Asiana Holloway
April 21, 2025
SAT5317

WellnessWave App Final Project Report

The WellnessWaveApp is a full-stack mobile application developed to support daily stress monitoring using both self-reported mood evaluations and real-time health data collected from a Fitbit wearable device. This application was built using Flutter for the frontend mobile client and integrated with various backend cloud services offered through Amazon Web Services (AWS), including Lambda functions, API Gateway, and DynamoDB. The goal was to provide users with a simple, elegant interface where they could log in, track how they feel on a daily basis, understand their heart rate trends, and store this information securely.

The overall architecture of the system consists of two primary components: the frontend mobile application and the backend cloud infrastructure. The mobile application allows users to log in through a dedicated login screen and navigate to a home screen that welcomes them and encourages stress management. From there, users can access a Daily Check-In screen, where they are prompted with the question "How are you feeling today?" and can choose between a happy, neutral, or sad mood icon. They can also access an educational stress reference page that provides helpful information around their heart rate and what the mood check-ins suggest. For example, the app explains that a resting heart rate above 90 bpm may indicate stress, and that Fitbit calculates stress levels based on heart rate variability (HRV), with lower HRV often suggesting higher stress levels.

A major feature of the app is the Query Heart Rate screen, where users can input a date range and retrieve heart rate data directly from the Fitbit API. Once retrieved, users are given the option to store this data through the Save Heart Rate screen, which confirms when the data has been successfully saved to the cloud. These functionalities are supported on the backend through several Lambda functions: myFitbit_auth handles the initial authentication with Fitbit using OAuth 2.0 and PKCE, myFitbit_refresh refreshes expired tokens, auto_TokenRefresh automates the refreshing process when needed, getHeartRate pulls heart rate data based on user-specified dates, and saveHeartRate writes the retrieved data to DynamoDB. All these Lambda functions are exposed through API Gateway under a single TestAPI setup, which securely connects the mobile client to the backend services.

In terms of data storage, two DynamoDB tables are used. The FitbitToken table stores access and refresh tokens per user, while the HeartRate table stores the actual heart rate data retrieved from Fitbit. These components work together seamlessly to support the mobile client's functionality while maintaining secure and scalable backend operations.

The most challenging part of this development process was managing the OAuth2.0 workflow with Fitbit. It required deep understanding of the PKCE flow, accurate encoding of credentials, and careful handling of authorization and refresh tokens. Getting the initial access token and properly refreshing it through AWS Lambda and Postman involved a significant amount of trial and error, especially in formatting headers and configuring redirect URIs.

Asiana Holloway
April 21, 2025
SAT5317

WellnessWave App Final Project Report

To resolve these challenges, I followed Fitbit's developer documentation carefully and used the AWS Console to test and debug Lambda functions. I also used Postman to verify that each endpoint worked as expected and added error handling in the backend to detect when tokens had expired. CloudWatch logs provided insights into backend function behavior, which was helpful during debugging. Once I had a stable refresh flow, I integrated these services into my mobile app using Dart and confirmed that the frontend could communicate reliably with the backend.

Overall, my WellnessWave App helped me develop hands on experience building a connected mobile health application from end to end. I learned how to combine secure API authentication, cloud function organization, and user-friendly mobile UI into a fully operational app. This project lays a strong foundation for future improvements, such as stress trend analytics, user notifications, and expanded health data integration.