WHISTLE README

NATIONAL UNIVERSITY OF SINGAPORE

ORBITAL PROJECT 2022

JODY TNG JIN ZI
SHAYER AHMED

Proposed Level of Achievement: Artemis
Difficulty Attempted: Extreme

TABLE OF CONTENTS

1. **Foreword by developers**

2. **Project Files**
   - 2.1 Poster
   - 2.2 Video
   - 2.3 Source Code (Code Base)
   - 2.4 APK File

3. **Project Overview**
   - 3.1 Team Name
   - 3.2 Level of Achievement
   - 3.3 Motivation
   - 3.4 Aim
   - 3.5 Target Audience
   - 3.6 How is our Application unique?

4. **Intended List of Features to be implemented**
   - 4.1 List of Features (I)
   - 4.2 List of Features (II)

5. **Explanation of Algorithm**
   - 5.1 Brief Explanation of Algorithm
   - 5.2 Detailed Explanation of Algorithm
     - 5.2.1 Algorithm
     - 5.2.2 Framework
     - 5.2.3 Frequency Extraction
     - 5.2.4 Mapping

6. **Detailed Explanation of Actual Features developed**
   - 6.1 Basic Feature Implementation
   - 6.2 Algorithm Implementation

7. **User Stories**

8. **Development Timeline**

TABLE OF CONTENTS

9. Design
- 9.1 Architecture Diagram
- 9.2 UI Overview
- 9.3 User Flow Activity Diagram
- 9.4 Brief Wireframe
- 9.5 Overview of all Pages
- 9.6 UI Style Guide
- 9.7 Logo Design

10. Implementation
- 10.1 Tech Stack
  - 10.1.1 Version Control
  - 10.1.2 IDEs
  - 10.1.3 Dependencies
  - 10.1.4 Frontend
  - 10.1.5 Backend
- 10.2 Coding Practices

11. How to use our app?

12. Quality Assurance
- 12.1 Multi-Layered Testing
- 12.2 Algorithm Testing
- 12.3 User Testing

13. Responses to Evaluation
- 13.1 Milestone 1 Responses
- 13.2 Milestone 2 Responses
- 13.3 Milestone 3 Responses

14. Final Thoughts on our Orbital Project

15. Future Developments

16. Project Log
- 16.1 Project Log – Milestone 1
- 16.2 Project Log – Milestone 2

17. Conclusion

1. FOREWORD BY DEVELOPERS

1. FOREWORD BY DEVELOPERS

A good day to all our readers. Welcome to **Whistle, a music application which allows you to get your transcribed scores within minutes!**

This section serves to provide a **brief overview of our project**, and is mainly targeted for **prospective employers**. We have decided to include this to showcase our project for **future job applications and interviews**. Specifically, we hope this section can help guide prospective employers on how they can navigate through our project easily. As such, for the teams who have been with us throughout this 3-month long orbital journey, you would already have a good idea of the different features available in our application, as well as the structure of our project. Hence, it would be alright for you guys to skip past this section an jump straight into reading the actual README, even though we encourage you to read this foreword as well.

In this long document of our README, our team starts off by **providing the rationale behind for our choice of project**. From there, we then discuss the **various user stories** integral for our feature implementation, and thereby proceed to **elaborate on the different features available in our application.** Thereafter, we provide readers with a better understanding of how we have structured our code base through the **usage of the Unified Modelling Language and Entity Relationship Diagrams**, and also discussing how we have utilised **Cognitive Walkthrough to improve on our UX design**. Lastly, we also took a closer look at how we have tested our software, through the **various software engineering principles** we have adopted while working on our project.

Where appropriate, we have also discussed **the limitations and constraints** we have face at the **different implementation stages of our project**. We have also decided to include how our team has come together to resolve the problem as we believe it would allow readers to better understand that our application does have its limitations, and where possible, our team would also like to work on them even after the project ends. Till today, software engineers continue to fight bugs on a daily basis and we hope that you – the readers can understand that should our application is unable to meet your expectations. We would also like to rest assure you that we will continue to work on our application, but would also hope that you guys appreciate the effort that we have put in during this limited time frame of 3 months to come up with the best music application to our own abilities,

With that, we have structured our README to be readable by all readers, despite the different level of proficiencies in understanding the technical languages. We hope that you will enjoy reading the README, which is essentially a documentation of our entire 3 month long journey working on the Whistle application.

Yours Sincerely,
Jody Tng and Shayer Ahmed
NUS Business Analytics Year 1 Students

2. PROJECT FILES

2.1 POSTER

WHISTLE

Get your transcribed notes within minutes!

INTRODUCTION

Problem Statement

Many musicians have felt the frustration during music composition, especially when they are **unable to figure out the notes** that they have in mind when composing their song.

Objectives

Make the process of music composition easier; where our application writes out the notes for users, and all they have to do is just sing the tune out you have in mind.

Why Whistle?

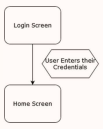
There are 2 similar applications in the app store that allow for note transcription, **"MusicTrans" and "Note Recognition".** Both applications also have very limited functionality by simply showing what notes are being played **but not allowing for users to export their melodies for future use.**

As such, our application **aims to resolve the issues faced** in current technology and **add additional features that music composers would wish to have.**

EVALUATION

Multi-Layered Testing

We conducted **Unit Testing**, **Widget Testing** and **Developer Testing** for each user action. A sample is shown as below.

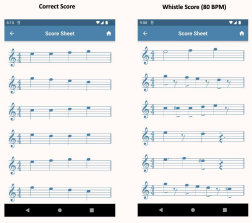


Test Type	Test	Pass/Fail
Unit	-NA-	-NA-
Widget	- Check if "Continue" Button leads to Home Screen	Pass
Developer	- Check if "Continue" Button leads to Home Screen	Pass
	- Check if users can type in any log in credentials	Pass

Accuracy of Transcribed Score

We also conducted Algorithm Testing. As shown below, our transcribed score is able to print out the correct notes and rests. There is also a **strong agreement between the actual score and the Whistle sample**.

Note Type	Image	Accurate?
Crotchet Note		Pass
Crotchet Rest		Pass
Quaver Note		Pass
Quaver Rest		Pass
Minim Note		Pass
Minim Rest		Pass
Whole Note		Pass
Whole Rest		Pass
Dotted Note		Pass
Dotted Rest		Pass



CONCLUSION

Interpretation of Results

The transcribed score is not perfect, but **best in the market as of now** as our application has managed to address the problems the only 2 similar ones in the current market.

Future Developments

1. Make the score sheet more accurate such that **repeated notes are not detected as a long note but instead individual notes**.
2. Users are able to **download the PDF** to their phones **instead of just storing them locally** on the Whistle application.
3. Include a **metronome function** in the application itself.

SYSTEM DESIGN

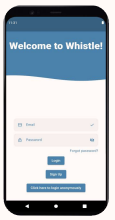
Key Features

1. Create new projects/ Return to old projects
2. **Analyse audio files** into manuscripts
3. Combine transcribed audio files
4. Users can **view individual manuscripts on the application**

Tech Stack

User Interface

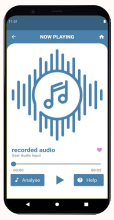
We chose **blue** as the primary colour for the app, since blue represents both the sky and the sea and is associated **with open spaces, intuition, imagination and inspiration**. This is in line with what we hope the users have when they compose their own songs through the Whistle application.



Login Screen

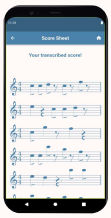
Home Screen

New Projects Page



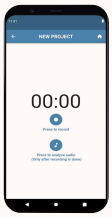
Audio Player Page

Loading Screen



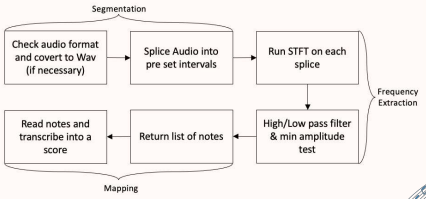
Score Sheet Page

Saved Projects Page



Record Audio Page

Explanation of Algorithm



Software Engineering Practices

1. **Refactored our documents** according to different pages of our app
2. Adhered to version control by creating new branches whenever we are working on an entirely new page.

Team Members: Shayer Ahmed, Jody Tng Jin Zi

2.2 VIDEO

The video created for our project documentation can be found at the link below:
https://drive.google.com/file/d/1b0JFx6hA_g5Xng79EpZCkBnSHKzPJ8wZ/view?usp=sharing

The video serves as an overview of the different features available on our application. For a more detailed explanation of the features available, we highly encourage you to read **Section 3 of our README.**

2.3 SOURCE CODE

Our source code can be accessed at: https://github.com/theshaydays/whistle.git
*Currently, our GitHub repository is public and can be accessed by all users.

2.4 APK FILES

The Android APK files for the Whistle Application can be downloaded from this folder:
https://drive.google.com/drive/folders/1PwAhSlAeXEKEL0o1hMMvBXozdLgXnxVM?usp=sharing

*If you are downloading on your Android phone, you may have to allow APK downloads from Google Drive and trust the developer of the app.

*If you are downloading it to an android emulator, download the file and drag it to the simulator. The application will install automatically.

*A folder of audio samples is also given. Simply download and drag the files into your emulator if you wish to use these samples on an emulator. Alternatively download these samples onto your android device and search the file in your file directory when using the app. The ideal BPMs for each file is written below.

NOTE: There are certain known errors and limitations in the current code, please refer to section 10 for more information when testing out the Whistle application.

Audio Sample	Ideal BPM
Happy Birthday - Xylophone	50
Happy Birthday - Whistle	50
Jingle Bells – Xylophone	80
Jingle Bells – Whistle	70
Ode to Joy – Xylophone	70
Ode to Joy – Whistle	80
Saints Go Marching On – Xylophone	80
Saints Go Marching On - Whistle	100

3. PROJECT OVERVIEW

3.1 TEAM NAME

We decided to name our team name Whistle, as we the inspiration of this project was to be able to see the transcribed score from a simple whistle. Our team also found this name simple and short, and hence decided to name our app Whistle, as it will be easier for users to remember as well.

3.2 LEVEL OF ACHIEVEMENT

Artemis.

Our team has decided to work on the **highest level of achievement for this year's Orbital project**. Both of us have a passion for music and we really hope to be able to come up with an improved version of the current music applications available on **Android play**, and we believe this will benefit users who are interested in the music making process.

We had a strong desire for Artemis because we want to have the best environment & resources available to us. We feel that resources such as mentorship are important because we can get guidance for design decisions regarding our project. Such guidance is quintessential to the success of this project. In addition, even though we may not be the most experienced in app-making, we believe that our passion and hard work will allow us to produce the music application that we envision it to be like described in this proposal.

Throughout the entire journey of our orbital project, our team was really grateful towards our student mentor, Miss Charisma Kausar and our mentor-in-charge, Mr Nishanth Elango for providing us with prompt replies whenever our group reaches out to them for help.

3.3 MOTIVATION

Singapore's music industry is relatively small compared to other countries, and Singaporeans also view our music scene as relatively low value due to pragmatism and other various reasons. As both of us have a passion for music, we feel that there is much **more potential and scope for our arts scene**, and hence would like to produce an application that is able to transcribe notes onto a score after detecting whistles or any musical sounds.

In addition, many musicians have felt the frustration during music composition, especially when they are unable to figure out the notes that they have in mind when composing their song. Figuring out the notes as they go along also slows down the entire music composing process. As we have friends currently pursuing music at the Yong Siew Toh Conservatory, we spoke to them about our current idea to create an application that transcribes whistles or any musical sounds into a score, and they also feel that it is something with **potential and they feel that it would benefit them to a large extent as well.** As such, wouldn't the process of music composition be made easier if an application writes out the notes for you, and all you have to do is just sing the tune out you have in mind?

This inspired us to **create an application that is able to access the in-built microphone in the users' phones and produce a musical score sheet based on the melody sung into the application.** The team's name is called whistle as people usually whistle their melodies out and record them on their phones.

3.4 AIM

We hope to make the **process of music composition much easier** for not just professionals, but also anyone who is interested in music composition. We hope to achieve this by coming up with an application that is able to **transcribe the notes onto a score**. All the application user needs to do is to sing out the tune that he/she has in mind, and the score will be generated for them within a short period of time. There will be a microphone feature on the application that is able to detect the whistles or any humming voices, and will be able to transcribe the notes as the music plays and the entire score sheet will be generated in a few minutes time. This application also makes music composition much easier for aspiring amateur music composers, and those composers without perfect pitch. Further improvements can also be made by allowing the application to recommend harmonies, melodies and dynamics used in the score sheet produced.

3.5 TARGET AUDIENCE

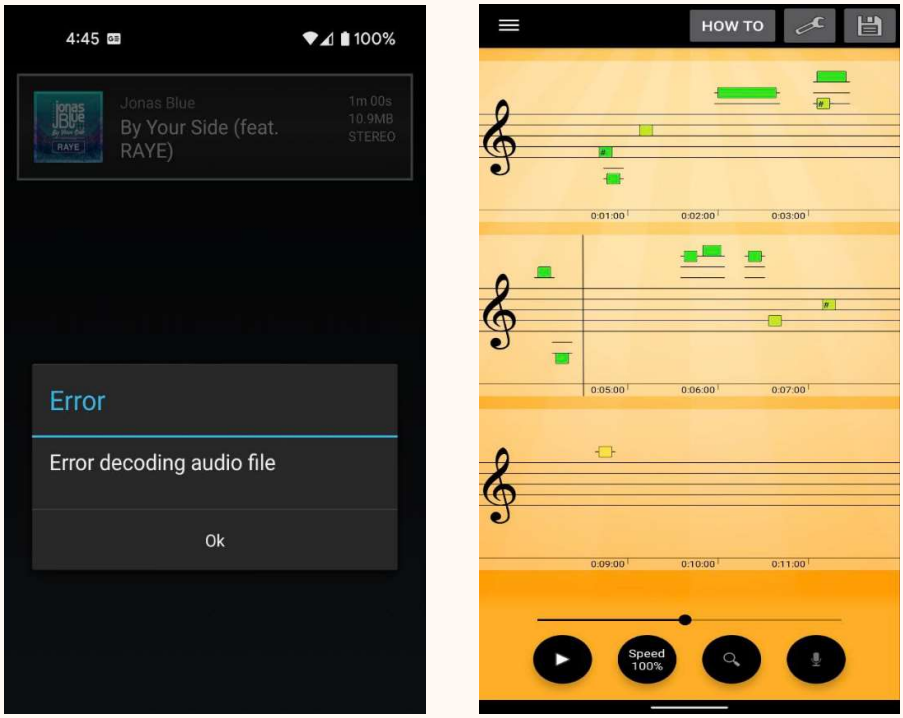
This project is mainly targeted at **professional musicians** who does music composition, as they will be able to **increase the efficiency of their music composition** through the usage of the Whistle application. It is also targeted at **beginners with no music background** but interested in music composition, to allow them to **pursue their interest in music composition**, **create nice melodies** and even a whole new song.

3.6 HOW IS OUR APPLICATION UNIQUE?

CURRENT APPLICATIONS IN THE MARKET

We have also done some research on the current music applications available on the App Store and Android Play Store. Currently there are 2 similar applications in the app store that allow for note transcription, **“MusicTrans” and “Note Recognition”.** However, the former app does not work well with recorded audio clips of people whistling or playing their instrument while the latter does not provide an accurate description of the notes that are played in the audio clip. Both applications also have very limited functionality by simply showing what notes are being played **but not allowing for users to export their melodies for future use.**

As such, our application **aims to resolve the issues faced** in current technology and **add additional features that music composers would wish to have.**



MusicTrans (Left) not allowing for personal audio clips to be analysed and
Note Recognition (Right) providing highly inaccurate and hard to read manuscripts

4. INTENDED LIST OF FEATURES TO BE IMPLEMENTED

4. INTENDED LIST OF FEATURES TO BE IMPLEMENTED

This **application** has a record function where it will record what you sing into a waveform audio file format. The algorithm in our application is able to identify the number of notes present in the audio file, and split the audio wave file into the number of musical notes identified. Afterwards, it will transform this audio wave file via an algorithm into a storable data type. The data will be matched to an existing database with musical notes and produce a singular musical note. After all the singular musical notes have been identified, it will then be printed through a graphical user interface where the composer can get to see the entire scoresheet of what he/she sang.

The application also has the potential to include other additional features as listed below. This then benefits them as they are able to transcribe music with greater ease and convenience.

Main feature of the application:

1. Transcribe audio input into a musical score.

(Additional features that were intended to be implemented can be found in the subsequent pages)

4.1 LIST OF FEATURES (I)

Feature	Elaboration	Technical Specification
Uploading of audio files	Users upload an mp3 file of themselves playing/whistling a melody.	Uploaded files will be saved into a user specific database for easy retrieval in the future
Score Transcription	Users will run the audio file through the digital signal processing algorithm* to view a manuscript depicting the notes played in the audio file.	mp3 file will be read and undergo digital signal processing in order to determine exact notes played in the audio recording Audio file will also have an associated MIDI file tagged to it detailing the transcribed notes present in the audio file
Score Review (Unable to complete)	Users can then listen to the melody recorded through an instrument of their choice and edit any notes to their liking.	MIDI file will then be edited according to consider note and instrument changes and replayed back to the user when they wish to do so.
App Recommendations* (Unable to complete)	Users can then view recommended accompaniments and chords that would sound well with their uploaded melody.	With the given notes, the app would suggest appropriate chords that would suit well by determining the key signature of the melody through a machine learning model.

*will only be implemented if we have sufficient time

4.2 LIST OF FEATURES (II)

Feature	Elaboration	Technical Specification
Music composition (Unable to complete)	Users can then combine multiple scores and melodies together to make a full composition of multiple instruments.	A new separate GUI will be created to allow users to combine multiple scores together and make a harmonious composition.
Exporting (Unable to complete)	Users can then export their manuscripts, melodies (individual and combined) and any MIDI files to their device. Exporting MIDI files will allow users to continue this project on other music composition applications such as Fruity Loops and Logic	App will write PDF files (for manuscript), mp3 files (for melodies) and MIDI files (for future uses in other music development programmes) to the user's device.

5. EXPLANATION OF ALGORITHM

5.1 BRIEF EXPLANATION OF ALGORITHM

Whistle's algorithm is design around **Digital Signal Processing**

Step 1: Reading audio file (Upload audio file into app)

Step 2: Using differences in volume levels throughout the audio, split the audio into different notes.

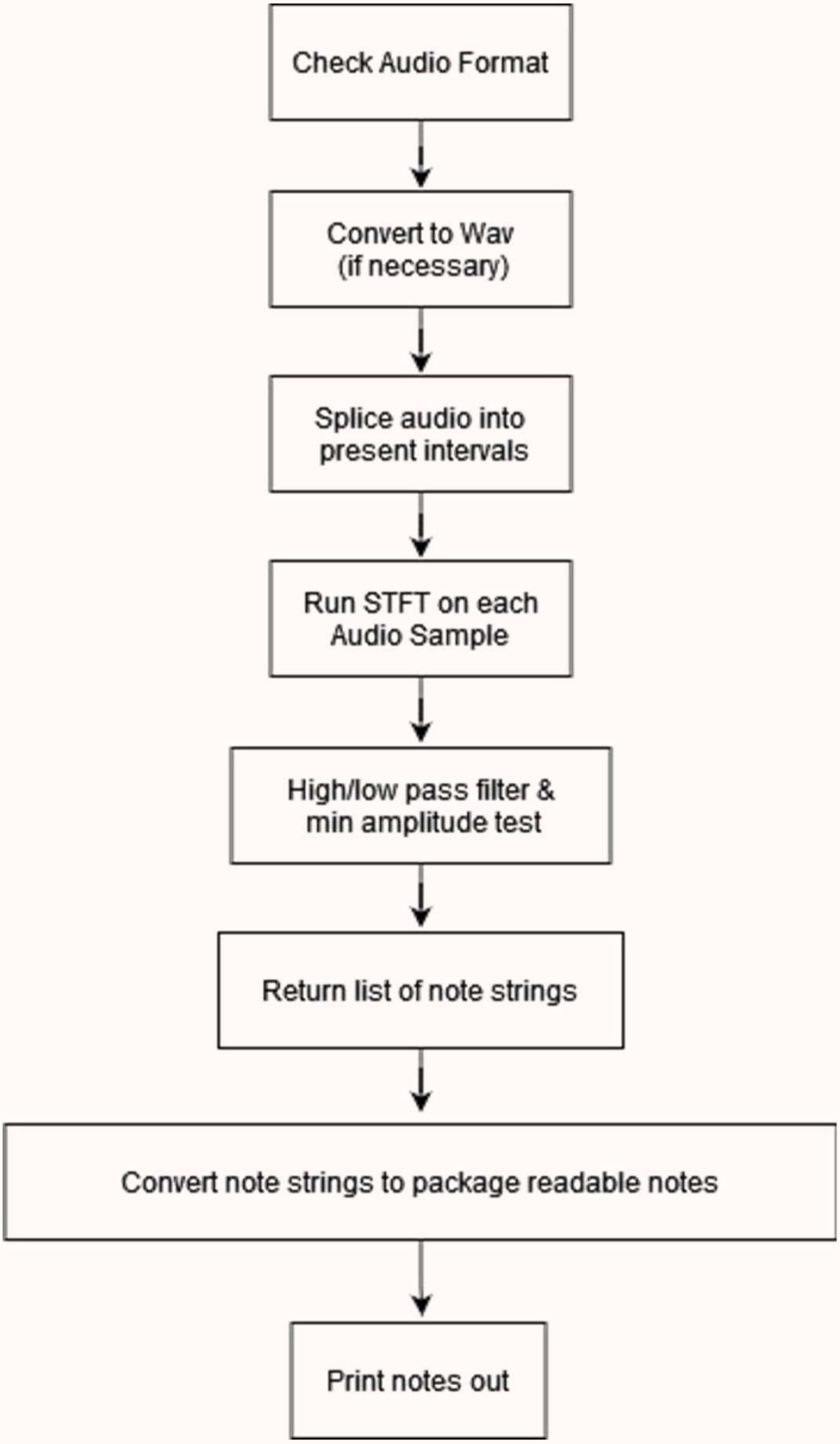
Step 3: Use short-time fourier transform present in via the Java and swift libraries to help identify which frequency is most prominent in that audio file in each short time period.

Step 4: Match the frequency to corresponding MIDI note numbers to identify the notes.

Step 5 (additional features): Given the input melody, run the input through a trained model to output suggested harmonies and chords. The model will be trained via input given from various existing musical compositions.

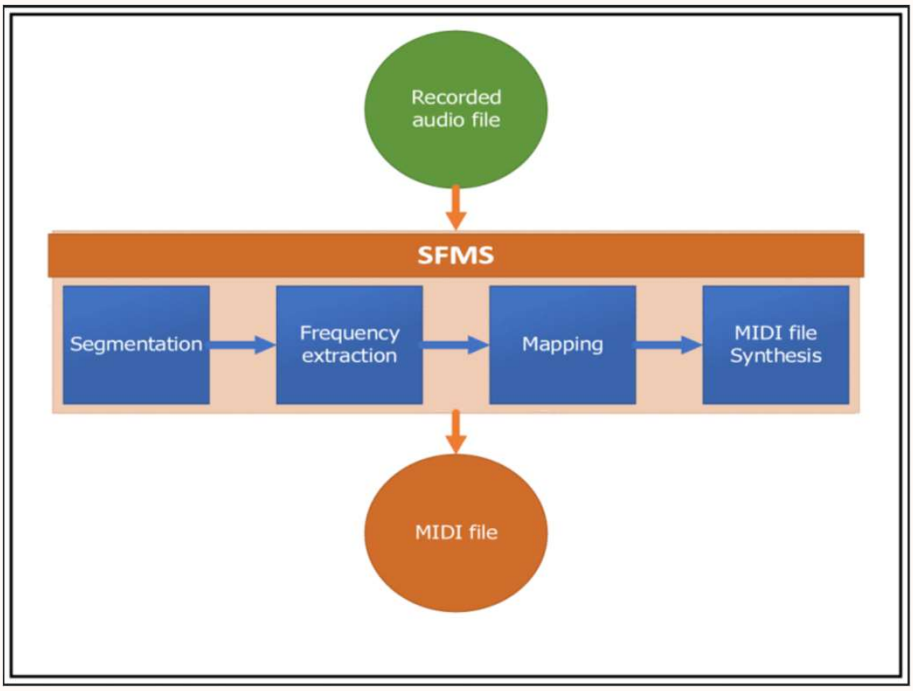
5.1 BRIEF EXPLANATION OF ALGORITHM

A Diagrammatic representation of the algorithm is shown below.



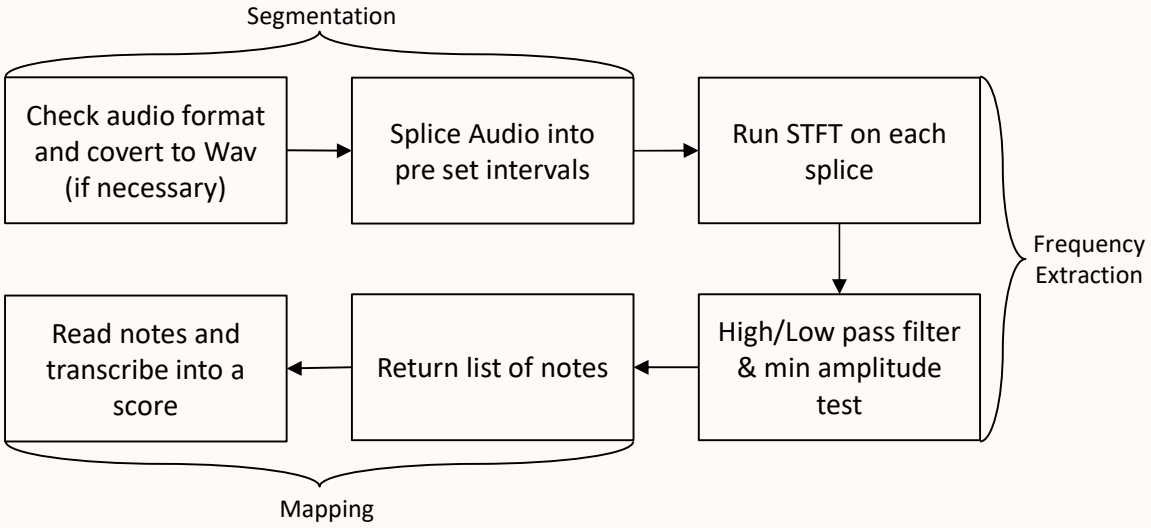
5.2.1 DETAILED EXPLANATION OF ALGORITHM - FRAMEWORK

To implement a proper framework into our algorithm implementation, we **conducted literature review** in the field on **note analysis and extracting melodies** from whistles. We then decided to design our algorithm framework similar to Danayi and Seyedin's algorithm based on time-frequency analysis for extracting melody[1].



