**Thomas Flynn**

**G00291875**

**01/10/15**

**Sprint 2: Week 1 Log**

**Update:**

My supervisor advised me to get an interactive GUI layout manager for designing and programming my Java Desktop Application. One where I can drag and move components and code at the same time. I settled on the Netbeans IDE. I have still yet to familiarize myself with this IDE but from what I've seen in videos online, it seems to be what I'm looking for.

I started Version Control of my Music Player GUI on Github.

[Link to Version Control on Github](https://github.com/g00291875/FYP-GUI)

I purchased a Bluetooth Dongle and successfully paired my phone to my laptop. I ran into various problems with this, which ended up eating up a lot of my time.

For developing my Android platform I downloaded the highly recommended GenieMotion Emulator. With the standard AVD that comes with IntelliJ, I was unable to run it on my own laptop due insufficient RAM. However GenieMotion seems to work reasonably well on my laptop which is a big improvement on having no emulator at all.

For integrating my database with my Desktop Application I did some research on the singleton Design pattern. I'm researching design patterns to prevent running into problems later on down the line. My project will be implementing threads at a later point, so creating a thread safe Project from the very beginning is an integral part of good software project planning practice.

I started following a tutorial video on the JDBC driver for connecting to mySQL database.

[Link to tutorial](https://www.youtube.com/watch?v=8-iQDUl10vM&list=PLEAQNNR8IlB4R7NfqBY1frapYo97L6fOQ)

This week I also started tracking my time on Toggl.com which has proved useful for documentation.

I probably won't be able to keep up this level of detail in my logs for the subsequent weeks as it's difficult to delegate time for documentation. I hope My weekly screenshots of my **Trello** Project management System and **Toggl** time log will suffice as weekly logs.

**Research done:**

**Singleton Design Pattern:**

With the **Singleton design pattern** you can: Ensure that only one instance of a class is created. Provide a global point of access to the object. Allow multiple instances in the future without affecting a **singleton** class's clients.

There are 2 ways of instantiating a database using the Singleton design pattern.

**New method:**

public static Database getInstance() {  
 return *instance*;  
}

This is the preferred way of instantiating a database

**Old method: "lazy instantiation"**

private static Database instanceOld;  
  
public static Database getInstanceOld() {  
 if(instanceOld == null) {  
 instanceOld = new Database();  
 }  
   
 return instanceOld;  
}

The old method is not thread safe.

One thread could call get instance and then another could also do the same before the first thread finishes creating it.

**Beans:**

has a no argument constructor. lightweight class that simply has get and set methods

**Tasks completed:**

- Installed Netbeans IDE

- Purchased Bluetooth Dongle

- Download Bluetooth Drivers

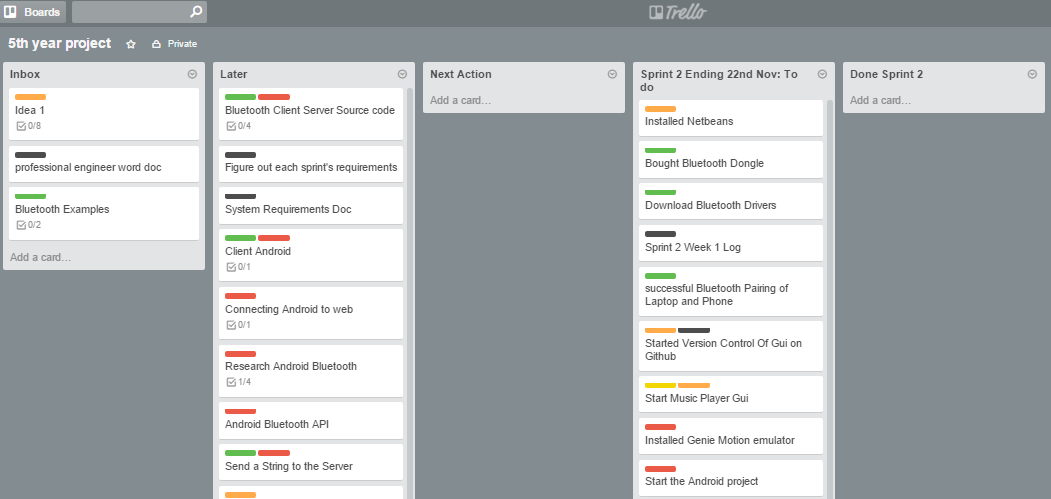
- Paired phone and laptop through bluetooth

- Start Music Player GUI

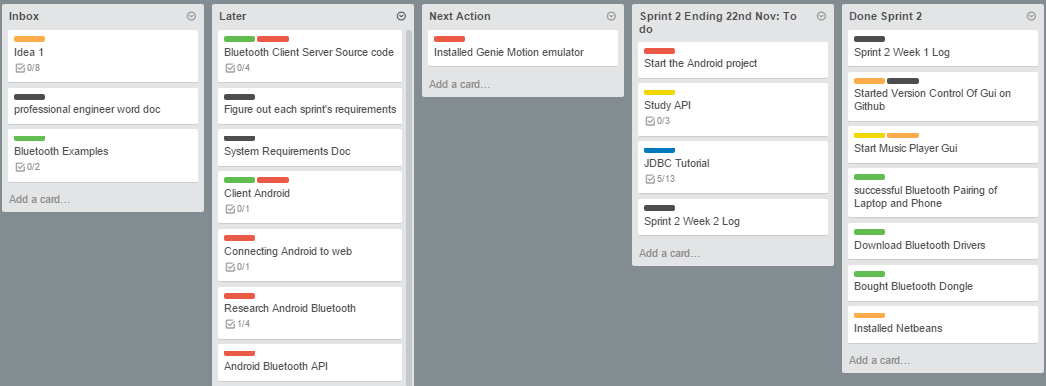
- Started Version Control of Music Player GUI on Github

- Installed GenieMotion Emulator for Android Development

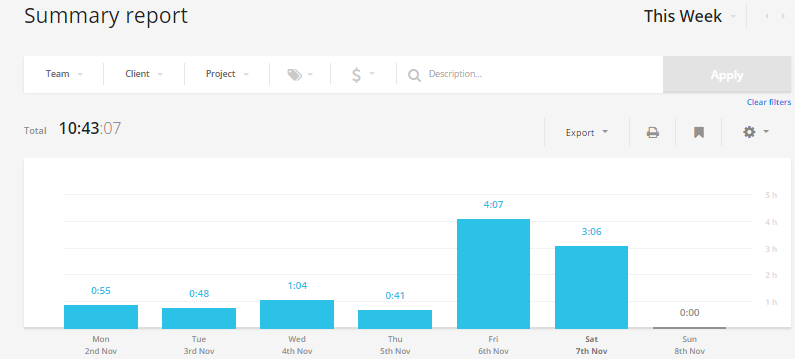
**Project Board Start of week**

****

**Project Board End of week**

****

**Weekly Time Bar Chart:**

****

**Toggl Time Log:**



**Pie charts of time allocated to each area**

