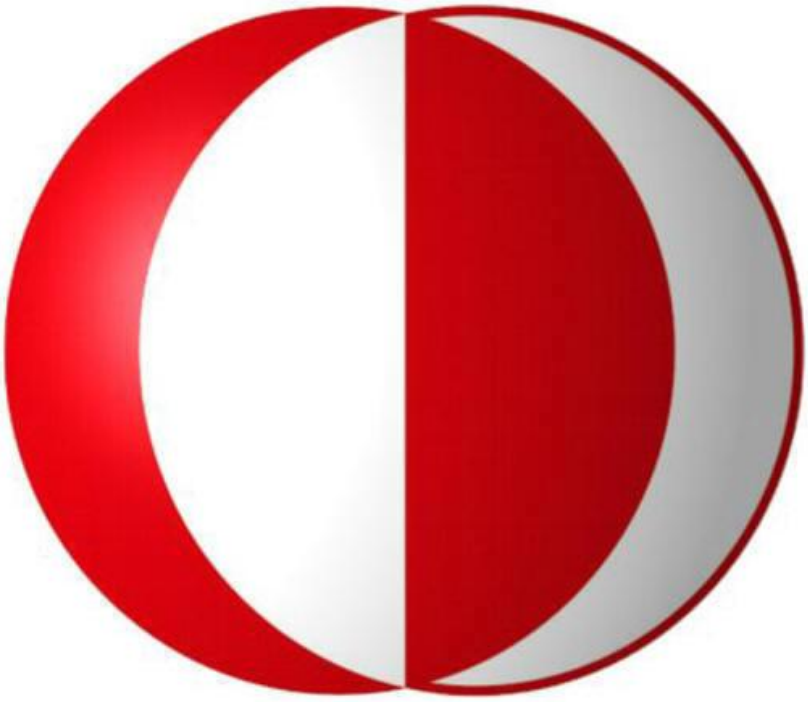
Middle East Technical University



CENG 491

Initial Design Report
for
"DigiMuse"

GOBIT
M. Burhan Şentürk
M. Yiğit Yıldırım
Kamila Kuchalieva
Ezgi Berberoğlu

1 Introduction

This document contains the initial design descriptions of “DigiMuse” which is a mobile application to convert printed sheet music into MIDI format or to read MIDI files directly to edit them. The approach used in this specification is adapted from IEEE recommended practices [1]. This document also abides the standards presented [2].

1.1 Problem Definition

Sheet music, which is known as a hand-written or printed form of musical symbols, is accessible also in digital environment. Whether professional or not, almost all the musicians are familiar with reading musical notes, in fact, they need it. However, reaching them and also maintaining and distributing the printed or hand-written ones are really hard. While this is one of the main reasons to develop “DigiMuse”, another reason is to make it possible to play with what is written on these sheets. Moreover, it is also an important feature to have the chance to edit what is written on the sheet and to play it whenever it is desired. Although, there are lots of applications doing that, there is a need for a mobile one for doing all these. “DigiMuse” will have all the capabilities to solve all the problems listed.

1.2 Purpose

This initial design report intends to provide complete description of all requirements of “DigiMuse”. The descriptions suggested in this document will serve as a guideline throughout the development process of this project. The end-product will be tested against the requirements ton ensure the quality of the software produced.

1.3 Scope

This document contains a complete description of the initial design of “DigiMuse”. Its main intention is to provide software design description of the system according to Software Requirements Specifications.

1.4 User and Literature Survey

Our product mainly targets to self-educated musician, music learners in fact all instrument players. In today's world, technology plays an important role on the process of learning to play an instrument. There are a lot of software products trying to help this process.

There are products that can convert sheet music to digital formats but they can only convert scanned documents. Some of them are; SmartScore, Capella-Scan and SharpEye. These programs are called Music OCR programs and they all have same technique behind their technology.You need a scanner to scan the sheet music and they use optical character recognition to interpret sheet music or printed scores into editable and, often, playable form. This process is not very practical and also these products are, in general, quite expensive.

Guitar Pro is the product that inspired us the most. It provides the similar functionality we intend to add to our product. It enables users to open a midi file and see the notes and tabs of the song. It also shows the finger positions on the keyboard as you listen the song. It helps a lot to learning process of playing an instrument. Our product will also provide all this features. Guitar Pro has both mobile and desktop versions. However, mobile version of the Guitar Pro is only available for iOS and it only supports Guitar Pro specific file formats not MIDI. We will build our product for Android OS and we will support the most common format, MIDI. Furthermore, Guitar Pro mobile is quite expensive. For Android OS there are a few applications such as; GuitarTapp and Ultimate Guitar Tabs but with these applications you can only view tabs online.

