12-1-2021

DESIGNING, DEVELOPING, AND DEPLOYING FAST AND SECURE SYSTEM ARCHITECTURE WITH AUTONOMOUS DATA IMPORTATION FEATURE FOR STATEFARM'S SPONSORED LIFEFIT APP

Sushant Gupta

DESIGINING, DEVELOPING, AND DEPLOYING FAST AND SECURE SYSTEM ARCHITECTURE WITH AUTONOMOUS DATA IMPORTATION FEATURE FOR STATEFARM'S SPONSORED LIFEFIT APP

by

SUSHANT GUPTA

Presented to the Faculty of the Honors College of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

HONORS BACHELOR OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2021

ACKNOWLEDGMENTS

First and foremost, I would like to thank my project mentor Dr. Chris Conly, an assistant professor at UT-Arlington, for providing me this opportunity to work on this great POC project by Statefarm Insurance Company. Professor Conly was always there to help us with any kind of assistances, including but not limited to, making available all required resources and helping with the better understanding of project requirements and specifications. I also must thank the Statefarm representatives, Dawsen Richins and Amy Simone, for setting the milestones high every week that inspired me to push limits and learn new things every day. I would also have to extend my gratitude to all UTA staff and faculties who have been the parts of my exciting journey of completing my first bachelor's degree at UTA.

I will never forget to acknowledge my family and friends for believing and always inspiring me. Their love and supports have always been my motivation to never back down and face every bit of challenges that came in my way of learning and growing academically. Lastly, I would like to thank my SpaceTabs team: Naveen Narayan Gnanapazham, Krishna Khadka, Shishir Pathak, and Puskar Dev, for being best team members and working hard to achieve our goal. Without all these above-mentioned people and/or entities, I would never be able to finish this project successfully.

August 20, 2021

ABSTRACT

DESIGINING, DEVELOPING, AND DEPLOYING FAST AND SECURE SYSTEM ARCHITECTURE WITH AUTONOMOUS DATA IMPORTATION FEATURE FOR STATEFARM'S SPONSORED LIFEFIT APP

Sushant Gupta, B.S. Computer Science

The University of Texas at Arlington, 2021

Faculty Mentor: Chris Conly

Our sponsor, State Farm Insurance, is interested in creating a system that will maintain a record of profile for its customers' daily health habits to get a better view of their overall health. The system should use Machine Learning on the user's recorded health data, along with other health related information, to determine how healthy the individual is and to provide better health and/or life insurance rates to that individual. Concerning the sponsor requirements, we created the system called Lifefit Application. It is a mobile (Android and IOS) and web-based application which, with the support of AWS cloud computing and storage services, gathers data from Fitbit watches, runs machine learning on the data to compute health scores, and finally displays the scores along with their daily health data like calories burnt, steps walked, heart rate, and sleep pattern, etc. to its

registered users. The health score can be used by the Statefarm Insurance Company to provide better insurance rates and incentives to its user base. The application is also a good source of inspiration to the users who can monitor their health activities and make significant changes needed to improve their scores as well as their health status.

TABLE OF CONTENTS

ACKNOWLEDGMENTS iii

ABSTRACT iv

LIST OF ILLUSTRATIONS viii

LIST OF TABLES ix

Chapter

1. INTRODUCTION 1

1.1 Problem Statement 1

1.2 Project Overview 1

1.3 Some Major Requirements 2

1.3.1 System shall collect data from Fitbit on its own 2

1.3.2 System shall use strong machine learning model to calculate health score of users 3

1.3.3 Applications shall have fast loading graphical representation of user's health data 3

1.3.4 System shall be secure and safe 3

2. METHODOLOGY 4

2.1 Designing System Architecture 4

2.2 Developing Systems 5

2.3 Deploying Systems 7

3. SYSTEM COMPONENTS 9

3.1 Mobile Application 9

3.2 Web Application 12

3.3 Machine Learning Module 13

4. AUTOMATION AND API 16

4.1 Task Automation 16

4.1.1 Automated Fitbit Data Importation 16

4.1.2 Automated Score Calculation 18

4.2 Web APIs for Data Handling 19

5. CONCLUSION 21

REFERENCES 23

BIOGRAPHICAL INFORMATION 24

LIST OF ILLUSTRATIONS

Figure		Page
2.1	System Architecture Diagram	4
2.2	Code Structure	7
2.3	Lambda Function Hosted	7
3.1	Mobile App Login UI	10
3.2	Mobile App Profile Page UI	10
3.3	Mobile App Dashboard UI	11
3.4	Mobile App Graphical UI	11
3.5	Web App Sign-up UI	12
3.6	Web App Sign-in UI	13
3.7	Web App Dashboard UI	13
3.8	RFC Model in Action	15
4.1	Fitbit Authentication Flow (Authorization Code Flow)	17
4.2	Automation Code Snippet	18
4.3	Web APIs Code Snippet	20

