****

**Computer & Electronic Engineering**

**Final Year Project "Music Host Interface"**

**Sprint 6: Week 2 Log**

**Thomas Flynn**

**Project Supervisor: Brian O'Shea**

**29/02/16 - 06/03/16**

**Entry 1: 29/02/16**

Today I worked on putting the byte array queried from the database into my Song class object.

I also managed to display the list of songs in a listView on the FX GUI.

**Entry 2: 01/03/16**

Today I worked on implementing a progress bar for the download songs thread. I managed to do this buy hard coding the init method to download 5 songs. After each iteration the progress bar updates 1/5th of the way. This took longer that I would have liked to get working. The solution was to use Platform.runlater() within the task to update the GUI.

I also added a Queuelist which is separate from the selection list. The list Song objects gets initialised first in the constructor with just the names of the song and artist as well as their database ID.

When the user clicks the "add song to queue" button it copies the song over to the Queue list and then downloads the byte array associated with it's ID from the database. When the song object is in this state it cannot be played just yet. It will stay in this state until it is the next song to be played (or next to next song). When this event fires, it will perform IO and create a temp file for itself from the byte array that it holds. This is necessary as FX media player objects can only take a file URI into it's constructor. I have researched this online to confirm that there is no workaround except for the implementation that I am suggesting here.

The above paragraph is a partial summary of what I completed today. I have all the functionality to accomplish the design suggested I just need to work on the finer details.

**Entry 3: 02/03/16**

Today I thought about how I might go about defining my interface for my FX GUI. I also worked on adding components to the DJScreenController. I spent far too much time trying get the DJ screen to update along with the MainScene controller. I should have done some much needed research on threading beforehand. I also started looking at how I will implement JSON arrays into my code.

I also added a feature where when the user clicks the skip song button. It updates the GUI by removing the song object from the list of song objects called QueueList. I want add this feature to an "onEndOfMediaPlayer" listener when I get the chance.

**Entry 4: 03/03/16**

Today I read some chapters of the book Java Concurrency in Practice by Brian Goetz. The book is written very well. It has a lot of "don't do this examples" and explains clearly why the implementation is bad. Threading is still way over my head so I am going to gravitate towards higher priority tasks first and then hopefully implement some elements of thread safety into my code in my final sprint.

**Entry 5: 05/03/16**

I moved my server object from the Mainscene Controller into my model Object. I also worked on parsing a hardcoded JSON string on my Android application.

I parse the JSON on my Android Application by hard coding it in the main activity and then putting it into a bundle to be processed by the second activity. The second activity receives the bundle and performs a doinbackground thread to process the JSON and updates the listview into the GUI on Activity 2. This took much longer than I thought it would.

**Entry 6: 06/03/16**

Today I continued to work on moving the Server object from the main scene controller into the model for my FX application.

Rather than connecting my Android phone to the server each time to test JSON parsing I went directly to the thread on the Android application where it processes the input stream.

I implemented a test to check if the first char in the hardcoded String is a '['. (Which will be my JSON array). I then use "sendhandler(a,b)" and pass it the arguments int WhatToDo and msg.Obj. The handler that is embedded into my main activity receives this msg.Obj and sends it to the second activity to be processed, depending on the whatToDo varaible.

This determination of what the first char is from the input stream will be my basis for connection protocol. I'm aware a better implementation is possible. But considering the deadline, this is how I will do it.

Overall I'm pretty happy with the progress I made this week. My Android application now successfully receives the JSON from the music host with actual values from the SQL database and displays it in the listview on activity 2.

I hope I can keep up this level of productivity going into my final 2 sprints.

**Task completed:**

Song Object holds byte array

simplify add song method

Songs appear in GUI and can be added to queue

Download song progress bar

Queue list removes song using skip button

Song objects getting initialized with SQL data.

Separate song queue and song list.

Defining interface for both controllers.

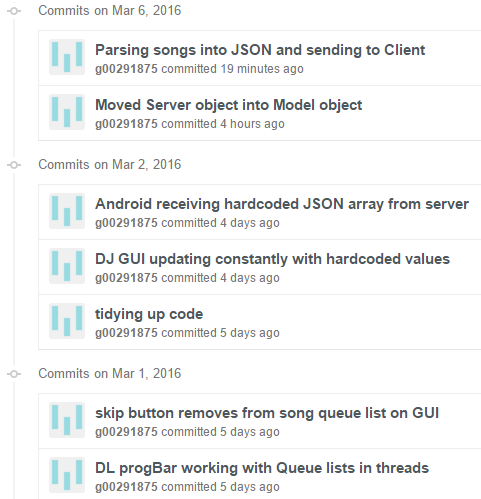
Android parsing hardcoded JSON array.

