Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Summer 2021

Team SpaceTabs
Life Fit

Sushant Gupta
Naveen Narayan Gnanapazham
Krishna Khadka
Shishir Pathak
Puskar Dev

Revision History

Revision	Date	Author(s)	Description
0.1	20.06.2021	SG	document creation
0.2	228.06.2021	SG, NG, KK, PK, SP	first version

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

Our sponsor, State Farm Insurance, is interested in creating a system that will maintain a profile for its customers' daily health habits to get a better view of their overall health, and using Machine Learning on their health data along with other information, determine better insurance rates for the individual. One can get the document for requirement specification at [System Requirements] and the architectural design at [Architectural Design].

The system consists of several subsystems such as Android app, Web app, AWS services, and Fitbit server. An AWS lambda script as well as android app is used to collect data from fitbit server that collects data from a fitbit watch the users wear every time. The user will be able to log in to their profile through android as well as web app which will display their health metric records and the fitness score they currently have. The health information to be recorded will include the heart rate, steps, sleep schedule, and exercising/burnt calories. This information will be graphed for the user in various time periods. The health record is stored in AWS S3 bucket and will be used to run Machine Learning to calculate health score. The web interface will also be the point where administrators will be able to review users.

2 SYSTEM OVERVIEW

The system consists of primarily three layers which interact with each other: the Presentation Layer, the Business Layer, and the Data Access Layer. Each layer consists of different subsystems. The presentation layer interacts with business layer to compute and produce several business cases results using several logic. The business layer subsystems provides medium to presentation layer subsystems to access the data access layer resources to have persistent data and file storage system. The following architectural design document provides more detail about each layer and their subsystem in our system.

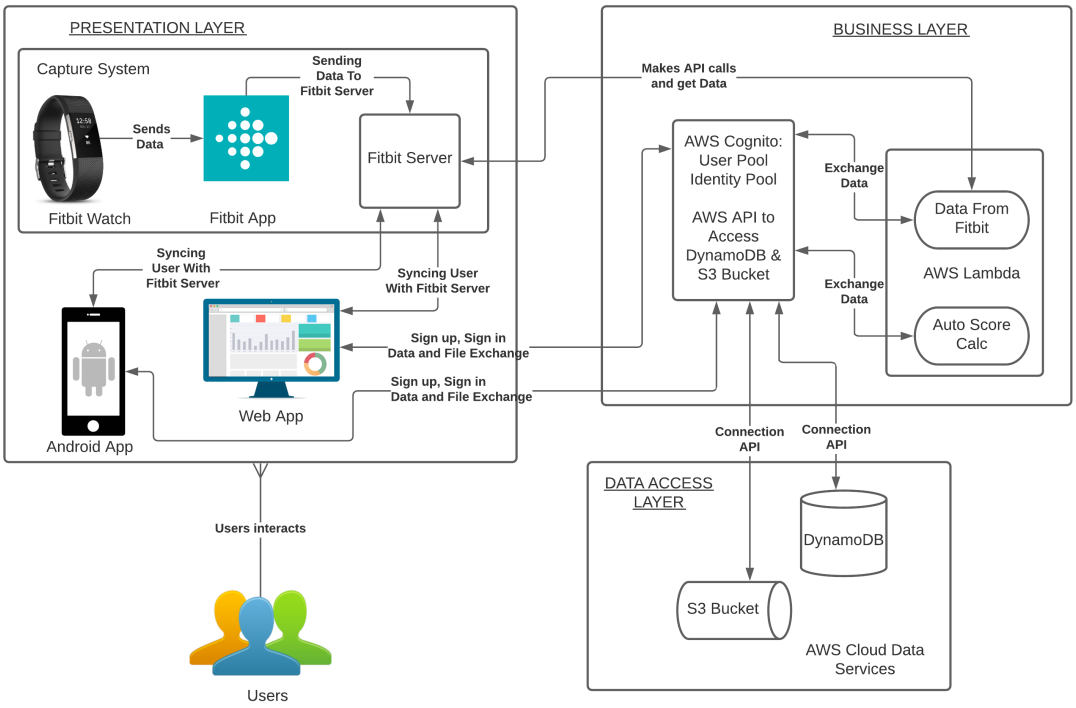

Fig.: Lifefit System Architecure

Figure 1: System architecture

3 Presentation Layer Subsystems

The presentation layer consists of three subsystems: Capture, Android App, and Web App. Each subsystem will have different use cases. The Capture Subsystem is responsible to collect user health data from a Fitbit smart watch. The Android and Web User Interface subsystems will enable the user to log in, register, and adjust their account, review and upload their activities on their mobile device, and view graphs illustrating their health metrics.