LIST OF TABLES

Table		Page
2.1	Tech Stack Used in Development	6
4.1	Routes for Web API	19

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

To determine a good insurance rate, one must consider several factors depending on the nature of the insurance. Determining a better life insurance rate can be challenging, as it involves a greater number of factors with comparatively higher uncertainties. With the wide development in science and technology, many insurance companies are looking for innovative techniques to achieve such goals. Since health status is a very important factor, traditionally a doctor visit is required to analyze a customer's health and provide a life insurance plan. But health being a dynamic factor, it is hard to predicate one's future health with a single doctor's visit, which only captures a snapshot of someone's health. An ongoing monitoring system would serve as a benefit for tracking someone's general health information across a long period of time and, thus, provide a better understanding of their health habits that can be used to determine their insurance plan. Our sponsor, Statefarm, is interested in creating a system that maintains profile of an individual's daily health habits to get a better view of his/her overall health by using Machine Learning on the health data along with other information and, thus, determine better insurance rates for that individual.

1.2 Project Overview

The project, Lifefit Application, is a sponsored project by Statefarm Insurance Company. It is a Proof-of-Concept (POC) project for Statefarm that has certain specific requirements with a wide range of possible additional features in the future. The project

consists of three major components: Mobile Application, Web Application, and Machine Learning Module. The main value proposition of the project is to provide Statefarm with a system or tool that they can use to monitor their customer's health habits to have more confidence while providing better insurance rates and other incentives. In short, it will serve as a tool for risk prediction and prevention so that the insurance company can produce more profit by attracting more customers with better insurance rates after analyzing the risk for every individual customer.

The system has different layers that work together to achieve the set goals. The system will get raw daily health data from user's Fitbit watch, format it in the proper .csv file, and then store them on AWS S3 buckets in systematic manner. Another component of the system will later get the data, run the machine learning model to calculate a score out of 10 (1 being least health and 10 being the healthiest), and finally stores the score in another .csv file for future use. Finally, the mobile and web applications will get those data and scores from S3 and display to the users. Statefarm will use the aggregate scores, calculated over a certain period, to provide better quote to their customers.

1.3 Some Major Requirements

Though Lifefit has most of the common application requirements such as sign-up, sign-in, password reset, user profile tab, and log-out, etc., it has some major specific requirements that are critical to this project. Some of these requirements are as follows:

1.3.1 System shall collect data from Fitbit on its own

The system should be able to collect daily health data such as steps walked, distance walked, calories burnt, heart rate, and sleep pattern, etc. from the user's Fitbit watch automatically, and store it in proper format to AWS S3 bucket for future uses.

1.3.2 System shall use strong machine learning model to calculate health score of users

The system should implement a strong machine learning model to calculate daily health scores based on the health data collected each day. Another algorithm should be used to find aggregate final scores using the past daily scores. The final aggregate score will be used by Statefarm to provide insurance quotes to its customers.

1.3.3 Applications shall have fast loading graphical representation of user's health data

Android and web applications should have the ability to render the user's daily health data using the most intuitive graphs. The applications also should be able to load graphs faster for better user experience. These features give users the ability to monitor their health status closely, make necessary changes in their health habits to achieve better health, and, thus, attain a better health score with better insurance rates from Statefarm Insurance Company.

1.3.4 System shall be secure and safe

Since the system deals with user's health and other sensitive information, the system is required to be secure and safe from data leaks and being hacked by external threats. It is important to have better authentication and API communications within the system.

CHAPTER 2

METHODOLOGY

2.1 Designing System Architecture

Since the project was inherited from prior teams, several security and performance issues were found in their existing system architecture. As a result, a new system architecture was proposed to the team, professor, and sponsors for better security and performance of the system. The current system architecture consists of three layers: Presentation Layer, Business Logic Layer, and Data Access Layer. Each layer communicates with each other in a safe and secure manner as shown in the diagram below.

