LLM-Driven Guidance for Piano Sight Reading

Undergraduate Honours Thesis
Computer Science, Faculty of Science
Ontario Tech University
Oshawa, ON, Canada

Keeran Srikugan

Supervisors: Mariana Shimabukuro & Dr. Christopher Collins

April 22, 2025

Abstract

This thesis looks at an application that helps deal with the challenge that newcomers and experienced pianists experience when learning how to sight-read. The Piano Sight Reading Application is a web-based application that leverages both Large Language Models (LLMs) and MIDI inputs to assess, analyze and improve the user's sight reading capabilities. The framework used to make the web application is React, and it is developed with Web MIDI API and OpenAI’s o3-mini to analyze user performance and categorize errors into groups. This allows the system to identify and take note of the users' weakest areas. With the help of OpenAI’s 4o-mini, personalized exercises are generated dynamically for the most frequent error categories, creating an adaptive system that will change as the user grows and develops their skill. This paper will showcase the potential of utilizing LLMs to create adaptive learning tools that can support the education of music in a new and individualized way.

Contents

Chapter 1: Introduction

Over the years, the emergence of new technologies and advancements across various fields has significantly impacted how people learn. This holds true in the realm of learning musical instruments, particularly the piano. One notable example of technology-enhanced learning is *PianoVision*, a Virtual Reality/Augmented Reality application designed for the Meta Quest headset, which allows users to learn piano pieces through visual cues that work in conjunction with the user's instrument. However, most of these applications found today, like *PianoVision,* focus primarily on learning songs, and one often overlooked aspect of learning how to play the piano is the development of sight-reading skills (reading music sheets). Sight-reading is an area of learning piano that is typically seen as a monotonous aspect of learning the piano, and most applications today teach users how to simply play songs on the instrument instead.

This thesis explores how Large Language Models (LLMs), specifically the OpenAI APIs, can be leveraged to support the development of sight-reading skills in an adaptive and personalized manner. The proposed application begins by assessing the user's current sight-reading abilities through a structured evaluation. Once the assessment is complete, GPT analyzes the results to identify the user's most common errors and provides tailored feedback and improvement strategies. Based on this analysis, GPT further recommends targeted exercises designed to address the user's specific weaknesses in sight-reading. This adaptive and personalized approach offers a novel method for enhancing sight-reading skills, filling a critical gap in existing piano learning technologies.

1.1 - Motivation

Learning and playing instruments has always been a core aspect of my life, and I have always enjoyed working on my musical skills. However, as someone who likes to self-teach using resources available online and physically, I found one of the most challenging aspects of learning how to play is sight reading. While there are books explaining music theory and teaching people how to sight read, it can be quite difficult to grasp and work on sight reading due to its inherently mundane aspects and lack of variety in methods of learning how to sight read. Most applications that can be found today will rather focus on simply teaching how to play specific songs on the piano, but gloss over the other skill sets of learning music theory and sight reading. An analogy for this can be teaching someone how to speak a language, but still lacking the ability to read or write in that language.

1.2 - Target Audience

The target audience for this application, Sight Reading Assistant, is people who are learning how to play the piano and specifically learning how to sight-read sheet music. This application is not meant to be used as a replacement for an instructor, but rather to be used as a supplement to help learn and improve skills. This application is not for complete beginners as it doesn't aid with the basics of sight reading, such as what letter each note corresponds to or what any of the symbols mean, and so on. The ideal stakeholders for this project are students who are learning how to sight-read under the guidance of a professional. It can also be used by anyone who is self-teaching and would like to have an application that will test and help them improve their sight-reading abilities.

Chapter 2: Related works

2.1 - Programs that do not use Large Language Model

2.1.1 - Piano Vision

During the research portion of my thesis, a variety of different applications were explored that focus on the idea of helping students learn how to play the piano with different pieces of technology. One application that took an interesting approach to teaching users how to play the piano is called PianoVision. This application takes advantage of utilizing Augmented Reality (AR) to help users learn how to play the piano. But the primary focus of this application is to help users learn how to play songs as opposed to teaching users music theory and sight reading.

One of the main advantages of this application is that it makes it stand out from other applications and methods is its creative use of the AR aspect of teaching how to play the piano. This application will show users falling keys through the VR/AR headset to visually show the user which keys they must play as shown in Figure 1. Since this was done using Mixed Reality technology, the users were able to use their live instruments to learn their music as opposed to needing to interact with a virtual piano to learn. This allows the user to comfortably play the piano naturally without the application, which is a very interesting concept for learning. Along with the falling keys, the sheet music is also shown in front of the user to follow along. And the falling keys that are being shown to the user are highlighted on the sheet, which helps the user follow along with the sheet music and also learn what each note represents on the piano as they play along. To give an analogy, this would be akin to watching a movie and listening to the words, and reading the subtitles to follow along. This is one of the tactics employed by the application to help the user learn how to play the piano. Once the user is finished with learning a piece of sheet music, they will be presented with statistics on how well they played, such as the number of incorrect notes played and their accuracy, which helps the user get a visual representation of their progress.

Some of these components were similarly used with the Piano Sight Reading Application to help the user learn and develop their sight reading skills. The aspect of visually representing the user's progress was also utilized within the Piano Sight Reading Application when the user completes an assessment, their errors are visually shown in a graphic format that is easier to understand, and they can visually see their areas. Along with this, the Piano Sight Reading Application has built-in exercises similar to the PianoVision, where these exercises are much more focused on the aspect of sight reading than just playing. Both of these applications depend on the usage of an electric piano to work at the highest accuracy and precision possible. These small aspects used by PianoVision are very effective in interacting with the user, and I have decided to consider these aspects while creating this application.

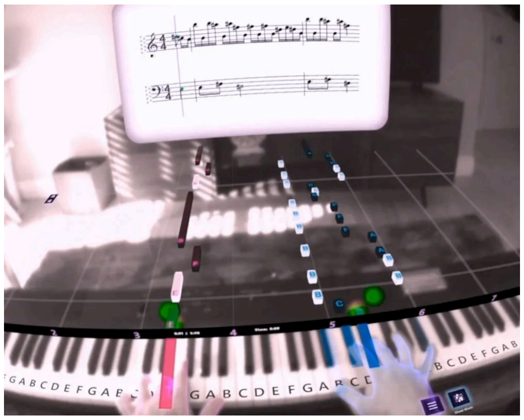


Figure 1: This image shows a screenshot of a user utilizing Piano Vision to learn how to play the sheet music shown [5].

2.1.2 - Learn Piano with BACh

Learn Piano with BACh is an interesting paper that looks at the idea of monitoring the musician's cognitive workload and uses it to help adjust the difficulty of musical exercises in real-time. What makes this paper stand out to me is the aspect of adjusting the exercise difficulty to match the user's playing ability, which shows the idea of adaptive learning that is the primary focus of the thesis.

To break down how this works, let's start by taking a look at how this paper measures the cognitive load. This is done by utilizing an FNIRS (functional near-infrared spectroscopy) device, which is an imaging technique that records the levels of activation in the player's brain, which can be seen in Figure 2 [1]. By being able to monitor the stress and cognitive workload of the users brain, the BACh program and algorithm would be able to identify if the user is on an appropriate level of sheet music, or if the piece is too difficult meaning it needs to be simpler, or finally if its too easier and needs to be more challenging [1]. The purpose of this application is to keep adjusting the difficulty level of the sheet music to keep the user's attention. For example, if a piece of sheet music/exercise is too difficult, then the user will become dejected and give up on the exercise, but with this application, the user will be able to stay focused if the difficulty level is matched to what they are comfortable with.

