**Read me**

**Weather Website**

**Overview**

This weather website is a comprehensive web application that provides real-time weather information and forecasts for cities worldwide. It combines data visualization, user-friendly interfaces, and a chatbot feature to deliver a rich user experience for weather enthusiasts and casual users alike.

**Features**

1. Current Weather Display : Shows temperature, humidity, wind speed, and weather description
2. 5-Day Weather Forecast : Provides a detailed outlook for the coming days
3. Interactive Charts :
4. Bar chart for temperature comparison
5. Doughnut chart for humidity visualization
6. Line chart for temperature trends
7. Unit Conversion : Toggle between Celsius and Fahrenheit
8. Chatbot : AI-powered assistant for general queries and weather-related questions
9. Responsive Design : Optimized for both desktop and mobile devices
10. Dynamic Background : Changes based on current weather conditions

Project Structure

weather-website/

│

├── dashboard.html # Main dashboard page

├── table.html # Weather forecast table page

├── data.js # API calls and data handling

├── weather.js # Weather data processing and chart creation

├── styles/

│ └── main.css # (Not provided, but assumed) Main stylesheet

├── images/ # (Not provided, but referenced in code)

│ └── weather-icons/ # Weather condition icons

└── README.md # This file

```

**Technologies Used**

- HTML5 for structure

- CSS3 for styling (assumed, not provided in the files)

- JavaScript (ES6+) for client-side logic

- Chart.js for data visualization

- OpenWeather API for weather data

- Gemini API for chatbot functionality

**Setup and Installation**

1. Clone the repository:

```

git clone https://github.com/yourusername/weather-website.git

```

2. Navigate to the project directory:

```

cd weather-website

```

3. Open `dashboard.html` in a modern web browser to view the main dashboard.

Note: This project doesn't require a server to run, but using a local server (like Live Server in VS Code) can help avoid CORS issues with API requests.

Usage Guide

1. Search for a City :

- Enter a city name in the search bar on the dashboard.

- Press Enter or click the search button.

2. View Weather Information :

- Current weather conditions will be displayed in the main weather widget.

- Charts will update to show temperature, humidity, and trend data.

3. Check Detailed Forecast :

- Navigate to the 'Tables' page using the sidebar to view the 5-day forecast.

4. Use the Chatbot :

- Type your query in the chat input at the bottom of the page.

- For weather-specific queries, start with "weather in [city name]".

- For general questions, simply type your query.

5. Switch Temperature Units :

- Use the unit toggle (Celsius/Fahrenheit) to change the temperature display.

**API Integration**

This project integrates two main APIs:

1. OpenWeather API :

- Used for fetching current weather and forecast data.

- API Key: `e1750e0ec352fa0a033fc738086e03e8`

- Endpoints used:

- Current weather: `https://api.openweathermap.org/data/2.5/weather`

- 5-day forecast: `https://api.openweathermap.org/data/2.5/forecast`

2. Gemini API :

- Powers the chatbot functionality for general queries.

- API Key: `AIzaSyCDBgE08ME\_jyHJNE7jYPWBZqGDOFSlehA`

Security Note : In a production environment, API keys should be kept secret and not exposed in client-side code. Consider using environment variables or a backend proxy to secure these keys.

**Customization**

To customize the website:

1. Modify `dashboard.html` and `table.html` for layout changes.

2. Update `weather.js` to alter chart configurations or add new visualizations.

3. Adjust the chatbot's behavior in `data.js`.

4. Add or modify CSS styles in `main.css` (not provided, but assumed to exist).

Performance Optimization

- The code uses debounce function to limit API calls, reducing unnecessary requests.

- Charts are destroyed and recreated to prevent memory leaks.

- Consider implementing caching mechanisms for frequently accessed data.

Security Considerations

1. API Key Protection : Move API key handling to a server-side component.

2. Input Sanitization : Implement proper sanitization for user inputs to prevent XSS attacks.

3. HTTPS : Ensure the website is served over HTTPS to encrypt data in transit.

**Testing**

(This section would typically include information about unit tests, integration tests, and end-to-end tests. As no testing files were provided, you may want to add testing in the future.)

**Deployment**

To deploy this website:

1. Choose a web hosting service (e.g., GitHub Pages, Netlify, or a traditional web host).

2. Upload all files to your hosting service.

3. Ensure that `dashboard.html` is set as the entry point (often renamed to `index.html`).

4. Configure your domain and DNS settings if using a custom domain.

**Contributing**

We welcome contributions! Please follow these steps:

1. Fork the repository

2. Create a new branch (`git checkout -b feature/AmazingFeature`)

3. Make your changes

4. Commit your changes (`git commit -m 'Add some AmazingFeature'`)

5. Push to the branch (`git push origin feature/AmazingFeature`)

6. Open a Pull Request

**Troubleshooting**

Common issues and solutions:

- API calls failing : Check your internet connection and verify API key validity.

- Charts not rendering : Ensure Chart.js is properly loaded and check browser console for errors.

- Chatbot not responding : Verify Gemini API key and check the console for any error messages.

**Future Enhancements**

- Implement user accounts for saving favorite locations

- Add more detailed weather information (e.g., UV index, air quality)

- Integrate weather alerts and notifications

- Expand chatbot capabilities with more weather-specific knowledge

**Contact**

For questions, suggestions, or issues, please open an issue in this repository or contact the maintainer at [i222448@nu.edu.pk].