

**Mood Streamer**

**Final Year Project Report**

DT228

BSc in Computer Science

**Matthew O’Neill**

**Supervisor: Paul Doyle**

School of Computing

Dublin Institute of Technology

**Thursday, 26th March 2015**



**Abstract**

Music has the power evoke strong emotions in the listener. From a melancholic piano piece which helps the listener through difficult times, to an uplifting pop song which evokes happiness and makes the listener want to dance, music shapes how we feel daily.

This project’s goal is to develop an application which will analyse the music collection of its users and categorize the tracks by the emotional traits that it assesses they contain. This will enable the user to select how they are feeling and be played a selection of their music collection which matches that mood.

The application was developed as an Android Application which streams the user’s categorized music collection from the web service which was developed to store and analyse the user’s music collection.

**Declaration**

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Matthew O'Neill

26/03/2015

**Acknowledgments**

**Contents**

[1 Introduction 9](#_Toc414633327)

[1.1 Project Aim 9](#_Toc414633328)

[1.2 Challenges Faced 9](#_Toc414633329)

[1.2.1 Introduction 9](#_Toc414633330)

[1.2.2 Subjectivity of Music 9](#_Toc414633331)

[1.3 Structure of this Document 9](#_Toc414633332)

[1.3.1 Section 2 – Research 9](#_Toc414633333)

[1.3.2 Section 3 – Design 9](#_Toc414633334)

[1.3.3 Section 4 – Architecture and Development 10](#_Toc414633335)

[1.3.4 Section 5 – System Validation 10](#_Toc414633336)

[1.3.5 Section 6 – Project Plan 10](#_Toc414633337)

[1.3.6 Section 7 – Future Work 10](#_Toc414633338)

[1.3.7 Section 8 – Conclusion 10](#_Toc414633339)

[2 Research 10](#_Toc414633340)

[2.1 Alternative Existing Solutions 10](#_Toc414633341)

[2.1.1 SensMe 11](#_Toc414633342)

[2.1.2 Moodagent 11](#_Toc414633343)

[2.1.3 StereoMood 12](#_Toc414633344)

[2.1.4 Conclusion 12](#_Toc414633345)

[2.2 Technologies Researched 12](#_Toc414633346)

[2.2.1 Mobile Development Platform(s) 12](#_Toc414633347)

[2.2.2 Android Emulators 13](#_Toc414633348)

[2.2.3 Database Selection 13](#_Toc414633349)

[2.2.4 Version Control Selection 14](#_Toc414633350)

[2.2.5 Storage Options 14](#_Toc414633351)

[2.3 Relationship between Music and Mood 15](#_Toc414633352)

[2.4 Musical Features Which Relate to Mood 15](#_Toc414633353)

[2.4.1 Introduction 15](#_Toc414633354)

[2.4.2 Tempo 15](#_Toc414633355)

[2.4.3 Modality 15](#_Toc414633356)

[2.4.4 Key 15](#_Toc414633357)

[2.4.5 Use of Staccato 15](#_Toc414633358)

[2.4.6 Timbre 15](#_Toc414633359)

[2.4.7 Lyrics 15](#_Toc414633360)

[2.4.8 Summary 15](#_Toc414633361)

[2.5 Extracting Relevant Features from Music 15](#_Toc414633362)

[2.5.1 ISMIR Conference 15](#_Toc414633363)

[3 Design 15](#_Toc414633364)

[3.1 UI Design 15](#_Toc414633365)

[3.1.1 Logo 15](#_Toc414633366)

[3.1.2 Representing Mood 16](#_Toc414633367)

[3.2 Technical Architecture 16](#_Toc414633368)

[3.2.1 Previous Technical Design 16](#_Toc414633369)

[3.2.2 Choice of Three-Tier Architecture 17](#_Toc414633370)

[3.2.3 Drawbacks 17](#_Toc414633371)

[3.3 Designing a RESTful API 18](#_Toc414633372)

[3.4 MVVM 18](#_Toc414633373)

[3.5 Methodologies 18](#_Toc414633374)

[3.5.1 Use of Source Control 18](#_Toc414633375)

[3.5.2 Adhering to Coding Guidelines/Standards 18](#_Toc414633376)

[4 Architecture and Development 18](#_Toc414633377)

[4.1 Programming Languages Employed 19](#_Toc414633378)

[4.2 Development Environment 19](#_Toc414633379)

[4.2.2 Language / Framework 19](#_Toc414633380)

[4.2.3 Database 19](#_Toc414633381)

[4.2.4 Development Virtual Machine 19](#_Toc414633382)

[4.2.5 Operating System 20](#_Toc414633383)

[4.2.6 Deployment 20](#_Toc414633384)

[4.3 Web Service 21](#_Toc414633385)

[4.3.1 Handling Uploads 21](#_Toc414633386)

[4.3.2 Organizing Files 21](#_Toc414633387)

[4.3.3 Developing a Model 22](#_Toc414633388)

[4.3.4 Selection of Tracks 22](#_Toc414633389)

[4.4 APIs Utilised 22](#_Toc414633390)

[4.4.1 Echonest 22](#_Toc414633391)

[4.4.2 ChartLyrics 22](#_Toc414633392)

[4.5 Third Party Modules and Libraries Used 22](#_Toc414633393)

[4.6 Overview of Code Written 22](#_Toc414633394)

[5 System Validation 25](#_Toc414633395)

[5.1 Testing 25](#_Toc414633396)

[5.1.1 Usability Testing 25](#_Toc414633397)