Danayi and Seyedin's framework for note analysis

Whilst they presented excellent results for their algorithm, the inflexibility in Flutter's FFT as well as our limited timeframe resulted us in creating a similar but simplified framework specially designed for this app.



1. A. Danayi and S. Seyedin, "A novel algorithm based on time-frequency analysis for extracting melody from human whistling," 2018 4th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), 2018, pp. 135-139, doi: 10.1109/ICSPIS.2018.8700531.

5.2.2 DETAILED EXPLANATION OF ALGORITHM - SEGMENTATION

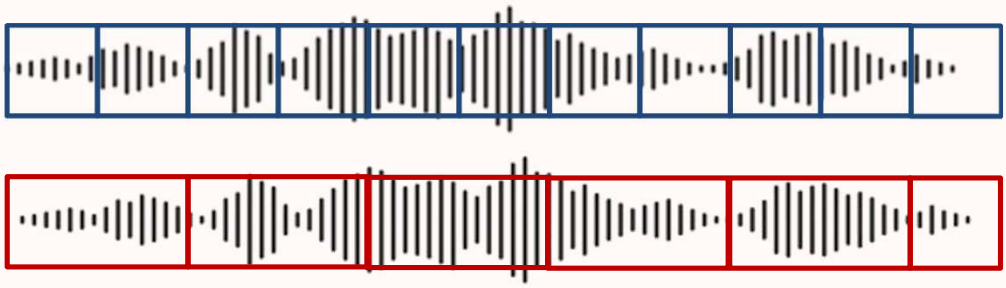
File format

The first step to analysing the audio file is to convert it to a **Wav format**. The file can then be read as a Float64List which can be read by the FFT transform. We accomplish this by using a very well known file converted, **FFMPEG**.

However, due to the nature of FFMPEG, we acknowledged that **m4a files cannot be converted to Wav files** as the FFMPEG implementation in dart is unable to convert such files. This is often an issue as most modern voice recorders record their files in an m4a format, disallowing many files to be used on our app unless converted by an external file converter. Nonetheless, we alleviated this problem by implementing our own voice recorder in the app.

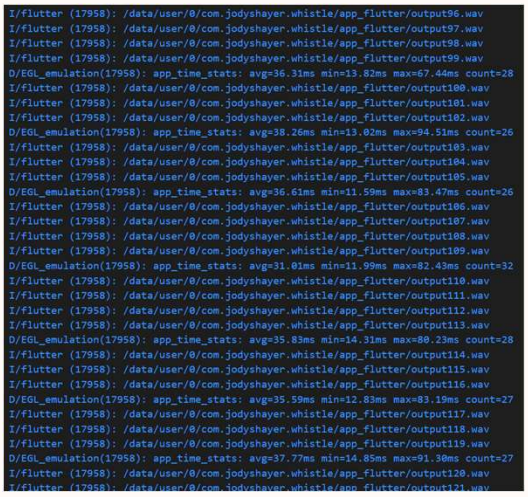
Audio splicing

Similar to the audio conversion, we used FFMPEG to splice the audio. The app takes in a user input of whatever BPM (Beats Per Minute) the audio sample is in and the audio is **sliced into quavers (half a beat)** related to the BPM input. Thus an input of 120BPM would splice the audio into 0.25s intervals while an input of 90 BPM would slice the audio into 0.375s intervals.



Saving Audio Splices

These splices are indexed and then **saved into the apps temporary storage** to be analysed later via frequency extraction. All files are then renamed to "output" such that future audio files will rewrite previous audio files in order to save memory.

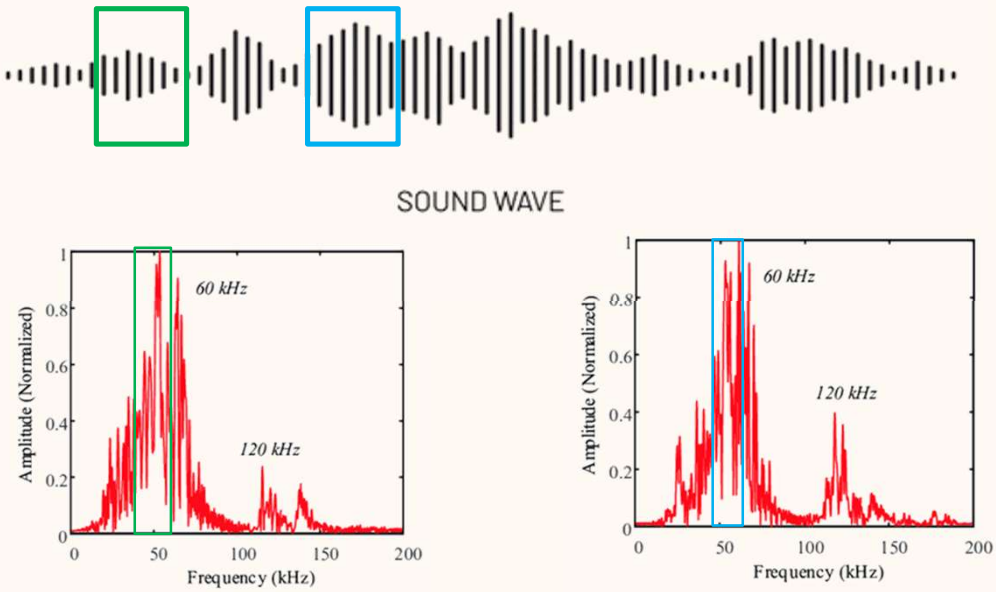


Representation of how the audio is spliced based off different BPM inputs.

5.2.3 DETAILED EXPLANATION OF ALGORITHM – FREQUENCY EXTRACTION

FFT analysis

Now that the audio has been spliced, we can retrieve each indexed audio splice and run an FFT transform on them.



Note: Graphs and audio wave are just a representation of how FFT analysis works

The diagrams above shows an example of the FFT Output wave that will be produced after the analysis of 2 audio splices. The **blue and green boxes** shows that the amplitude of the detected sound and thus return two different frequencies that appear most prominently in each sound sample.

```
I/flutter (19532): Key frequency is 609.375
2 I/flutter (19532): Key frequency is 515.625
I/flutter (19532): Key frequency is 796.875
D/EGL_emulation(19532): app_time_stats: avg=314.44ms min=13.04ms max=626.89ms count=4
I/flutter (19532): Key frequency is 796.875
4 I/flutter (19532): Key frequency is 703.125
D/EGL_emulation(19532): app_time_stats: avg=303.99ms min=14.62ms max=599.65ms count=5
2 I/flutter (19532): Key frequency is 515.625
2 I/flutter (19532): Key frequency is 1078.125
D/EGL_emulation(19532): app_time_stats: avg=331.87ms min=12.83ms max=651.31ms count=4
2 I/flutter (19532): Key frequency is 890.625
I/flutter (19532): Key frequency is 703.125
D/EGL_emulation(19532): app_time_stats: avg=337.24ms min=328.09ms max=342.75ms count=3
I/flutter (19532): Key frequency is 703.125
2 I/flutter (19532): Key frequency is 656.25
D/EGL_emulation(19532): app_time_stats: avg=250.33ms min=13.13ms max=331.54ms count=4
3 I/flutter (19532): Key frequency is 609.375
I/flutter (19532): Key frequency is 46.875
D/EGL_emulation(19532): app_time_stats: avg=315.24ms min=304.83ms max=328.37ms count=4
2 I/flutter (19532): Key frequency is 1078.125
2 I/flutter (19532): Key frequency is 890.625
D/EGL_emulation(19532): app_time_stats: avg=317.28ms min=11.46ms max=624.36ms count=4
2 I/flutter (19532): Key frequency is 703.125
I/flutter (19532): Key frequency is 796.875
D/EGL_emulation(19532): app_time_stats: avg=267.34ms min=14.22ms max=396.66ms count=4
I/flutter (19532): Key frequency is 796.875
2 I/flutter (19532): Key frequency is 703.125
D/EGL_emulation(19532): app_time_stats: avg=336.45ms min=330.20ms max=343.15ms count=3
2 I/flutter (19532): Key frequency is 703.125
2 I/flutter (19532): Key frequency is 46.875
D/EGL_emulation(19532): app_time_stats: avg=316.65ms min=285.06ms max=341.26ms count=4
I/flutter (19532): Key frequency is 93.75
```

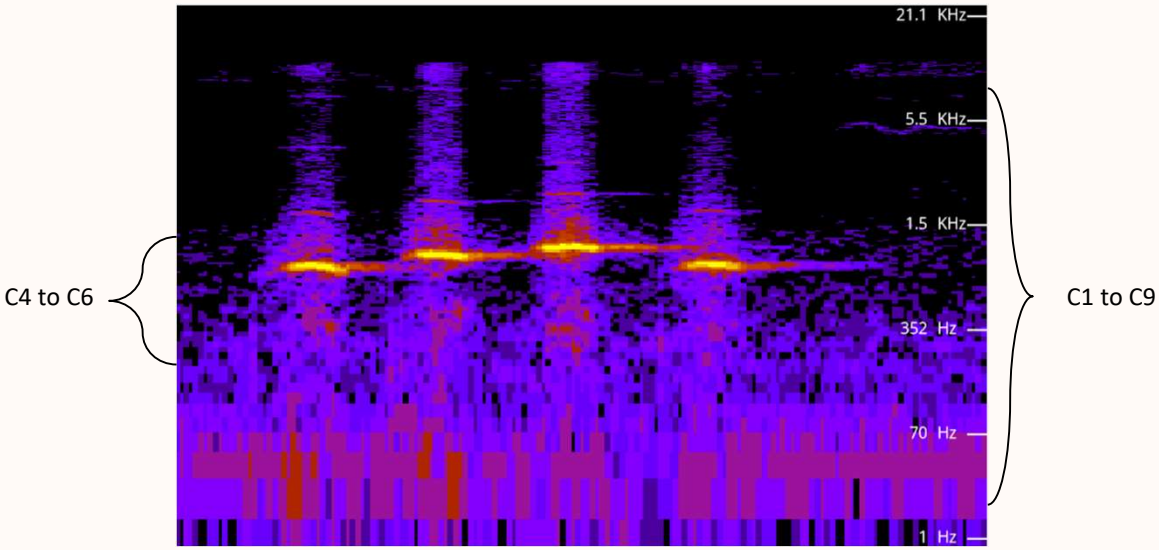
Each splice then returns a certain frequency which is then collected for future analysis and filtering. From this, we are able to get a **list of the main frequencies** that have been played throughout the audio sample.

5.2.3 DETAILED EXPLANATION OF ALGORITHM - FREQUENCY EXTRACTION

Frequency Checks

We note that the human hearing range is from 20Hz to 20,000Hz. Through further study, we also recognised that a humans whistle ranges from **500HZ to 5000Hz**[2]. This is equivalent to the note range of C5 to E8. However, noting that the treble clef generally recognises notes from C4 to C6, we decided to limit our score sheet to just the note range of the treble clef, with the FFT specifically focusing on **C5 to C6**.

We confirm this analysis by visualising a whistle sample through a spectrum analyser.



Frequency spectrum of a 4 note whistle sample modelled on a logarithmic scale

We notice that the notes appear clearly at the densely populated regions, which are equivalent to the peaks seen in the FFT graphs shown on the previous page. Thus, by restricting the frequencies **to C4 to C6**, we are able to get more accurate representations of what frequency is played at each splice. If the region returns a frequency **outside that range**, we simply report the value at 0.00Hz.

2. Nilsson, M., Bartunek, J. S., Nordberg, J., & Claesson, I. (2008, May). Human whistle detection and frequency estimation. In *2008 Congress on Image and Signal Processing* (Vol. 5, pp. 737-741). IEEE.

5.2.4 DETAILED EXPLANATION OF ALGORITHM - MAPPING

Note Mapping

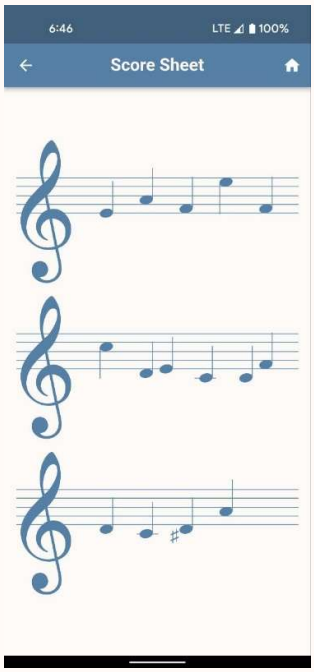
After obtaining the relevant frequencies for each audio splice, we are able to **map them to the relevant note** that matches closest to the frequency given. If two consecutive frequencies map to the same note, we merge them to represent a longer frequency. However, audio splices which return 0.00Hz as their frequency will then be mapped to 'rest', indicating that there is no audio of interest in that time period. From this we are able to obtain a readable list for our score painter to read and create a score. An example of such a list is shown below.

```
I/flutter (19532): [[C5, 1.2], [D#5, 0.6], [rest, 0.6], [C5, 0.6], [rest, 0.6], [F5, 0.6], [rest, 0.6], [E5, 1.2], [rest, 1.2], [C5, 1.2], [D#5, 0.6], [rest, 0.6], [C5, 0.6], [G5, 1.2], [rest, 0.6], [F5, 0.6], [rest, 1.7999999999999998], [C5, 0.6], [C#6, 1.2], [rest, 0.6], [A5, 0.6], [rest, 0.6], [F5, 0.6], [rest, 0.6], [E5, 0.6], [D#5, 1.2], [rest, 1.7999999999999998], [C#6, 0.6], [A5, 1.2], [rest, 0.6], [F5, 0.6], [rest, 0.6], [G5, 0.6], [rest, 0.6], [F5, 0.6], [rest, 1.7999999999999972]]
```

Score Painting

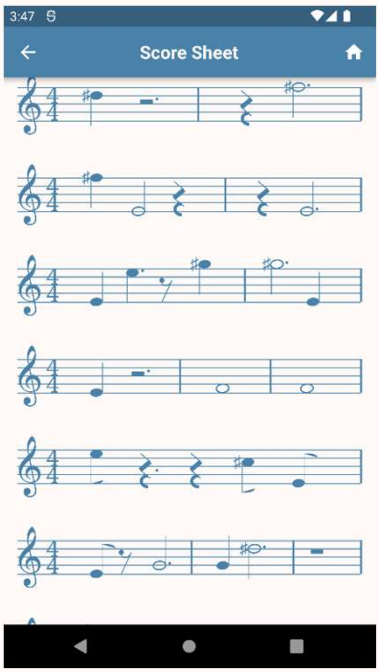
In order to represent this list as a score, we **create a custom score painter**. The score painter creates an empty stave with no notes, then paints each note with respect to the values given by the list given by the frequency extraction steps.

To create such a score painted, we **forked a score painter made from craigomac and santoxyz on Github**. We then **modified their code to not only represent crotchets, but notes and rests of different time durations**. We also modified the list in order to represent the notes as full bar lengths of 4 beats to make the score look as realistic as possible.



Initial score painter

Working off an open-sourced score painter



Final score painter

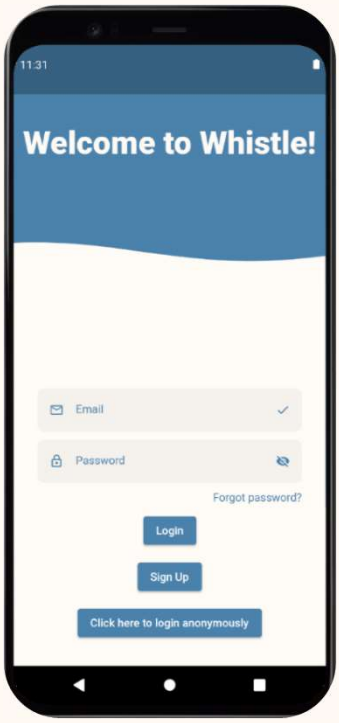
With that, a full run through of the algorithm is shown from an audio sample to a score.

6. DETAILED EXPLANATION OF ACTUAL FEATURES IMPLEMENTED

6. DETAILED EXPLANATION OF ACTUAL FEATURES DEVELOPED

On review, we came to realise that we were **overly ambitious** in the features that we wished to implement, especially due to the difficulty of digital signal processing. However, we are still proud to report that **we managed to clear our primary objective** of transforming audio into notes with good accuracy, a feature that is underdeveloped in the music scene.

6.1 BASIC FEATURE IMPLEMENTATION



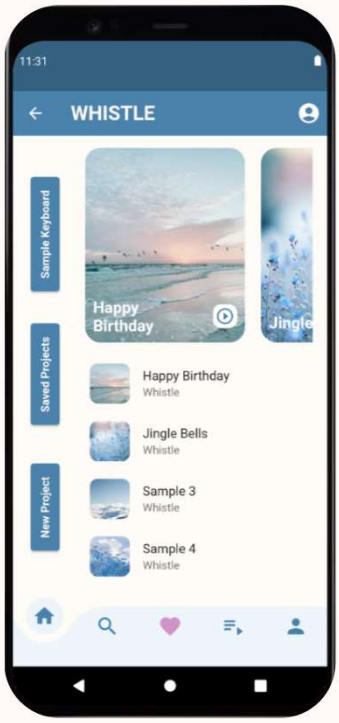
UI/UX Design:

As soon as the application is launched, the user is taken to the login page as shown above. Our team has decided to make the login page simple, with the top part of the page showing a **wave** moving. The intention of this design was because the main purpose of the Whistle application was to take in an **audio wave** and produces a transcribed score through the algorithm of our application. As such, we decided to design our login page with the ”wave” pun.

Implementation:

User authentication features are currently not implemented yet, but our team will be continuing to work on the login feature using Firebase authentication with email and Google logins enabled. We are also intending to have Flutter Toasts displayed for any error messages while submitting forms. A new user is created in our Cloud Firestore database upon sign up. For the implementation of social login, we are also intending to add in the google_sign_in dependency.

6.1 BASIC FEATURE IMPLEMENTATION



UI/UX Design:

For the home page, our team decided to split the page into a few segments. For the left-hand side, it would be the main buttons for the user to navigate into other pages. We have also decided to include a **bottom navigation bar** at the bottom of the screen, and similarly, the bottom navigation bar has a “wavy” component to it. Due to the lack of time, we were unable to implement functionalities to the icons, but our team will also be continuing to work on them even after milestone 3. As for the middle part of the screen, we felt that it would be convenient for the user to look at the top few current projects that they are working on, and they can easily click onto them and continue working on it after they have logged in to their account. We have included a scrollable feature, so that the users are able to scroll ”left”, “right” and “up”, “down” respectively.

Implementation:

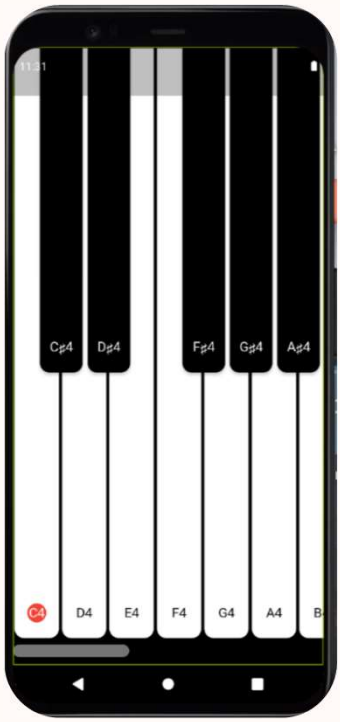
The Home Page showcases 3 minor features that enhance the UI/UX of the app.

Firstly, audio samples are presented in **ListTiles** wrapped in a **SingleChildScrollView** to enable infinite scrolling adapted to any and all phone sizes.

Key action buttons such as “New Project” were designed with the Open-closed principle. As such they are wrapped in a **NavigationRail** which allows us as developers to add more buttons in the future without requiring modification to current existing buttons.

The bottom bar is then designed with a **CurvedNavigationBar** to improve aesthetic quality as well as improve fluidity of the app.

6.1 BASIC FEATURE IMPLEMENTATION



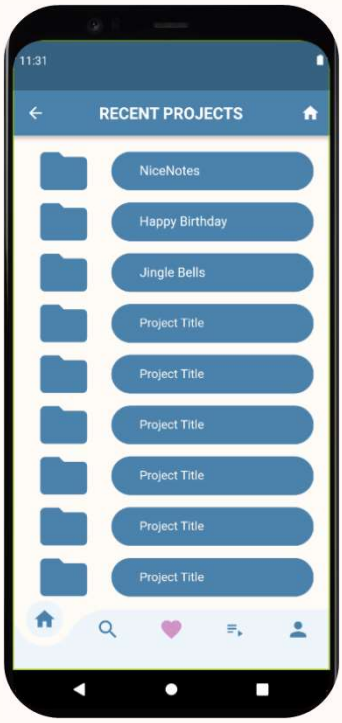
UI/UX Design:

There was not much to design for this page, because we wanted the keyboard to take up the entire page of the application. We felt that this was easier and more convenient for the user to use the keyboard. As we could have a choice for the number of octaves to be displayed on the screen, we have decided to show from C4 to A5, which is the standard singing range for most people.

Implementation:

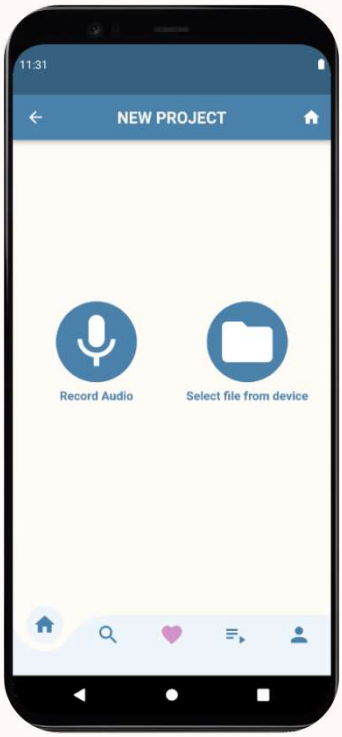
The UI of this feature was implemented from the **piano.dart** package (see dependencies). To implement piano notes, an audio player from the **just_audio.dart** package was utilised, which taps into the device's native audio player. A library of piano notes was then downloaded from the University of Iowa to be used as a reference for all the piano notes. With that, every note press creates a new audio player instance which plays the respective note. To ensure the app does not take an excessive amount of memory, all audio players are disposed when leaving this page.

6.1 BASIC FEATURE IMPLEMENTATION



UI/UX Design: Our team had the intention to make this page as simple as possible, so that the user can easily identify their recent projects. Similarly, the bottom navigation bar remains for all pages for consistency.
Implementation: Similar to the home page, a list of saved projects are created in a **SingleChildScrollView** to ensure infinite scrolling. We have also implemented it in such a way that the user can **change the name of their project easily** by clicking on the space beside the “folder” icon.

6.1 BASIC FEATURE IMPLEMENTATION



UI/UX Design:

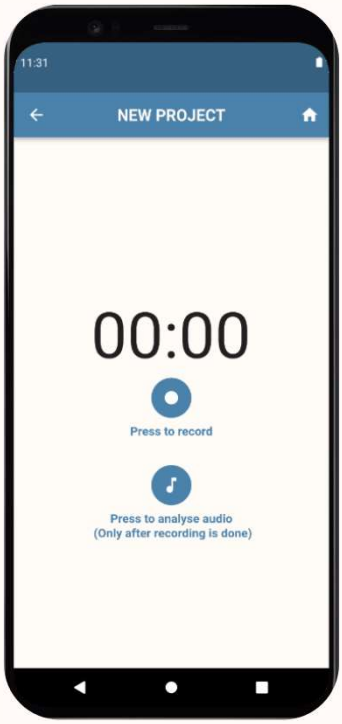
This page was designed such that it would be self-explanatory for users to use our application. It was designed such that users will understand that there are only 2 options available to them, to either “record audio” or “select file from device” before they can obtain their transcribed score. Similarly, the bottom navigation bar remains for all pages for consistency.

Implementation:

In this page, device storage access is requested on selecting a file from device. We also implemented additional checks on whether permission was given or denied. The app then throws the user an error popup to notify and prevent access to moving on to the next page is permissions were denied.

Implementing the ability for users to pick files was done with the help of the **file_picker.dart** package. We ensured that only audio files could be selected to ensure that the audio player on the next page would not throw an unsupported file exception.

6.1 BASIC FEATURE IMPLEMENTATION



UI/UX Design: We have included a timer once the user presses the record button, to allow users to know the length of their audio as they are recording.
Implementation: Upon opening this page, a recorder is initialized. Similar to the previous page, the microphone access is requested upon recorder initialization, which throws an error popup if permissions were denied. In order to tap into the device's native recorder, the **flutter_sound.dart** package was implemented. From there, basic recorder settings, such as play, pause and record, were set in order to give users control of the audio recorder. The recording button then reactively changes icon depending on the state (playing, paused, stopped) of the recorder. The recorded audio is then saved into the device cache which then can be read, played back and analysed on the next page.

6.1 BASIC FEATURE IMPLEMENTATION

UI/UX Design:

For simplicity and standardization purposes, we have decided to use the same image for every audio the user is listening to. The main functionality of the page would be be the slider feature for the user to fast forward or go back to any part of the audio he/she wishes to listen to as we believe that this will help to bring convenience to them. The other buttons such as the “Analyse”, “Play” and “Help” button are self explanatory, so that the user can use our application conveniently.

