# Department of Computer Science & Engineering
# The University of Texas at Arlington


# Detailed Design Specification
# CSE 4317: Senior Design II
# Fall 2023


# Melody Masters
# SoundSync


**Adrian Ramos**
**Angel Aguirre**
**Edgar Hernandez**
**Patrick Ferguson**
**Benjamin Farmer**

## Revision History

| Revision | Date | Author(s) | Description |
|---|---|---|---|
| 0.1 | 9.20.2023 | AA, AR, BF, PF, EH | document creation |
| 0.2 | 9.24.2023 | AA, AR, BF, PF, EH | complete draft |
| 1.0 | 9.25.2023 | AA, AR, BF, PF, EH | Release Candidate V1 |

# CONTENTS

## List of Figures



## List of Tables

# 1 Introduction

SoundSync is a mobile application with the main goal of eliminating the need to manually turn a sheet music page when playing an instrument. SoundSync will allow its users to scan a physical piece of sheet music that will then be copied into a digital format that the user can then save to their account. Using this digital sheet music, SoundSync will also utilize audio recording to detect which notes are being played by the musician in order to keep track of where the user is on the page. Once the user is close to playing the notes displayed on the screen, the application will automatically display the next page to be played for them. If this automatic note detection fails, the user will have the option to use our bluetooth pedal to turn the page manually, but still hands-free. In order to perform these tasks, SoundSync will utilize cloud services, live audio processing techniques, and optical character recognition software. This document is heavily based on our architectural design document. The layer and subsystems are adapted from the ADS documents with more in depth descriptions of what each subsystem entails. We also used our system requirement specification constraints as a way to find what subsystem we would need for each layer.

# 2 System Overview

This is an overview of our entire system that makes up the SoundSync app. Starting with the database layer, this layer will be in charge of keeping track of users as well as images they upload. We will be using Firebase for this task as it makes authenticating and storing data easy. The UI layer will be everything the user can see and interact with. This is essential as since this is an app we need a UI for our users. The Sheet Music Tracker layer purpose is to take data from Sheet Scanner and Note Recognition and sync it so it can signal to the UI when it time to turn the page of the music sheet. The Sheet Scanner layer takes input from the user or the Database and scans the music sheet for all the notes. It then sends this data along with a formatted music sheet to the Sheet Music Tracker layer. The Note Recognition layer processes the audio from the user and arranges it in a useful data form so it can be used by the Sheet Music Tracker layer. Finally the Manual Page Turning Pedal layer is a pedal that sends a signal to the UI to turn the page.

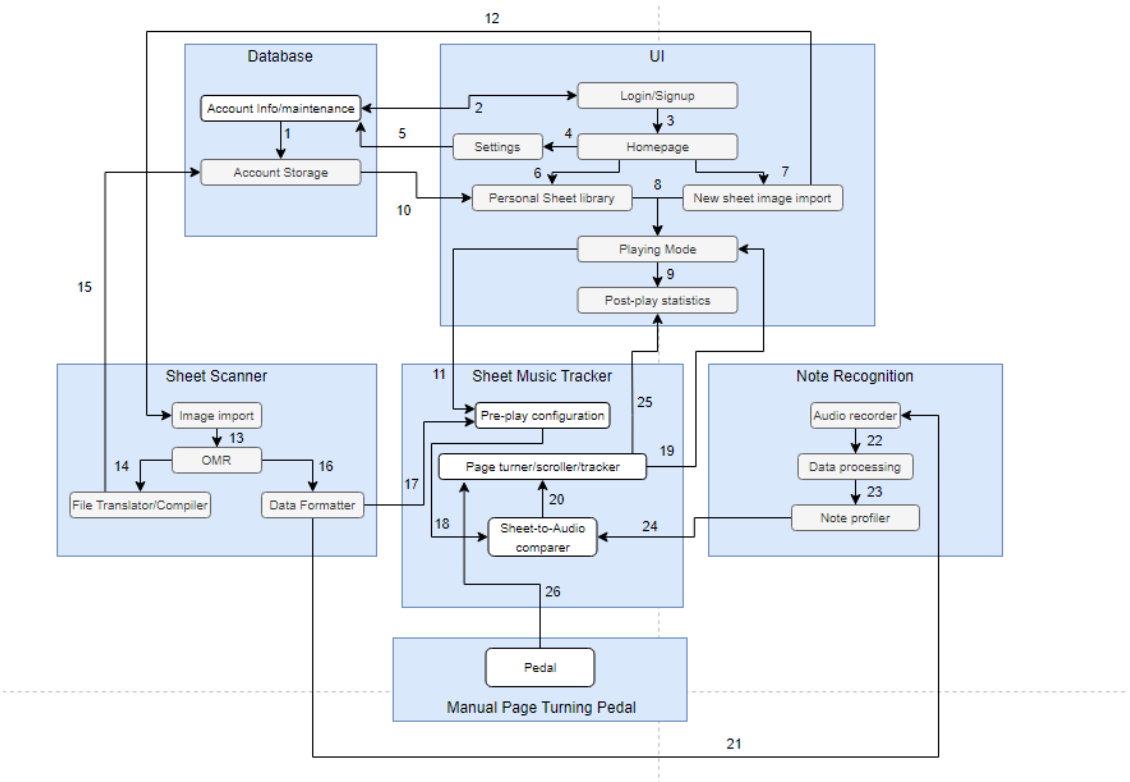


Figure 1: System architecture

# 3 Database

Description of the subsystems that make up the database layer of SoundSync

## 3.1 Layer Hardware

NA.

## 3.2 Layer Operating System

The database will be accessible from IOS and Android OS.

## 3.3 Layer Software Dependencies

From Firebase

- AUTH - Used for authenticating users within the Firebase database.
- STORAGE - A local storage used to store images from scanning music sheets.
- ref - Creates reference to specific location in storage.
- getDownloadURL - Gets download URL for a file from Firebase.
- onAuthStateChanged - Creates observer for AUTH state change (user logs in or out).
- uploadBytesResumable - Uploads data to Firebase storage.
- getMetadata - allows you to retrieve the metadata associated with a specific file or object stored in Firebase Cloud.

- createUserWithEmailAndPassword - Creates a user account in Firebase with email and password.
- signInWithEmailAndPassword - Uses AUTH service with the entered email and password to sign in the user.
- uploadBytesResumable - Uploads data to Firebase storage.
- sendPasswordResetEmail - Sends an email to the user's email from which they can change their password.

## 3.4 Account Info/Maintenance

We are using Firebase authentication services to verify and change login credentials. This includes the user's email and password. The user will also be able to reset their password in the case they forget it, and this will be reflected in the stored credentials.

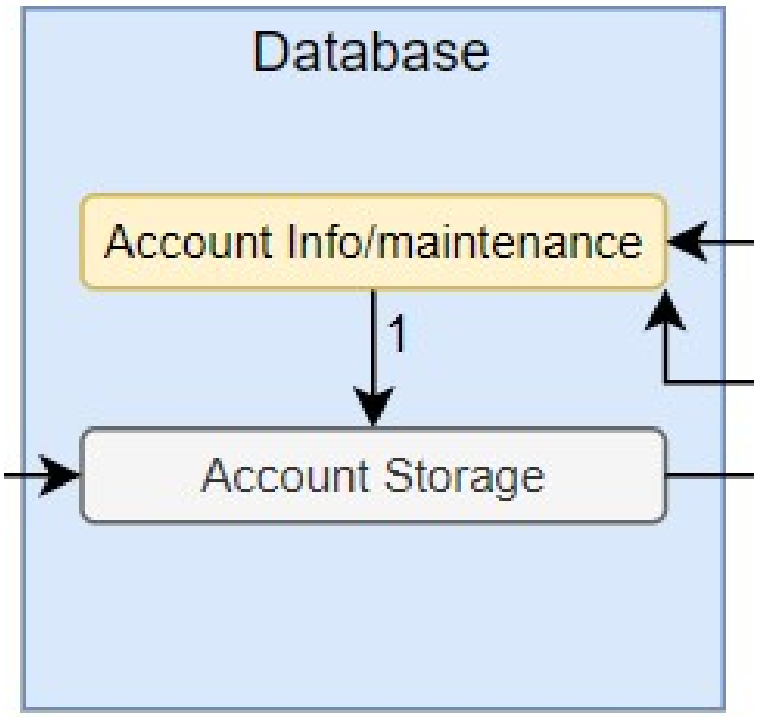


Figure 2: Account Info/Maintenance subsystem of Database layer

### 3.4.1 Subsystem Hardware

NA.

