**SETSTATS**

A picture containing antenna

Description automatically generated

Mark Byrne Richard Collins Liam Denning Florian Scheunert

**Table of Contents**

Table of Contents

[Team Members 3](#_Toc90840432)

[What is SetStats? 3](#_Toc90840433)

[What are our goals? 3](#_Toc90840434)

[Github Link: 3](#_Toc90840435)

[Hardware Required 4](#_Toc90840436)

[Setup version 1: 6](#_Toc90840437)

[Setup version 2: 6](#_Toc90840438)

[Explanation: 7](#_Toc90840439)

[Example: 7](#_Toc90840440)

[Data the device will be collecting: 8](#_Toc90840441)

[How the data will be gathered: 8](#_Toc90840442)

[How we are going to be processing the data collected: 8](#_Toc90840443)

[Frequency with which each sensor can record a value and how the sensors work: 8](#_Toc90840444)

[Instance Security: 8](#_Toc90840445)

[Database Implementation: 10](#_Toc90840446)

[Database Security: 10](#_Toc90840447)

[Paper prototype page descriptions: 11](#_Toc90840448)

[Alpha Release Website 12](#_Toc90840449)

[Website Security: 13](#_Toc90840450)

[Connection: 14](#_Toc90840451)

[User Profiles 15](#_Toc90840452)

[Software-Testing 18](#_Toc90840453)

[Hardware-Testing 18](#_Toc90840454)

[User-Testing 18](#_Toc90840455)

**About Us**

# Team Members

**Mark Byrne: Liam Denning:**

A person in a yellow shirt

Description automatically generated with low confidenceA picture containing person, person, male

Description automatically generated**Role:** Database developer **Role:** Web Developer

**Richard Collins Florian Scheunert**

**Role:** IoT Developer **Role:** Tester/ Designer

A person in a suit and tie

Description automatically generated with medium confidenceA picture containing person, indoor, posing

Description automatically generated

# What is SetStats?

SetStats is an application created to help everybody improve their form in the gym. Whether you are a new user or a 20-year veteran, SetStats will surely help you. Trainers can use it to see how their trainees are doing and trainees can use it to find a trainer to watch them.

# What are our goals?

* We want to create a platform where you can track progress and improve your form.
* Beginners will have a platform to watch and learn from more experienced lifters through our tutorial section.
* SetStats hope to avoid injury by making sure lifters are not lifting too heavy or with incorrect form.
* Trainers should be able to see all of their trainees’ stats and with the help of SetStats they will have a better idea of how the trainee can improve their training

## Github Link:

<https://github.com/MarkByrneDKIT/setstats>

**Hardware**

# Hardware Required

1. **Raspberry Pi Zero** – this will be used as it is the more compact option whilst also having low power consumption. It will collect, process and further send the sensor data to a database on the Amazon Web Services, where it will be stored for later use.
2. **Accelerometer (MPU-6050)** – The accelerometer is used to measure the displacement on the z-axis and displayed on the x-axis of the graph (sway). When reading in data, there is jitter with the values. From running my code and collecting a sample of 10 accelerometer values at rest. I have found that the sway does not return to 0cm at rest but returns to an average of -0.85cm with the max being -0.72cm and the min being -0.92cm. This can be combatted by changing any value in the range of -0.95cm – -0.75cm to 0cm.

If the sway is measured to be <= 2 cm and >= -2 cm, it is considered perfect/safe. Otherwise, if it measures > 2 cm but < 4 cm and vice versa with negative values, it is considered okay/good and finally if it reads anything > 4 cm or < -4 cm, it is a fail.

Perfect = green    Good = yellow    Fail = red

[Data Sheet](https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf)

1. **Ultrasonic Sensor (HC-SR04)** – This sensor is being used to measure the distance on the y-axis (height). This data can then be compared to the x-axis to pinpoint at what time the lift becomes bad.

The sensor sends out a soundwave at a frequency of 40KHz and it travels through the air and if there is an object it will bounce back to the sensor.

“*The sensor is small, easy to use in any robotics project and offers excellent non-contact range detection between 2 cm to 400 cm (that’s about an inch to 13 feet) with an accuracy of 3mm.”.*

This extract was taken from <https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/>, this shows that the accuracy of the sensor is down to 3mm, which for our application is a fairly acceptable tolerance rate. This is the case **only**if the ideal conditions are met i.e., sensor is pointed directly at the floor and the floor isn’t an uneven/irregular surface.