Reading this paper helped identify a new way to keep the user's attention and engage them through adaptive real-time changes. But while both this paper and my thesis have a similar goal of looking at adaptive exercises, the method of approaching and tackling this problem is fundamentally different. The BACh application monitors users' cognitive load and brain activity to adjust the difficulty of the exercises. But Piano Sight Reading Learning Assistant harnesses and uses a Large Language Model to analyze and adjust exercises to help the user work on their primary weaknesses. Not only does this application adjust the exercise based on the user's current playing capabilities, but it also suggests a variety of exercises to target a user's most lacking skill to overall enhance the user's sight-reading capabilities.

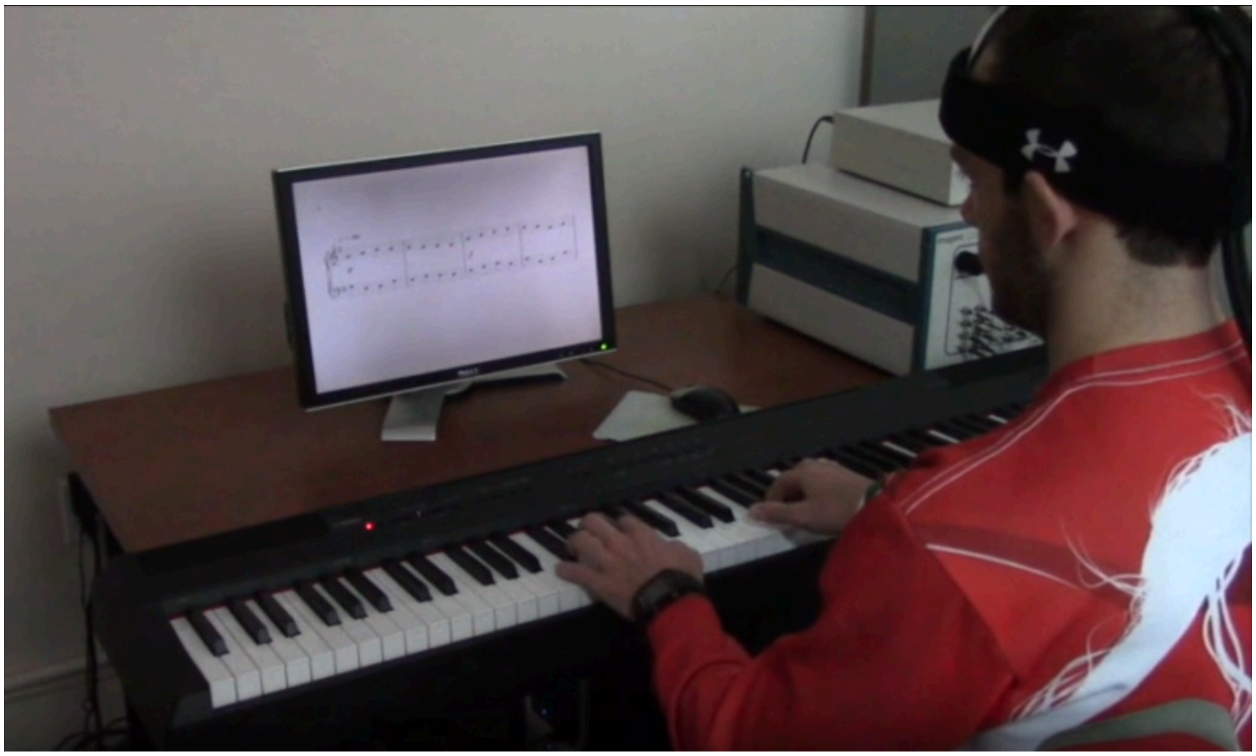


Figure 2: This figure shows the image of a user using Learn Piano with BACh while having their cognitive workload recorded using an FNIRS device, and the sheet music adjusting depending on the recorded workload values [1].

2.1.3 - Flowkey

Flowkey is a web-based application that provides the service of teaching users how to play the piano using a few methods. This application works in cooperation with Yamaha and can be utilized through either a MIDI input or with the use of a microphone if the user does not have access to an electric piano. How it tackles this method is by having two separate parts to teach the user, where the first part you can learn how to play specific songs, and there is also a course option as well for the user to follow along with. All of this is provided to the user for a monthly subscription to gain access to all of these functionalities.

The functionality of teaching the user how to play specific songs is, in ways, quite similar to how PianoVision approaches teaching users new songs. While PianoVision uses falling keys and takes advantage of the mixed reality component, Flowkey simply has the notes displayed on the screen for the user to follow along as it is highlighted. They claim to have over a thousand songs to choose from, and the different functionalities available while learning a sheet of music include wait mode, video and sheet music, slow motion, loop function, and select a hand. Wait mode will wait for the user to press the correct key to then progress with the song., Video and sheet music will show the user an "expert" playing the sheet music from a top-down point of view, and this can be seen in Figure 3. Slow motion will reduce the tempo and speed of the song to make it easier for the user to follow along and adjust to playing the song. The loop function will allow the user to select a section of the music to loop over until they are comfortable playing that section. Finally, Select a Hand gives the user the ability to isolate the hand they were to practice with specifically. Other than learning how to play these songs, the other feature provided with a subscription is giving the user modules to learn with. These modules/courses are personalized videos that go over all the aspects of playing the piano, from a very beginner aspect of understanding how the keys of a piano are broken down to learning intermediate skills.

While this application does help the user learn how to play the piano and develop some sight-reading skills through the provided features, it does not, however, provide any in-depth and personal forms of learning that the Piano Sight Reading Assistant does. While there is some personalization with learning songs, such as being able to slow down the song to match the user's pace, the piano sight reading application looks to adaptively teach users how to sight read sheet music by completely customizing and giving users individualized activities each time. This level of personalization can not be seen by the Flowkey web application.

Figure 3: The following figure shows an image of a pianist utilizing Flowkey on their iPad to follow along with tutorials to learn how to play the sheet music and song [6].

2.2 - Programs that use Large Language Models

2.2.1 - LLMS for personalized learning

The paper written by Bhutoria, called "Personalized education and Artificial Intelligence in the United States, China, and India: A systematic review using a Human-In-The-Loop model," is a paper where the aspects of AI in education are explored in depth [3]. The paper looks at the benefits of utilizing AI in the education system and all the different methods that AI can implement to teach.

Some of the key points of this paper include exploring the aspect of adaptive learning paths, where the AI system will fully utilize all information it has with the learner, such as their habits and any preferences, to personalize how they learn. Another method of how AI can be integrated into learning that has been explored is looking at how it can introduce the aspect of adaptive learning paths. This is the idea that is implemented in the Piano Sight Reading Application, where the idea of adaptively changing activities to suit the current user's playing level is shown. One interesting application of AI in education is the predictive analytics aspect, which is taking a look at analyzing how the user learns and also predicting any roadblocks or issues with learning that need to be addressed [3].

This paper predominantly looks at the theoretical implications of AI in student learning, but the Piano Sight Reading application takes some of these theories and puts them into practice. The specific implementation is the use of adaptive learning in the exercises. The webpage utilizes GPT o1-mini to identify what skills the user needs to focus on developing and customizes the activity to target the weak points of the user. This implementation of AI follows the ideology and theory explained in the paper, but emphasis needs to be placed on the challenge that AI will not be able to motivate and engage learners as well as real people, and this is something that cannot be solved. That is the exact reason why this application was created to be used as a tool that teachers can use to help students learn how to sight read, and why the purpose is specifically not meant to be a replacement for teachers.

Chapter 3: Piano Sight Reading Design

Piano Sight Reading Application Assistant is a learning application that is designed to help users learn and practice their sight reading skills through assessments and exercises. This application focuses on assessing the user's current sight reading capabilities, and with the user or Large Language Model (LLM), exercises are assigned to the user in which each exercise will be customized to target the user's primary weakness. This approach of learning how to sight read is meant to help keep the user's attention while learning, while taking a look at a new approach of adaptive learning to keep up with the user's reading capabilities.