### 3.4.2 Subsystem Operating System

Accessible from IOS and Android OS.

### 3.4.3 Subsystem Software Dependencies

From Firebase

- AUTH - Used for authenticating users within the Firebase database.

- onAuthStateChanged - Creates observer for AUTH state change (user logs in or out).
- createUserWithEmailAndPassword - Creates a user account in Firebase with email and password.
- signInWithEmailAndPassword - Uses AUTH service with the entered email and password to sign in the user.
- sendPasswordResetEmail - Sends an email to the user's email from which they can change their password.

### 3.4.4 Subsystem Programming Languages

Firebase will be integrated into React Native/JavaScript code.

### 3.4.5 Subsystem Data Structures

Email and password will be entered as strings and then passed to signInWithEmailAndPassword or createUserWithEmailAndPassword.

### 3.4.6 Subsystem Data Processing

No note worth algorithms. This system works by when the user logs in or create an account, Firebase perform authentication and create an account in it Auth section with the user email and password. This Auth will be used whenever the user uploads anything to Firebase or downloads anything from Firebase. The user needs to be authenticated in order to interact with the web app.

## 3.5 Account Storage

We are using Firebase Cloud Storage. This is where user sheet music and images are stored. From here, users will be able to retrieve their music without re-scanning each sheet.

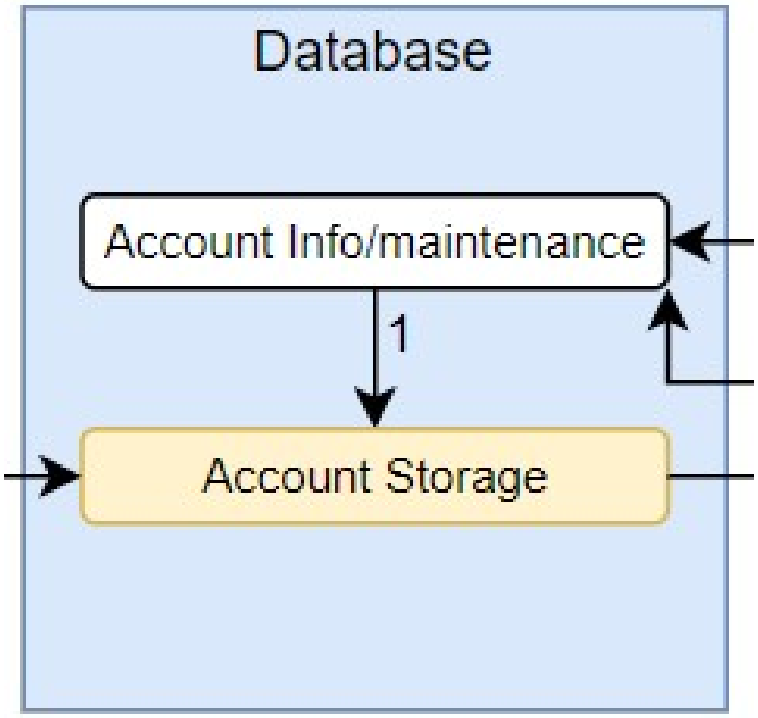


Figure 3: Storage subsystem of Database layer

### 3.5.1 Subsystem Hardware

NA.

### 3.5.2 Subsystem Operating System

Accessible from IOS and Android OS.

### 3.5.3 Subsystem Software Dependencies

From Firebase

- STORAGE - A local storage used to store images from scanning music sheets.
- ref - Creates reference to specific location in storage.
- getDownloadURL - Gets download URL for a file from Firebase.
- uploadBytesResumable - Uploads data to Firebase storage.
- getMetadata - allows you to retrieve the metadata associated with a specific file or object stored in Firebase Cloud.
- uploadBytesResumable - Uploads data to Firebase storage.

### 3.5.4 Subsystem Programming Languages

Firebase will be integrated into React Native/JavaScript code.

### 3.5.5 Subsystem Data Structures

uploadBytesResumable will upload the image jpg file to the images folder reference.

### 3.5.6 Subsystem Data Processing

No note worthy algorithms. The account storage is responsible for storing all pictures and Json for the user. When the user scan an image, the image is run through the Api so a note Json and a coordinate Json can be created. The image or images is sent to the user storage in Firebase. The Json are stored in folders in the same directory as the images. When the user is ready to go to play mode these images and Json are download from the storage for use.

# 4 UI Layer Subsystems

Description of the hardware components and software components that make up the UI layer.

## 4.1 Layer Hardware

Not applicable. No hardware components for this subsystem.

## 4.2 Layer Operating System

The program for this app will be running on an android tablet and as such will need a Linux operating system so it can run.

## 4.3 Layer Software Dependencies

From React Navigation

- NavigationContainer - responsible for setting up the navigation context and state for the entire navigation tree.
- createNativeStackNavigator - provide a more native stack navigator for React Native applications, which can improve performance and give a more native feel to your app's navigation.
- createBottomTabNavigator - creates a bottom tab navigator. A bottom tab navigator is a common navigation pattern in mobile applications where the main navigation options are displayed as tabs at the bottom of the screen.

From expo

- Ionicons - icon library used for adding icons to web and mobile applications,
- FontAwesome - provides a wide range of icons, both free and paid, and it is widely used in web and mobile development for adding visual elements to user interfaces.
- AntDesign - provides icons from the set used in web and mobile applications.
- Entypo - icons that are simple, clean, and versatile, making them suitable for various user interface elements in your mobile app.

From Screens

- Scan - Allows the system to create a scan button.
- Login - Allows the system to create a login button.
- AudioRecorder - Allows the system to create an Audio Record button.
- Create - Allows the system to create a Create button.
- Forgot - Allows the system to create a Forgot button.
- Profile - Allows the system to create a Profile button.
- Library - Allows the system to create a Library button.
- Folder - Allows the system to create a Folder button.

From Firebase

- Firebase Auth - Used for authenticating users within the Firebase database.
- Firebase Storage - A local storage used to store images from scanning music sheets.
- ref - Used to reference and image to a user.
- getDownloadUrl - Get download URL for a file from Firebase.
- listAll - method that allows you to retrieve a list of all items (files) in a specific Firebase Storage reference or location.
- getMetadata - allows you to retrieve the metadata associated with a specific file or object stored in Firebase Cloud Storage.

From components

- Card - common user interface pattern used in web and mobile applications to present information in a structured and visually appealing way.
- Drop - component allows users to drag elements and drop them onto designated areas, enabling interactive and intuitive user interfaces for tasks like rearranging items, file uploads, or building interactive dashboards.
- styles - styles are used to apply custom styling to the components, allowing you to control their appearance, layout, and visual behavior.
- SheetScanPrompt - component used to display text of confirmation for prompt.
- CollectionNamePrompt - component used to display text of confirmation for prompt.
- Header - eaders are commonly used to display information such as titles, navigation options, branding elements, and other important content.
- FadeTransition - type of animation or transition effect that gradually changes the opacity of an element in a user interface.

## 4.4 Login/Signup

The Login/Signup subsystem is supposed to be a method for any user to create a new account or use one they have previously created. This subsystem is vital for allowing the user a way to verify who they are so that the system knows what saved data to send to them.

### 4.4.1 Subsystem Hardware

No hardware component.

### 4.4.2 Subsystem Operating System

This subsystem will require the use of the Linux OS as that is the standard OS for android.

### 4.4.3 Subsystem Software Dependencies

From React

- useState - allows for updating a variable on screen when something changes.

From React Native

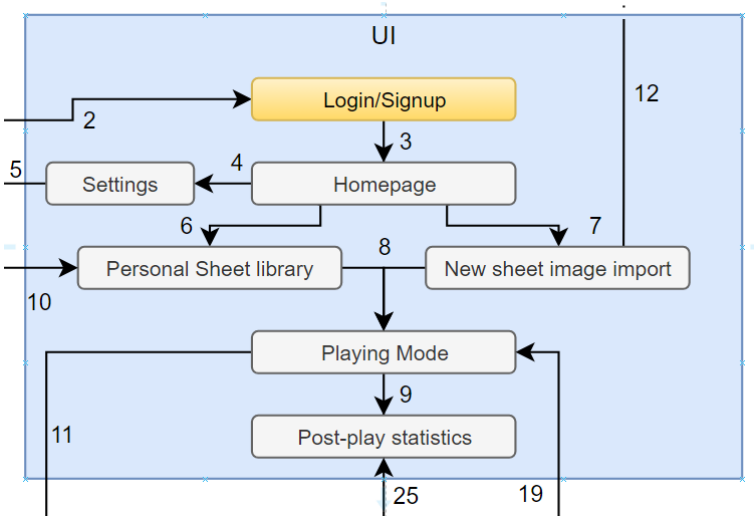


Figure 4: Login/Signup diagram

- View - A container that supports layout with flexbox, style, some touch handling, and accessibility controls. View maps directly to the native view equivalent on whatever platform React Native is running on, whether that is a UIView, , android.view, etc.
- TouchableOpacity - A wrapper for making views respond properly to touches. On press down, the opacity of the wrapped view is decreased, dimming it.
- Text - A React component for displaying text.
- TextInput - A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad.
- Image - component used to display images within your mobile app.