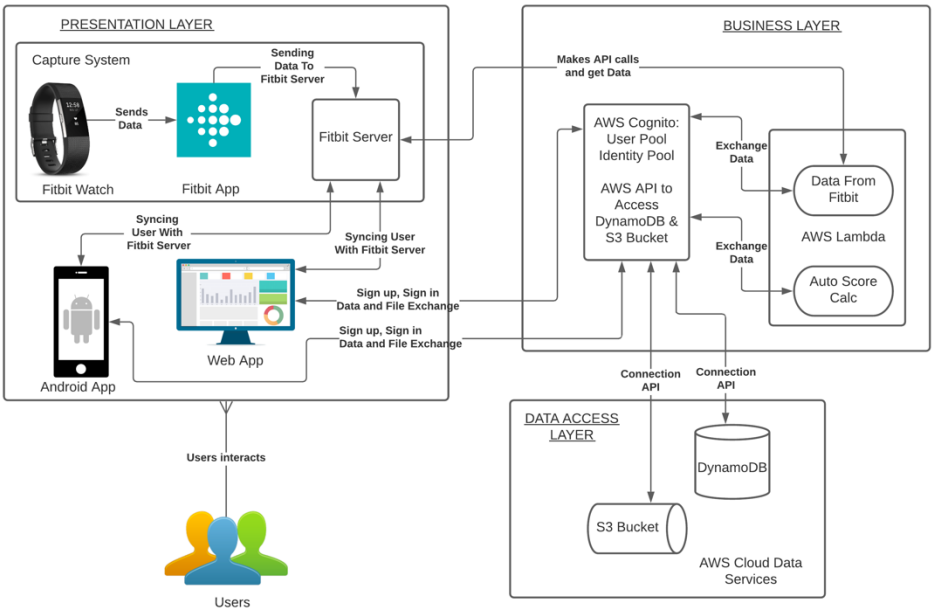

Fig.: Lifefit System Architecure

Figure 2.1: System Architecture Diagram

The Presentation Layer has the components in which users interact. It consists of three subsystems viz. Capture Subsystem, Mobile Application, and Web Application. The capture subsystem is made up of the Fitbit application and server that are responsible for recording the user's health data of every second the user is using the Fitbit watch. The Mobile and Web applications are responsible for display information like daily health data, scores, and profile details to the authenticated users.

The Business Logic Layer contains all the backend processing of the system. It has most of the AWS cloud computing services in our system. It is responsible for the automated data importation from the Fitbit server, using Fitbit API and calculating health scores. It also provides several API endpoints that are used to exchange data between data storage and android/web applications' user-interface (UI).

The Data Access Layer is the last, but critical, layer of the system where all kinds of data are stored. It consists of application databases and storage facilities. Subsystems in the data access layer communicate with subsystems in the presentation layer, such as mobile and web apps, through secure APIs hosted in the business logic layer.

2.2 Developing Systems

After architecting a strong and secure system, it is equally important to develop the system with the same amount of security and safety concerns in mind. This project has several subsystems, and, thus, several code bases. Different subsystems require different programming languages and frameworks. To choose the correct and most efficient tech stacks for the project, a significant amount of time was spent researching what programming languages and/or frameworks would be suitable for the kind of tasks in the system. Any programming language and/or framework selected for a particular task must

be able to successfully satisfy the complexity and security needs of the subsystem. Table 2.1 shows the complete list of the tech stacks used in various components of the project.

Table 2.1: Tech Stack Used in Development

	Programming Language	Framework (if any)
Mobile Application	JavaScript, Java, Others	React-Native
Web Application	JavaScript, HTML, CSS	ReactJS, Bootstrap
Web APIs	Python	Flask
Automation Code	Python	-

The use of reactJS and bootstrap frameworks increases the responsiveness and performance of the web application by a huge factor in comparison to the old web application designed by prior teams. The python Flask App API is used as an interface between the AWS data management services (DynamoDB and S3 bucket) and the web application for smooth and fast data transmissions. This increases the web performance by 60-70% in terms of data processing and rendering speed. Using React-Native instead of Android SDK for the mobile application, the application becomes OS independent and efficient enough to handle synchronous activities in it. AWS Amplify is used to implement the AWS Cognito and AWS Identity Pool to handle user authentication processes across the system with a greater security. Finally, the python script is developed to make the data importation process easy and autonomous with the help of AWS lambda.

To increase the productivity of the team as well as to minimize the loss if something goes wrong during development, proper management of the codebases was essential. This was accomplished by using GitHub as the version control tool and arranging all independent tasks in separate folders.