3.1 - Sight Reading

Sight reading is the ability to read and play sheets of music fluently, similar to what it is like to read a book in your native language. You should effortlessly be able to read and perform the sheet of music once seen. To train this skill, it takes an incredible amount of practice and skill. To understand how this program identifies different errors and aspects of sight reading that the user can struggle with, some background knowledge of sheet music is required. It is important to stress that there are many more aspects of playing the piano that can be observed and checked, but this current iteration of the application only accounts for four different groups of errors.

Before the different types of errors are discussed, let's take a quick look at the breakdown of the basic aspects of the piano needed to be able to follow along with the errors. The key phrasing for piano keys goes in this order: "C, (C#/D♭), D, (D#/E♭), E, F, (F#/G♭), G, (G#/A♭) A, (A#/B♭), B" and it will repeat until the last key on the piano. The layout for visual purposes can be seen in Figure 4, which shows one full octave on the piano. Every time the pattern is repeated on the piano, we call that an "octave", which indicates the level of pitch of the specific note that is being played. For example, a low-key octave C will have a much deeper and lower sound compared to a higher octave C. Depending on the size of the piano, the number of keys can differ depending on the type of electric piano. This application was created with the use of a full-length 88-key piano, but some pianos can be 49 or even 61 keys in length [11]. Along with the individual notes on the sheet music, there is something called the key signature, which tells the user what key the song will be in. For example, if a song is in an E-flat key signature, it means that every E and B key must be played as an E flat or B flat [11].

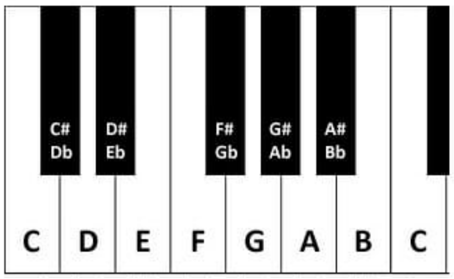


Figure 4: This figure shows all the keys that would be found in one scale. These 12 notes repeat over time, and each time this pattern is repeated on a piano, it is a new octave [7].

Now, taking a look at the first error, it is identifying incorrect **single note** errors. What this aspect of mistakes while playing GPT looks for are the instances of when the user has simply misread the key that is being played. For example, if the D key was expected to be pressed, but an E key was pressed instead, the error would be picked up as an issue with the ability to read the correct notes, and that will be noted down later for analysis.

The second error the system checks for is errors playing **chords**. Chords are a group of 3 or more notes that are played with one hand in unison on the piano [12]. This is an important error to check for, because chords can be quite difficult to play due to needing to read multiple keys at once to play a chord, making it more challenging than issues with the single note errors. An example of a chord error can be expected notes being [C, E, G] being played in unison, but the user plays [C, E] and then G after, meaning that the keys on the chord were not played properly.

The third error occurs when the user is having issues with identifying and keeping in mind the **key signature** of the sheet of music. The key signature indicates the scale that a piece of music is written in, and the specific notes that are sharps and flats in that scale [13]. For example, if a sheet of music is in the G major scale, it means that every regular/natural F key on the sheet is an F sharp unless specified otherwise (with a natural sign "♮" beside the note). The reason why this is tricky is that on the sheet of music, the notes will not have the sharp or flat symbols beside them if they are a sharp or flat in the key signature. Therefore it is easy to forget that you are playing in a specific key signature and play the natural key instead of the sharp or flat key. The system looks for any natural keys that are played, and if the error constantly keeps recurring, where the flat or sharp is missed, it is flagged as a key signature error.

The fourth and final error that is checked for is errors in regards to playing the correct **octave**. As explained earlier, the phrasing of the piano keys goes in a certain order, and every time the phrasing is repeated, that is called a new octave. This means that there are multiple instances of the same notes just repeated over and over on the piano. What this system looks for is when the user can identify the note correctly, but they play the note either too high or too low. If this occurs, then it will be classified as an octave error.

3.2 - Application Design

3.2.1 - Equipment Setup and Initialization:

This application is created with React as a web-based application, which runs on any device, such as an iPad or laptop, connected to their electronic piano. It is important to note that for the application to function, users will need an electric piano that has a USB port and a connection to their device of choice to be able to read the MIDI inputs of the piano as the user plays. For the back-end portion, what is utilized is prompt engineering from OpenAI, specifically the o3-mini model, where information is passed into and retrieved from an API call.

Once the piano is successfully connected to the device, the user is introduced to the application with a main menu screen, where the user has three options to select from.

1. Assessment: This section directs the user to an assessment where an evaluation of the user's sight-reading capabilities is conducted and where OpenAI is used to analyze the user's skills.

2. Exercises: This page leads the user to an area where they can select from a multitude of different exercises that the user can access and complete to work on specific areas of sight reading.

3. About: This section describes what the Piano Sight Reading Application Assistant is and its purpose for effective usage.

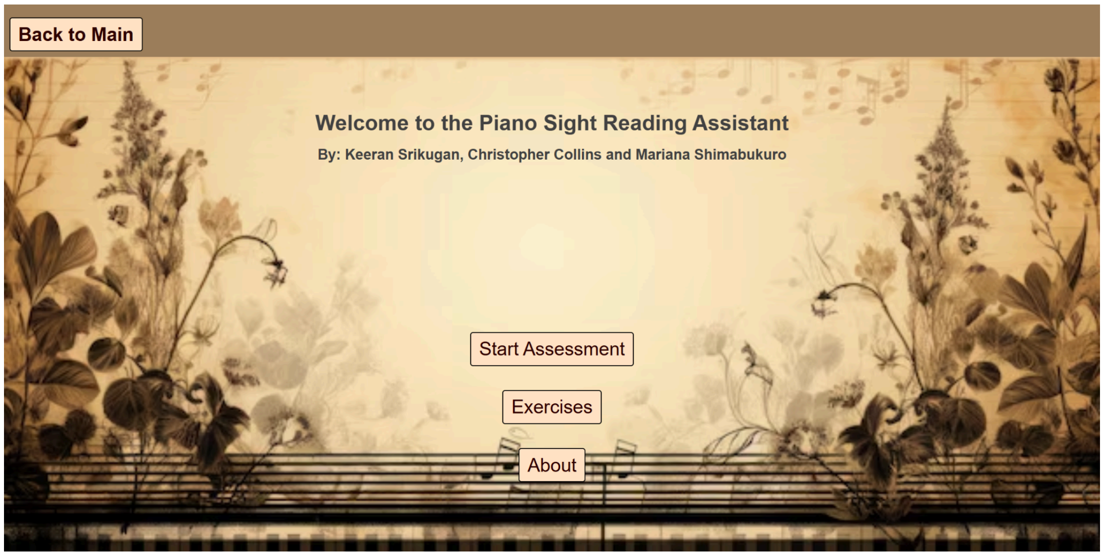
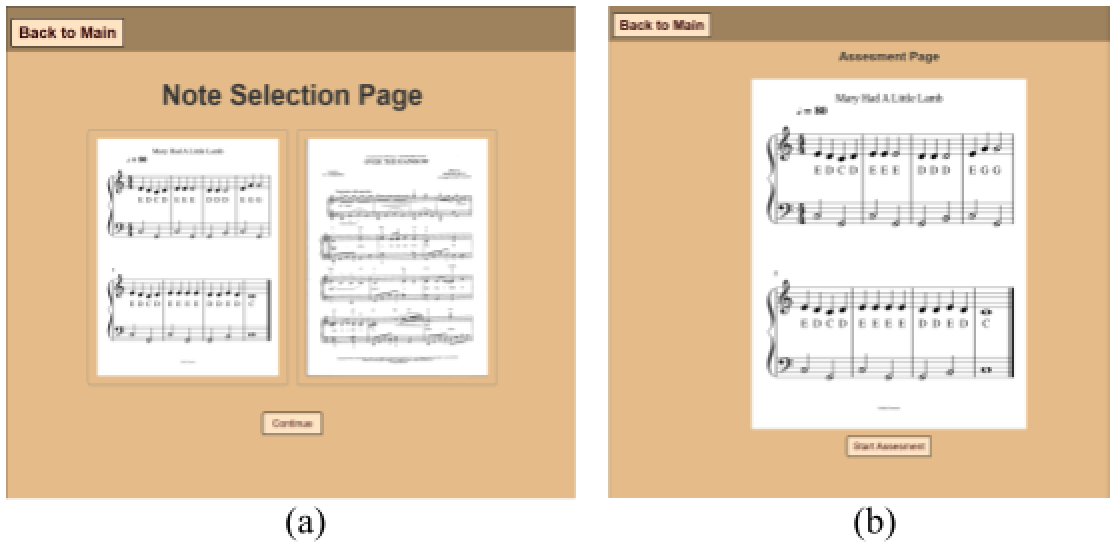

