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

Detailed Design Specification
CSE 4317: Senior Design II
Summer 2020

Team Ingenuity
State Farm Fit

Chelsea May
Wei Shi
Devi Tripathy
Sudeep Bhadel

Revision History

Revision	Date	Author(s)	Description
0.1	06.25.2020	CM, WS, DT, SB	document creation
0.2	07.04.2020	CM, WS, DT, SB	complete draft

CONTENTS

List of Figures

List of Tables

1 Introduction

Our mobile application will be similar to a fitness app and will extract data from a Fitbit in order to track the user's health information. The user should be able to log-in, view their health information, as well as their current fitness score. The health information that will be recorded will include the person's heart rate, steps, sleep, driving habits, and how often they exercise/calories burned. Our application will also have GPS capabilities in order to track the user's location. The requirement specification and architectural design documents provide more information on the system's features and functions, intended audience, etc. upon completion of the project.

2 System Overview

Our system consists of 3 layers that interact with one another: the presentation layer, business layer, and data access layer. The presentation layer will include the android interface that the user would interact with as well as a web UI for an administrator to interact with. The business layer will include AWS, from which we will be using many different services. The business layer will also be where the data analysis is done. The data access layer will store the data that will be collected and provide access to the data. This layer will be composed of 4 subsystems: User, Administrator, Raw data, and Analyzed data. It will receive data from the business layer and also send data to be stored in the DynamoDB. More information on each layer of our system can be found in the architectural design document.

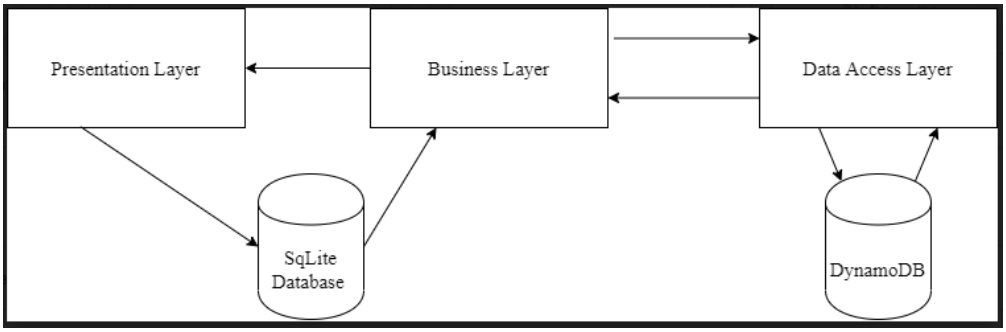

Figure 1: A simple architectural layer diagram

3 PRESENTATION LAYER SUBSYSTEMS

Presentation layer includes three isolated subsystems which run on a android device or an end system.The capture subsystem response for collecting user Location, heart rate, activities, and sleep data from aFitbit smart watch. The android user-interface subsystem will allow user to Login, register , and modifytheir account, get a quote from the insurance company, and review his or her activities on their mobiledevice when the device is online. They also are allowed use a website to access the data.

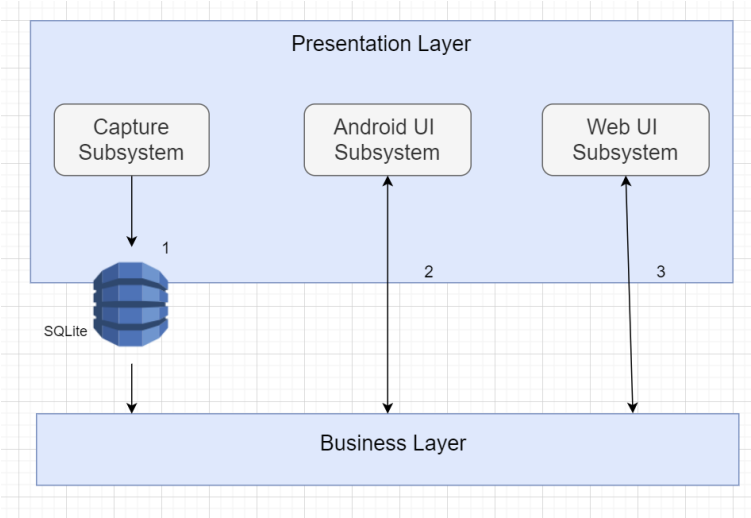

Figure 2: Presentation layer

3.1 LAYER HARDWARE

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

3.2 LAYER OPERATING SYSTEM

There is no common operating system be used for the layer level.