3.1 Layer Hardware

There is no common hardware to be used for the layer level.

3.2 Layer Operating System

Though each subsystems in this layer has their own OS, Android OS is used for Android devices, but Fitbit App is available for iOS as well. Web app can be run on any OS supporting browser.

3.3 Layer Software Dependencies

There is no common software dependencies for each subsystems. Each software dependencies is explained below.

3.4 Capture/Fitbit Operation

Capture subsystem includes the process of recording health data from user's fitbit watch, send it to connected Fitbit app, which then send the data to Fitbit server. It is also responsible to give proper authorization to our system to get user's health data through API call which then will be stored in AWS

S3 bucket through AWS API services. It will be accessed by the AWS lambda function to automatically collect data and stored at proper location.

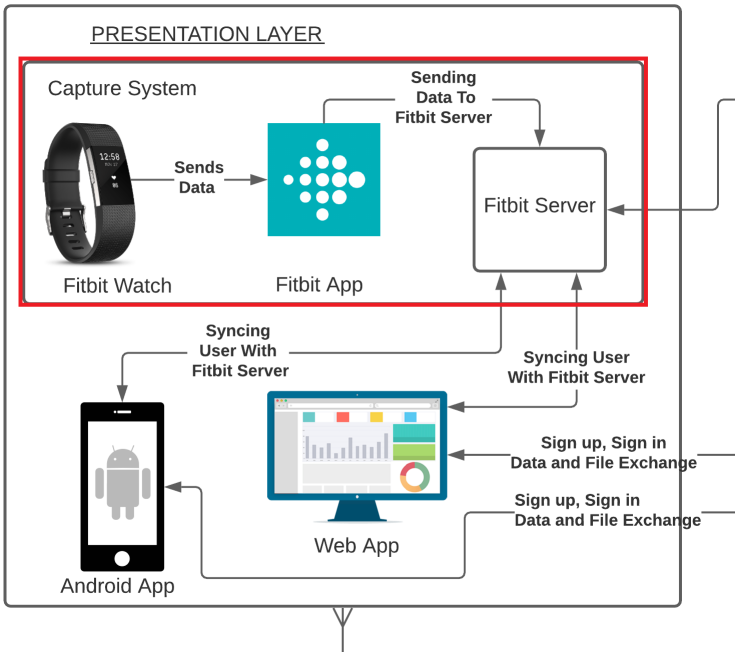

Figure 2: Capture Subsystem

3.4.1 Subsystem Hardware

- A FitBit smart watch which can collect important health data from the users.
- An Android device which can run the mobile app and connect with a FitBit smart watch.

3.4.2 Subsystem Operating System

Any Android OS using API 29 or higher

3.4.3 Subsystem Software Dependencies

Fitbit Services

3.4.4 Subsystem Programming Languages

No specific programming languages.

3.5 Android App UI

The android UI subsystem allows user to login, sign up, modify personal information and view their health data such as distance travelled, sleep hour, calories, and health score on their mobile device.

3.5.1 Subsystem Hardware

An android device that can run a mobile app.

3.5.2 Subsystem Operating System

Any android OS using API 29 or higher.