Back to Main
Welcome to the Piano Sight Reading Assistant
By: Keeran Srikugan, Christopher Collins and Mariana Shimabukuro
Start Assessment
Exercises
About

Figure 5: Screenshot of the main menu page for the application. Here, all the buttons leading to the assessment, exercises and about page can be found.

3.2.2 Assessment

When the user decides to complete an assessment of their current playing capabilities, they are introduced to a section where they are prompted to select a sheet of music to sight read. There will be a variety of different sheets of music in a list form for the user to choose from. This layout can be seen in Figure 6 (a) with the different sheets of music. Once the desired sheet is selected, the user will then continue and have the sheet of music on full display on the screen, with a start button at the bottom of the sheet. When users are ready to begin playing, they will press the button and begin sight reading and playing what they are able to read from the sheet music, as seen in Figure 6 (b). As the user is playing while reading the sheet of music, the inputs will be read into the application and saved. Once the user has completed playing the sheet of music, they will press the “End assessment” button, which will stop the inputs from being read by the application. Immediately after, the information will be processed into a text format to be used later on.



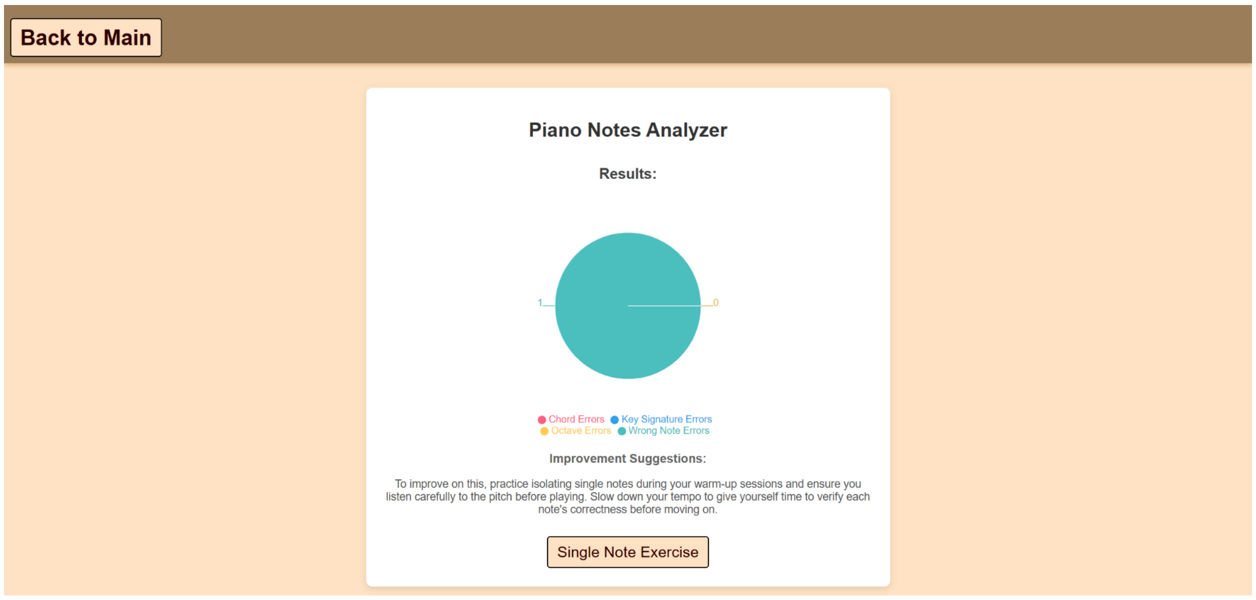
Figures 6 (a) and (b): Figure 6 (a) shows the Note Selection Page, which has a list of all the notes to choose for the assessment. Figure 6 (b) shows the assessment page where the user will start and complete the assessment.

3.2.3 Assessment Analysis:

Once the analysis page is reached, the user will be prompted with a button called “Analyze”. Once this button has been clicked, an analysis of how the user played based on the recording of the user and the expected notes from the sheet of music is conducted. This will invoke an API call to GPT o3-mini and create an analysis of the users' playing capabilities. Once the results have been returned, the information will be processed into a digestible form for the viewer.

To begin with, a pie chart will be shown, which will list all of the errors and the classification of errors that have been captured and identified from the users' play. The 4 classifications that will be seen in the colourful pie chart are Chord Errors, Key Signature Errors, Octave Errors, and Wrong Note Errors. This pie chart will visually show the number of errors for each category to the user, so that it is much easier to follow. Figure 7 shows the layout of the pie chart and how all the different categories are colour-coded, separated and clearly defined.

After the pie chart has been displayed, a section called Improvement Suggestions will be shown. This second will provide the user with 1-2 sentences on how they can improve on their most prominent error while playing. This will give some insight into what the user can do off the application to improve, as well as any useful insight to help tackle handling the specific type of error. And finally, based on the most frequent and prominent error, a button will be presented, which will lead the user to an exercise to specifically target the most recurring mistake the user makes.


Back to Main
Piano Notes Analyzer
Results:
1
0
● Chord Errors ● Key Signature Errors
● Octave Errors ● Wrong Note Errors
Single Note Exercise

Improvement Suggestions:

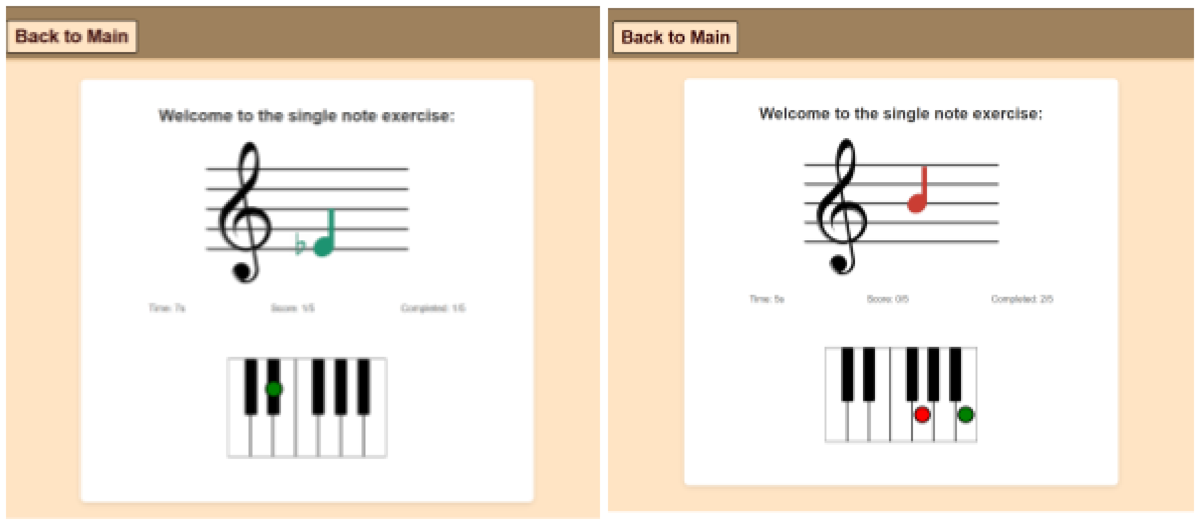
To improve on this, practice isolating single notes during your warm-up sessions and ensure you listen carefully to the pitch before playing. Slow down your tempo to give yourself time to verify each note's correctness before moving on.

