Amadeus

Stephen Getty
Mitchell Hymel
Jonathan Lowe
Raaj Parasuraman
Alex Schimpf
Nicolas Patenode
Hiren Pithadia

Table Of Contents

Project Description

Our application serves the purpose of helping the user learn about music. Specifically, it helps users learn to write and compose music. The user can either write their own music from scratch or play a song into their phone. If the user plays a song into their phone, it will be transcribed by our application. In either case, the user can then playback their input in order to hear what they have written or recorded. Edits can be made once the notes are written. These compositions can be saved and accessed at a later time. The user can also play along to a previously composed song and receive feedback about how accurately their playing was to their song. Lastly, the user can also look up the chords to their favorite songs from the internet.

Process Followed

Over the course of the semester we followed the process of XP as described in class with a few modifications discussed below. Here are the principles we followed/modified:

Planning Game

For each iteration we created a set of user stories to be completed during that iteration. At the beginning of each iteration we estimated the amount of time in hours that we believed each user story would take to complete. Upon completion of each iteration we would compare this to the actual time in order to compare our actual vs expected time. We strove over the sequence of iterations to get better at our estimations in order to increase our productivity.

Refactoring for Design

At the beginning of each new iteration we would refactor the code from the previous iteration in order to improve the code base.

Pair Programming

We followed pair programming and at the beginning of each iteration split the team into pairs with one group consisting of three members. We made sure to choose new partners each iteration so everyone would have a chance to work with everyone

Testing

We made sure that each piece of code submitted also came with a set of unit test associated with it. Before checking new code into the repository each member was

required to make sure that all the tests for the project passed as not to commit broken code

Coding Standards

We each followed the standard android and java coding standards to ensure that the code base was consistent and readable

Test Driven Development (TDD)

Due to the nature of application it was very hard to adhere to strict TDD. GUI's are very hard to test programmatically and usually require manual testing plans. We tried whenever possible to write tests first, but often this just was not practical or possible.

Requirements and Specifications

Main features of our application

- Composing sheet music
  - I want to be able to record sound and have the app generate sheet music.
  - I want the ability to play back recorded/created sheet music.
  - I want the ability to manually create sheet music.
  - I want the ability to save sheet music and open previously created sheet music.
  - I want to be able to create and save sound profiles.
  - I want the ability to play back sheet music with a sound profile.

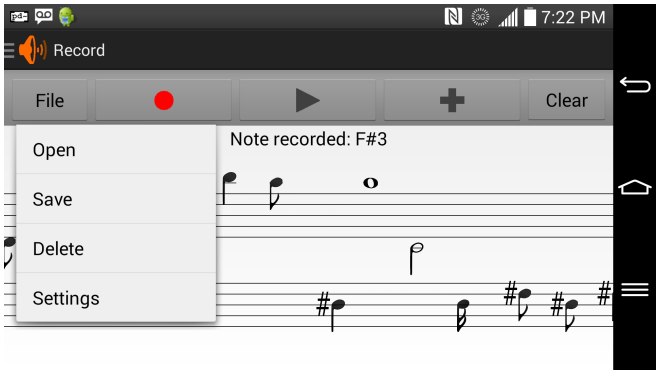

Figure a. The sheet music screen.

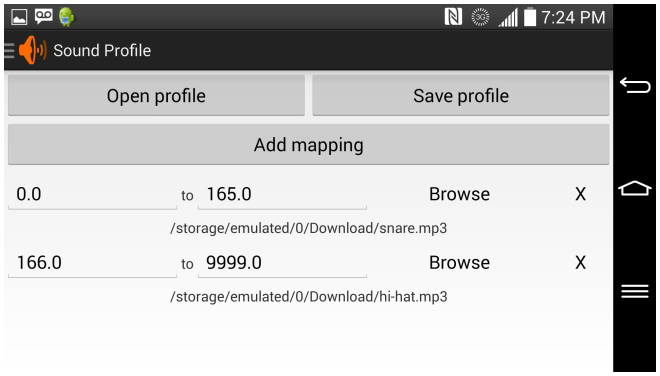

Figure b. The sound profile screen.

- Play along
    - I want to be able to play along to saved sheet music and have it grade me on how correct I was.
    - I want to be able to see my played notes versus what should be played.