There is a problem with invalid data when the ultrasonic sensor is <= 2cm from a surface. When the sensor is this close to a surface, readings can vary from 140cm to 2500cm which clearly cannot be used. To fix this issue I wrote code that will set any value from the ultrasonic sensor to 0 if the accelerometer is at rest.

[Data Sheet](https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf)

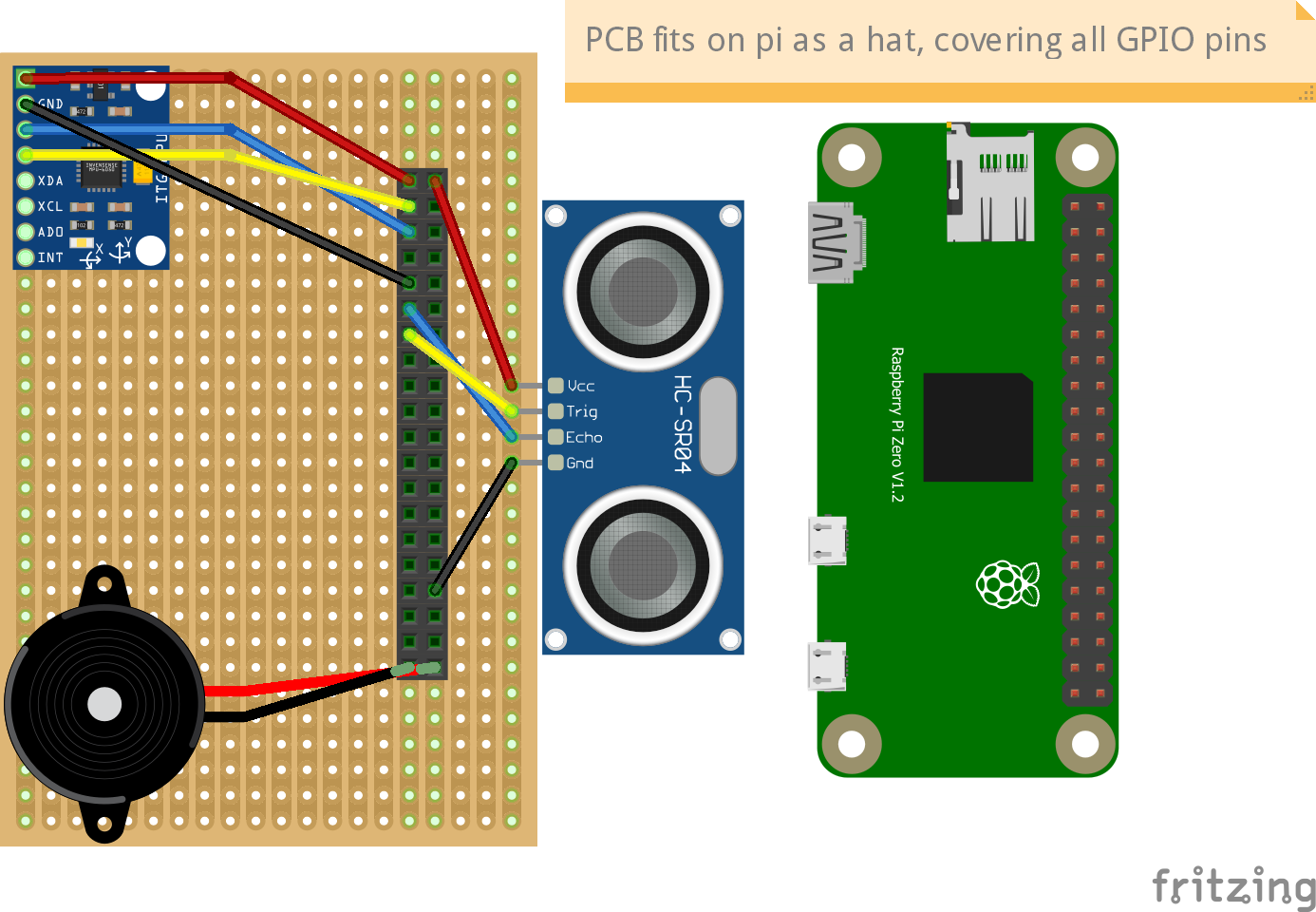
1. **Peizo Buzzer –** The buzzer is being used to alert the user if the lift becomes bad i.e., the accelerometer reads in data > 4 cm or < -4 cm.

