**CS322 TEAM PROJECT DOCUMENT**

|  |  |
| --- | --- |
| TEAM NAME: | EZ tunes |
| TEAM MEMBERS: | Kevin O'Flynn  Seán Freeman  John Handy |

Our original proposal was to develop a midi sequencer app for android, however this idea pivoted and we instead decided to focus on building an equaliser with which we could manipulate the various attributes of .wav files. After the equaliser was complete, we felt that we could expand the scope of the app, and decided to include a guitar tuner. The tuner does not ‘listen’ to an instrument when tuning it, but rather works like a pitch pipe, allowing the user to chose between generated midi tones or actual guitar sounds when tuning.

**Development:**

In the initial phase of development, we researched what resources already exist within android that we could use. We found that the android API android.media.audiofx had several useful classes, (most notably a visualizer and equaliser class). Once we had a good idea of the resources at our disposal, we began to design the UI of the app with pen and paper diagrams. When we had designed these diagrams we used them as a point of reference all throughout development.

Having more experience coding in android, Seán and I did the bulk of the coding for the app. We implemented pair programming, both of us taking turns being the driver and the observer and began to build the app on Seán’s laptop. We set up screens, passed intents between them and designed the layout of the various screens. While Seán and I were doing this, John was gathering resources to be used in the app (wav files etc), recording sounds for the app, and working on specific pieces of coding that would add to the overall polish of the app (such as the splash screen and the code which generated midi tones). After John completed one of these pieces of code, we would test it, and integrate it into the app.

When one of us got stuck on an error, we would all work together to debug it.

Once development was completed, we went through the code as a team and made sure that every function and method in the app was well commented in such a way that any part of the code was easily understandable to the whole team.

**Problems Encountered:**

We encountered some problems with the android.media.audiofx API. It’s range of classes were a bit more limited than we had initially hoped. Some of the more specific classes were repeated in more general classes, and some classes didn’t seem to work at all (such as the bass boost effect). As a result of this, we had a functional, but somewhat limited equalizer. It is because of this that we decided to include the tuner functionality to the app.

Mid way through development, the laptop we were mainly using for coding hard drive crashed. Luckily, we had been regularly backing up new code we added to the app, so we were able to recover a significant portion of the work we had already done. However, this set back still cost us several hours of work.

**EVALUATION: EQUAlISER**

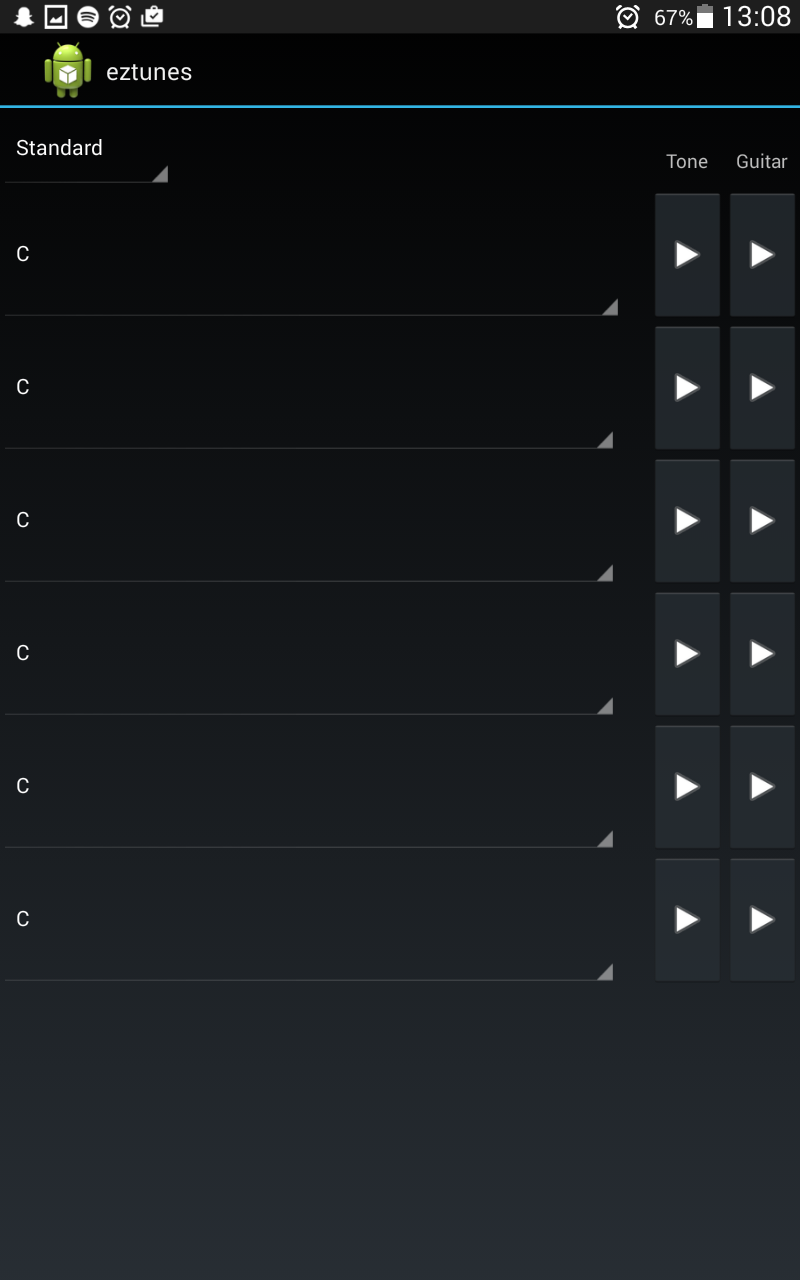
On opening the equaliser, the user is prompted to pick from a selection of songs. We used a list view to represent the choice of songs. The users choice is then passed to the next screen via put extra’s and intents. The main equaliser screen includes a visualizer to provide a visual representation of what song is playing, several slider bars which can be used to boost or reduce songs at a certain frequency range and a spinner which can be used to select preset equaliser settings for different genres of music. We tried to include songs from varying genres in order to better highlight this feature. The equaliser also contains a media player with controls for playing, stopping, fast forwarding and rewinding songs.