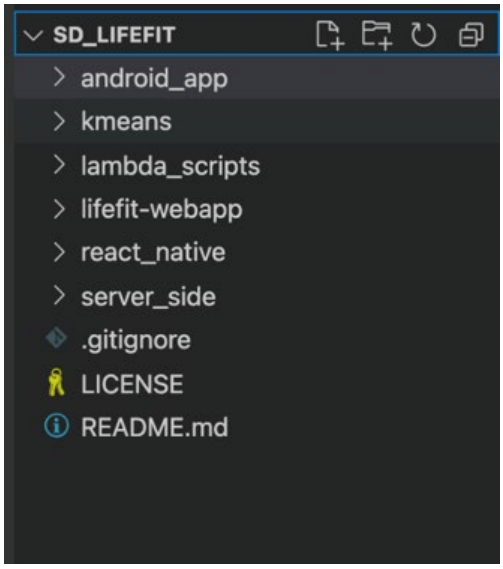

Figure 2.2: Code Structure

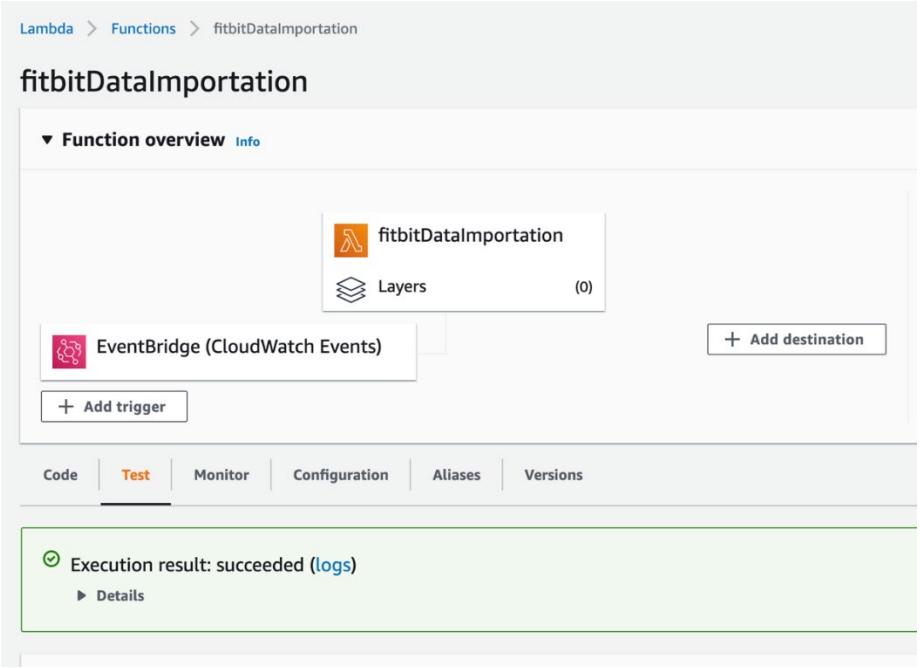

Figure 2.3: Lambda Function Hosted

2.3 Deploying Systems

Now that the system has been designed and developed, it is also very important to deploy the system in a safe and secure environment. For the project, some of the subsystems were needed to be hosted on the web, whereas others needed to be hosted

natively on the development machine. To accomplish the purpose of automated data importation, the AWS lambda function is used to serve the script that makes API calls to the Fitbit server for each registered and synced user, to obtain daily health data, and stores it safely in the AWS S3 bucket. These data are used by the machine learning model to calculate the user's health score. The AWS EC2 is used to host the python flask app API to provide better computing power. Finally, the web app is hosted on Google’s free firebase hosting service.

CHAPTER 3

SYSTEM COMPONENTS

There are three major components in the system: Mobile App, Web App, and Machine Learning Module. Each component has its own significance added to the project. This chapter will briefly explore the purpose of each component with the screenshots of them in action.

3.1 Mobile Application

Mobile application is one of the most important factors of the system. It is meant to serve all general-purpose activities. Users should be able to signup, login, view their profile details, update the profile, check their health scores, see historical health data, and reset passwords, etc. The mobile application also lets users sync our system with the Fitbit server and store the returned access tokens and refresh tokens in the database. As explained in chapter 2, it is developed using React-Native framework, which makes it compatible with both android as well as iOS mobile phones.