From Firebase

- Firebase Auth - Used for authenticating users within the Firebase database.
- Firebase signInWithEmailAndPassword - method provided by Firebase Authentication that allows a user to sign in to your application using their email and password.

#### 4.4.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

#### 4.4.5 Subsystem Data Structures

Not applicable

#### 4.4.6 Subsystem Data Processing

The user enters login information to be compared with Firebase Auth. If the login credentials match then a token is issued for the user so they can perform other actions while interacting with the app.

## 4.5 HOMEPAGE

The Homepage subsystem is in charge of showing the user what pages they have scanned previously in an easily browseable format. This homepage will act as a sort of dashboard and is the main UI component the user will see when they are not actively playing an instrument. This homepage will be where the user can visually see what data has been stored/generated from their use, such as sheets, performance data, and account info.

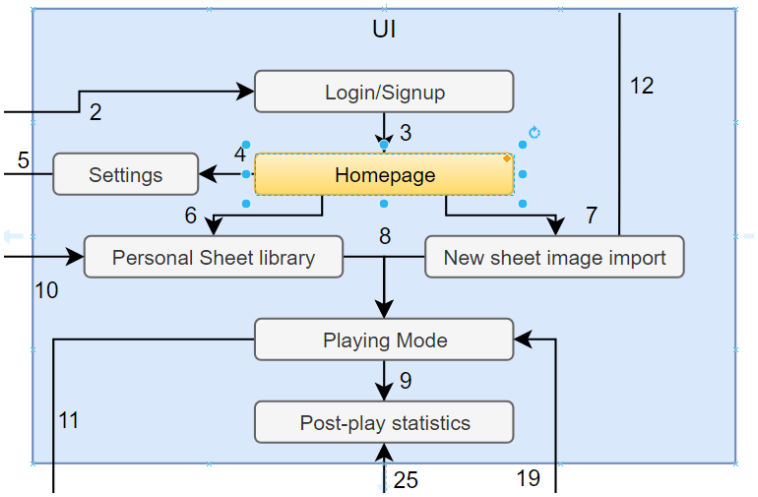


Figure 5: Homepage diagram

### 4.5.1 SUBSYSTEM HARDWARE

No hardware component.

### 4.5.2 SUBSYSTEM OPERATING SYSTEM

This subsystem will require the use of the Linux OS as that is the standard OS for android.

### 4.5.3 SUBSYSTEM SOFTWARE DEPENDENCIES

From React Navigation

- NavigationContainer - responsible for setting up the navigation context and state for the entire navigation tree.
- createNativeStackNavigator - provide a more native stack navigator for React Native applications, which can improve performance and give a more native feel to your app's navigation.
- createBottomTabNavigator - creates a bottom tab navigator. A bottom tab navigator is a common navigation pattern in mobile applications where the main navigation options are displayed as tabs at the bottom of the screen.

### 4.5.4 SUBSYSTEM PROGRAMMING LANGUAGES

The subsystem will be using react native code which is written in JavaScript.

### 4.5.5 SUBSYSTEM DATA STRUCTURES

Not applicable

### 4.5.6 Subsystem Data Processing

No note worthy algorithms. The homepage is where the users will see all of the UI. It will display tabs for the library and scanner page. Each tab corresponds to it own component. The scan page is where images are uploaded and Json data is created. The library is all the music sheets the user has scanned. From both pages the user can navigate to the tracker to perform in the play mode.

## 4.6 Personal Sheet Library

The Personal Sheet library subsystem will be responsible for organizing and presenting the previously scanned sheets to the user in a way that is both aesthetically pleasing and practical. When a musical piece is selected from the library, the subsystem should redirect the user to Playing Mode.

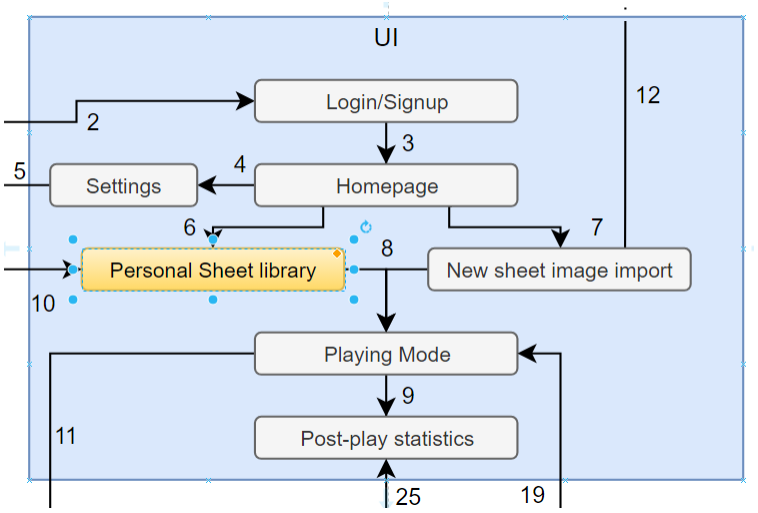


Figure 6: Personal Sheet Library diagram

### 4.6.1 Subsystem Hardware

No hardware components-.

### 4.6.2 Subsystem Operating System

This subsystem will require the use of the Linux OS as that is the standard OS for android.

### 4.6.3 Subsystem Software Dependencies

From React

- useState - allows for updating a variable on screen when something changes.
- useEffect - he useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.

From React Native

- View - A container that supports layout with flexbox, style, some touch handling, and accessibility controls. View maps directly to the native view equivalent on whatever platform React Native is running on, whether that is a UIView, , android.view, etc.

- ScrollView - A generic scrolling container that can contain multiple components and views. The scrollable items can be heterogeneous, and you can scroll both vertically and horizontally (by setting the horizontal property).
- TouchableOpacity - A wrapper for making views respond properly to touches. On press down, the opacity of the wrapped view is decreased, dimming it.
- Text - A React component for displaying text.
- TextInput - A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad.
- StyleSheet - A StyleSheet is an abstraction similar to CSS StyleSheets.

From Firebase

- Firebase Auth - Used for authenticating users within the Firebase database.
- Firebase Storage - A local storage used to store images from scanning music sheets.
- ref - Used to reference and image to a user.
- getDownloadUrl - Get download URL for a file from Firebase.
- listAll - method that allows you to retrieve a list of all items (files) in a specific Firebase Storage reference or location.
- getMetadata - allows you to retrieve the metadata associated with a specific file or object stored in Firebase Cloud Storage.

From components

- Card - holds the information for each collection and their subsequent images.
- Drop - custom component shows the user their settings and an option to sign out.

From expo

- Entypo - icons that are simple, clean, and versatile, making them suitable for various user interface elements in your mobile app.
- AntDesign - provides icons from the set used in web and mobile applications.

#### 4.6.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

#### 4.6.5 Subsystem Data Structures

Not applicable

#### 4.6.6 Subsystem Data Processing

The personal sheet library will pull from the user Firebase storage based on the sheet collection they pick. The app will send the chosen sheet collection name so it location in storage can be found then all corresponding music sheets can be downloaded and sent back to the app for use in the tracker.

## 4.7 New Sheet Image Import

No noteworthy algorithims. The New sheet image import subsystem is responsible for providing the user a way to take their real sheet music and digitally upload it to our application. This subsystem will take an image import, confirm it to the user, and then prompt the user if they would like to save the rendered result from the scanner.

