Symphonia

A synchronous
music player for
everyone.

Problem Statement

Music listeners encounter the problem that the volume of any individual device they own is often too low. Although the combined volume that can be obtained on all the devices available is adequate for their needs, this is underutilised due to a lack of the ability to play the music synchronously across all the devices.

Functionalities

- Common music functionalities
- Send song
- Receive Song
- Play/Pause Synchronously

Development Phase

All OOP coding guidelines have been followed:

- Classes have been used for encapsulation and modularity.
- Private members have been defined for data abstraction.
- Class inheritance has been used.

Development Phase

Naming Conventions

- Names representing packages should be in all lower case.
- Names representing types must be nouns and written in mixed case starting with upper case.
- Variable names must be in mixed case starting with lower case.
- Names representing constants (final variables) must be all uppercase using underscore to separate words.
- Names representing methods must be verbs and written in mixed case starting with lower case.
- Abbreviations and acronyms should not be uppercase when used as name.
- In XML, all code is in lowercase, separated by underscores(_)

Development Phase

The package statement must be the first statement of the file. All files should belong to a specific package.

- Imported classes should always be listed explicitly. Class and Interface declarations should be organized in the following manner:
- Class/Interface documentation.
- Class or interface statement.
- Class (static) variables in the order public, protected, package (no access modifier), private.
- Instance variables in the order public, protected, package (no access modifier), private.
- Constructors.
- Methods (no specific order).

Testing Phase

- Unit testing
- Integration testing
- Use Case testing

Results:
https://docs.google.com/spreadsheets/d/1g94N8Xj91KLzUveA0_FE_TqAgv_3EGh3BxJSVaHoiw/edit#gid=1001621097