Implementation:

The main feature behind this page would be the ability for users to play audio and drag it to any position in the song. Firstly, a single audio player instance is created to tap into the device’s native audio player. The audio file is the taken from the user’s selected audio file from their phone’s device or from the app cache (stored by the audio recorder).

```
Stream<PositionData> get _positionDataStream =>
    Rx.combineLatest3<Duration, Duration, Duration?, PositionData>(
        _audioPlayer.positionStream,
        _audioPlayer.bufferedPositionStream,
        _audioPlayer.durationStream,
        (position, bufferedPosition, duration) => PositionData(
            position, bufferedPosition, duration ?? Duration.zero));
```

The seek bar then checks for the which position in the audio the audio player is currently playing and dynamically updates it to ensure a continuous moving seek bar. The player also overrides the audio player when the user manually changes the position of the seek bar and forces the audio player to play the audio at the position of the song.

6.1 BASIC FEATURE IMPLEMENTATION

UI/UX Design:

We have decided to feature our application logo while the algorithm takes time to generate the transcribed score sheet. We have decided to make the entire loading page black instead of the theme colour of our application (blue) as we felt that there was a need to differentiate between the main pages of our app and the loading screen.

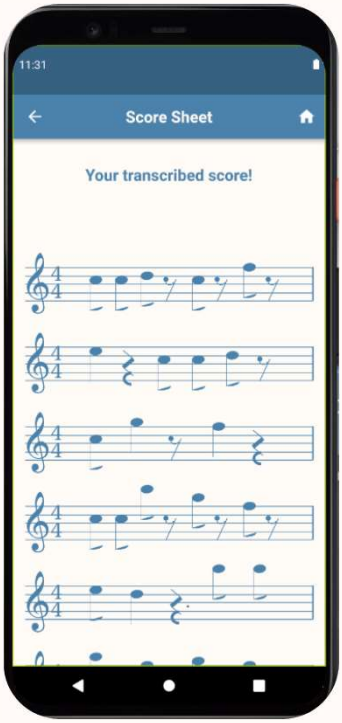
Implementation:

This loading screen was created with the help of **flutter_spin_kit.dart**. Adapted off the design shown on the bottom right, the spinner was customized to be more relevant to our app.

The images are also randomly generated on each initialization from a library of images. With the open-closed principle in mind, the library was also designed to accept an infinite number without changing any previous functionality.

```
Future _initImages() async {
  // >> To get paths you need these 2 lines
  final manifestContent = await rootBundle.loadString('AssetManifest.json');
  final Map<String, dynamic> manifestMap = json.decode(manifestContent);
  // >> To get paths you need these 2 lines
  final imagePaths = manifestMap.keys
      .where((String key) => key.contains('images/LoadingScreen/'))
      .where((String key) => key.contains('.jpg'))
      .toList();
  setState(() {
    images = imagePaths;
    randomList = List.generate(images!.length, (index) => index + 1);
    randomList?.shuffle();
  });
}
```

6.1 BASIC FEATURE IMPLEMENTATION



UI/UX Design:

Our team has included the sentence “Your transcribed score!” at the top part of the page, as we felt that the users will feel a sense of excitement to see their score generated from an audio file. In addition, we felt that it was necessary to include that line for clarity purposes instead of just showing the transcribed score on the page. We have also made the score sheet scrollable for ease of viewing.

Implementation:

As mentioned earlier, the current score painter is adapted off an open-source score painter. Our implementation focuses on adding addition features such notes of different timings and rests. By introducing a noteLength property, notes and rests of differing rests could be added. In order to ensure that notes and rests are drawn correctly despite the size of the screen, the notes are drawn respective to the bounds of the device’s screen. As such, we are then able to draw individual notes and rests which are sized correctly despite differing screen sizes. An example of how a crotchet is drawn is shown below.

```
final ovalRect = Rect.fromLTWH(
    bounds.left +
        clefSize.width +
        (bounds.width - ovalWidth * 1.5 - clefSize.width) *
            noteImage.offset,
    bounds.height - (noteIndex * noteHeight) - noteHeight / 2,
    ovalWidth,
    ovalHeight);
canvas.save();
canvas.translate(ovalRect.left, ovalRect.top + noteHeight * 0.1);
canvas.rotate(-0.1);
_notePaint.color = noteImage.color ?? noteColor;
```

6.2 ALGORITHM IMPLEMENTATION

Main Packages Used

To implement our algorithm, we utilized 3 main packages: **ffmpeg_kit.dart, fftea.dart** and **piano.dart**. These packages help us in the segmentation, frequency extraction and mapping phases of the algorithm.

Segmentation

In this portion, we are required to obtain information regarding the audio sample (like duration and name), as well as edit the audio piece to our needs (such as slicing and conversion). We are able to achieve this with the help of FFmpeg. However, to integrate this package into our app, we utilized the ffmpeg_kit.dart package which emulates the Ffmpeg environment within dart code. From there, we initialize FFmpeg sessions where we can input commands to help us slice and convert audio. From there, the audio files are cached into our desired output path for future reference.

```
Future<String> convertFile() async {
  String path = this.filePath;
  Directory appDocumentDir = await getApplicationDocumentsDirectory();
  String rawDocumentPath = appDocumentDir.path;
  String fullFileName = this.filePath.split('/').last;
  String fileName = fullFileName.split('.').first;
  String outputPath = rawDocumentPath + "/" + fileName + ".wav";
  String command = '-i ' + path + ' ' + outputPath;
  FFmpegSession sess = await FFmpegKit.execute(command);
  //print('issit the entire thing ' + outputPath);
  return outputPath;
}
```

An example of how a FFmpeg session is initialized and executed

Frequency Extraction

The main objective of this portion is obtaining the frequency of an audio sample using a fast fourier transform (FFT). However, in our search for a suitable FFT transform, we discovered the need to balance speed and accuracy. While the name does suggest the process is fast, splicing, converting and conducting a FFT on over 500 samples (for a 2 min sample) would take an increasing amount of time. As such, implementing transforms which are efficient and method calls which do not have exponential time were of the essence. More details about our discussions and implementation of this section will be done later in “Speed vs Accuracy”.

Note Mapping

The last part of the algorithm left is implementing a note mapping procedure that will convert Strings of notes and timings into actual visual images. As mentioned earlier on the discussion of the scoresheet page, we created a custom score painter (adapted from the **piano.dart** package) which helps us paint notes/rests of the correct length. To implement notes of different key values (C6, D5, etc), each note was already designated a certain position on the stave. All that’s left is creating the number of staves required and painting each note with each note having an equal amount of spacing between each other.

```
int get pitch {
  int offset;
  switch (this.note) {
    case Note.C:
      offset = 24; // C1
      break;
    case Note.D:
      offset = 26;
      break;
```

Notes being assigned a “height” to show how high they are from the bottom of the stave

```
noteImages.add(NoteImage(
    isPause: false,
    //60 referes to seconds in a minute
    noteLength: (noteResults[i][1] / (60 / BPM)) / 4,
    notePosition: NotePosition(
        note: noteInfo[0], accidental: noteInfo[1], octave: noteInfo[2]),
    offset: (i) * spacing)); // NoteImage
```

Notes being iteratively added into a list of notes which the score painter can read and draw with the correct spacing

6.2 ALGORITHM IMPLEMENTATION

Choosing the right FFT

There are many dart packages which already have in build FFT transforms that would accomplish our goal. In order to distinguish which FFT transform would be most suitable, we used the following table as a reference on speed of FFTs.

Size	package:fft	smart	smart, in-place	scidart	scidart, in-place*	fftea	fftea, cached	fftea, in-place, cached
16	424.4 us	33.5 us	23.3 us	359.8 us	133.6 us	71.0 us	48.2 us	38.7 us
64	657.7 us	7.6 us	2.9 us	310.2 us	271.8 us	136.8 us	134.6 us	103.5 us
256	614.1 us	15.5 us	11.1 us	984.9 us	1.02 ms	100.3 us	51.9 us	38.3 us
2^10	1.74 ms	50.0 us	46.2 us	9.14 ms	9.17 ms	39.5 us	25.7 us	23.5 us
2^12	8.01 ms	219.7 us	203.1 us	133.10 ms	138.19 ms	119.8 us	109.9 us	104.8 us
2^14	39.69 ms	903.5 us	860.3 us	2.15 s	2.03 s	677.9 us	536.0 us	436.2 us
2^16	225.97 ms	5.36 ms	4.76 ms	42.53 s	42.71 s	3.43 ms	3.14 ms	2.21 ms
2^18	1.21 s	27.89 ms	25.84 ms	Skipped	Skipped	12.95 ms	12.53 ms	10.99 ms
2^20	7.25 s	164.35 ms	149.33 ms	Skipped	Skipped	89.84 ms	85.69 ms	74.99 ms

Source: fftea.dart

While other packages provided FFTs with greater functionality, we decided to employ the **fftea.dart** package to conduct our FFTs as it performed the best for our use cases, especially at greater sizes.

FFT accuracy

The next discussion we undertook was determining what accuracy to set for the FFT. The "accuracy" determines the number of divisions in the audio spectrum that the FFT will make. As such, a larger accuracy would lead to a slower computation times. However, FFT transforms view the audio spectrum linearly, while the frequency of notes increase logarithmically. As such, we notice that at lower accuracies, lower frequency notes (such as C4 and C#4) get binned into the same frequency, losing resolution. As such, we managed to determine the optimal accuracy that would be able to accurately split notes, while reducing the speed of the algorithm as much as possible. The next page shows how we were able to determine that accuracy.

```
//setting up stft
final chunkSize = pow(2, accuracy) as int;
final stft = STFT(chunkSize, Window.hanning(chunkSize));
```

Setting the number of divisions (accuracy) for the FFT

6.2 ALGORITHM IMPLEMENTATION

FFT accuracy (continued)

Below is a table which shows our experiments in determining the best accuracy.

Divisions (Accuracy)	Note	Observed Frequency (Hz)	Correct Frequency (Hz)	Accurate?
10^{12}	C4	258.39	261.63	Pass
	C#4	279.93	277.18	Pass
10^{10}	C4	258.39	261.63	Pass
	C#4	258.39	277.18	Fail
10^{14}	C4	258.39	261.63	Pass
	C#4	277.23	277.18	Pass

We observe that at 10^{12} divisions, the ability to distinguish between C4 and C#4 is first present. Thus, even though 10^{14} divisions provide higher accuracy with more correct observed frequencies, we still keep the accuracy at 12 for the best balance between accuracy and speed.

Minimizing Time Complexity

We then collect the results from the FFT analysis and obtain a list of notes-timing pairs which we collect as a list. From which, we then calculate and split this list into further list of bars which where each bar only contains 4 beats worth of notes and rests. From there, we converted this list into our score painters note format to be printed by the score painter.

In these processes, we ensured that our time complexity for all methods **never exceeded O(n).** This meant that every list manipulation method had no more than 1 iterative loop through the list. We also employed the use of dart **Maps** which have constant look up time. This way we ensure that our algorithm is capped at O(n) complexity, preventing extremely long computation times for long audio pieces.

```
var notes = {
  'bar': ['bar'],
  'rest': ['rest'],
  //0th octave
  'C0': [Note.C, Accidental.None, 0],
  'C#0': [Note.C, Accidental.Sharp, 0],
  'D0': [Note.D, Accidental.None, 0],
  'D#0': [Note.D, Accidental.Sharp, 0],
  'E0': [Note.E, Accidental.None, 0],
  'F0': [Note.F, Accidental.None, 0],
  'F#0': [Note.F, Accidental.Sharp, 0],
  'G0': [Note.G, Accidental.None, 0],
  'G#0': [Note.G, Accidental.Sharp, 0],
  'A0': [Note.A, Accidental.None, 0],
  'A#0': [Note.A, Accidental.Sharp, 0],
  'B0': [Note.B, Accidental.None, 0],
```

Employing dart maps to reduce look up time

```
String getNote(double freq) {
  List<double> differenceList = [];
  for (int i = 0; i < _freqList.length; i++) {
    differenceList.add((freq - _freqList[i].keys.first).abs());
  }
  double minVal = differenceList.reduce(min);
  int idx = differenceList.indexOf(minVal);
  return _freqList[idx].values.first;
}
```

List manipulation methods capped at 1 iterative loop

7. USER STORIES

7. USER STORIES

CORE USER STORIES

1. As a professional musician who does music composition, I want to be able to increase the efficiency of music composition.
2. As a beginner with no music background but interested in music composition, I want to be able to pursue my interest in music composition, create nice melodies and even a whole song.

8. DEVELOPMENT TIMELINE

8. DEVELOPMENT TIMELINE (I)

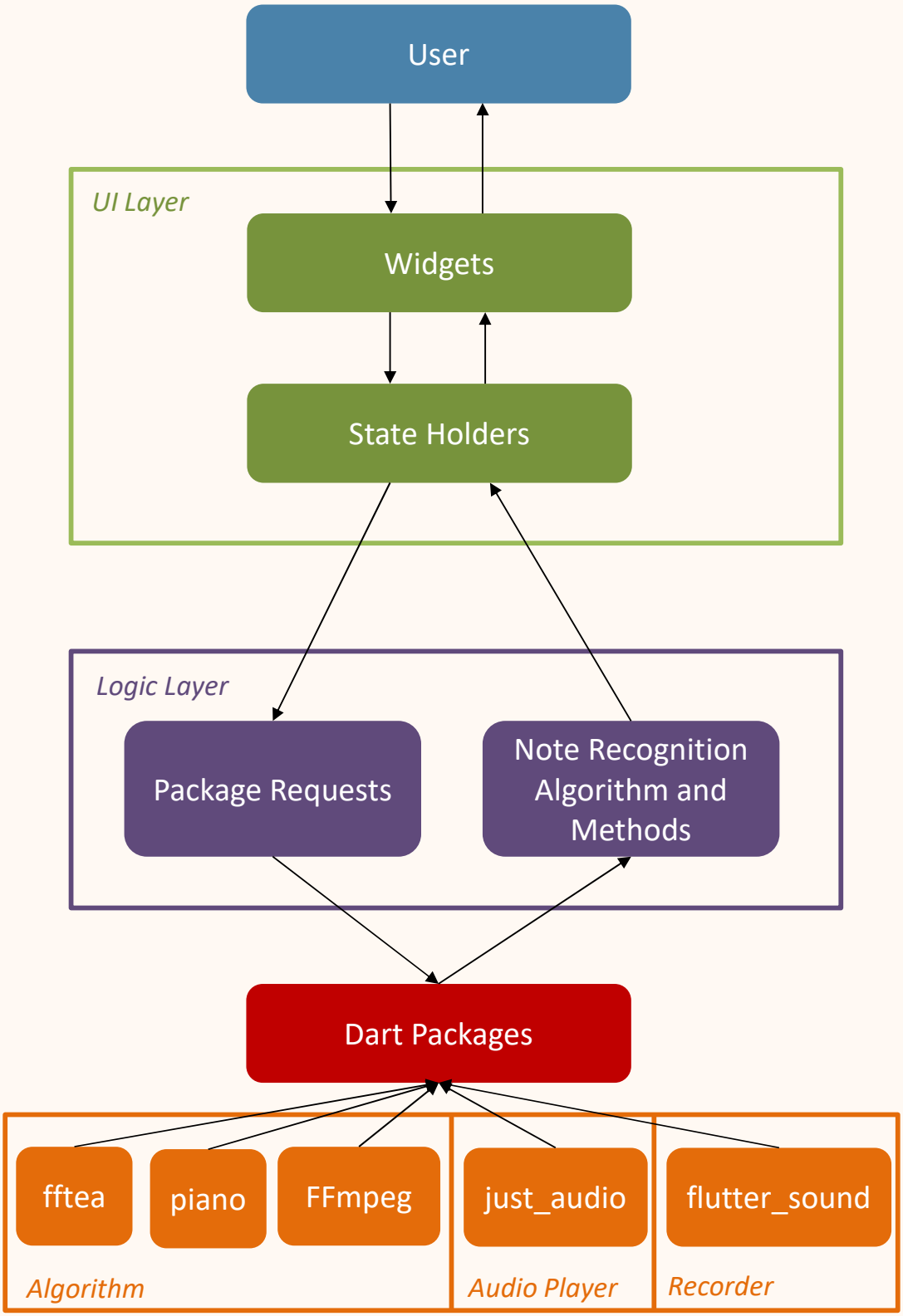
Time Frame	Agenda
Now till 15 June 2022	Create a basic application structure to host the algorithm. Basic GUI created to allow user to create new projects and upload audio files. The application accepts uploaded audio files and allows audio files to be undergo digital signal processing to identify individual notes.
16 June till 27 June 2022	Refine algorithm via multiple tests will be conducted to ensure accuracy of algorithm and repeatability of algorithm (primary back testing phase). Users can replay their melodies via different musical instruments. Update GUI to allow for notes to be transcribed and appear on a score for the user to see the full manuscript of the uploaded audio file. Users can export their desired deliverables. (Manuscripts as pdfs, audio files as mp3 and MIDI files)

8. DEVELOPMENT TIMELINE (II)

Time Frame	Agenda
28 June till 31 July 2022	Run collection obtained notes through a machine learning model to generate suggested harmonies and chords. Users allowed to edit individual notes to their liking. Finalise GUI for combining multiple melodies in order for users to create a full composition.
31 July till 24 August 2022	Run GUI touch ups and improve algorithm accuracy and efficiency for more accurate and efficient transcription. Debugging code where necessary.
Beyond 24 August 2022	Refinement and launch of application.

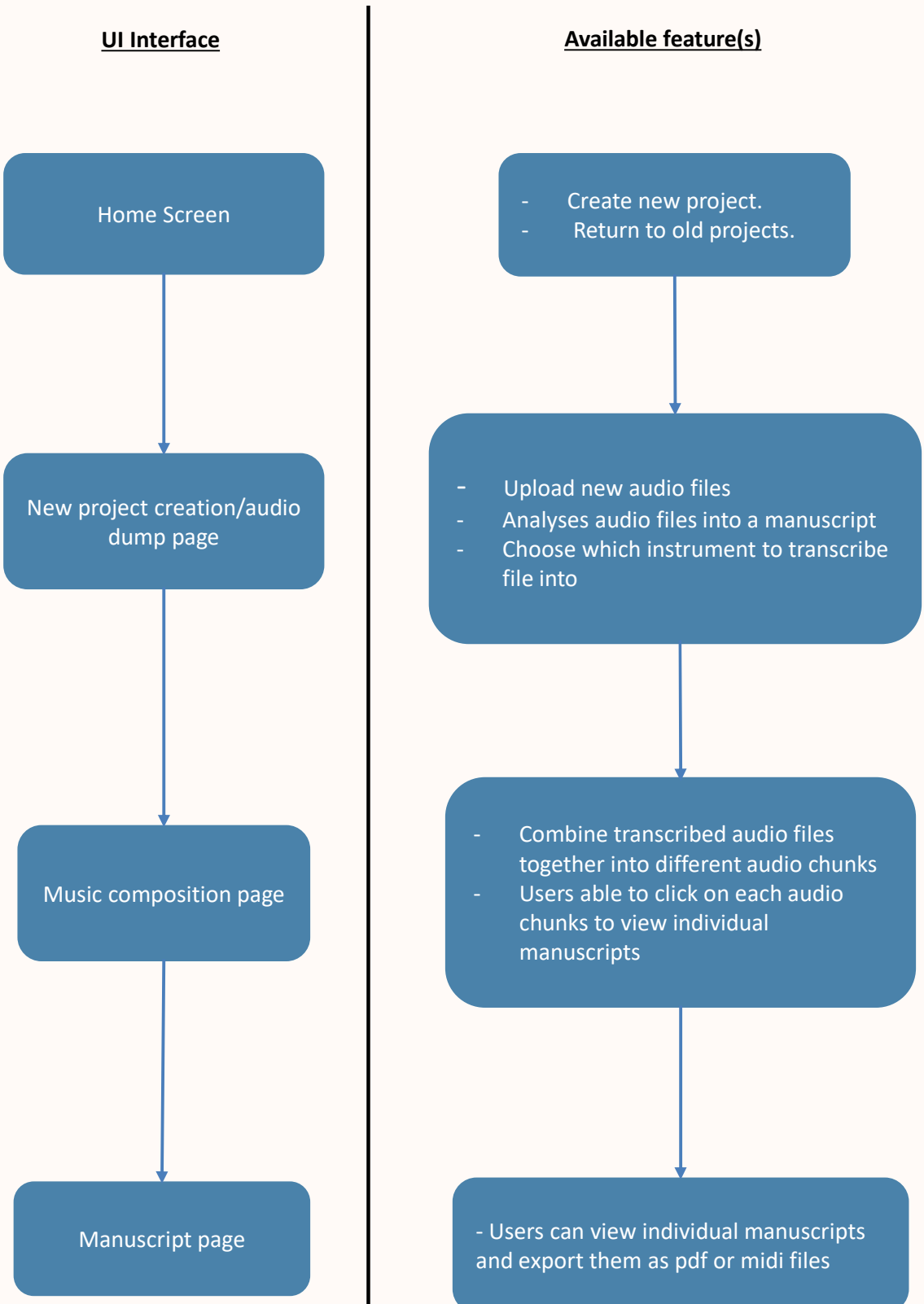
9. DESIGN

9.1 ARCHITECTURE DIAGRAM



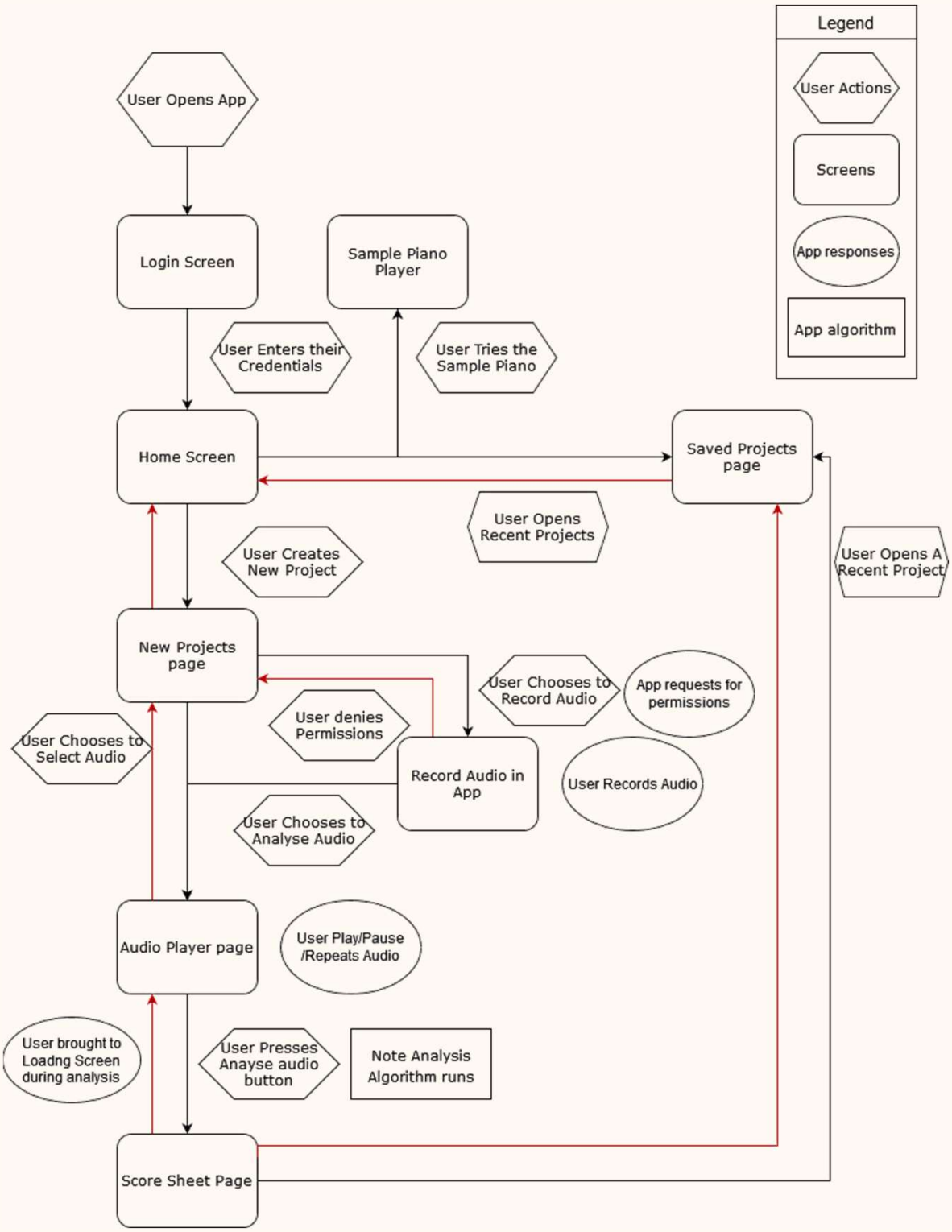
We designed our app with the objective of setting clear boundaries between different parts of the app. By doing so, we establish a clear call and answer relationship between the user's input data (audio files) and any dart packages we used in our implementation. Our architecture allows us to call for services that our implemented packages can provide and relay it back in a format that is understandable to the user, while ensuring that no layer is aware of the code in other layers.