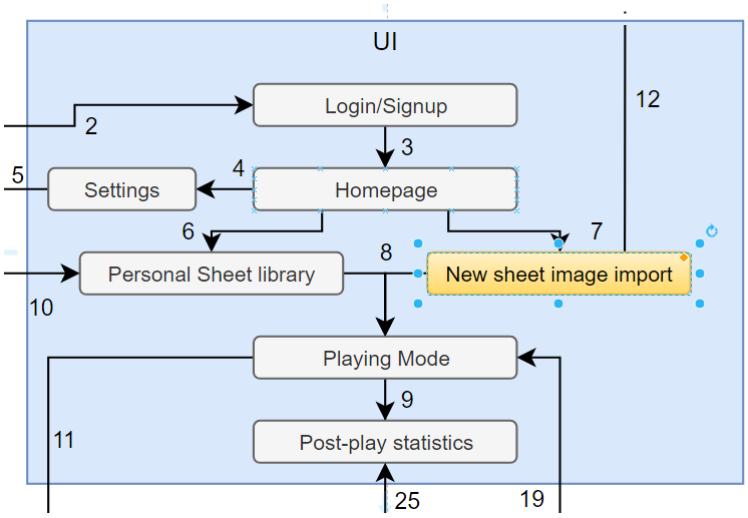


Figure 7: New Sheet Image Import diagram

### 4.7.1 Subsystem Hardware

No hardware components-.

### 4.7.2 Subsystem Operating System

This subsystem will require the use of the Linux OS as that is the standard OS for android.

### 4.7.3 Subsystem Software Dependencies

- useState - allows for updating a variable on screen when something changes.
- useEffect - he useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.

From React Native

- View - A container that supports layout with flexbox, style, some touch handling, and accessibility controls. View maps directly to the native view equivalent on whatever platform React Native is running on, whether that is a UIView, , android.view, etc.

From components

- styles - styles are used to apply custom styling to the components, controlling their appearance, layout, and visual behavior.
- SheetScanPrompt - custom component used to direct the user on image input

- CollectionNamePrompt - custom component used to direct the user on the firebase collectionName for the output
- Header - custom component used to display the top bar on the screen.
- FadeTransition - custom type of animation or transition effect that gradually changes the opacity of an element in a user interface.

### 4.7.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

### 4.7.5 Subsystem Data Structures

not applicable

### 4.7.6 Subsystem Data Processing

No noteworthy algorithms. This system will allow the user to pick a music sheet or sheets saved on their device and upload them through the app. Once the app has the images it needs to upload it will process them through the audiveris API. This will then process again through lilypond. After which the user will see the result and have the choice to proceed to the tracker.

## 4.8 Playing Mode

This subsystem will be responsible for rendering the required information to the user, such as their digital sheet music copy and an indication of where the system thinks they are on the page. The subsystem will also provide the next sheet in the book if the user is close to finishing the current one and there is more than one.

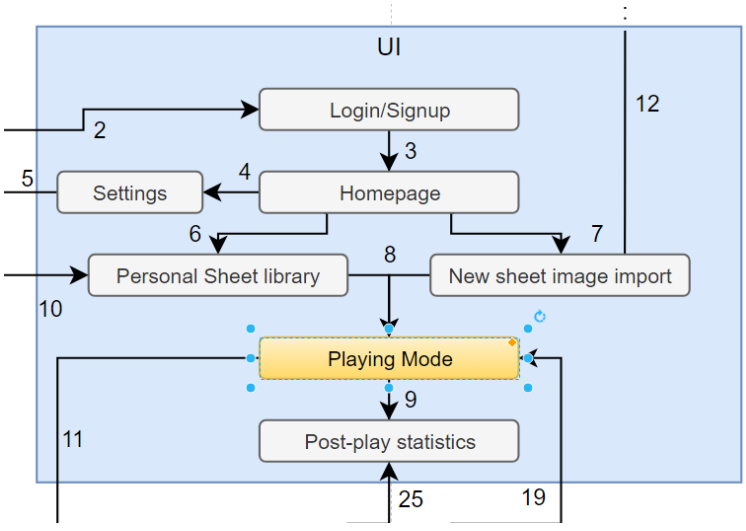


Figure 8: Playing Mode diagram

### 4.8.1 Subsystem Hardware

No hardware components.

### 4.8.2 Subsystem Operating System

This subsystem will require the use of the Linux OS as that is the standard OS for android.

### 4.8.3 Subsystem Software Dependencies

Playing mode will be entered with a stack navigator from the library.

### 4.8.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

### 4.8.5 Subsystem Data Structures

Will activate the pre-play configuration from the sheet music tracker.

### 4.8.6 Subsystem Data Processing

No note worthy algorithms. This system will display the users music sheets either from the library or the scanner. The user can hit a toggle button to begin playing and as they play a live audio server will process the notes and compare it with a json array downloaded from Firebase. If they match it will display highlighting on the music sheet so the user can follow along. And once the user reaches the end of the page scrolling will take place automatically.

## 4.9 Post-Play Statistics

The Post-Play statistics subsystem is responsible for showing the user how our system thinks they performed in terms of accuracy and time precision.

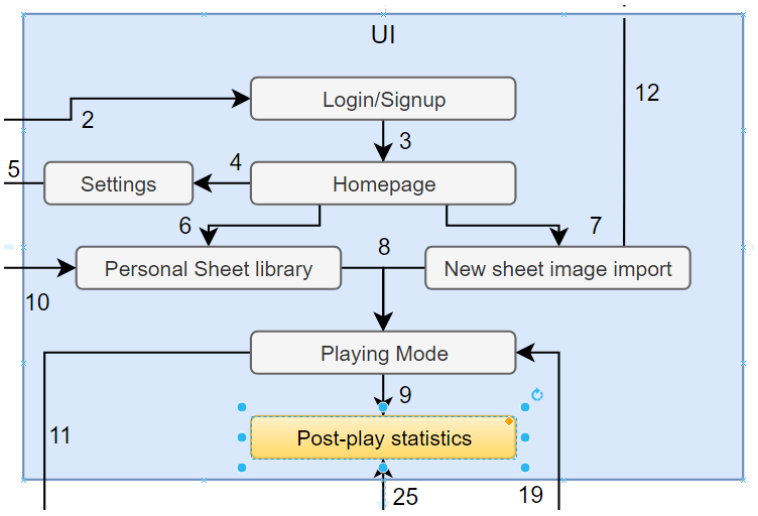


Figure 9: Post-Play Statistics diagram

### 4.9.1 Subsystem Hardware

No hardware components-.

### 4.9.2 Subsystem Operating System

This subsystem will require the use of the Linux OS as that is the standard OS for android.

### 4.9.3 Subsystem Software Dependencies

Post play Statistics will be entered after playing mode is finished.

### 4.9.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

### 4.9.5 Subsystem Data Structures

This will be entered with the information from the playing data.

### 4.9.6 Subsystem Data Processing

No note worthy algorithm.

# 5 Sheet Music Tracker

Description of the hardware components and software components that make up the sheet music tracker layer.

## 5.1 Layer Hardware

Not applicable no hardware components for this subsystem.

## 5.2 Layer Operating System

The program for this app will be running on an android tablet and as such will need a Linux operating system so it can run.

## 5.3 Layer Software Dependencies

From React

- useState - allows for updating a variable on screen when something changes.
- useEffect - he useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.
- useRef - A React Hook that lets you reference a value thatâs not needed for rendering.

From React Native

- View - A container that supports layout with flexbox, style, some touch handling, and accessibility controls. View maps directly to the native view equivalent on whatever platform React Native is running on, whether that is a UIView, , android.view, etc.
- ScrollView - A generic scrolling container that can contain multiple components and views. The scrollable items can be heterogeneous, and you can scroll both vertically and horizontally (by setting the horizontal property).
- Modal - A basic way to present content above an enclosing view.
- Alert - Launches an alert dialog with the specified title and message.
- Button - A basic button component that should render nicely on any platform. Supports a minimal level of customization.
- Animated - The Animated library is designed to make animations fluid, powerful, and painless to build and maintain. Animated focuses on declarative relationships between inputs and outputs, configurable transforms in between, and start/stop methods to control time-based animation execution.
- Easing - The Easing module implements common easing functions. This module is used by Animated.timing() to convey physically believable motion in animations.
- TouchableOpacity - A wrapper for making views respond properly to touches. On press down, the opacity of the wrapped view is decreased, dimming it.
- Text - A React component for displaying text.

- TextInput - A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad.
- StyleSheet - A StyleSheet is an abstraction similar to CSS StyleSheets.
- Dimensions - Allows for obtaining an application width and height.
- StatusBar - Component to control the app's status bar. The status bar is the zone, typically at the top of the screen, that displays the current time, Wi-Fi and cellular network information, battery level and/or other status icons.

From Firebase

- Firebase Auth - Used for authenticating users within the Firebase database.
- Firebase Storage - A local storage used to store images from scanning music sheets.
- ref - Used to reference and image to a user.
- getDownloadUrl - Get download URL for a file from Firebase.