Song from database into JSON array

FXGUI sends JSON array successfully

**GIT Repositories:**

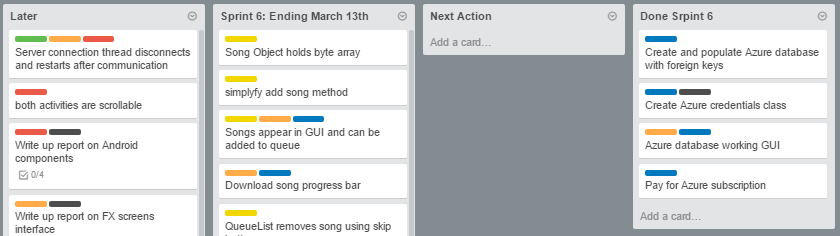
**Repo: FYP-GUI**

****

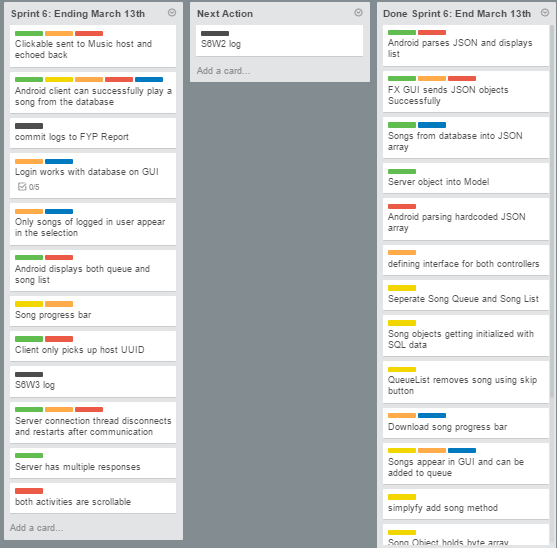
**Repo: FYP-Android**

****

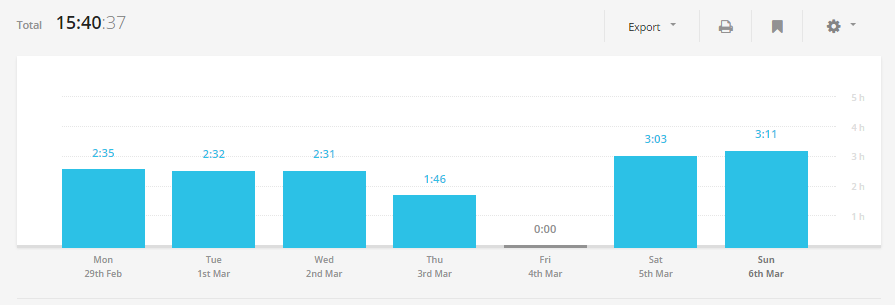
**Board at the start of the week:**

****

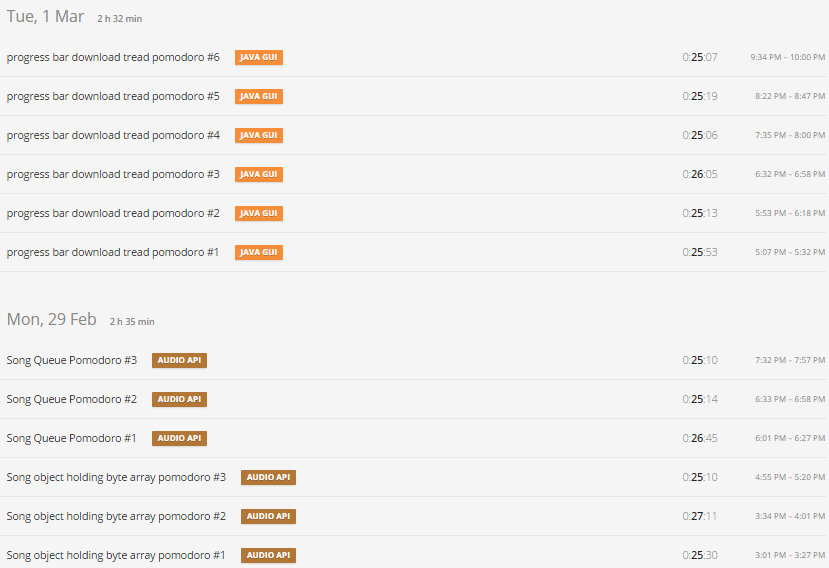
