Big Data and Cloud Computing - Project #1

João Varelas - up201609013
Bárbara Neto - up202106176
Leandro Costa - up202408816

March 2025

1 Introduction

The goal of this project is to design and implement a scalable, reliable, and highly available backend system for a hospital electronic health record database, inspired by the MIMIC-III dataset, which contains real, anonymized medical data. Leveraging Google Cloud Platform (GCP) services, the system aims to efficiently manage hospital data, support patient admissions, store multimedia files, track medical progress, and facilitate patient-doctor interactions through a question-and-answer mechanism. This report outlines the system's design, key implementation details, and planned evaluation, demonstrating how cloud resources enable a robust solution for scalable applications.

2 Design

The system architecture is built around RESTful principles and integrates multiple GCP services to ensure scalability, reliability, and high availability. The main components and their interactions are as follows:

- **Google App Engine (GAE):** Hosts the REST API, implemented in Python 3 with Flask web server, providing CRUD endpoints for managing patients, admissions, progress, media, and questions. GAE's auto-scaling ensures the system adapts to varying loads.
- **Cloud SQL (MySQL):** A managed database storing structured data (e.g., patients, admissions, progress, questions). The schema is a simplified version of MIMIC-III, with tables for `PATIENTS`, `ADMISSIONS`, `PROGRESS`, and `QUESTIONS`. It uses a private IP and VPC Serverless Access Connector for secure, internal communication with GAE, enhancing reliability and security.
- **Cloud Storage Buckets:** Stores unstructured data such as images, videos, and documents linked to patients as Blobs. The REST API facilitates upload and retrieval, interfacing with Cloud Storage via GCP APIs.
- **Google Cloud Functions (FaaS):** Planned for periodic tasks, such as updating a list of patients with the longest waiting times based on admission data when triggered by a HTTP request.

Figure 1 illustrates the system architecture, showing how GAE interacts with Cloud SQL and Cloud Storage via secure, private connections, while Cloud Functions handle background computations.

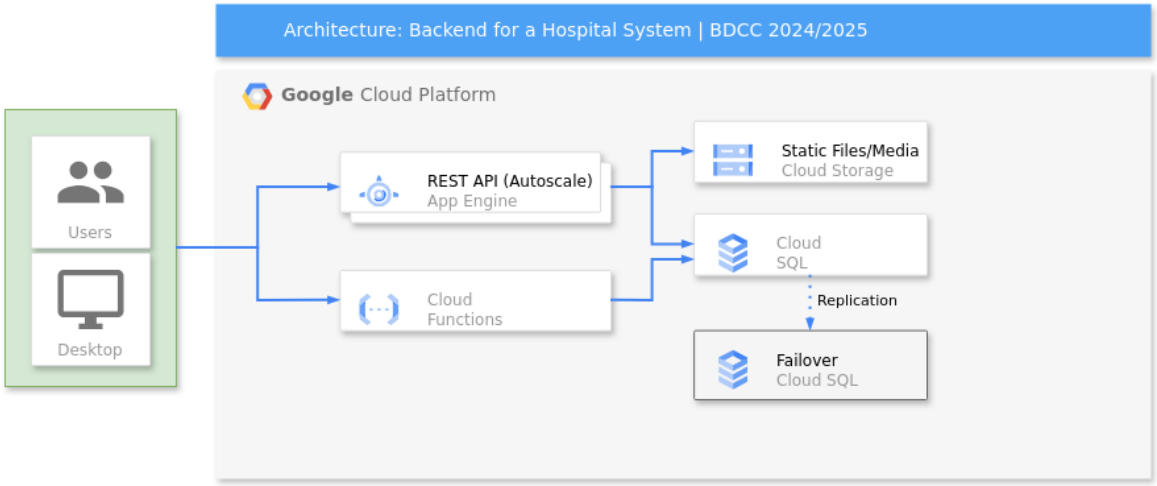

Figure 1: System Architecture Overview

3 Implementation

The REST API, deployed on Google App Engine, is developed using Python 3 and Flask, leveraging Google Cloud API libraries for seamless integration with GCP services. Key implementation highlights include:

- **REST API with CRUD Operations:** The API provides full CRUD (Create, Read, Update, Delete) functionality for managing patient and admission records. These operations allow clients to create new entries, retrieve information, update existing records, and delete data as needed. All endpoints are designed for scalability and performance under load. The API was tested locally using Google `dev_appserver.py` for emulating Google App Engine locally without having to deploy to cloud everytime.
- **Database Connectivity:** During development, a local MariaDB server mirrors the Cloud SQL MySQL instance. In production, GAE connects to Cloud SQL via a VPC Serverless Access Connector, ensuring a private, secure link. The database is configured for high availability with automatic scaling and failover.
- **Media Management:** Patient-related media (e.g., ID images, medical scans) are uploaded to Cloud Storage buckets.
- **FaaS:** A Google Cloud Function is planned to periodically compute and cache the list of patients with the longest waiting times. This function will query Cloud SQL, process admission timestamps, and store results for quick API access.

Scalability is achieved through GAE's auto-scaling and Cloud SQL's ability to dynamically adjust resources, while high availability is ensured by GCP's managed services and private networking.

4 Load Testing

The system's performance was evaluated using Artillery, a robust load-testing tool, to assess its scalability and responsiveness under heavy traffic. The tests simulated multiple concurrent users performing actions such as creating patient records, uploading media, and querying data. Key performance indicators, including response times, throughput, and error rates, were closely monitored.

To ensure scalability, Google App Engine (GAE) was configured to automatically scale instances based on CPU usage and request volume. Cloud SQL was set up with read replicas and automatic storage expansion to handle increasing data loads.

High availability was tested by deploying the system specifically on the europe-west1 region. Latency and failover behavior were measured to ensure consistent performance during potential regional disruptions.

The system is expected to handle hundreds of requests per second, with sub-second response times under normal conditions.

5 Conclusion

This project demonstrates the effective use of GCP services to build a scalable and reliable hospital backend system. Google App Engine provides a flexible environment for the REST API, while Cloud SQL ensures secure data management with high availability. Cloud Storage efficiently handles multimedia files, and Google Cloud Functions optimize real-time data processing.

By leveraging these tools, the system can manage hospital workflows like patient admissions, medical records, and progress tracking, ensuring performance under high demand through load testing and multi-region deployment. Overall, the system is designed for scalability, reliability, and security.