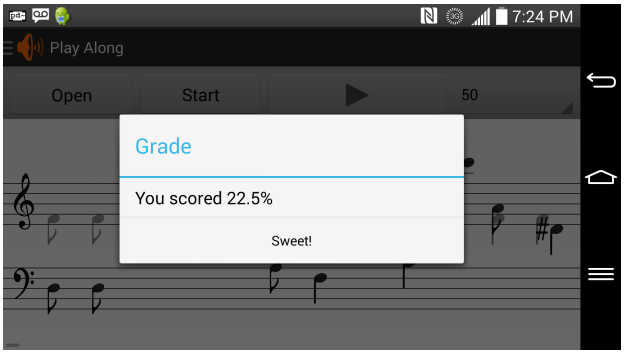

Figure c. The play along screen.

- Guitar tabs and chords
    - I want the ability to search for guitar tabs.
    - I want the ability to view common guitar chords.

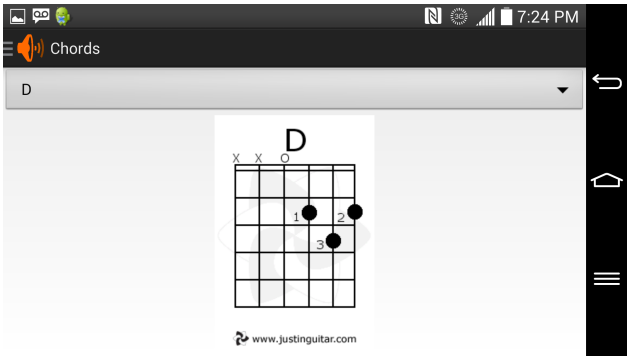

Figure d. The guitar chords screen.

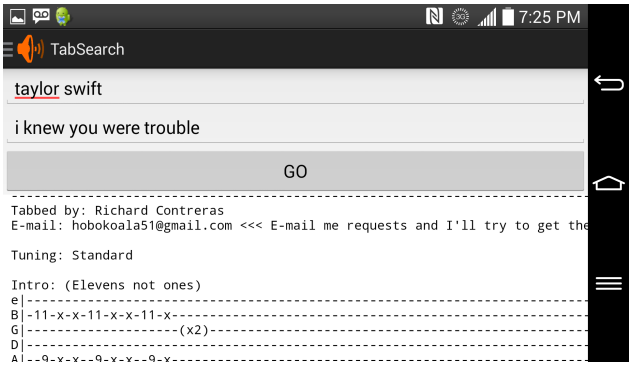

Figure e. The guitar tabs screen.

Architecture

Our application runs within one Android activity called ***`MainActivity`***, which is contained within the package *com.cs429.amadeus.activities*. This activity simply handles switching between fragments.

Within this activity, there exist multiple fragments, which are related to the different tools/screens of the application. They are contained within the package *com.cs429.amadeus.fragments.* These fragments are the following:

`GuitarChordFragment`

- Where the user can view different chords
- Users picks chords from a spinner that are then displayed

`HomeFragment`

- Where the user is initially brought to when the application is run
- Contains buttons for navigating to the main fragments of the application

`PlayAlongFragment`

- Where the user can play along with a saved sheet and get graded on accuracy
- Contains a button bar for opening sheet music, recording, selecting BPM, etc.
- Also contains a `PlayAlongStaffLayout`, where the recorded and opened notes are displayed
- Uses a PureData service to constantly poll for frequencies recorded by the phone

`RecordingFragment`

- Where the user can go to create/save/open/play sheet music
- Contains a button bar for opening/saving, recording, manually adding notes, etc.
- Also contains a *`StaffLayout`* for displaying creating sheet music
- Uses a PureData service to constantly poll for frequencies recorded by the phone

`SoundProfileFragment`

- Where the user can create a sound profile
- Contains open/save buttons and a button for adding individual mappings

`TabSearchFragment`

- Where the user can search for guitar tabs from ultimate-guitar.com
- Includes text boxes for entering tab queries and a text box to display the fetched results

---

Within some fragments, we need custom Android views, which are visual elements that users can interact with. They are contained within the package *com.cs429.amadeus.views.* These views are the following:

`FallingNotesView`

- Is the background of the HomeFragment
- Shows notes continuously falling at random down the screen

`NoteView`

- Represents a graphical note, that is meant to be added to a staff
- Touch events cause it to be removed from a staff
- Is closely associated with the *`Note`* class

`PlayAlongStaffLayout`

- This is an extension of *`StaffLayout`*, that is used to uniquely display opened and recorded sheet music notes
- Makes opened notes partially transparent and recorded notes fully opaque

`StaffLayout`

- Represents a musical staff, to which *`NoteViews`* are added
- Contains staff lines and clef notation

We then use various helper classes to assist with things like recording, processing frequencies, playing midi notes, and opening/saving files, to name a few. They are contained within the package *com.cs429.amadeus.helpers.* These classes are the following:

`Metronome`

- Used to update the application state (user-defined) when the metronome “ticks”
- This relies on the user-selected beats per minute (BPM)