**Board at the end of the week:**

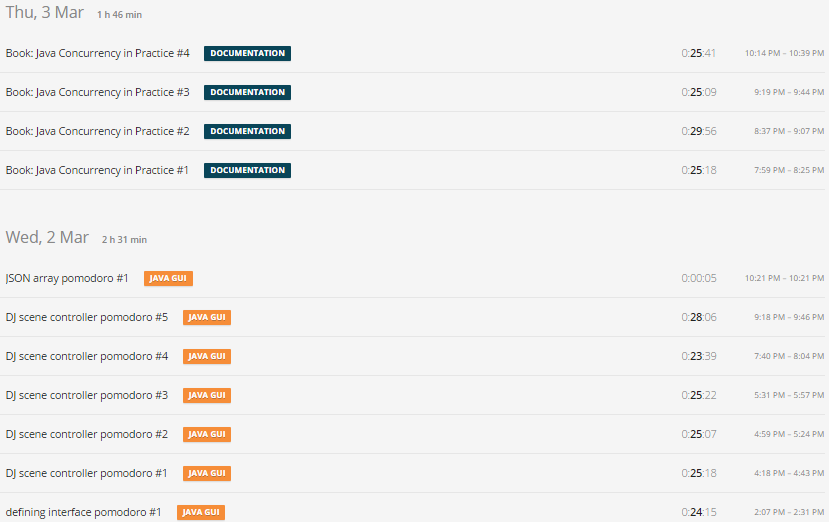
****

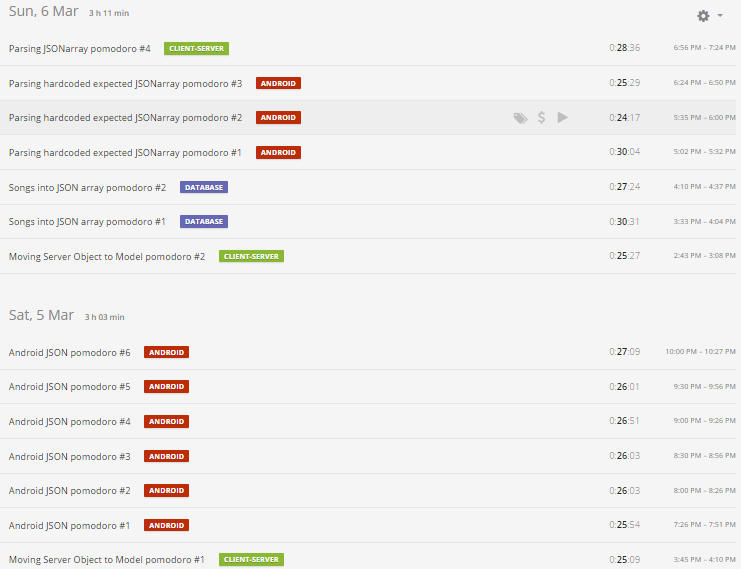
**Weekly time Log bar chart:**

****

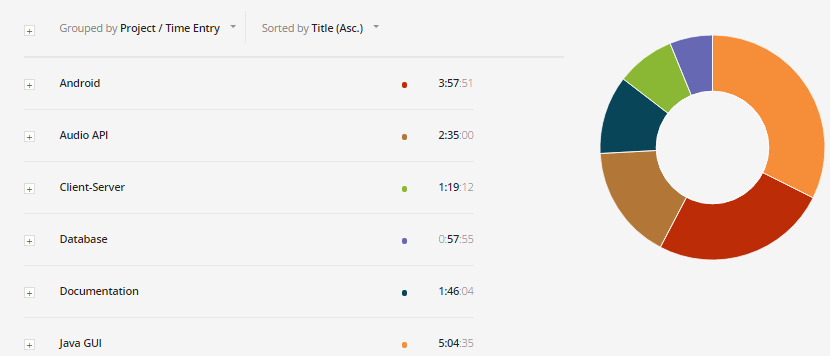
**Weekly Time Log:**

****

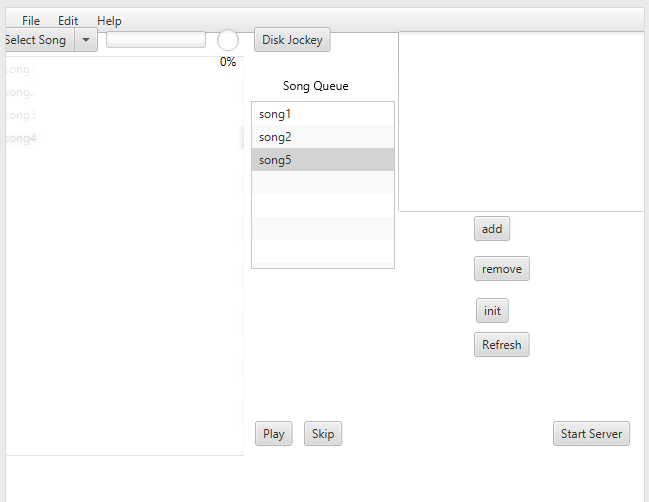
****

****

**Weekly log Pie Charts:**

****

**Screenshots progress:**

****