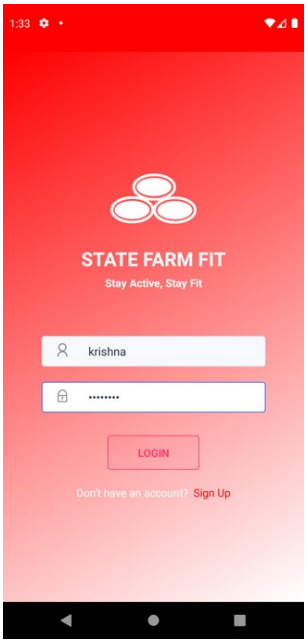

Figure 3.1: Mobile App Login UI

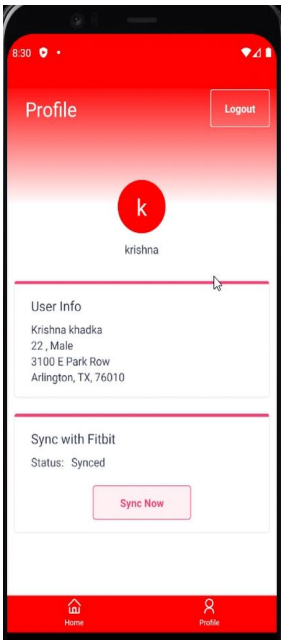

Figure 3.2: Mobile App Profile Page UI

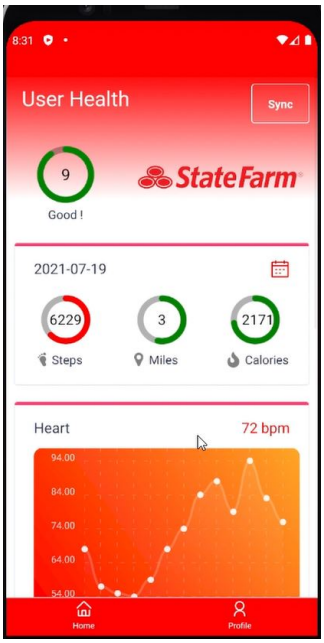

Figure 3.3: Mobile App Dashboard UI

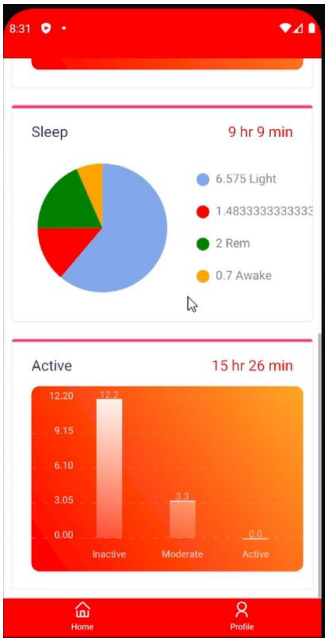

Figure 3.4: Mobile App Graphical UI

3.2 Web Application

The web application is another important component of the system. It has all the features/use-cases that the mobile application has. The only difference between the web application and mobile application is that the web application is accessible through any computing device that supports the modern web browsers, whereas the mobile application can only be run on mobile devices (Android and iOS). As mentioned in Chapter 2, it is developed using the ReactJS library as well as the Bootstrap 5.0 CSS framework. The web application uses web API, developed in Python's Flask Framework, running on AWS EC2, to reduce the data accessing and processing load on the client-side, and, thus, increase the application's performance and security. The client-side of the web application makes calls to the server-side flask application's endpoints to obtain formatted health data and other user details stored in the remote database and storage. Some of the screenshots of working web application are:

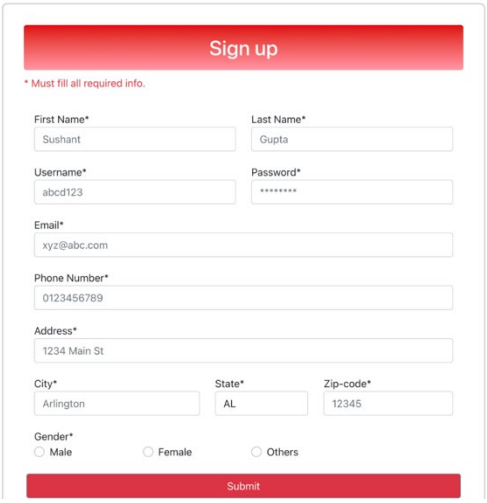

Figure 3.5: Web App Sign-up UI

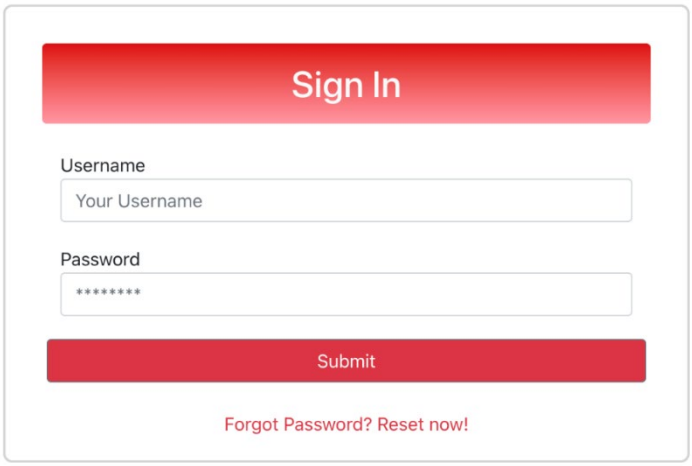

Figure 3.6: Web App Sign-in UI

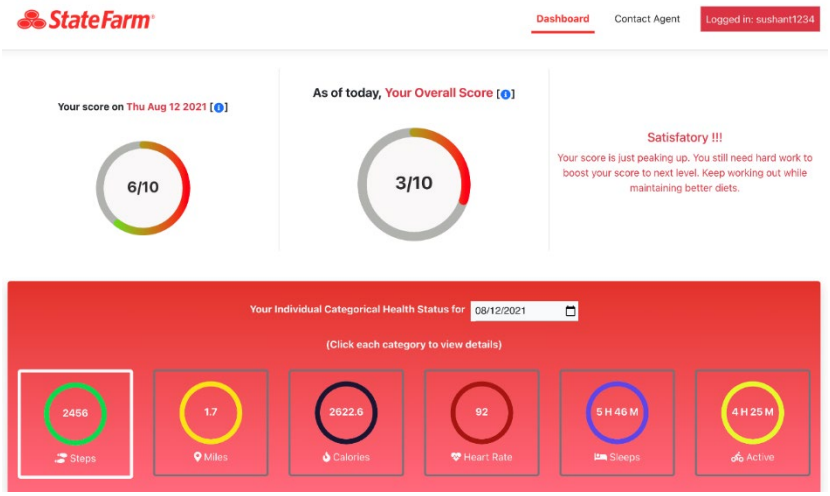

Figure 3.7: Web App Dashboard UI

3.3 Machine Learning Module

Finally, the last but the most important component of the system is the machine learning module. Machine learning is used to calculate health scores for individual users on a daily basis. To achieve a convincing accuracy rate, various classifiers and models were trained and tested. All three types of machine learning techniques (Supervised Learning,