3.3 LAYER SOFTWARE DEPENDENCIES

There is no common software be used for the layer level.

3.4 CAPTURE SUBSYSTEM

Capture subsystem will run on a mobile device to capture important data from user though Fitbit API, include heart rate, location, activities, and sleep data. The raw data will be stored at a local database SQlite, a subsystem in the business layer will upload and update the data between local database and mongo DB.

3.4.1 SUBSYSTEM HARDWARE

1. a Fitbit smart watch which can collect important health data from the users.
2. an Android device which can run the mobile app and connect with a Fitbit smart watch.

3.4.2 SUBSYSTEM OPERATING SYSTEM

1. Any Android OS higher than Android 7.0 Nougat (API 24)

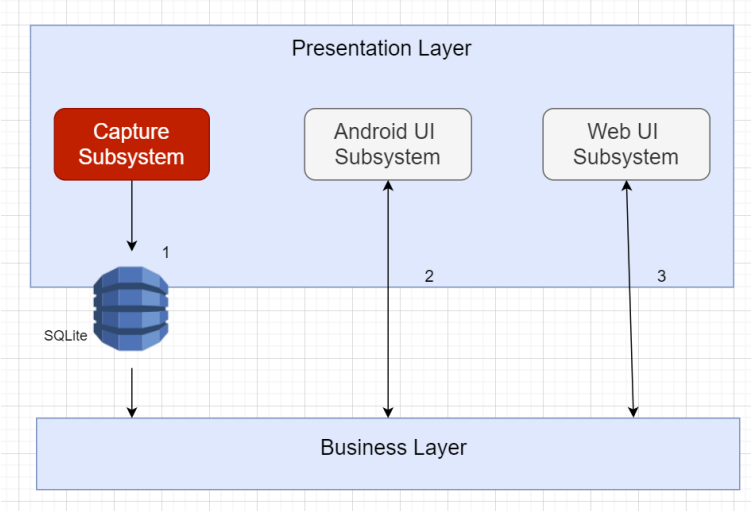

Figure 3: Capture Subsystem

3.4.3 Subsystem Software Dependencies

1. the Fitbit SDK. An API can help us to connect to Fitbit watch. 2. JDBC, a package helps java developer to interact with SQlite.

3.4.4 Subsystem Programming Languages

Java and Android Studio.

3.5 Android UI Subsystem

Android UI subsystem will run on a mobile device to allow user to review their health data, manage their account, and acquire a quote from the insurance company. The quote will based on the regular quote and the user's fitness level in the database.

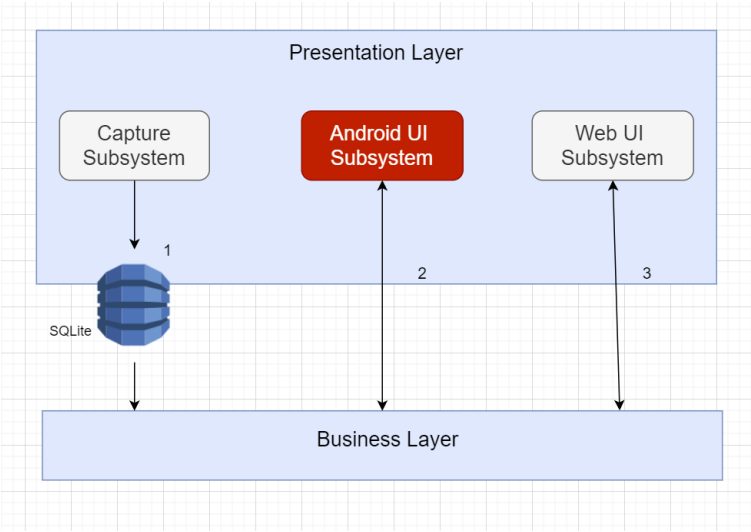

Figure 4: Android UI Subsystem

3.5.1 Subsystem Hardware

1. an Android device which can run the mobile app.