*"Piezo buzzers are simple devices that can generate basic beeps and tones.  They work by using a piezo crystal, a special material that changes shape when voltage is applied to it.  If the crystal pushes against a diaphragm, like a tiny speaker cone, it can generate a pressure wave which the human ear picks up as sound.  Simple change the frequency of the voltage sent to the piezo and it will start generating sounds by changing shape very quickly!"*

from <https://learn.adafruit.com/using-piezo-buzzers-with-circuitpython-arduino>

[Data Sheet](https://www.mouser.com/datasheet/2/400/ef532_ps-13444.pdf)

1. **Power Bank (5000 mAh)** – the power source which will be connected via micro-USB, it can be recharged and reused.



**How we will power it and how we will connect it to the internet Is it mobile?**

The system will be connected to an external rechargeable battery source (Power Bank), which will give long lasting life due to the low power consumption of the Pi. It will be connected to the gym’s Wi-Fi or your phone’s hotspot. It would be mobile as long as you’re connected to a phone’s hotspot.

**Mounting Setup**

# Setup version 1:

Setup one is the testing setup, a simple setup. We got a broom stick and used zip ties to connect the pi and sensors to it so that we would be able to somewhat accurately get lift data to use for testing.

# Setup version 2:

Setup version two will include a more solid mount to the bar. We will use an actual barbell clip and make a small box for the raspberry pi to sit in and have holes for the ultrasonic sensor to be able to poke out and see the floor. The box will be super glued to the clip and the clip will be put on the end of the bar where the weights are put on. This version has the usefulness of being easy to setup and also has the second function of acting as a clip to hold on the weights.



Chart, histogram

Description automatically generatedShape

Description automatically generated

**Diagram

Description automatically generated**

**System Architecture**

# Explanation:

* Client -> website
* Pubnub -> real time communication platform
* Server -> AWS flask app & database
* Pi sensors -> raspberry pi with sensors connected to barbel

# Example:

The client requests a login, the login request is then sent onto the server and the server sends back it’s response, which is either login success or login failure. The user was returned a login fail, so now the user decided to register, the client sends a register request and the server sends back a register success. Along with this, the client sends a message to pubnub saying the user has registered. Now the user is logged in and starts a session, in the session the user clicks start lift, the client sends a request auth key to the server which then sends on the same request to pubnub, who then ends back a auth key to the server and from the server then back to the client. Clicking the lift button also sends a message to pubnub, which will then send a message to the raspberry pi to start the sensors. Now the pi sensors will send it’s sensor data to pubnub, which will then send on the data to the server and then to the client. The client will see the sensor data and once the stop button is clicked it will send a message through pubnub to the pi sensors to stop the sensors hence stopping the data being sent. The necessary data will then be sent to the server to be saved to the database and a message will be sent back from the server declaring if the data was saved or failed to save.

Data

# Data the device will be collecting:

* Height of the bar (y co-ordinate)
* Sway of the bar (x co-ordinate)
* Time
* Number of reps and sets
* Highest point (in which bar was lifted)

# How the data will be gathered:

The data will be sent from the device to a database hosted on a server. Every 0.75 seconds the sensor picks up a change in distance/sway etc the data will be sent to the database and stored in a table for that lift.

# How we are going to be processing the data collected:

The data collected will be sent to pubnub, through the website we grab the data and display it on the website and then save it to our database that is on the AWS instance.

# Frequency with which each sensor can record a value and how the sensors work:

Both sensors will collect data every 0.75 seconds.

**Accelerometer (MPU-6050) –** converts mechanical energy into electrical energy, it measures change in motion(acceleration)

**Ultrasonic Sensor (HC-SR04) –** Sends out a soundwave at a frequency of 40KHz and it travels through the air and if there is an object it will bounce back to the sensor.

# Instance Security:

The EC2 instance on AWS has a private key that only we have, this private key allows us to be able to SSH into the instance. Only we are able to log in as nobody else will have this private key hence securing the instance. Along with this we have a root password set, so if somebody does manage to grab our private key, they still need to know the password of the root to be able to do any major damage.

Diagram

Description automatically generated**Database**

**Trainee Table:**

The trainee table contains all the information of the regular user who uses the application. They have their own id and linked to them is a trainer id, but they don’t necessarily have to have a trainer. Their password will be hashed for security. You will be able to access all the data for each user through their unique trainee id.

