Course Project

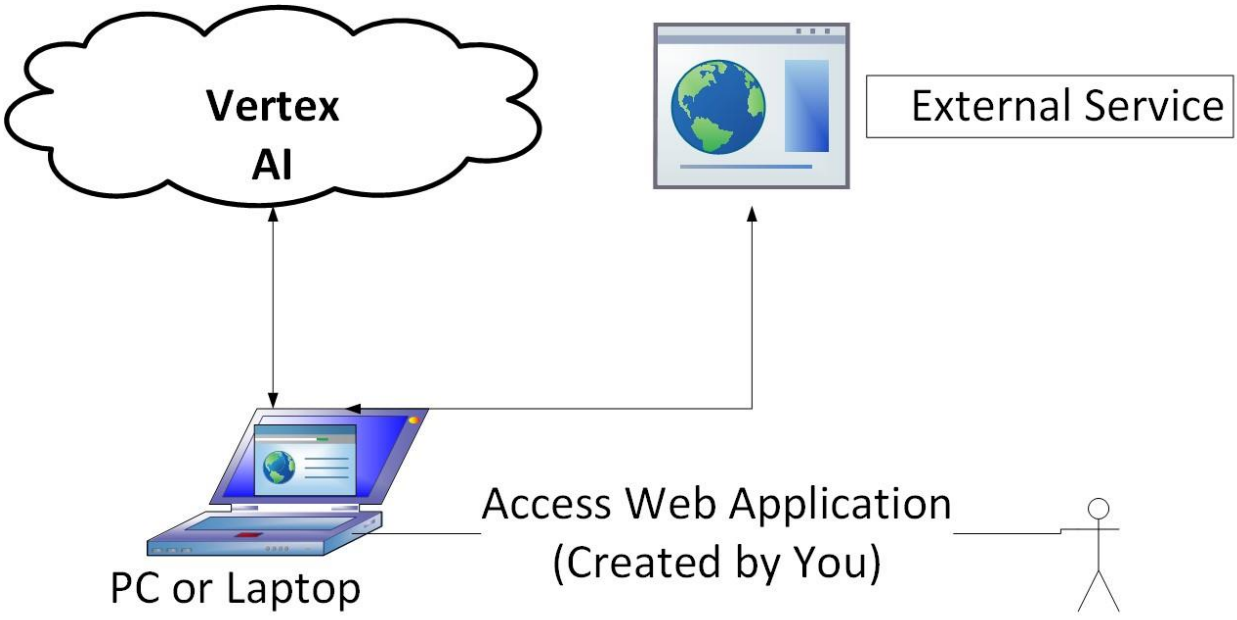
Deadline: February 29th, 11:59PM ET (8:59PM PT)

Accept this project by accessing GitHub classroom via the following URL:
https://classroom.github.com/a/FF1Kikho

You are required to submit your GitHub Repository URL on Canvas. A penalty will be applied if you don't do so. If you are working in a team, one submission is sufficient.

Your ReadMe file should include the exact steps that can be used to reproduce a functional version of your solution.

You may choose one peer to work with you on the project. You MUST list your peer's name on the ReadMe file on your GitHub Repository. Failure to list your peer's name will lead to no grade for your peer.



General Description:

You are requested to build a GenAI agent that will offer an updated content to users without having to retrain the LLM/Diffusion model. Your solution will include the following components:

- Web Application, deployed on your laptop, to interface with the users (accepts user input and displays output)
- GenAI Agent, deployed on your laptop and interfacing with Vertex AI, to conduct the following activities:
    - Accepts user input.
    - Identifies areas of user inquiry where updated content is needed.
    - Communicates with the LLM/Diffusion Model to generate a standard template answer without the updated data.
    - Communicates with the external service to receive the most updated version of the data.
    - Updates the LLM/Diffusion model template answer with the updated data.
    - Sends the response back to the user.
    - For Use Case Scenarios 1 and 2, your agent should use at least 2 different models from Vertex AI.

In addition, your solution will address **one** of the following use cases:

- Provide GenAI services in English and another language of your choice – Difficulty level: Easy
- Provide up-to-date weather information (and forecasts) – Difficulty level: Moderate
- Create new objects and add them to your Metaverse – Difficulty level: Challenging

The end goal of this solution is to provide your user with context-aware solution that uses external live services to provide accurate up-to-date answers to their users.

Use Case#1: Provide GenAI services in another language of your choice

Difficulty level: Easy

LLMs mostly understand English only and are trained to provide extensive knowledge in English. However, some users don't speak English. So, your solution aims to democratize the access to LLM features for non-English speakers.