9.2 UI OVERVIEW



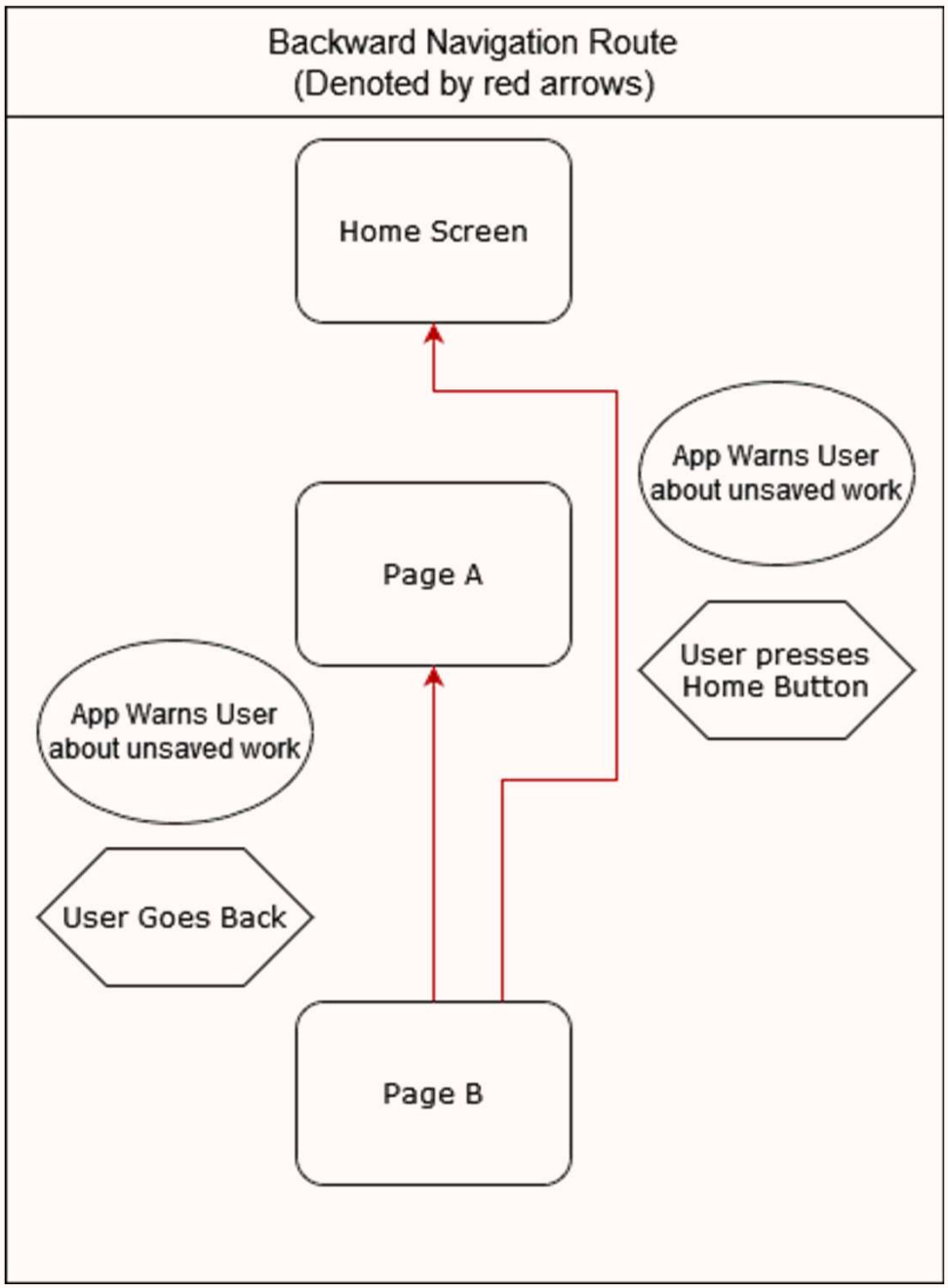
9.3 USER FLOW ACTIVITY DIAGRAM

The diagram below illustrates the user flow activity.

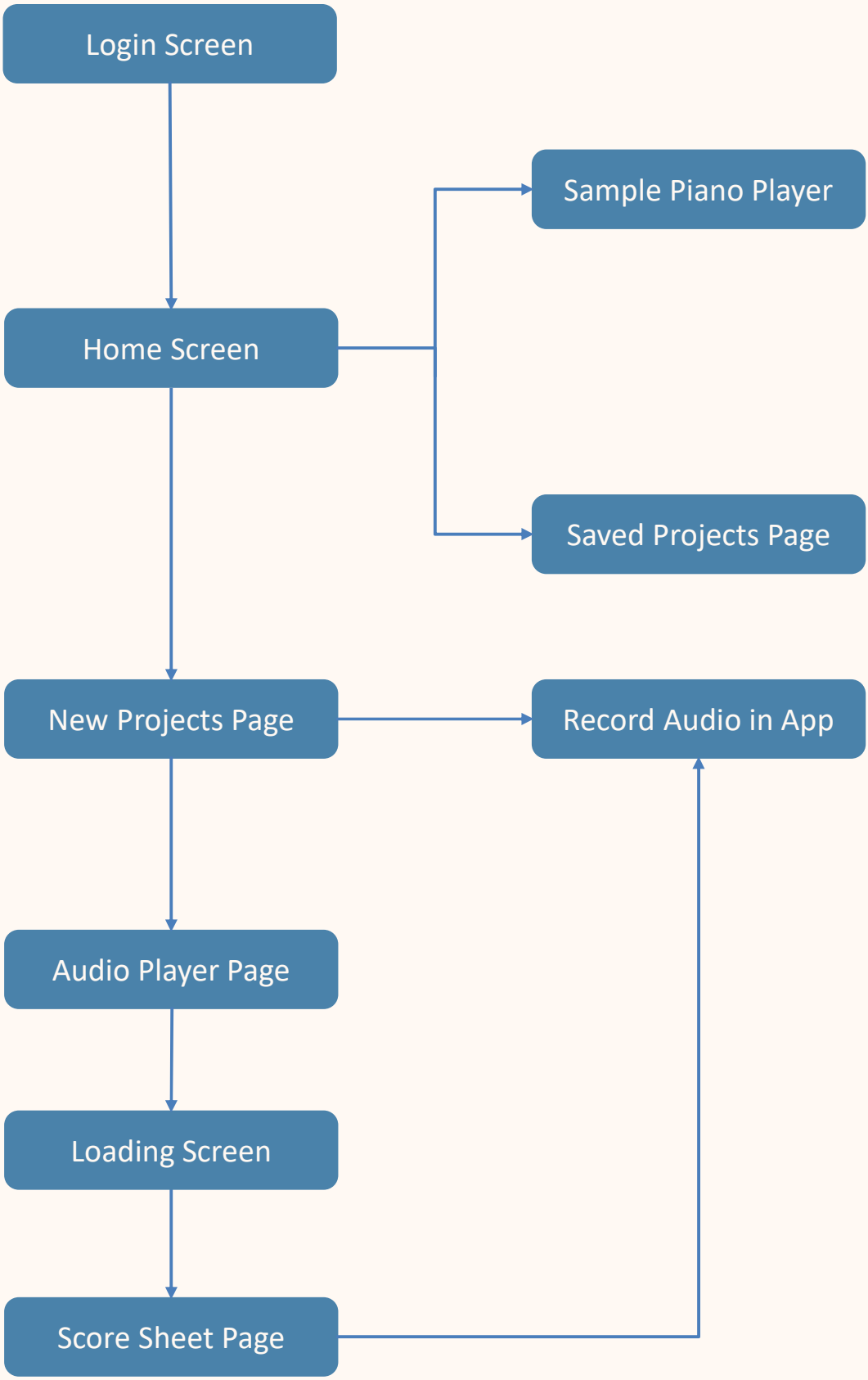


9.3 USER FLOW ACTIVITY DIAGRAM

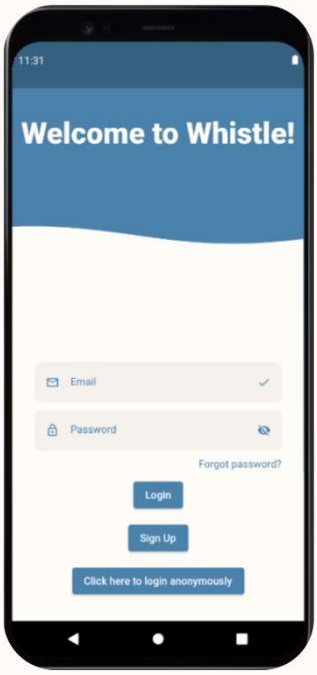
The diagram below illustrates the user flow activity. (Continued)



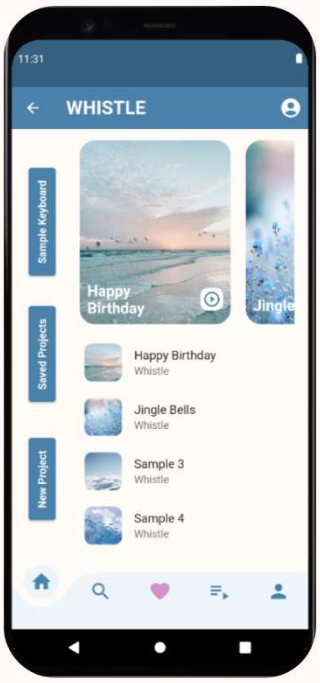
9.4 BRIEF WIREFRAME



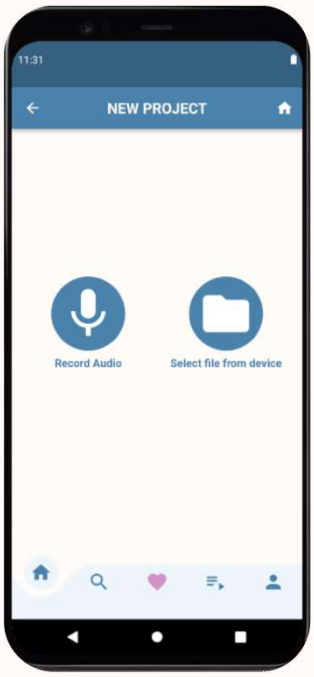
9.5 OVERVIEW OF ALL PAGES



Login Screen

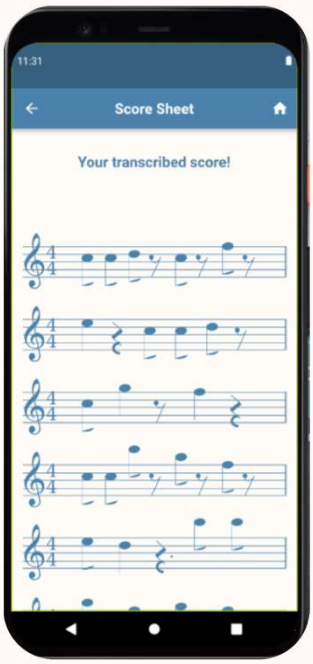


Home Screen



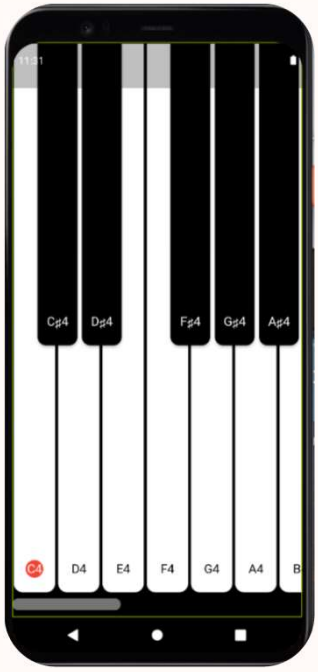
New Projects Screen

Audio Player Page

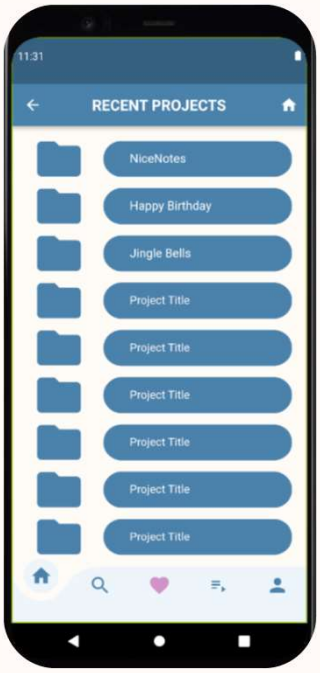


Score Sheet Page

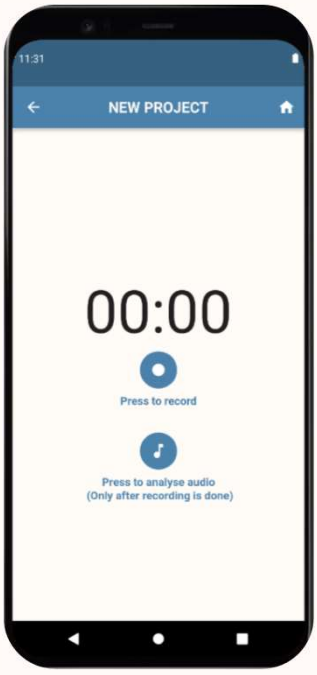
9.5 OVERVIEW OF ALL PAGES



Sample Piano Player



Saved Projects Page



Record Audio in app

9.6 UI STYLE GUIDE

UI Style Guide

Whistle has responsive designs that **adapt to the conventional screen size on iOS and Android mobile device**s as it was built with resizable material design widgets. The size of the widgets are dependent on the screen, and can be demonstrated using the **size_config.dart** code shown below.

whistle > lib > size_config.dart > ...

```dart
import 'package:flutter/material.dart';

class SizeConfig {
  static late MediaQueryData _mediaQueryData;
  static late double screenWidth;
  static late double screenHeight;
  static double? defaultSize;
  static Orientation? orientation;

  void init(BuildContext context) {
    _mediaQueryData = MediaQuery.of(context);
    screenWidth = _mediaQueryData.size.width;
    screenHeight = _mediaQueryData.size.height;
    orientation = _mediaQueryData.orientation;
  }
}

// Get the proportionate height as per screen size
double getProportionateScreenHeight(double inputHeight) {
  double screenHeight = SizeConfig.screenHeight;
  // 812 is the layout height that designer use
  return (inputHeight / 812.0) * screenHeight;
}

// Get the proportionate height as per screen size
double getProportionateScreenWidth(double inputWidth) {
  double screenWidth = SizeConfig.screenWidth;
  // 375 is the layout width that designer use
  return (inputWidth / 375.0) * screenWidth;
}

```

Theme Colours

Our team has also decided to standardise the colours used throughout the application for consistency. We have decided to choose the colour **blue** as the primary colour for the app, since blue represents both the sky and the sea and is associated **with open spaces, intuition, imagination and inspiration**. This is in line with what we hope the users have when they compose their own songs through the Whistle application. We believe that these qualities are essential in a music composition setting.

9.7 LOGO DESIGN

As our team has decided to name our application Whistle, we decided to make our application icon a whistle as well, playing on the synonym of both the noun and the verb forms of the word. We changed the orientation of a whistle and modified it a little, such that it looks like a bass clef, with the 2 dots flattened out to make it look like the whistle is being blown. This is because we wanted to integrate the music element inside our logo, so that users can easily identify our app.

Bass Clef

Whistle Logo

10. IMPLEMENTATION

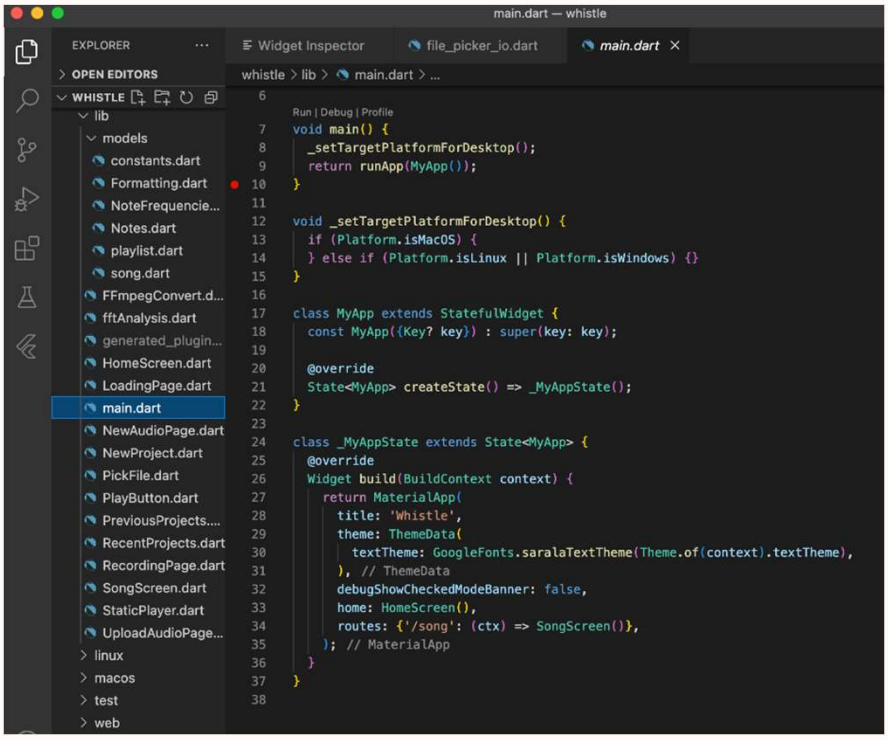
10.1 TECH STACK

LANGUAGES

1. Dart
2. C++

Front End: Flutter

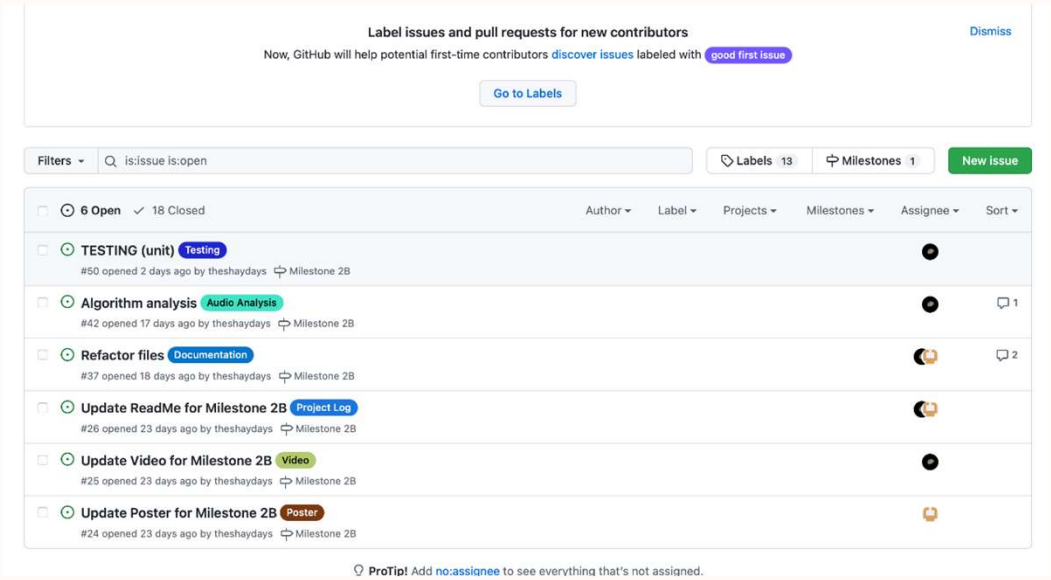
Our team chose to use **Flutter**, with the programming language **Dart**, as our frontend. This will help us to create a cross-platform mobile application with native performance for both iOS and Android users. Its 'hot reload' function has also helped us to create the UI of our app easily and simplify documentation. Another reason why we chose to use flutter is due to its extensive documentation. This can be exemplified by the image below, which is a sample of the main.dart code we have currently.



10.1.1 VERSION CONTROL

Our team has decided to use **Git** on the command line and **GitHub** for version control. Commits are pushed frequently to the whistle repository, and **issues are tracked using GitHub.**

To avoid merge conflicts, we have delegated different tasks to be worked on for each orbitee, and we also use different branches when we work on our individual tasks before merging to the main branch. With GitHub, it has helped us to reset to a previously working version of the code during built data conflicts and retrieve the code we deleted which was required later. The image below shows our current issues to be resolved.



10.1.2 IDES

Our team is currently using **Visual Studio Code** for the development of the Flutter app for its features such as code completion, syntax colouring, error highlighting and refactoring. We are also running our unit tests on the same platform with Flutter plugins installed.

In addition, our team is using Dart DevTools such as

1. **Flutter inspector:** to view the widget ntree and show debug paint for designing UI efficiently.
2. **Flutter performance:** to keep track of frame rendering times and memory usage of our application.

10.1.3 DEPENDENCIES

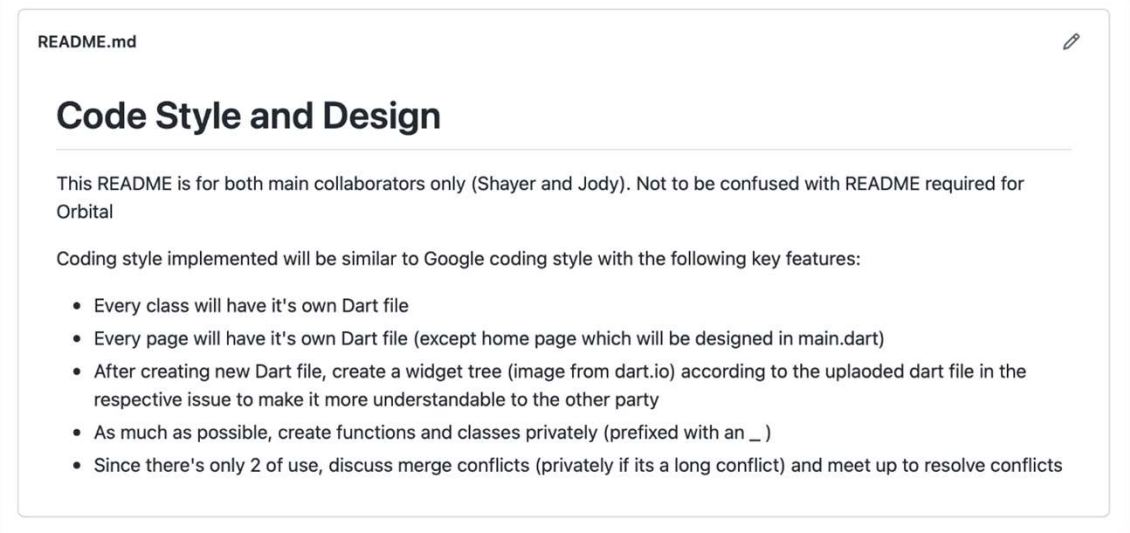
- **google_fonts: ^3.0.1**
- **cupertino_icons: ^0.1.3**
- **just_audio: ^0.9.21**
- **wav: ^1.1.0**
- **fftea: ^1.0.1**
- **file_picker: ^4.6.0**
- **piano: ^1.0.3+1**
- **sheet_music: ^0.3.0**
- **ffmpeg_kit_flutter: ^4.5.1**
- **path_provider: ^2.0.10**
- **flutter_sound: ^9.2.13**
- **permission_handler: ^10.0.0**
- **flutter_spinkit: ^5.1.0**
- **wave_progress_bars: ^0.1.0**
- **curved_navigation_bar: ^1.0.3**
- **freezed_annotation: ^0.14.3**
- **cloud_firestore: ^0.12.9+4**
- **quiver: ^3.1.0**
- **firebase_auth: ^0.15.1+1**
    - For user authentication using Firebase
- **firebase_core: ^0.4.2+2**
    - For using Firebase features

10.2 CODING PRACTICES

The table below summarises some of the coding practices our team adopted while working on the Whistle application.

Naming Practices	Variables and classes are named with nouns and Functions are named with verbs to differentiate between the two and make it closer to a real-world scenario.
Code Structure	Our code is structured logically such that class attributes, instance attributes, constructors, and methods are separated with empty lines. Indentation practices were also followed. Our team placed emphasis on single responsibility principle where each class was designated to a singular job. We also ensured that open-closed principle was thoroughly adhered to ease further modifications.
Code Review	During team programming sessions, we do pair programming which involves reviewing the code and suggesting ways to improve it. This has helped both orbitees to share ideas and knowledge about coding practices and control logic.

The image below is also taken from out github website where we ensure that both of us standardise our code style and design.



README.md

Code Style and Design

This README is for both main collaborators only (Shayer and Jody). Not to be confused with README required for Orbital

Coding style implemented will be similar to Google coding style with the following key features:

- Every class will have it's own Dart file
- Every page will have it's own Dart file (except home page which will be designed in main.dart)
- After creating new Dart file, create a widget tree (image from dart.io) according to the uplaoded dart file in the respective issue to make it more understandable to the other party
- As much as possible, create functions and classes privately (prefixed with an _)
- Since there's only 2 of use, discuss merge conflicts (privately if its a long conflict) and meet up to resolve conflicts

11. HOW TO USE OUR APP?

11. HOW TO USE OUR APP?

In this section, our team will be giving a **detailed walk-through on how users can navigate between pages in our application**, as well as **how they can fully utilise the different features available in our application.**

Login Page

When the user first opens the application in their phone, they will be able to see the login page as shown below.

Current User

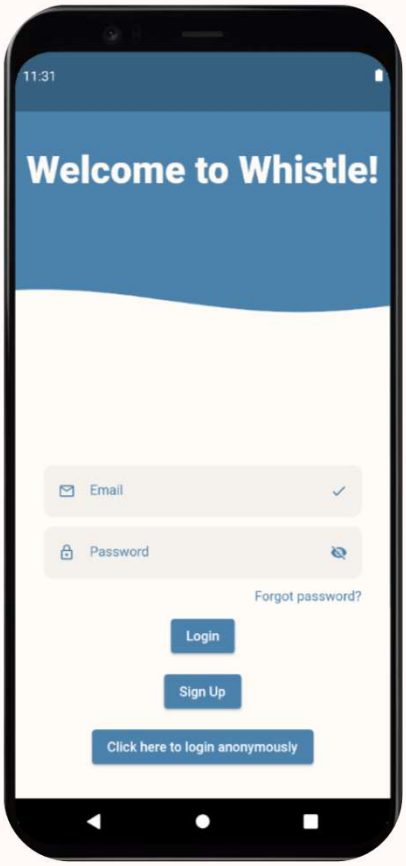
Users will be required to key in their email address as well as their password to log in to their own account.

New User

If the user is new to the application, they will be required to click the sign up button instead to sign up and an email will be sent to the email address they signed up with for verification.

Sign in anonymously

The sign in anonymously button is also available for users who choose not to have an account registered under whistle and wishes to login in anonymously.