**Trainer Table:**

The trainer table contains all of the data of a trainer who uses the app. The trainer can have multiple trainees and we can use sql to grab them uses the foreign trainer\_id in the trainee table. With this you can then view the history of each trainee so the trainer will be able to view the progress of their trainee.

**Current Session Table:**

The lift details table will contain the data for a specific trainees current lift, the x and y will be updated a lot as the user moves the bar, while the rep, set and time will change on a slower pace but still quickly enough. The data will help us for the history page of the trainee so they can see when they trained.

**Current Lift Table:**

At the moment the xy is stored in geometry LineString which means a real-time graph is not possible right now.  The lift\_id is how we link the currents\_session table and we use the lift\_id to determine which lift it is in each rep. After each lift is completed the data is stored in xy.

**History Table:**

The history table will be used for the trainee to be able to look at their progress overtime by looking at previous sessions. This will be able to grab the users best lift of the session and the rest of the data like reps and sets and how long the session was. The trainer can use this data also to be able to observe their trainee’s improvements.

**Session Table:**

The session table is what stores the data that the history table grabs useful data from the current\_session and current\_lift table are stored here after there session is done.

# Database Implementation:

With the AWS ec2 instance, we use ubuntu which is where we can store the database using the MySql install. With this we also use the secure installation of Mysql so we can disable a feature like anonymous user and also we can create a root password that is hashed. We can have the choice to disable remote logins so that only on the device can we login to the MySql.

We import the database into MySql and call it "SetStats" with this we can access all the tables and the data inside of them.

Now on the python file for the flask app, we import flask\_mysqldb and install this onto the AWS ec2 instance. Using mysqldb we can write into the database once it is connected. We will be putting the queries into the flask app and inserting the data grabbed from the sensors into the database along with user registrations.

We can now also read data from the "SetStats" database so with this we can tell if a user is registered to the database so we are able to log them in, also we can now display the history of each trainee using SELECT statements and dates sorted by descending so we get the newest sessions first in the list.

# Database Security:

On the instance we use mySql for our database. In order to stop un-authorised users from entering the database we set up a root for mySql. To do this we create a hashed password for the root user using mysql\_secure\_install and this will then disable anybody from entering the database without the password, the non-root user is then disabled also. In the code connecting to the database we then need to login as root and grab the password, you can do this by hardcoding the password into the code, but the better option would be to grab the password using os.getenv(). This will grab the password and not allow somebody who managed to access the code to be able to view the password.

**Website**

# Paper prototype page descriptions:

**1. Login Page**

User uses this page to login by entering their username and password that they registered with.

**2. Register Page 1**

User will use this page to begin their registration by entering their details like their username, password, and email. They will use these details to log into their account.

**3. Register Page 2 (About You)**

The second register page is to get the users physical details such as their height, weight and age. This data can be used for trainers and for the user to track their progress.

**4. Main Menu Page**

The main menu page contains the links to most of the applications pages, this is where the user will be able to navigate the app.

**5. Before Lift Page**

This screen is where the user has setup the device and is getting ready to lift, the user can click the start button to begin the session.

**6. During Lift Page**

This is what the user see’s while they are lifting, the graph displays the position of the bar compared to the optimal form line, the user can also see details such as the time of the session, what rep they are on and what set they are on.

**7. Lift Summary Page**

This is what the user can see after their session is finished or when they click on this session from their history. The summary page shows the users best lift of the session, their max weight lifted and time of session, along with the reps they done and sets.

**8. Session History Page**

This is where the user and their trainer can see the trainee’s session history, click one of the days and you will be linked to the lift summary page of that session.

**9. Side Bar Menu**

This is where the user can sign out and view other links to parts of the application which are not displayed on the main men u page and this side bar can be accessed from any page on the app.

**10. Trainees Page**

This is the page where the trainer can see their trainees and once they click on their name, they can see the trainee’s session history page.

# Alpha Release Website

**Home page:**

The home page is a CSS template taken from <https://templatemo.com> edited to suit the SetStats project. We have a video of a deadlift with good form in the background of the website so the user is engaged with the idea of SetStats when they open the page.

**Login/ Register:**

The trainer and trainee will have separate login and register pages. When a user fills in the login or register form a query is sent to the data base and an account is created or the session data is created.

**Drop Down Menu:**

The dropdown menu contains links to all the other necessary pages.

**Live Graph Page:**

On the live graph page, the user can tap a “Start Lift” button and the sensors will start getting data. In the CA3 release the graph will be drawn as the barbell is lifted.

