University of Glasgow

School of
Computing Science

**NAME**: Ziyi Wang

**ID NUMBER**: 3026887w

**DATE**: 6 March, 2022

# Assessed Coursework

# COMPSCI5094 – Conversational Interfaces (M)

**2025**

## Part A

### Question 1: Weather agent development

For my project, I used Dialogflow as the NLU platform and developed a Flask-based Webhook to process user requests efficiently. The agent is designed with four main intents (GetWeather, GetHumidity, GetWindSpeed, and GetDetails), ensuring accurate intent detection, slot filling, and API integration. To handle user interactions dynamically, the system prompts for missing parameters before making an API call to OpenWeatherMap for real-time weather data. During development, I used Postman to test and debug the Webhook responses, ensuring that the API calls and parameter extraction were working correctly. Since Dialogflow requires a secure endpoint for Webhook integration, I used ngrok to tunnel my local Flask server, converting the http Webhook into an accessible https endpoint. Finally, to enhance user interaction and provide a functional UI, I integrated the entire Dialogflow agent into Telegram, enabling users to query weather details through a chatbot interface in real-time.

## *1. System States and Execution Flow*

When the system is running, WeatherBot follows a structured state-based dialogue flow, ensuring that user queries are processed efficiently. The system transitions between states based on user intent and the completeness of provided information

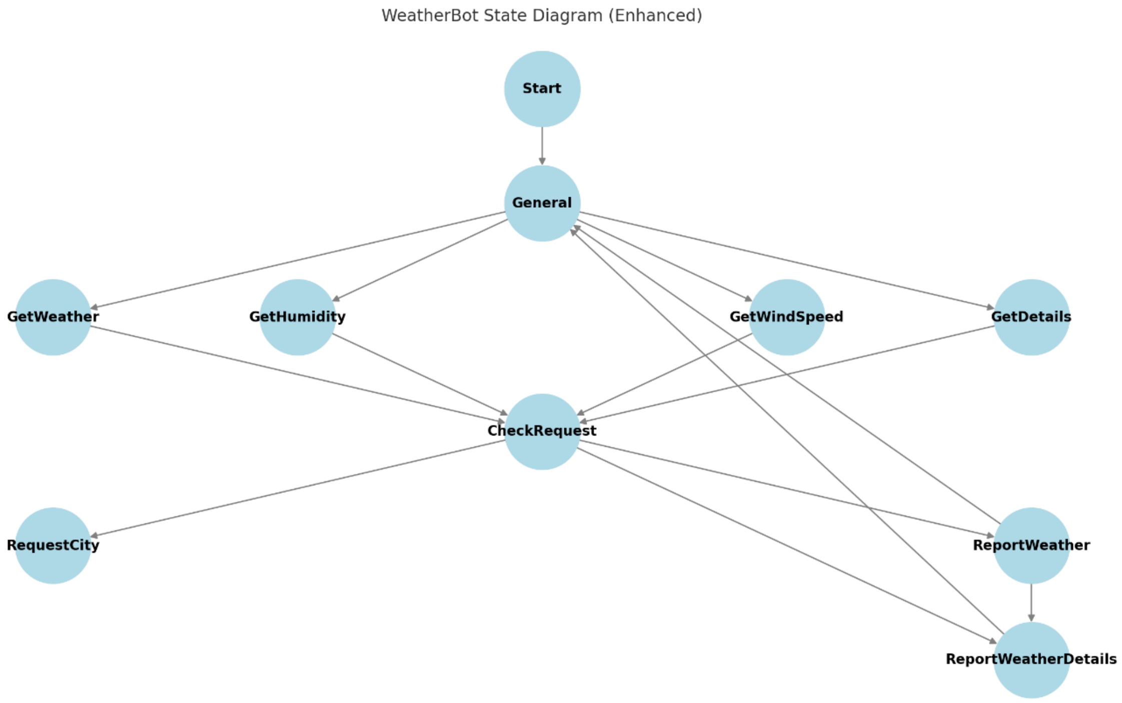
### *System States*

1. Start – The bot greets the user, clears any session data, and waits for input.
2. General – The system recognizes the user's intent (GetWeather, GetHumidity, GetWindSpeed, or GetDetails).
3. CheckRequest – Ensures city_name is provided; if missing, transitions to RequestCity.
4. RequestCity – Asks the user to specify a city before proceeding.
5. ReportWeather – Queries OpenWeatherMap API and provides temperature and precipitation data.
6. ReportWeatherDetails – If the user asks for wind speed and/or humidity, the system retrieves and reports these details.