Other Imports

- Audio from expo-av - Allows you to implement audio playback and recording in your app.
- fft from mathjs - Calculate N-dimensional fourier transform
- entypo from @expo/vector-icons - An icon from expo vector icons
- Audiveris from api - Audiveris is a software used to pull the notes from a music sheet.
- lilypond from api - Lilypond allows for displaying a music sheet in a manner that suits are needs.

## 5.4 Pre-play configuration

The Pre-Play Configuration is supposed to account for any errors in building the sheet music page. Incase the beats per minute cant be found or an incorrect musical note is in the music sheet, the user can go in and modify it themselves. It will be a screen within our react native app that will use and import other components in order to be operational.

### 5.4.1 Subsystem Hardware

No hardware component.

### 5.4.2 Subsystem Operating System

This subsystem will require the use of the Linux OS as that is the standard OS for android.

### 5.4.3 Subsystem Software Dependencies

From React

- useState - allows for updating a variable on screen when something changes.
- useEffect - he useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.

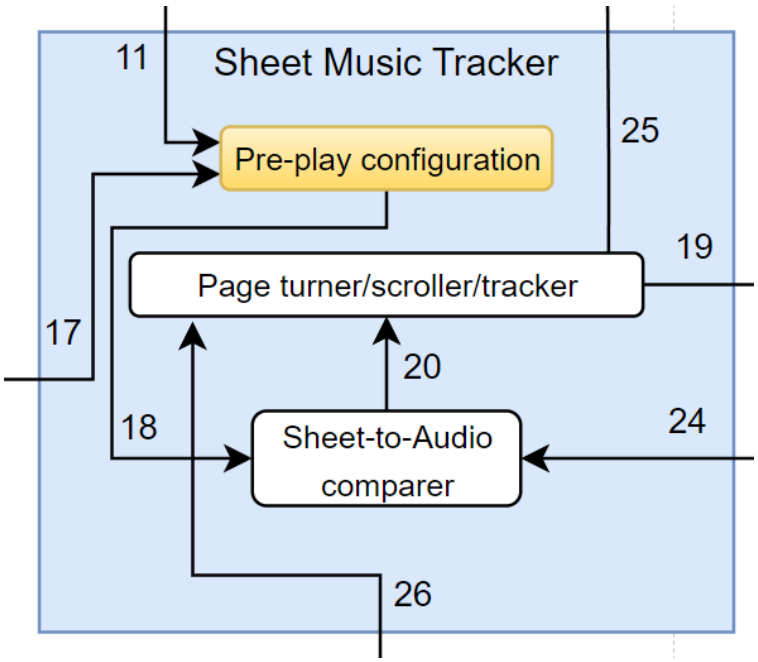


Figure 10: Pre-Play Configuration diagram

- useRef - A React Hook that lets you reference a value thatâs not needed for rendering.

From React Native

- View - A container that supports layout with flexbox, style, some touch handling, and accessibility controls. View maps directly to the native view equivalent on whatever platform React Native is running on, whether that is a UIView, , android.view, etc.
- ScrollView - A generic scrolling container that can contain multiple components and views. The scrollable items can be heterogeneous, and you can scroll both vertically and horizontally (by setting the horizontal property).
- Modal - A basic way to present content above an enclosing view. platform. Supports a minimal level of customization.
- Animated - The Animated library is designed to make animations fluid, powerful, and painless to build and maintain. Animated focuses on declarative relationships between inputs and outputs, configurable transforms in between, and start/stop methods to control time-based animation execution.
- Easing - The Easing module implements common easing functions. This module is used by Animated.timing() to convey physically believable motion in animations.
- TouchableOpacity - A wrapper for making views respond properly to touches. On press down, the opacity of the wrapped view is decreased, dimming it.
- Text - A React component for displaying text.
- TextInput - A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad.

- StyleSheet - A StyleSheet is an abstraction similar to CSS StyleSheets.
- StatusBar - Component to control the app's status bar. The status bar is the zone, typically at the top of the screen, that displays the current time, Wi-Fi and cellular network information, battery level and/or other status icons.

From Firebase

- Firebase Auth - Used for authenticating users within the Firebase database.
- Firebase Storage - A local storage used to store images from scanning music sheets.
- ref - Used to reference and image to a user.
- getDownloadUrl - Get download URL for a file from Firebase.

Other Imports

- Audiveris from api - Audiveris is a software used to pull the notes from a music sheet.
- lilypond from api - Lilypond allows for displaying a music sheet in a manner that suits are needs.

#### 5.4.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

#### 5.4.5 Subsystem Data Structures

An array of notes will be delivered to this subsystem from the audiveris api. It will also receive a png of the music sheet to be ready for viewing.

#### 5.4.6 Subsystem Data Processing

No note worthy algorithms. This system will display a page before the user enters the tracker so they may make any last minute changes. If the scanner was not able to find the tempo for the current song then the user may adjust it manually. This will change the timing between notes on the page during the play mode.

### 5.5 Page Turner/Scroller/Tracker

This subsystem is in charge of tracking where the user is in the current song and determining when it needs to turn the page of the music sheet. It will be a screen within our react native app that will use and import other components in order to be operational.

#### 5.5.1 Subsystem Hardware

No hardware component.

#### 5.5.2 Subsystem Operating System

This subsystem will require the use of the Linux OS as that is the standard OS for android.

#### 5.5.3 Subsystem Software Dependencies

- useState - allows for updating a variable on screen when something changes.
- useEffect - he useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.

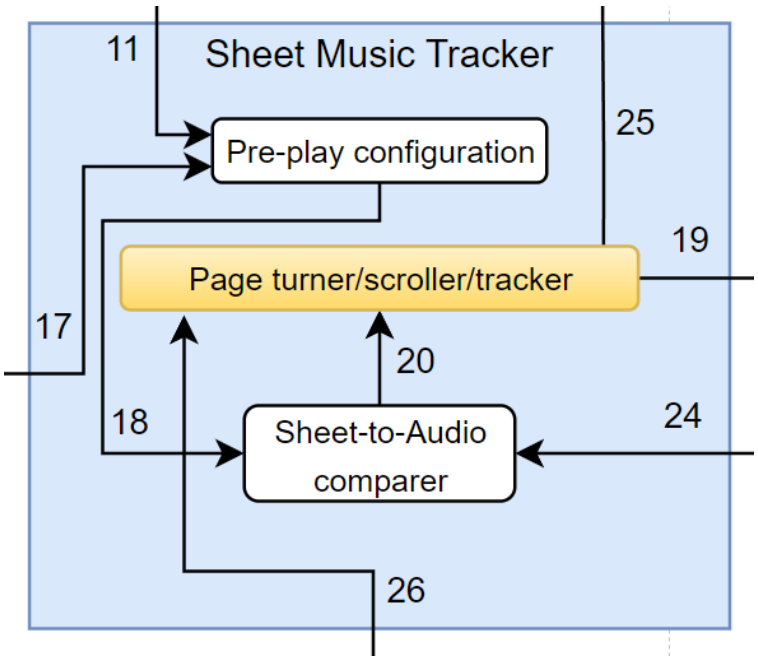


Figure 11: Pre-Play Configuration diagram

- useRef - A React Hook that lets you reference a value thatâs not needed for rendering.

From React Native

- View - A container that supports layout with flexbox, style, some touch handling, and accessibility controls. View maps directly to the native view equivalent on whatever platform React Native is running on, whether that is a UIView, , android.view, etc.
- ScrollView - A generic scrolling container that can contain multiple components and views. The scrollable items can be heterogeneous, and you can scroll both vertically and horizontally (by setting the horizontal property).
- Modal - A basic way to present content above an enclosing view. platform. Supports a minimal level of customization.
- Animated - The Animated library is designed to make animations fluid, powerful, and painless to build and maintain. Animated focuses on declarative relationships between inputs and outputs, configurable transforms in between, and start/stop methods to control time-based animation execution.
- Easing - The Easing module implements common easing functions. This module is used by Animated.timing() to convey physically believable motion in animations.
- TouchableOpacity - A wrapper for making views respond properly to touches. On press down, the opacity of the wrapped view is decreased, dimming it.
- Text - A React component for displaying text.
- TextInput - A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad.

- StyleSheet - A StyleSheet is an abstraction similar to CSS StyleSheets.
- StatusBar - Component to control the app's status bar. The status bar is the zone, typically at the top of the screen, that displays the current time, Wi-Fi and cellular network information, battery level and/or other status icons.

### 5.5.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

### 5.5.5 Subsystem Data Structures

Will receive signal inputs from the Sheet-to-Audio class.