3.5.2 Subsystem Operating System

1. Any Android OS higher than Android 7.0 Nougat (API 24)

3.5.3 Subsystem Software Dependencies

1. SQlite, a local database.
2. JDBC, a package helps java developer to interact with SQlite.

3.5.4 Subsystem Programming Languages

Java and Android Studio.

3.5.5 Subsystem use case

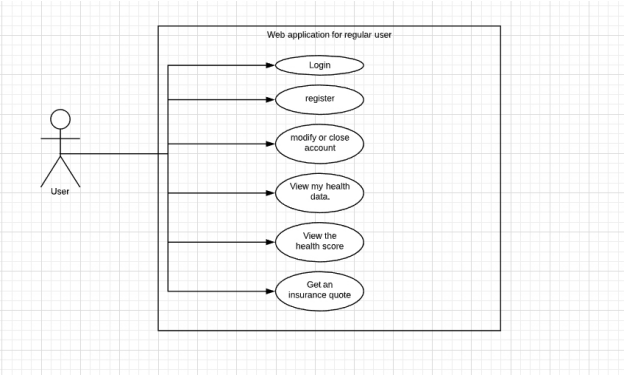

Figure 5: Mobile UI use case diagram

3.6 Web UI Subsystem

Web UI subsystem provide user another way to access their account. Also, from the web UI, administrators will be able to view user health information and insurance rates, as well as their own account information.

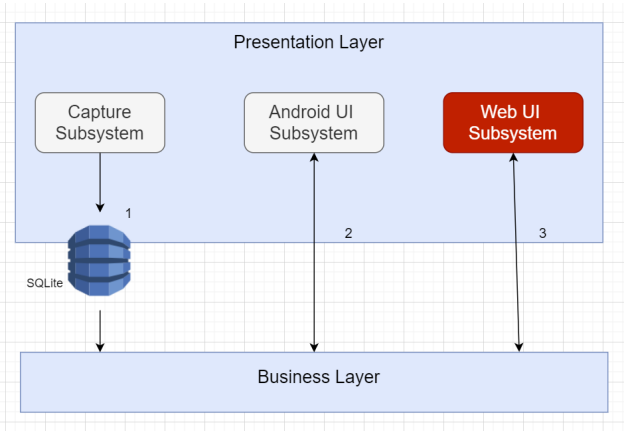

Figure 6: Web UI Subsystem

3.6.1 Subsystem Hardware

1. a computer can run a browser.

