

# QnA System with Weather Information Integration Documentation

By: Shimaa Ahmed

## Table of Contents

1. Use Case Definition
2. Development Phase
  - Tools and Libraries Used
  - Detailed Development Steps
  - Creating User Interface (UI)
3. Deployment Phase
  - Required Files
  - Deployment Steps
4. Test Phase
  - Example Queries and Expected Responses
5. User Manual
  - Using the QnA System
  - Screenshots

## 1. Use Case Definition

The QnA system with weather information integration is designed to handle natural language queries related to both NLP domain and real-time weather information. Users can ask questions about various topics, and if the query is weather-related, the system will provide current weather data for the specified location. This system aims to serve as a comprehensive assistant that can seamlessly switch between answering NLP questions and providing up-to-date weather information.

## 2. Development Phase

### Tools and Libraries Used

- **Python:** The primary programming language used for development.
- **Flask:** A micro web framework for Python used to create the web application.
- **OpenAI API:** Used to process natural language queries and determine the context.
- **Requests:** A library for making HTTP requests to fetch weather data.

## Detailed Development Steps

1. **Setup Flask Application:**
  - Create a new directory for the project and initialize a Flask application.
  - Install Flask and set up a basic route to handle incoming requests.
2. **Integrate OpenAI API:**
  - Install the OpenAI library and set up the API key.
  - Create a function to handle queries by sending them to the OpenAI API.
  - The OpenAI model is instructed to determine if the query is about the weather and identify the location if applicable. If the query is not about the weather, it responds as a domain expert in natural language processing.
3. **Fetch Weather Data:**
  - Sign up for a weather API service and obtain an API key.
  - Install the requests library and create a function to fetch weather data for a given location.
  - This function will be used to retrieve real-time weather information based on the identified location from the user's query.
4. **Return Response:**
  - Modify the Flask route to handle incoming queries and return appropriate responses.
  - The system parses the response from OpenAI to check if it includes a location. If a location is identified, it fetches the current weather data for that location and returns it to the user. If no location is identified, it returns the response from the OpenAI API directly.

## Creating User Interface (UI)

1. **Setup HTML Template:**
  - Create an HTML template using Bootstrap for a responsive and user-friendly interface.
  - Design the form to take user input and display the response from the server.
2. **Serve the HTML Template:**
  - Modify the Flask application to render the HTML template.
  - Implement JavaScript to handle form submission and display the response without reloading the page.

## 3. Deployment Phase

### Required Files

- **main.py:** The main Flask application file.
- **templates/index.html:** The HTML template for the user interface.
- **requirements.txt:** A file listing all the required libraries and their versions.
- **app.yaml:** Configuration file for deploying the application on a cloud platform.

## Deployment Steps

1. **Prepare the Environment:**
  - Ensure all required files are present and properly configured.
  - Set up environment variables for the OpenAI API key and the weather API key.
  - Ensure Python and necessary libraries are installed.
  - Install dependencies using `pip install -r requirements.txt`.
2. **Configure Google App Engine:**
  - Create a Google Cloud project.
  - Set up Google App Engine and configure it with the `app.yaml` file.
3. **Deploy the Application:**
  - Use the Google Cloud SDK to deploy the application.  
`gcloud app deploy`

## 4. Test Phase

### Example Queries and Expected Responses

1. **Weather Query: "What's the current weather in Paris"**
  - Expected Response: Contains weather information for Paris, e.g., "The current temperature in Paris is 15°C".
2. **Non-Weather Query: "What's attention in transformers"**
  - Expected Response: Explanation about attention mechanism in transformers, e.g., "Attention is a mechanism in transformers that allows the model to focus on relevant parts of the input when generating output."
3. **Ambiguous Weather Query: "Is it cold"**
  - Expected Response: Prompt to specify a location, e.g., "Please specify a location to get the weather information."
4. **Non-Weather Query: "Give 5 examples of Large Language Models"**
  - Expected Response: List of language models, e.g., "Examples of large language models include GPT-3, BERT, T5, XLNet, and RoBERTa."

## 5. User Manual

### Using the QnA System

1. **Access the System:**
  - Open a web browser and navigate to the URL where the system is deployed.  
<https://qnaweathersystem.de.r.appspot.com>
2. **Ask a Question:**
  - Enter your question in the input box and click the "Submit" button.
  - The system will process your query and display the response on the same page.