### 5.5.6 Subsystem Data Processing

No note worthy algorithms. This system will auto scroll to the next page when the user reaches the end of the current page. Once the final note for the current page has been evaluated to a json array of all the notes.The page will scroll down by a distance equal to the height of the music page.

## 5.6 Sheet-To-Audio Comparer

This subsystem uses audio to determine where the user is on the music sheet. It will be represented as a class using data from the scanning side and audio side to make the connection for the page turner subsystem.

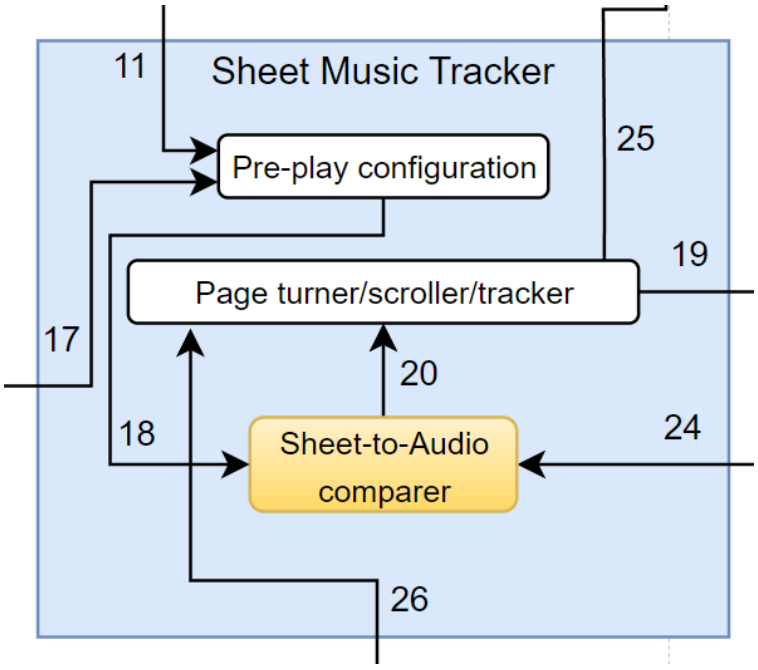


Figure 12: Pre-Play Configuration diagram

### 5.6.1 Subsystem Hardware

No hardware components-.

### 5.6.2 Subsystem Operating System

This subsystem will require the use of the Linux OS as that is the standard OS for android.

### 5.6.3 Subsystem Software Dependencies

- useState - allows for updating a variable on screen when something changes.
- useEffect - he useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.
- useRef - A React Hook that lets you reference a value thatâs not needed for rendering.

From React Native

- View - A container that supports layout with flexbox, style, some touch handling, and accessibility controls. View maps directly to the native view equivalent on whatever platform React Native is running on, whether that is a UIView, , android.view, etc.
- ScrollView - A generic scrolling container that can contain multiple components and views. The scrollable items can be heterogeneous, and you can scroll both vertically and horizontally (by setting the horizontal property).
- Modal - A basic way to present content above an enclosing view.
- Alert - Launches an alert dialog with the specified title and message.
- Button - A basic button component that should render nicely on any platform. Supports a minimal level of customization.
- Animated - The Animated library is designed to make animations fluid, powerful, and painless to build and maintain. Animated focuses on declarative relationships between inputs and outputs, configurable transforms in between, and start/stop methods to control time-based animation execution.
- Easing - The Easing module implements common easing functions. This module is used by Animated.timing() to convey physically believable motion in animations.
- TouchableOpacity - A wrapper for making views respond properly to touches. On press down, the opacity of the wrapped view is decreased, dimming it.
- Text - A React component for displaying text.
- TextInput - A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad.
- StyleSheet - A StyleSheet is an abstraction similar to CSS StyleSheets.
- Dimensions - Allows for obtaining an application width and height.
- StatusBar - Component to control the app's status bar. The status bar is the zone, typically at the top of the screen, that displays the current time, Wi-Fi and cellular network information, battery level and/or other status icons.

Other Imports

- Audio from expo-av - Allows you to implement audio playback and recording in your app.

- fft from mathjs - Calculate N-dimensional fourier transform
- entypo from @expo/vector-icons - An icon from expo vector icons
- Audiveris from api - Audiveris is a software used to pull the notes from a music sheet.
- lilypond from api - Lilypond allows for displaying a music sheet in a manner that suits are needs.

#### 5.6.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

#### 5.6.5 Subsystem Data Structures

Will take a png and an array of audio data to synchronise and make the connection for the page turner and scroller subsystem.

#### 5.6.6 Subsystem Data Processing

Fast Fourier Transform algorithm - An algorithm that transform and converts a function into a form that describes the frequencies present in the original function.

# 6 Sheet Music Scanner

Description of the hardware components and software components that make up the sheet music scanner layer.

## 6.1 Layer Hardware

Not applicable. No hardware components for this subsystem.

## 6.2 Layer Operating System

The program for this app will be running on an android tablet and as such will need a Linux operating system so it can run on an external server to be called via API POST requests.

## 6.3 Layer Software Dependencies

From React

- useState - allows for updating a variable on screen when something changes.
- useEffect - The useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.
- useRef - A React Hook that lets you reference a value thatâs not needed for rendering.

From React Native

- View - A container that supports layout with flexbox, style, some touch handling, and accessibility controls. View maps directly to the native view equivalent on whatever platform React Native is running on, whether that is a UIView, , android.view, etc.
- ScrollView - A generic scrolling container that can contain multiple components and views. The scrollable items can be heterogeneous, and you can scroll both vertically and horizontally (by setting the horizontal property).
- Modal - A basic way to present content above an enclosing view.
- Alert - Launches an alert dialog with the specified title and message.
- Button - A basic button component that should render nicely on any platform. Supports a minimal level of customization.
- Animated - The Animated library is designed to make animations fluid, powerful, and painless to build and maintain. Animated focuses on declarative relationships between inputs and outputs, configurable transforms in between, and start/stop methods to control time-based animation execution.
- Easing - The Easing module implements common easing functions. This module is used by Animated.timing() to convey physically believable motion in animations.
- TouchableOpacity - A wrapper for making views respond properly to touches. On press down, the opacity of the wrapped view is decreased, dimming it.
- Text - A React component for displaying text.

- TextInput - A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad.
- StyleSheet - A StyleSheet is an abstraction similar to CSS StyleSheets.
- Dimensions - Allows for obtaining an application width and height.
- StatusBar - Component to control the app's status bar. The status bar is the zone, typically at the top of the screen, that displays the current time, Wi-Fi and cellular network information, battery level and/or other status icons.

From Firebase

- Firebase Auth - Used for authenticating users within the Firebase database.
- Firebase Storage - A local storage used to store images from scanning music sheets.
- ref - Used to reference and image to a user.
- getDownloadUrl - Get download URL for a file from Firebase.

Other Imports

- expo-image-picker - an image picking component that open's the devices photo library to upload photos.
- xml2js - Takes in an XML file and generates an equivalent JSON object.
- Audiveris from api - Audiveris is a software used to pull the notes from a music sheet.
- lilypond from api - Lilypond allows for displaying a music sheet in a manner that suits are needs.

## 6.4 Image Import

The Image Import prompts the user to upload an image to use with the rest of the Sheet Scanner system. The Image Import will support all basic image formats (jpg, png, etc.). This system is used as a sort of confirmation of successful input from both the user visually and the system checking that the file type is valid.

### 6.4.1 Subsystem Hardware

No hardware component.

### 6.4.2 Subsystem Operating System

This subsystem will require the use of the Linux OS as that is the standard OS for android.

### 6.4.3 Subsystem Software Dependencies

From React

- useState - allows for updating a variable on screen when something changes.
- useEffect - he useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.

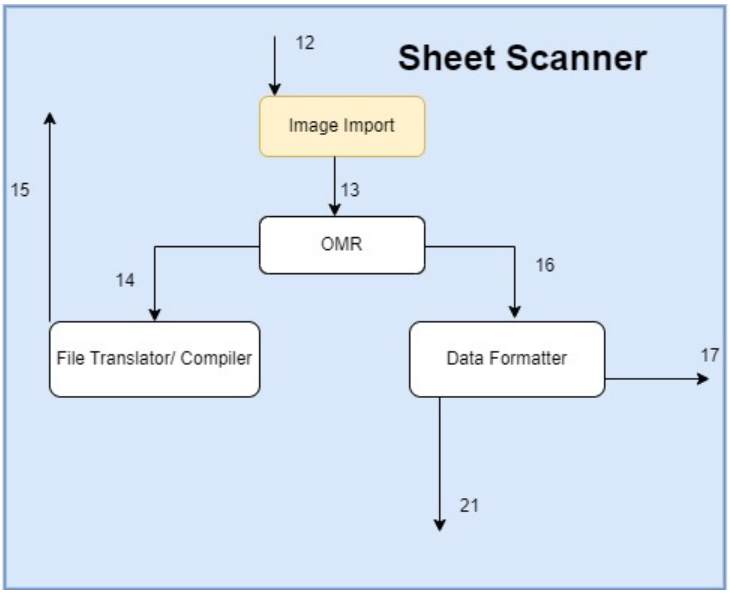


Figure 13: Image Import diagram

- useRef - A React Hook that lets you reference a value thatâs not needed for rendering.