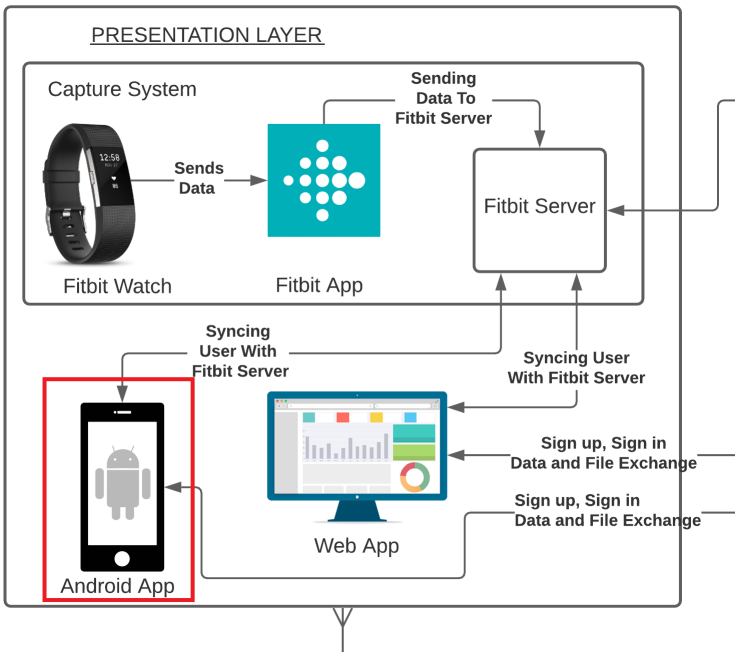

Figure 3: Android App Subsystem

3.5.3 Subsystem Software Dependencies

Android studio common framework used for app development. Libraries that are used for graphing health data.

3.5.4 Subsystem Programming Languages

The main programming language used in the android app system is Java. It is developed using Android Studio tool.

3.5.5 Subsystem Use Case Diagram

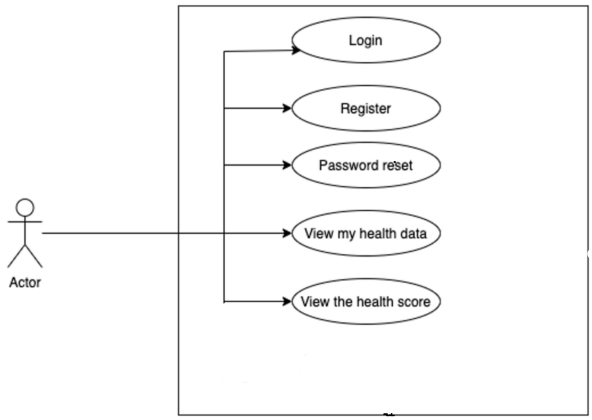

Figure 4: Android UI use case diagram

3.6 Web App UI

The web UI subsystem provides a different way for the user to obtain their account information where they can sign in, sign up, and view their data on various graphs and compare their daily activities.

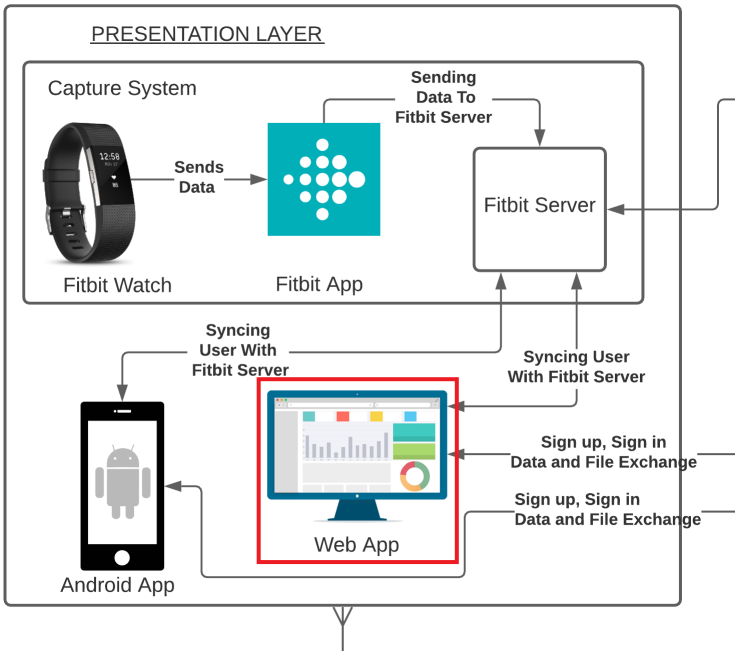

Figure 5: Web App Subsystem

3.6.1 Subsystem Hardware

Any device that can run a browser.

3.6.2 Subsystem Operating System

Any operating system that supports a browser.

3.6.3 Subsystem Software Dependencies

React and Flask framework with bootstrap CSS.

