



**UNSW**  
A U S T R A L I A

**COMP6733 - IoT Design Studio**  
**A Robust Step Counter**  
**T3, 2019**

**Submitted by**  
**Team – J.A.R.V.I.S.**

Ajinkya Biswas	<a href="mailto:z5196315@student.unsw.edu.au">z5196315@student.unsw.edu.au</a>
Daniel Hocking	<a href="mailto:z5184128@student.unsw.edu.au">z5184128@student.unsw.edu.au</a>
Suchithra Suchithra	<a href="mailto:z5230257@student.unsw.edu.au">z5230257@student.unsw.edu.au</a>

## Contents

Background	3
Current Devices	3
Scope	3
Solution Overview	4
1. SensorTag	4
2. Android App	5
3. Backend Service	5
4. Machine Learning Model	5
Implementation	6
SensorTag Details	6
Android App Details	6
AWS Backend	8
Machine Learning	12
Value Addition	16
Future Improvement Scope	16
Mobile App	16
AWS Backend	16
Machine Learning	16
Energy Saving	17
Conclusion	17
References	18
Appendix	19
A. Energy conservation data from Android app	19
B. AWS Setup – API Gateway	20
C. AWS Setup – Lambda Function	20
D. AWS Setup – Kibana	20
E. AWS Setup - DynamoDB	21
F. AWS Setup – EC2	22
G. AWS Setup – Elasticsearch	22
H. DynamoDB Table – Android_user_credentials	22
I. DynamoDB Table – Sensor_acc_data	22
J. DynamoDB Table – Steps_count	23
K. DynamoDB Table – Training_data	23

## Background

Healthcare is one of the basic necessities. With the advancement and innovation in medical science, people are receiving better healthcare and life expectancy is increasing around the world. With increasing life expectancy and an aging population, healthcare costs are also increasing significantly. In 2018-19, Australia's total Medicare cost was \$24.1 Billion and in 2017-18, the cost was \$23.2 billion. That's an almost \$1 billion increase in just one year.

Healthier lifestyle reduces healthcare cost significantly. Therefore, many governments and insurance companies around the world are encouraging people to have an active lifestyle. For example, the Queensland Government provides CQUniversity with the funding to run the 10,000 steps program. The program encourages people to take 10,000 steps every day. Bupa, a major insurance provider has a healthier workplace program which encourages people to participate in fitness programs using Garmin, Apple Watch, Fitbit and Smart Phones.

While the statistics are not difficult to obtain, a dishonest user may attempt to cheat the system by launching different attacks (e.g., data replay attacks, data modification attacks, Sybil attacks etc.). Although these attacks need some reverse engineering of the fitness devices, there are simpler ways that a normal user may try to cheat the system. For example, getting something else to generate steps, or one user carrying multiple devices to generate steps for multiple people. Or, swinging the device in a simple harmonic motion using a thread.

## Current Devices

When we walk, we swing a leg forward and our body tilts to one side. Then our body tilts the other way and we swing the other leg forward too. Each tilt is detected by the MEMS inertial sensor inside pedometers and that is the step count. Under normal circumstances these devices work properly. Drawback of these devices is that user can mimic walk like movements (like a pendulum), these devices sensors keep counting the number of tilts and it increases step count.

## Scope

A person's motion should be tracked by a wearable IoT device with the following properties:		
Required	Description	Achieved
Y	Inexpensive (manufacturing cost < \$50 / unit)	Y, the main component of the solution is a SensorTag CC2650 which costs \$45 AUD / device, less if buying larger quantities ( <a href="https://www.ti.com/store/ti/en/p/product/?p=CC2650STK&amp;HQS=corp-uni-null-hackster-cs-storeevm-CC2650STK-wwc">https://www.ti.com/store/ti/en/p/product/?p=CC2650STK&amp;HQS=corp-uni-null-hackster-cs-storeevm-CC2650STK-wwc</a> )
Y	Compact (no larger than comparable wearable devices, eg. smart watch)	Y
N	Battery life (should last for at least 7 days between charge)	N, due to the SensorTag firmware being used this is not possible
N	Durable (not easily damaged by everyday use, last for at least 12 months continuous use, water resistant)	N, would need to make a custom case to provide this level of protection
N	Comfortable (pleasant to wear, like a watch)	Y
N	Aesthetic (appearance should be appealing)	N

An Android smartphone app will be created with the following properties:		
Required	Description	Achieved
Y	Ability to register a profile in the system (email, name, password)	Y
Y	Ability to pair a wearable device with account and allow syncing to occur	Y
Y	App can take data received from wearable device and send it to backend system for processing	Y
Y	A simple dashboard will display progress towards goals: steps, exercise time, movement / hour	Y, it shows steps over time, broken down by hour, doesn't track exercise time though
A backend system will be setup with the following properties:		
Required	Description	Achieved
Y	Ability to receive data from Android app and process it	Y
Y	Processing should allow for the following: Detect steps, Identify who is currently wearing the device based on gait and see if it matches the profile previously built, Ensure that steps are genuine and not duplicate	Y
General requirements of the system:		
Required	Description	Achieved
N	End to end security starting with device based encryption	Y/N, encryption takes place within the Android app rather than from the device

## Solution Overview

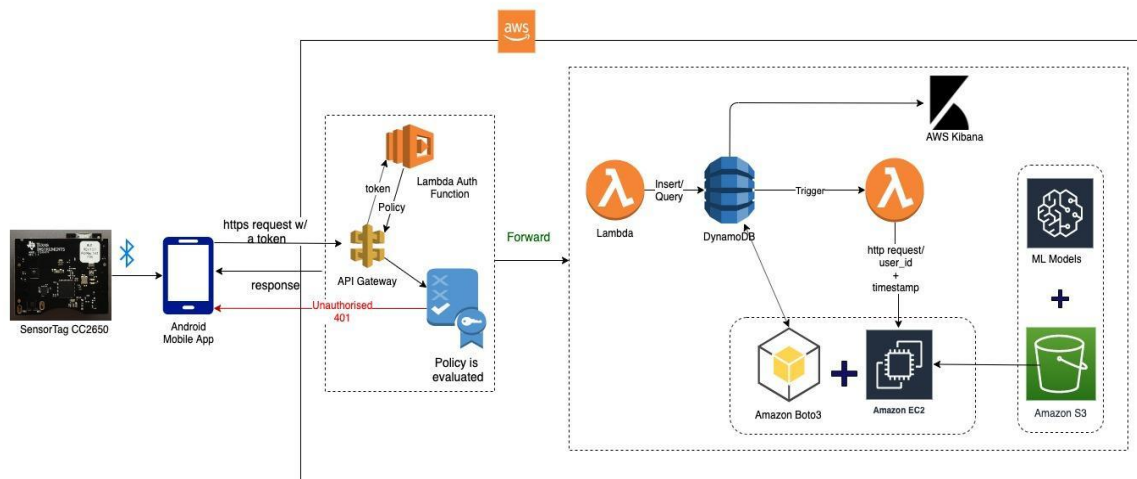


Figure 1: Solution Architecture

The solution has four main components:

### 1. SensorTag

Hardware: CC2650 SensorTag, battery and wristband

Software: SensorTag is programmed with TI-RTOS

Integration: It is integrated with Android app using the BLE stack of TI-RTOS

Functionality: It is used to capture the motion of a person. This data will be used to analyse the walking pattern and to count the steps. The motion is tracked by the 3-axis accelerometer in a MPU-9250 that is present in the CC2650. There is no data processing done on the sensor. The raw x, y, z values are sent to the android mobile application in real time.

## 2. Android App

Hardware: Smartphone

Software: Android OS

Integration: It is integrated with sensor tag using BLE and with backend service via a REST API

Functionality: It reads the accelerometer data from the sensor at a specified interval of time and sends it to the backend for processing. The app scans all the devices that are bluetooth enabled. The user needs to select his CC2650 sensor from the list. Once connected, the user needs to press the start recordings button on the app before he walks, the Android app collects accelerometer data provided by the SensorTag as they walk. Periodically the sensor data is sent to the backend, this will occur roughly every 30 seconds if the user is currently active. The dashboard will update regularly to show the current count of validated steps broken down by hour of the day. Also, to reduce the SensorTag battery usage, If there is not much variation in the walking pattern for a continuous period of time, then the frequency at which the sensor data is sampled will be reduced.

## 3. Backend Service

Platform: AWS Cloud

Integration: It is integrated with mobile app via API gateway

AWS Services: API Gateway, Lambda, DynamoDB, EC2, S3, IAM, Kibana

Functionality: AWS API receives, saves and processes the sensor data

API Gateway: acts as an entry point for the https request that comes from the mobile app. The app calls the post method passing a token in the header along with the sensor data that is encrypted using the shared secret key which was sent to the app when the user logged in. API Gateway calls the Lambda authoriser, which takes the caller's identity i.e token and returns the IAM policy. The gateway evaluates the policy and if the access is denied, 401 response is returned to the app. If the call succeeds, it calls the lambda function decrypts the sensor data and saves it into the DynamoDB table sensor\_acc\_data. This table maintains the sensor data and has the partition key as username concatenated with the timestamp. On insertion of data, the trigger activates lambda function that calls EC2 instance that hosts the python code, which does the machine learning part. The data is processed, and the information is updated in the DynamoDB. This data can be queried by mobile app to fetch the step info and is also used in dashboards for data visualisation. AWS also has a user registration and verification API that adds the user into DB when a new user is registered and verified every time when the user logs in the StepCount app.

## 4. Machine Learning Model

### Data Collection

For building an ML model, the SensorTag is collecting 10 samples per second and sending this data to AWS DynamoDB using the mobile app. Raw data has timestamps, x-axis (direction of walking) readings, y-axis (gravitational direction) readings, z-axis (sideways movement) readings. The sensor tag also provides the result of the combination vector, which we used for our final ML models.

### Data Processing

Since raw sensor data has significant noise, we filter out the noisy readings by passing raw data through a low-pass Butterworth filter. The output is a smooth signal whose peaks represent steps. An AWS EC-2 instance is hosting python code for signal processing and machine learning.

## Implementation

### SensorTag Details

- Although the original proposal specified Contiki-OS this is no longer being used on the SensorTag as it doesn't sufficiently support BLE, instead the BLE-Stack [1] provided as part of the TI-RTOS is being used, the specific version being used can be found in the following folder: ble\_sdk\_2\_02\_03\_08/examples/hex/cc2650stk\_sensortag\_rel.hex if the files have been downloaded from the above referenced link
- After the Android app connects to the SensorTag it sends the following Bluetooth GATT characteristic: 0b1000111000 this tells the SensorTag to turn on the x/y/z axis of the accelerometer with a range of 8g
- It also initially sets the sensor period with the following value: 0x0A which indicates a period of  $10 \times 10 = 100\text{ms}$  between readings
- When energy conservation mode is activated it will send: 0x64 which indicates a period of  $100 \times 10 = 1000\text{ms}$  between readings
- Based on the Bluetooth Power Calculation excel sheet [8] it indicates that the battery should last for a period of at least 2 weeks before discharge, it does describe a slightly different scenario, so this is only an estimate

State	Time [μs]	Current [mA]	Time * Current
1 Wake Up & Pre-Processing	1317.80	3.06	4034.87
2 Radio Preparation	328.00	4.06	1332.93
3 Transmit (TX)	1118.70	7.07	7909.32
4 TX to TX transition	570.00	3.61	2056.20
5 Transmit (TX)	1118.70	7.07	7909.32
6 TX to TX transition	570.00	3.61	2056.20
7 Transmit (TX)	1118.70	7.07	7909.32
8 TX to TX transition	570.00	3.61	2056.20
9 Transmit (TX) – on secondary channel	1653.22	7.07	11688.43
10 TX to RX transition	145.56	5.86	853.37
11 Receive (RX) – on secondary channel	456.78	6.73	3074.74
12 Post-Processing	887.44	2.82	2506.23
13	0.00	0.00	0.00
14	0.00	0.00	0.00
15	0.00	0.00	0.00
16	0.00	0.00	0.00
Total time of advertising event [us]		9854.90	
Total time * current [us*mA]			53387.1
Average Current draw during advertising event [uA]			5417.3

<b>Avg. current draw when advertising:</b>	<b>535.0</b> uA
<b>Expected battery life:</b>	<b>421</b> Hours
<b>Expected battery life:</b>	<b>18</b> Days

Figure 2: BLE power estimate for transmit every 100ms

### Android App Details

App features and UI: The Android app has three interfaces

- Login: when the app is first started a username/password must be entered to authenticate the user
- Device selection: a list showing available BLE devices is shown so the user can select theirs, this is remembered for future use of the app
- Data gathering/display: Shows current SensorTag readings along with a chart of their recorded step count throughout the day, user has the ability to enable a training mode from a drop-down menu, or navigate to previous days using a side swipe gesture on the graph



Figure 3: Main dashboard of the Android app

Staying connected to the SensorTag: The Android app has a number of threads that run continuously to monitor the connection with the SensorTag and ensure it is operating smoothly