***Figure** 7*: This figure shows a sample screenshot of the analysis page after the user has made one "Wrong Note Error".

3.2.4 Exercise Page:

Once the analysis is completed, the user will be directed to an exercise page, which will depend on the type of exercise the user needs to focus on. This exercise is a very simple one, where the user will be shown a set of notes and they will have to press the correct corresponding key on the piano. The notes that are shown will be dependent on two factors: the type of activity and how they played during the assessment. If the user hits the key correctly, the note will go green as shown in Figure 8 (a), which is one of the many indicators to tell the user that they were correct. If the user presses an incorrect key, that key will be drawn on the sheet music as a red note to show that they were incorrect and what note the user played on the piano, as opposed to what note was expected of them, as seen in Figure 8 (b).

To help the user visually, a layout of the piano is shown below, where red and green circles are drawn on the layout based on whether the key pressed was correct or incorrect. This component was added to the exercise to be used as another visual component that will clearly show the user their mistakes in an easy-to-understand and comprehensible format. These visual components are what make these exercises more engaging for the user as they practice. Along with these visual elements, a timer is added to the exercise as well to introduce a sense of urgency while playing. Sight reading is a skill that requires users to be able to read notes as quickly as possible; therefore, introducing a timer can help with adjusting to the difficulty of the exercise for the user. Both the timer, the score, and the number of completed exercises are shown textually to the user. Once the user has completed the exercise, the application will inform the user that the exercise has ended and to either return to the main menu or complete the exercise again with new notes.


Back to Main
Welcome to the single note exercise:
Back to Main
Welcome to the single note exercise:

(a) (b)

Figures 8 (a) and (b): Figure 8 (a) shows the exercise page when the note played by the user is correct. Figure 8 (b) shows when the note played by the user is incorrect.

3.3 System Architecture & Implementation:

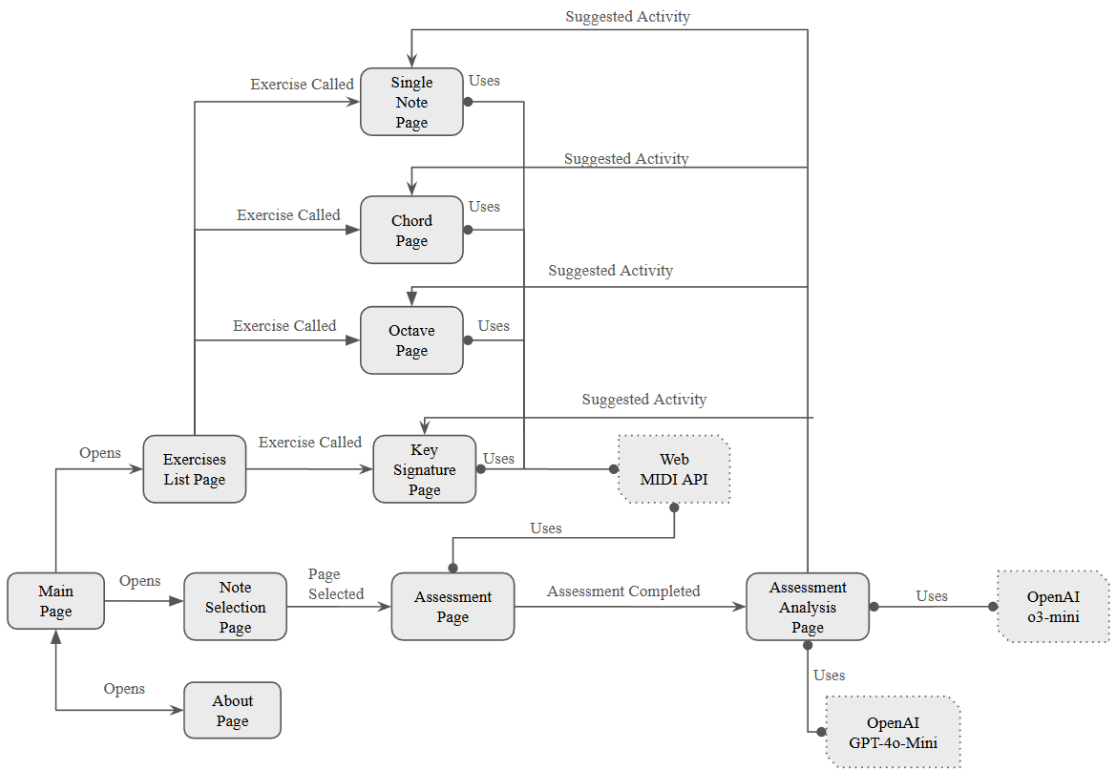


Figure 9: This figure shows the relationship between all the pages within the application and the external resources that are utilized in these pages to function.

3.3.1 - User Interface with React

For the creation of the user interface and front end of the application, the React Framework was utilized to create this application. To reiterate, React is a JavaScript library that is used to create both interactive and intuitive user interfaces for users.

It is also known as a component-based architecture, which allows for pages to be reused and easy to maintain since all important functionalities can be separated into their components. For example, the single note training component/file was referenced multiple times, to ensure that a user can simply do the training without needing a prior assessment and to allow for the training modules to be referenced after the assessment results, when deemed necessary by the system.

React also allows for state management through the use of hooks. This was used very often in the application when reading user inputs, and also for updating the screen in real time. For example, the counters for the time, score, and exercise number are set up using "useState", which is a React hook. React hooks were also pivotal in being used to show and hide both images and buttons. The exercise shows the changing notes and visually represents a correct key press and an incorrect key press was also used by changing the state of images through "useState".

React was also helpful with the styling of the components and integrating external libraries. This was especially seen with the analysis page, where the "recharts" library was imported and utilized. This library was necessary to add more visual elements to the assessment page to allow for easier comprehension of the results, while also being able to grasp and retain the user's attention. In the assessment page, a pie chart is created using the imported library to visually give a breakdown of all the errors made by the user. CSS sheets were used throughout the entire project to help with styling the pages to be more pleasant to look at and make things easier to understand. These are the major points of how the React framework was implemented in the creation of this project.

3.3.2 - MIDI API

An interface was required to allow the electronic piano to connect and communicate with the application. This application utilizes the Web MIDI API to complete this connection, and it plays a vital role in the architecture of the system. To integrate this architecture into React, first, the program must request access to the MIDI devices that are connected to the device, and set up Event Listeners to listen and wait for any MIDI inputs that are coming in from the electronic piano. Now that the information is being read in, it needs to be processed to be utilized by the system. There are external libraries that could have been imported to be used, but the information was processed manually in the system to allow for more customizability. The format the information comes in is "[status, note, velocity]", and the only necessary information for this project as of right now is status and note. Status tells us what type of message it is, and the values that are relevant are when status is "0x90" to "0x9F", indicating that it is a "Note On" message. Then the second value is the note number, where each number correlates to a key found on the electric piano. These two pieces of information are what is used and collected to record the user's sight-reading capabilities [8].