`NoteCalculator`

- Contains static methods for converting between frequencies, midi notes, and musical notes

`OpenSaveHelper`

- Contains static methods for opening/saving sound profiles and sheet music in XML format

PlayAlongAnalyzer

- Contains static methods for “grading” a given PlayAlong instance
- Does this by comparing pairs of recorded/opened notes and deciding the similarities

`PlayAlongStaffMidiPlayer`

- An extension of the *`StaffMidiPlayer`*, that is used to play opened sheet music in *`PlayAlongFragment`*
- Includes a minor tweak to only play opened notes (not recorded ones)

`Recorder`

- Used to process given frequencies and convert them into a series of musical notes, which are then supplied to a staff
- Used by *`RecordingFragment`* and *`PlayAlongFragment`* to do the heavy frequency processing

`SoundProfile`

- Represents a user-defined mapping from frequency ranges to sound file paths on your phone, which is then used by the midi players
- These are manually created/edited in *`SoundProfileFragment`*

`StaffMidiPlayer`

- Handles the playing of musical notes that have been added to a staff
- This relies on the user-selected BPM

- It is used by `RecordingFragment`

We also have a few miscellaneous classes. The classes within the package *com.daidalos.afiledialog* are from an outside source and are related to a custom dialog box for opening files from your phone's storage. There is one other class called ***Note***, which simply represents a musical note. It is contained within the package *com.cs429.amadeus.*

By Android framework standards, we have many XML files, which describe various layouts our application uses. These are all contained within the *res/layout* folder. We also have global strings contained in *res/values/strings.xml*. All images are contained in the *drawable* folders in *res*. The Android framework has thus forced us to break the application into many different components, which end up simplifying and modularizing everything.

A class diagram (Figure f) of our project is located at the end of the documentation.

Installation

For end users, simply download the Amadeus.apk file from our GitHub repository. You must then transfer this to your Android phone. When this is done, you will be asked to install the application.

For developers, you first need to clone our repository at https://github.com/amadeus428/trunk. Once this is done, import the project to your development workspace (e.g. Eclipse).

Future Plans

Alex Schimpf

This was a great project to work on for a number of reasons. Firstly, we learned about a lot of different technologies, including Android, Pure Data, GIT, and GUI testing. I specifically learned a lot about GIT and creating custom Android views and fragments. We also learned a lot about music and sound processing, in general. It was also neat to work on an application that did not exist. We encountered and solved problems that others have not. In addition, we further practiced working as a team, using a formal software development

process. This allowed us to hone our communication, refactoring, testing, and version control skills. I would like to open source our code on GitHub (as it currently is) for others to potentially benefit from or improve.

Jonathan Lowe

I liked this project a lot because it presented unique challenges. We learned about signal processing and sound input as well as Android coding. Many members of our team did not have a music background and so also learned a lot about music. Since the topic of detecting sound input is an ongoing research topic, there was not a lot of already existing libraries or technologies that did what we were doing. This allowed us to be original with our design and encounter problems that few have come across before. However, with what was already existing, we were also able to figure out how to integrate some features into our own application which showed us the power and helpfulness of open source projects. We also learned a lot about working in a software development team, especially working with a GIT repository, which helped us learn the value of a version control system. In the future, I would like to see this project expanded to be a fully developed music composition app.

Stephen Getty

I enjoyed working on this project and feel that I have learned a lot throughout the course of the semester. Coming into the project I was unfamiliar with many of the principles and technologies that we were using. Technologically, I feel that I learned a lot about android development and GIT. I had never used either of these prior to the start of the semester and upon the completion of the project feel that I have gained some competence in them. I also learned how to more effectively work in a group setting. It was difficult at times shuffling schedules in order to find times to meet, but I feel that we really came together as a team throughout the course of the project. I do not plan on contributing to the code base post completion of the project. I would like to open source the code on GitHub for others to play with. It would be really cool to see this app expanded in order to add more of the musical principles that we just didn't have time to put into it.

Raaj Parasuraman

This was a great project idea that posed novel challenges in terms of the processing and the mathematics of music. We worked with a bunch of great technologies such as android and Git , but also worked with an open source framework called Pure Data and interfaced with it. Learning the formal software development process is also very useful for the real world, as well as the practices associated with it (pair programming, continuous integration, planning game etc) . The future plan for the project seems to open source the whole project on github and make it available to the developer community. Hopefully, we can market it to other developers and new features can be added to the application.