- Before a connection has been established: if a connection hasn't been successfully after 25 seconds then it will be reset, and a connection will be attempted again
- After a connection has been established: every two seconds if new data hasn't been received then it is assumed that the connected has been lost, this will trigger the other thread to start attempting a new connection

Securing communications with backend: Communications with the backend have been secured in multiple ways, described as follows

- HTTPS TLS is used to protect the confidentiality of all communications between the Android app and the backend
- An auth token is presented in the message headers to ensure only the app can make calls to the backend, this is described further in the AWS backend section
- When a user logs in they are provided with a shared secret that only the backend and that one user will share, it allows messages to be encrypted using AES-128 [9] this ensures that a user can only submit data for themselves and not anyone else, if they attempt to impersonate another user the data will not be decrypted successfully by the backend

Energy conservation mode: In order to reduce energy consumption a conservation strategy has been devised as follows

- Every one second the Android app will store the current combined sensor reading into a list
- After ten readings have been recorded the variance is calculated based on the last ten readings
- If conservation mode is off and variance falls below THRESH\_LOW (currently set to 0.0005) then conservation mode is enabled, and the sensor period is set to 1000ms

- If conservation mode is on and the variance exceeds THRESH\_HIGH (currently set to 0.001) then conservation mode is disabled, and the sensor period returns to 100ms
- The values have been tuned based on testing, refer to appendix item A for test data

## AWS Backend

AWS Cloud platform is used as the backend service provider. It is used primarily for

1. Data Storage - DynamoDB, S3
2. Expose microservice - Lambda, API Gateway
3. Security and Authentication - IAM Roles and Policy
4. Data Processing - EC2, Lambda
5. Monitoring and Visualization – Elasticsearch, Kibana

### Data Storage:

DynamoDB table and S3 bucket are used for data storage.

#### DynamoDB table details:

1. **android\_user\_credentials**  
Usage: Stores the android user credentials, name password and a secret key that is used for encryption/decryption of the request  
Primary/Partition Key: username
2. **sensor\_acc\_data**  
Usage: Stores the sensor data that is received from android app.  
Primary/Partition Key: user\_timestamp
3. **step\_counts**  
Usage: Stores the step count info.  
Primary/Partition Key: user\_id
4. **training\_data**  
Usage: Stores the training related data of the user  
Primary/Partition Key: user and timestamp

*Note: Please refer appendix for the table definition*

### Simple Storage Service (S3):

A S3 bucket named comp6733-jarvis is used to save the .pkl files that are used in data processing in machine learning

### Data Processing:

Micro-services are created using Lambda functions. Due to code size limitation of Lambda function, the machine learning code is made to hosted on EC2 instance. This code is wrapped with a python flask API. This API is used by the Lambda function for invocation.

Below are the Lambda Functions:

**post\_data\_db:** Parses the requests received from the android app. This request is AES encrypted using the shared key that is given to the user when he login in the app. The request is decrypted using the same key at AWS. If the request is from the genuine user, the decryption will be successful, and it is inserted in sensor\_acc\_data table.

**add\_android\_app\_users:** Registers the new android app user in the system. The password in the incoming request will be hashed before it is saved in the table. A new record with the user and



hashed password will be inserted in the table. A shared secret key which is the hashed username is also added against this newly inserted row as a shared secret column value.

**verify\_android\_app\_user:** Verifies the user credentials provided in the request against the user and password that is stored in the table while the user was registered. On successful verification, a shared key is passed to the user in the response. If the user is not in the database, failure response will be sent.

**get\_steps\_date:** This fetches the hourly step information of the user for the date provided from steps\_counts table. The input parameters, date and username are sent as a query parameter. The result includes the number of genuine steps and also false steps (if any)

**appuser\_auth:** Uses an authorization token to allow or deny a request. This function is called by the Lambda authorizer to check if the token that comes in the TokenHeader is correct or not. If the token is empty, Invalid token response message is sent. If the token is wrong, null is response is sent back. If the token is correct, then the request is forwarded for processing.

**processNewStepData:** This function is invoked whenever the data is inserted into Dynamo DB table - sensor\_acc\_data. This in turn calls the python flask api that does the machine learning.

## Exposed microservices

The backend microservices are exposed via API gateway. Below are the apis that are created

### 1. pedometer\_api [POST]

Communication protocol: http

Method: POST

Request Format: JSON

Authentication: Uses Lambda Authorizer feature, expects a name/value parameter with values - tokenHeader/xx-allow123456333\*\*\*\*\* in the request header. And also, the uses the shared secret key for request verification.

Functionality: Updates the received request data in the sensor\_acc\_data table

Lambda function: post\_data\_db

Dynamo DB Table: sensor\_acc\_data

Request	Response
"Suchithra 19:11:07:442,- 0.08935546875,0.1030273438,- 0.9602050781,0.9698416497 19:11:07:442,- 0.08154296875,0.1110839844,- 0.958984375,0.9688343196"	{ "statusCode": 200, "statusMessage": "Success" }

### 2. pedometer\_api [GET]

Method: GET

Request Format: JSON

Functionality: Fetches the step count from the sensor\_acc\_data table.

Lambda function: get\_steps\_date

Dynamo DB Table: step\_counts

Request	Response
Query parameter: user = "username" date = ""	<pre>{   "statusCode": 200,   "body": {     "false_steps": 0,     "total": 0,     "hourly": [0,0,0,0,0,0, 0,100,200,0, 0,600,0, 0,0,0,1000,0,0, 0,0, 0,0,0 ]   } }</pre>

### 3. android\_user\_register\_api

Communication protocol: http

Method: POST

Request Format: JSON

Functionality: Registers/adds the Android app user credentials.

Lambda function: add\_android\_app\_users

DynamoDB Table: android\_user\_credentials

Request	Response
<pre>{   "username": "user",   "password": "password" }</pre>	<pre>{   "statusCode": 200,   "statusMessage": "Success" }</pre>

### 4. android\_user\_verify\_api

Communication protocol: http

Method: POST

Request Format: JSON

Functionality: Verifies Android app user credentials.

Lambda function: verify\_android\_app\_user

Dynamo DB Table: android\_user\_credentials

Request	Response
<pre>{   "username": "user",   "password": "password" }</pre>	<pre>{   "statusCode": 200,   "statusMessage": "shared_secret" }</pre>

## Security and Authentication

Lambda authoriser feature provided by AWS is used to control the access to API. To access AWS resources, Roles are created with policy as mentioned below.