[5.1.2 Comparison with Existing Systems 25](#_Toc414633398)

[5.1.3 User Testing 25](#_Toc414633399)

[5.1.4 Automated Testing 25](#_Toc414633400)

[5.2 Summary 26](#_Toc414633401)

[6 Project Plan 26](#_Toc414633402)

[7 Future Work 26](#_Toc414633403)

[7.1 Web Frontend 26](#_Toc414633404)

[7.2 Securing the Service 26](#_Toc414633405)

[7.3 Improving the Audio Analysis 26](#_Toc414633406)

[7.4 Python 3 Support 27](#_Toc414633407)

[7.5 Better Support for Classical Music 27](#_Toc414633408)

[7.6 Tablet and Landscape Layouts 27](#_Toc414633409)

[7.7 Expansion to Other Platforms 27](#_Toc414633410)

[8 Conclusion 28](#_Toc414633411)

[9 Bibliography 29](#_Toc414633412)

**Table of Figures**

[Figure 1 Technical Architecture Diagram 15](#_Toc414578419)

[Figure 2 Part of a python function which adheres to pep8 and pep257 16](#_Toc414578420)

[Figure 3 Overview of the DigitalOcean VPS 18](#_Toc414578421)

# Introduction

## Project Aim

The aim of this project is to develop a software system that will allow the user to select a mood or degrees of multiple moods that they are currently experiencing by inputting this choice in a mobile application. The user will then be played music that the system has deemed to be matching this mood. Many music listeners create playlists of music categorized by how they are feeling, for example they might create a ‘Rainy Day’ playlist containing pieces of music that evoke in them the emotions of a rainy day. This application will automatically create mood based playlists. Should the user disagree with a tracks categorisation, he or she is offered the option of providing feedback as to how they think it should be categorized, influencing the categorization for both that user and all other users (to a lesser extent) on future uses of the application.

## Challenges Faced

### Introduction

Over the course of this project many challenges had to be overcome in order to reach the desired result.

### Subjectivity of Music

Music is a particularly subjective topic; a piece of music may have particular meaning to a listener, such as having been played during a particularly difficult period. Therefore it is necessary to not simply rely on the system’s analysis of a piece of music but also include the option for the user to provide feedback which changes the rating of the track in future should the categorisation not match their expectation.

## Structure of this Document

This document is structured into 8 sections (including this introduction); they are as follows:

### Section 2 – Research

This section explores the research undertaken before and during development of the system. It first details existing solutions and systems in similar areas to this project as well as outlines pros and cons of these systems relative to this project.

This chapter then outlines the technologies researched before development began, as well as the justification for the choices made in this area.

### Section 3 – Design

This section outlines the design methodologies employed in this project. It also looks at the UI design approach taken and compares the approach taken with other possible approaches.

### Section 4 – Architecture and Development

This section outlines the overall technical architecture of the system and documents the development of the system that has taken place over the course of this project.

### Section 5 – System Validation

Validating the completed system was a major part of this project; section 5 attempts to outline this validation. Validation consisted of comparisons with existing system as well as user testing, automated and unit testing.

### Section 6 – Project Plan

This section documents the planning stage of the project and evaluates how well the final system stuck to that plan as well as identifies areas in which it differed.

### Section 7 – Future Work

This section outlines some possible future work that may be done on the project, including additional features that may be implemented and improvements to existing features and user interfaces.

### Section 8 – Conclusion

This final section attempts to sum up the outcome of this project and offers speculation as to whether the project was successful.

# Research

## Relationship between Music and Mood

## Musical Features Which Relate to Mood

### Introduction

Music theory is the study of the elements which comprise music. The Cambridge History of Western Music Theory describes the field as asking the question “what is the essential nature of music?” [5]

This subsection outlines research performed in order to identify which elements of music relate to emotions in music and which do not.

### Tempo

### Modality

### Key

### Use of Staccato

### Timbre

### Dynamics

### Lyrics

While not a feature present in all pieces of music, the content of a song’s lyrics can affect our emotional response to that piece of music.

#### Sentiment Analysis

### Summary

In conclusion, in order to produce an effective mood detection model, many of the features outlined above must be acquired about the music added to the system, either by direct analysis, an existing knowledge base, or a combination.

## Extracting Relevant Features from Music

Music Information Retrieval (MIR) is the study of extracting information from the characteristics of music and is a widely researched area in both computer science as well as music studies and psychology. Information extracted from the

### ISMIR Conference

The International Society for Music Information Retrieval is “a non-profit organisation which, among other things, oversees the organisation of the ISMIR Conference. The ISMIR conference is held annually and is the world's leading research forum on processing, searching, organising and accessing music-related data.” [6]

## Alternative Existing Solutions

When evaluating similar existing solutions in the area of mood recognition music players, several criteria were used to assess the feasibility of developing a system in this field.

Some existing systems perform mood analysis only on the tracks found on the user’s device, using similar techniques to ones which are to be employed on this project: tempo and beat detection, key recognition and pitch analysis. This localized approach has the downside of limiting the tracks a user may listen to those on the device; this in turn is limited by the relatively low storage capacity found on many devices today. A further limitation of using only those tracks that reside on the device is the inability to learn from other music tracks. A sophisticated system in this area is one that can learn from other music and build ratings models from large amounts of music. As such, a system that learns from such a limited number of tracks might not be as accurate as one that can learn from a user’s entire music library, as well as the libraries of other users.

Other applications do not perform any analysis on the actual file to be played, but instead consult an existing online database of rankings for tracks. This method has the drawback of potentially trying to ascertain the mood of a track in a user’s library that has not been ranked by the system.

