Intelligent Fitness Music Companion

Team Members:

1. Manikandan Gunaseelan (magu9411)
2. Rutuja Nikumb (runi8240)

Project Goal:

Our goal is to build a cloud-based intelligent fitness companion that captures users' performance trend during their workouts and **dynamically adjusts music** (picking the next song on the workout playlist) based on users' heart rate and activity level.

We are both runners and we have faced the problem of constantly changing our music during runs and would love to have it done automatically.

List of software and hardware components:

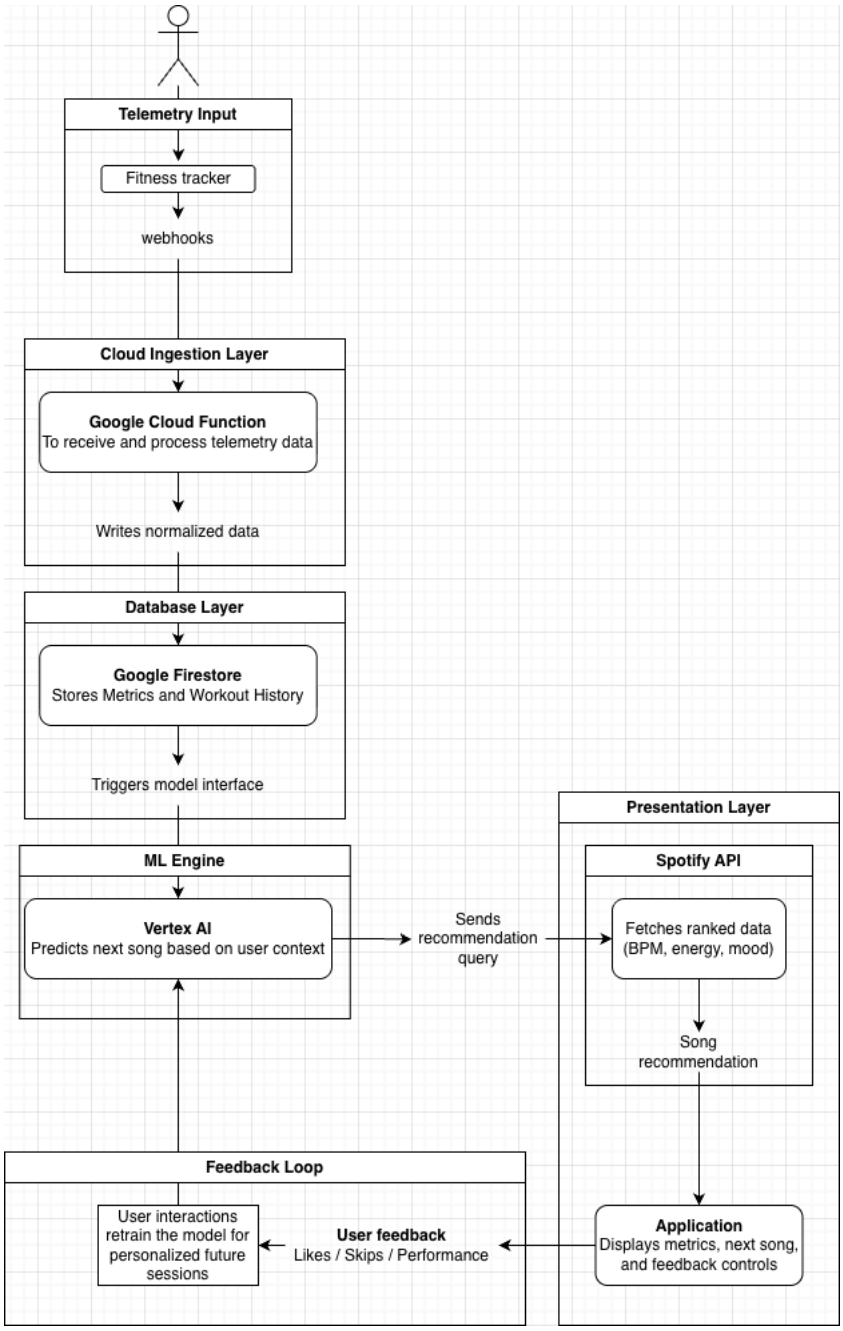
Hardware Components:

1. Fitness Tracker (Fitness watch or band that broadcasts heart rate)
2. Mobile Phone (Fitness tracking applications – Strava, Apple Fitness)

Software Components:

1. Presentation Layer:
    a. React
    b. APIs – Strava API (telemetry, workouts) and Spotify (songs metadata and preferences)
2. Cloud Ingestion Layer:
    a. Cloud Functions
3. Streaming:
    a. Google Pub/Sub for ingesting real-time telemetry.
4. Database and Caching:
    a. Google Firestore
    b. Redis
5. Storage:
    a. Google Cloud Storage
6. Authentication:
    a. Firebase Authentication
7. Machine Learning
    a. Vertex AI

System Architecture Diagram:



Interaction between different hardware and software components:

1. Fitness tracker captures real-time telemetry.
2. Data is transmitted via webhooks to the Cloud Function endpoint in GCP.
3. Cloud Function processes the incoming telemetry data and writes it to Firestore in the Database Layer.
4. Telemetry updates are also published to Pub/Sub for asynchronous event handling.
5. Firestore triggers a Vertex AI model inference when new data arrives, or some activity update occurs.
6. Vertex AI retrieves relevant historical and real-time metrics to predict:
   - Optimal song bpm or a suitable genre.
7. Vertex AI model then sends a recommendation query (including BPM, genre) to the Spotify API, which returns a ranked list of tracks matching those characteristics.
8. Recommended songs are returned to the Frontend Application, which displays real-time user metrics, current playing song, and suggested upcoming track.
9. The user provides explicit feedback (likes/skips) or implicit feedback (heartrate increasing or decreasing), which is logged in Firestore as user interaction data. Vertex AI periodically uses this feedback data for model retraining and improving personalization.

Debugging the project and train-test mechanism:

<u>Debugging:</u>

1. We will debug this project using mock fitness data (heartrate, cadence) to simulate live webhook events.
2. Enable logging/debugging in the workflow pipeline wherever possible.

<u>Train – Test Mechanism:</u>

1. Dataset creation – combine historical user telemetry with Spotify track metadata
2. Label the data on “effective motivators” – songs that improved performance
3. Model training – Train a supervised learning model that maps user context to music features, on Vertex AI workbench, split the dataset and evaluate metrics such as accuracy
4. Model testing – test inference pipeline on unseen telemetry data to ensure generalization. Compare predicted songs with the actual preferred tracks in user logs.
5. Retraining loop – periodically retrain model using new feedback data.

Why our project idea will meet the eventual project requirements –

Our project makes use of the following datacenter software components –

1. Message Marshalling – Incoming telemetry is encoded and normalized by Cloud Functions before publishing to Pub/Sub.
2. API interfaces – In webhooks to send the telemetry data from the fitness tracker.
3. Message queues – Google Pub/Sub for ingesting real-time telemetry.
4. Databases – Google Firestore for storing telemetry data and triggering Vertex AI model inference.
5. Storage services – Google Cloud Storage to store ML model artifacts, logs, and raw telemetry archives for retraining or analytics.