3.6.2 Subsystem Operating System

1. Windows 10.
2. Linux.
3. macOS.

3.6.3 Subsystem Software Dependencies

None

3.6.4 Subsystem Programming Languages

1. Python.
2. HTTP+CCS+Bootstrap

3.6.5 Subsystem use case diagram

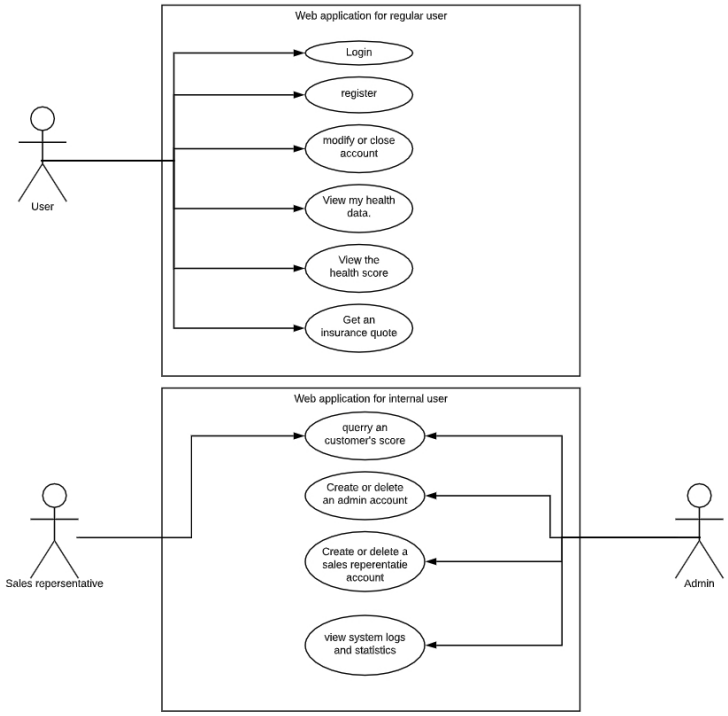

Figure 7: Web UI use case diagram

3.6.6 SUBSYSTEM WEBSITE MAP

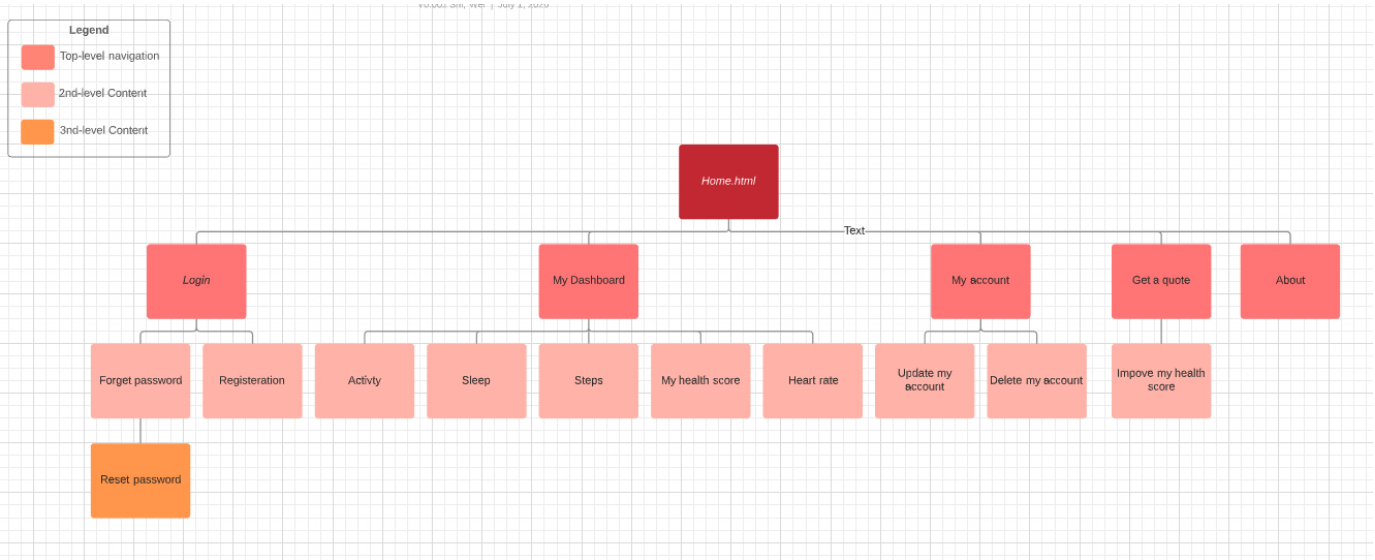

Figure 8: Web UI website map

4 DATA ACCESS LAYER SUBSYSTEMS

The data access layer is integrated onto the cloud service, Amazon Web Service (AWS). For the database, Amazon DynamoDB is used and is accessed via various APIs which is written in python using NoSQl query language or more specifically DynamoDB query language.

4.1 LAYER HARDWARE

There is no hardware being used if we view this as an physical perspective, but taking into consideration that AWS is being used we need to launch Virtual Machine as Amazon EC2.

4.2 LAYER OPERATING SYSTEM

Amazon Linux

4.3 LAYER SOFTWARE DEPENDENCIES

This layer will use Amazon DynamoDB for database. Package com.amazonaws.services.dynamodbv2 has all the required methods.

