# My Pain Manager Beta Release

Bas Peden: 5471490
Harrison Ellerm: 4862241
Hussam Al-zahrani: 8087364

During the time period intervening the alpha most of the effort has been spent on creating functionality between the homepage, database and summary page. User input data has been reworked, the database has been reformatted, continuous integration has been implemented, homepage aesthetics have undergone a makeover and unit tests have been set up. The summary page is also a new feature which has been added. It hosts graphing functionality allowing the user to see a visual representation of their pain levels over a given time period.

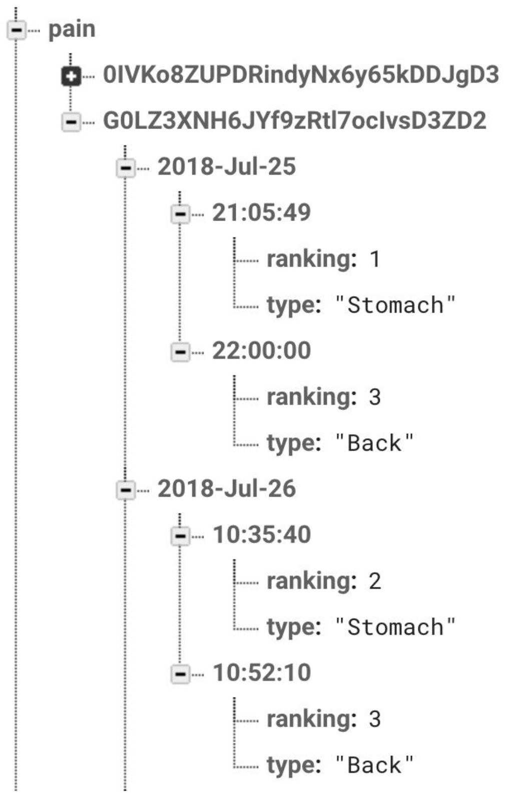
## → User Input

User input data has been reworked. Users now have an extra option to log their general lethargy levels. This gives users with chronic fatigue an option instead of graphing symptoms associated with individual body parts. When users are logging their data time interval accuracy has been increased to seconds. This allows for a greater precision when logging data over smaller time periods so that entries are not overwritten if the user logs data in quick succession. An area for notes/medication has been added so that when a user logs pain they can add any additional comments necessary information which will also be stored in the database.

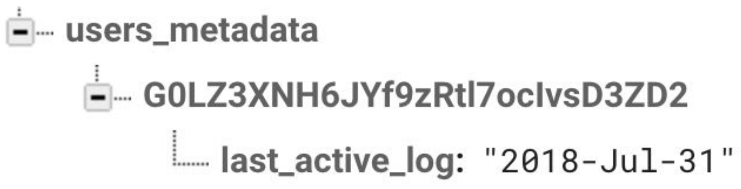
## → Database

The database needed to be re-formatted so that we could query the data we needed more efficiently. Instead of searching through "body area" nodes and then having nested "date" and "time" nodes to search through, we found that reversing this structure was a more convenient way to store the data. This way we could query between data ranges and the output was easier manipulate, given the context we needed for graphing.

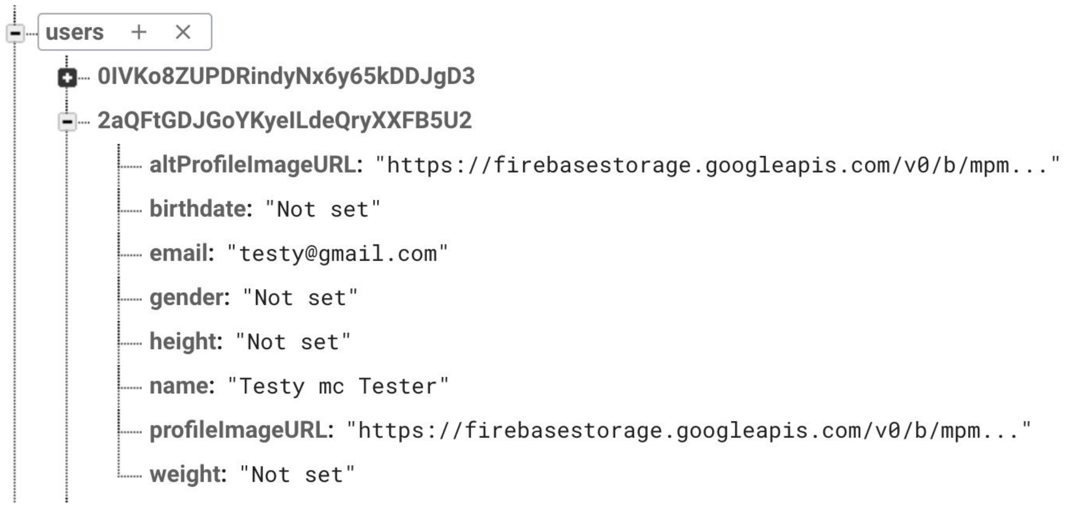
Storing metadata under a separate node was added so that we had a place to update the date when the user was last active. When the summary page is loaded, in the case that the user enters an end date that does not exist (i.e. they did not have data leading up to that date) it queries this node and will display data over the date range leading up to this date.



