data = await response.json();

document.querySelector(".city").innerHTML= data.name;
document.querySelector(".temp").innerHTML= Math.round(data.main.temp) +"°c";
document.querySelector(".humidity").innerHTML= data.main.humidity +"%";
document.querySelector(".wind").innerHTML= data.wind.speed +" km/h";

if(data.weather[0].main == "Clouds"){
  weatherIcon.src = "images/clouds.png";
}
else if(data.weather[0].main == "Clear"){
  weatherIcon.src = "images/clear.png";
}
else if(data.weather[0].main == "Rain"){
  weatherIcon.src = "images/Rain.png";
}
else if(data.weather[0].main == "Drizzle"){
  weatherIcon.src = "images/Drizzle.png";
}
else if(data.weather[0].main == "Mist"){
  weatherIcon.src = "images/Mist.png";
}

document.querySelector(".weather").style.display = "block";
document.querySelector(".error").style.display = "none";
```

```
        }



    }



    searchBtn.addEventListener("click", ()=>{
        checkweather(searchBox.value);
    })
  </script>
</body>
</html>
```

CSS code [Java script]

```
*{
  margin: 0;
  padding: 0;
```

```
    font-family: 'poppins', sans-serif;
    box-sizing: border-box;
}
body{
    background: #222;
}
.card{
    width: 90%;
    max-width: 470px;
    background: linear-gradient(135deg, #00feba, #5b548a);
    color: #fff;
    margin: 100px auto 0;
    border-radius: 20px;
    padding: 40px 35px;
    text-align: center;
}
.search{
    width: 100%;
    display: flex;
    align-items: center;
    justify-content: space-between;
}
.search input{
    border: 0;
    outline: 0;
    background: #ebfffc;
    color: #555;
    padding: 10px 25px;
    height: 60px;
    border-radius: 30px;
    flex: 1;
```

```
    margin-right: 16px;
    font-size: 18px;
}
.search button{
    border: 0;
    outline: 0;
    background: #ebfffc;
    border-radius: 50%;
    width: 60px;
    height: 60px;
    cursor: pointer;
}
.search button img{
    width: 16px;
}
.weather-icon{
    width: 170px;
    margin-top: 30px;
}
.weather h1{
    font-size: 80px;
    font-weight: 500;
}
.weather h2{
    font-size: 45px;
    font-weight: 400;
    margin-top: -10px;
}
.details{
    display: flex;
    align-items: center;
```

```
    justify-content: space-between;
    padding: 0 20px;
    margin-top: 50px;
}
.col{
    display: flex;
    align-items: center;
    text-align: left;
}
.col img{
    width: 40px;
    margin-right: 10px;
}
.humidity, .wind{
    font-size: 28px;
    margin-top: -6px;
}
.weather{
    display: none;
}
.error{
    text-align: left;
    margin-left: 10px;
    font-size: 14px;
    margin-top: 10px;
    display: none;
}
```