**History Page:**

On the history page a table of all lifts by a user will be shown.

**Trainer Page:**

On the trainer page the trainer will see a list of his trainees. When a trainee is selected, they will see their history page.

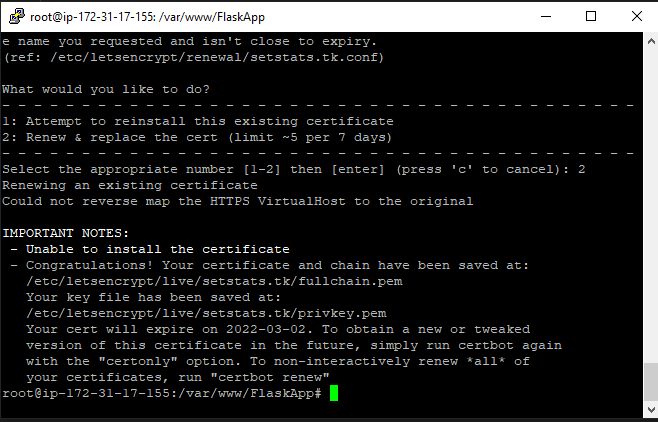
Graphical user interface, text

Description automatically generatedChart, line chart

Description automatically generatedGraphical user interface, application

Description automatically generated

# Website Security:

Originally, we were going to use certbot to get an SSL certificate but ran into some issues such as: 

So, to solve this issue, we resorted to using cloudflare to secure the website as Richard had previous experience using it.

User-Passwords:

To make sure the user chooses a secure password we used regex to give some rules to how the password needs to look like.

^(?=.\*[A-Za-z])(?=.\*\d)[A-Za-z\d]{6,}$

This regex allows us to make sure all passwords have at least one number, one letter and a minimum length of 6.

Redirect:

The website contains different pages and to make sure no one can access other without being logged-in we always check on a redirect if the user is currently logged-in and if not redirect him back to the login.



# Connection:

Connection to the device and app is through a Wi-Fi connection where the device is connected either to your hotspot/gym Wi-Fi. If there is no connection, a message should be displayed on the app telling the user that there is no solid connection and that they need to setup the device correctly.

**Users**

# User Profiles

Graphical user interface, text

Description automatically generated

Graphical user interface

Description automatically generatedGraphical user interface, text

Description automatically generated

Diagram

Description automatically generated**Use Case Diagram**

**Testing**

# Software-Testing

The software can be tested with suitable Unit tests. There are test frameworks for Python as well as JavaScript which covers all our used languages. The unit tests can be used to ensure all coded methods work as intended and help by finding bugs or improve the ability to add new features later more simply.

# Hardware-Testing

Once we have a running prototype the accuracy and measurements of the sensors need to be checked. In order to do that we track the data provided by the sensors and check whether the data is plausible. For example, someone can test lift with the prototype and someone else will measure the lifted height by hand and compare the results to the measurements of the ultra-sonic sensor.

We should also test the actual battery capability. How long it will last, how durable the setup is and how the sensors react do different unpredicted movements. For example, dropping the handle, carrying the handle sideways, spinning it etc. We need to make sure that the system does not break when doing unexpected things.

In order to get a good overview, it is necessary to work on a decent test concept which tracks all information and the results of the tests for further improvements and easy access.

# User-Testing

When the initial testing is done, and we have a running prototype we need to make sure to let other people from our different target groups test our app in a real scenario.

We could ask friends or just people and trainers in the gym to try out our app. This not only helps to resolve problems me might not have noticed yet, but it also makes gathering feedback to the overall usage of our app way easier. Everyone should be given a decent questionnaire to give feedback about the usage, design, and functionality of our app. This helps to get a real user perspective from all target groups on our app. Only through the perspective of different users we can spot missing features we did not think about and find problems we on our own had never seen.

People that should test our app in this state are experienced lifters, casual gym users, people who never lifted before and trainers.

**MoSCoW**

* **Must** - Accelerometer and Ultrasonic sensor data i.e., time, x and y co-ordinates. Feedback on lift
* **Should** - Drawing a live graph with data, saving after each rep. Use accelerometer to calculate tilt on bar.
* **Could** - Use heartbeat sensor to track heartrate/pulse during rep. Have a real-time graph from data. Have screen display in front of the user wirelessly.
* **Would** - Make mobile app. co2 sensor to measure air quality.