4.4 USER SUBSYSTEM

User layer is where the credentials i.e. user id and password of the user is verified, and fetch data on user request operated through a running instance on AWS accessing Amazon DynamoDB.

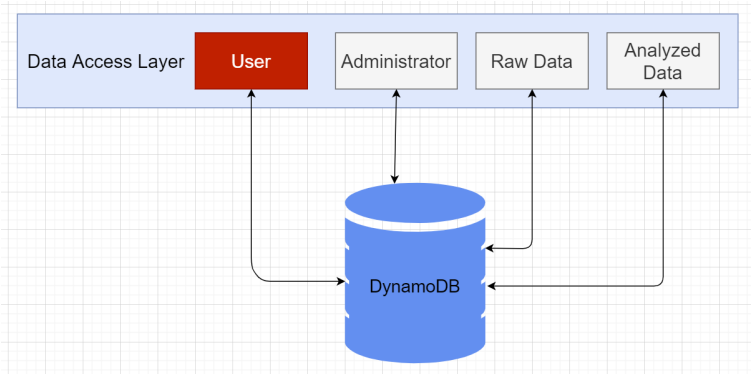

Figure 9: User Subsystem

4.4.1 SUBSYSTEM HARDWARE

No specific hardware

4.4.2 SUBSYSTEM OPERATING SYSTEM

No specific OS

4.4.3 SUBSYSTEM SOFTWARE DEPENDENCIES

Package com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient has all the required methods.

4.4.4 SUBSYSTEM PROGRAMMING LANGUAGES

Python, DynamoDB query language

4.4.5 SUBSYSTEM DATA STRUCTURES

The data of the user is stored in the Amazon DynamoDB. Amazon DynamoDB stores data in partition, each partition is allocted in a SSD for a table which is replicated throughout the AWS region.

4.5 Administrator Subsystem

This subsystem gets the admin login credentials and admin requests with the help of Web UI Subsystem in the presentation layer, then a subsystem in business layer transfers data to data access layer operated through a running instance on AWS accessing Amazon DynamoDB.

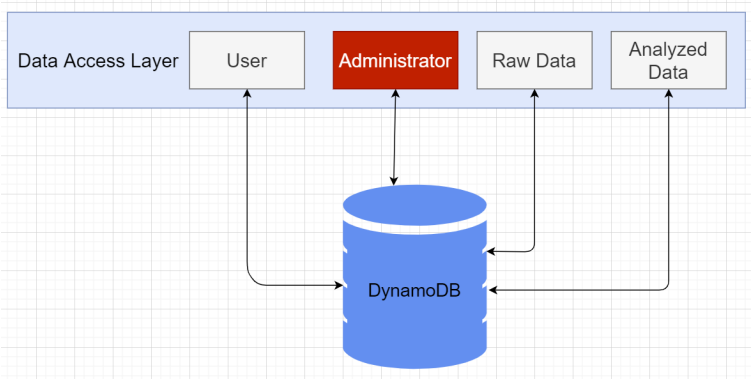

Figure 10: Administrator Subsystem

4.5.1 Subsystem Hardware

No Specific hardware

4.5.2 Subsystem Operating System

No Specific OS

4.5.3 Subsystem Software Dependencies

Package com.amazonaws.services.dynamodbv2.AmazonDynamoDB has all the required methods.

4.5.4 Subsystem Programming Languages

Python, DynamoDB query language

4.5.5 Subsystem Data Structures

The data of the admin is stored in the Amazon DynamoDB. Amazon DynamoDB stores data in partition, each partition is allocted in a SSD for a table which is replicated throughout the AWS region.

4.6 Raw data Subsystem

This subsystem simply communicates the raw data captured from the FitBit watch in presentation layer streamed through AWS Kinesis to multiple destination for Data analysis and Data presentation.