3.6.4 Subsystem Programming Languages

It uses several programming languages such as Javascript, HTML, CSS, and Python.

3.6.5 Subsystem Use Case Diagram

3.6.6 Subsystem Website Map

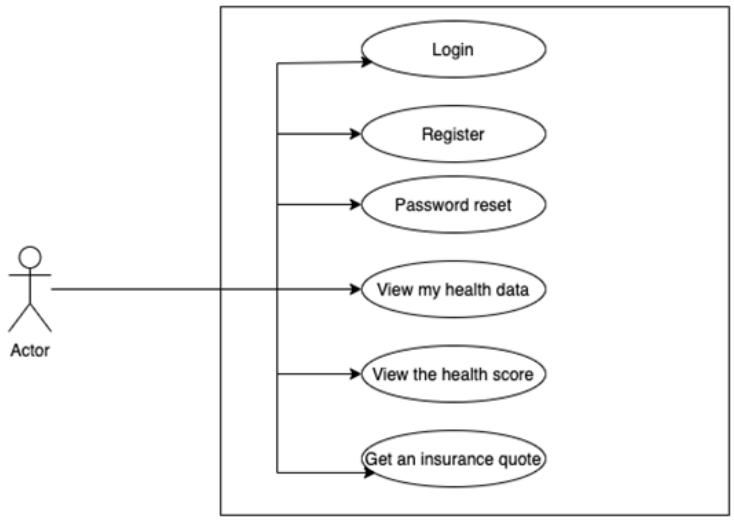

Figure 6: Web App use case diagram

Figure 7: Web Map Chart

4 Business Layer Subsystems

Hosts all of the RESTful API endpoints for the web, lambda, and mobile app. It also performs the data analysis on user data. This layer will respond to incoming transactions from the presentation layer and will send relevant data. Besides, it handles the authentication process along with the lambda services that calculates the score and get data automatically with API calls. AWS services is used at this layer to achieve the goal.

4.1 Layer Hardware

No hardware needed. Cloud service is used.

4.2 Layer Operating System

No specific but Lambda runs on AWS services.

4.3 Layer Software Dependencies

Python, Javascript, Scikit-Learn, Boto3, urllib, Amazon Java SDK, OAuth2.0

4.4 Lambda Functions

AWS Lambda is our subsystem 1. This subsystem has little to no hardware. It helps in automation of certain tasks in the app as a whole. Like automation of data collection, score calculation and ML Model training.

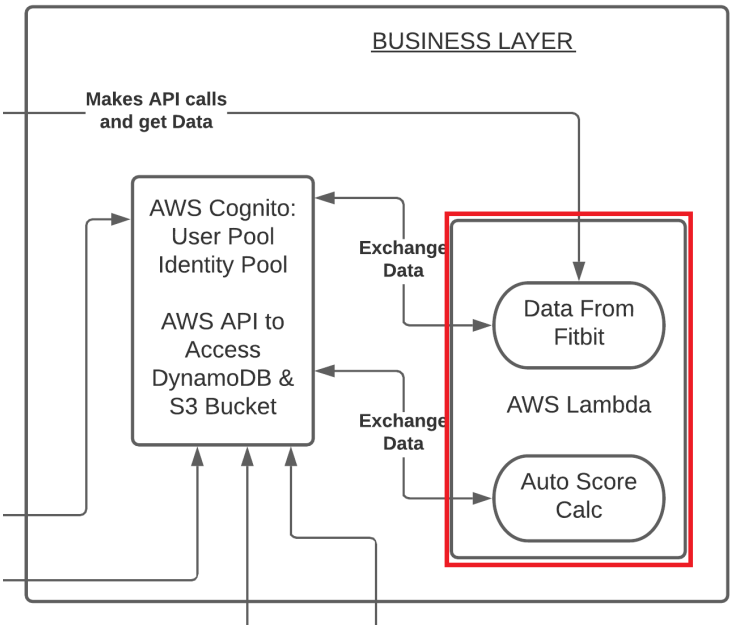

Figure 8: AWS Lambda Subsystem

4.4.1 Subsystem Hardware

This subsystem practically has no hardware.

4.4.2 Subsystem Operating System

AWS Lambda is completely based off of cloud computation and space. Hence no need for any particular Operating system.