Unsupervised Learning, and Reinforcement Learning) were explored and tested. The K-means clustering, an unsupervised machine learning technique, was used to find the clusters and the centroid of the clusters within the health data. The centroid was used to populate synthetic data, which is then used to train the model. After training and testing various ML models, it is observed that the Random Forest Classifier (RFC) has the highest accuracy rate, almost 80% on average, for most of the classifications. The next closest model with a better accuracy rate was Bagging Classifier.

The daily score, calculated using the machine learning technique, is then used to calculate the final aggregate score using Exponential Moving Average (EMA) algorithm. EMA is a type of moving averages that places greater weight and significance on the most recent data points (Chen). The formula used to calculate the EMA is as follow:

$$EMA_{Today} = \left(Score_{Today} * \left(\frac{Smoothing}{1 + No.of\ Days} \right) \right) + \left(EMA_{yesterday} * \left(1 - \left(\frac{Smoothing}{1 + No.of\ Days} \right) \right) \right)$$

where,

Smoothing = 2.0

Therefore, the final aggregate score represents the average score of all past daily scores with greater weight on the most recent health data. This score is then used by Statefarm Insurance Company to provide life and/or health insurance quotes to the users.

```
[ ] rfc = RandomForestClassifier()

[ ] rfc.fit(X_train,y_train)

[ ] rfc_acc_train = rfc.score(X_train,y_train)
    print("Random Forest Training accuracy",rfc_acc_train)

[ ] rfc_pred = rfc.predict(X_test)

[ ] from sklearn import metrics
    mae_rfc = (metrics.mean_absolute_error(y_test, rfc_pred))
    mse_rfc = (metrics.mean_squared_error(y_test, rfc_pred))

    print("Mean Absolute Error : ",mae_rfc)
    print("Mean Sqaured Error : ",mse_rfc)

    Mean Absolute Error :  0.09863333333333334
    Mean Sqaured Error :  0.09876666666666667

[ ] import numpy as np

    def mean_absolute_percentage_error(y_test, rfc_pred):
        y_test, y_pred = np.array(y_test), np.array(rfc_pred)
        return np.mean(np.abs((y_test - rfc_pred) / y_test)) * 100

    mean_absolute_percentage_error(y_test,rfc_pred)

    3.6281190476190472

[ ] rfc_pred_score = (rfc.score(X_test,y_test))
    rfc_pred_score

    0.9014333333333333
```

Figure 3.8: RFC Model in Action

CHAPTER 4

AUTOMATION AND API

4.1 Task Automation

The focus throughout the duration of the project was to automate several tasks in the system such as data importation from the Fitbit server to the AWS S3 bucket and the calculation of users' health scores using their daily health data stored in the S3 bucket. This chapter will explore the implementation of such tasks in detail below.

4.1.1 Automated Fitbit Data Importation

The system is designed in such a way that every user who owns a Fitbit watch can sync their watch by logging into our mobile or web app. Anyone syncing their Fitbit watch to the app is giving permission to the system to collect their daily health data through series of API calls provided by Fitbit. After successful syncing, access tokens and refresh tokens provided by the Fitbit server, are stored in AWS DynamoDB for each user. These tokens are used later to collect all user's data.

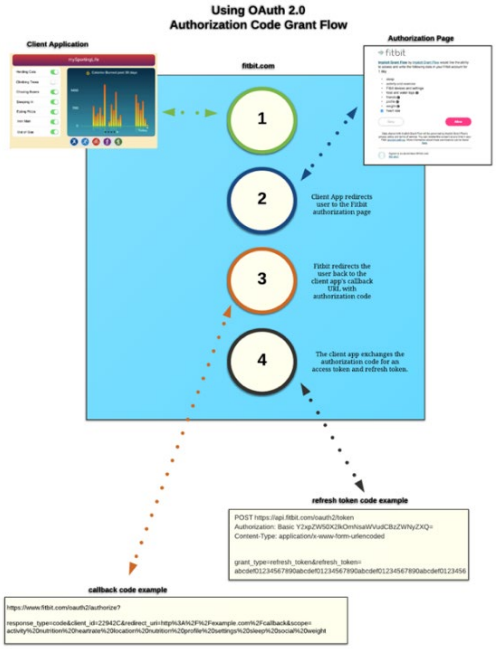

Figure 4.1: Fitbit Authentication Flow (Authorization Code Flow)