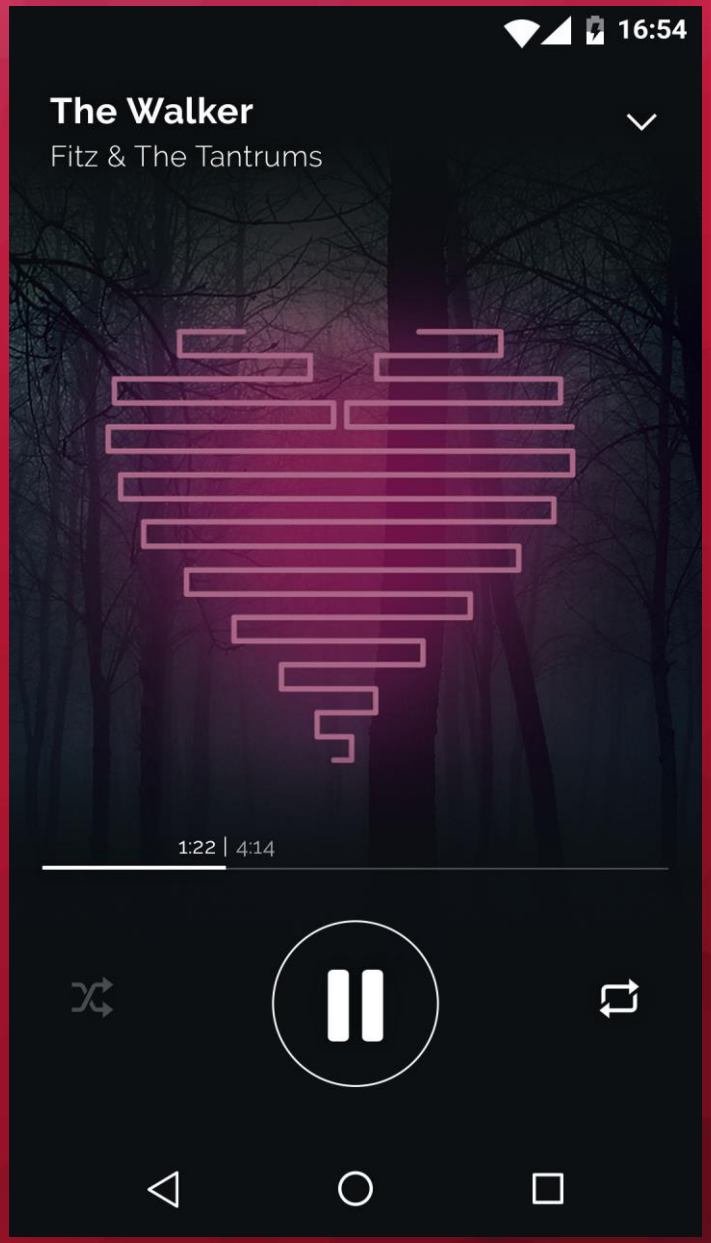
Testing Phase

Usability Testing is a way to see how easy to use something is by testing it with real users.
We made our friends both with and without CS background use our app, but did not instruct them on what to do. We let them learn the system by themselves.
Observation: Users were able to use the app without any struggle since it was an intuitive design. We had to teach them how to send and receive song, because they were the new non intuitive functions

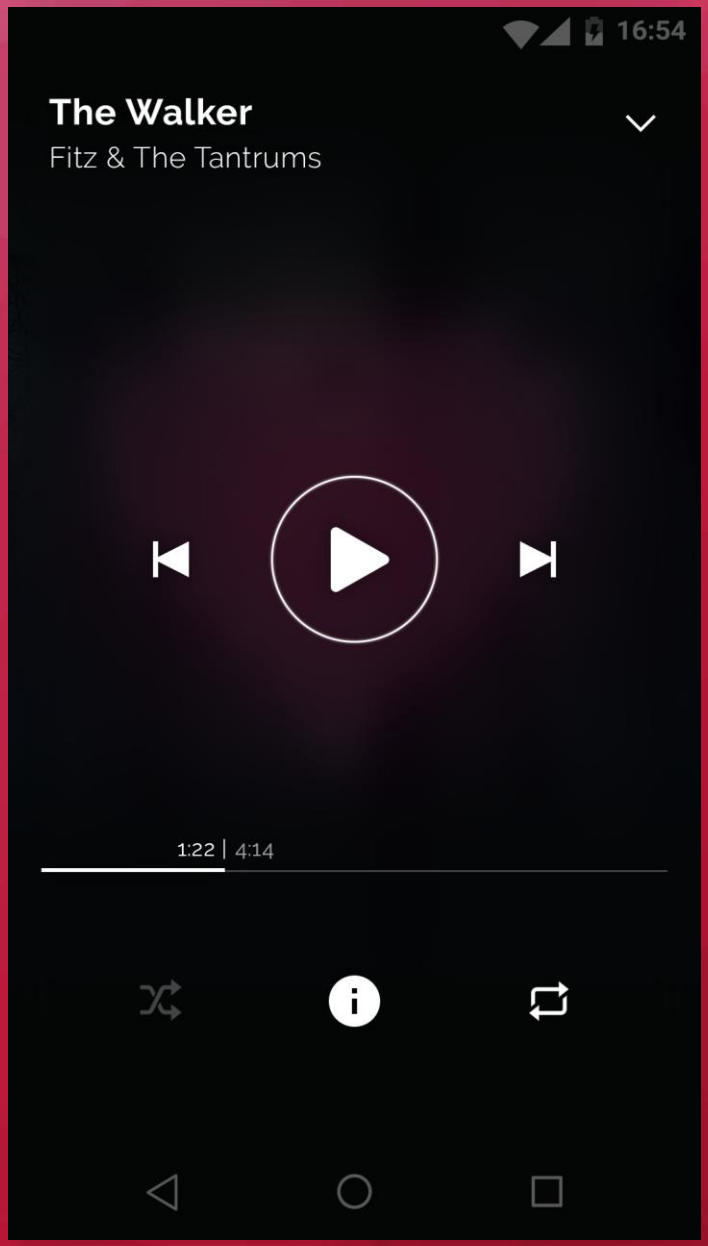
Regression testing was done by installing it on different Systems.
A bug was found when the system had Android 6 or above. It needed extra file permissions for accessing the songs.