*\*The Equailzer screen*

**EVALUATION: TUNER**

The tuner consists of several spinners, one for selecting whether you’d like to tune off guitar sounds or midi tones, one for preset tuning and several for the different notes you might want to play when tuning your instrument, one for each string.



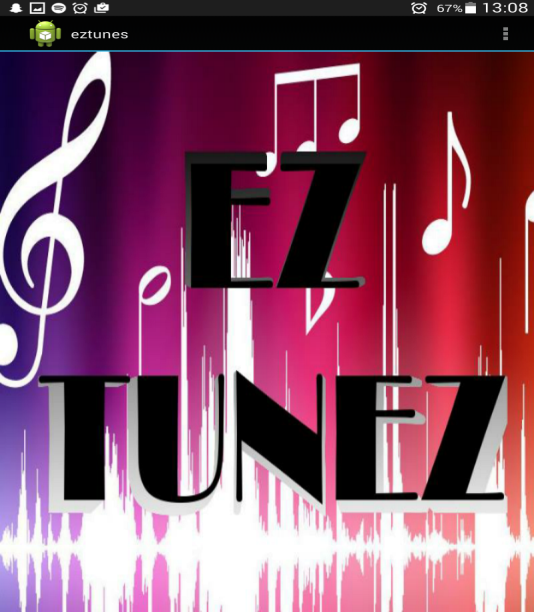
\**The Tuner Screen*

When creating the tuner, we wanted to make it as general purpose as possible, to this end we included not only wav files of live guitar notes recorded by John, but also a wide range of midi tones which could be used for tuning any instrument. As well as this, we didn’t limit the guitar tuner to standard EADGBE tuning, but gave the user the option to set the tuner to drop D tuning or drop C, making the tuner more versatile.

The real guitar sounds are played using a media player when their respective button is pressed. Changing which note you want to play will change the sound file the media player uses. The tone sounds use code to generate a note. Changing the note you want will change a frequency value corresponding to the selected note, and calls a genSound method.

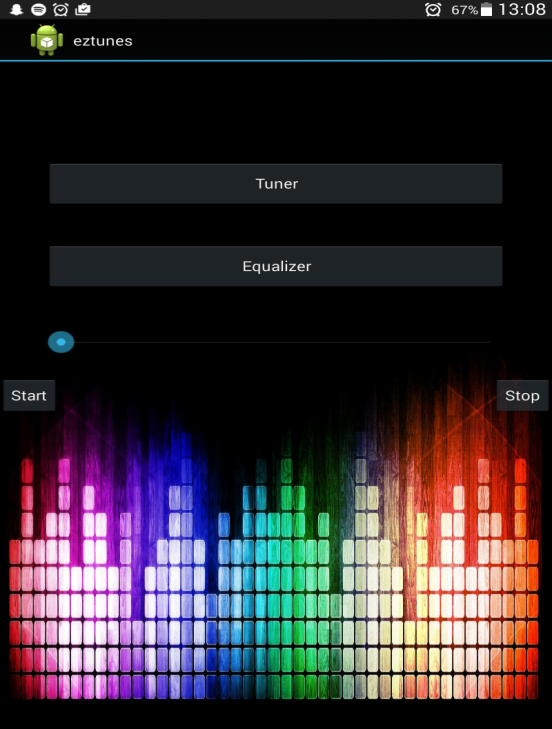
**Other Features:**

**Splash Screen:** We added a splash screen to the start of the app. It shows a logo which we designed for two seconds upon launch. We felt that this splash screen added to the overall polish of the app as well as helping establish a visual theme.



*\*The splash screen*

**Menu Screen:** We felt the menu screen of the app, which allows the user to go to either the tuner or equailzer screens, was too plain. We decided to use some code we found on moodle to add a midi frequency slider to the menu screen. We felt that this would make better use of the empty space at the bottom of the screen.



*\*The Menu Screen with a frequency slider at the bottom*

**Testing:**

Once we had implemented each feature of our app, we asked some of our friends to try and use the test build of the app without telling them how to use it. If they were able to operate the app without instruction from us, we considered the feature to be easy to use. If they had trouble with a certain feature we would change the non-functional requirements of that feature until they were easier to use. Some things we changed as a result of this testing included making play and pause buttons larger on the media player, changing the duration of the notes playing in the tuner to be shorter and including fast forward, stop and rewind buttons in the media player so that the user could control playback of the song.

**Potential Future Development:**

When designing the app, we had several ideas on expanding the functionality for future developments with more work and time, some of which are listed below:

* Adding more instrument sounds in the tuner, e.g ukulele, violin, mandolin etc.
* Including more tuning presets and further octaves
* The ability for users to download their own songs for the equaliser
* Implementing more varied audio effects in the equaliser

**Resources Used:**

Code for Synth Slider taken from:

<https://2015.moodle.maynoothuniversity.ie/mod/folder/view.php?id=34179>

Used <http://developer.android.com/index.html> for references and information about android API’s