1.5 Definitions and Abbrevations

CPU	Central Processing Unit
SRS	Software Requirements Specification
SDD	Software Detailed Design
MIDI	Musical Instrument Digital Interface
OS	Operating Sistem
OCR	Optical Music Recognition

IEEE	Institute of Electrical and Electronics Engineers
RAM	Random Access Memory
SD	Secure Digital
GUI	Graphical User Interface

1.6 References

[1] IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications

[2] CSS.06 – Yazılım Gereksinimleri Belirtimi Standardı Rev.7.0

[3] www.creately.com

[4] http://agutie.homestead.com/files/Inca_Music.htm

[5] http://www.lib.virginia.edu/artsandmedia/dmmc/Music/UnicodeMusic/

[6] http://en.wikipedia.org/wiki/OpenCV

[7] http://opencv.willowgarage.com/wiki/

[8] http://en.wikipedia.org/wiki/Eclipse_(software)

[9] http://developer.android.com/sdk/ndk/overview.html

[10] http://developer.android.com/guide/developing/tools/index.html

2 System Overview

This product is mainly intended for users who are in some way or another involved in music. Potential users will be able to convert their sheet music into digital sheet music and play it easily on different instruments using their mobile phones. The project has two main modules to achieve the specified purpose.
The software of the product is going to have a relatively simple structure which consists of two main modules. Detailed information about the components will be given in the following sections.

The product will be providing the set of functions in order to supply the user with the facilities explained throughout this document. We plan to keep functionality design as simple as possible for the user to use the application most effectively.

3 Design Considerations

3.1 Design Assumptions, Dependencies and Constraints

DigiMuse is going to be a mobile application. Therefore, in terms of the hardware requirements we are going to consider any mobile device which has to be supporting the Android OS. The mobile device is supposed to have the camera.

Since our product is going to be a mobile application, there will be a considerable factor of limitation in terms of the screen size. We are going to have four different screen options for users to choose from and decide on the most appropriate one to view. There will be one screen option where the user will view the screen with all displayable parts displayed, and there will be other three options where the user will view the combination of any two displayable parts. The user will be free to choose which two parts he/she wants to see on the screen in a larger scaled

manner. Furthermore, for the sake of the efficient usage of the device screen we will avoid displaying any unnecessary information on the screen. The screen layout will be balanced in terms of the information content and size.

We may have to deal with the time constraint issues due to the fact that one of the main tasks that we are going to implement while developing DigiMuse is going to be the image processing. The CPUs on the mobile devices are not as fast as the ones on the computers for instance, therefore we will have to bring in some solutions or constraints in order to provide the reasonable time performance.

We know that modern mobile devices have cameras of quite high quality which allows the pictures taken buy them to be of large ratios. This is again an obstacle for us in terms of the mobile device's CPU capabilities and memory constraints. We will have to play with the size of the image before we perform the image processing task Thereby, we will put less efford in order to process the image.
Since the image processing task is quite complicated one when is comes to processing different note symbols, we may have a certain amount of performance loss in terms of correctness of the symbols been processed.

3.2 Design Goals and Guidelines

DigiMuse is going to be used by users who are interested in learning or playing musical notes. So our end-users will mostly use this software product as a tool which would help them to enhance their note reading /playing skills. Therefore, from the user's point of view the design must be simple enough to understand and use. Thereby, design simplicity is on of the most important design goals that we as the developers of the product care about. Users will enjoy the user friendly design of the product.

As already mentioned above, we may not be able to correctly process each note symbol all the time, so we are going to provide the user with feature of editing the notes been processed when it is needed to so.