3.3.3 - Prompting Engineering with GPT o3-mini

Once all of the necessary information has been gathered from the Web MIDI API, it needs to be processed and analyzed to show the user all the errors they have made. This is done through the use of OpenAI o3-mini. This is a reasoning AI model that was released in 2025 and has been shown to be the most effective in identifying errors when prompted with the information [9]. The tested models were GPT-4o, GPT-4o-mini, and o3-mini. While the gpt-4o and gpt-4o-mini models were significantly faster than the o3-mini, it was able to distinguish between the errors much more accurately and given the desired outputs more frequently based on the sample data used. And in this specific scenario, speed is not the most important aspect, as it is not doing any real-time immediate work, but instead analyzing information, in which the user can wait a few seconds. Therefore, accuracy was significantly prioritized over speed, and ultimately, the reason why the o3-mini was used for this application. In regard to prompt engineering, the most important aspect is the developer message, seen in Figure 10, which gives the model context and instructions on how it should behave. Giving context and instructions appropriately is the most crucial part of prompting, and Figure 10 shows the text used for the developer message.

You are a professional piano assistant. Given a list of notes played by the user and notes that are expected, identify the errors and categorize them into categories. For each error found with a note or chord, list and classify it as one of the following: "Failed to read chord", "Failed to read key signature", "Wrong octave", and "Failed to read note". For context, number 21 is the first key (an A key) and number 108 is the final key (a C key) on the piano. For played notes, each line of notes represents a chord or notes played together in unison and the order of the notes does not matter per line. For the expected notes, () brackets show what notes should be played at the same time and are not chords. Notes that are in [] are part of a chord.

"Failed to read chord" - The three expected notes for the chord are not all played within the line.

"Failed to read key signature" - Based on the key signature provided, if there is an error with a note, check if it is because they did not play the key signature, which is a sharp or a flat.

"Wrong octave" - The note letter is correct, but it is played in a different octave.

"Failed to read note" - This is simply when there is an error with playing a note and it does not fall under any of the previous categories.

Only return a count of how many errors were caused while playing for each section. If no errors are found, return "You played it perfectly. ". After the count is completed, return one line that states "Most recurring error:" and list which of the 4 errors is the most recurring. If it was played perfectly, state "None" in this field. If two more categories are tied, use the following list from most important to least important to select the error: "Failed to read note", "Failed to read chord", "Failed to read key signature", "Wrong octave".

Figure 10: The figure is the text that is used as the system message for prompting with the o3-mini OpenAI model.

After the developer message is created, some sample prompts and outputs can be given to the model in the playground to use as a basis for its expected outputs and learns using them. This is very important because these prompts help the model learn and shape future outputs as well. For this specific model of o3-mini, only 5 pairs of inputs and outputs were used, which can be seen in Table 1.

Prompt Information	**Data Input**	**Resulting Output**	**Did it need correction?**
1. The user plays the song with no errors	Key Signature: None Correct Notes: (48 [52 55 60]), 60, 67, 67, (57 53 69), 69, (55 52 67), (53 65), 65, (52 64), 64, (62 50), 62, (48 [52 55 60]),	You played it perfectly. Most recurring error: None	No

	Played Notes: 48 55 52 60 60 67 67 57 53 69 69 67 55 52 53 65 65 52 64 64 62 50 62 52 48 55 60		
2. The user plays two regular note errors	Key Signature: None Correct Notes: (48 [52 55 60]), 60, 67, 67, (57 53 69), 69, (55 52 67), (53 65), 65, (52 64), 64, (62 50), 62, (48 [52 55 60]), Played Notes: 48 55 52 60 60 66 67 57 53 69 69 67 55 52 53 65 65 52 64 62 62 50 62 52 48 55 60 0	Failed to read chord: 0 Failed to read key signature: 0 Wrong octave: 0 Failed to read note: 2 Most recurring error: Failed to read note	No
3. The user plays one regular note error and one chord error	Key Signature: None Correct Notes: (48 [52 55 60]), 60, 67, 67, (57 53 69), 69, (55 52 67), (53 65), 65, (52 64), 64, (62 50), 62, (48 [52 55 60]), Played Notes: 48 51 52 60	Failed to read chord: 1 Failed to read key signature: 0 Wrong octave: 0 Failed to read note: 1 Most recurring error: Failed to read note	No

	60 66 67 57 53 69 69 67 55 52 53 65 65 52 64 64 62 50 62 52 48 55 60		
4. The user plays one octave error	Key Signature: None Correct Notes: (48 [52 55 60]), 60, 67, 67, (57 53 69), 69, (55 52 67), (53 65), 65, (52 64), 64, (62 50), 62, (48 [52 55 60]), Played Notes: 48 55 52 60 60 67 67 57 53 69 69 67 55 52 53 65 53 52 64 64 62 50 62 52 48 55 60	Failed to read chord: 0 Failed to read key signature: 0 Wrong octave: 1 Failed to read note: 0 Most recurring error: Wrong Octave	Yes, the error originally got caught as a single note error
5. The user plays one wrong chord, one wrong octave, and one wrong singular note	Key Signature: None Correct Notes: (48 [52 55 60]), 60, 67, 67, (57 53 69), 69, (55 52 67), (53 65), 65, (52 64), 64, (62 50), 62, (48 [52 55 60]), Played Notes: 48 51 52 60 60 67 67	Failed to read chord: 1 Failed to read key signature: 0 Wrong octave: 1 Failed to read note: 1 Most recurring error: Failed to read note	No

	57 53 69 69 67 55 52 53 65 65 52 64 62 62 50 74 52 48 55 60		

Table 1: This table shows the examples used to train the assistant.

Most of the inputs used for the sample prompts returned results fine, but while distinguishing between octaves and single notes, there seemed to be an issue. Therefore, the output was changed to what was expected, given the model and example of an octave error, and in the following example, where both errors were present, the system was able to distinguish the difference between the two. It is important to note that in the 5 examples, an example of a key signature being incorrect was not added due to the system being able to identify key signature errors from the test cases. But proceeding forward, sample prompts for key signatures should be added to ensure the model is more accurate.

After this OpenAI API call is completed, a second call is made to a new model for a different purpose. This model is using the GPT-4o-mini model, and its purpose is to generate a random list of notes based on the information provided. The information used for the prompt shown in Figure 11 is the notes that were expected to be played, the notes that were played, the breakdown of all the errors, and the statement of the most recurring error. Once the system is provided with all of this information, it will generate a list of just 5 notes based on the correct notes expected and the errors the user has made. The figure below is the developer prompt that was used to create the exercise values to be presented to the user. It can be seen in the prompt, it starts by giving it context on the information that it will be receiving and how to process that information. Then, clear instructions on what it should do with that information are given, and the format of the output is explicitly declared to make the result easier to read and handle. These are all of the OpenAI prompts completed by the system for it to function correctly.

You are given the notes a user has played, the expected notes, and a breakdown of all the errors that have been identified and the most recurring error. Based on the most recurring error and the correct notes the user was meant to play, generate 5 notes to be given to the user as practice.

For context, number 21 is the first key (an A key) and number 108 is the final key (a C key) on the piano. The 5 numbers generated must be between the numbers "60" and "72" and make them correlate to the "Correct Notes" as much as possible. Notes that the user kept messing up on are shown as one of the numbers. Make sure each number is separated by a comma and ONLY the five numbers is returned to the user.

Figure 11: This figure is the system message used for prompting when generating the random 5 notes for the exercises.

