Project Report for Spotify Journal

Participants

- Sriram Yadavalli

Accomplishments

Wear OS Application

- On launch, creates 3 VMs in my GCP project.
- Next, it starts collects the user's fitness data in the background.
- This data is then uploaded to a storage bucket

VMs running scripts

- I ended up using 2 VMs for this application. One that gets the users song data from Spotify and one that gets, processes, and creates a journal playlist.

Storage buckets

- There are two storage buckets in my application. One stores the user's spotify data in a JSON file and another stores the fitness data in text files.

Cloud run functions

- I have a cloud run function that monitors the buckets and checks if they are filled the expeted amount of data. For instance, the fitness data should have 24 entries since I collect data every hour. When it meets the requirements the playlist creation stage can begin on a VM.

Hardware and Software Components

Hardware

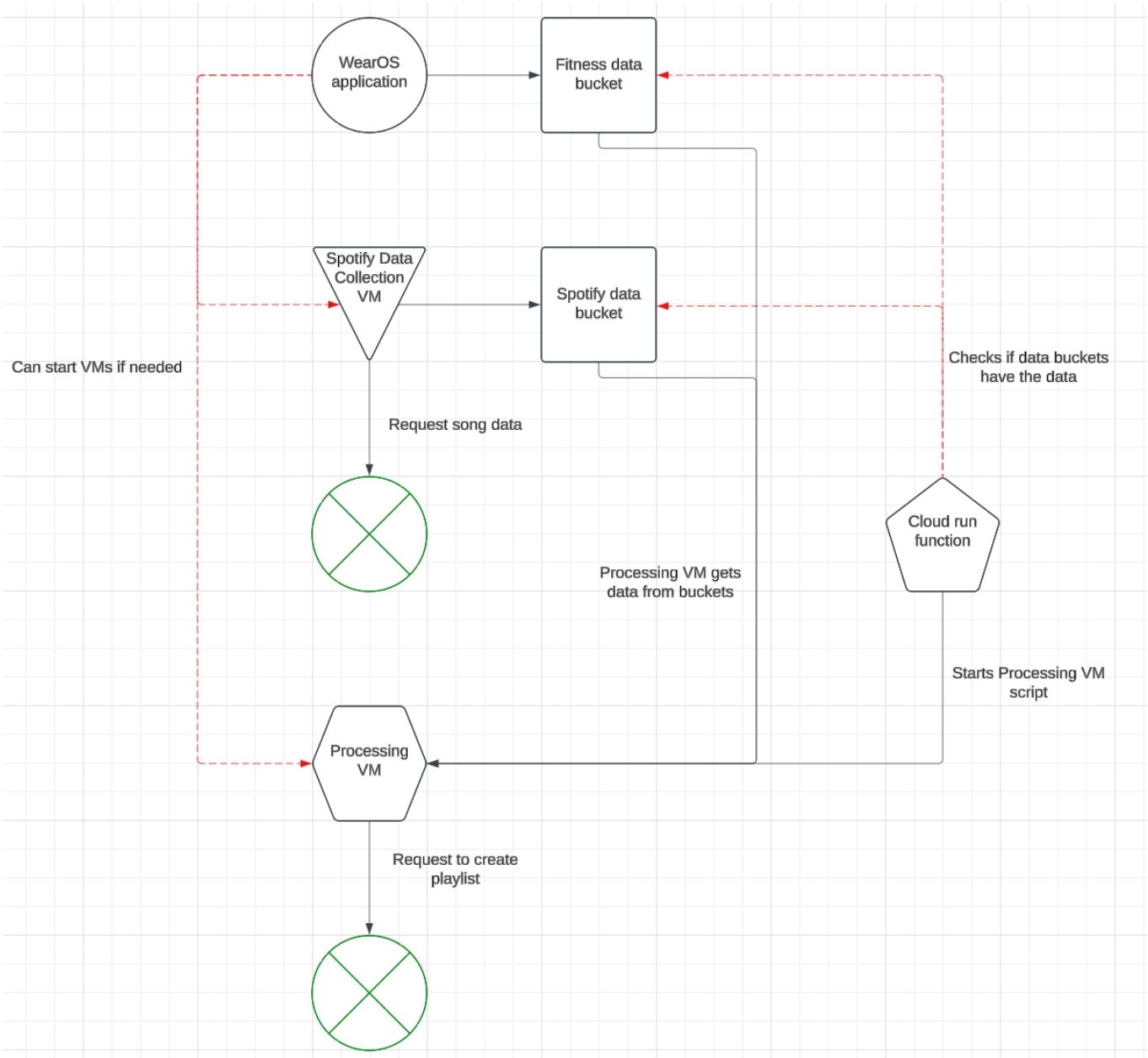
- Google Pixel Watch 2

Software

- GCP Compute Engine (Python scripts)
- GCP Cloud Storage
- GCP Cloud Run Functions (Python script)

- WearOS Application (Java/Kotlin)

Architectural Diagram



Debugging/Testing

- Android Studio device emulator
- Physical device testing
- Practical tests (real world)
- Automated tests

System capabilities and limitations

- The system can handle 24/7 operation of the VMs, storage buckets, and cloud run functions. They are set to never turn off unless forced by the user.
- On the Spotify user data collection and processing side the VMs are able to collect large amounts of data, but this take a very long time. To ensure there operation is not slow. Spotify user data collection occurs once a day instead of every hour or every few minutes.
- However, we cannot do this on the fitness data collection side since we need to be constantly collecting user data in the background and sending larger files takes much longer. If the user has bad network connectivity, uploads can be missed and the application might not have enough information to create a playlist.
- Even though we run the spotify data collection infrequently, the amount of time it takes is still long. This can cause an issue on the playlist creation side. This might not be visible to the user but it can take several minutes to get, process, and create the playlist as there are many small calls occuring to Spotify, sifting through track characteristics, aggregating the results, and creating a playlist with that data.
- Due to Spotify deprecating the ability to get track audio features using the API. I wasn't able to map the characteristics of a song to the fitness data in a way that I initially envisioned. Now, I look at the affinity values over the medium term (Spotify provides this data)