4.6.1 Subsystem Hardware

No specific hardware

4.6.2 Subsystem Operating System

No specific subsystem

4.6.3 Subsystem Software Dependencies

DynamoDB package: Package com.amazonaws.services.dynamodbv2.AmazonDynamoDB AWS Kinesis (Python Package): kcl and amazon_kclpy AWS Kinesis (Java Package): amazon-kinesis-client AWS CLI

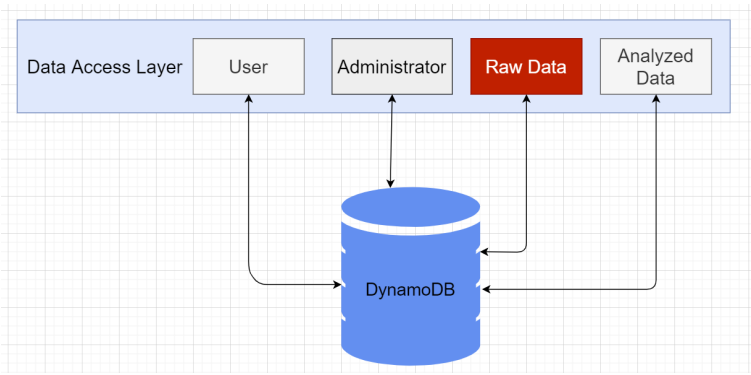

Figure 11: Raw Data Subsystem

4.6.4 Subsystem Programming Languages

Python(Backend), Java(Android) DynamoDB query language

4.6.5 Subsystem Data Structures

The raw data is stored in the Amazon DynamoDB. Amazon DynamoDB stores data in partition, each partition is allocted in a SSD for a table which is replicated throughout the AWS region.

4.7 Analyzed data Subsystem

The main purpose of this subsystem is to store and retrieve analyzed data such as, fraud activity, fitness score and risk factor from a subsystem of the business layer to Amazon DynamoDB.

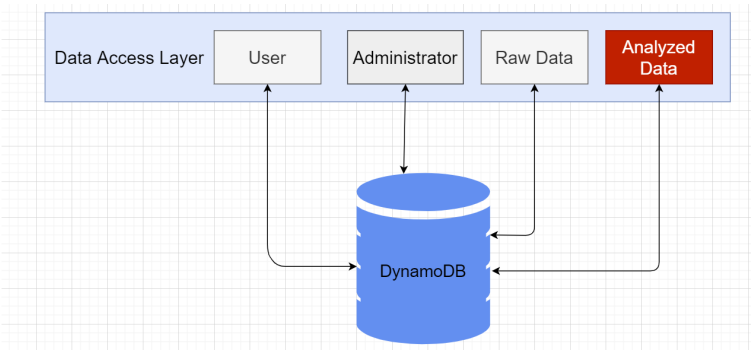

Figure 12: Analyzed Data Subsystem

4.7.1 Subsystem Hardware

No specific hardware

4.7.2 Subsystem Operating System

Ubuntu Docker Container, Kubernetes deployed via KubeFlow containing End-to-End Machince learning logic

4.7.3 Subsystem Software Dependencies

Docker used to for modularized code base Kubernetes will consume docker containers for scalability purposes KubeFlow will encapsulate Kubernetes instance for each module to a CI/CD pipeline Package com.amazonaws.services.dynamodbv2.AmazonDynamoDB has all the required methods.

4.7.4 Subsystem Programming Languages

Python, DynamoDB query language

4.7.5 Subsystem Data Structures

The analyzed data is stored in the Amazon DynamoDB. Amazon DynamoDB stores data in partition, each partition is allocted in a SSD for a table which is replicated throughout the AWS region.

5 Business Layer

This subsystem is the brain of the system and is responsible for delegating and running processes to perform Data analysis, Data cleaning, API endpoint processes. This layers will listen to incoming transaction from presentation layer and in response will send appropriate data.

5.1 Layer Hardware

This layer does not specifically involves any hardware. All the hardware are hosted on AWS as EC2 instances using AWS Elastic Beanstalk (Kubernetes).