Screenshots

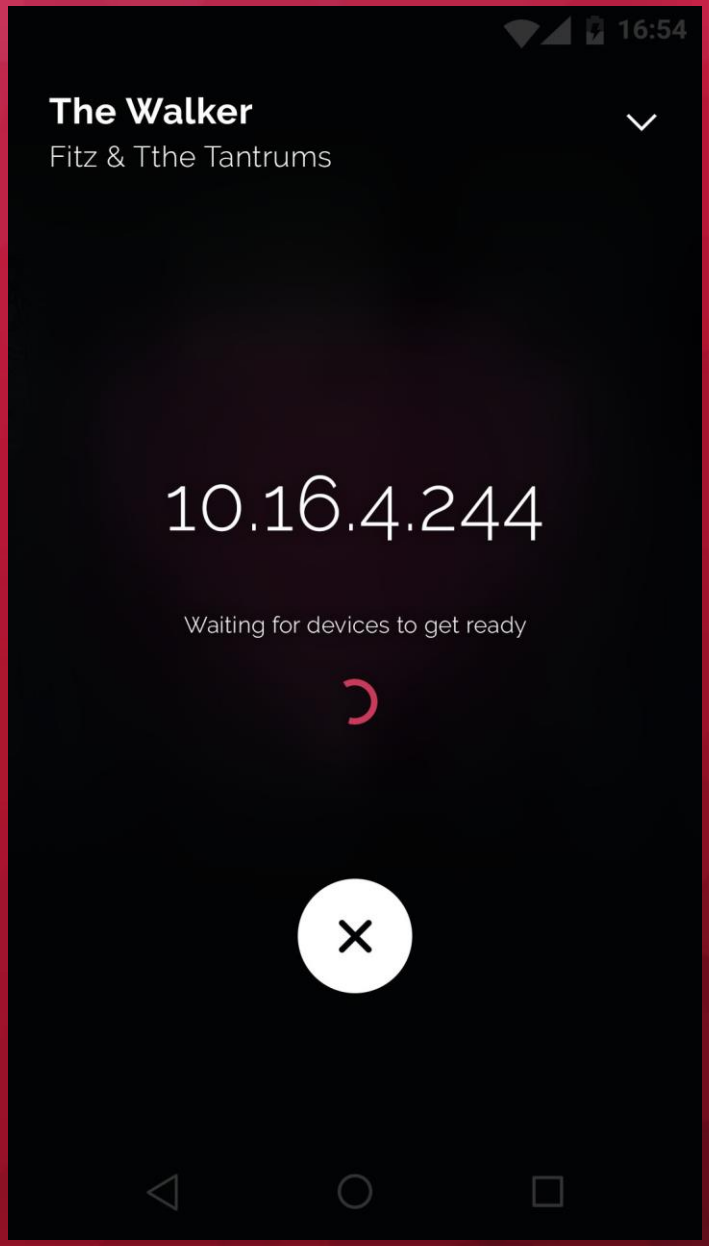
Playlist



Now Playing

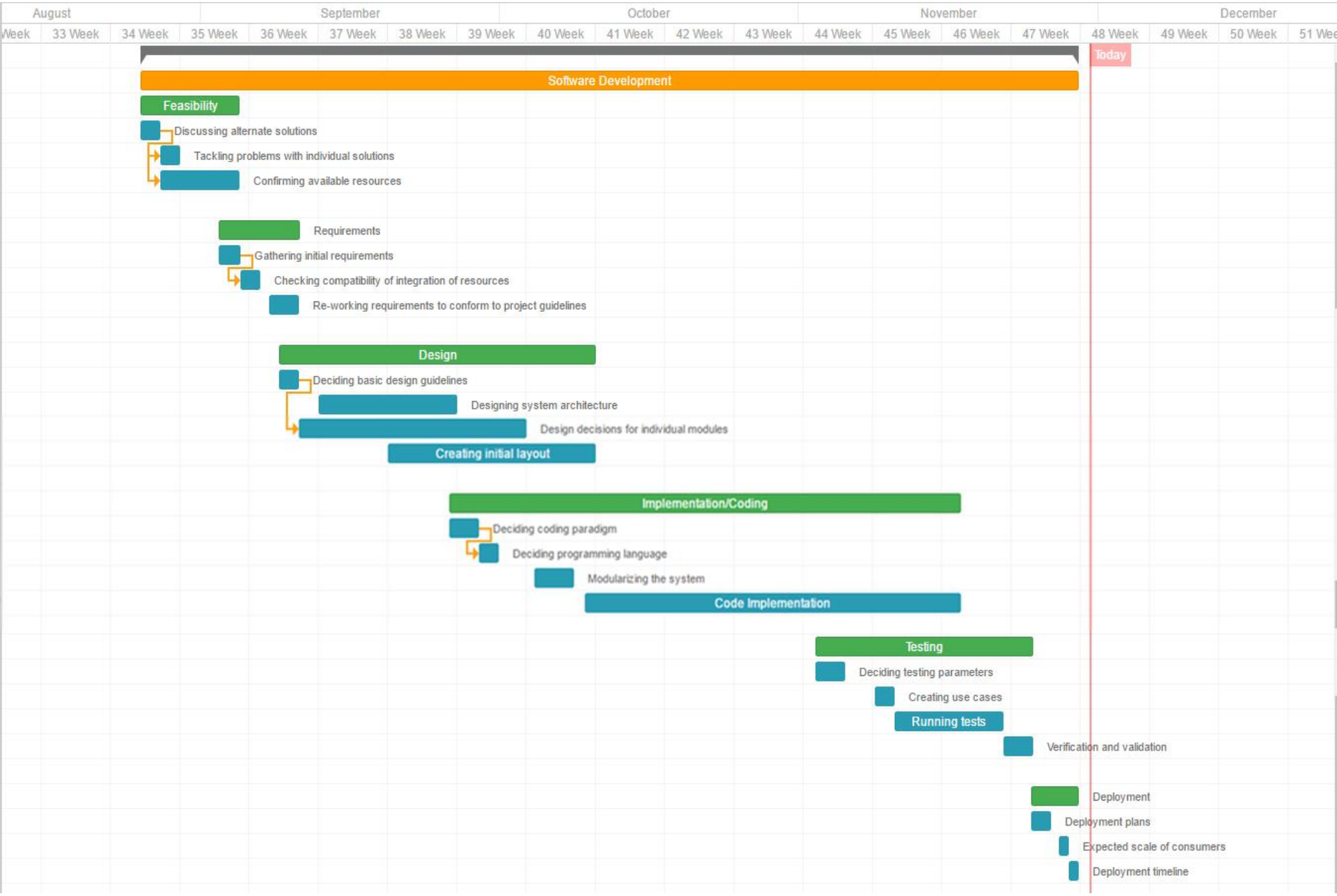


Paused



Sending

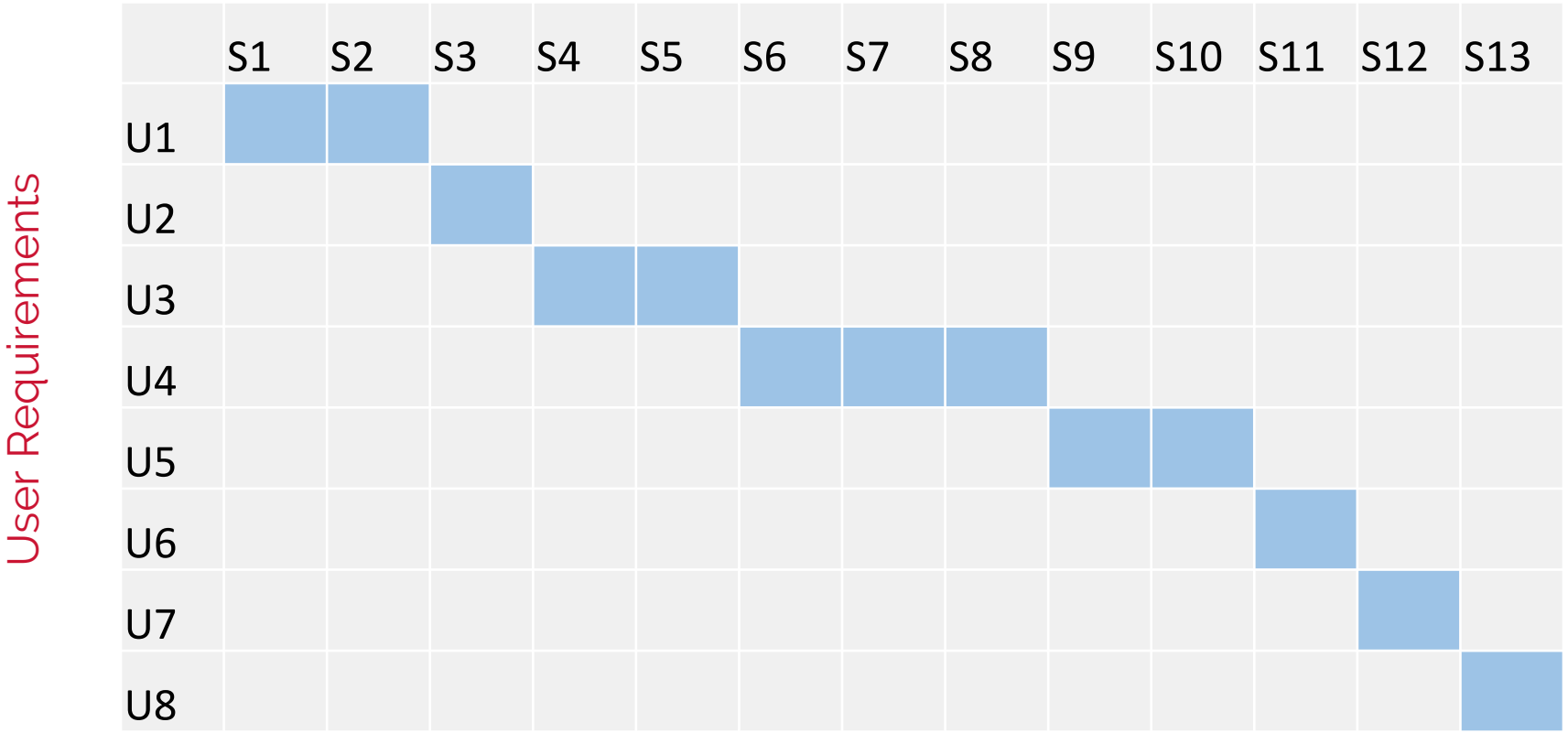
Gantt Chart


August
September
October
November
December
Week
33 Week
34 Week
35 Week
36 Week
37 Week
38 Week
39 Week
40 Week
41 Week
42 Week
43 Week
44 Week
45 Week
46 Week
47 Week
48 Week
49 Week
50 Week
51 Wee
Today
Software Development
Feasibility
Discussing alternate solutions
Tackling problems with individual solutions
Confirming available resources
Requirements
Gathering initial requirements
Checking compatibility of integration of resources
Re-working requirements to conform to project guidelines
Design
Deciding basic design guidelines
Designing system architecture
Design decisions for individual modules
Creating initial layout
Implementation/Coding
Deciding coding paradigm
Deciding programming language
Modularizing the system
Code Implementation
Testing