Role	Policy
------	--------

processNewStepDataRole	AmazonDynamoDBFullAccess, AWSLambdaBasicExecutionRole, AWSLambdaMicroserviceExecutionRole
save_android_app_user_data	AWSLambdaBasicExecutionRole And all type of action on table android_user_credentials
save_sensor_data	AWSLambdaBasicExecutionRole, PutItem action on tablesensor_acc_data, And all type of action on table android_user_credentials
newrolefordynamodbandawslambda	AWSLambdaFullAccess, CloudWatchFullAccess, AmazonSESEFullAccess
6733-lambda-dynamoDB-S3	AmazonS3FullAccess, AmazonDynamoDBFullAccess, AmazonESFullAccess, AmazonLambdaBasicExecutionRole
appuser_auth-role-3ljgkk1d	AWSLambdaBasicExecutionRole

## Monitoring and Visualization

AWS Kibana is used for data visualisation. It is accessed via Elastic search service. The dashboard shows the below data

1. Top 5 Users – True Steps count
2. Top 5 Users – False Steps count
3. Total True Steps daily count
4. Total False Steps daily count
5. Total Steps (Genuine and False steps) per hour

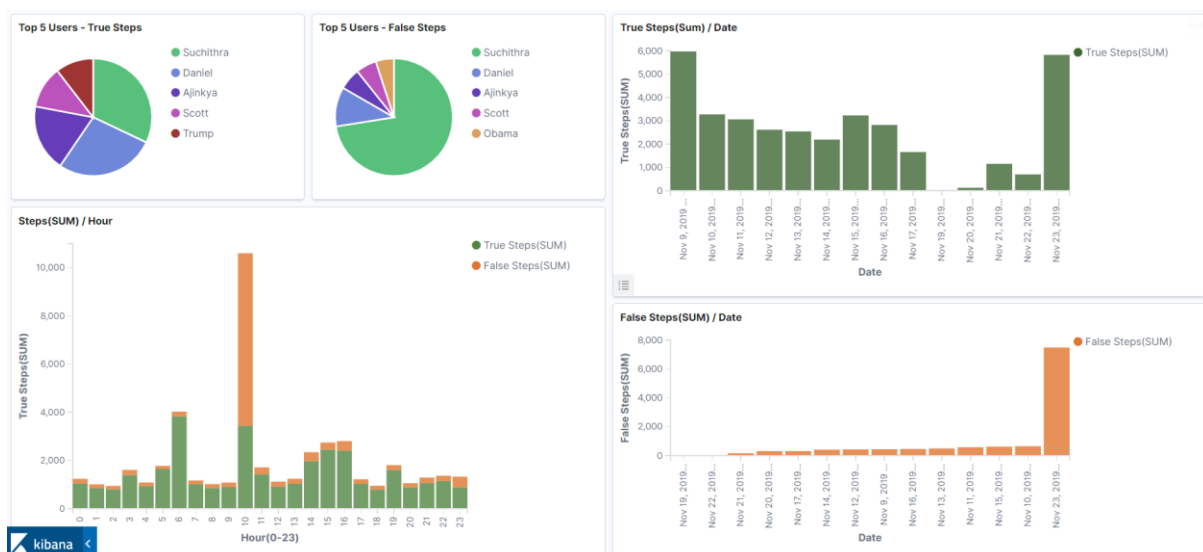


Figure 4: Kibana Dashboard

## Machine Learning

### Data Collection

For building machine model, accelerometer readings are collecting 10 samples per second and sending this data to AWS DynamoDB using the mobile app in every 30 seconds. Raw data has timestamps, x-axis (direction of walking) readings, y-axis (gravitational direction) readings, z-axis (sideways movement) readings. The sensor tag also provides the result of the combination vector, which we used for our final machine learning models.

Sample Data:

```
{
  "time": ["14:51:25:786", "14:51:25:889", ..., "14:51:55:814", "14:51:55:911"],
  "training_mode": "Y",
  "upload_time": "2019-11-21_14:51:56",
  "user_timestamp": "JohnSmith_2019-11-21_14:51:56",
  "x": ["-0.25146", "7.32421", ..., "-0.075927", "-0.275390625"],
  "y": ["-0.769287109375", "-0.477783203125", ..., "-1.07763671875", "-1.0578613"],
  "z": ["-0.6857015625", "-0.386474609375", ..., "-0.574951171875", "-1.07397460"],
  "combined": ["1.060822724", "0.6145261596", ..., "0.7952993269", "0.983242806"]
}
```

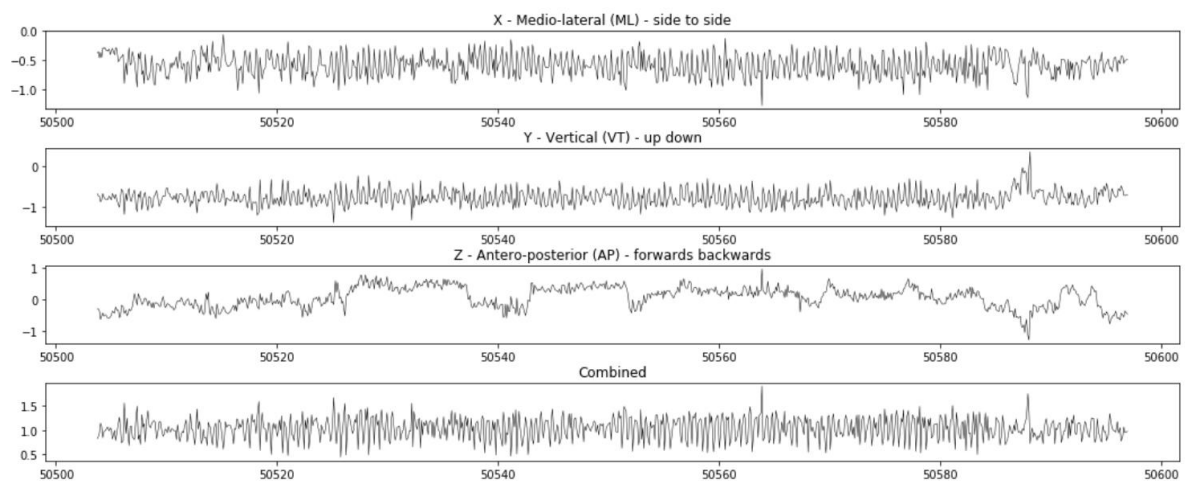


Figure 5: Accelerometer Data

### Data Processing

Since raw sensor data has many noises, we filter out the noisy readings by passing raw data through a low pass Butterworth filter. The output is a smooth signal whose peaks represent steps. An AWS EC-2 instance is hosting python codes for signal processing and machine learning.