5.2 Layer Operating System

System is served using Ubuntu 18.04

5.3 Layer Software Dependencies

This layer depends on various software packages. API endpoint and End-to-End Machine Learning algorithm is on Python 3.7, Keras Framework is used to develop and train ML model. Kubernetes, Docker, Kubeflow and software pacakges and frameworks used to build and sever business logic at scale.

5.4 EC2/Kubernetees

The purpose of this subsystem is to host business logic at scale, by deploying Kubernetees cluster via KubeFlow this allows customer (Product owner) to managed their resources nicely.

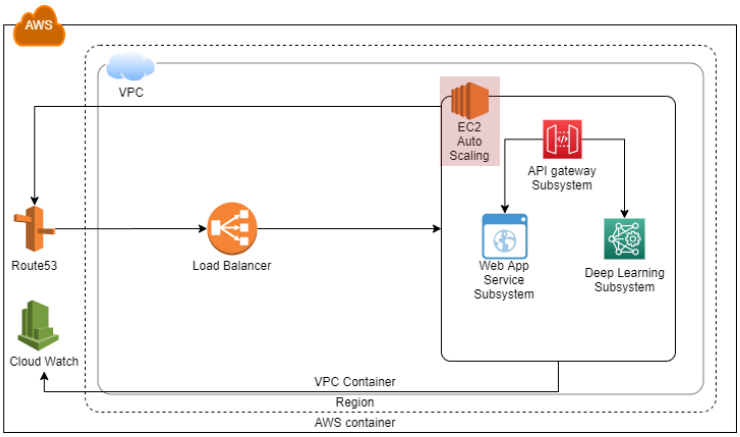

Figure 13: EC2: Business Layer

5.4.1 Subsystem Hardware

t2.medium AWS EC2 machine with 2 replicas.

5.4.2 Subsystem Operating System

Ubuntu 18.04 is running across all the replicas.

5.4.3 Subsystem Software Dependencies

Docker, Kunbernetees, KubeFlow software packages that the system is depended upon.

5.4.4 Subsystem Programming Languages

This is an interface with no programming required.

5.4.5 Subsystem Data Structures

No specific Data Structures

5.4.6 Subsystem Data Processing

All error logging and Data flow in this pipeline will be managed my Python endpoints.

5.5 Data Analysis Subsystem

This subsystem is responsible for handling task delegated KubeFlow API endpoint to change/create a Deep Learning based user bio profile model.

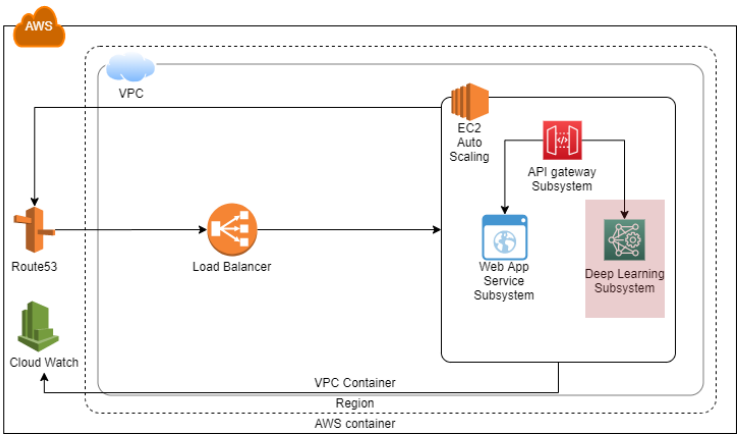

Figure 14: Data Analysis: Business Layer

5.5.1 Subsystem Hardware

No specific hardware

5.5.2 Subsystem Operating System

Docker container using Ubuntu 18.04

5.5.3 Subsystem Software Dependencies

Python, Keras, Flask, Redis, Kafka.

5.5.4 Subsystem Programming Languages

Python is primary programming language used for all Data Analysis task.