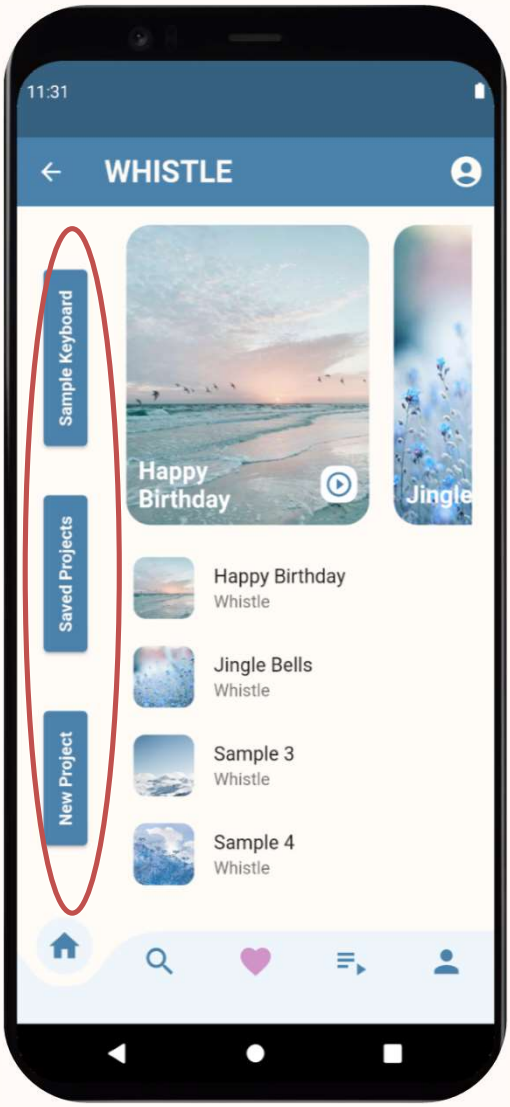


11. HOW TO USE OUR APP?

For demonstration purposes, we have decided to click on the “Click here to login anonymously” button, and the application will lead us to the next page, which is our Home Page.

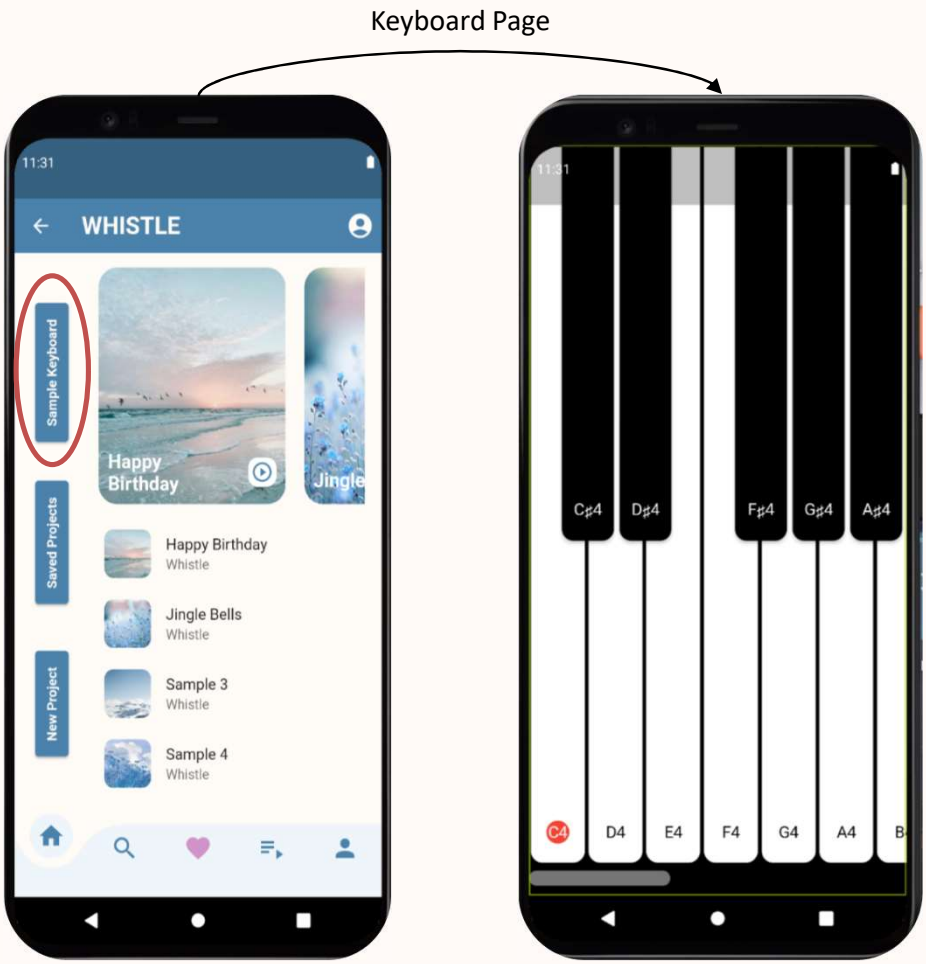
The home page features a dashboard that will allow users to view their projects saved locally on the application. Currently, we have 2 projects saved – Happy Birthday and Jingle Bells saved on the application for unit testing purposes.

Users can also click on the 3 buttons at the left side of the Home page.



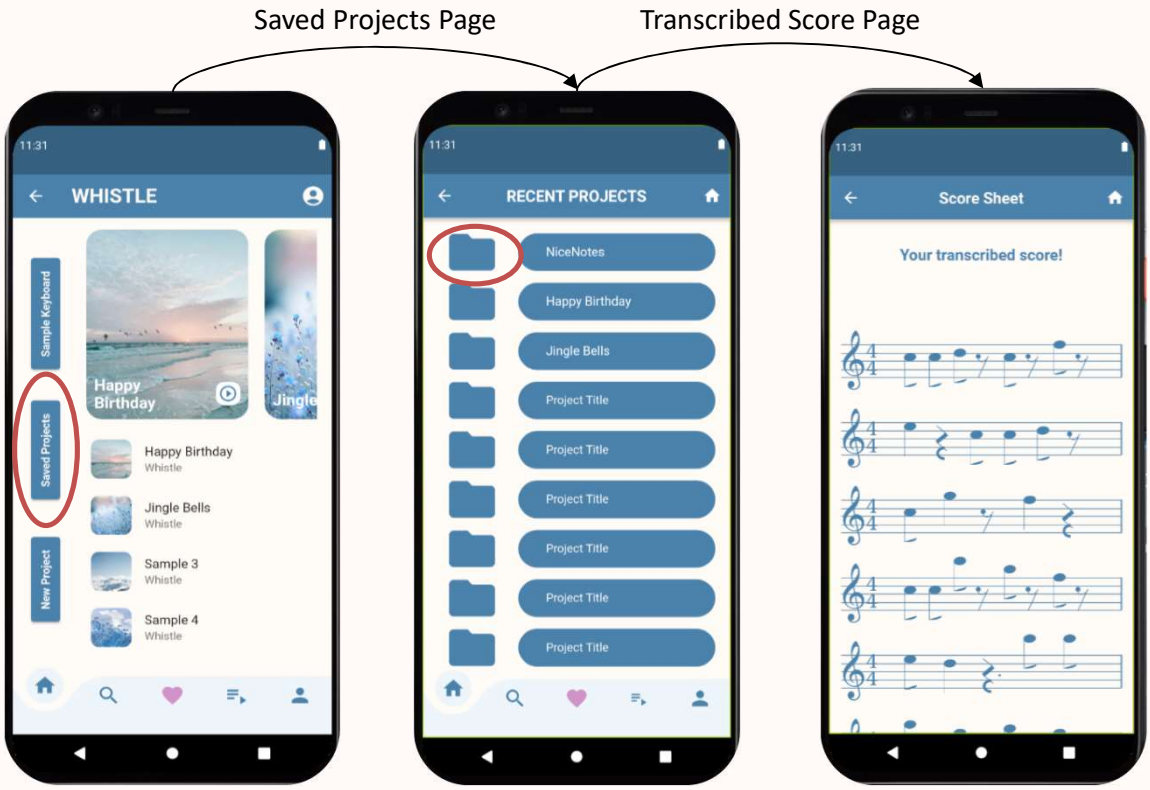
11. HOW TO USE OUR APP?

If the user chooses to click on the **keyboard page**, he/she will be redirected to this page, where he/she will be able to use the keyboard for their own purposes should they wish to compose their music. The keyboard page is included in our application for the convenience of the users who do not have a physical piano with them.



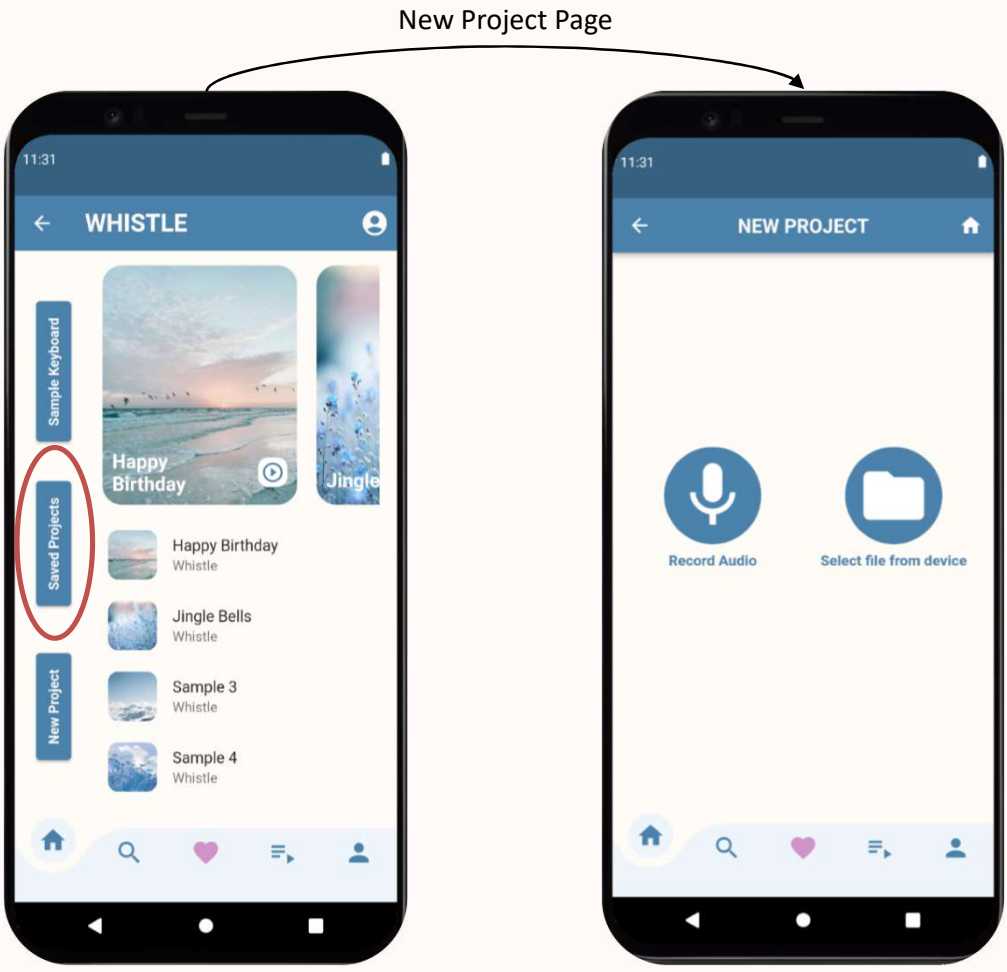
11. HOW TO USE OUR APP?

If the user chooses to click on the **Saved Projects page**, they will be redirected to the Recent Projects page as shown below. They will be able to **view all the projects they have been currently working on**. In addition, should they wish to view the transcribed score, they can click on the “Folder” icon and they will be redirected to the transcribed score page.



11. HOW TO USE OUR APP?

If the user chooses to click on the New Projects page, they will be redirected to the New Projects page as shown below. They will be able to view all the projects they have been currently working on. In addition, should they wish to view the transcribed score, they can click on the “Folder” icon and they will be redirected to the transcribed score page.



11. HOW TO USE OUR APP?

If the user chooses to record the audio he can press the record audio button as shown below.

11. HOW TO USE OUR APP?

When the user is at the audio player page, he/she can choose to press the analyse button, the play button or the help button.

"Help" message that pops up

12. QUALITY ASSURANCE

12. QUALITY ASSURANCE

To ensure the quality of our Whistle application, our team has decided to conduct software testing using multiple approaches. In order to attain extensive and thorough testing regiments, we have decided to conduct **four** forms of testing: **Unit, Widget, Developer and Algorithm**.

To ensure we covered every aspect of the application, we devised a testing system **devised around our user activity diagram**. We dissected **every possible action** and listed out possible tests to be conducted to ensure app quality. From each action, we then conduct multi-layered tests (of Unit, Widget and User tests) to locate any and all possible flaws. The details of such will be in the following pages.

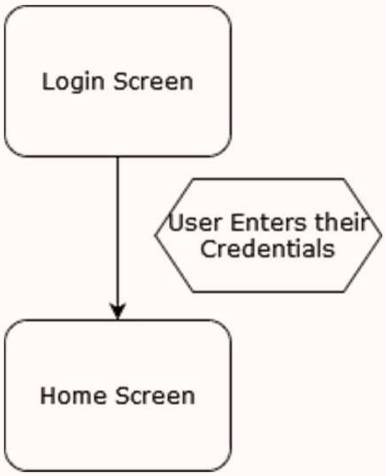
Our algorithm testing procedure will be explained later.

Continuous Integration (CI)

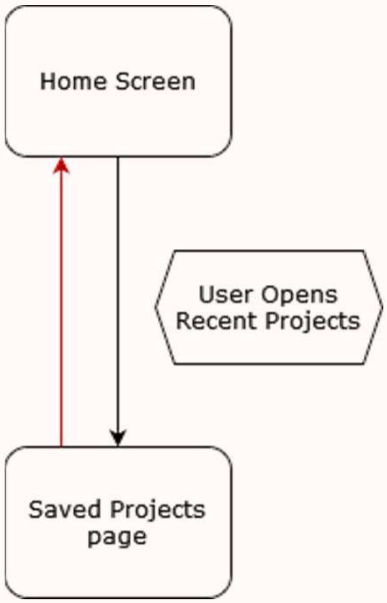
Our team uses **continuous integration services** for automated testing and building when we make any updates or amendments to our code. This service provides us with feedback and allows us to ensure that the new code works as expected with no errors. Our team also uses **GitHub actions** for running our continuous integration workflows, as they are user friendly, since they operate on the cloud and are tightly integrated with GitHub. We have also made our GitHub repository **open-source to have access to unlimited minutes for running our workflows**.

12.1 MULTI-LAYERED TESTING

We tested each user action and devised the following tests as follows.



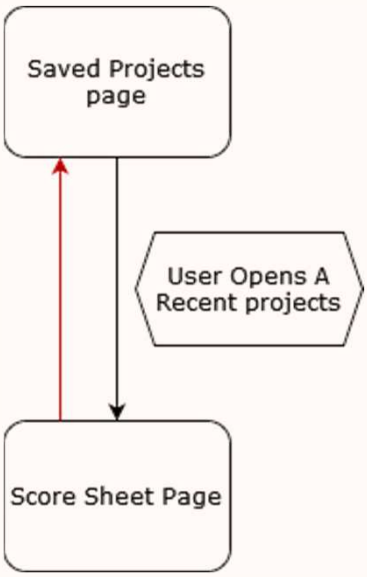
Test Type	Test	Pass/Fail
Unit	-NA-	-NA-
Widget	- Check if “Login" Button leads to Home Screen	Pass
Developer	- Check if “Login" Button leads to Home Screen	Pass
	- Check if users can type in any log in credentials	Pass



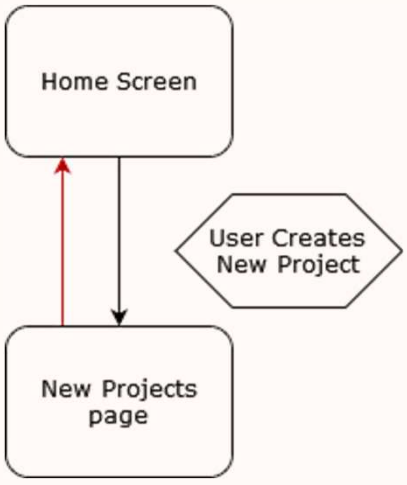
Test Type	Test	Pass/Fail
Unit	-NA-	-NA-
Widget	- Check if tapping on the "Recent Projects" button leads to Saved Projects page	Pass
Developer	- Check if tapping on the "Recent Projects" button leads to Saved Projects page	Pass

** Full User Activity Diagram can be found on previous pages

12.1 MULTI-LAYERED TESTING



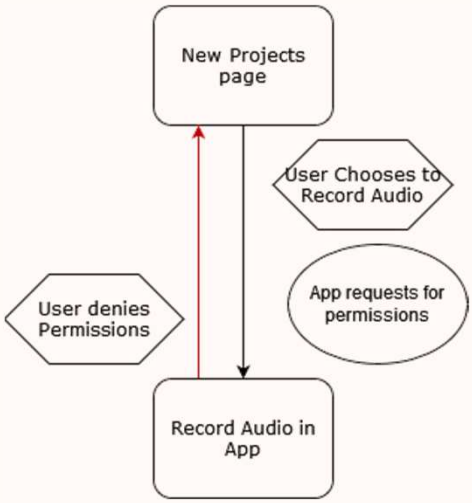
Test Type	Test	Pass/Fail
Unit	-NA-	-NA-
Widget	- Check if tapping on the first folder leads to Score Sheet page	Pass
Developer	- Check if tapping on the first folder leads to Score Sheet page	Pass
	- Check that tapping other folder icons do not lead to anywhere	Pass



Test Type	Test	Pass/Fail
Unit	-NA-	-NA-
Widget	- Check if "New Project" Button leads to New Projects page	Pass
Developer	- Check if "New Project" Button leads to New Projects page	Pass

** Full User Activity Diagram can be found on previous pages

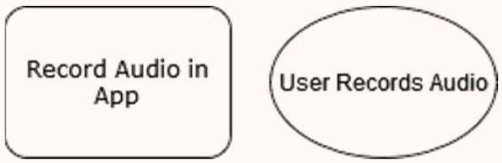
12.1 MULTI-LAYERED TESTING



Test Type	Test	Pass/Fail
Unit	-NA-	-NA-
Widget	- Check if "Record Audio" Button leads to Recording page	Pass
	- If App permissions denied, check if user is redirected back to previous page	Pass
Developer	- Check if "Record Audio" Button leads to Recording page	Pass
	- If App permissions denied, check if user is redirected back to previous page	Pass
	- If App permissions denied, check if user is redirected back to previous page	Pass

** Full User Activity Diagram can be found on previous pages

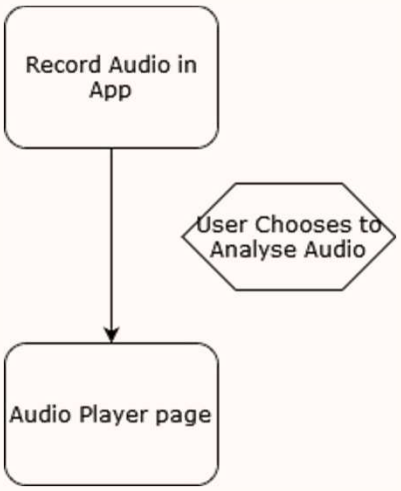
12.1 MULTI-LAYERED TESTING



Test Type	Test	Pass/Fail
Unit	- Playing Recorder	Unable to test
	- Stopping Recorder	Unable to test
	- Pausing Recorder	Unable to test
Widget	- Check for icon changes when user is recording	Pass
	- Check that alert dialog appears if no recording is found	Pass
Developer	- Check for icon changes when user is recording	Pass
	- Check that alert dialog appears if no recording is found	Pass
	- Check that user microphone is in use (green microphone icon on phone should show on newer android versions)	Pass
	- Check for long duration recordings	Pass

* Unit tests cannot be implemented as packages cannot be implemented on mac and windows OS
** Full User Activity Diagram can be found on previous pages

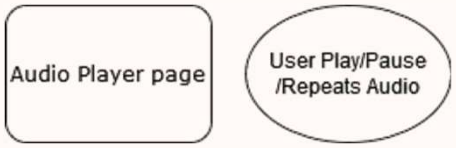
12.1 MULTI-LAYERED TESTING



Test Type	Test	Pass/Fail
Unit	-NA-	-NA-
Widget	- Check if "Analyse Audio" Button leads to Audio Player Page if recording exists	Pass
	- Check if "Analyse Audio" produces popup that prompts user to record audio if recording does not exist	Pass
Developer	- Check if "Analyse Audio" Button leads to Audio Player Page if recording exists	Pass
	- Check if "Analyse Audio" produces popup that prompts user to record audio if recording does not exist	Pass

** Full User Activity Diagram can be found on previous pages

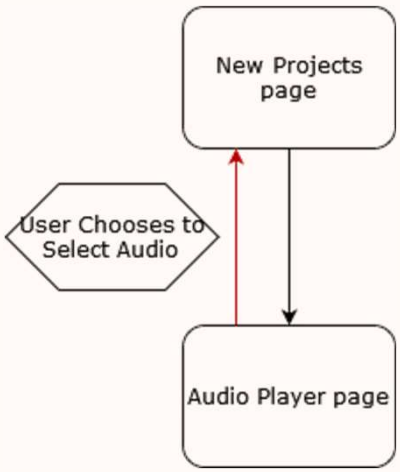
12.1 MULTI-LAYERED TESTING



Test Type	Test	Pass/Fail
Unit	- Playing Audio Player	Unable to test
	- Stopping Audio Player	Unable to test
	- Pausing Audio Player	Unable to test
Widget	- Check for icon changes when user is playing audio	Pass
	- Check that music bar runs as audio is playing	Pass
	- Check for circular progress bar when music player is buffering	Pass
Developer	- Check that music bar runs as audio is playing	Pass
	- Check for circular progress bar when music player is buffering	Pass
	- Check for dynamic updating of current song time	Pass
	- Check that audio is played natively from device	Pass
	- Check Users can drag player bar	Pass
	- Check Users can drag music player bar	Pass

* Unit tests cannot be implemented as packages cannot be implemented on mac and windows OS
** Full User Activity Diagram can be found on previous pages

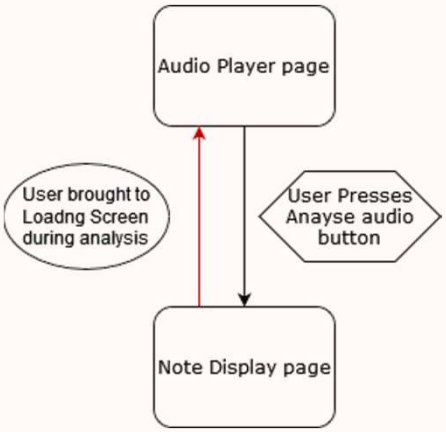
12.1 MULTI-LAYERED TESTING



Test Type	Test	Pass/Fail
Unit	- Getting Duration of audio file	Unable to test
	- Getting Name of audio file	Unable to test
	- Getting Path of audio file	Unable to test
Widget	- Check if "Select File" Button leads to device folders which leads to Audio player page	Pass
	- If App permissions denied, check if user is prevented access to next page and dialog appears	Pass
Developer	- Check if "Select File" Button leads to device folders which leads to Audio player page	Pass
	- Check if users can find their audio files	Pass
	- Check if only files with audio formats are allowed to be chosen	Pass

* Unit tests cannot be implemented as packages cannot be implemented on mac and windows OS
** Full User Activity Diagram can be found on previous pages

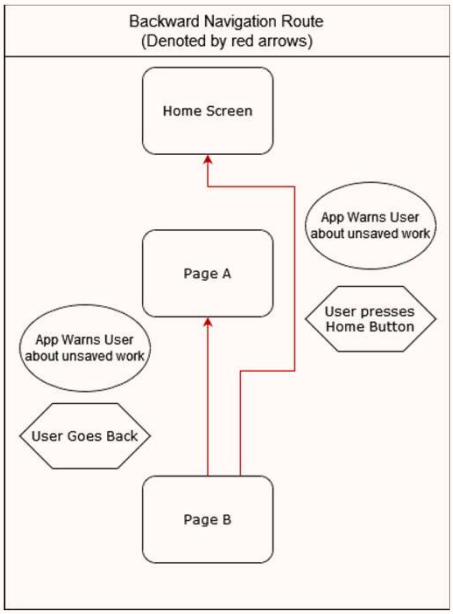
12.1 MULTI-LAYERED TESTING



Test Type	Test	Pass/Fail
Unit	-NA-	-NA-
Widget	- Check if "Analyse Audio" Button Loading Screen which leads to Note Display when calculation is done	Pass
Developer	- Check if "Analyse Audio" Button Loading Screen which leads to Note Display when calculation is done	Pass
	- Check that all file formats are appropriately converted	Fail* (m4a not accepted)
	- Check that Loading page icons are generated in a random order	Pass
	- Check whether all audio file names are accepted	Pass
	- Check that users cannot leave the page while while loading	Pass with issues (users will not have an issue but the error is thrown on dev side)
	- Check whether only BPMs of 40 – 220 are allowed	Pass

* Not intending to fix as m4a is not an audio format
** Full User Activity Diagram can be found on previous pages

12.1 MULTI-LAYERED TESTING

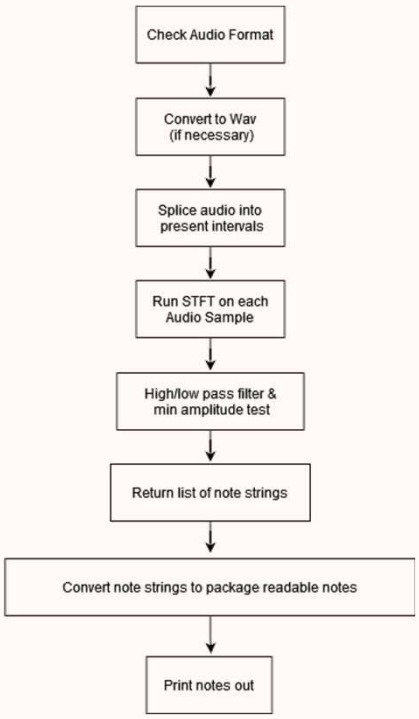


Test Type	Test	Pass/Fail
Unit	-NA-	-NA-
Widget	- Check if going back gives users a warning that work will be unsaved	Pass
	- Check if pressing home icon leads user to Home Screen with dialog option	Pass
Developer	- Check if going back gives users a warning that work will be unsaved	Pass
	- Check if pressing home icon leads user to Home Screen with dialog option	Pass

** Full User Activity Diagram can be found on previous pages

12.1 MULTI-LAYERED TESTING

NOTE:
Not to be confused with algorithm testing, this section is to merely check whether the app returns notes in the correct file format and not whether the notes are accurate.