4.4.3 Subsystem Software Dependencies

Apart from keeping version up to date in software, no other major component in this subsystem has software dependencies. But the scripts has Boto3, urllib, Amazon Java SDK, OAuth2.0

4.4.4 Subsystem Programming Languages

Python and its multitude of ML libraries like Sci-kit Learn, TensorFlow, etc. is used. Knowledge of building AWS Lambda services is key.

4.4.5 Subsystem Data Structures

The key aspect of this subsystem is being able to convert live data being fed into the âconverterâ model. In which data comes in a structured form but gets converted to â.csvâ for model building training purposes and storage purpose.

4.4.6 Subsystem Data Processing

Data comes in an already pre-packaged structured form from the Fitbit Server, but we need to make this permanent for ML model building purposes. Hence the conversion from this form to .csv file form.

4.5 AWS Cognito API

AWS Cognito and API is our subsystem 2. This subsystem has little to no hardware. AWS Cognito helps in authenticating our existing users and adding new users without any major security issues and with as minimal human intervention as possible. It does this by creating User pools and identity pools. This is a service provided by AWS to help increase security aspects of the early stages of project development. AWS SDK API which is an additional tool provided in JAVA SDK helps in using the AWS services with more control over the AWS services and customizing it to our needs. The mentioned Services have practically no hardware and our solely a cloud service.

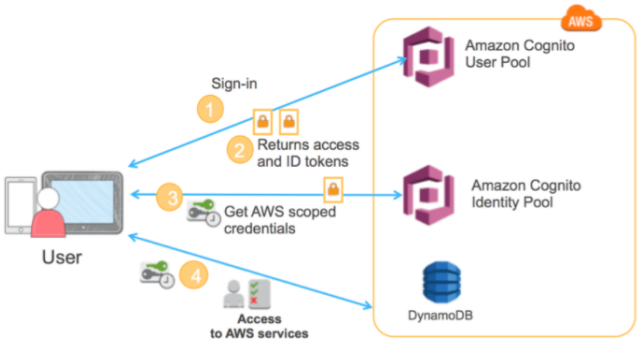

Figure 9: A representation of how AWS Cognito User and Identity Pools work.

4.5.1 Subsystem Hardware

No hardware is involved in this subsystem.

4.5.2 Subsystem Operating System

There is no specific Operating System requirement since, most of this is a cloud service and these services were designed to work in a variety of operating systems.

4.5.3 Subsystem Software Dependencies

Apart from keeping version up to date in software, no other major component in this subsystem has-software dependencies. But the custom code has Amazon Java SDK.

4.5.4 Subsystem Programming Languages

Mainly Java was used to customize the use of the AWS SDK. Apart from that the knowledge of how to use AWS Cognito Services is required.

4.5.5 Subsystem Data Structures

In AWS Cognito there is no specific data structure type followed. AWS just calls it a "Data Store" and follows this pattern for a multitude of its services.

4.5.6 Subsystem Data Processing

The Data Processing is a key aspect because the custom AWS API script written in Java SDK, communicates with the App to send data of the user directly to DynamoDB after which data exchange occurs between AWS lambda service. Once this process is done the user authentication occurs. AWS Cognito uses a token and access ID based service for its authentication.

5 Data Access Layer Subsystems

The data access layer is hosted by Amazon Web Service (AWS) cloud service. Amazon DynamoDB is used as a database to store a user login information and relevant details. Similarly, when a user consents to share their data from the app, their data is stored in AWS S3 bucket. User health data will be accessed from S3 bucket to perform analysis and send back their health score.

5.1 Layer Hardware

Server which hosts AWS cloud services. Otherwise, no specific hardware in this layer.

5.2 Layer Operating System

Amazon Linux.

