Implementation of The Features - Phase 2

Deadline: November 17

Objective: You will upgrade the project app with two more features for this milestone. We request you implement the following features for this milestone:

- Showing weather information of a selected city to the user
- Showing LLM-suggested weather-related questions, allowing users to receive personalized weather insights directly from the LLM
- Showing the interactive map of a selected city to the user

Deliverables: To implement the requested features, please follow the tasks explained under the "Tasks" section. Your deliverables are: (1) the code pushed in the team's GitHub repository and (2) a video (no more than five minutes) that captures the following actions:

1. Start the app, where it shows the login page
2. Sign in and show the list of the following cities: Champaign, Chicago, New York, Los Angeles, and San Fransisco.
3. Select an arbitrary city from the list and click the "Weather" button
4. Go to a new screen and show the weather information of the selected city
5. Click the "Weather Insights" button, show the LLM-suggested weather-related questions on the weather insights screen, click on the suggested questions, and show LLM-generated answers to the questions.
6. Go back to the main screen with the five cities listed
7. Choose another city from the list (different from the one selected in Step 3) and click on the "Weather" button
8. Show the weather information of the selected city
9. Go back to the main screen with the five cities listed
10. Select an arbitrary city from the list and click the "Map" button
11. Go to a new screen and show the map of the selected city
12. Zoom in and zoom out on the map several times to show the map is interactive (not an image)

Grading rubric: Please find the rubric below.

Activity	Points	Team/Individual
Implementation of two buttons for weather and map (for each city or the entire list, or other implementations following the same logic)	10 (5 each)	team
Implementation of weather screen (with requested details in Task 1)	10	team
Implementation of weather feature (ability to collect up-to-date weather information)	30	team

Implementation of the LLM-based “Weather Insights” feature (show LLM-generated weather-related questions and answers)	20	team
Implementation of map screen (with requested details in Task 2)	10	team
Implementation of the map feature	30	team
Up-to-date weekly report	10	team
Code documentation	20	team*
Peer evaluation	10	individual**

* You will lose **“2 points” out of 20 for every callback/method that you forgot to add a comment for**. The points you get from the documentation range from 0 to 20.

** Note that those team members who fail to submit the peer evaluation form **will lose 10 points compared to their teammates**. We accept no peer evaluation form submissions after the deadline. We accept no request for regrading if you fail to properly submit the form (forgetting to submit the form, using the wrong team number, and using a non-Illinois email address).

*** Students reported by their peers not helping the team to progress on this milestone **will lose “at least” 50% of the total grade**.

Note: Your TA will run the app on their local machine under different scenarios (different cities). Avoid any hard coding, or you will lose all the points for this milestone. Please push the API keys as well since it makes it easier for the TAs to run the application.

Tasks:

<<Main page modification>>
We expect you to navigate to the weather and map screens from the main activity, i.e., the screen that shows the list of cities. There are several ways to do that, and you have the freedom to implement it the way you desire. The simplest way is to add two buttons from each city, one of them named “Weather” and the other “Map.” An alternative approach is considering check boxes for each city and two buttons, “Weather” and “Map,” for the entire list. Feel free to implement the feature differently if your implementation follows the same logic.

<<Task 1: Collecting and displaying up-to-date weather information>>
We expect that by clicking on the “Weather” button in the main activity for a selected city, the app redirects the user to a new screen, where they can see the following information:

- Name of the city

- Date and time
- Relevant weather data
    - Temperature
    - Weather (sunny, partly sunny, cloudy, partly cloudy, foggy, etc.)
    - Humidity
    - Wind condition
- A “Weather Insights” Button (See Task 2 for details)

To get the weather information, the best way is to use APIs offered by weather servers. Such servers, including AccuWeather, WeatherBit, and OpenWeatherMap, provide users with free API keys (unlimited, for a trial period, or a specific number of API calls). Using them, you can connect to those servers and fetch the current weather information for a specific location. The weather information fetched from these servers is usually in the form of JSON, which you need to parse and get the information we have asked you to show on the weather screen.

<<Task 2: Weather Insights: Generating weather-related questions & answers using LLM>>

To enhance the weather page, you should implement a "Weather Insights" button at the bottom, positioned below all the weather details like Temperature, Weather, Humidity, and Wind Conditions in the weather information page. When a user clicks this button, a new screen should appear, offering at least two context-relevant questions generated by the LLM based on the weather data retrieved from the API.

For instance, if the weather data indicates cool temperatures and potential rain, the LLM might suggest questions like "What should I wear today?" and "What should I prepare for an outdoor event today?" These questions should be presented as buttons to the user. When the user clicks one of these buttons, the app should prompt the LLM again with the weather data and the question, and show the LLM-generated response on the screen.

Please note that the questions presented in the interface and the response to the questions must be dynamically generated by the LLM using LLM APIs. TAs will check the source code to identify any hard-coded questions and answers. If they find any hard-coded questions or answers in the source code, you may lose all your points in this task.

For example, you can prompt the LLM to generate weather-related questions using the following prompt template:

“Today’s weather is… Please generate a context-specific question based on the given weather data that users might ask to help them make decisions about their day…”

You can try different ways (e.g., adding questions examples, adding hints, limiting scope) to prompt the LLM to generate questions that make sense.

To create the chat interface within the app, you are required to use Gemini LLM. Gemini LLM is an advanced language model designed for text generation. It offers a free tier with usage limits, allowing experimentation with the model without cost. For more information and to explore its capabilities, you can visit https://ai.google.dev/gemini-api/docs. For pricing information, https://ai.google.dev/pricing. To use Gemini API in the Android app, you can follow the tutorial at https://ai.google.dev/gemini-api/docs/quickstart?lang=android.

You may also explore other tutorials or resources available for integrating an LLM into your Android app, in addition to the official guides. Alternatively, if you find it more convenient, you can consider using Python to host an endpoint for the LLM and connect it to your app. However, please be aware that our instruction team may not be able to provide support if you choose the Python-based approach.

<<Task 3: Showing the interactive map>>
We expect that by clicking on the "Map" button in the main activity for a selected city, the app redirects the user to a new screen, where they can see the following information:

- Name of the city
- Latitude and Longitude of the city
- Interactive map of the city

You can show the map of a location using different ways. One possible way is to use Google Map SDK (https://developers.google.com/maps/documentation/android-sdk/overview). However, this option requires you to create an account on Google Cloud, get an API key, and use it in your app (Google Cloud offers a $0.00 charge trial that expires at either the end of 90 days or after the account has accrued $300 worth of charges, whichever comes first. The trial is enough for the project's lifetime, and you can cancel anytime). Another alternative is to embed Google Maps (or other map providers such as MapQuest) for the selected city in a WebView widget (note that this should be inside the map and not open a window to the browser): https://developer.android.com/reference/android/webkit/WebView. To avoid hardcoding using the latter option, you should generate the embedding URL based on the latitude and longitude of the city (Hint: the embedding URL can be generated as 'https://maps.google.com/maps?q=' + latitude + ',' + longitude + '&t=&z=15&ie=UTF8&iwloc=&output=embed').

You have the freedom to choose other approaches. However, please note that this should be part of your applications and not opening another app to show the map. Also, you should implement an interactive map, not a static image captured from Map providers.

<<Peer Evaluation>>
Each team member should submit the following form for the peer evaluation **for Milestone 4** (we accept no peer review for this milestone after the milestone deadline):
https://forms.gle/CeDwa6Pj2Pbg9PUw6