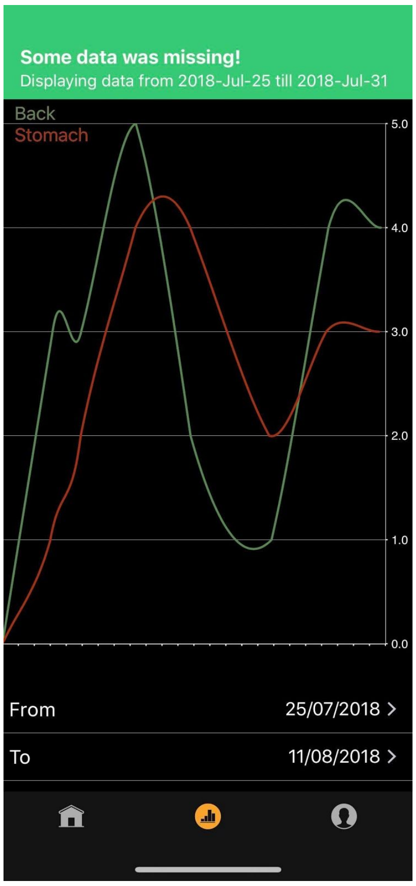
```
pain
  0IVKo8ZUPDRindyNx6y65kDDJgD3
  G0LZ3XNH6JYf9zRtl7oclvsD3ZD2
    2018-Jul-25
      21:05:49
        ranking: 1
        type: "Stomach"
      22:00:00
        ranking: 3
        type: "Back"
    2018-Jul-26
      10:35:40
        ranking: 2
        type: "Stomach"
      10:52:10
        ranking: 3
        type: "Back"
```


```
users_metadata
    G0LZ3XNH6JYf9zRtl7oclvsD3ZD2
        last_active_log: "2018-Jul-31"
```

User nodes were also added where profile information could be stored including their profile picture.


```
users
    0IVKo8ZUPDRindyNx6y65kDDJgD3
    2aQFtGDJGoYKyelLdeQryXXFB5U2
        altProfileImageURL: "https://firebasestorage.googleapis.com/v0/b/mpm..."
        birthdate: "Not set"
        email: "testy@gmail.com"
        gender: "Not set"
        height: "Not set"
        name: "Testy mc Tester"
        profileImageURL: "https://firebasestorage.googleapis.com/v0/b/mpm..."
        weight: "Not set"
```

## → Graphing


**Some data was missing!**
Displaying data from 2018-Jul-25 till 2018-Jul-31

From 25/07/2018 >
To 11/08/2018 >

Graphing was added to the summary page where the user can enter a “From” date and a “To” date and see their pain levels for different body areas graphed over the period. Currently it will graph data for every area that has data logged for it in the database. The plan is to have the user be able to select which body parts information they would like to have displayed. The pain values are also currently discrete values and we think continuous values would give a more accurate representation as well as removing the issue where lines consistently graph directly on top of one another.

Warnings have been added in cases where the user enters the wrong information. If a user enters a date range which does not have data in it a warning will be displayed and the dates will be truncated to show a time period in which data exists.

Colours for the lines are currently being randomly selected. In theory this could lead to multiple lines being indistinguishable from each other or from the background. Although practically this hasn’t been an issue yet, this may need to be changed.

## → Unit Tests

Unit tests have been added to check the graphing logic. As this is one of the most complex operations we needed to make sure that the x values were being calculated correctly for each log, and scaled according to the “period” they fell within.

For example:

```
0                                   1                                   2                                   3
|-----------------------------------|-----------------------------------|-----------------------------------|
25th                               26th                                27th                                28th
```

Consider the above timeline. Within this timeline there are four “periods”, associated with each of the dates. If a log fell on the 25th, at any point throughout the day, it’s time should be unadjusted. If it fell at any point on the 26th, it should be scaled by 24 hours, on the 27th by 48 hours and so on. The unit test written for this ensures that we see this behaviour, and is essential as we may adjust the logic in the future and need to make sure we don’t break this functionality.

We had hoped to have more unit tests written for other functionality within the app, and Hussam was incharge of this. Unfortunately, because we are using a realtime database it would mean we would have to mock the database, which none of us are familiar with, nor had time to investigate between the alpha and beta.

