Cloud Computing “My diary” project

Milestone 2

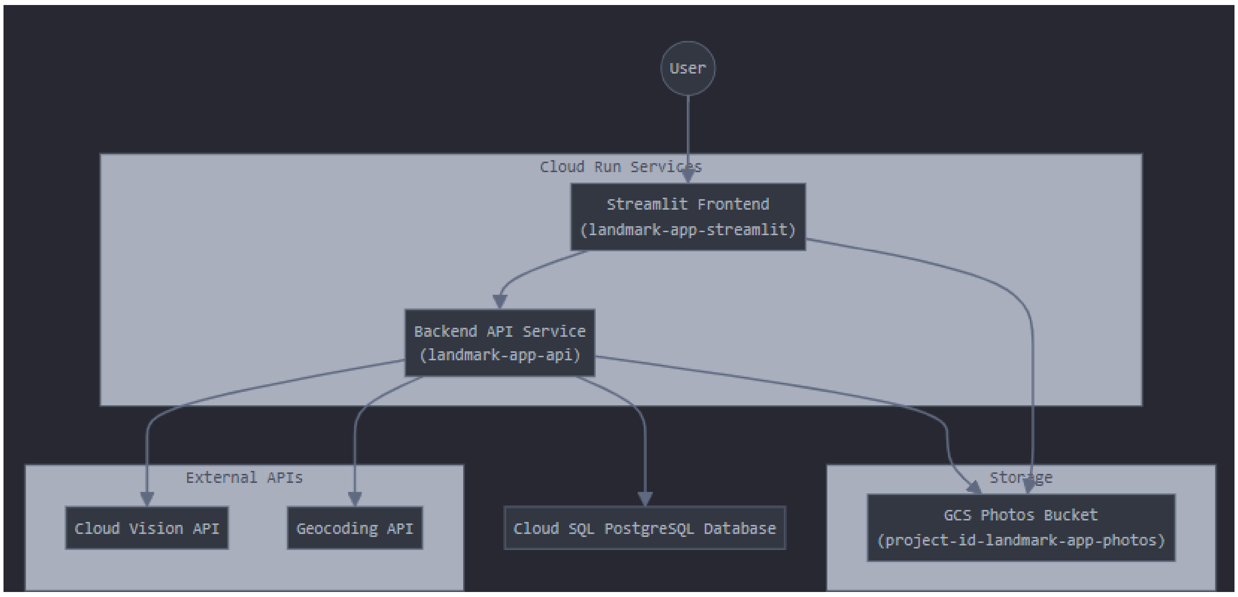
Jakub Rymarski, Jakub Sawicki, Kacper Trębacz

1. Project Overview

The goal of this project is to build a cloud-based application using Google Cloud Platform and Terraform for infrastructure automation.
The application allows users to upload photos (e.g. from vacations) and then uses Google Cloud Vision API and Google Geocoding API to detect landmarks in the photo and retrieve their geographic coordinates.

Infrastructure is deployed entirely through Terraform, ensuring that the setup process is automated, repeatable and easy to maintain. The architecture follows a microservices pattern where services are loosely coupled and communicate via event triggers and HTTP APIs. The design emphasizes scalability, fault tolerance and operational simplicity.

2. Diagram with microservices and their connections



The system architecture is organized as follows:

- Users create accounts and log in.
- Frontend is implemented in Streamlit and core backend services are provided with Django and served using Cloud Run.
- Database (PostgreSQL) stores structured data, including:
  user profiles, uploaded photo metadata, results of the landmark detection and geocoding operations.
- Google Cloud Storage is used to store the actual image files (photos uploaded by users).
- Google Landmark Detection API is used to analyze uploaded images and detect landmarks automatically.
- Google Geocoding API is used to obtain geospatial coordinates (latitude, longitude) from the landmark names detected.

3. Design APIs

The system exposes a RESTful API over HTTPS, providing a simple and intuitive interface for client applications to interact with backend services. The most important endpoints include:

POST /api/upload_photo/
This endpoint allows users to upload a photo to the system. The request will contain an image file, which is stored in Google Cloud Storage, along with metadata.

POST /api//{photo_id}/
In case the analysis process (landmark detec*trigger_analysis*tion and geocoding) did not run automatically after upload, or if the user wishes to re-trigger the analysis this endpoint can be used.

GET /api/photo/{photo_id}/
This endpoint retrieves the metadata associated with a specific uploaded photo. Metadata includes details such as the detected landmark, geolocation, upload time, and the processing status (whether the photo has been analyzed or is still pending).

