Final Project: Cloud Integration Mini-Capstone

Overview

In this final assignment, you will design a small, realistic healthcare cloud solution that **integrates multiple services** you have used this semester (e.g., storage, compute, databases, AI/analytics, containers/serverless).

You have **two levels** of expectations/options to complete this assignment:

1. **Part 1 Required: Design:**
   Produce a clear **solution architecture and implementation plan** that shows *how* you would use cloud services together to solve a specific healthcare problem.

2. **Part 2 Optional (Part 1 + Prototype for Extra Credit):**
   Try your best to implement a **small working prototype** (in Python/Flask) that demonstrates a subset of your design (e.g., a simple web endpoint that reads/writes to cloud storage or a database). Please use AI/LLMs to assist. Would recommend creating a simple flask based application so that way you can code everything in Python.

Learning Objectives

At the end of this project, you will be able to:

1. Propose a **healthcare-relevant use case** that benefits from cloud services.
2. Design an **end-to-end architecture** that integrates multiple services (storage, compute, databases, and optionally AI/analytics).
3. Map architecture components to **specific Azure/GCP/OCI services** and to earlier course assignments.
4. (Optional) Build a small **Python/Flask prototype** that interacts with at least one cloud resource.
5. Reflect on **trade-offs** (cost, complexity, governance) in your design.

Required: Design-Only Track

At minimum, you must complete **all** of the following.

1. Use Case Description (≈ 1 pages)

Define a concrete healthcare scenario, for example:

- A simple patient intake and triage app for a clinic.
- A claims or encounter dashboard that loads data into storage and runs basic summaries.
- A remote monitoring mini-pipeline (e.g., wearable/device data into storage, then summarized for a clinician).
- A teaching/education analytics scenario (e.g., logging student interactions and generating simple reports).

Your description must include:

- **Problem statement:**
  What problem are you solving? Who are the users (e.g., clinicians, staff, patients, students)?
- **Data sources:**
  What data you would store (e.g., CSV files with encounters, JSON from APIs, sensor readings), and where it roughly comes from.
- **Basic workflow:**
  A step-by-step description (1 → 2 → 3…) of how data flows through your system.

Deliverable file:

- `use_case.md`

---

2. Architecture Diagram

Create a **high-level architecture diagram** showing how your components fit together.

Your diagram must show:

- **Frontend / access layer**
  - E.g., a Flask app, an API endpoint, or a scheduled job (if no GUI).
- **Compute layer**
  - At least one of:
    - VM or Compute Engine / Azure VM
    - Container (Cloud Run / Azure Container Apps / similar)
    - Serverless function (Cloud Functions / Azure Functions)
- **Data layer**
  - At least one of:
    - Cloud Storage / Blob Storage
    - Managed database (Cloud SQL, BigQuery, Azure SQL, etc.)
- **Optional but encouraged:**
  - An AI / analytics component (e.g., notebook in Vertex AI / Azure ML, or calling a pre-trained AI service)

You may:

- Use a hand-drawn diagram and save as `architecture_diagram.png` or `.jpg`, **or**
- Use text-based diagramming (e.g., Mermaid) embedded in your markdown.

Deliverable files:

- `architecture_diagram.png` (or `.jpg`)
  and/or
- `architecture_plan.md` (with embedded Mermaid or a screenshot link)

---

3. Architecture & Implementation Plan (≈ at least 1-2 pages, can be more)

In `architecture_plan.md`, describe how each cloud service will be used, and connect back to prior course work.

Your plan must include:

1. **Service mapping**

   A table that lists at least 3–4 distinct services you will use and how:

Layer	Service (Cloud)	Role in Solution	Related Assignment/Module
Storage	e.g., GCP Cloud Storage / Azure Blob Storage	Store raw uploaded CSV files	Assignment/Module X
Compute	e.g., Cloud Run / Azure Container Apps	Run containerized Flask API	Assignment/Module X
Database/SQL	e.g., Cloud SQL / Azure SQL / BigQuery	Store cleaned/aggregated tables for reporting	Assignment/Module X
Analytics/AI	e.g., Vertex AI Notebook / Azure ML Notebook	Run simple analytics or a pre-trained model (optional)	Assignment/Module X

2. **Data flow narrative**

   A short narrative (or bullets) describing the end-to-end flow, for example:

   - Step 1: User uploads a CSV file via Flask.
   - Step 2: File is stored in Cloud Storage.
   - Step 3: A scheduled job or function loads the data into a database.
   - Step 4: A notebook or API runs simple analytics and writes back summary tables.
   - Step 5: Flask displays summary stats and graphs.

3. **Security, identity, and governance basics**

   In 1–2 paragraphs, briefly outline:

   - How you would*manage credentials (e.g., environment variables).
   - What kinds of access controls / RBAC would be needed.
   - From a high level, how you avoid putting real PHI into public environments?

4. **Cost and operational considerations**

   In 1–2 paragraphs, discuss:

   - Which components might cost the most (compute vs. storage vs. AI).
   - Where you might use serverless / scheduled jobs instead of always-on VMs.

    - Any decisions you'd make to keep this in a "student budget" or free tier.

Deliverable file:

- `architecture_plan.md`

---

4. Reflection (≈ 1 page)

In `reflection.md`, address:

- Which parts of the design you feel most confident about, and which parts you are least sure about.
- At least one alternative architecture you considered and why you did *not* choose it.
- If you had 4–8 more weeks and unlimited credits, what next steps you would implement.

Deliverable file:

- `reflection.md`

Optional: Prototype Track (Extra Credit)

For extra credit, you may implement a small working prototype of part of your design, using Python + Flask.

This does not need to implement the entire architecture. A minimal prototype could include/based based on:

- Use Flask with at least one route (`/` or `/upload`).
- Interact with at least one cloud resource, for example:
    - Read from or write to a Blob/Cloud Storage bucket.
    - Query a managed SQL DB or BigQuery table.
    - Call a serverless function or container endpoint that you deployed.
- Include clear configuration instructions in a `README` under a `prototype/` folder.
- A **video would also be VERY HELPFUL** in reviewing the application, in case I am not able to get it running on my end.

Suggested minimal structure:

```
assignment_final/
  readme.md                 # High-level instructions for the assignment
  use_case.md
  architecture_plan.md
  architecture_diagram.png  # or .jpg or mermaid in markdown
  reflection.md

  prototype/                # Only if you attempt the optional track
    app.py
    requirements.txt
```

```
    video_link.text          # Optional, video would be helpful (e.g.,
youtube, loom, etc...)
    README.md                # How to run the prototype and what it does
```