Test Type	Test	Pass/Fail
Unit	- Audio Conversion	Unable to test
	- Audio Splicing	Unable to test
	- FFT Analysis	Unable to test
	- Amplitude and High/Low Pass filter	Unable to test
	-Converting list of frequency to list of notes	Pass
	- Converting List of notes to package readable notes	Pass
Widget	-NA-	-NA-
Developer	-NA-	-NA-

* Unit tests cannot be implemented as packages cannot be implemented on mac and windows OS
** Full User Activity Diagram can be found on previous pages

10.2 ALGORITHM TESTING

Algorithm testing is mainly catered towards testing whether the algorithm returns an **accurate transcription** of what notes are being played in the audio. However, this section will be **focused on over the course of the next month** and will be the number 1 priority for the next milestone. As such, users will find that their notes **are very poorly transcribed** as of now. However, the following table will show the **proposed testing structure** we wish to implement for algorithm testing.

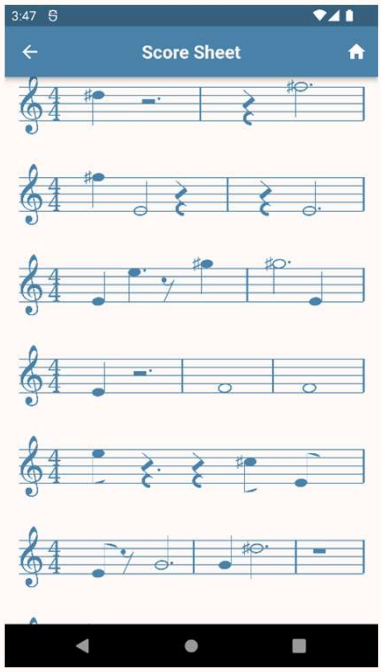
Test Type	Audio Source	Accuracy
Single Notes (4^{th} to 6^{th} Octave)	Raw Sound Waves	
	Clean Recorded Piano	
	Noisy Recorded Piano	
	Clean Recorded Whistle	
	Noisy Recorded Whistle	
5 Second Sample	Raw Sound Waves	
	Clean Recorded Piano	
	Noisy Recorded Piano	
	Clean Recorded Whistle	
	Noisy Recorded Whistle	
1 min Sample	Raw Sound Waves	
	Clean Recorded Piano	
	Noisy Recorded Piano	
	Clean Recorded Whistle	
	Noisy Recorded Whistle	

12.2 ALGORITHM TESTING

Algorithm testing is mainly catered towards testing whether the algorithm returns an **accurate transcription** of what notes are being played in the audio. In order to thoroughly test the algorithm, we will test the 3 main components of our algorithm: **Segmentation, Frequency Extraction and Mapping**.

Mapping

We first test Mapping component to determine whether the algorithm is correctly splitting up notes and painting each component correctly. To do so, we created a testing list of notes to print and change the BPM to determine whether the correct notes and rests are being printed. We also checked whether all notes of all types are being printed correctly



Note: We do acknowledge that there may be some painting errors when choosing unconventional BPMs

We also do acknowledge that there are **missing ledger lines** for notes above G5, making the notes hard to read

BPM	Splice length	Accurate?
120	0.25	Pass
60	0.5	Pass
90	0.3333	Pass
100	0.3	Pass
75	0.4	Pass

Note Type	Image	Accurate?
Crotchet Note		Pass
Crotchet Rest		Pass
Quaver Note		Pass
Quaver Rest		Pass
Minim Note		Pass
Minim Rest		Pass
Whole Note		Pass
Whole Rest		Pass
Dotted Note		Pass
Dotted Rest		Pass

12.2 ALGORITHM TESTING

Frequency Mapping

Before testing on audio pieces, we first have to look towards ensuring that every frequency can be picked up. As such, we ran the algorithm through piano notes from C4 to C6.

Note	Observed Frequency (Hz)	Correct Frequency (Hz)	Accurate?
C4	258.39	261.63	Pass
C#4	279.93	277.18	Pass
D4	301.46	293.66	Pass
D#4	307.25	311.13	Pass
E4	333.76	329.63	Pass
F4	355.29	349.23	Pass
F#4	366.06	369.99	Pass
G4	387.59	392.00	Pass
G#4	419.90	415.30	Pass
A4	441.43	440.00	Pass
A#4	462.96	466.16	Pass
B4	495.26	493.88	Pass
C5	527.56	523.25	Pass
C#5	559.86	554.37	Pass
D5	592.16	587.33	Pass
D#5	624.46	622.25	Pass
E5	656.76	659.25	Pass
F5	699.82	698.46	Pass
F#5	742.90	739.99	Pass
G5	785.96	783.99	Pass
G#5	839.79	830.61	Pass
A5	882.86	880.00	Pass
A#5	936.69	923.33	Pass
B5	990.52	987.77	Pass
C6	1055.12	1046.50	Pass

12.2 ALGORITHM TESTING

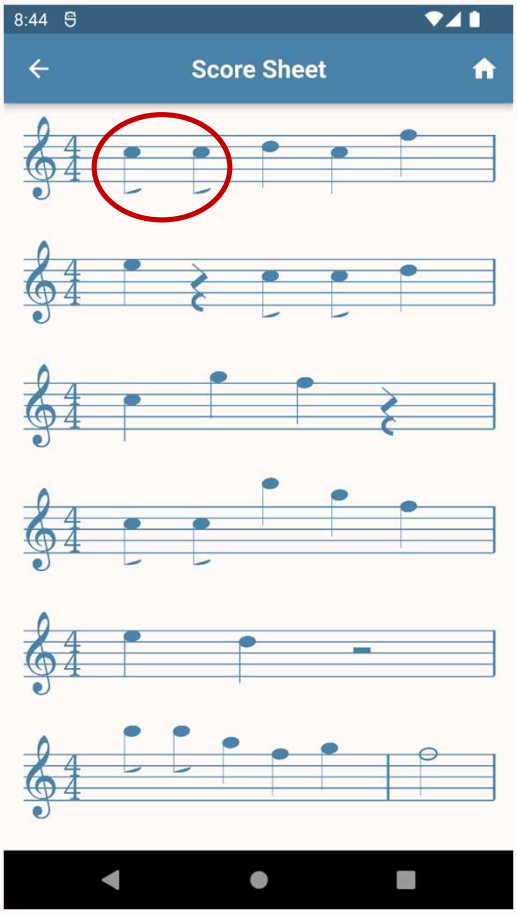
Audio Testing

Now that the algorithm has shown strength in accurately depicting the correct note for each audio slice, we test the algorithm on actual known melodies. In order to accomplish this, we test the melody in 2 different ways: through a xylophone playing the melody and an actual whistle of the melody. Xylophones were chosen as they usually play notes ranging from C5 to C6, the same range as a human whistle.

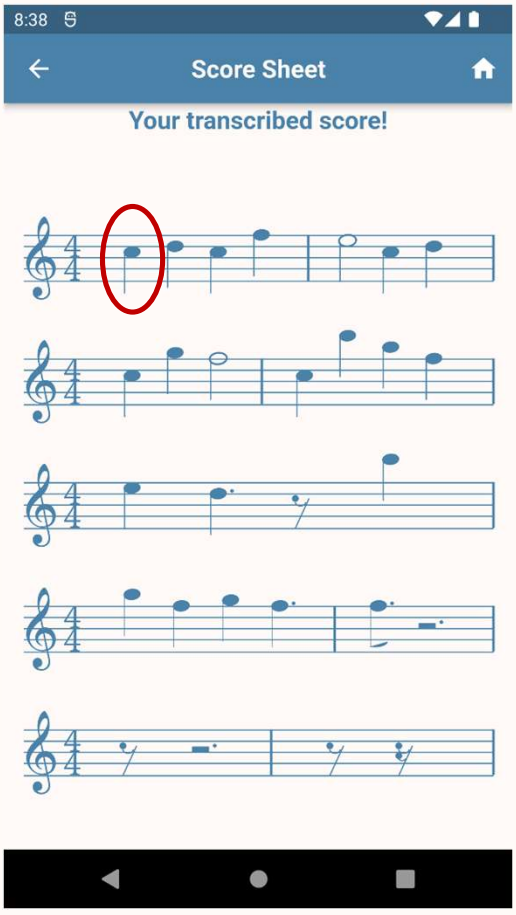
Happy Birthday!

For the first run, we tested the algorithm with Happy Birthday, a famous song everyone sings! Below are the results.

Correct Score



Xylophone Score (50 BPM)



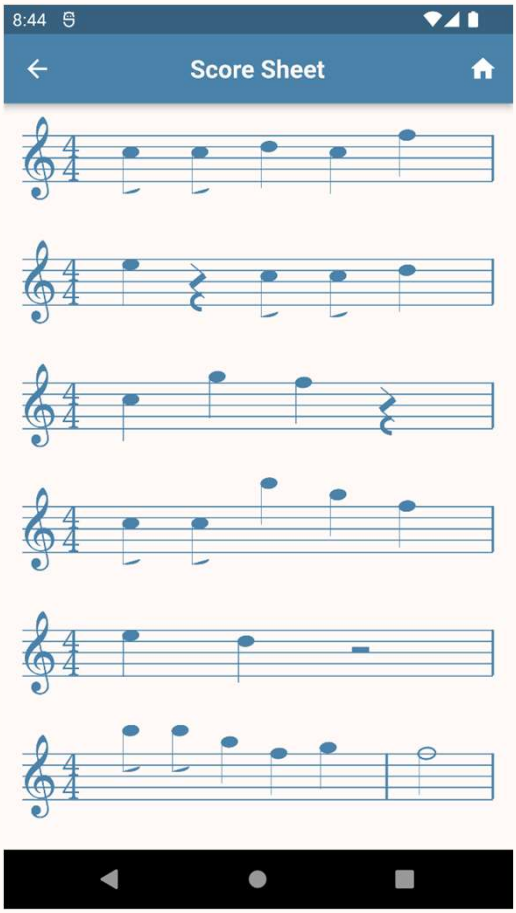
Here we can see that the xylophone sample shows a great degree of accuracy when running through the analysis. Other than missing or adding a few extra rests, the pitches of the notes are generally correct and are very similar to what is to be expected (grab a piano and try!). However, the algorithm struggles to detect the difference between 1 note and multiple notes (as shown by the circled notes), and as such, merged the 2 quaver beats into 1 crotchet. A discussion on this issue will be done later.

12.2 ALGORITHM TESTING

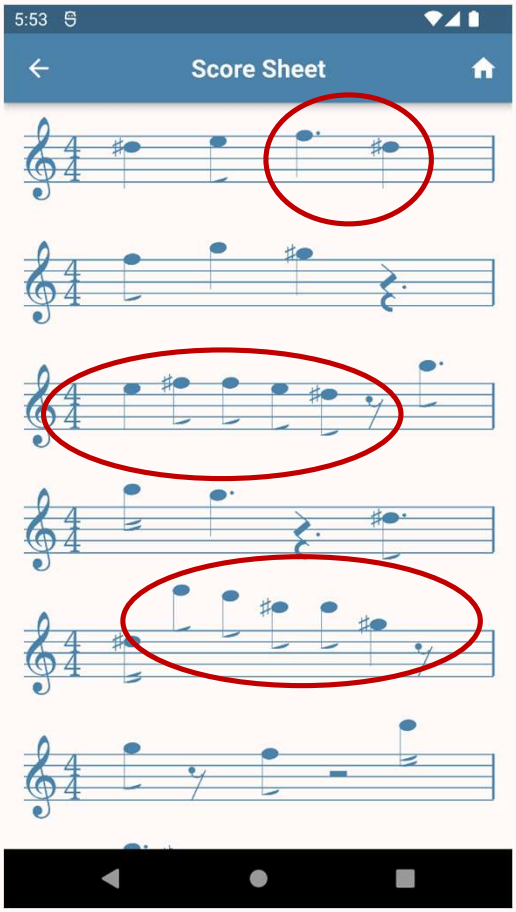
Happy Birthday!

Running the same song with a noisy whistle sample, however, greatly decreases the accuracy of the algorithm. This is seen below.

Correct Score



Whistle Score (50 BPM)



Here we see that the whistle score performs much worse than expected. However, there are still many positives to take away. While the notes may not be perfectly accurate, **the general intonation and melody flow is relatively correct.**

The large errors in note accuracy is also attributed to the fact that we could not get a professional whistler to whistle the tune properly. As such our whistle sample was severely out of tune and out of beat (we apologise for not being able to whistle well).

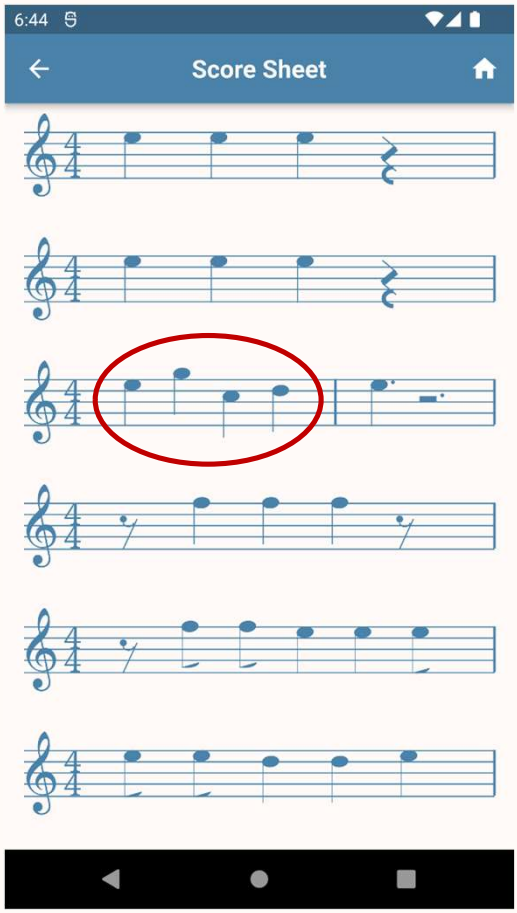
Nonetheless, when replaying this score on the piano, we note that the circles parts in red are largely accurate to what we hear in the whistle sample (once again the sample is out of tune, explaining the difference in scores with the correct score).

12.2 ALGORITHM TESTING

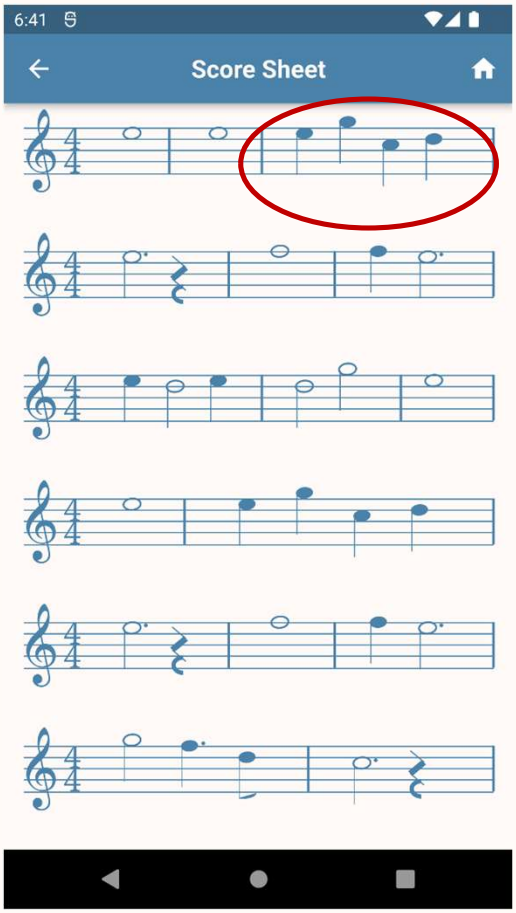
Jingle Bells

We then try another song, Jingle Bells to determine the extent of the inability to split note for the algorithm.

Correct Score



Xylophone Score (80 BPM)



Note: The full scores are not shown here. View the full score in the app itself.

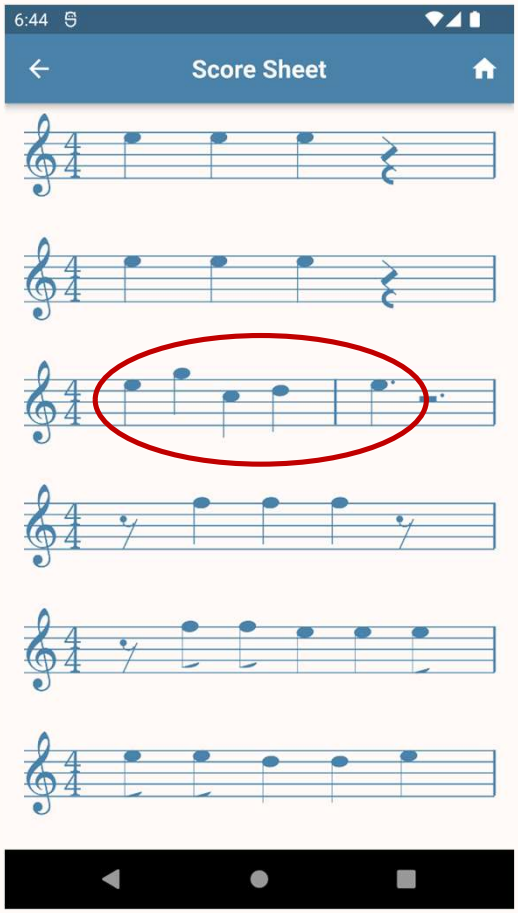
Like Happy Birthday, the algorithms inability to split notes are shown here. However, beyond that and a few rests, the notes remain very accurate. This is clearly seen by the circled portions.

12.2 ALGORITHM TESTING

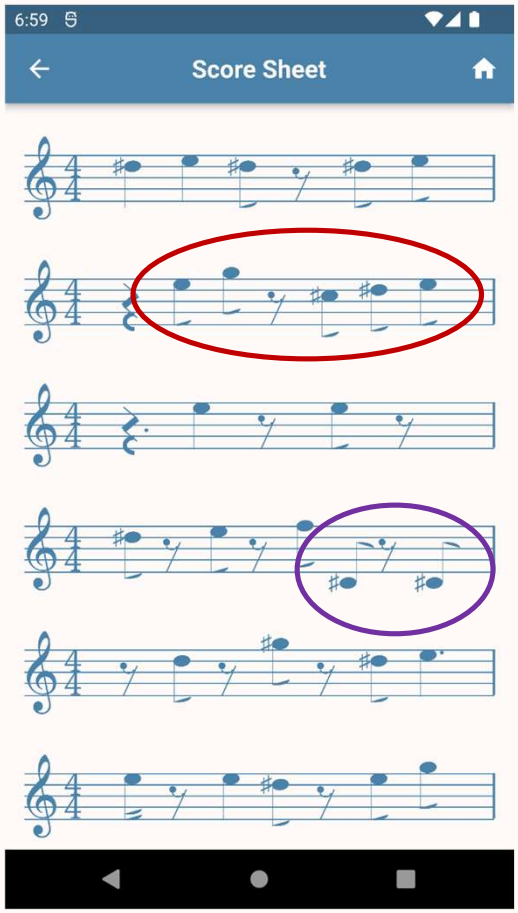
Jingle Bells

Similarly, we test with a poorly whistled out rendition of jingle bells.

Correct Score



Whistle Score (70 BPM)



Note: The full scores are not shown here. View the full score in the app itself.

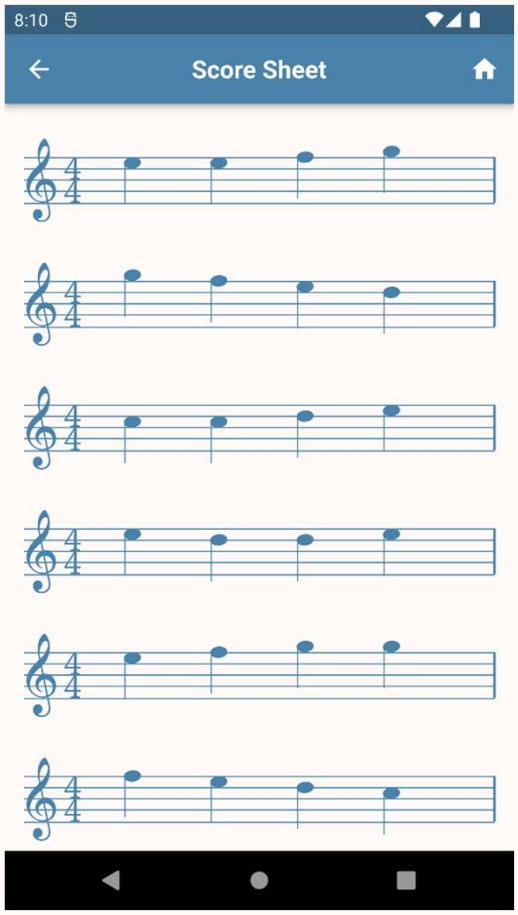
Here we see that the algorithm performs much better in trying to capture the true melody of jingle bells. However, it does also show out of place notes (circled in purple) where the algorithm may be catching background noise. We have made efforts to reduce background noise but decided against it (explained later).

12.2 ALGORITHM TESTING

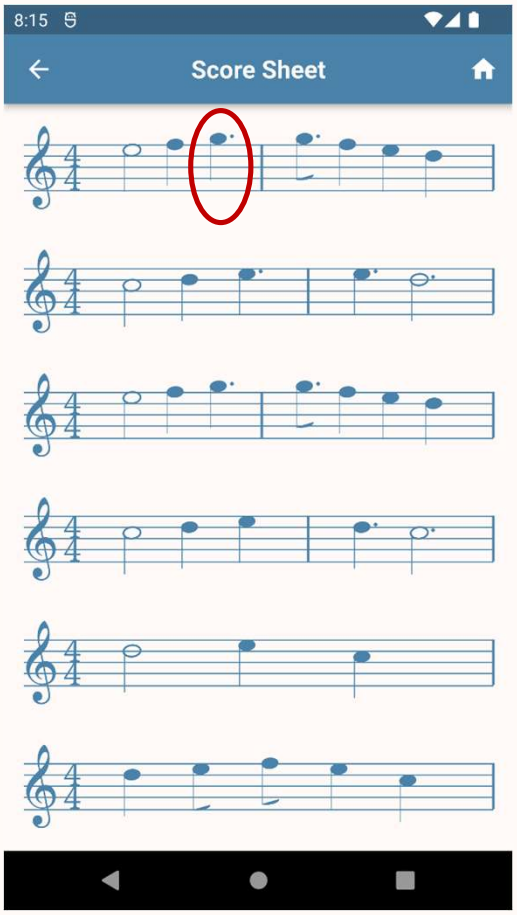
Ode to Joy

To test irregular BPMs, we continued testing on Ode to Joy played at 70 BPM.

Correct Score



Xylophone Score (70 BPM)



Note: The full scores are not shown here. View the full score in the app itself.

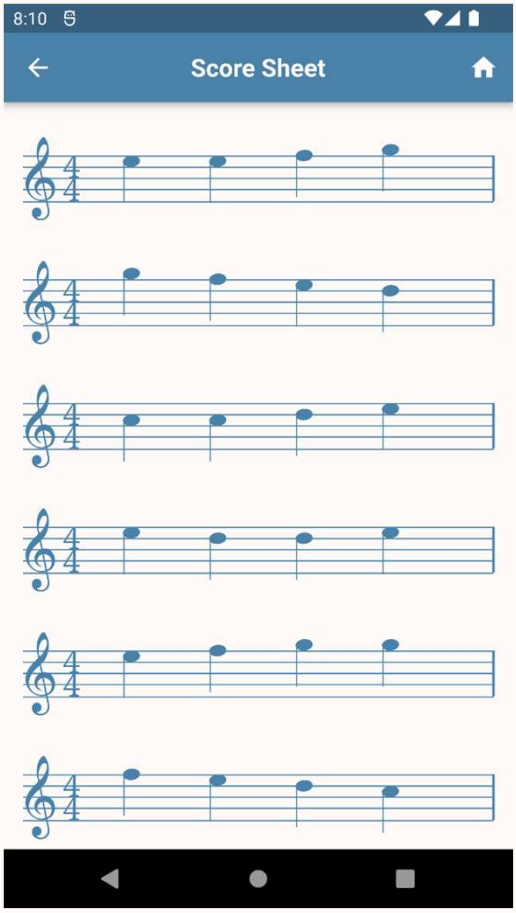
The notes are generally still accurate; however, a new problem arises. We notice that the circled note is written as a dotted crotchet instead of a normal crotchet. We looked through and recognized that the error stems from a rounding error where 2/3 != 0.6666. This will be fixed in future updates.

12.2 ALGORITHM TESTING

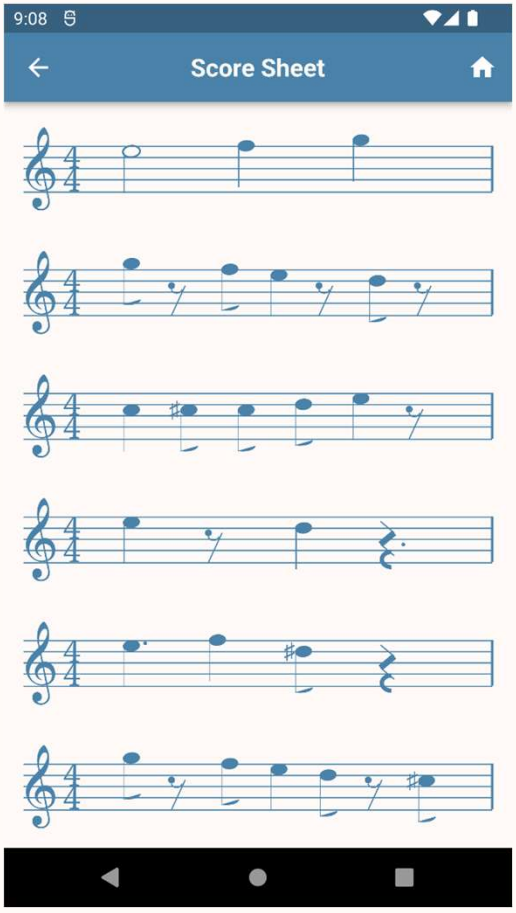
Ode to Joy

We then tested the algorithm with a whistle rendition of the song

Correct Score



Whistle Score (80 BPM)



Note: The full scores are not shown here. View the full score in the app itself.