Other applications do not allow the user to use their own music collection at all; instead choosing music from an existing streaming service. This practice comes with some downsides, one of which is cost. Should the user be streaming music that has not been bought by them, it will usually need to be paid for. Other applications have circumvented this barrier by streaming from popular free music streaming sites such as SoundCloud and BandCamp, where the music artist allows their music to be listened to for free. These services suffer from a greatly reduced music selection as most record labels and musicians do not offer their work for free.

### SensMe

#### Introduction

SensMe is an application developed by SONY, which has been included on a selection of their MP3 players, smartphones, and games consoles since 2009. This application allows the user to transfer their music collection to the device. Once music has been added to the device, the software analyses a subset of the music according to such factors as beats per minute (BPM) and key. Once analysis has been performed on the tracks, the software visualises them as small white dots and scatters them on a four axis graph. These axes are labelled ‘Happy’, ‘Sad’, ‘Fast’ and ‘Slow’. Once the user touches a point on this graph, a playlist of tracks which have been deemed to match the labels of the axis/axes closest to the contact point.

#### Evaluation

A downside to the SensMe system that shall be addressed with this system is the storage limitations present on the devices on which it runs. As no music is stored on the client, but rather on the server ready to be streamed to a client device, much more storage space can utilised. Another advantage of this system over one which analyses local files only is that other users’ files can be analysed alongside one another, enabling the system to learn.

### Moodagent

#### Introduction

Another existing application, Moodagent, which can be downloaded from the Google Play store for Android devices and from the Apple App Store for IOS devices addresses the task in a slightly different manner. Instead of performing the analysis locally, the application consults a pre-existing online database of mood ratings for a track. This has the advantage of not being limited to learning from the relatively small set of music added by the user. The user is presented with a series of sliders labelled ‘Sensual’, ‘Tender’, ‘Happy’, ‘Angry’ and ‘Tempo’, and sliding these up or down adjusts the mood of the playlist of tracks to be played to the user accordingly.

#### Evaluation

Moodagent limits the track selection that can be played to those which are on the device, greatly limiting selection, it does however allow the user to hear new music similar to the current mood-based playlist and gives them the ability to buy it on services such as Amazon.com. It also limits the rating of a user’s music to those tracks which have already been rated and stored in the Moodagent database.

### StereoMood

StereoMood is an Android and iOS application as well as web-based music streaming service which, according to the company’s website “plays music tailored to your mood and daily activities” [1]. The music recommended by this service is aggregated from music blog postings and streamed using the popular SoundCloud music hosting platform.

### Conclusion

As a result of comparing applications that presently exist in the area of music players which play tracks based on their mood it was discovered that while a combination of such applications offer all the features of this project, one which offers all of them was not found.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **SensMe** | **Moodagent** | **StereoMood** |
| Streaming from the Web | No | No | Yes |
| Analyse local files before they’re played | Yes | No | N/A |
| Use user’s own music collection | Yes | Yes | No |

## Technologies Researched

Many decisions had to be made as to the technologies to employ for this project as it consists of three distinct software components that all need to communicate together effectively.

### Mobile Development Platform(s)

When it came to deciding on the mobile platform to develop the application, certain factors needed to be taken into account. These factors included:

* Platforms which run on devices currently accessible
* Cost of licences for those platforms
* Additional hardware required to develop for a platform (Mac OS is required to develop applications for iOS, for example)

It was decided that initial development will be done for the android platform as there is no cost to develop for it. Development took place using the Xamarin [2] toolkit for mobile application development. It was decided to use this technology as it allows for the development of an application for other platforms, namely Windows Phone and iOS further into the project. A decision had to be made whether to develop native applications for one or more platforms using the Java language for Android, Objective C or Swift for iOS and C# for Windows phone, or to use Xamarin to write for any of these platforms using the C# language and the mono runtime. It was decided to use Xamarin for a number of reasons. One such reason is that a shared code base can be used across all platforms for functionality which does not relate to the user interface; UI features are largely specific to the mobile platform, but all are wrapped in C# for uniformity.

An influencing factor in the decision to use the Xamarin platform was an existing familiarity with the C# language and the .NET framework, which the Mono framework attempts to emulate. This familiarity with these technologies will lead to increased productivity during the development of the mobile application, as well as cleaner, more idiomatic code.

While Xamarin affords the developer the opportunity to write mobile applications in the C# language, it is still necessary to become familiar with the APIs specific to each platform; Xamarin doesn’t re-implement these, it merely affords idiomatic C# access to them. As Android is the primary development platform at this stage of the project, some time was spent reading the android tutorials found on the developer.android.com website. These tutorials and supporting documentation primarily use the Java language, however given the similarity between the two programming languages, they proved and continue to prove highly useful in the development of the Android application for this project.

### Android Emulators

#### Google Android SDK Emulator

#### Genymotion

#### Debugging on a Physical Device

### Database Selection

In the planning stages it was decided that two databases were needed for this project; one for the web service and one for the Windows upload client, with a possible third should the settings facilities provided by the mobile platform not be sufficient to store the information required by the client.

#### NoSQL Databases

In recent years NoSQL (Not Only SQL) databases have seen an upturn in interest and adoption. One such database system is MongoDB. MongoDB is, according to its website [3], “a document database that provides high performance, high availability, and easy scalability”. Rather than interacting with this database using SQL, the developer or DBA instead uses mongo’s built in functions, such as find, findOne and insert.