Below figure shows the filtering of noisy sensor data. The gray line is the original sensor data and the red line shows the filtered data.

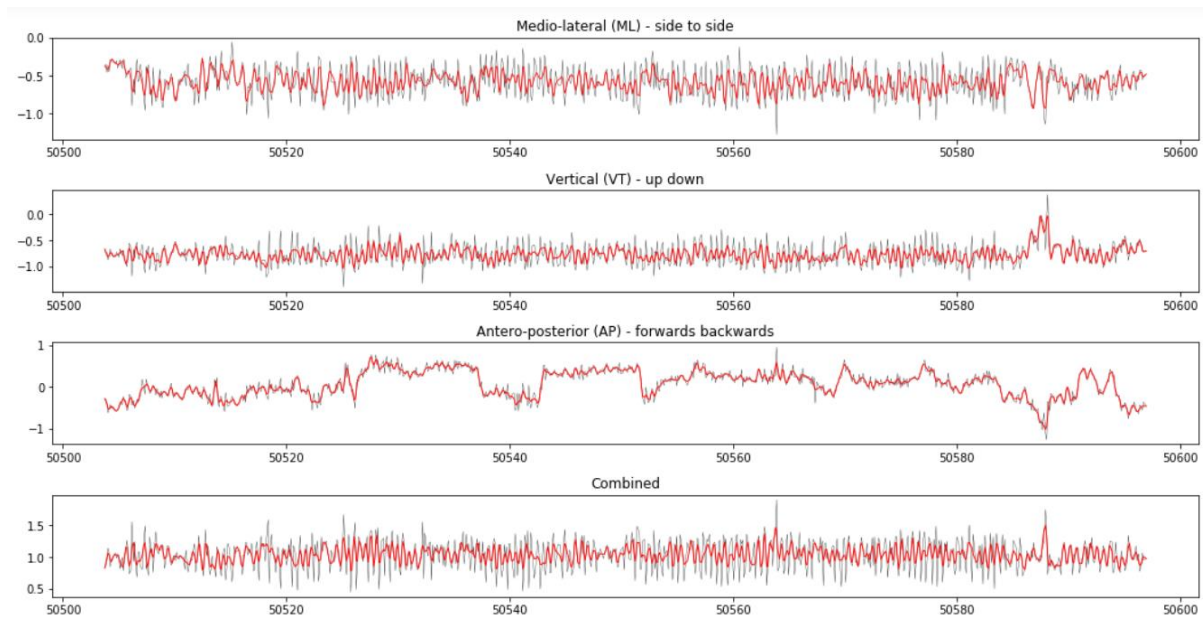


Figure 6: Sensor Data after filtering

## Feature Extraction and Feature Selection

By counting peaks of the filtered signal gives us step counts. By processing peak times, we calculate the number of steps per minute, mean step time, standard deviation of step times, coefficient of variations.

We have used these features to train our machine learning model and predict during test.

Sample Training Data:

user	cadence_comb	step_time_comb	step_time_cov_comb	step_time_sd_comb	Label
Ajinkya	85.28	0.69	0.17	0.12	1
Ajinkya	88.19	0.68	0.16	0.11	1
Ajinkya	92.05	0.65	0.11	0.07	1
Daniel	88.55	0.67	0.34	0.23	0
Daniel	98.88	0.6	0.31	0.19	0
Suchithra	111.69	0.54	0.26	0.14	0
Suchithra	106.39	0.55	0.27	0.15	0
Suchithra	113.38	0.53	0.24	0.12	0

Figure 7: Training for User "Ajinkya"

## Model Creation and Model Selection

We have created machine learning models for each user and determined if the user can be verified by our model or not (Binary classification). We instantiated different machine learning models (list of models) and peaked the best model for each user.

We created these models and evaluated each model against each user.

1. Support Vector Machine
2. Multilayer Perceptron
3. Decision Tree

4. k-Nearest Neighbour
5. Logistic regression
6. Gaussian Naïve Bayes

Pseudo code for machine learning model selection:

```

all_users = [user_1, user_2, ....., user_N]
all_models = [model_1, model_2, model_3, ....model_K]
user_data = get_all_training_data(user)
user_data.set_label(1)
non_user_data = get_all_training_data(non-user)
non_user_data.set_label(0)
training_data = user_data + non_user_data

previous_score = 0
selected_model = None
for model in all_models:
|   model.fit(training_data.X)
|   score = cross_val_score(model, training_data.X, training_data.y)
|   if score > previous_score then
|       |   selected_model = model
|   End if
end for
save selected_model

```

## Execution

Since the machine learning model is a binary classification for each user, we created four users (three team members and a fake user) in the system and walked in different surfaces separately to create enough training data for each user.

After training is completed, we tested each user for both positive and negative scenarios. Test results are in the next section.

Model Performance:

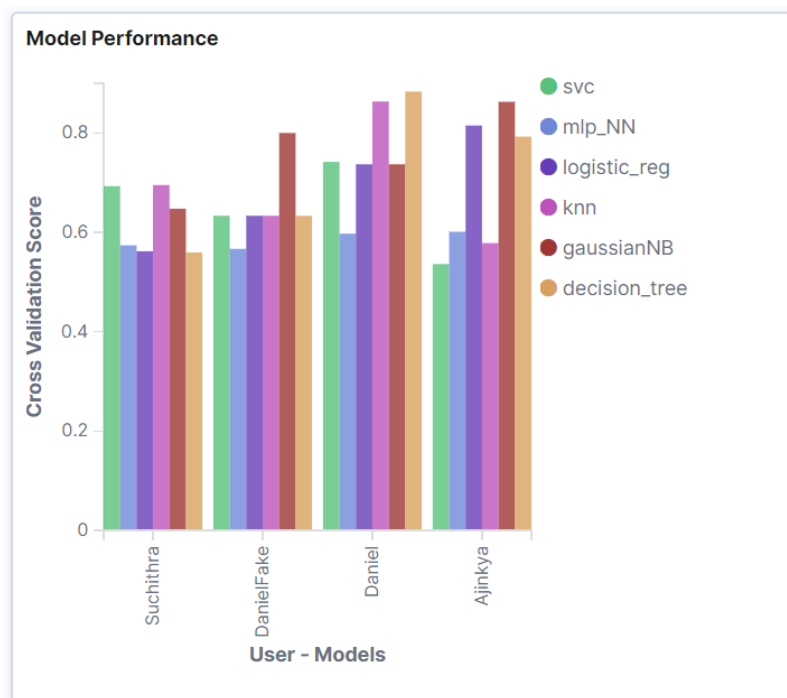


Figure 8: Model Performance on Users

user	model	cross_val_score	f1_macro	f1_micro	f1_weighted
Ajinkya	svc	0.5361111111	0.7333333333	0.75	0.75
	mlp_NN	0.6011111111	0.384615385	0.625	0.480769231
	decision_tree	0.7927777778	0.873015873	0.875	0.876984127
	knn	0.5783333333	0.7333333333	0.75	0.75
	logistic_reg	0.815	0.619047619	0.625	0.607142857
	gaussianNB	0.8627777778	1	1	1
Suchithra	svc	0.692857143	0.8194444444	0.846153846	0.835470085
	mlp_NN	0.573809524	0.380952381	0.615384615	0.468864469
	decision_tree	0.55952381	0.763636364	0.769230769	0.772027972
	knn	0.695238095	0.8375	0.846153846	0.846153846
	logistic_reg	0.561904762	0.675	0.692307692	0.692307692
	gaussianNB	0.647619048	0.8375	0.846153846	0.846153846
Daniel	svc	0.741818182	1	1	1
	mlp_NN	0.597272727	0.388888889	0.636363636	0.494949495
	decision_tree	0.8833333333	1	1	1
	knn	0.8633333333	1	1	1
	logistic_reg	0.736969697	0.607142857	0.636363636	0.636363636
	gaussianNB	0.736969697	0.607142857	0.636363636	0.636363636
DanielFake	svc	0.6333333333	0.25	0.3333333333	0.3333333333
	mlp_NN	0.566666667	0.4	0.666666667	0.5333333333
	decision_tree	0.6333333333	0.25	0.3333333333	0.3333333333
	knn	0.6333333333	0.666666667	0.666666667	0.666666667
	logistic_reg	0.6333333333	0.666666667	0.666666667	0.666666667
	gaussianNB	0.8	1	1	1

## Test Results

During training, we calculated and stored training accuracies for each user. This application is developed in a way to store these results for future also without any manual intervention and it has Kibana visualization as well.

User		Positive	Negative	Total Count by "Jarvis"	Pedometer Count	Accuracy
Daniel	TRUE	835	147	982	980	0.85
	FALSE	135	756	891	887	0.848
Suchithra	TRUE	998	434	1432	1427	0.69
	FALSE	91	245	336	334	0.73
Ajinkya	TRUE	1065	272	1337	1329	0.797
	FALSE	86	221	307	312	0.72

Figure 9: Test Results

## Value Addition

- This application is completely automated to onboard new users, train with their gait information. With the use of AWS backend, the entire application is totally scalable. With use of different machine learning models and automatically selecting best model for individual user separately, its accuracy is high.
  - Mobile app has different modes to either automatically train the system, or a normal mode where the application counts steps and detects if the user is verified or not.
  - With different visualizations, users can keep track of their walks and if any false step is detected or not in their mobile phones.
- Kibana dashboard has been developed to help both developers and the service provider (insurance company, government etc.). All users and their step counts (both true and false counts) can be analysed from overall to hourly level.
- Different dashboards are developed to analyse the training data, check model performance to help developers and maintenance.

## Future Improvement Scope

### Mobile App

Goal setting functionality: Allows to set step-goals for a day/week/month

Reminders and Notifications:

Achievement board: Show all the goals achieved

Inbox: To receive info regarding the health, sport related info etc

Connect People: Connect pedometer users online

### AWS Backend

AWS Backend has been built as per standard and no further improvement is required at current stage. If some modifications are done in mobile app or machine learning program, some modification may be required accordingly. Although, that is less likely to be the case since we have set everything up to be scalable.

### Machine Learning

This project is primarily focused on processing sensor(accelerometer) data and using that data in a meaningful way. Considering the duration and scope of the project, although significant work has been done and the results are satisfying, still there is some room for improvement in machine learning.

- More models can be added to see if they can improve accuracy or not  
Current code is written in a way so that to add new models, minimum coding effort will be required.

Current code sample:

```
models = {'svc': svc_clf, 'mlp_NN': mult_layrer_perceptron_clf, 'decision_tree': dt_clf, 'knn': knn_clf, 'logistic_reg': log_clf, 'gaussianNB': gaussianNB}
```

New models can be added here and the code will automatically select the best performing model (ref. Earlier mentioned pseudo code).

- Experiment with more features and data  
We have selected the features based on our understanding and some testing. Further experiment with more data potentially can show other features providing more accuracy. This can be done only with more usage of the application over time.



## Energy Saving

The vibration can be harvested using any KEH (Kinetic Energy Harvester) devices. We can build a prototype based on piezoelectric energy harvester (PEH) We can exploit the output voltage signal of the kinetic energy harvester for gait recognition directly. By not using the accelerometer, KEH-Gait can save the energy that is used to sample the accelerometer. The saved energy can be further used to power other components in the wearable device

## Conclusion

In this project, we designed and implemented an advanced accelerometer-based pedometer, which uses gait of the user for detecting false steps. With the aid of advanced machine learning and other techniques we have achieved significant improvements in robustness over the traditional pedometer. In addition to that we have built a complete system that works from end to end, from onboarding new users and training the system to recognise them, to allowing third parties like health companies visibility over the analysis provided by the system.

## References

- [1] 'Bluetooth Low Energy software stack', 2019, [Online], accessed 7 November 2019, <<http://www.ti.com/tool/BLE-STACK>>
- [2] 'Simplelink SensorTag', 2019, [Online], accessed 7 November 2019, <<https://play.google.com/store/apps/details?id=com.ti.ble.sensortag>>
- [3] 'An Android app connecting to a TI CC2650 SensorTag for acceleration measurements', 2018, [Online], accessed 7 November 2019, <<https://github.com/martindisch/SensorTag-Accelerometer>>
- [4] 'Python package for analyzing sensor-collected human motion data (e.g. physical activity levels, gait dynamics)', 2018, [Online], accessed 7 November 2019, <<https://github.com/sho-87/sensormotion>>
- [5] 'Lambda Authoriser feature', 2019, [Online], accessed 28 October 2019, <<https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html>>
- [6] 'SVM Neural Network model', 2019, [Online], accessed 2 November 2019, <<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>>
- [7] 'MLP Neural Network model', 2019, [Online], accessed 4 November 2019, <[https://scikit-learn.org/stable/modules/generated/sklearn.neural\\_network.MLPClassifier.html](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html)>
- [8] 'Bluetooth Power Calculator Tool', 2019, [Online], accessed 22 November 2019, <<http://www.ti.com/tool/BT-POWER-CALC>>
- [9] 'Exchanging AES encrypted data between Python and Java', 2019, [Online], accessed 22 November 2019, <[https://github.com/ijl20/python\\_java\\_crypto](https://github.com/ijl20/python_java_crypto)>
- [10] 'Detect swipe gesture in fragment', 2019, [Online], accessed 22 November 2019, <<https://stackoverflow.com/questions/22981337/detect-swipe-gesture-in-fragment>>
- [11] 'sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.21.3 documentation', 2019 [Online], accessed 2 November 2019, <<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>>
- [12] '1.10. Decision Trees — scikit-learn 0.21.3 documentation', 2019 [Online], accessed 2 November 2019, <<https://scikit-learn.org/stable/modules/tree.html>>
- [13] 'sklearn.linear\_model.LogisticRegression — scikit-learn 0.21.3 documentation', 2019 [Online], accessed 2 November 2019, <[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LogisticRegression.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)>
- [14] 'sklearn.naive\_bayes.GaussianNB — scikit-learn 0.21.3 documentation', 2019 [Online], accessed 2 November 2019, <[https://scikit-learn.org/stable/modules/generated/sklearn.naive\\_bayes.GaussianNB.html](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html)>