From React Native

- View - A container that supports layout with flexbox, style, some touch handling, and accessibility controls. View maps directly to the native view equivalent on whatever platform React Native is running on, whether that is a UIView, , android.view, etc.
- ScrollView - A generic scrolling container that can contain multiple components and views. The scrollable items can be heterogeneous, and you can scroll both vertically and horizontally (by setting the horizontal property).
- Modal - A basic way to present content above an enclosing view. platform. Supports a minimal level of customization.
- Animated - The Animated library is designed to make animations fluid, powerful, and painless to build and maintain. Animated focuses on declarative relationships between inputs and outputs, configurable transforms in between, and start/stop methods to control time-based animation execution.
- Easing - The Easing module implements common easing functions. This module is used by Animated.timing() to convey physically believable motion in animations.
- TouchableOpacity - A wrapper for making views respond properly to touches. On press down, the opacity of the wrapped view is decreased, dimming it.
- Text - A React component for displaying text.
- TextInput - A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad.

- StyleSheet - A StyleSheet is an abstraction similar to CSS StyleSheets.
- StatusBar - Component to control the app's status bar. The status bar is the zone, typically at the top of the screen, that displays the current time, Wi-Fi and cellular network information, battery level and/or other status icons.

Other Imports

- expo-image-picker - an image picking component that open's the devices photo library to upload photos.

### 6.4.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

### 6.4.5 Subsystem Data Structures

N/A

### 6.4.6 Subsystem Data Processing

No note worthy algorithms. The subsystem takes an image file from the user storage and converts it into an image URI. This URI will then be displayed to the user as preview image.

## 6.5 OMR

This subsystem is in charge of taking the user's image input and transforming it into a digital copy of itself as well as relevant note attribute data. This subsystem is an API that is called with data compiled from the overarching system information (user ID) and the information provided in Image Import. The subsystem results in 2 musicXML files: 1 from the scan on the user's input file and 1 from the scan on the resulting digital copy later on in the system.

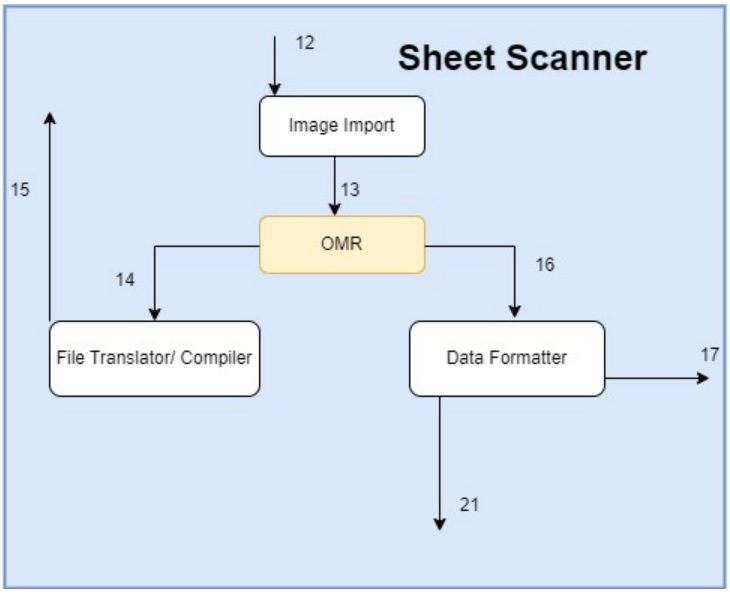


Figure 14: OMR diagram

### 6.5.1 Subsystem Hardware

No hardware component.

### 6.5.2 Subsystem Operating System

This subsystem will require the use of the Linux OS as that is the standard OS for android. This subsystem will also require a separate server environment in order to host the API.

### 6.5.3 Subsystem Software Dependencies

- useState - allows for updating a variable on screen when something changes.
- useEffect - he useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.
- useRef - A React Hook that lets you reference a value thatâs not needed for rendering.

Other Imports

- Audiveris from api - Audiveris is a software used to pull the notes from a music sheet. It scans the given image and transforms the data into a musicXML file.
- getDownloadURL from firebase/storage - Firebase storage is used to store the digital copy when the processing is done

### 6.5.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

### 6.5.5 Subsystem Data Structures

The processing in this subsystem will result in a musicXML file that represents all detected music sheet symbols on the page.

### 6.5.6 Subsystem Data Processing

In order to acquire the relevant note position of the newly generated digital copy of the music sheet, we need to run the Audiveris process twice in order to get the correct figures.

## 6.6 File Translator/Compiler

This subsystem uses the generated musicXML file to parse all relevant data for the lilypond input file. After parsing and writing the relevant lilypond file, it then runs the lilypond executable file via batch command in order to produce the digital copy.

### 6.6.1 Subsystem Hardware

No hardware components-.

### 6.6.2 Subsystem Operating System

This subsystem will require the use of the Linux OS as that is the standard OS for android.

### 6.6.3 Subsystem Software Dependencies

- useState - allows for updating a variable on screen when something changes.
- useEffect - he useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.
- useRef - A React Hook that lets you reference a value thatâs not needed for rendering.
- lilypond from api - Lilypond allows for displaying a music sheet in a manner that suits are needs.

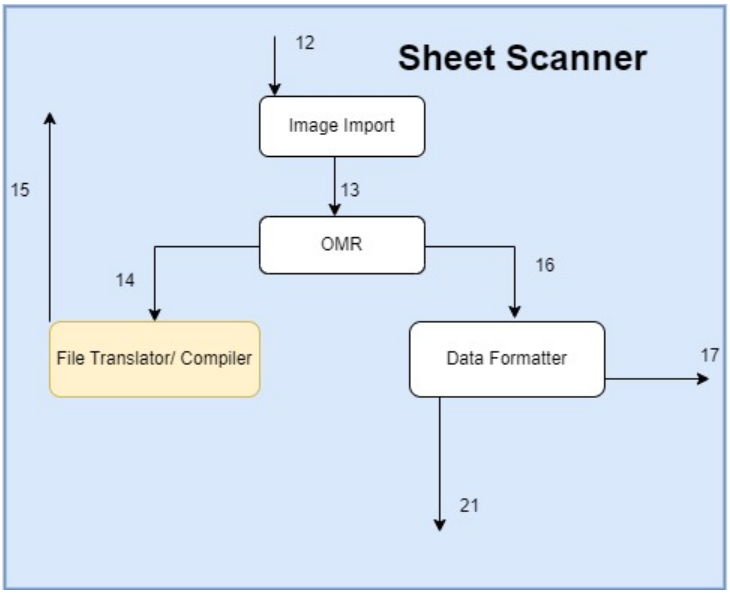


Figure 15: File Translator/Compiler diagram

### 6.6.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

### 6.6.5 Subsystem Data Structures

N/A

### 6.6.6 Subsystem Data Processing

This subsystem will parse the given musicXML from the first Audiveris scan of the user input image and translate it into a readable lilypond file for batch execution.

## 6.7 Data Formatter

The subsystem will gather parsed data from both muxicXML files (from the 2 scans, one on the user copy, the other on the digital copy) and compile it into a JSON object that can then be used for the tracker system.

### 6.7.1 Subsystem Hardware

No hardware components-.

### 6.7.2 Subsystem Operating System

This subsystem will require the use of the Linux OS as that is the standard OS for android.

### 6.7.3 Subsystem Software Dependencies

- useState - allows for updating a variable on screen when something changes.
- useEffect - he useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.
- useRef - A React Hook that lets you reference a value thatâs not needed for rendering.
- xml2js - Takes in an XML file and generates an equivalent JSON object.