Moving from a relational database to a NoSQL database such as MongoDB can prove a challenge to users used to the concept of joins.

#### Open Source Relational Database Systems

A more traditional data storage route is that of the Relational Database Management System (RDBMS).

PostgreSQL offers some advantages over MySQL, one such advantage is its support of multiple [4] procedural languages for the development of stored procedures and triggers, including python, and Java using a third party plugin.

Another advantage is that PostgreSQL offers many datatypes not found in its open source competitors, such as a JSON type for inserting and querying data found in Java Script Object Notation blocks.

#### File-Based Relational Databases

SQLite is a small, lightweight relational database that stores the entire contents of the database in a single file.

A downside to SQLite is its reduced concurrency support relative to a larger RDBMS, meaning it is not well suited to an environment which may have many simultaneous connections to the database at once, such as the database used by the web service for this project. While there would be no such difficulties with the low number of users the system has at present, it was decided that its use may cause difficulties should the application need to scale in the future.

#### Conclusion

It was concluded that a relational database be used on both the upload application and the web service. This was the case for the web service as much of the data to be stored by the service was thought to be of a relational nature, and it was seen to require a number of joins in queries. The database selected for this component was PostgreSQL due to the advantages laid out in the above comparison.

A relational database, SQLite, was also employed in the Windows upload client. This choice was not made primarily due to its being a relational database, but rather because it is a lightweight, single file-based database, meaning it can easily be packaged with the application, without the need to install a database server or connect to one remotely. The database file can be stored in the AppData directory found on windows on first run of the application and accessed again on each subsequent run.

### Version Control Selection

Version control is an invaluable tool for software developers

#### Distributed Version Control vs Centralized Version Control

There are two prevailing styles of Source Control systems: Distributed and Centralized.

#### Conclusion

It was decided to use the git version control system due to the features it provides above that of SVN outlined above, as well as an existing familiarity with this particular tool. GitHub was chosen to remotely host the code repository for the project as it provides very useful features such as issue tracking, as well as private repositories, five of which are free for use by students for educational purposes.

### Storage Options

# Design

## UI Design

### Logo

Due to a lack of design skills, it was decided to outsource the design of a logo for the application to a third party. The website fiverr [7] was used to contract a designer to design a logo which could be used on the login screen of the application, as well as the icon used on the application launcher to identify the application.

### Representing Mood

It was necessary to investigate methods of conveying mood to the user and allowing them to input how they are feeling

#### Grid

#### Colour

It was decided to represent the user’s selection of their current mood by using colours to differentiate different feelings. Much research has been undertaken in the area of correlating human emotions to different colours. According to a study from Columbia University regarding our emotional response to colouring found in film [8], red is often associated with “…, Hatred, Life, Noble” feelings, while blue is often associated with “peace” and “tranquillity”. Black, which was described as “indefinite”, was selected for use in this project due to its achromaticity, to symbolise neutrality. This lets the user see they have not selected any mood yet. White could also have been used here, however it was decided that black matched the existing aesthetic of the application more appropriately.

Red and blue, therefore, were selected to represent aggressiveness and calmness respectively.

## Technical Architecture

### Previous Technical Design

#### Standalone Application

Initially it was decided to develop the project as a standalone mobile application whereby the files located on the device would be analysed, rankings stored in a databases on the mobile device and files played directly from local storage.

It was decided that this approach did not provide enough novelty; this approach has been taken before by many similar applications. Implementing the project in this way also means that only the music found on the (usually very small) storage may be used where as many people’s music collection is often much larger. This option also reduces the possibility of pooling suggested improvements to the track analysis from the user; a server would be required to coordinate the suggestions among all the devices.

#### Web Application Only

Another planned technical design that was considered was implementing the system as a web application; removing the native mobile application and replacing it with a web page the user visits when they want to listen to their music. This option has the benefit of being accessible on any internet browser-equipped device by utilising a modern, responsive design.

Mobile usage has now overtaken desktop computer usage [9], and the experience on mobile would suffer with this method; web pages to not get the same priority from a mobile operating system and tabs may be killed to free memory, interrupting music playback. Moreover, some mobile platform cease audio output from a browser tabe when that tab looses focus.

### Choice of Three-Tier Architecture

It was decided to break the system into three separate components which each have a separate task