## Appendix

### A. Energy conservation data from Android app

T (s)	Sensor combined (g)	Variance	Conservation mode
0	0		OFF
1	1.09889412		
2	1.11256182		
3	1.104874549		
4	1.052193639		
5	1.120141905		
6	1.11142257		
7	1.101322148		
8	1.109388835		
9	1.069867508	0.1088712	
10	1.113974131	0.0004195	ON
11	1.114378191	0.0004393	
12	1.114378191	0.0004438	
13	1.100552062	0.0004423	
14	1.100552062	0.000183	
15	0.806803874	0.0081078	OFF
16	0.788149353	0.0151108	
17	0.629421757	0.0295481	
18	0.980256593	0.0280881	
19	2.036286931	0.1291607	
20	1.013815633	0.1293524	
21	1.045561294	0.1291466	
22	1.030472115	0.1288942	
23	1.101998966	0.1289081	
24	1.04257955	0.1286631	
25	1.046000044	0.122296	
26	1.040894492	0.1137245	
27	1.021989949	0.0909044	
28	1.040433997	0.089356	
29	1.034626635	0.0005014	
30	1.040699437	0.0004158	ON
31	1.09852563	0.0006793	
32	1.09852563	0.0008327	
33	1.065649246	0.0006218	
34	1.065649246	0.0006216	
35	1.055786916	0.000612	
36	1.055786916	0.0005862	
37	1.041179193	0.000482	
38	1.041179193	0.0004792	

Figure 10: Energy conservation data from Android app

## B. AWS Setup – API Gateway

**APIs (3)**

Find APIs

	Name ▲	Description ▼	ID ▼	Protocol ▼	Endpoint type	Created
<input type="radio"/>	android_user_register_api		3gchazpaci	REST	Edge	2019-11-14
<input type="radio"/>	android_user_verify_api		pvk32jnthd	REST	Edge	2019-11-14
<input type="radio"/>	pedometer_api		9yypng2mga	REST	Edge	2019-11-01

## C. AWS Setup – Lambda Function

**Functions (7)**

Filter by tags and attributes or search by keyword

	Function name ▼	Description	Runtime ▼	Code size ▼	Last modified ▼
<input type="radio"/>	post_data_db	This Lambda function saves the data in the sensor_acc_data table	Python 3.7	10.4 MB	2 days ago
<input type="radio"/>	verify_android_app_user	This Lambda function verifies Android app user credentials	Python 3.7	738 bytes	8 days ago
<input type="radio"/>	get_step_count	This Lambda function fetches the step count from the sensor_acc_data table	Python 3.7	798 bytes	13 days ago
<input type="radio"/>	get_steps_date	This Lambda function fetches the step count from the sensor_acc_data table as per the date provided	Python 3.7	574 bytes	22 seconds ago
<input type="radio"/>	add_android_app_users	This Lambda function registers/adds the Android app user credentials	Python 3.6	601 bytes	8 days ago
<input type="radio"/>	appuser_auth	This Lambda function uses an authorization token to allow or deny a request	Node.js 8.10	938 bytes	7 days ago
<input type="radio"/>	processNewStepData	An Amazon DynamoDB trigger that logs the updates made to a table.	Python 3.6	1005.9 kB	48 minutes ago

## D. AWS Setup – Kibana

**Visualize**

Search...

1–10 of 10

	Title ↑	Type
<input type="checkbox"/>	False Steps(SUM) / Date	Vertical Bar
<input type="checkbox"/>	Model Performance	Vertical Bar
<input type="checkbox"/>	Steps(SUM) / Hour	Vertical Bar
<input type="checkbox"/>	Top 5 Users - False Steps	Pie
<input type="checkbox"/>	Top 5 Users - True Steps	Pie
<input type="checkbox"/>	Training Data - Cadence	Line
<input type="checkbox"/>	Training Data - Step Time	Line
<input type="checkbox"/>	Training Data - Step Time Co-Variance	Line
<input type="checkbox"/>	Training Data - Step Time Standard Deviation	Line
<input type="checkbox"/>	True Steps(Sum) / Date	Vertical Bar

0 items selected

1–10 of 10



- DynamoDB
  - Dashboard
  - Tables
  - Backups
  - Reserved capacity
  - Preferences
- DAX
  - Dashboard
  - Clusters

