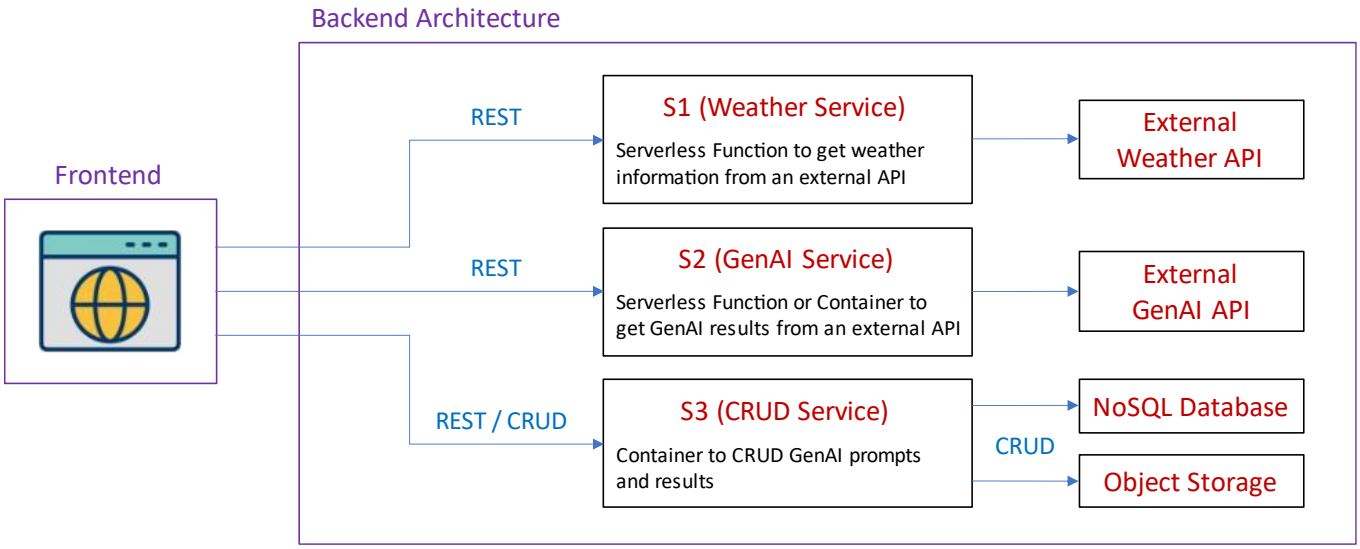
College of Computing and Mathematics @ KFUPM
Computer Engineering Department
COE 558: Cloud and Edge Computing, Term 242, Section 2

Course Project

Due on Monday May 19th at 11:59 PM

Project Description

Start by developing a web application that consists of a frontend and a backend as shown below:



Backend Architecture

The backend should contain the following:

1. **S1 (Weather Service)**

   RESTful API, implemented as a serverless function, that gets the current weather condition from an external API. Use any weather API on the Internet. Here are few examples:

   - WeatherAPI
   - WeatherStack
   - OpenWeather
   - Xweather

2. **S2 (GenAI Service)**

   RESTful API, implemented as a serverless function or a container, that utilizes an external Generative AI (GenAI) API to process or produce media (images, audio, or video). Here are few examples:

   - Picsart
   - Segmind
   - Gemini API
   - Adobe Firefly
   - Hugging Face
   - Eden AI

3. **S3 (CRUD Service)**

   CRUD RESTful API, implemented as a container to keep and manage the history of S2 (GenAI Service) prompts and the produced results. CRUD operations should be performed on a NoSQL database and object storage.

4. **NoSQL Database**

   NoSQL document database to store the prompts along with their results. For media files (images, audio, or video), you can upload them to an object storage service and just keep a reference (e.g. object reference or a URL) in the NoSQL database documents.

5. **Object Storage**

   Object storage should be used to store media binary input or output files (images, audio, or video).

Frontend

- The frontend can be one or few web page(s).
- It should ideally be hosted on a container.
- It should get the weather temperature and condition from S1 (Weather Service) and display them on the page.
- It should facilitate sending a prompt to S2 (GenAI Service) and display the result(s).
- It should allow saving the results from S2 (GenAI Service) using S3 (CRUD Service).
- It should facilitate listing, displaying, and deleting saved results using S3 (CRUD Service).
- The frontend should be user-friendly.

API Specifications

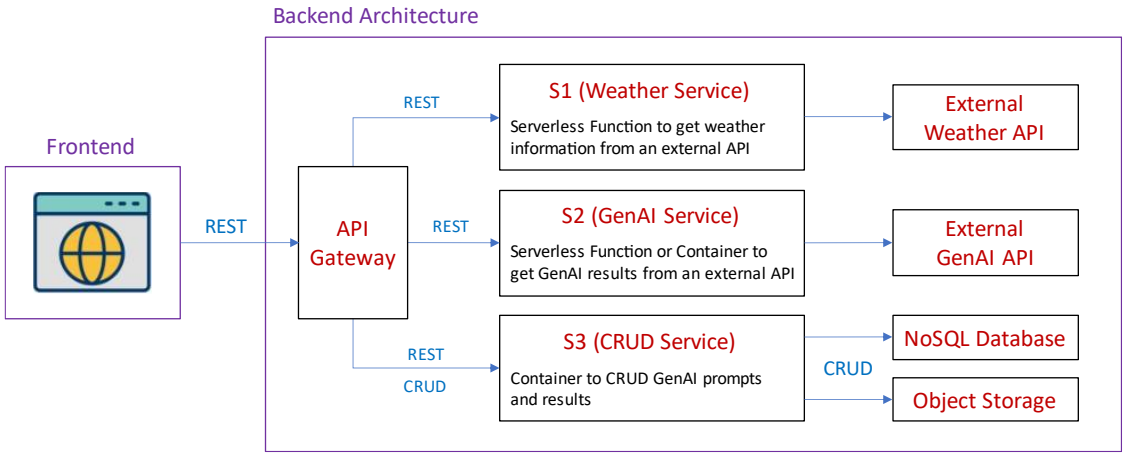
- Develop the OpenAPI specification for the APIs you developed (S1, S2, and S3).
- Generate Swagger UI to visualize the APIs your developed and allow interacting with them.