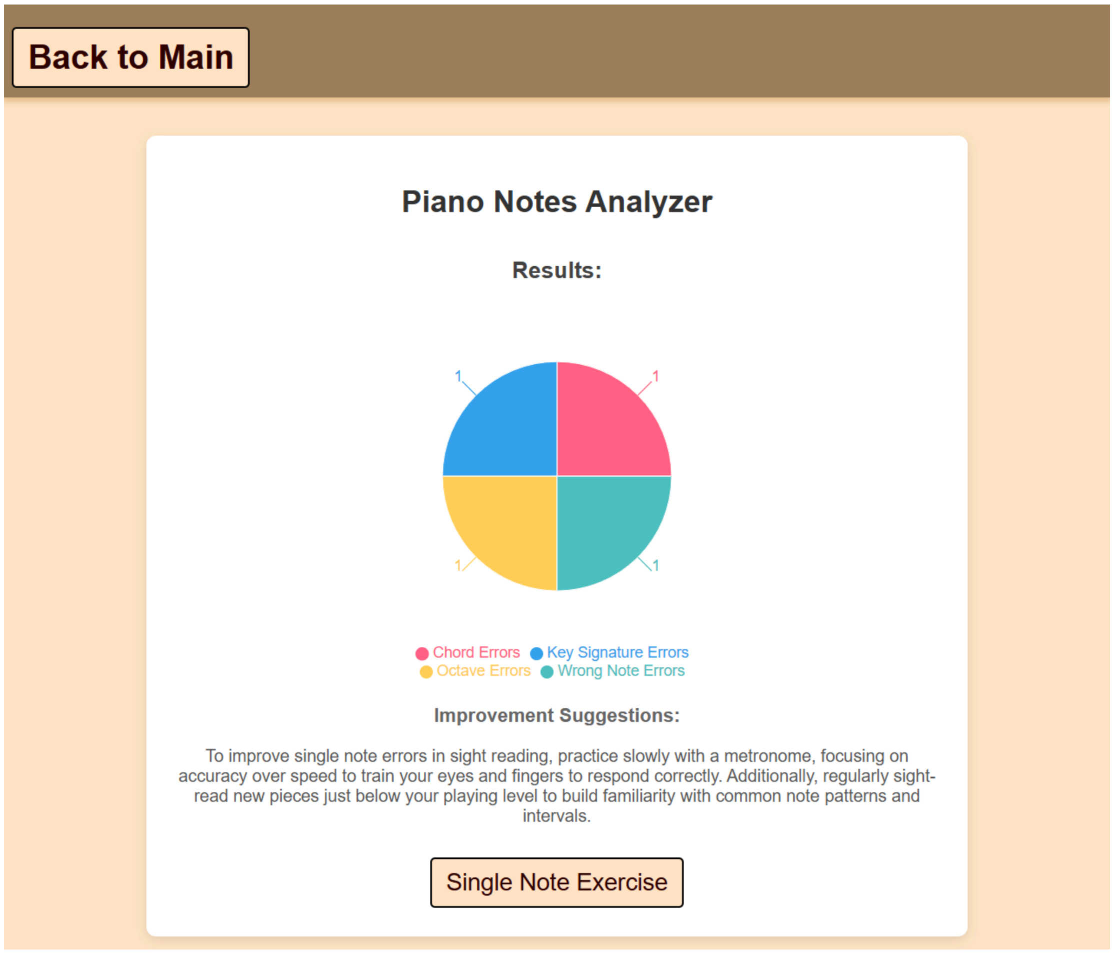
Chapter 4: Use Case and Evaluation

In this chapter, I present a use case using the current prototype and discuss a high-level user evaluation plan for future work.

4.1 - Use case

This use case tackles a dilemma faced during the creation of the system message for prompting. This issue and use case look at when all the errors are categorized into equal portions for each section.
For this specific use case, the user has made 1 chord error, 1 key signature error, 1 single note error and 1 octave error while playing the song “Mary Had a Little Lamb”. The system now would need to decide on which of the errors to take priority over and focus on. Figure 12 shows the pie chart with the errors equally distributed for this use case.

To solve the issue with the test case, it is specifically stated in the system instructions “ *If two more categories are tied, use the following list from most important to least important to select the error: "Failed to read note", "Failed to read chord", "Failed to read key signature", "Wrong octave".* ”. This specific instruction will allow the system to know that the singular “Wrong Note Errors” category should be prioritized and choose that as the most recurring/ important category to target. Once that has been decided, there are improvement suggestions made to the user on how to be better at sight reading for simple single notes and prompt the user to start the single note exercise. Once the user progresses to the exercise, the single note exercise is played with the generated notes for the user and will be prompted to start it. Once the exercise is completed, the improvement suggestion will be shown once again, and the user can go back to the main menu or complete the activity again.


Back to Main
Piano Notes Analyzer
Results:
1
1
1
1
● Chord Errors ● Key Signature Errors
● Octave Errors ● Wrong Note Errors
Improvement Suggestions:

To improve single note errors in sight reading, practice slowly with a metronome, focusing on accuracy over speed to train your eyes and fingers to respond correctly. Additionally, regularly sight-read new pieces just below your playing level to build familiarity with common note patterns and intervals.

Figure 12: This figure shows the analysis page for the specific use case of having an equal number of errors for each group. This shows that the single note errors will always take the highest priority and shows that the priority listing is functioning as expected.

4.2 - User Study and Evaluation

To evaluate the performance and capability of this application, an evaluation plan has been crafted to see the effectiveness and efficiency of this application in terms of guiding users. The evaluation plan will require the participation of volunteers who are either complete beginners or very inexperienced when it comes to playing the piano and sight-reading sheet music. This group of participants will be split up into two separate groups: group A, who will not be utilizing the application, and group B, who will be using the application.

At the start of the evaluation, both groups of volunteers will be given sheets of music to play, and an evaluation will be done based on the number of mistakes they make as they complete sight-reading the sheet music. They will be handed back written notes on the errors they made and an overall evaluation of how well they sight read. After this evaluation is completed, both groups will be given a time of 1 week to practice their sight reading capabilities and to be reevaluated at the end of this period.

During this one week, both groups would be required to practice sight reading for at least an hour each day to count as practice and training completed. This is to ensure that all users in both groups get the same adequate amount of time playing and practicing. Group A would be the group that does not have access to the application, which means that they would need to try and work on their sight-reading skills only based on the feedback received on the day of the evaluation. Group B will have access to the application while practicing, where they must utilize the tool within 1 hour of daily practice sight reading to qualify as completing practice for the week.

Once the one week of practice is completed, both groups will be retested on different sheets of music with the same playing difficulty. Similarly, the volunteers will be once again evaluated and given a breakdown and score of how well they were able to sight-read and play the piece of music presented to them. If much more significant improvements are seen with group B compared to group A, this will show that the application successfully achieves its intended purpose of being a resourceful tool to help users learn how to sight read.

A test that can be completed with group B to check personalization will involve examining the variety of exercises among all users. Do certain sight-reading pieces cause one type of error group to be more prominent? What is the difference in exercises for the same sight reading sheet music? Taking a look at these statistics will help evaluate the performance of the system's adaptive exercise generation for group B. It will also give a good idea of the future steps required to make the system more flexible and how to make the system more personalized moving forward.

Chapter 5: Discussion

Piano Sight Reading Assistant is an application that can show in practice the usability of Large Language Model in helping people learn how to play the piano and sight-read sheet music. But this is just the start, as many aspects can be improved on, and many features can be added to improve usability and accuracy.

5.1 - Limitations:

An important point regarding this application is that it is only compatible with an electric piano that has a port for USB connections. The current form of the application needs direct connection to the piano to successfully read and collect information on how the user is playing and the mistakes the user is making while sight reading. Therefore, without any MIDI inputs, the application cannot function and is a major limitation. This excludes all acoustic pianos, a large demographic of users who can utilize and access the application, and is a significant limitation that must be addressed.