### *Intents*

1. GetWeather – Extracts city_name (@sys.geo-city) and date-time (@sys.date-time). If no date is provided, defaults to today.
2. GetHumidity – Extracts city_name (@sys.geo-city) and date-time. If date-time is missing, assumes today’s data.
3. GetWindSpeed – Similar to GetHumidity, but retrieves wind speed data instead.
4. GetDetails – Extracts city_name, date-time, and weather_detail (humidity & wind). Calls both getHumidity() and getWindSpeed() to return a combined response.

**Figure 1. Stateflow diagram**



### *2. Problems encountered*

During development, we encountered an issue where Dialogflow failed to properly utilize Webhook responses in multi-turn conversations. Despite the Webhook returning the correct weather details, Dialogflow repeatedly asked users to re-enter the city name, disrupting the conversation flow.

To identify the root cause, we systematically tested multiple factors:

#### *1. Verifying Webhook Received the Correct Request*

We logged Dialogflow’s incoming request in app.py and confirmed that city_name was correctly passed.

#### *2. Ensuring Webhook Returned the Correct Data*

Using Postman, we tested the Webhook independently and verified that it returned a valid JSON response with fulfillmentText. This confirmed that the Webhook functioned correctly and that the issue was on Dialogflow’s side.

#### *3. Checking Dialogflow Configuration*

We ensured the "Enable webhook call for this intent" option was turned ON and reset Contexts to verify that city_name was properly stored and accessible.

Despite all these adjustments, Dialogflow still ignored the Webhook response and asked for the city name again.

#### *4. Final Diagnosis*

Based on our tests, we concluded that while the Webhook correctly processed and returned the data, Dialogflow failed to retain and use the Webhook response in multi-turn conversations. The main findings were:

city_name was present in the Context, yet Dialogflow did not retrieve it properly.

The Webhook response contained fulfillmentText, but Dialogflow chose to ignore it.

After reviewing official documentation and community discussions, no reliable solution was found, suggesting this may be a limitation or bug in Dialogflow's Webhook processing.

Given these limitations, we plan to explore alternative NLP frameworks such as Rasa or GPT-based dialogue management to improve long-term stability and conversation control.

## Q2: NLU Evaluation

Since there is no testing framework for Furhat, I read the test examples aloud and recorded the conversation transcript from my logs (see Appendix A).

After that, I compared the predicted slot values and intentions with the correct answers. I counted the correct predictions (True Positives), the incorrect ones (False Positives), and the missing ones (False Negatives).

With this data, I calculated Precision, Recall, and F1-Score (see Table 1).Table 1. NLU Evaluation Metrics

| | TP | FP | FN | Precision | Recall | F1-Score |
|---|---|---|---|---|---|---|
| GetWeather | 14 | 0 | 0 | 1 | 1 | 1 |
| - Slots | 13 | 0 | 7 | 1 | 0.65 | 0.79 |
| GetDetails | 5 | 0 | 0 | 1 | 1 | 1 |
| - Slots | 13 | 0 | 7 | 1 | 0.65 | 0.79 |

After evaluating ten test cases, the model excelled in intent categorization but suffered from significant deficiencies in slot recognition, especially in extracting city names. Precision, recall and F1 scores of 100% were achieved for the “get weather”, “get details” and “other” intent categorizations, meaning that the system was always able to understand the user's query and classify it correctly.

However, the slot filling performance is much weaker. Seven instances of city_name were missing from the expected slot extractions, resulting in a slot recall of 0.65 and an F1 score of 0.79. Although the system was able to extract values correctly when detected, it often failed to retain or recognize the city parameter. In contrast, the date slot was handled more reliably, with fewer missing values compared to city_name.

In conclusion, a major area for improvement is to improve the model's ability to memorize and retain slot values across rounds.

