

**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.

This project aims to provide a system for the music listener to categorize and play his or her favourite music based on the emotions that music induces in the listener.

The result of the project was the development of a mobile application and server-side back end which allows the user to select a mood they are experiencing and be played music from their personal collection which matches that mood

**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 7](#_Toc414536964)

[1.1 Project Aim 7](#_Toc414536965)

[1.2 Challenges Faced 7](#_Toc414536966)

[1.2.1 7](#_Toc414536967)

[1.3 Structure of this Document 7](#_Toc414536968)

[1.3.1 Section 2 – Research 7](#_Toc414536969)

[1.3.2 Section 3 – Design 7](#_Toc414536970)

[1.3.3 Section 4 – Architecture and Development 7](#_Toc414536971)

[1.3.4 Section 5 – System Validation 7](#_Toc414536972)

[1.3.5 Section 6 – Project Plan 7](#_Toc414536973)

[1.3.6 Section 7 - Conclusion 7](#_Toc414536974)

[2 Research 7](#_Toc414536975)

[2.1 Alternative Existing Solutions 8](#_Toc414536976)

[2.1.1 SensMe 8](#_Toc414536977)

[2.1.2 Moodagent 9](#_Toc414536978)

[2.1.3 StereoMood 9](#_Toc414536979)

[2.1.4 Feature Matrix 9](#_Toc414536980)

[2.2 Technologies Researched 9](#_Toc414536981)

[2.2.1 Mobile Development Platform(s) 9](#_Toc414536982)

[2.2.2 Database 10](#_Toc414536983)

[3 Design 10](#_Toc414536984)

[4 Architecture and Development 10](#_Toc414536985)

[4.1 Development Environment 10](#_Toc414536986)

[4.1.2 Language / Framework 10](#_Toc414536987)

[4.1.3 Database 11](#_Toc414536988)

[4.1.4 Development Virtual Machine 11](#_Toc414536989)

[4.1.5 Operating System 11](#_Toc414536990)

[4.1.6 Deployment 11](#_Toc414536991)

[5 System Validation 12](#_Toc414536992)

[5.1 Testing 12](#_Toc414536993)

[5.1.1 Usability Testing 12](#_Toc414536994)

[5.1.2 Comparison with Existing Systems 12](#_Toc414536995)

[5.1.3 User Testing 12](#_Toc414536996)

[6 Project Plan 13](#_Toc414536997)

[7 Conclusion / Future Work 13](#_Toc414536998)

[7.1 Future Work 13](#_Toc414536999)

[7.1.1 Web Frontend 13](#_Toc414537000)

[7.1.2 Securing the Service 13](#_Toc414537001)

[7.1.3 Improving the Audio Analysis 13](#_Toc414537002)

[7.1.4 Python 3 Support 13](#_Toc414537003)

[7.1.5 Better Support for Classical Music 13](#_Toc414537004)

# 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. The system will consist of three components:

## Challenges Faced

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

### 

## Structure of this Document

This document is structured into 7 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 areas in which it differed.

### Section 7 - Conclusion

This final section reflects on the learning that has taken place over the course of the project. Suggestions are also presented as to possible additions and improvements that could be made to the final system should development continue into the future.

# Research

## 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

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.

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

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.

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” [4]. The music recommended by this service is aggregated from music blog postings and streamed using the popular SoundCloud music hosting platform.

### Feature Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **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 [5] 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.

### Database

# Design

# Architecture and Development

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

* Web Service
* Client Application
* Desktop Upload Utility

## Development Environment

#### IDE and Text Editors

#### 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 on going development but this will be made public after submission.

### 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.

### 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

### 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 30gb of solid state storage, but may be expanded as more users of the system begin to tax these resources.

#### 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 ~

# 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.

### User Testing

# Project Plan

# Conclusion / Future Work

## 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.

### 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.

**There are no sources in the current document.**