The Weather Service

- Show a symbol or an image along with a text label for the weather condition.
- Allow the user to see the weather temperature and switch between Celsius and Fahrenheit.
- Get the weather information for the current user location by their IP or from the browser (the user will be asked to give your web application permission to know their location).
- Allow the user to change the location by searching for a city or selecting a specific location on a map.

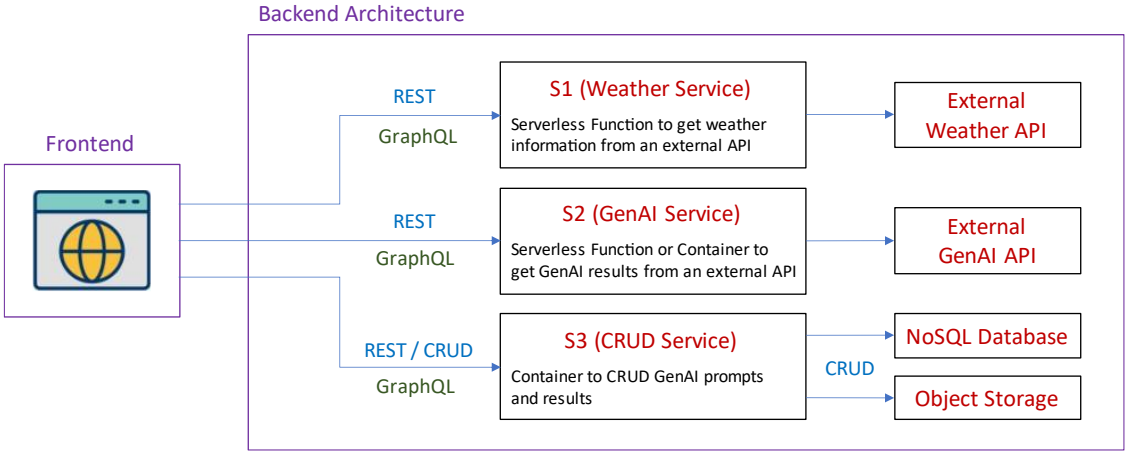
API Gateway

- Deploy an API Gateway in front of the APIs you developed, publish your APIs on the gateway, and then modify the frontend to use these published APIs.



GraphQL APIs (extra credit)

- Develop a GraphQL version of the APIs you developed and modify the frontend to allow using RESTful or GraphQL APIs.



Guidelines

- Form a **team** of up to 4 students. Please send me the team you formed by April 10th.
- You can use **any Cloud Service Provider(s)** to host your web application. **Google Cloud** is recommended since you have access to its Skills Boost learning platform.
- You should be able to utilize the **free trials or tiers** for the Cloud Service Provider(s) and external APIs.
- Follow the **best practices** discussed in class **when designing your APIs**.
- Reach out to me via email, Teams, or WhatsApp if you get stuck to seek help.
    - To get a faster response and have efficient communication, send me a screenshot or a video recording explaining the issue you are facing.
    - We can schedule a help session on Teams when necessary.

Submission

- Make sure you **submit the project on Blackboard by May 19th** at 11:59 PM.
- For each team, **one student should submit** the project on behalf of other team members.
- You should **submit all artifacts** developed for the project in one ZIP file.
- For each API you developed, create a **Postman collection** with saved sample responses.
    - Label the collections, requests, and sample responses properly.
    - Export the collections and include them in your submission.
- **Record a demo** and include the recording (or a link to it) in the submission. It can be on YouTube, Vimeo, or any video or file sharing service. Make sure the link is publicly accessible without authentication.
- **Create a report** for your project that contains at least:
    - **Team members** with a table showing who worked on what part(s) of the project.
    - A **diagram** for what you developed showing the different layers and components labeled with the products or services you used and on which cloud provider(s) or using what external API providers.
    - Links to the **external APIs** you used (for weather and GenAI, or others).
    - **Full URLs** (links) for the **frontend** and each service in the **backend**.
    - **Full URL** (link) or filename of the **demo recording**.
    - **Documentation of the APIs** you developed. For each endpoint, you should include:
        - Brief description.
        - HTTP Verb (method).
        - Endpoint path and the format of the query string, if any.
        - Format of the body, if any.
        - Format of successful and all implemented failed responses.
        - Sample request and response.
    - **Screenshots** of the **Cloud resources** used for the project.
    - **Screenshots** of the **frontend** open on a browser.
- If you have implemented extra credits or additional functionality, please indicate it clearly in your report in a separate section with screenshots.
- **Keep your cloud resources up and running** until you see your course letter grade.
- Grant me **read access** to your cloud project(s) if some components are partially implemented or having issues for me to check your progress and give you partial credits.