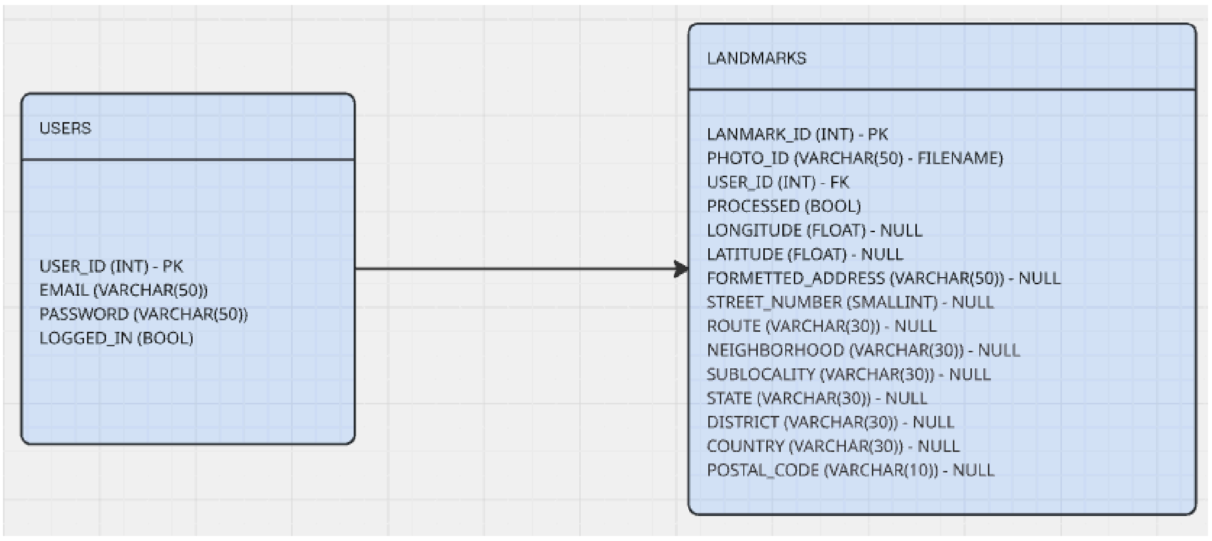
DELETE /api/photo/{photo_id}/
This endpoint allows users to delete a specific photo from the system, including both the image file from Google Cloud Storage and the corresponding metadata from the Cloud SQL database.

GET /api/user/{user_id}/photos/
This endpoint allows users to retrieve a list of all photos they have uploaded to the system. The photos will be returned with basic metadata such as photo IDs, upload times, and processing statuses.

4. Storage characteristics

The system handles both structured and unstructured data. Uploaded photos are stored in Google Cloud Storage, leveraging its scalable, highly durable, and globally available object storage service. Metadata about each photo (e.g. user ID, landmark name, coordinates, upload time) is stored in Cloud SQL (PostgreSQL), offering strong consistency and transactional capabilities.

Data access includes read and write, with a higher volume of write operations during photo uploads and read operations when retrieving photos or results. The expected data volume is initially small to moderate, measured in gigabytes, but it is designed to scale as the system grows. Regular data interactions occur, as new records and photos are added during uploads, while user interfaces frequently query the database for results.


USERS
USER_ID (INT) - PK
EMAIL (VARCHAR(50))
PASSWORD (VARCHAR(50))
LOGGED_IN (BOOL)
LANDMARKS
LANMARK_ID (INT) - PK
PHOTO_ID (VARCHAR(50) - FILENAME)
USER_ID (INT) - FK
PROCESSED (BOOL)
LONGITUDE (FLOAT) - NULL
LATITUDE (FLOAT) - NULL
FORMETTED_ADDRESS (VARCHAR(50)) - NULL
STREET_NUMBER (SMALLINT) - NULL
ROUTE (VARCHAR(30)) - NULL
NEIGHBORHOOD (VARCHAR(30)) - NULL
SUBLOCALITY (VARCHAR(30)) - NULL
STATE (VARCHAR(30)) - NULL
DISTRICT (VARCHAR(30)) - NULL
COUNTRY (VARCHAR(30)) - NULL
POSTAL_CODE (VARCHAR(10)) - NULL

5. Terraform

The system infrastructure is fully managed and deployed using Terraform, which automates the setup and configuration of Google Cloud resources. Terraform scripts enable the necessary GCP APIs (such as Cloud SQL, Cloud Storage, Cloud Run, and IAM) and create Google Cloud Storage buckets for storing photos. It provisions the Cloud SQL database instance and deploys the Streamlit application on Cloud Run, enabling efficient and scalable hosting. The API Gateway is configured to route requests to the Streamlit service. Additionally, monitoring resources are defined to track and analyze application performance.

6. SLA, SLO, SLI

A clear strategy for service reliability and performance monitoring is established through the use of SLA (Service Level Agreements), SLO (Service Level Objectives), and SLI (Service Level Indicators).

SLA:

The system benefits from Google Cloud Platform's native SLAs, which guarantee high availability for managed services. For example, Cloud Run offers a 99.95% uptime SLA, and Cloud SQL provides a 99.99% regional availability SLA. These guarantees ensure that core platform services experience minimal downtime, contributing to overall system reliability. In addition to relying on GCP's SLAs, we enforce our own SLAs, aiming to ensure that critical services: photo upload and photo processing. The internal SLA ensures: 99.95% uptime for the photo upload and photo analysis APIs.

SLO:

Internal service objectives are set to maintain a consistently high-quality user experience. Specifically, the system targets an API availability of over 99.95% measured over a rolling monthly period and aims to complete photo processing within 30 seconds of upload in at least 95% of cases. These goals are aligned with user expectations for responsiveness and system dependability.

SLI:

Concrete operational metrics have been defined to track service health and performance. Key SLIs include:

- Metrics provided with GCP including Cloud Run and Cloud SQL availability.
- The percentage of API requests that return a successful (HTTP 2xx) status code for photo upload and processing endpoints (target: 99.95%).
- The time elapsed from successful photo upload to the completion of the photo analysis process (95% of photos processed within 30 seconds of upload).