As shown in Figure 4.1, a valid access token is needed to make a successful API call to the Fitbit server. An access token is only valid for 8 hours after its generation. A valid refresh token is used to update the access token for each user. A python script is developed, which retrieves the list of every user's access tokens, refresh tokens, and user id. The script then iterates over the list and makes several API calls to the Fitbit server, using the access token and its associated user-id, to get all kinds of health data for that user. Then, it formats the data properly in a .csv file and sends the file to the AWS S3 bucket after giving a proper name to make it easy to retrieve it later when needed. Using the refresh token, the script also updates the access token, if expired, in the DynamoDB. The script is hosted on AWS Lambda with the CRON job scheduled at midnight so that the complete data for the day can be accessed and retrieved from the Fitbit server. AWS Cloud Watch is used to perform the CRON job at the scheduled time every day.

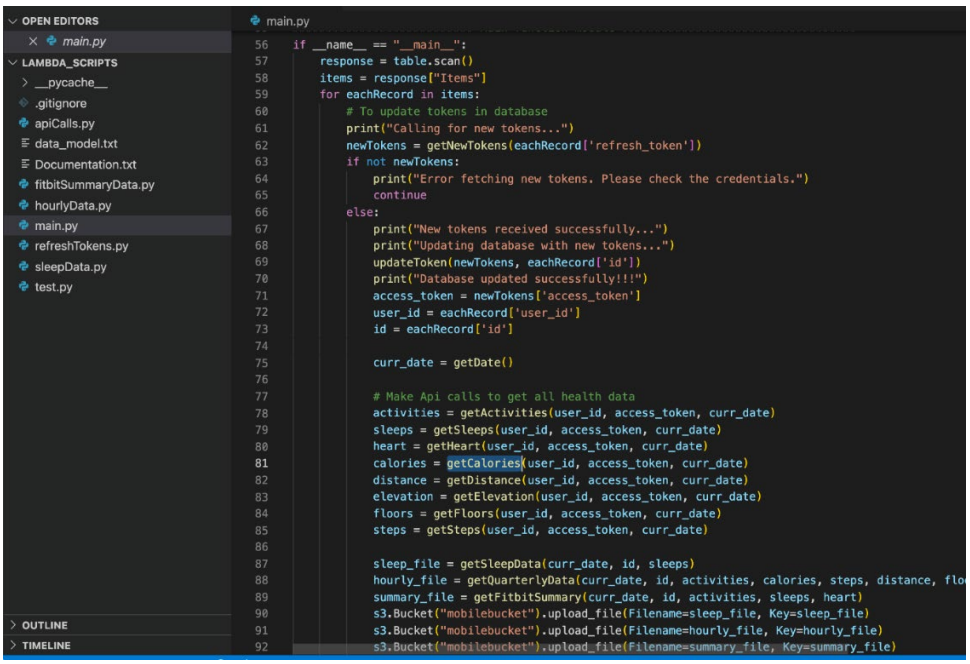

Figure 4.2: Automation Code Snippet

4.1.2 Automated Score Calculation

The selected machine learning model is used in the automated script to calculate the score automatically at a specified point of the day. Since all the resources, such as DynamoDB and S3 bucket storage, which contains user ids and health data are contained in the AWS cloud, it was decided to keep the score calculation process on the AWS cloud too. That is why AWS Sage-maker studio and Notebook instances are used to run the model on the user's daily data and predict their health score. The AWS Cloud-watch is again used to execute the CRON job established on the AWS Lambda function. Here, the lambda function initiates the Notebook instance and runs the ML model on every user's health data for that day to calculate their scores and update the score in a .csv file that contains the score history for the user.

4.2 Web APIs for Data Handling

Using Python-based Flask Framework, a server-side application is developed. This application consists of several routes, which are used by the web application to get different data for the users. The following table shows the provided routes and their respective return values in the application.

Table 4.1: Routes for Web API

Routes	What does it return?
'/getDailyTotal/<string:uid>/<string:date>'	This returns the total values of each health metric of the specified user for the specified date.
'/getGraphData/<string:uid>/<string:date>'	This returns the hourly data for each heath metric of the specified user for the specified date.
'/getSleepsData/<string:uid>/<string:date>'	This returns the sleep related data of the specified user for the specified date.
'/getScoreHistory/<string:uid>'	This returns the score history of the specified user.

The app gets an HTTP request from the web application for a specific type of data, and then it fetches the data from the AWS S3 bucket, formats the data in a proper JSON object, and finally sends it to the web application for an easy and fast rendering in the client-side application. The main purpose of these Web APIs is to reduce the load of the web app and, thus, enhance the performance. The flask app is currently hosted on the AWS EC2 instance, which provides public endpoints that can be requested with proper header and security credentials from any client-side application.