5.3 Layer Software Dependencies

1. DynamoDB
2. Flask Framework in Python
3. AWS SDK for Java

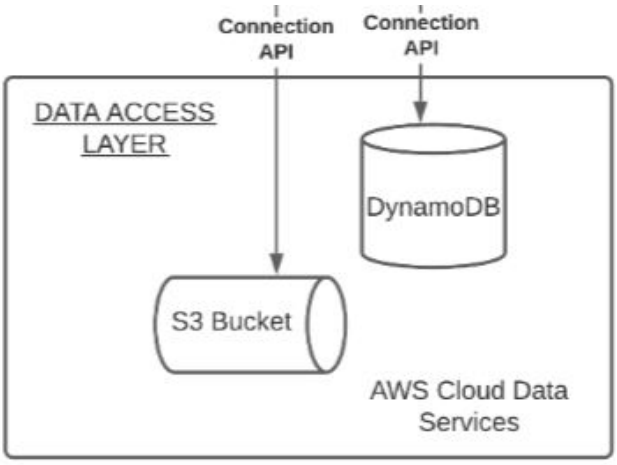

Figure 10: Data Access Layer

5.4 Amazon DynamoDB

Amazon DynamoDB is the subsystem 1 in the Data access layer. When a user signs up, their account information will be stored in DynamoDB. API requests will be performed on DynamoDB to retrieve a user information such as when a user signs in.

5.4.1 Subsystem Hardware

No specific hardware.

5.4.2 Subsystem Operating System

No specific OS. It supports any operating system.

5.4.3 Subsystem Software Dependencies

AWS Java SDK for Amazon DynamoDB to make API connections.

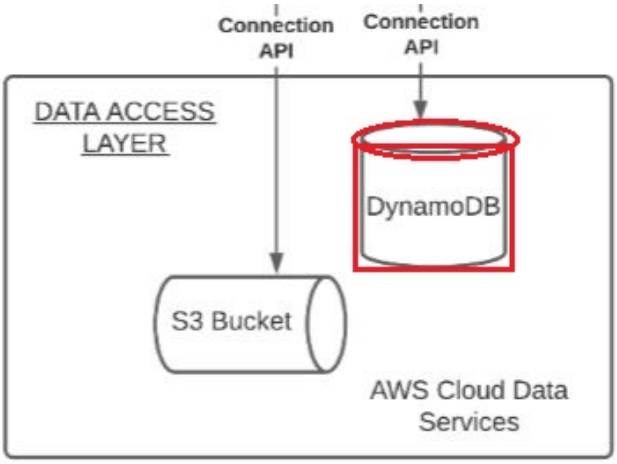

Figure 11: Amazon DynamoDB

5.4.4 Subsystem Programming Languages

Supports a variety of programming languages.

5.4.5 Subsystem Data Structures

The user account information data is stored in DynamoDB.

5.4.6 Subsystem Data Processing

No Data Processing required.

5.5 AWS S3 Bucket

AWS S3 Bucket is the subsystem 2 in the Data Access Layer. S3 Bucket stores a userâs health data accessed through FitBit API call from the app. Various health data that a user consents to provide will be uploaded to the S3 Bucket once they provide the access through the app. Such health information will be accessed API connections to compute the health score for a user and render it back.

5.5.1 Subsystem Hardware

No specific hardware.

5.5.2 Subsystem Operating System

Amazon Linux

5.5.3 Subsystem Software Dependencies

AWS SDK for Java.

5.5.4 Subsystem Programming Languages

Supports multiple programming languages.

5.5.5 Subsystem Data Structures

User health records will be stored here in csv files.

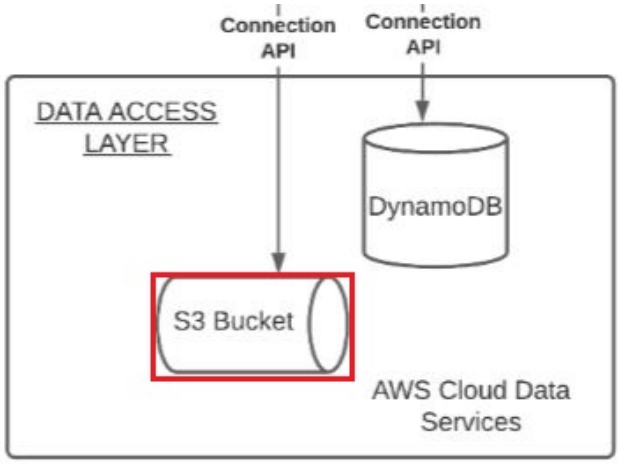

Figure 12: AWS S3 Bucket

5.5.6 Subsystem Data Processing

No Data Processing required.

6 APPENDIX A

Include any additional documents (CAD design, circuit schematics, etc) as an appendix as necessary.

References