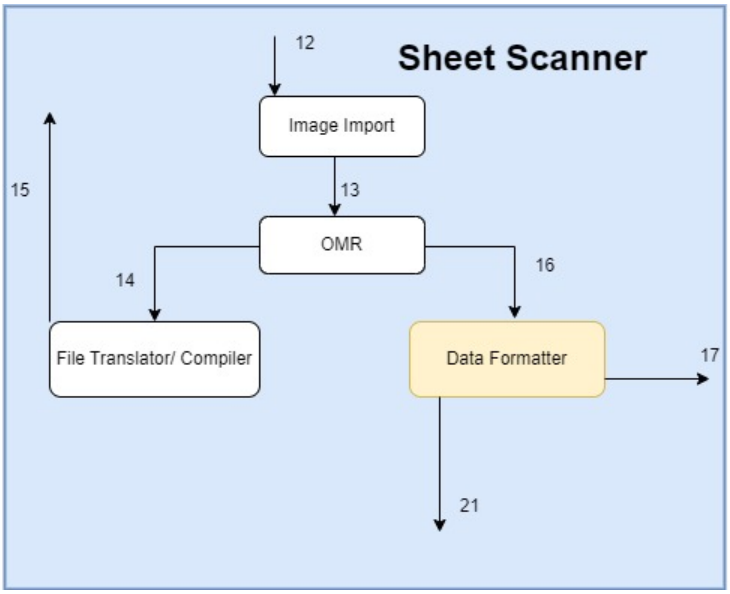


Figure 16: Data Formatter diagram

### 6.7.4 Subsystem Programming Languages

The subsystem will be written in JavaScript.

### 6.7.5 Subsystem Data Structures

The output JSON object

### 6.7.6 Subsystem Data Processing

The subsystem will utilize xml2js and then immediately parse the result with only relevant data that we need for the tracker.

# 7 Note Recognition

Description of the hardware components and software components that make up the note recognition layer.

## 7.1 Layer Hardware

Not applicable. No Hardware components for this subsystem.

## 7.2 Layer Operating System

The program for this app will be running on an android tablet and as such will need a Linux operating system so it can run.

## 7.3 Layer Software Dependencies

From React

- useState - allows for updating a variable when something changes.
- useEffect - The useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, and playing the audio file. useEffect accepts two arguments. The second argument is optional.

From expo-av

- Audio - Used for getting the raw audio from the microphones in the Android tablet.

From mathjs

- fft - used for applying a Fourier transform to the raw audio data to receive the amplitudes of the raw audio data.

## 7.4 Audio Recorder

The audio recorder subsystem allows for the recording of the instrument being played by accessing the microphone and creating a .wav file.

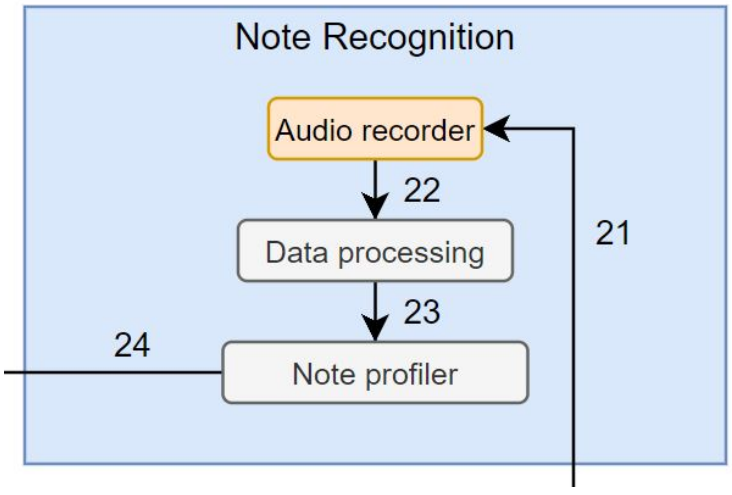


Figure 17: Audio Recorder Diagram

### 7.4.1 Subsystem Hardware

Not applicable. No Hardware components for this subsystem.

### 7.4.2 Subsystem Operating System

The program for this app will be running on an android tablet and as such will need a Linux operating system so it can run.

### 7.4.3 Subsystem Software Dependencies

- useState - allows for updating the recording states, and updating the .wav file when new audio is received.
- Audio - Used for getting the raw audio from the microphones in the Android tablet and creating a .wav file.

### 7.4.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

### 7.4.5 Subsystem Data Structures

Will create an audio .wav file that will be used for processing.

### 7.4.6 Subsystem Data Processing

No note worthy algorithms. This system will take a recording and process it into a format that can be delivered to the data processing subsystem.

## 7.5 Data Processing

The data Processing receives an audio .wav file that will be processed by a Fourier transform to receive the amplitudes at certain frequencies. This data is stored in an constant variable array.

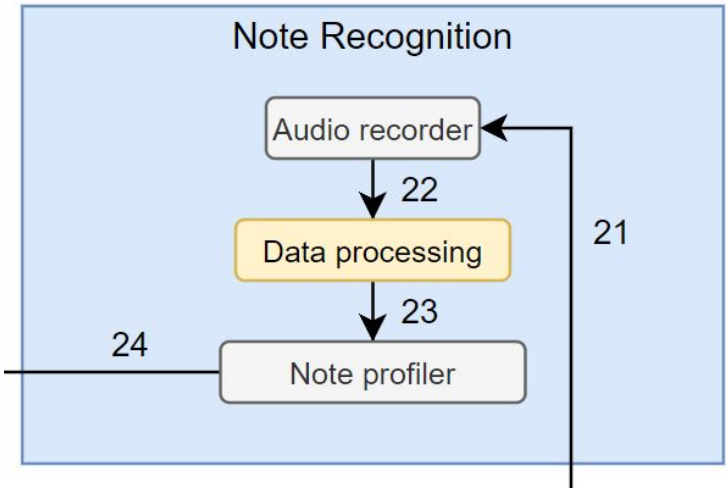


Figure 18: Data Processing Diagram

### 7.5.1 Subsystem Hardware

Not applicable. No Hardware components for this subsystem.

### 7.5.2 Subsystem Operating System

The program for this app will be running on an android tablet and as such will need a Linux operating system so it can run.

### 7.5.3 Subsystem Software Dependencies

- Audio - Used for getting the raw audio from the microphones in the Android tablet and creating a .wav file.
- fft - Used to get the amplitudes at certain frequencies.

### 7.5.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

### 7.5.5 Subsystem Data Structures

The .wav file is processed with the fft library to get an array of amplitudes at certain frequencies.

### 7.5.6 Subsystem Data Processing

Fast-Fourier-Transform for getting frequencies and amplitudes of played notes.

## 7.6 Note Profiler

The note profiler subsystem uses the constant variable array of amplitudes at certain frequencies to identify the notes that were played. These notes will be stored in a constant variable array.

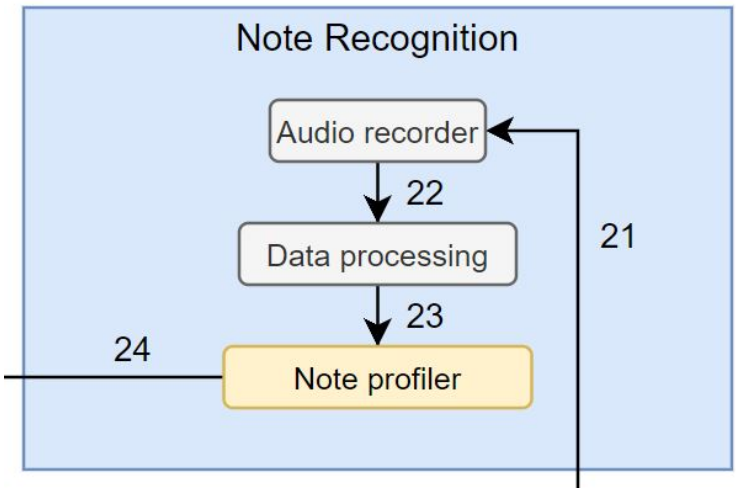


Figure 19: Note Profiler Diagram

### 7.6.1 Subsystem Hardware

Not applicable. No Hardware components for this subsystem.

### 7.6.2 Subsystem Operating System

The program for this app will be running on an android tablet and as such will need a Linux operating system so it can run.

### 7.6.3 Subsystem Software Dependencies

useState - allows for updating the notes array.

### 7.6.4 Subsystem Programming Languages

The subsystem will be using react native code which is written in JavaScript.

### 7.6.5 Subsystem Data Structures

The constant variable array of notes being played.

### 7.6.6 Subsystem Data Processing

This system loops through the frequencies and compares the max frequency to a list of frequencies corresponding to notes. If the frequency is within a certain range. The note is identified and placed in an array.

# 8 Appendix A

# References