## → Continuous Integration

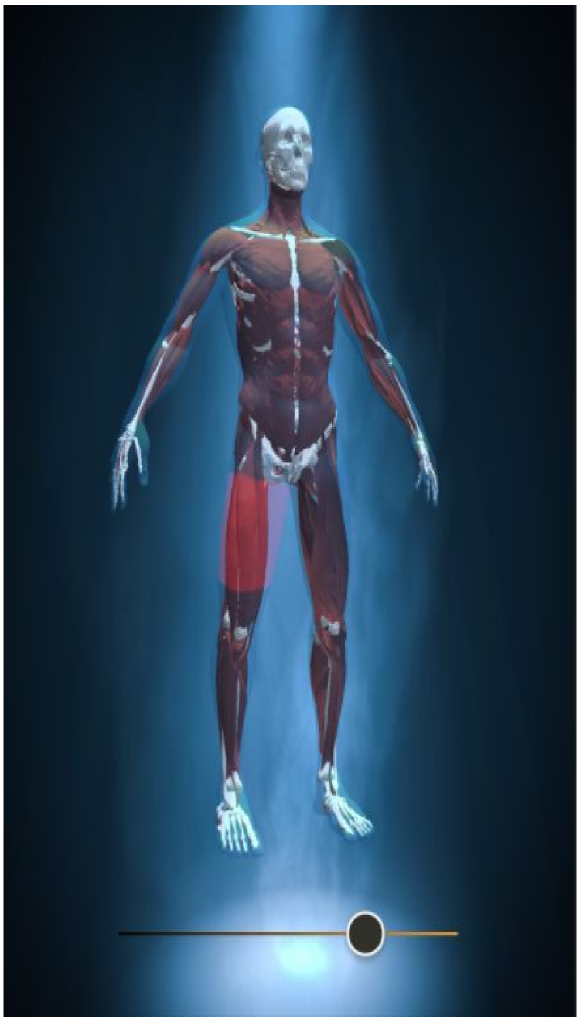
The road to setting up continuous integration was definitely not an easy one. Initially we were using Travis (successfully) to compile, build and run our unit tests, but unfortunately when one of our libraries was updated, Travis refused to build and compile it (regardless of the fact it built and compiled fine locally). We spent several hours trying to fix this issue, including downgrading to the previous version of the library, downgrading to an older version of Xcode, but unfortunately to no avail.

It eventually got so frustrating and was eating up so much development time that we decided to rip out Travis all together, and implement another form of CI, CircleCI (https://circleci.com/). Although this still wasn’t as straight forward to setup as you would think (various settings inside the project and its schemes had to be configured), we eventually managed to get the project building and run our unit tests!

## → Aesthetics

The home page has undergone an aesthetic makeover. The skin shader has been changed to have transparency fall off relative the polygonal normals. The lighting has been adjusted to strong overhead lighting with a blue hue.

The background has been changed to give an alien abduction vibe.



Several muscles have been adjusted to reduce awkward positioning and protrusion.

Bugs have been fixed where body areas were not being responsive to a single click.

Highlight areas and colours have been adjusted to match the new transparency level.

Ambient lighting has been reduced dramatically to increase shadow contrast.

## → Conclusion

Although from the users perspective it doesn’t feel like we have got a lot done, we are happy with how much has changed under the hood in terms of functionality and structure. In terms of teamwork the group has worked relatively cohesively together without any complaints or quibbles and we have managed to achieve the majority of what we had planned to accomplish.  With regards to group engagement and effort everyone contributed to the project.

## → macOS Install Script

Harry has written a script that should make cloning and building the project a breeze. It uses GIT and Homebrew (https://brew.sh/) to install GIT LFS (https://git-lfs.github.com/) if it is missing.

1. Create a new file in the folder you wish the repository to be cloned into (using your favourite editor).

```
$ nano installation_script
```

2. Copy and paste the following into the editor that appears and save it.

```
#! bin/bash

echo "Checking if Homebrew package manager is installed..."
if ! [ -x "$(command -v brew)" ]; then
echo -n "Homebrew not installed, install? (y/n)? "
read response

if [ "$response" != "${response#[Yy]}" ]; then
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
else
exit 1
fi
fi
echo "Homebrew installed"
echo "Performing GIT LFS install if necessary..."
brew install git-lfs
echo "GIT LFS installed"
echo "Cloning repository & ignoring LFS files"
GIT_LFS_SKIP_SMUDGE=1 git clone
https://github.com/HarrisonEllerm/COSC345_SoftwareEngProj.git
cd COSC345_SoftwareEngProj
echo "Setting up LFS Hooks in repo and pulling LFS files"
git lfs install && git lfs pull
cd ..
echo "Installation finished OK"
```

3. Turn the file into an executable:

```
$ chmod 700 installation_script
```

4. Run the executable (note, not ./installation_script as the script itself needs to change directories while executing):

   ```
   $ sh  installation_script
   ```

5. You should be good to go, open Xcode and find the .xcworkspace file within the folder that was created when you clone the repository. If you at some point wish to uninstall GIT LFS then you can run:

   ```
   ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/uninstall)"
   ```