The total number of error groups currently recognized and accounted for during the assessment is only four categories: single note errors, chord errors, key signature errors, and octave errors. Many other complex aspects of sight reading need to be taken into consideration, such as looking at the timing of when keys are played or the duration of time a key is pressed down. All of these different areas of sight reading are crucial and must be considered when evaluating someone's performance [10].

The current iteration of the application has fully completed the exercise for only the singular note errors. The other errors for chords, key signatures, and octaves are not completed with the adaptive approach. These exercises need to be completed and functional to ensure that the application functions correctly, with all possible errors, with an exercise that is appropriate for the error. These are some of the limitations that are found in the current version of the application.

5.2 - Future Work

While the current version of the application is able to implement the core functionalities required, there are many aspects that can be improved upon. There are significant limitations and constraints in regards to the pianos that are compatible with the application, the customizability and prompting can be enhanced, and new features can be added in future versions to enhance the application.

5.2.1 - Microphone

Future work predominantly looks at resolving many of the limitations found with the application, along with adding features to further its capabilities. One important addition would be to include a feature to allow the application to function with a microphone to record what the user is playing. If a microphone that is able to record and distinguish between the keys on a piano is introduced to the application, this allows for the usage of acoustic pianos as well. PianoVision similarly tackles this issue, hence why they can accommodate both electric and acoustic pianos [2]. The microphone would need to be sensitive enough to be able to completely and clearly distinguish every key on a piano and the duration of every key. For this, a system would need to be written, and the blockage of background noise would need to be investigated and applied for the most accurate results. With the implementation of this feature, many more users will be able to utilize the application and further their sight-reading capabilities.

5.2.2 - Error groups

A significant aspect of this thesis is taking a look at the adaptability and customizability of the program. To work on this core aspect, more variations in errors which can be identified need to be expanded. One aspect of sight reading that has yet to be incorporated is looking at the tempo the user is playing the sheet of music. Testing for tempo would essentially be checking to make sure that the user plays the song on beat correctly. Ensure that at no point does the user slow down the song or start picking up speed playing the song. Another error group that can be added is checking that the user can distinguish between the different types of notes and rests. There exist different symbols for notes and rests to determine how long each note/rest should be played within the song. Without being able to identify this, the user will not be able to play the melody and song correctly, and this is something that should be taken notice of. The last example of another error group to include in the future is observing the dynamics while playing. Dynamics looks at the intensity of the notes that the user plays while sight reading. For example, on a sheet of music, if it says “mf” (Mezzo forte), it tells the user to play the notes in a moderately loud tone. Getting this side of sight reading is important because this is what sets the tone of the music, and that is a vital part of expressing the sheet of music. These are just a few vital groups that I would add in the future for error detection.

5.2.3 - Server Implementation

The final addition for future work is the implementation of a server to hold information about the user's current and past assessments. This would be a great addition to the application, because this would allow for the tracking of the user's progress to be gathered and shown back to the user. Information such as the user's accuracy and error percentage for each category can be shown through a graphical interface to allow the user to see how much they have progressed and also to show when errors have the slowest growth rate and which errors have the quickest growth rate. Seeing this type of information will most definitely allow the user and system to identify areas that would need more focus on improvement based on their growth, and allow them to have a more dynamic system that will become more tailored to the user.

Chapter 6: Conclusion

This thesis investigates how to put into practice the ideology of integrating Large Language Models with learning instruments and sight reading sheet music. The creation of Piano Sight Reading Assistant is a web-based application that assesses the user's sight-reading capabilities and uses this information to generate exercises that best suit the user's lacking areas. This application shows that the idea of adaptive learning is possible through the use of OpenAI and GPT prompting as the medium to create these assessments and exercises. But there is still room for a lot of improvement in regards to availability of the software and working on the customizability of errors and exercises moving forward with this application.

Bibliography

[1] Yuksel, B. F., Oleson, K. B., Harrison, L., Peck, E. M., Afergan, D., Chang, R., & Jacob, R. J. (2016, May 7). *Learn piano with Bach: An adaptive learning interface that adjusts task difficulty based on Brain State*. ACM Digital Library. https://dl.acm.org/doi/10.1145/2858036.2858388

[2] PianoVision Inc. (2023, October 10). PianoVision. https://www.pianovision.com/

[3] Bhutoria, A. (2022, April 9). *Personalized education and artificial intelligence in the United States, China, and India: A systematic review using a human-in-the-loop model*. ScienceDirect. https://www.sciencedirect.com/science/article/pii/S2666920X22000236

[4] Flowkey. (2015). Learn how to play piano online - piano learning app. https://www.flowkey.com/

[5] Baker, H. (2022, August 4). *Pianovision brings ar piano lessons to Quest via App lab*. UploadVR. https://www.uploadvr.com/pianovision-app-lab-release/

[6] Flowkey. (n.d.). *Yamaha - Africa / Asia / cis / Middle East / oceania*. Yamaha. https://asia-oceania.yamaha.com/en/musical-instruments/pianos/explore/flowkey/form.html

[7] Nixon, S. (2013, June 9). *Piano major scales: How to build and play them in any key*. PianoLessonsOnline.com. https://www.pianolessonsonline.com/piano-major-scales-construction/

[8] Wilson, C., & Wilson, M. (Eds.). (2025, January). *Web MIDI API*. W3C. https://www.w3.org/TR/webmidi/

[9] OpenAI. (n.d.). *OpenAI - Prompt Engineering*. OpenAI. https://platform.openai.com/docs/guides/prompt-engineering

[10] Gratovich, E. (2020, April 28). *10 tips to improve your sight-reading*. The Strad. https://www.thestrad.com/playing-hub/10-tips-to-improve-your-sight-reading/16.article

[11] Maloney, B. (2022, February 4). *Piano keys 101: Complete beginner's guide*. nkoda. https://www.nkoda.com/blog/piano-keys-101-complete-beginners-guide

[12] Reina, V. (2023, April 12). *Piano Chords explained*. Music To Your Home. https://www.musictoyourhome.com/blog/piano-chords-explained/

[13] Martin, A. (n.d.). *Key signatures - a comprehensive beginner's guide*. Creators in Music. https://creatorsinmusic.com/music-theory/key-signatures/

[14] Li, P. (2020). Research and application of performance evaluation model of rural preschool teachers based on big data algorithm. *2020 International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*, 330–333. https://doi.org/10.1109/ICBASE51474.2020.00076

[15] *What are octaves on piano?*. Los Angeles Music Teachers. (2022, December 19). https://www.losangelesmusicteachers.com/blog/what-are-octaves-on-piano

[16] Rahul & Katarya, Rahul. (2019). A Review: Predicting the Performance of Students Using Machine learning Classification Techniques. 36-41. 10.1109/I-SMAC47947.2019.9032493.

[17] Posada, S. P. (2025, February 5). *Key signatures in music: A comprehensive guide*. Piano Blog by Skoove - Piano Practice Tips. https://www.skoove.com/blog/key-signatures-beginners-guide/

[18] Randieri, C. (2024, July 22). *Personalized learning and AI: Revolutionizing education*. Forbes. https://www.forbes.com/councils/forbestechcouncil/2024/07/22/personalized-learning-and-ai-revolutionizing-education/

[19] Jabbour, J., Kleinbard, K., Miller, O., Haussman, R., & Reddi, V. J. (2025, February 1). *Socratiq: A Generative AI-Powered Learning Companion for Personalized Education and broader accessibility*. arXiv.org. https://arxiv.org/abs/2502.00341

[20] Lin, C. C., Huang, A. Y. Q., & Lu, O. H. T. (2023). Artificial intelligence in intelligent tutoring systems toward sustainable education: A systematic review. *Smart Learning Environments, 10*, 41. https://doi.org/10.1186/s40561-023-00260-y