Create table

Delete table

Filter by table name

×

Choose a table group

▼

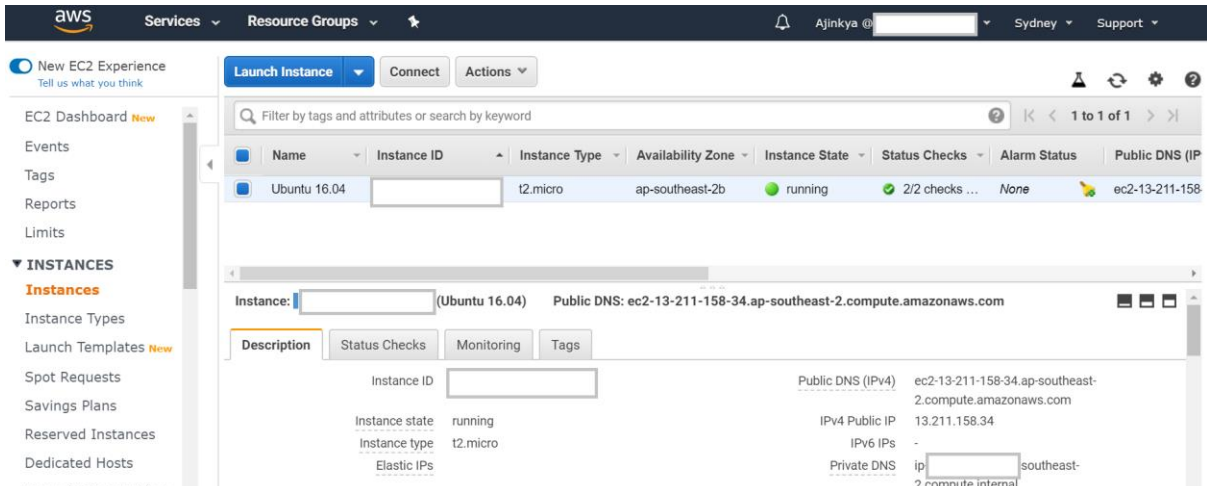
Actions

▼

ⓘ

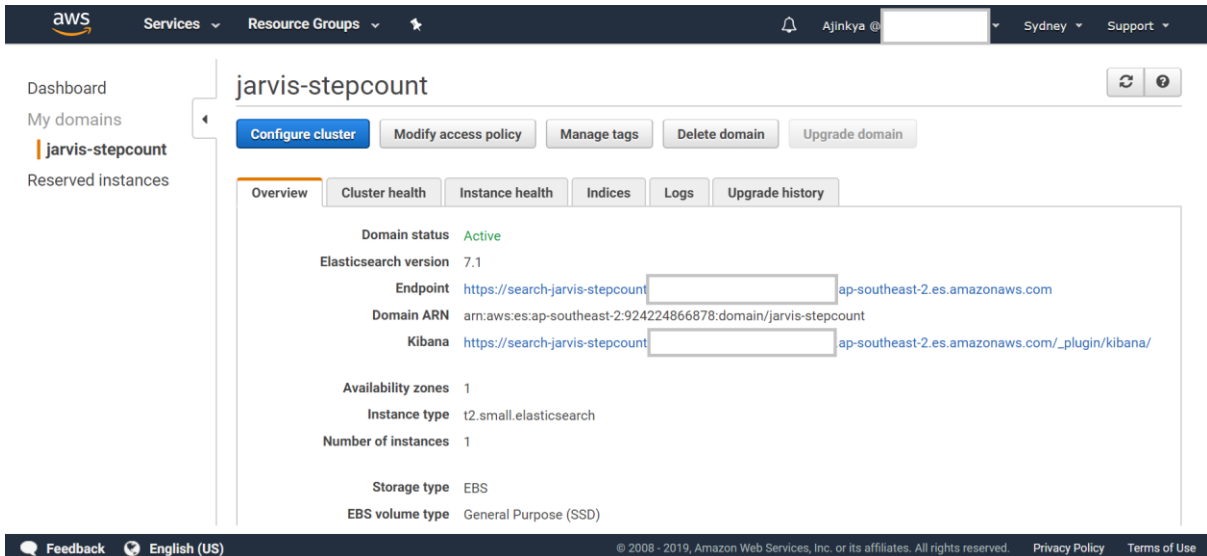
	Name	Status	Partition key	Sort key
<input type="radio"/>	android_user_credentials	Active	username (String)	-
<input type="radio"/>	sensor_acc_data	Active	user_timestamp (String)	-
<input type="radio"/>	step_counts	Active	user_id (String)	date (String)
<input type="radio"/>	training_data	Active	user (String)	timestamp (String)

## F. AWS Setup – EC2



The screenshot shows the AWS Management Console interface for an EC2 instance. The instance is named 'Ubuntu 16.04', has an Instance ID, and is of type 't2.micro' in the 'ap-southeast-2' availability zone. It is currently in a 'running' state with 2/2 status checks passed. The public DNS is 'ec2-13-211-158-34.ap-southeast-2.compute.amazonaws.com'. The console also shows a list of instances on the left and a detailed view of the selected instance on the right, including its description, status checks, monitoring, and tags.

## G. AWS Setup – Elasticsearch



The screenshot shows the AWS Management Console interface for an Elasticsearch domain named 'jarvis-stepcount'. The domain is in an 'Active' state. The console displays various details including the endpoint 'https://search-jarvis-stepcount-...ap-southeast-2.es.amazonaws.com', domain ARN 'arn:aws:es:ap-southeast-2:924224866878:domain/jarvis-stepcount', Kibana URL 'https://search-jarvis-stepcount-...ap-southeast-2.es.amazonaws.com/\_plugin/kibana/', availability zones (1), instance type 't2.small.elasticsearch', number of instances (1), storage type 'EBS', and EBS volume type 'General Purpose (SSD)'. The console also shows a list of domains on the left and a detailed view of the selected domain on the right, including its configuration, health, and upgrade history.

## H. DynamoDB Table – Android\_user\_credentials

Column Name	Column Description
Username	Name of the Android app user
password (hashed)	Password of the Android app user
shared_secret key	The user key for data encryption/decryption

## I. DynamoDB Table – Sensor\_acc\_data

Column Name	Column Description
user_timestamp	pedometer username along with the time stamp
Combined	The combined accelerometer value
Falsified	Indicates if the step are genuine (1) or not (0)
step_count	Number of steps taken by the user
Time	Time at which the steps were taken

training_mode	Indicates if the data is gathered during training (Y) or not (N)
upload_time	The time at which the request was uploaded to table at AWS end
X	A list of accelerometer x value
Y	A list of accelerometer y value
Z	A list of accelerometer z value
gait_verified	Indicates if the gait is verified (Y) or not (N)

## J. DynamoDB Table – Steps\_count

Column Name	Column Description
user_id	username
Date	Indicates the date when the steps were taken
false_steps	Indicates the total number of false steps taken
false_steps_hourly	Indicates the number of false steps taken per hour
Step_count	Indicates the total number of genuine steps taken

## K. DynamoDB Table – Training\_data

Column Name	Column Description
User	username
Timestamp	Timestamp at which the training date was taken
cadence_comb	Cadence combined value
cadence_x	Cadence x value
cadence_y	Cadence y value
cadence_z	Cadence z value
step_count_comb	Combined step count
step_count_x	step count x value
step_count_y	step count y value
step_count_z	step count z value
step_time_comb	
step_time_conv_comb	
step_time_conv_x	
step_time_conv_y	
step_time_conv_z	
step_time_sd_comb	
step_time_sd_x	
step_time_sd_y	
step_time_sd_z	
step_time_x	Step time of x value
step_time_y	Step time of y value
step_time_z	Step time of z value