Notes:

- Your web application needs to provide the user with the ability to choose the language to inquire with. Your options should include English in addition to any other language of your choice.
- You may need to leverage a public reliable translation service that would accept the user input and send you proper/understandable output.
- Test the cohesiveness and understandability of the output generated by the translation service before sending it to the LLM.
- You will need to translate the input to the LLM to English and translate the LLM output from English to a language of your choice.
- Use LLMs that don't support the language of your choice by default.
- Your agent should use at least 2 different models from Vertex AI, and you will need to comment on their accuracy in your ReadMe file.

Use Case#2: Provide up-to-date weather information (and forecasts)

Difficulty level: Moderate

LLMs suffer from being bound to their training date. In this solution, you will build an application that is able to find the most updated weather information relevant to user inquiry and combine them with a template answer provided by LLM to provide a comprehensive, cohesive response.

Notes:

- Find a public weather service that offers several information regarding the weather.
- On your web application front page, list the various weather updates you can provide.
- Leverage prompt engineering to build enough context and ask the LLM to provide an answer with a placeholder for the dynamic data generated by the external service.
- Your agent should use at least 2 different models from Vertex AI, and you will need to comment on their accuracy in your ReadMe file.

Use Case#3: Create new objects and add them to your Metaverse

Difficulty level: Challenging

You will build a solution that aims to use Generative AI to extend the Metaverse.

Notes:

- A sample demo is provided on Canvas home page.
- You may use an open source Metaverse named Vircadia but feel free to use other metaverse platforms as long as they are open source.
    - Vircadia's GitHub repository: https://github.com/vircadia/
- In your solution, the web application will be used to accept user prompt and the output will be reflected on the Metaverse server with a re-rendered screen.
- You may find the following resources helpful:
    - https://cloud.google.com/vertex-ai/docs/generative-ai/multimodal/send-multimodal-prompts
    - https://cloud.google.com/vertex-ai/docs/generative-ai/image/overview (if you will use imagen, you will need to submit a request to get access)
    - https://lablab.ai/t/imagen-vertexai-tutorial

Additonal Notes:

- You may use any programming language for the web application.
- Keep any private keys for your GCP account outside of the code on GitHub. Instead, submit them on Canvas along with your repository URL.
- Submit a video to show a demo of a functional app and a code walkthrough. More details are provided in the **Submission Guidelines** section.

Submission Guidelines:

- Post URL for your GitHub repository to Canvas along with your Peer's name.
- Your GitHub repository should have a ReadMe.md file that lists the "exact" steps on how to get your solution to work.

    - Make sure you list any installations, configurations, or environment-specific settings in your ReadMe file.

- You should record a video demonstrating two elements:
    - Code Walkthrough: Explain your code changes, making sure to discuss how your implementation addresses at least one of the following challenges: hallucinations, Standardized Output Consistency, response latency, API reliability and availability, or data privacy.
    - Application Demo: Demoing the running application while you are navigating through EVERY functionality that is working in your application. Make sure to discuss how your current UI design enhances the functionality of the LLM in achieving your product's objectives. Also, discuss potential features you might add to leverage user feedback for improving the LLM model, particularly if you had ownership of the model.
    - Note: "The demonstration video is crucial for grading. Any functionality not demonstrated in the video may result in a deduction of points."
- Your video size may be large to be uploaded to GitHub. You may use Box to upload the video and add the URL to your ReadMe.md file in your GitHub repository.
    - Make sure that your video is publicly shared. Private videos won't be visible to the instructor and TAs and therefore, your project grade will be impacted.

Grading Criteria:

- Getting the web application up and running to accept input and display output: **25% of the total project grade.**
- Choosing the LLM/Diffusion Model, customizing it, and interfacing with it from the web app: **40% of the total project grade**
- Communicating with the external service (e.g., weather service, translation layer or metaverse platform) successfully: **15% of the total project grade**

- Integrating the LLM/Diffusion Model output with the external service output: **20% of the total project grade**

Grading Notes:

- Unlike HW-assignments that has late penalty, late project submissions on Canvas or GitHub will receive 0 points (won't be graded).
- **Not submitting the GitHub video (for both code walkthrough and functionality demo): you will get up to 80% of the maximum grade.**
- Not providing clear details in the ReadMe file on how to run the application (or any variables that need to be updated/replaced): **you will get up to 90% of the maximum grade.**