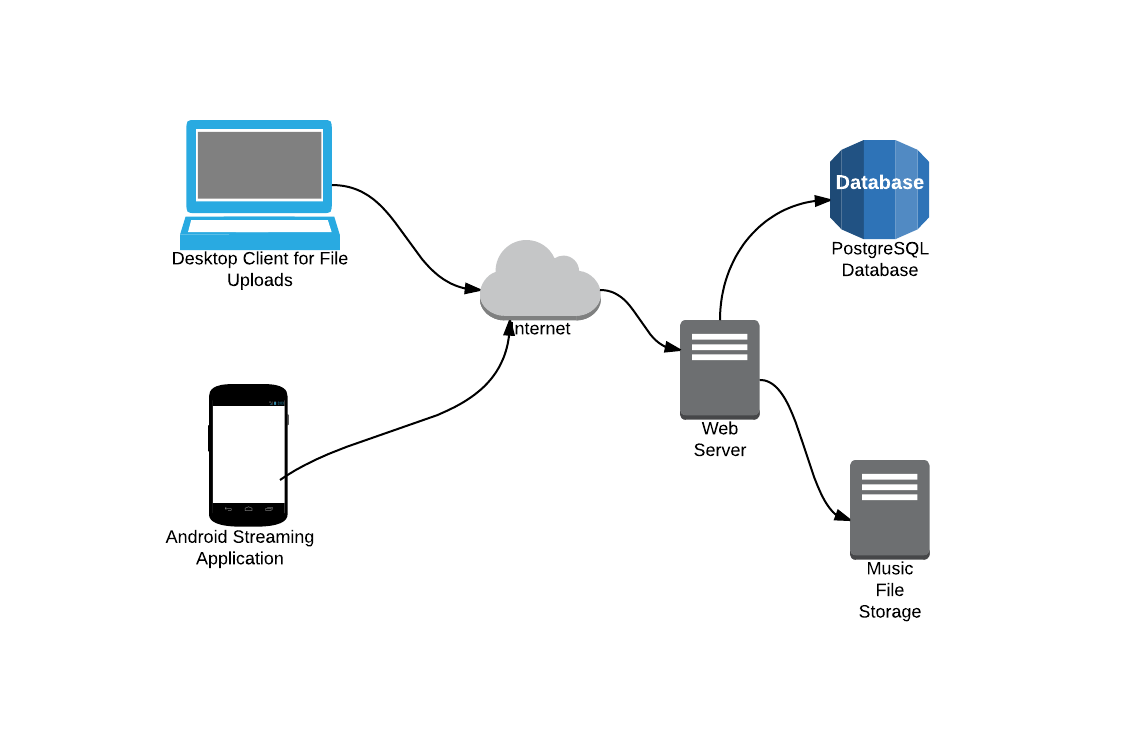


Figure 1 Technical Architecture Diagram

An influencing factor in the choice to separate the music analysis from the client was that by having the analysis algorithms and the resulting analysis data reside on the server as opposed to the client, the initial analysis of a track only had to be performed once; subsequent uploads of the same track could be discarded and the previous analysis and file could be used. This also means that the users can provide feedback on the analysis of a particular track which would improve the quality of track selection relative to mood for all users with that track in future.

### Drawbacks

Using native applications for each platform increases the workload of developing the project. Any user of a platform that an application has not been developed for is unable to use the system.

Desktop only users are also left out here as they are unable to run mobile applications. A solution here may be to develop a Windows Universal application [10], this would enable users of both Windows Phone and Windows 8 and above to run the same application.

Adding a web frontend as another client could also improve this technical architecture as Desktop users from all Operating Systems would be able to use the system.

## Designing a RESTful API

## MVVM

## Methodologies

### Use of Source Control

Source control played an integral part in the development of this project.

### Adhering to Coding Guidelines/Standards

For readability and uniformity it was decided to follow the respective guidelines and standards of the two development languages used, C# and Python.

#### PEP-8

Python Enhancement Proposal (PEP) No. 8 [11] is the de-facto python style guide. Written in part by Guido van Rossum, Python’s creator, the guidelines outline recommends such practices as using spaces rather than tabs for indentation, limiting line length as well as naming styles. There is also a PEP8 software tool which takes as input python source code and identifies where the source does not conform to the standards. This tool was incorporated into the Sublime Text editor during development to ensure constant conformance.

Similarly, PEP 257, Docstring Conventions [12] outlines best practices for documenting

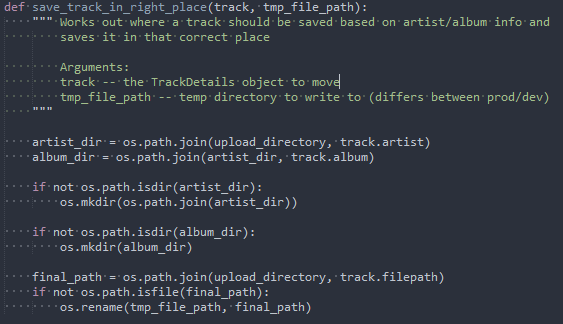


Figure 2 Part of a python function which adheres to pep8 and pep257

#### C# Coding Conventions

# Architecture and Development

The system is developed using a multi-tier architecture consisting of:

* Web Service
* Client Application
* Desktop Upload Utility

## Programming Languages Employed

## Development Environment

#### IDE and Text Editors

Three editors were used in the development of this project: Visual Studio for the Windows upload client,

#### Version Control

Git was used for version control as discussed in the Methodology subsection of the previous section. GitHub was chosen for the remote hosting of the repository. A private repository was used during ongoing development but this will be made public after submission. The repository can be found at https://www.github.com/matthew14/Final-Year-Project

### Language / Framework

It was decided to use the Python programming language to implement the web service along with Flask ‘micro-framework’ to handle common web application tasks such as handling http requests as well as forming and sending http responses to these request. Additional features of Flask used in the web service included its session handling.

### Database

PostgreSQL was the database used for the web service. A development database was configured on a headless Debian Virtual Machine.

### Development Virtual Machine

A virtual machine was created on Oracle’s VirtualBox virtualisation software and was configured to mimic the production environment as well as possible. This included installing the apache webserver and PostgreSQL database engine as well as all other software found on the production server. This was done in order to minimize the changes that needed to be made in a production environment due to differences between the Windows development environment and the Debian GNU/Linux production environment.

This also allowed the development database to be installed, offering a separation between it and the production database. This allowed mistakes that were made during development to have no effect on the running production snapshot at that particular point in time.

Using a virtual machine also allowed transferring the development environment from one computer to another with ease. This proved extremely useful when a computer developed a fault and had to be replaced midway through this project.

### Operating System

Windows was the primary development environment for all three components primarily due to an existing familiarity and access to a Windows machine. Another reason for the use of Windows to develop on was the requirement of the Upload Client to run on Windows; using the platform it runs on simplifies development.