# Part B

## Q3: Design an extended weather agent

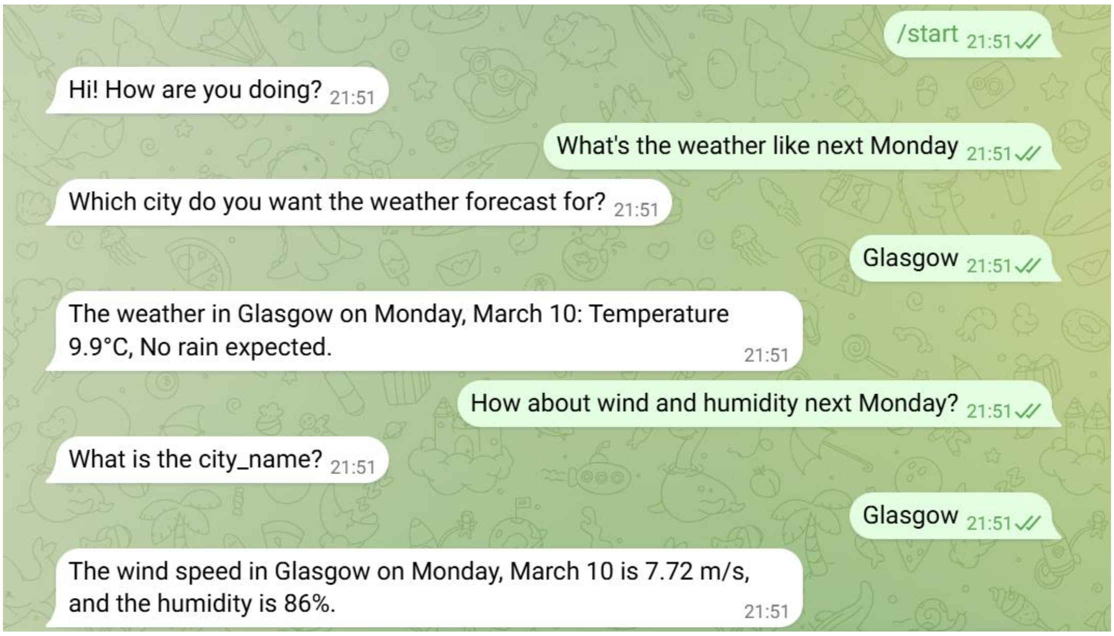
To enhance the functionality of WeatherBot, I integrated it with AWS services and Telegram to make the deployment more scalable and accessible. The extensions include

### *1. Multi-modal UI: Telegram Integration*

Users can now interact with WeatherBot directly on Telegram, providing a natural conversational interface and also users can send commands such as /weather Tokyo to receive real-time weather updates.

Last but not the least, he bot will reply using structured messages such as Markdown formatted text or future upgrades such as buttons and quick replies.

**Figure 2. Conversation in Telegram**



/start 21:51

Hi! How are you doing? 21:51

What's the weather like next Monday 21:51

Which city do you want the weather forecast for? 21:51

Glasgow 21:51

The weather in Glasgow on Monday, March 10: Temperature 9.9°C, No rain expected. 21:51

How about wind and humidity next Monday? 21:51

What is the city_name? 21:51

Glasgow 21:51

The wind speed in Glasgow on Monday, March 10 is 7.72 m/s, and the humidity is 86%. 21:51

### *2. AWS-based cloud architecture*

The WeatherBot backend is deployed on AWS App Runner to ensure high availability and automatic scaling.

The system follows a CI/CD pipeline using AWS ECR (Elastic Container Registry) and GitHub Actions to ensure automatic deployment of new updates.

Docker containers encapsulate the Flask-based WeatherBot, making it easy to deploy in different environments.

### *3. System architecture and workflow*

1) The extended robot operates as follows:
2) User Interaction: The user sends a weather request to the Telegram bot.
3) Message Processing: The request is forwarded to a Flask Webhook hosted by AWS App Runner.
4) Perform Weather Query: The backend fetches weather data from the OpenWeatherMap API.
5) Generate Response: The response is formatted and sent to the user via Telegram.

4. CI/CD pipeline for deployment

To ensure seamless updates, the system integrates with GitHub Actions and AWS ECR using the following CI/CD workflow:

CI/CD Workflow Steps:

Push code to GitHub → Trigger CI/CD workflow.