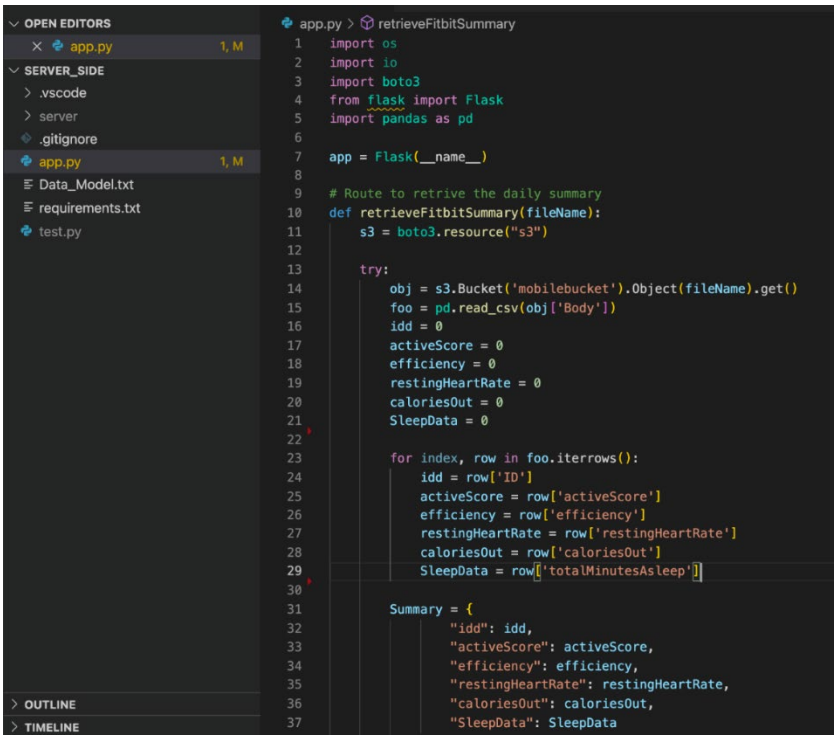

Figure 4.3: Web APIs Code Snippet

CHAPTER 5

CONCLUSION

The main purpose of this project was to understand and test the system of analyzing a person's health data to predict their future health status. The future health status is assumed based on a score that is calculated using machine learning techniques using the user's prior health data. The score is used to provide Statefarm's health and/or life insurance quotes to its customers. A partially working system was handed to the team to fix the issues and improve the system. After an initial assessment of the existing system, a completely new system architecture was proposed that has far better performance and security compared to the old one. The implementation of the AWS Cognito and Identity pool, along with the AWS DynamoDB, made the system more secure by providing persistence and encrypted user authentication data flow between various components of the system.

Similarly, the addition of the automation of Fitbit data collection was another great improvement in the system. This allowed the system to have the complete persistence health data of every user without the need for any personal interaction. It also helps to prevent the possible fraud whereby users hide their certain day data when they do not perform enough exercise to maintain good health. The AWS Lambda service made this automation possible, and the AWS Sage-maker made the score calculation automated. Again, the development of web API increased the performance of the web application by a huge factor. Finally, this project presented a lot of opportunities to learn several new

things. It also revealed what a real-world development project looks like, and how one can lead a team and handle the pressure of multiple work at the same time.

REFERENCES

"Authorization Code Flow." *Accessing the Fitbit API,* dev.fitbit.com/images/reference/web-api/Authorization_Code_Flow-e38dbc200462c4364886550be7133510.png.

Chen, James. "Exponential Moving Average (Ema)." Investopedia, Investopedia, 19 May 2021, www.investopedia.com/terms/e/ema.asp.

BIOGRAPHICAL INFORMATION

Sushant Gupta is currently a graduating senior at the University of Texas at Arlington and an application development intern at Triencon Services Inc., Arlington, TX. He is acquiring his Honors Bachelor of Science in Computer Science with a minor in Mathematics. He already has an Associate of Science degree with a field of study in Computer Science from Dallas College. Sushant is highly interested in the research of software development, mostly related to the medical and health care system. He is also very good at web development and writing solution patches to improve an existing system. He has strong leadership skills too, which he has demonstrated being a project lead and involved in the student body senate at UTA. Sushant loves working on a variety of projects. He has worked a couple of on-campus jobs, one being a math tutor and another a research assistant. He has also done a couple of internships so far. During his first internship, he helped to develop a python-based application that manages huge bidding data and analyzes them to predict the range of bid value of any similar open contract in that location. This application can be used by the bidding companies to increase their bid winning odds and, thus, making more profit from the contract. At the second internship, he built an IT Monitoring web application using ASP.NET and ReactJS. One can find the full list of the projects he has worked on in his GitHub repo at www.github.com/sushantcode. Sushant is expecting to graduate in the Fall 2021 with Summa Cum Laude and is planning to look for a job as a full-time software developer at a company.