### Deployment

#### Operating System / Machine

The web service is deployed on a Virtual Private Server (VPS) hosted by DigitalOcean. The VPS runs version 7 (aka Wheezy) of the Debian distribution of GNU/Linux. At present this VPS has access to 512mb of ram and 20 gigabytes of solid state storage, but may be expanded as more users of the system begin to tax these resources.

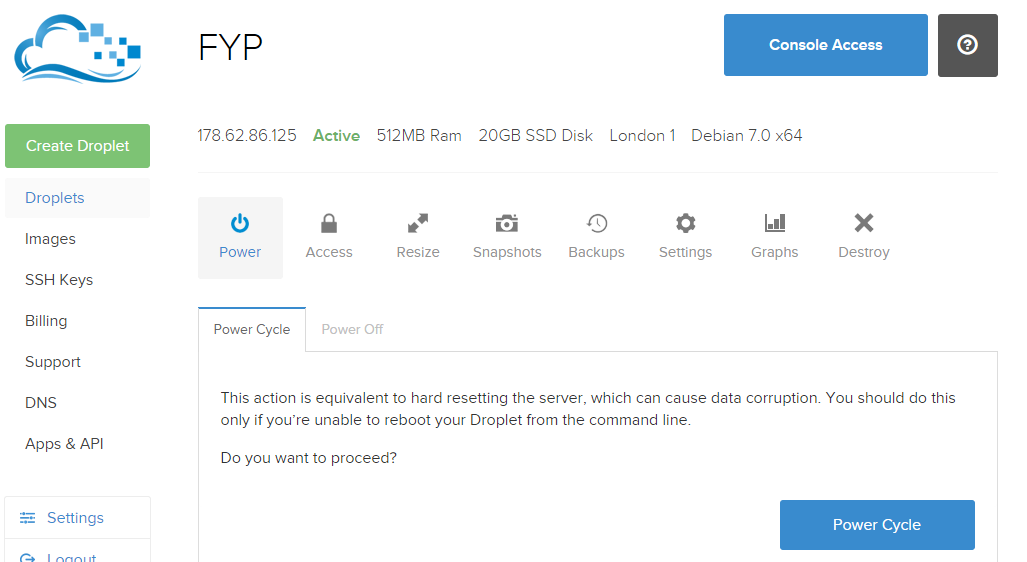


Figure 3 Overview of the DigitalOcean VPS

#### Domain Name

A subdomain of an existing personal domain, matthewoneill.com, was created for the project: fyp.matthewoneill.com. The DNS A record for this subdomain was set to map to the IP address of the DigitalOcean VPS mentioned above. While easier to remember than the IP address of the server, this has another advantage: should the machine that the web service is deployed to need to be changed, only the DNS record requires changing, not the service address that each installation of the client application is configured to connect to.

#### Web Server

Apache was the web server of choice for the project. This choice was made based on the existence of quality documentation regarding using the web server with flask, as well as an existing familiarity with Apache.

In order to make the apache web server work with Flask the mod\_wsgi extension to apache had to be installed WSGI, or the Web Server Gateway Interface is an interface Python uses to enable web servers and python web enabled applications to communicate. While it is somewhat similar in nature to the Common Gateway Interface (CGI), it does not come as standard with the apache web server.

#### Using git to Deploy

As mentioned, git was used for version control, it did however have another use in this project: deployment. The steps to deploy the application from the development machine to the production machine were as follows:

1. Commit changes to local repository on the development machine
2. Push the latest commit to the remote repository on GitHub
3. SSH into the production VPS
4. Pull the latest changes from the remote repository
5. Restart the apache webserver to ensure it is serving the latest version of the python

A simple bash script was written to simplify the deployment on the server:

cd /home/fypuser/Final-Year-Project/ServerCode

git pull

sudo service apache2 restart

cd ~

## Web Service

### Handling Uploads

Music files are uploaded to the web service over using HTTP POST, which is handled by the Flask framework. A file does not necessarily have to be uploaded by the designated file upload application, however, the user uploading the file does need to be logged in at the time of upload, and locations where a user can log in are limited.

### Organizing Files

. Once a file is uploaded to the service by a logged in user, the file’s ID3 tags are read to discern the Artist, Track Name, Album name and Duration of the track. Once this information has been gathered, the system checks to see if a file matching this latest upload has previously been uploaded, and if not it is stored in a folder using the following naming convention:

Uploads/{ARTISTNAME}/{ALBUMNAME}/{TrackName}

#### Data Deduplication

Data Deduplication is the process of “finding and removing duplication within data without compromising its fidelity or integrity” [13]. It was decided to employ this progress when organising tracks that are uploaded to the web service.

### Developing a Model

When tracks have been received by the service and have been organised, the analysis begins on the newly uploaded track.