Deciding testing parameters
Creating use cases
Running tests
Verification and validation
Deployment
Deployment plans
Expected scale of consumers
Deployment timeline

Traceability Matrix

System Requirements

User Requirements



ID	User Requirement	Forward Traceability	Test ID
U1	User should play a song	S1, S2	Sym_1
U2	User should pause the song	S3	Sym_2
U3	User should choose a song	S4, S5	Sym_3
U4	User should receive a song from host	S6, S7, S8	Sym_4, Sym_5
U5	Host user should send a song	S9, S10	Sym_6
U6	Host user should synchronize the song for receivers	S11	Sym_7, Sym_8
U7	Host user should synchronize the song for receivers	S12	Sym_9
U8	User should navigate to next song while sending	S13	Sym_10

ID	System Requirements	Backward Traceability
S1	System should display selected song name	U1
S2	System should start playing the song	U1
S3	System should pause current song	U2
S4	System should display the list of songs on device	U3
S5	System should redirect to playing phase and change current song	U3
S6	System should show page asking for sender IP	U4
S7	System should search for given IP and check if it is available	U4
S8	System should start receiving the file from host	U4
S9	System should have a response for a long press of the play button	U5
S10	System should display the device IP	U5
S11	System should adjust seek time to synchronize with the host device	U6

Future Plans

1. The application will be ported to iOS, and Mac, and allow cross platform playback for all systems.
2. The application will also be extended to act as a live song streaming service.
3. Future usage could involve streaming audio podcasts, for purposes of education and entertainment.
4. To combine the application with a large scale music database, to pull lyrics, as well as additional songs from.
5. This app will work in conjunction with other music players, by listening to the song playing on another device, and seek to the point at which that song is.