Mitchell Hymel

For me, contributing to this project was a helpful learning experience. Though I had prior experience with both Android development and Git, I learned even more about these topics over the course of the semester. One of the biggest takeaways from this project for me is the introductory knowledge of Pure Data that I gained over the course of the semester. I'm very interested in signal processing, and I learned so much about the subject by simply learning about the Pure Data library. I learned a lot about how to work in a collaborative environment. My team members were great. Though I don't have any future plans to continue to contribute to this particular application, I feel like we accomplished something that is, at the very least, the start of something amazing.

Nicolas Patenode

I had a lot of fun working on this project and with this team. I had worked with Android and Java in the past, but working with a team of this size motivated me to learn and apply more good design practices and patterns. I got to learn a lot about designing, implementing, and refactoring GUIs, but my biggest take-away from this project is the experience of working with a team of people on a project that they are all interested in. We had a lot of fun working together and we really came together to accomplish some interesting things. I intend to make the project open-source, but I want to keep working on this or a similar project now that I have learned a bit about music programming.

Hiren Pithadia

I found this project to be very rewarding. I had never worked with Android or Git before, so this was a new experience for me. I really liked working with this team. Each person has left their mark on the project is some way, which was really cool to see all the contributions to make this project successful. Furthermore, I found it enjoyable to work with these people as well, and coming up with new ideas and implementations that made our project better than what it was. At the same time, we were constantly learning new things together, such as integration with Pure Data, or even GUI development. This was the first time I actually worked on a project related to music, and after completing this project I feel that I have a basic understanding of musical programming, and would love to develop a similar sort of project like this in the future. I do not have any future plans to contribute to this project, but I feel we all agree to make this project open source so that other developers can add more to this project.

Class Diagram

Figure f. Our UML class diagram

Sequence Diagrams For Important Use Cases

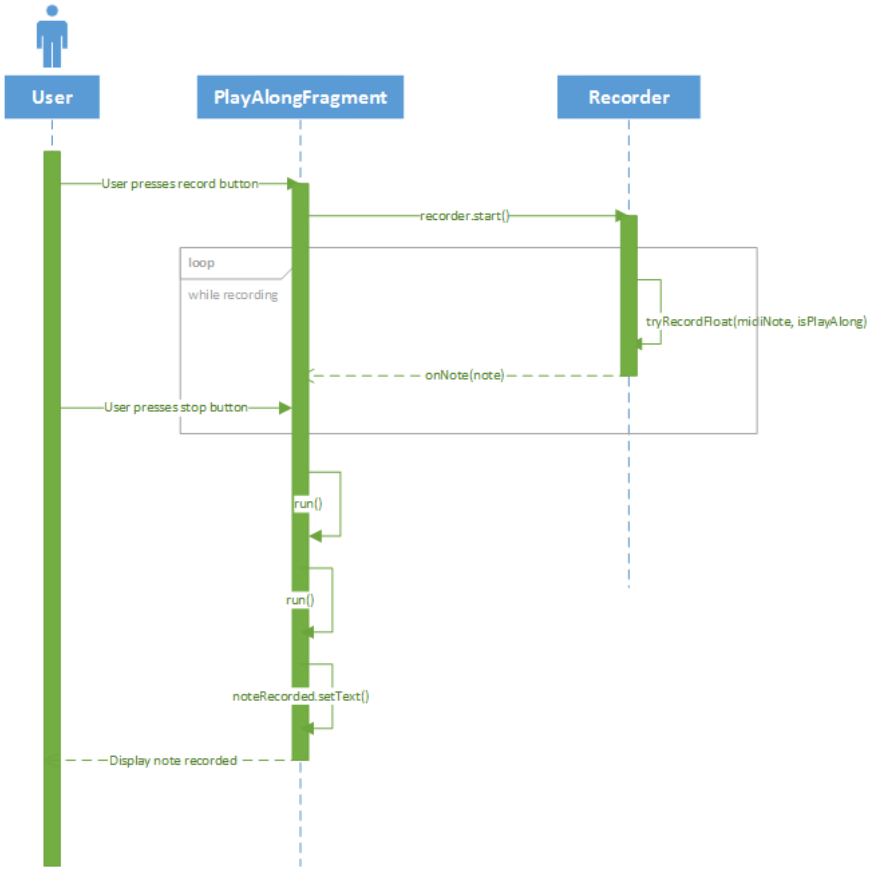

Figure g. Sequence diagram for use case: “I want to be able to record sounds and have the app create sheet music”

