NORTHERN ARIZONA UNIVERSITY
College of Engineering, Forestry & Natural Sciences

SCHOOL OF INFORMATICS, COMPUTING, AND CYBER SYSTEMS
CS 386 – Software Engineering
Prof. Marco Gerosa

Team Project – D.5 Release 1

1. Introduction

Our system will help users listen to music without having to discover music. It will lets users broaden their horizon with good music, without having to search through random phishing websites. The problem of not having a music player that satisfies all the needs of the modern user affects populations of various taste and age; the impact of which is not having a reliable music streamer and a place to listen to the music of their choice. Users will have the chance to broaden their horizon with good music without having to search through random phishing websites. For music lovers of all age who are struggling to find a centralized place to put all their music, *Mood* music streaming app that allows the users to play music based on their current mood, ability to create and modify easily and have an easy way to download their songs unlike Spotify or Pandora our product will not rely on premium plans to let the user create playlists or listen to music without having to be interrupted.

https://github.com/turan1393/Mood-Streaming-App
https://trello.com/b/uVRpOEtI/mood-streaming-app

2. Implemented requirements

Requirements we implemented from the start were the promised features of selecting a mood, searching and downloading music, streaming music, and creating a playlist for a specific mood. However, not all functionality went according to plan. During one of the testing phases, a user by the name of James reported having an issue upon opening the application. James reported the app shutting down on him sometimes while other times it would appear to continue to work but prevent the use of other functions. When a mood was selected, it wouldn't always produce its respective type of music. Some moods that would be selected for a more relaxing feeling would list more upbeat music. Karsten worked on this and was able to fix the problem by fine tuning different moods that different bands or artists had. Another problem was that selecting play or pause wouldn't always give that output. Mike worked on this part, among the other button functionalities, and was able to figure the invisible perimeters of the buttons extended further than they should have and resulted in the user pressing play and pause at the same time without knowing it. In addition, streaming music proved to be a problem when the stream itself was down or slow. The problem was fixed by Austin C. once the group realized the application wasn't originally set to search other streams if one was down so it would continue to try to access music from the one that wasn't working properly. The last problem we had, the app shutting down for no reason immediately after opening, was fixed by Turan and Austin T. They found out the app was closing due to searching too many music databases, streams, artists, etc, before the user actually selected a mood. This led to the application overloading itself and simply shutting down. This was fixed by limiting how many music databases, streams, artists, etc, could be searched at once. Once each member fixed he feature they were working on, we made progress with the application and are closer to its final form.

3. Adopted technologies

Include a list of adopted technologies with a brief description and justification for choosing them.

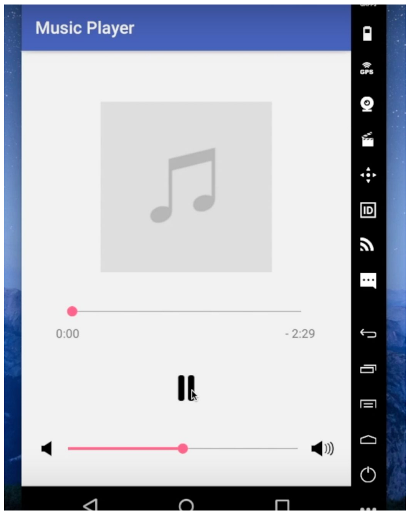
1. Android Studio
    a. It is built on JetBrains IntelliJ IDEA and is exclusive to android development. Even though we want our application to be able to b downloaded on Android and iOS devices, we used Android Studio because it can be used across Windows, Macs, and Linux operating systems. Meaning, we can all contribute to it with our own computers at our own discretion.
2. Xamarin
    a. This IDE is specifically meant for cross platform compatibility among iOS, Android, and Windows and uses the C# programming language. We used Xamarin because it can also be used for desktop solutions as we would like it to be able to be used on mobile and desktop devices.
3. Meteor
    a. It uses Node.js to create open source isomorphic web framework and automatically propagates data changes. We chose Meteor because it would work great with our application being that music it produced based on a mood selection. The other thing we liked about it is that is uses JavaScript, C++, and C.

4. Licensing

We decided to adopt the license **Mozilla Public License 2.0 (MPL-2.0)** though opensource.org. That consensus includes giving permission to all to use, study improve and share the code without prejudice, the license is an open source license. The Open Source Definition provides an objective test of evaluating that such a license is indeed an open source license and delivers the software freedom we all expect. We thought that **Contributor License Agreement (CLA)** would work best for us, because the original contributor retains copyright ownership of their contributions, but grants the project a broad set of rights such that the project can incorporate and distribute the contributions as it needs to.

5. Look & feel

We wanted our user interface to be easy and simple. This way users will understand the basis of changing volume, pausing/playing music, and having access to scrubbing the music to exactly where the user may want the song to be at. Along with those features, we wanted to include the album photo (if available).



6. Learning/training

In order to get proper training for all of us, we found multiple options online which gave proper instructions to programming an app. We went through these tutorials to understand how to create a usable interface, along with designing the application exactly how we wanted it to look.

7. Lessons learned

We learned that we rushed our application. We could of fully implemented our original idea of a music streaming application. We tried to simplify our application to meet the release date. We figured that breaking the function into separate state, makes the whole process smoother. Instead of giving each member a section of the code to do, next we will split the coding among two seperate groups. This process will hopefully speed the programming process and combing the program better. We want to be able to organize as a group better for the next release.

8. Demo

We decided to send a video file zipped to this document. Thought it be easier on both parties.

9. Group participation

Each member of the group was active in this document. Turan Naimey focused hard on section 1, and task break ups. Turan also helped around multiple sections during the process of this document. (20%). Karsten focused on sections 2 and 3, and assisted others in other sections as well. (20%). Austin C provided help on sections 1 and 2 along with general help on sections 3, 4 and 5. (20%). Austin T worked on sections 6 and 7, while also providing help within 1, 2, 3, and 4. (20%). Mike Ewers worked on sections 8 and 9, while helping with sections 1, 2 and 6 (20%). (100%) of us as group collectively worked on developing the code and the interface. Austin Collins tested the application and took the video for the demo.