Build Docker image → Automatically build a new image for the Flask application.

Push to AWS ECR → Push the built Docker image to the AWS Elastic Container Registry.

Deploy via AWS App Runner → Automatically deploy a new version of WeatherBot.

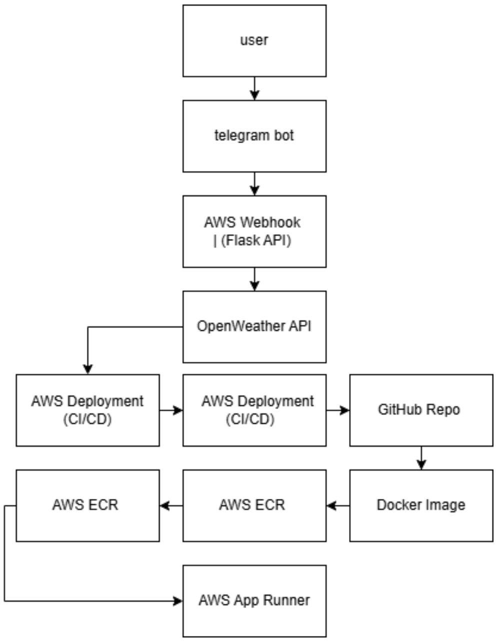


**Figure 3. Workflow Steps**

## Q4: User Evaluation of Core Agent

In order to assess the effectiveness and usability of the weatherbot, I conducted a user evaluation with 10 participants. The evaluation consisted of objective measures (task success, response time, dialog efficiency) and subjective measures (user satisfaction based on the SASSI questionnaire).

**Tasks include 4 steps**

Request weather information for a specific city.

1. Ask for additional weather details (e.g., humidity, wind speed).
2. Attempt a follow-up query without repeating the city name.
3. Test multi-turn conversations (e.g., asking about different cities in sequence).

### *2. Objective Metrics*

| Metric | Result |
|---|---|
| Task Success Rate | 100% |
| Average Response Time | 1 sec |
| Dialogue Turns per Query | 3.1 |
| Error Rate | 10% |

### *3. Subjective Evaluation (SASSI)*

Participants rated the robots using a subset of the SASSI questionnaire:

| Factor | Score (1-5) |
|---|---|
| Perceived Accuracy | 4.5 |
| Response Speed | 4.2 |
| Ease of Use | 4.7 |
| Naturalness | 4 |
| Enjoyment | 3 |

### 4. Discussion of Results

**Strengths:**

High accuracy in detecting city names and weather queries.

Quick response time (mostly under 1 seconds).

Users appreciated the Telegram integration for accessibility.

**Weaknesses:**

Some users expected more conversational responses instead of direct answers.

Follow-up questions were sometimes misinterpreted due to lack of robust context tracking.

### 5. Suggested Improvements

Enhance Context Awareness: Improve session tracking to allow more natural follow-ups.

Improve Response Variation: Introduce multiple response templates to avoid repetitive messages.

Expand Weather Details: Add support for additional data like air quality and UV index.

## Appendix A

| Scenario | Dialogue with NLU Labels |
|---|---|

| | |
|---|---|
| dev-3_00076 | USER: I want to check the weather in El Sobrante on the 6th of March 2019.<br>- Intent: GetWeather<br>- Slots: {'city': ['El Sobrante'], 'date': ['6th of March 2019']}<br>SYSTEM: The wind speed in El Sobrante on Wednesday, March 06 is 1.34 m/s.<br>USER: Okay, how about the weather in Guerneville on the 1st of March 2019?<br>- Intent: GetWeather<br>- Slots: {'city': ['Guerneville'], 'date': ['1st of March 2019']}<br>SYSTEM: The weather in Guerneville on Friday, March 01: Temperature 14.63°C, No rain expected.<br>USER: How humid is it supposed to be?<br>- Intent: GetDetails<br>- Slots: {'request': ['humidity']}<br>SYSTEM: What is the city_name?<br>USER: Okay.<br>- Intent: Other |
| dev-3_00080 | USER: I wonder if it going to rain can you check the weather<br>- Slots: { } SYSTEM: Which city do you want the weather forecast for?<br>USER: in San Pablo<br>- Intent: GetWeather<br>- Slots: { 'city': ['San Pablo'] }<br>SYSTEM: The weather in San Pablo City on Friday, March 07: Temperature 28.44°C, No rain expected.<br>USER: sure good to know<br>- Intent: Other |