### 3. Use the Realtime Weather Service

- directly Use the realtime weather service
- specify the location and unit either "celsius or fahrenheit" and click get weather button

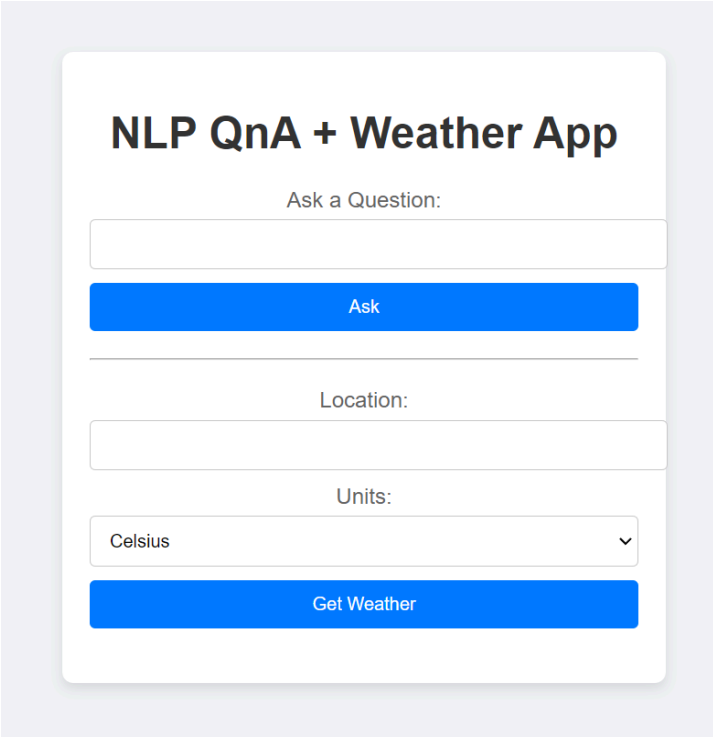
### 4. Example Queries:

- "What's the current weather in New York?"
- "What is the temperature in London?"
- "Explain the concept of neural networks."

## Screenshots

### 1. Home Page:

- Screenshot of the home page showing the input form.



The screenshot shows a web application titled "NLP QnA + Weather App". It features a form with the following elements:

- A heading "NLP QnA + Weather App" in bold black text.
- A label "Ask a Question:" above a text input field.
- A blue button labeled "Ask" below the first input field.
- A horizontal line separator.
- A label "Location:" above a text input field.
- A label "Units:" above a dropdown menu.
- The dropdown menu is currently set to "Celsius" with a downward arrow icon.
- A blue button labeled "Get Weather" at the bottom of the form.

○

### 2. Query Response:

- Screenshot of the page after submitting a weather-related query.

## NLP QnA + Weather App

Ask a Question:

Ask

**Question:** What is the temperature in London  
**Answer:** The real time weather report for <london> is: temperature:25.35

---

Location:

Units:

Celsius ▾

Get Weather

- Screenshot of the page after submitting a general knowledge query.

## NLP QnA + Weather App

Ask a Question:

Ask

**Question:** define attention in transformers  
**Answer:** As an expert in natural language processing, I can help clarify that in the context of transformers, attention refers to the mechanism that allows the model to weigh the importance of different words in a sentence when processing and generating text. It enables the model to focus on relevant parts of the input sequence and has been a key component in the success of transformer-based models such as BERT and GPT.

---

Location:

Units:

Celsius ▾

Get Weather

- Screenshot of the page after submitting a direct realtime get weather .

The screenshot displays a web application titled "NLP QnA + Weather App". It features a form with three input fields: "Ask a Question:", "Location:", and "Units:". The "Location:" field contains the text "Cairo", and the "Units:" dropdown menu is set to "Fahrenheit". A blue "Get Weather" button is positioned below the "Units:" field. At the bottom of the form, a light gray box displays the results: "Location: Cairo" and "Temperature: 88.56°F".

**NLP QnA + Weather App**

Ask a Question:

Ask

Location:

Cairo

Units:

Fahrenheit

Get Weather

**Location:** Cairo  
**Temperature:** 88.56°F

## Source Code and Deployment files

- <https://github.com/shimaa-ahmed/QNA-System>