5.5.5 Subsystem Data Structures

Subsystem involves various different data structures some of them are listed below: 1. Tensor 2. Dictionary 3. Parquet 4. Redis Object

5.5.6 Subsystem Data Processing

Data is processed using a custom implementation of Transformer (DL architecture) and DLRM recommendation engine. Transformer architecture is a self-attention encoder based Deep learning architecture, traditionally used for NLP task and is currently State of the art is most NLP task, but is touted to be a great solution for forecasting problem. This project includes a custom implemented Transformer model which is solely responsible to forecast user behaviour pattern. DLRM is a recommendation engine which is used by Facebook across all of it product, this is used predict user behaviour and influence/bias forecasting model to similar data set

5.6 API Gateway

This responsible to allow access to the various Views/FrontEnd. Secured with TLS.

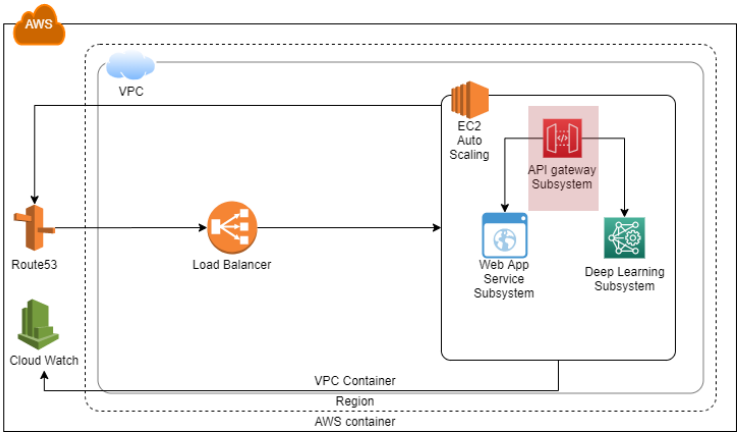

Figure 15: API: Business Layer

5.6.1 Subsystem Hardware

No specific hardware

5.6.2 Subsystem Operating System

Ubuntu 18.04

5.6.3 Subsystem Software Dependencies

Python, Flask

5.6.4 Subsystem Programming Languages

Python is the programming language used to implement this subsystem

5.6.5 Subsystem Data Structures

JSON data structures is used to make inter communication with clinets possible

5.6.6 Subsystem Data Processing

No specific data processing is done. Only data that is being processed is the data received from API calls.

5.7 Web App Service

This service is responsible for Admin portal access it is aimed help Data Scientists to see and visualize entire pipeline.

5.7.1 Subsystem Hardware

Hosted on AWS EC2 instance

5.7.2 Subsystem Operating System

Ubuntu 18.04 is operating system that is being used

5.7.3 Subsystem Software Dependencies

Python, HTML, Bootstrap, flask.

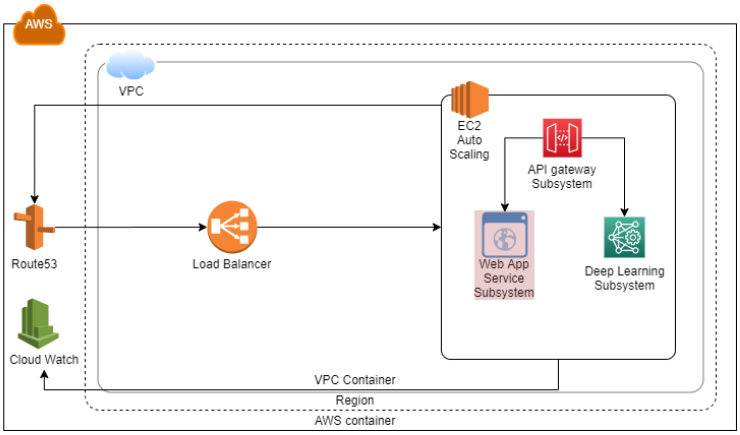

Figure 16: Web App: Business Layer

5.7.4 Subsystem Programming Languages

Python, HTML, CS and JavaScript are the programming languages that are being used.

5.7.5 Subsystem Data Structures

JSON Data Structure is being used to ingest and send data to API gateway

5.7.6 Subsystem Data Processing

No data is being processed only being displayed.

6 APPENDIX A

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

REFERENCES