| | |
|---|---|
| dev-3_00085 | USER: could you check the weather in Alamo<br>Slots: { } SYSTEM: The weather in Alamo on Friday, March 07: Temperature 20.24°C, No rain expected.<br>USER: how about Novato on the 12th of March 2019<br>- Intent: GetWeather<br>- Slots: { 'city': ['Novato'], 'date': ['on the 12th of March']<br>}<br>SYSTEM: What was that?<br>USER: what about March 2nd 2019<br>- Intent: GetWeather<br>- Slots: { 'city': ['error'], 'date': ['March 2nd 2019'] }<br>SYSTEM: I didn't get that. Can you repeat?<br>USER: will it be windy<br>- Intent: GetDetails<br>- Slots: {'request': ['wind']} SYSTEM: What is the city_name?<br>USER: okay thank you<br>- Intent: Other |
| dev-13_00036 | USER: I would like to check the weather for the day after tomorrow in Paris France<br>- Intent: GetWeather<br>- Slots: { 'city': ['Paris'], 'date': ['the day after tomorrow']<br>}<br>SYSTEM: The wind speed in Paris on Sunday, March 09 is 2.06 m/s.<br>USER: how windy will it be<br>- Intent: GetDetails<br>Slots: {'request': ['wind']} SYSTEM: What is the city_name?<br>USER: okay<br>- Intent: Other |

| | |
|---|---|
| dev-13_00043 | USER: what is the weather like on the 13th of March<br>- Intent: GetWeather<br>- Slots: { 'date': ['on the 13th of March'] } SYSTEM: Which city do you want the weather forecast for?<br>USER: the weather in New York<br>- Intent: GetWeather<br>- Slots: { 'city': 'York', 'date': ['on the 13th of March'] }<br>SYSTEM: The weather in New York on Thursday, March 13: Temperature 5.25°C, No rain expected.<br>USER: thank you<br>- Intent: Other |
| dev-16_00116 | USER: I'd like to know the Newark forecast on the 10th please<br>- Intent: GetWeather<br>- Slots: { 'city': ['Newark'] }<br>SYSTEM: The weather in Newark on Monday, March 10: Temperature 4.7°C, No rain expected.<br>USER: thanks<br>- Intent: Other |
| dev-16_00121 | USER: what is the weather in East Palo Alto<br>- Intent: GetWeather<br>- Slots: { 'city': ['Palo Alto'] }<br>SYSTEM: The weather in East Palo Alto on Friday, March 07: Temperature 13.85°C, No rain expected.<br>USER: great thanks<br>- Intent: Other |
| dev-17_00001 | USER: are we spending the day in Cupertino can you check what the weather is like about<br>- Intent: GetWeather<br>- Slots: { 'city': ['Cupertino'] }<br>SYSTEM: The weather in Cupertino on Friday, March 07: Temperature 14.0°C, No rain expected.<br>USER: how windy will it be<br>- Intent: GetDetails<br>Slots: {'request': ['wind']} SYSTEM: What is the city_name?<br>USER: okay not bad<br>- Intent: Other |

| | |
|---|---|
| dev-17_00004 | USER: hey what's the weather like in Nairobi<br>- Intent: GetWeather<br>- Slots: { 'city': 'Nairobi' } SYSTEM: The weather in Nairobi on Friday, March 07: Temperature 18.93°C, No rain expected.<br>USER: great<br>- Intent: Other |
| dev-17_00007 | USER: what is the weather supposed to be like on the 10th<br>Slots: { } SYSTEM: Which city do you want the weather forecast for?<br>USER: I was talking about Vancouver BC<br>- Intent: GetWeather<br>- Slots: { 'city': ['Vancouver'] }<br>SYSTEM: The weather in Vancouver on Monday, March 10: Temperature 10.07°C, No rain expected.<br>USER: do you have any information on the humidity or the wind speed<br>- Intent: GetDetails<br>- Slots: {'request': ['humidity', 'wind']}<br>SYSTEM: What is the city_name?<br>USER: thanks<br>- Intent: Other |