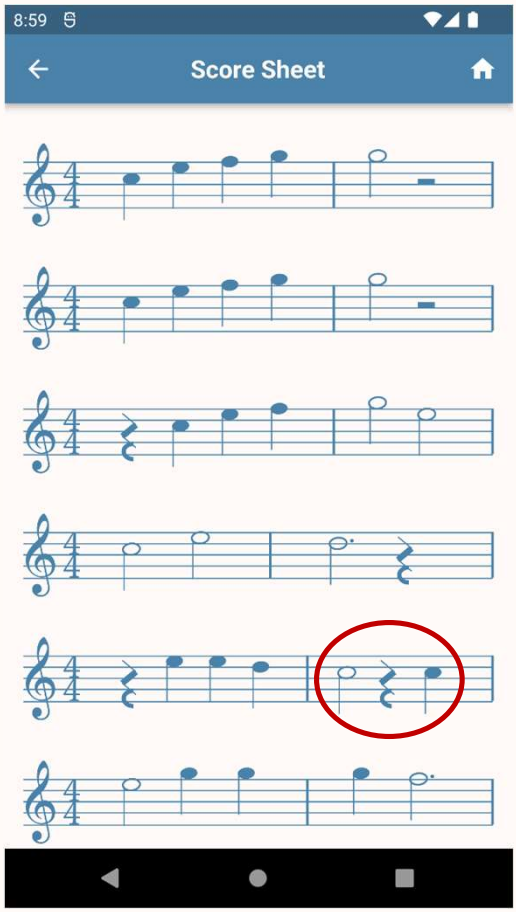
Impressively, we see strong agreement between the actual score and the whistle sample. However, we notice some excessive rests, which could be since whistles decay much faster than xylophones, and as such would render a rest note more often than a xylophone would.

12.2 ALGORITHM TESTING

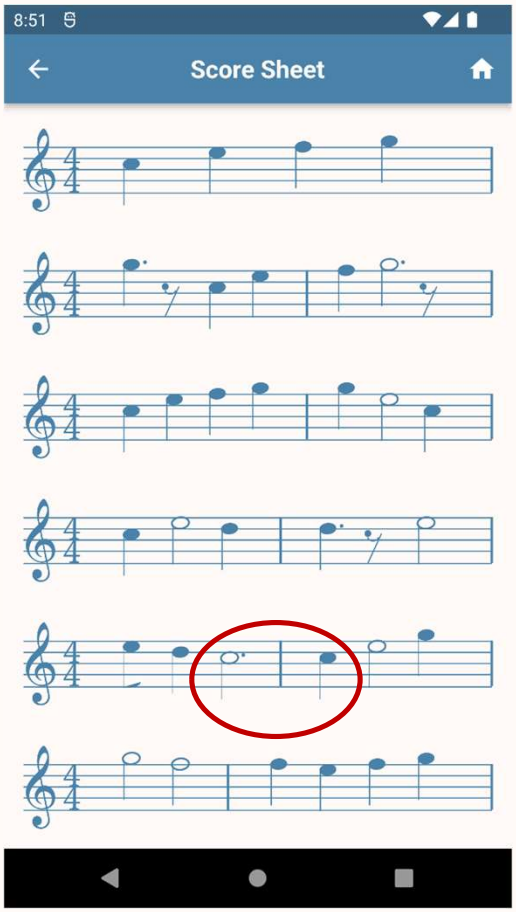
Saints Go Marching On

Lastly, we test out with Saints Go Marching On to determine the extent at which the algorithm is unable to split notes.

Correct Score



Xylophone Score (80 BPM)



Note: The full accurate score is not shown here. View the full score in the app itself.

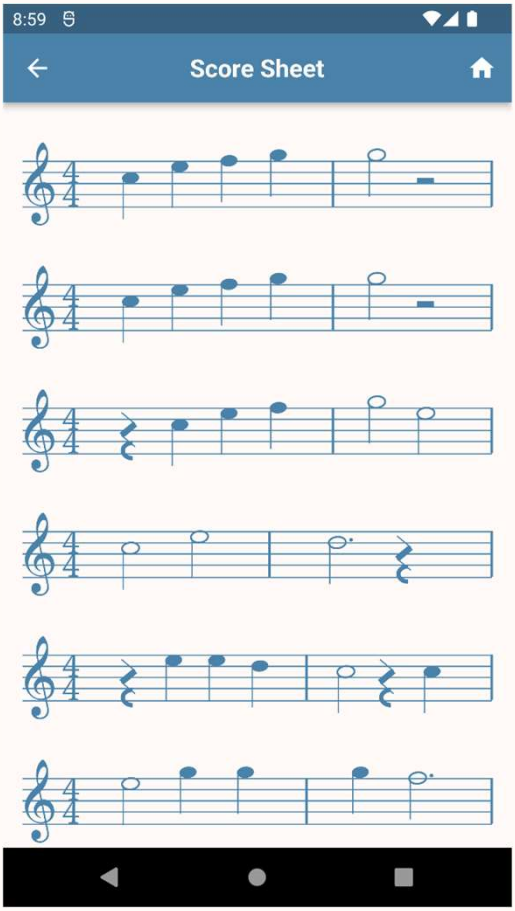
The rationale behind choosing this melody is due to the highlighted region. The song has a clear pause between these 2 notes in the song. However, it seems that the algorithm did not pick up on the pause and still managed to catch the tail end of the first note and not establish it as a pause. However, alleviating this would cause another issue in note reading which would be explained later.

12.2 ALGORITHM TESTING

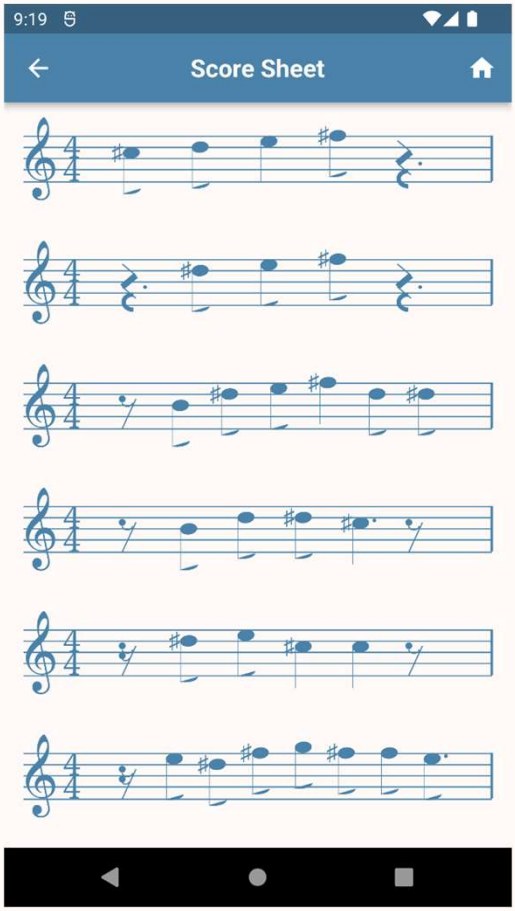
Saints Go Marching On

Lastly, we also test out the whistled rendition of the song. Here, we focus on a higher tempo to see whether the algorithm can handle faster paced melodies.

Correct Score



Whistle Score (100 BPM)



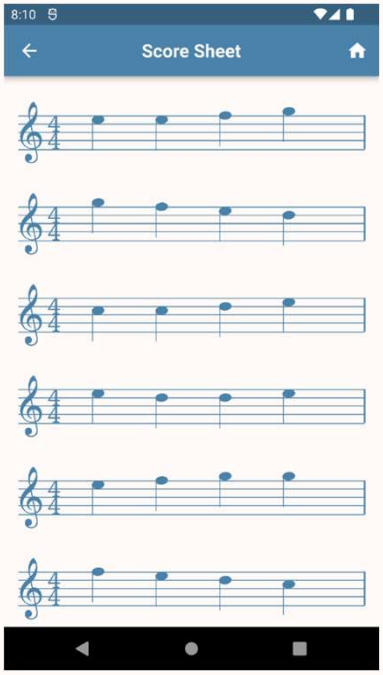
Note: The full accurate score is not shown here. View the full score in the app itself.

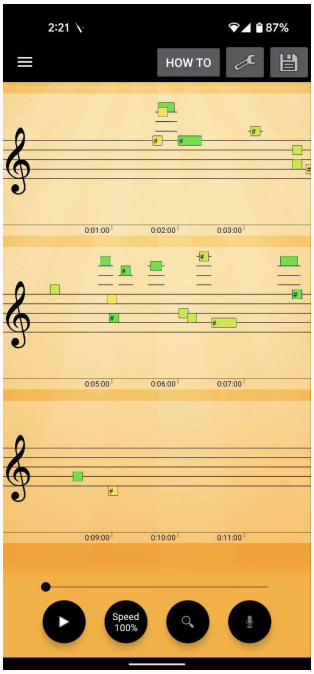
While the notes are not perfectly accurate, **the general trends of the notes seems to show good results.** Considering the whistle was not at the perfectly accurate pitch and that the whistled rendition was played at a much faster pace, this shows that the algorithm does fare decently well with faster paced samples while still playing them with relative accuracy.

12.2 ALGORITHM TESTING

Algorithm Comparisons

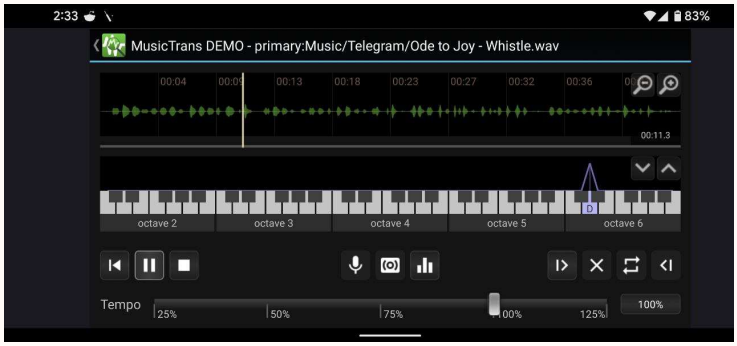
We then compare our algorithm performance with currently known note recognition applications. In both apps, we used Ode to Joy - Whistled as our sound sample.





Note Recognition

As the app only allows 15 seconds of audio recordings on their free trial, we testing this with a whistled sample of Ode to Joy and we can see that the application **does not perform well** with whistled samples.



Music Trans

This app performed much better in attaining the correct values for note transcription. However, the app does not provide a score of what notes were played in the audio file, making **it hard for the user to record a melody** for future purposes.

Altogether, we are proud to present a note recognition app which is **much more accurate than current apps** present in the market, as well as **provides more useful utility** to future musicians.

12.2 ALGORITHM TESTING

Post Testing Analysis

We are proud to report rather accurate results when analysing whistles and monophonic sounds like a xylophone. However, there are 2 main areas which present bigger errors and require work. Here are our current attempts at resolving them.

Note Splitting

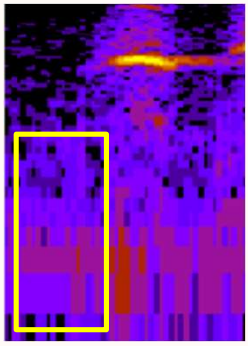
Often, it is seen that 2 notes are being conjoined into one singular note. The rationale behind this behaviour is because consecutive frequencies mapped to a singular note would add up, making the programme unable to register multiple counts of the same note in a row.

To address this, we have considered the amplitude of each audio splice and determining whether its higher than the previous splice, indicating a new note.

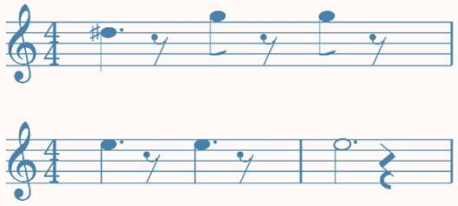
Background Noise

In some instances, we can see that there are spikes of incorrect notes, most likely attributed to noise. We have tried to alleviate this by implementing a minimum amplitude value a splice has to have before being recognized as a note. This was done as shown below.

By recognising the amplitude of the noise that often exists in sound samples, we intended to use this value as a minimum threshold level for all sound splices to have (as shown in the yellow box below). However, this leads a major issue in note mapping.



When implementing the threshold value, we noticed that rests were always reported after notes (as shown below). Unfortunately, the decay for whistles are too short and soft to be recognized. As such, for a cleaner and more accurate score, we decided to take the lesser of two evils and not implement a minimum amplitude.



By not stringently accounting for noise, we made the algorithm more susceptible to incorrect notes and rests. However, we will also look towards other forms of noise reduction to better our algorithm.

12.3 USER TESTING

As mentioned in section 2.4 of the README, the Android APK files for the Whistle Application can be downloaded from this folder: https://drive.google.com/drive/folders/1PwAhSlAeXEKEL0o1hMMvBXozdLgXnxVM?usp=sharing

*If you are downloading on your Android phone, you may have to allow APK downloads from Google Drive and trust the developer of the app.

*If you are downloading it to an android emulator, download the file and drag it to the simulator. The application will install automatically.

*A folder of audio samples is also given. Simply download and drag the files into your emulator if you wish to use these samples on an emulator. Alternatively download these samples onto your android device and search the file in your file directory when using the app.

We will be **going through the different screens** of the application in order, and mention the **current known errors and limitations**. It is also important to take note that **not all the buttons in the app have functionalities**, as we have yet to implement them for this current milestone.

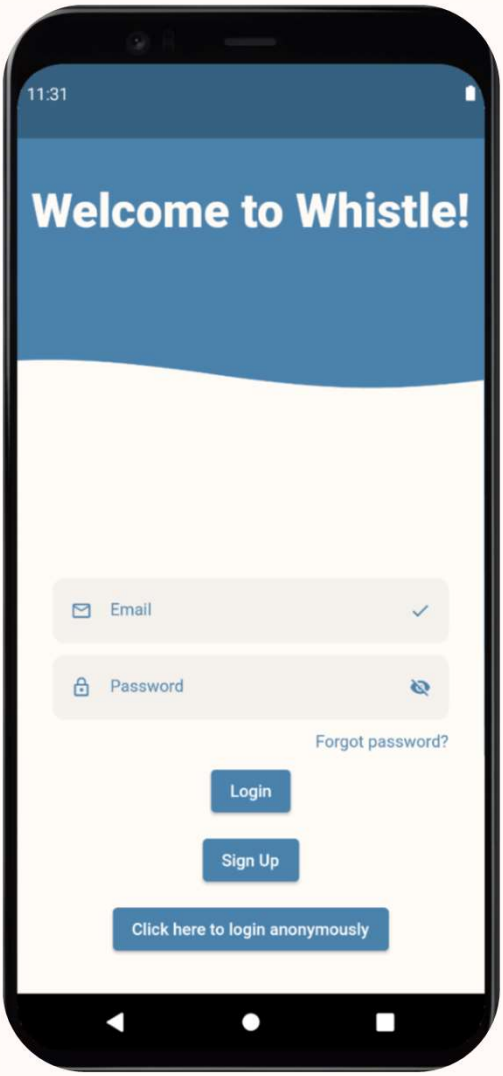
Detailed description of each screen can be **found in the subsequent pages.**

A table of ideal BPMs to each for each sound sample is also shown below.

Audio Sample	Ideal BPM
Happy Birthday - Xylophone	50
Happy Birthday - Whistle	50
Jingle Bells – Xylophone	80
Jingle Bells – Whistle	70
Ode to Joy – Xylophone	70
Ode to Joy – Whistle	80
Saints Go Marching On – Xylophone	80
Saints Go Marching On - Whistle	100

12.3 USER TESTING

Login Screen

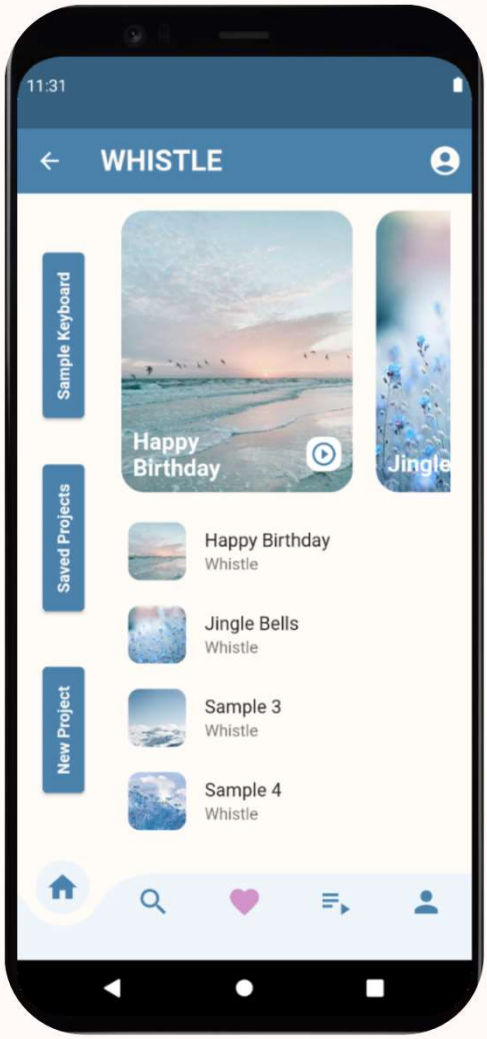


Known Errors
NIL

Limitations
- We have yet to implement the database required for the login details for this milestone.
- As such, to proceed to the home screen page of the app, simply **press the "Click here to login anonymously" button found at the bottom of the page.**

12.3 USER TESTING

Home Screen

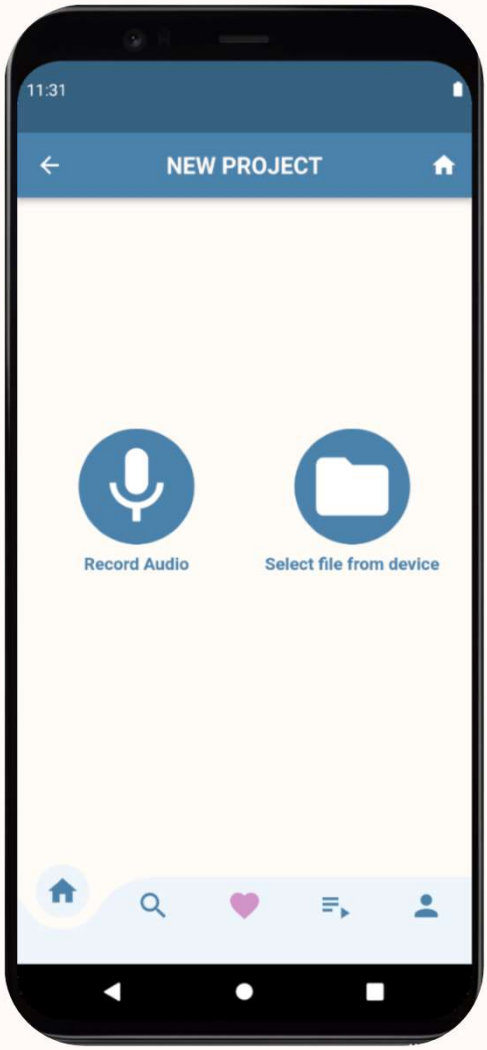


Known Errors
NIL

Limitations
- The only buttons with functionality are the 3 buttons at the side, namely **“Sample Keyboard”, ”Recent Projects”** and the **“New Project” button**.
- More functionality will be added, and the rest of the screen is there for design purposes.

12.3 USER TESTING

New Projects Page

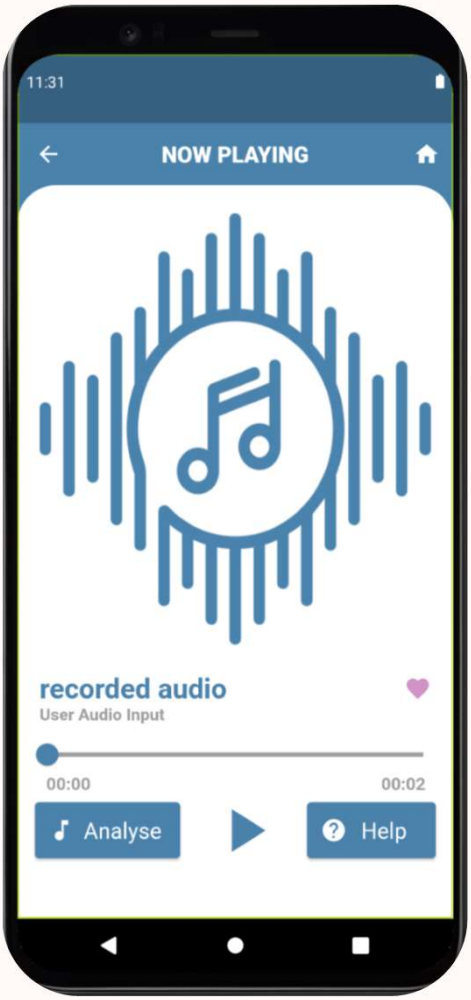


<u>Known Errors</u>
NIL

<u>Limitations</u>
- NIL
- Both the “Record Audio” and “Select file from device” button works.

12.3 USER TESTING

Audio Player Page



<u>Known Errors</u>
NIL

<u>Limitations</u>
NIL

12.3 USER TESTING

Loading Screen

Known Errors

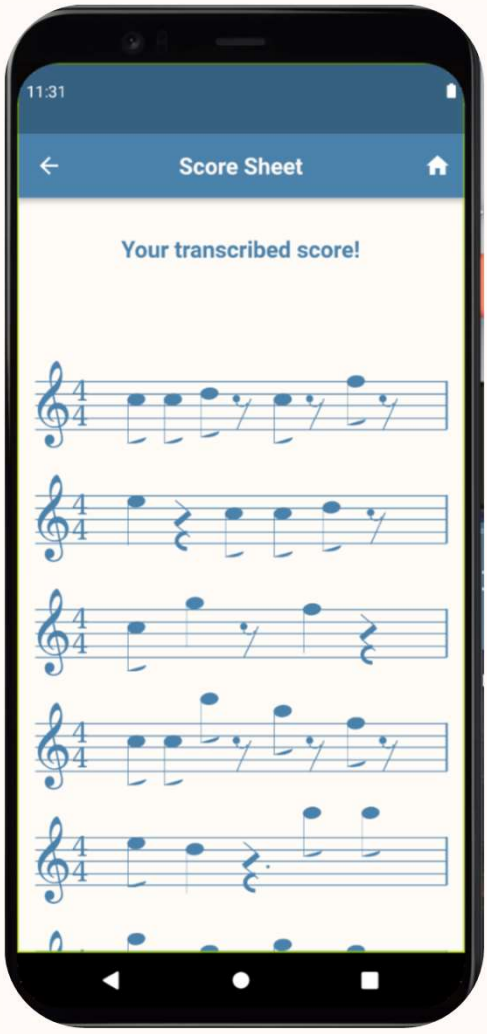
- The **Whistle application may crash** if users **press the "back" button.**
- Currently, our code does not allow for users to press the "back" button due to the way our code was structured. However, we will be fixing this problem such that when users try to press the "back" button, an error message will pop up and warn the users that the current project will not be saved.
- m4a files cannot be analysed because it isn't exactly in an audio format (mixture of audio and video format), as such the FFMPEG converter does not work. We will be fixing this issue for the next milestone submission.

Limitations

NIL

12.3 USER TESTING

Score Sheet Page

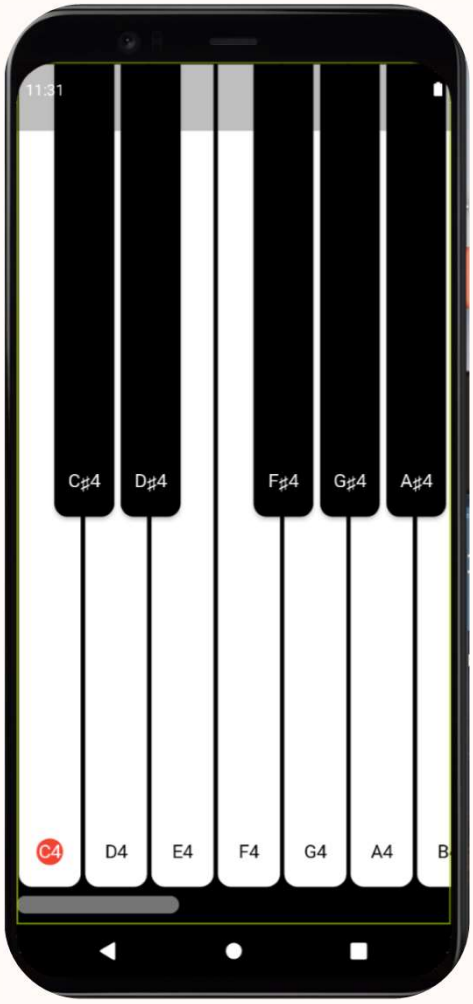


Known Errors
NIL

Limitations
- If there are repeated notes played, the algorithm will detect them as a long note, and **one note will be printed out instead of multiple notes.**
- Our team will still be working on the algorithm even after milestone 3 as we hope to make the transcribed score as accurate as possible.

12.3 USER TESTING

Sample Piano Player



Known Errors
NIL

Limitations
NIL

12.3 USER TESTING

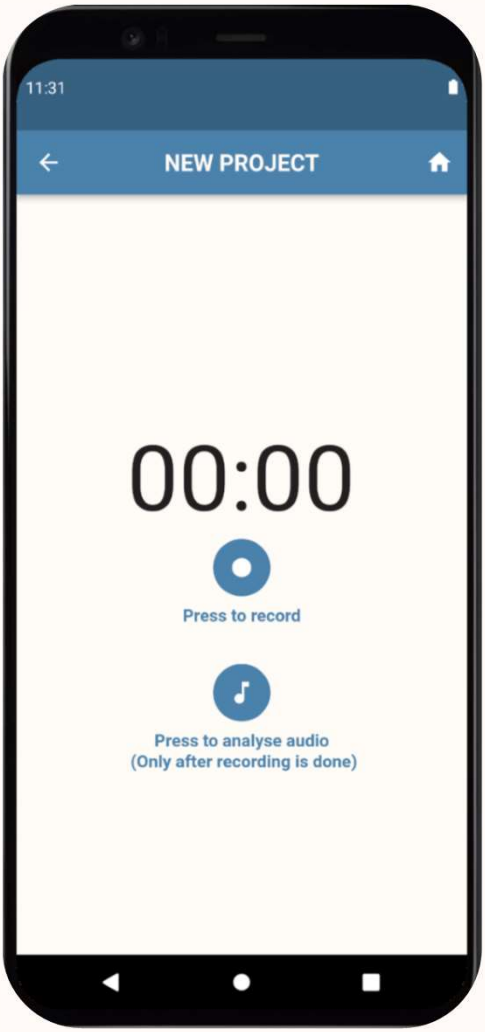
Saved Projects Page

Known Errors
NIL

Limitations
- Currently, users are able to change the name of the project title accordingly.
- Our team will be implementing a database to allow users to save their different projects on the app for the next milestone.