### Selection of Tracks

## APIs Utilised

### Echonest

### ChartLyrics

## Third Party Modules and Libraries Used

## Overview of Code Written

The following table is a non-exhaustive list of source code files written in the development of this project. As there are three separate components, there are a large number of files to outline (approx. 50). Files which have been auto-generated by the development environment are excluded, apart from those which have seen a large amount of modification since generation (android XML layouts, for example, are initially auto-generated, but all xml was hand written, so as such are included here).

|  |  |  |
| --- | --- | --- |
| Filename | Description | No. Lines |
| *ServerCode/* | ***Folder Containing all code for the web service*** |  |
| routes.py | This file contains all route mappings for the service’s end-points. Requests are received here and dealt with accordingly. | 235 |
| app.py | The main entry point of the web service. This is the file the web server runs to start run the application. | 14 |
| config.py | In order to avoid “hard-coding” and values, all configurable elements are specified here, or loaded by this file from external flat text files. | 61 |
| pg\_db.py | Provides a Database Access Object (DAO) which includes methods to store and retrieve track, user and other information from the server’s PostgreSQL database. | 179 |
| analysis/mood\_assesment.py | This file defines an EchoNestTrack class, which uses that API to retrieve information about itself. Methods in this class exist to build an “excitedness” and “positivity” score using this information. | 127 |
| analysis/lyric\_analysis.py | This file Defines a LyricAnalyser class which attempts to retrieve lyrics of a track from the ChartLyrics API. As calls to this API often fail, it falls back to attempting to scrape the lyrics from a variety of lyric sites. Once lyrics have been obtained, an “excitedness” and “positivity” score is assigned to them. |  |
| analysis/essential.py | Uses the Essentia library to load the track and extract relevant information |  |
| user\_details.py | Provides a class to encapsulate details about a user of this service. | 7 |
| lastfm.py | Provides functions for interaction with the last.fm API to retrieve album artwork. | 15 |
| reanalyse.py | A script designed to be run as a sub-process by the server in the background as execution can take a long time which reanalyses all tracks that a user has stored in the system. |  |
| http\_codes.py | File containing all HTTP error codes for use by the web service |  |
|  |  |  |
| *ClientApp/* | ***Folder containing all code for the mobile application*** |  |
| MainActivity.cs | Code implementing the behaviour of the application’s main screen. | 175 |
| Main.axml | XML layout of the main screen | 24 |
| PlayerActivity.cs | Code implementing the behaviour of the application’s music playing screen | 217 |
| Player.axml | XML layout of the player screen | 80 |
| LoginActivity.cs | Code implementing the login screen of the application | 132 |
| Login.axml | XML layout of the login screen | 27 |
| RegisterActivity.cs | Code implementing the registration screen of the application |  |
| Register.xaml | XML layout of the registration screen |  |
| SettingsActivity.cs | Code implementing the settings page using android’s shared preferences |  |
| Prefs.xml | The layout of the Settings screen |  |
| Menu.xml | Defines the menu items which are located in |  |
| Track.cs | A class which encapsulates the track information which is received from the web service | 24 |
| TrackManager.cs | Class which exposes methods to the player to get the next track from the service | 30 |
| LoginManager.cs | Class which exposes user authentication and registration methods to the main application | 50 |
| MoodRestClient.cs | Internal class which wraps the web service api methods. This is used by the publically accessible Login and Track managers. | 100 |
|  |  |  |
| *UploadApp/* | ***Folder containing all code for the windows WPF upload application*** |  |
| MainWindow.xaml.cs | The code for the main View of the upload application |  |
| MainWindow.xaml | The layout xaml for the main view |  |
| MainViewModel.cs | The ViewModel associated with the main view | 69 |
| SettingsWindow.xaml.cs | The code for the Settings View |  |
| SettingsWindow.xaml | Layout XAML for the Settings View |  |
| RelayCommand.cs | The relay command [14] which is bound to buttons in the Views | 30 |
| Database.cs | A DAO which provides methods to record which files have been uploaded by the system. | 109 |
| UploadClient.cs | This class wraps the REST API methods for uploading files and provides helper methods to do so | 50 |
| FolderWatcher.cs | The class which spawns a thread to continuously check all the files in the configured directories for new files which have not been uploaded to the web service. | 136 |

# System Validation

## Testing

Testing the quality of the mood analysis of the application comprised a large part of testing process for this project.

### Usability Testing

### Comparison with Existing Systems

Many existing systems attempt to recommend similar music to a user based on music they listen to using the system. Such systems include Spotify and iTunes Genius. It was also decided to compare the analysis with existing music mood rating systems such as Mood Agent.

A starting point was chosen as a song that was subjectively sad to the ear of the tester as well as one which scored a low rating in the positivity and energy analysis by the system. This song was then played in Spotify and the ‘start radio’ feature was used to obtain a list of tracks that Spotify considers similar to this starting track. These recommended tracks were then put through the system to be analysed and it was expected that the ratings for these tracks were to be similar. A subset of tracks analysed using this process is shown in the table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Artist** | **Track** | **Positivity Rating** | **Excitement Rating** |
| Low | Lullaby |  |  |
| Low | Down |  |  |
| Red House Painters | Dragonflies |  |  |
|  |  |  |  |

It was noted that these seemingly similar tracks provided varying results when passed through the mood analysis; this provided a basis on which to tweak the weightings assigned to the various parameters used in the formation of the rankings of tracks.

### User Testing

### Automated Testing

Roy Osherove defines Unit Testing in his Book ‘The Art of Unit Testing’ as follows:

“*A unit test is a piece of a code (usually a method) that invokes another piece of code and checks the correctness of some assumptions after-ward. If the assumptions turn out to be wrong, the unit test has failed. A unit is a method or function.*” [15]

Unit testing was carried out during development of the system using the NUnit framework for the upload application, NUnitLite, a lightweight testing framework for testing mobile applications developed using Xamarin for the mobile application, as well as pytest for the web service.

#### Subset of Unit Tests

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected Output** | **Actual Output** |
|  |  |  |

## Summary

In summary, the validation of this system

# Project Plan

# Future Work

## Web Frontend

As web technologies have progressed, more software services are being made available as a web application rather than, or in addition to the native applications offered for platforms such as iOS, Windows Phone and Android. Popular music streaming services such as Spotify, Google Play Music and Rdio all offer a web application in addition to their native applications. This can prove useful to the user when they are away from their primary machines and wish to listen to music using the service.