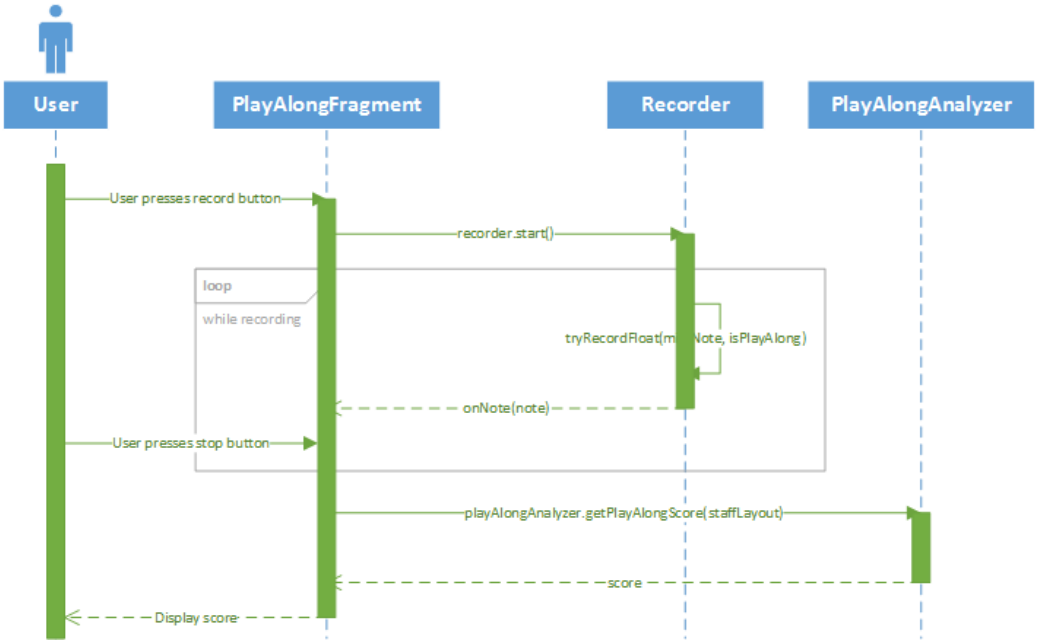

Figure h. Sequence Diagram for use case: “I want to be able to play along to previously created sheet music and be graded on my performance”

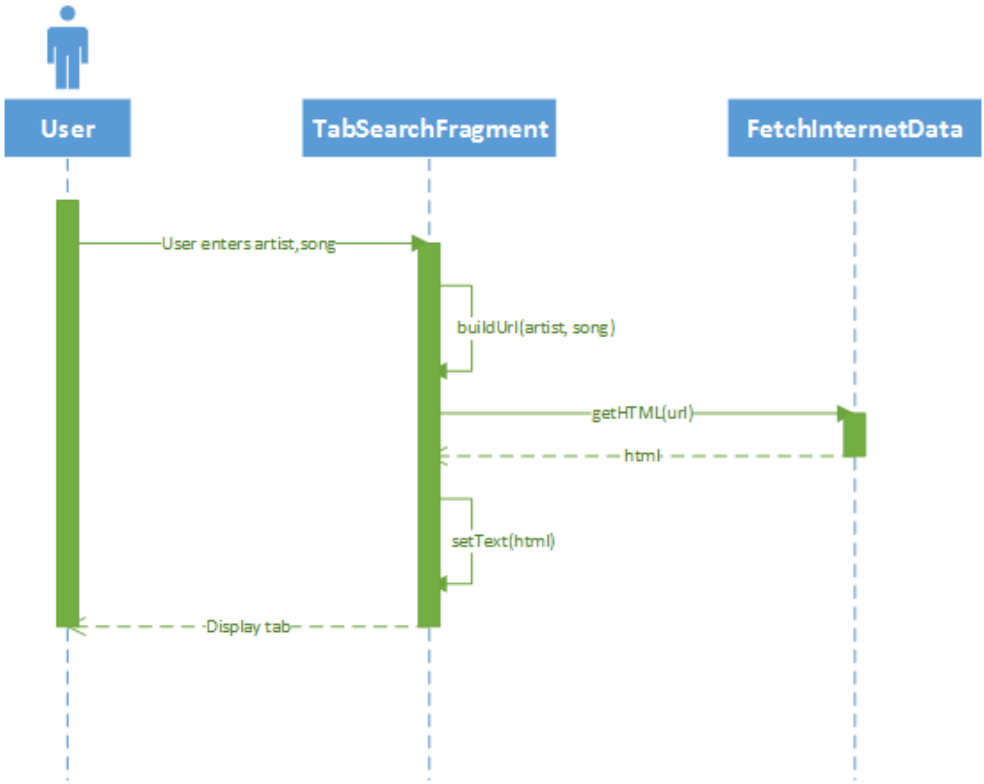

Figure i. Sequence Diagram for use case: “I want to be able to look up guitar tabs and view them in the app”