12.3 USER TESTING

Record Audio in App



Known Errors
NIL

Limitations
NIL

13. RESPONSES TO EVALUATION

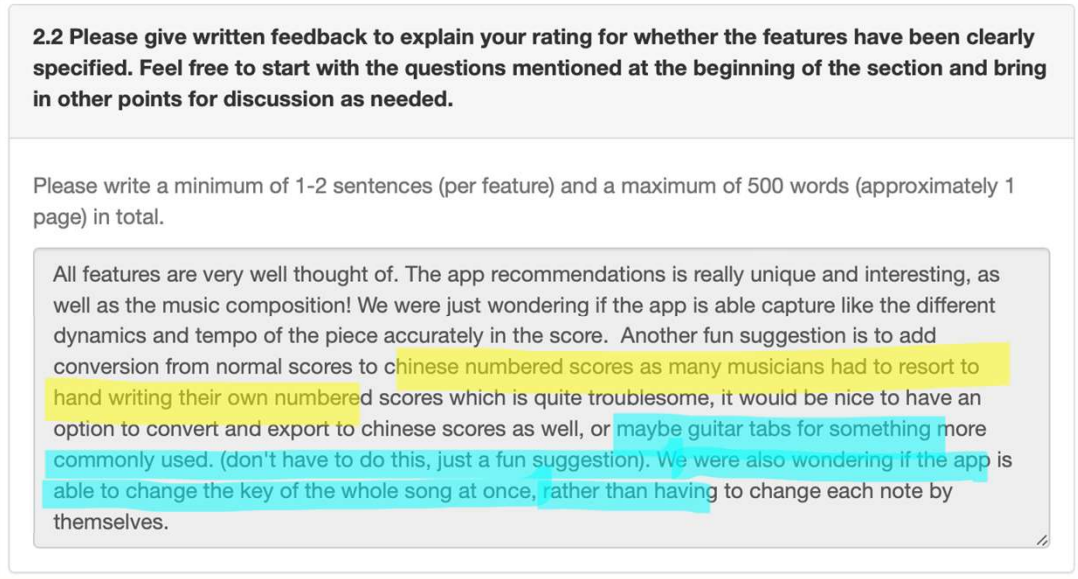
13. RESPONSES TO EVALUATION

We have compiled a few responses from our advisor, Miss Charisma Kausar, as well as the other teams, and we would like to respond to them as we feel that they have **greatly helped us in improving our application.** In addition, we feel that the **feedback given by the other groups and our advisor was also a form of user testing**, as they had to try and use our whistle application before they could provide us with feedback. We had received **a total of 9 user testing from other groups** throughout our journey in orbital. As such, we would also like to **thank them for taking the time to look through our project during the milestone evaluations**.

While there were many interesting suggestions given to us, we regrettably can only implement these many features in the short span of three months throughout our Orbital journey. Nonetheless, we would like to assure you that we have taken all these suggestions into consideration and hence would like to take the opportunity to express our gratitude by responding to the feedback we have received throughout the 3 milestones.

The responses have been categorized according to the different milestones and can be found in the subsequent pages.

13.1 RESPONSES TO EVALUATION - MILESTONE 1

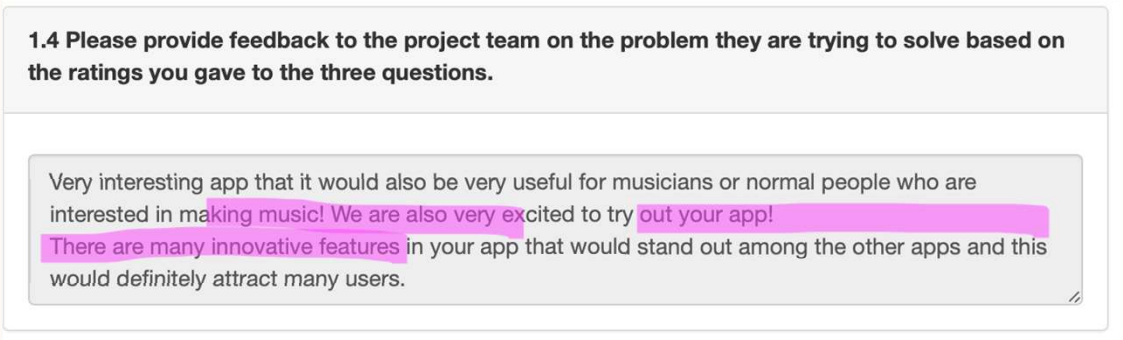


2.2 Please give written feedback to explain your rating for whether the features have been clearly specified. Feel free to start with the questions mentioned at the beginning of the section and bring in other points for discussion as needed.

Please write a minimum of 1-2 sentences (per feature) and a maximum of 500 words (approximately 1 page) in total.

All features are very well thought of. The app recommendations is really unique and interesting, as well as the music composition! We were just wondering if the app is able capture like the different dynamics and tempo of the piece accurately in the score. Another fun suggestion is to add conversion from normal scores to chinese numbered scores as many musicians had to resort to hand writing their own numbered scores which is quite troublesome, it would be nice to have an option to convert and export to chinese scores as well, or maybe guitar tabs for something more commonly used. (don't have to do this, just a fun suggestion). We were also wondering if the app is able to change the key of the whole song at once, rather than having to change each note by themselves.

Response: We would like to first thank this team for having the faith in us by mentioning that the "app recommendations is really unique and interesting" *(highlighted in yellow)*. With regards to their suggestion to "add conversion from normal scores to Chinese numbered scores" *(highlighted in blue)*, we understand this is a real concern for Chinese musicians as some are still reading numeric scores. However, upon further research, we realised that most Chinese musicians are slowly turning towards reading stave notations as most of the Chinese scores are being transcribed into stave notations. As such, our team has decided to **prioritise printing out a transcribed score in stave notations** and **improving on the accuracy of the printed notes.**

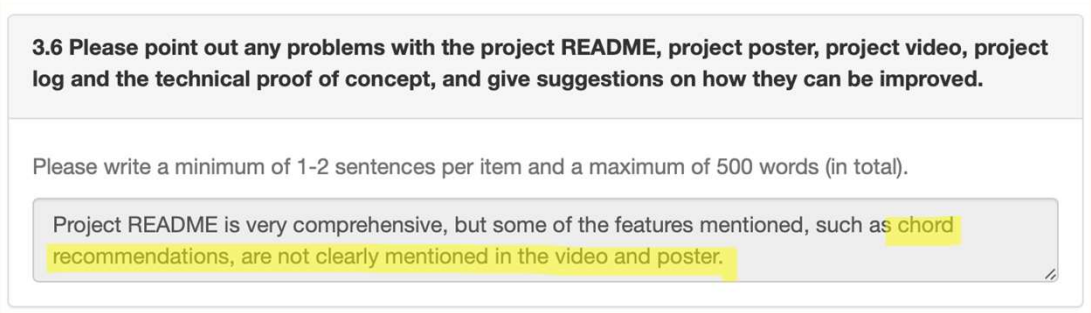


1.4 Please provide feedback to the project team on the problem they are trying to solve based on the ratings you gave to the three questions.

Very interesting app that it would also be very useful for musicians or normal people who are interested in making music! We are also very excited to try out your app!
There are many innovative features in your app that would stand out among the other apps and this would definitely attract many users.

Response: We would once again like to thank this group once again for mentioning that "there are many innovate features in your app" and it will "stand out among the other apps and this would definitely attract many users" *(highlighted in purple)*. We are glad to receive this feedback as this was the main drive our of project – which is **to improve on the current music applications as we felt that there were too many features lacking, such as the inaccuracy of the notes printed out and the inability to allow users to export their score sheets for future purposes.**

13.1 RESPONSES TO EVALUATION - MILESTONE 1

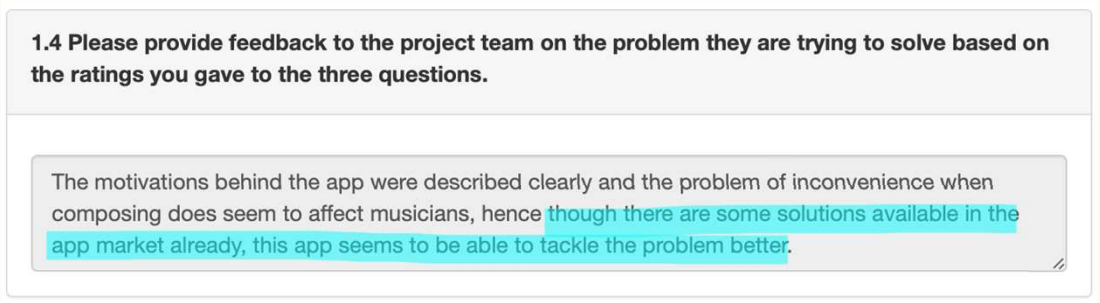


3.6 Please point out any problems with the project README, project poster, project video, project log and the technical proof of concept, and give suggestions on how they can be improved.

Please write a minimum of 1-2 sentences per item and a maximum of 500 words (in total).

Project README is very comprehensive, but some of the features mentioned, such as chord recommendations, are not clearly mentioned in the video and poster.

Response: We would like to thank this team for their valuable input. We understand that we have included chord recommendations as one of the features of our application. In the feedback, it was mentioned that "some of the features, such as chord recommendations, are not clearly mentioned in the video and poster" *(highlighted in yellow)*. To address this concern, our group has decided to **prioritise the accuracy of the notes printed out on the staves**, before we proceed with implementing the more complicated features such as chord recommendations. Therefore, we have decided to not include them in the video and poster first in case we do not have sufficient time to implement this feature.



1.4 Please provide feedback to the project team on the problem they are trying to solve based on the ratings you gave to the three questions.

The motivations behind the app were described clearly and the problem of inconvenience when composing does seem to affect musicians, hence though there are some solutions available in the app market already, this app seems to be able to tackle the problem better.

Response: We would like to once again thank this team for mentioning that "though there are some solutions available in the app market already, this app seems to be able to tackle the problem better" *(highlighted in blue)*. As mentioned earlier, we are glad to receive this feedback as this was the main drive our of project – which is **to improve on the current music applications as we felt that there were too many features lacking, such as the inaccuracy of the notes printed out and the inability to allow users to export their score sheets for future purposes.**

13.1 RESPONSES TO EVALUATION - MILESTONE 1

2.4 Please explain your rating on whether the set of features selected for implementation is appropriate. Can the system be demonstrated after the feature set is completed in this phase?

Please write a minimum of 1-2 sentences and a maximum of 500 words (approximately 1 page)

Set of features seems complex and sufficient enough to develop a working system by the end of the phase.

3.6 Please point out any problems with the project README, project poster, project video, project log and the technical proof of concept, and give suggestions on how they can be improved.

Please write a minimum of 1-2 sentences per item and a maximum of 500 words (in total).

All aspects have been clearly addressed. Good job!

Response: We will be addressing these 2 feedback together. We would like to thank this team for taking their time to provide us with constructive feedback and we will **continue to keep up the effort and work on implementing our features for the next milestone.**

13.1 RESPONSES TO EVALUATION - MILESTONE 1

Peer Evaluation from Advisor (Miss Charisma Kausar)

1.4 Please provide feedback to the project team on the problem they are trying to solve based on the ratings you gave to the three questions.

It is clear that the target audience is anyone interested in music composition. The problem is well-described in the README. The team has also discussed the problem with friends pursuing music. The solution proposed is a useful and innovative way to solve the problem with their own algorithm, and the research group approves of their app as well.

3.6 Please point out any problems with the project README, project poster, project video, project log and the technical proof of concept, and give suggestions on how they can be improved.

Please write a minimum of 1-2 sentences per item and a maximum of 500 words (in total).

Poster - Poster looks great!

Video - The video demonstrates the current capabilities of your app very well. Impressed with the sample data and the music player UI you have on your app right now! Love the splash screen as well. I believe you've used your own Android phone to demo the app, but using an emulator would be better since we can see where the mouse pointer is moving and where you are clicking to follow your walkthrough better. And it would nice to hear both the team members speak about the newly developed features. But still, great demo of the note recognition features!

README - Great design and descriptions on the README! Though I appreciate the hard work a lot :), I would recommend lesser background graphics for pages other than the cover page just to keep the format consistent with other teams and to have a continuous flow of information as Artemis READMEs could get really long. Your software engineering practices are evident from the activity diagram that very closely follows the UML format and the explanation you have given about your algorithm. However, for your next milestone submission, you could mention about version control, project management and your architecture diagram minimally as evidence of your software engineering practices. Do include the Android APK file for your app next time as well!

Project Log - The spreadsheets give a very good idea of the workload of each team member for their Orbital project.

Response: We would like to express our heartfelt gratitude to our advisor, Miss Charisma Kausar for supporting us along the way and for always providing valuable insights on how we can improve our application.

Regarding the feedback on "I would recommend lesser background graphics for pages other than the cover page" *(highlighted in yellow)*, our team has decided to keep the theme as we feel that the brown strip at the side with the whistle logo at the top and the page number at the bottom of the page is to keep **the consistency of our README, so that it would be easier for the users to follow through.**

Regarding the feedback on "you could mention about version control, project management and your architecture diagram minimally as evidence of your software engineering practices" and "include the Android APK file for your app next time as well" *(highlighted in blue)*, **we have included them for our milestone 2 submission**.

13.2 RESPONSES TO EVALUATION - MILESTONE 2

Uploading of audio files - seems to work. Tested with multiple wav files. Accepted.

Score transcription - able to transcribe but not very accurate at the moment. Marginally accepted

Score review - very useful for when we make mistakes in our recording or want to improve something

App recommendation - good feature to have but not the most important of all the other features

Music composition - useful feature

Exporting - useful but not the most important

Improving accuracy of notes - very essential as this is the main purpose of the app (some files produced blank transcripts, not sure if its because our audio was too soft or emulator issues, maybe can test more with different volumes). Getting the rhythm correctly shown is also quite important.

Fix m4a file issue - good to implement so we do not have to convert our files

All features are of sufficient complexity for Artemis

README - very detailed

Poster - Poster not updated to A1 size with the necessary details, check microsoft teams for more details.

Video - Clearly shows how to use the features

Project log - Clearly shows hours spent for both members

Prototype - Working well

Response: Our team would like to thank this team for accepting most of the functions we came up with. With regards to the score transcription function, our team would also be working on the accuracy for milestone 3's submission. One team also mentioned that the exporting function is useful, but not the most important, and we agree with that as well. Our team has decided to work on coming up with a login function for our app first instead of the exporting function. This is because after much consideration, we **realised that users can easily view the transcribed scores on their phones through the app** since it will already be stored locally on the Whistle application and the function for them to export it is not as high of priority compared to having a login function for the user.

13.2 RESPONSES TO EVALUATION - MILESTONE 2

Feature 1 (Transcribe audio input into a musical score): Working. Accepted.
Feature 2 (Uploading of audio files): Working. Accepted.
Feature 3 (Score Transcription): Working. Accepted.
Feature 4 (Score Review): Working. Accepted.
Feature 5 (App Recommendations): Working. Accepted.
Feature 6 (Music composition): Working. Accepted.
Feature 7 (Export): Working. Accepted.

- Login feature - No authentication needed but can access home page. Marginally accepted
- New projects feature - Can record audio and replay it. Cannot test import file. Accepted
- Score sheet page feature - Clicking on music icon leads to the score sheet. Accepted
- Sample piano player - Can access. Accepted
- Recent project - Can access. Accepted

I can see a clear direction of the app and who the target audience is. Albeit incomplete, the buttons are working and accessible. Overall, features are reasonable and sufficient in complexity, especially the score generator.

Response: Our team would like to thank this team for accepting all of the functions we came up with. Similar to the previous response, with regards to the score transcription function, our team would also be working on the accuracy for milestone 3's submission. One team also mentioned that the exporting function is useful, but not the most important, and we agree with that as well. Our team has decided to work on coming up with a login function for our app first instead of the exporting function. This is because after much consideration, we **realised that users can easily view the transcribed scores on their phones through the app** since it will already be stored locally on the Whistle application and the function for them to export it is not as high of priority compared to having a login function for the user. We will also be working on the authentication part of the login as users can only log in anonymously currently.

13.2 RESPONSES TO EVALUATION - MILESTONE 2

Peer Evaluation from Advisor (Miss Charisma Kausar)

Feature 1 - Record audio: Was able to record audio but not able to hear it when I play it back even though my simulator's volume was at full. May be an Android emulator issue. The music playing progress animation is very helpful to the user. Accepted as shown in video.
Feature 2 - Upload audio: Cannot be tested as I do not have audio files on my emulator. Good job on the error handling though when file selection is not allowed or file has not been selected by the user. Accepted as shown in the demo video.
Feature 3 - Music transcription: Great loading animation! I wasnt able to see the transcribed score for a 1 second audio even after about 5 minutes. Not sure what the issue is. Accepted as shown in video.

^Most/all of these features had to be accepted based on what was seen in the video. Perhaps next time, you can let the testers know what is the recommended way for testing (real Android device over emulator) so that such issues do not come up again.

M3 features:
1. Improve accuracy: very important as it is the main purpose of their app.
2. Export scores: helpful to continue work on other platforms.
3. Save projects: use of database; saving progress helps users come back to it later.
4. Login feature: helps user continue their work on the same app in another device or after local data is cleared.

The planned features when implemented will help the team reach the necessary requirements for their aimed level of achievement (Artemis).

The test cases used are appropriate for the app and cover all main use cases. Amazing!

README - Amazing quality documentation as always! Your detailed explanation of the note recognition algorithm, activity flow diagrams and multi-layered testing approach are strong evidence of your software engineering skills.
Poster - Love the new color scheme!
Video - Great video where all of the implemented (and not implemented) features are stated clearly with voiceovers.
Project Log - Gives clear idea of the workload of both teammates.
Prototype - Beautiful UI screens. The functionalities have been implemented well with sufficient error messages. Perhaps any commonly faced issues when using your app on an emulator or any audio files for testing can be provided by the team for the next milestone.

Response: Our team would like to thank these teams for accepting most of the functions we came up with. With regards to the score transcription function, our team would also be working on the accuracy for milestone 3's submission. One team also mentioned that the exporting function is useful, but not the most important, and we agree with that as well. Our team has decided to work on coming up with a login function for our app first instead of the exporting function. This is because after much consideration, we **realised that users can easily view the transcribed scores on their phones through the app** since it will already be stored locally on the Whistle application and the function for them to export it is not as high of priority compared to having a login function for the user.

14. FINAL THOUGHTS ON OUR PROJECT

14. FINAL THOUGHTS ON OUR ORBITAL PROJECT

As our team approaches the last milestone of our Orbital project, we would like to include our **final thoughts to summarise our takeaways from this project**. Our team also hopes that these insights will be useful to whoever is reading our README currently.

Simplified Representation of Software Engineering

We noticed that there was a **great difference in practical software engineering and Orbital, as there were no changes to our user stories as we worked on our project from milestone 1 to 3.** However, in the real life world, we understand that there bound to be differences as there will be changes in demands by the clients. These last minute changes will in turn result in re-evaluation of the client needs by the Business Intelligence teams of the companies and this will then result in the change of the user stories, which is usually done by the Development team of the company. As such, our team feels that orbital may not have given us a full representation of practical software engineering as we did not have to operate under the pressure of last minute user story changes. These changes could have meant that we may have to refactor our codes to suit the needs of the users. Nevertheless, our team still felt that the orbital journey had allowed us to experience majority of the software engineering practices in the real life world.

Alternative Tech Stacks

Our team decided to go with Flutter and Firebase because our mentor and advisor had prior experience to using them, and hence we thought it would be most suitable for us to choose these 2 tech stacks as we can ask them for help easily. Initially, we took a very long time to decide on our tech stacks and went around researching for the most efficient tech stack for us to work on our project. Eventually, we decided to use Flutter even though most of our friends were working on React Native.

The learning curve was definitely steep for the both of us as we had no prior experience coding in dart language and we spent a large amount of time learning to debug our code and it did compromise the number of features we managed to push out at the end of the project. However, we were able to receive help easily from our tutor and as such the debugging time was shortened. Hence, we would like to encourage the current reader of our README to do up your research for the most appropriate tech stack to work on your project, instead of rushing into it and following what the majority of your friends were using.

Developer vs Businessmen Perspective

Throughout the entire project, our team has been working on our project from a developer's point of view, which focuses more on the functionality of our application, and seldom as entrepreneurs and marketers of the application. We feel that this may have restricted us when we were developing our Whistle application as users will only pay if the solution that we are offering solves their problem. We understood that a product can be perfectly functional with many functioning features but is viewed to be poorly designed from a business perspective to make profits. As such, we feel that it would be beneficial for our team to invest more time into understanding how viable our solution is from a business perspective for future development projects. This is especially significant for us since our team had the intention to commercialise our application from the start.

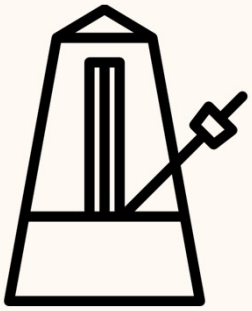
15. FUTURE DEVELOPMENTS

15. FUTURE DEVELOPMENTS

Due to the limited time given to work on our Whistle application, our team do have a few features that are currently under development but were unable to be completed. However, we will still continue working on them even after milestone 3.

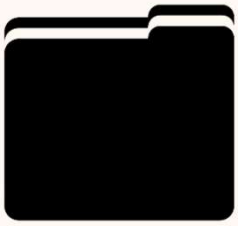
Metronome Function

We are looking towards adding a metronome at both the recording page as well as the audio player page in order for users to both keep tempo when they're whistling and decipher the best BPM that is suited to their song.



Saving Function

Unfortunately, we were not able to let users to save their scoresheet onto their phone as the our scores are drawn by flutter after analysis. As such, we could not find a quick way to encapsulate the entire score into PDF that can be easily read. In the future, we will be looking towards deciphering the notes into a MIDI format and writing the score on an external recognised score painter which users can download, or decipher how to encapsulate our score widget into a PDF.

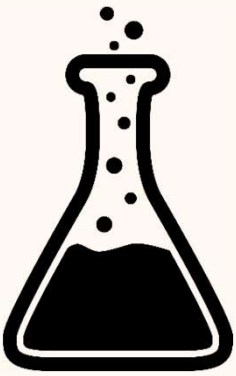


Account Features

Currently, our login features are under development and could not be completed before the end of milestone 3. However, with the completion of this feature, users will be able to create an account where they can store their own audio files recorded in app for future use and analysis.

Analysis Option

Lastly, we wish to give users more freedom to change settings of the analysis to attain more accurate readings. This includes allowing users to change the sensitivity of the algorithm (by changing how thorough they wish to analyse each note) as well as include an option for users to reduce noise if they feel like they have a noisy background.



16. PROJECT LOG

16.1 PROJECT LOG - MILESTONE 1

Lift-Off + MileStone 1

For milestone 1, our team spent most of the time watching YouTube videos and reading the documentation to learn the specifics of the tech stack that we are using in our project. We felt that it was necessary for us to learn the basics properly instead of jumping in blind to reduce the pain we have to debug our code later on.

S/N	Task	Date
1	Learning Tech Stack	9/5/22 to 11/5/22
2	Poster & Video for Liftoff	12/5/22
3	1st Meet-up for Milestone 1 (Discuss project idea & finalise updated proposals, poster & video)	12/5/22
4	Working on individual allocated tasks at home	12/5/22 to 29/5/22
5	2nd Meet-up for Milestone 1 (Finalise submission for Milestone 1)	30/5/22
6	Weekly meetings with mentor	Every Wed 9pm from 11/5/22 to 25/5/22

For a more detailed version of our project log:
https://docs.google.com/spreadsheets/d/1nYzfyp-o94Ntj4W5zVQ-1aLxdxw21uzjtwOvqPTqbyw/edit?usp=sharing

16.2 PROJECT LOG - MILESTONE 2

Milestone 2

For milestone 2, our team spent most of the time **implementing the basic features** and functions that we had promised, such as producing the transcribed scores from passing in an audio file. We also spent time finalising the colour theme of our User Interface and design, thought through the different pages we will need, to prevent minimal changes after this milestone.

S/N	Task	Date
1	Working on the algorithm to print out the scoresheets (Shayer)	25/5/22 to 30/6/22
2	Working on the User Interface, update certain pages and coming up with more essential pages (Jody)	25/5/22 to 30/6/22
3	Working on individual allocated tasks at home	25/5/22 to 30/6/22
4	Meet-up for Milestone 2 (Finalise submission for Milestone 1)	27/6/22
5	Weekly meetings with mentor	Every Wed 9pm from 30/5/22 to 27/6/22

For a more detailed version of our project log:
https://docs.google.com/spreadsheets/d/1nYzfyp-o94Ntj4W5zVQ-1aLxdxw21uzjtwOvqPTqbyw/edit?usp=sharing

16.3 PROJECT LOG - MILESTONE 3

Milestone 3

For milestone 3, our team spent most of the time **improving on the algorithm, such that the notes printed out would be more accurate.** We also spent more time working on improving the user interface.

S/N	Task	Date
1	Improving on the algorithm to print out more accurate notes on the scoresheets (Shayer)	30/6/22 to 25/7/22
2	Improving on the User Interface and implementing a login function (Jody)	30/6/22 to 25/7/22
3	Working on individual allocated tasks at home	30/6/22 to 25/7/22
5	Weekly meetings with mentor	Every Wed 9pm from 30/6/22 to 25/7/22

For a more detailed version of our project log:
https://docs.google.com/spreadsheets/d/1nYzfyp-o94Ntj4W5zVQ-1aLxdxw21uzjtwOvqPTqbyw/edit?usp=sharing

17. CONCLUSION

17. CONCLUSION

Our orbital project journey has been a fulfilling one where we had many takeaways from this project, such as learning an entirely different language to code in, and also the usage of Flutter and Firebase. We hope you had an enjoyable time reading our README for our application Whistle.