## Securing the Service

At the moment the web service providing the analysis and streaming of the users’ music files is not encrypted using SSL or TLS standards. This could prove problematic as user credentials and copyrighted content associated with the user could easily be intercepted by an attacker.

## Improving the Audio Analysis

The mood analysis performed on the tracks the user uploads to the service can be improved upon greatly with a combination of user feedback and investigation into other methods of extracting relevant features from music. Other musical features which are not currently being extracted may also prove useful in improving the system.

## Python 3 Support

Python 3 was released in December 2008 and brought with it many improvements and new features, such as Unicode strings by default and a difference in division results. A result of this major overhaul of the language is the backwards incompatibility with Python 2 code that was necessary to introduce.

This application was written using Python version 2 as library support at the time of development, primarily for the essential library used for music feature detection.

While it is not presently possible to port the application to python 3 due to the outlined concerns, steps such as using python 2’s future module to ease the transition may be employed in the near future.

## Better Support for Classical Music

At present the system does not take into account the composer of the piece of music it is analysing, but instead uses the artist data. This could prove difficult as multiple recording artists have recorded pieces by popular classical composers, and the system would see these not as the same piece of music but instead two tracks by different artists.

## Tablet and Landscape Layouts

At present, the Android application does not scale well to make use of the additional screen ‘real estate’ afforded to the developer by a tablet. Users of tablets expect tailor made layouts for tablet screens and a blown up version of the phone version of the application is often seen as unacceptable.

Moreover, at present the application does not adjust well to the client device being rotated into landscape mode; the mood selection grid in particular does not scale to the edges of the screen in this orientation. To provide a more polished user experience, the application at present will not adjust orientation when the device is rotated. This, however, may frustrate the user who is not accustomed to this behaviour.

## Expansion to Other Platforms

At present the application is only available on the Android platform. While Android leads market share of mobile operating systems at present, with 76.6% percent of smartphones running Google’s platform [16], due to the extremely large number of smartphone users in the world, Apple iOS’s 19.7% of users and Microsoft Windows Phone’s 2.8% of users represent millions of users who are unable to access the application due to incompatibility with their device.

The use of Xamarin and Mono for the development of the client application means that porting the application to the largest two other platforms will be relatively trivial; only user interface components need be redeveloped per platform, code to interact with the web service can largely be shared across platforms. The requirement of an Apple Mac to develop Xamarin for iOS applications is the only foreseeable obstacle.

# Conclusion

In conclusion

# Bibliography

|  |  |
| --- | --- |
| [1] | Stereomood srl., “About - StereoMood,” [Online]. Available: http://www.stereomood.com/company/about. |
| [2] | Xamarin inc., [Online]. Available: http://xamarin.com/platform. |
| [3] | MongoDB inc., “Introduction to MongoDB,” 2013. [Online]. Available: http://www.mongodb.org/about/introduction/. [Accessed 2014]. |
| [4] | T. P. G. D. Group, “Chapter 39. Procedural Languages,” [Online]. Available: http://www.postgresql.org/docs/9.3/static/xplang.html. |
| [5] | T. Christensen, The Cambridge History of Western Music Theory, T. Christensen, Ed., Cambridge University Press, 2002. |
| [6] | ISMIR, “The international society for music information retrieval,” 2014. [Online]. Available: http://www.ismir.net/. [Accessed 2014]. |
| [7] | Fiverr International Ltd., “Fiverr,” 2015. [Online]. Available: https://www.fiverr.com/. [Accessed 2015]. |
| [8] | N. D. S.-F. C. Cheng-Yu Wei, “Color-Mood Analysis of Films Based on Syntactic and Psychological Model,” 2004. |
| [9] | R. Murtagh, “Mobile Now Exceeds PC: The Biggest Shift Since the Internet Began,” [Online]. Available: http://searchenginewatch.com/sew/opinion/2353616/mobile-now-exceeds-pc-the-biggest-shift-since-the-internet-began. |
| [10] | Microsoft, “Building Universal Windows Apps,” 2015. [Online]. Available: https://dev.windows.com/en-us/develop/building-universal-windows-apps. [Accessed 2015]. |
| [11] | G. v. Rossum, B. Warsaw and N. Coghlan, “PEP 8 - Style Guide for Python Code | Python.org,” 05 July 2001. [Online]. Available: https://www.python.org/dev/peps/pep-0008/. [Accessed 2014]. |
| [12] | G. v. R. David Goodger, “PEP 257 - Docstring Conventions | Python.org,” 2001. [Online]. Available: https://www.python.org/dev/peps/pep-0257/. [Accessed 2014]. |
| [13] | Microsoft, “Data Deduplication Overview,” 2012. [Online]. Available: https://technet.microsoft.com/en-ie/library/hh831602.aspx. [Accessed 2015]. |
| [14] | Microsoft, “Commands, RelayCommands and EventToCommand,” 2013. [Online]. Available: https://msdn.microsoft.com/en-us/magazine/dn237302.aspx. [Accessed 2014]. |
| [15] | R. Osherove, The Art of Unit Testing, 2nd Edition, New York: Manning Publications Co.. |
| [16] | IDC Corporate USA, “IDC: Smartphone OS Market Share 2014, 2013, 2012, and 2011,” IDC, 2015. [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Accessed 2015]. |