4. Data Design

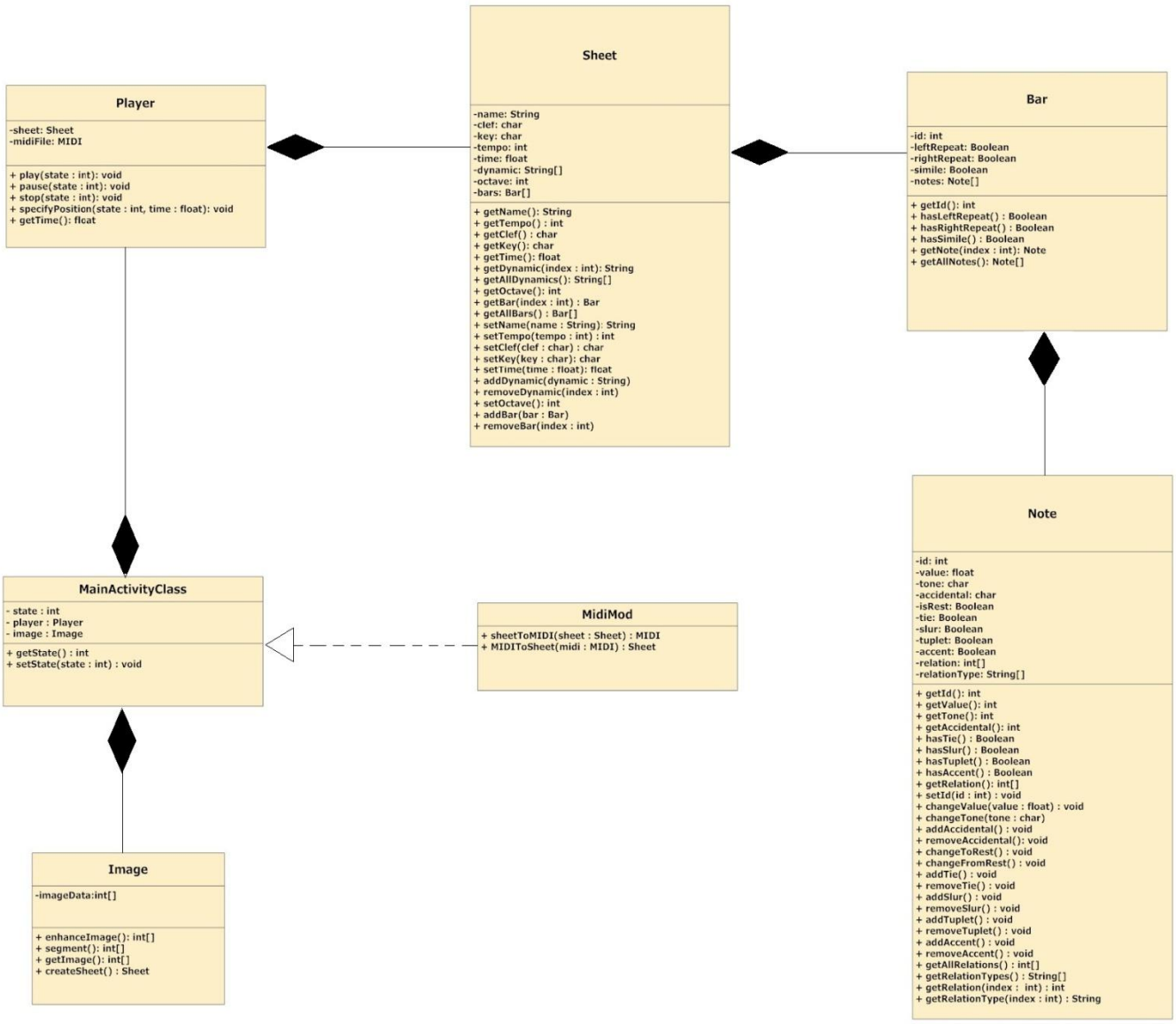


Figure 4.1. Simplified Class Diagram

4.1. Data Dictionary

4.1.1 MidiMod Class

This class provides full functionality for converting between MIDI format and sheet instance.

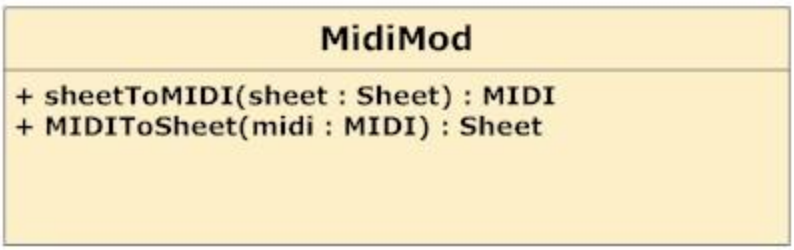


Figure 4.2. MidiMod Class Diagram

Attributes:

No attributes.

Methods:

sheetToMIDI: This method converts sheet object to a MIDI formatted file. This will provide us to save currently playing MIDI to SD card.

MIDIToSheet: This method converts MIDI formatted file to a Sheet object. Sheet object is used to visualize the MIDI file.

4.1.2 Image Class

This class performs image processing techniques with its methods on the image that is the photograph of the sheet music taken by the camera of the mobile device.

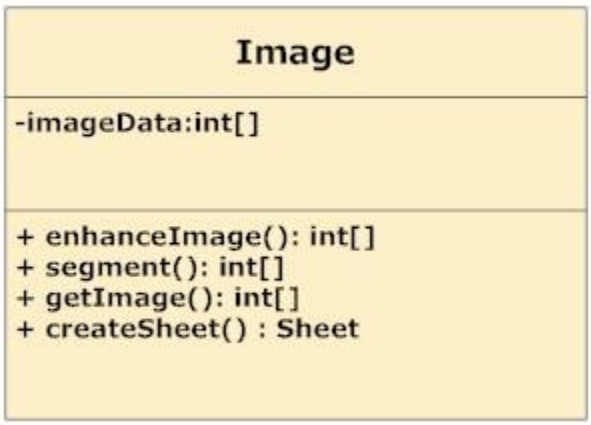


Figure 4.3. Image Class Diagram

Attributes:

imageData: Contains the data of the provided image constructed from the photograph

Methods:

Getter and setter for the image.

enhanceImage: Image enhancement techniques will be applied with this function. Image enhancement is the process of improving the quality of a digital image for further processes.

segment: Extract the note positions and values from the image.

createSheet: This methods creates the sheet object using the knowledge which is gotten from image processing methods.

4.1.3. Player Class

This class enables to run the MIDI that is given or produced by Image object synchronized with the sheet object.

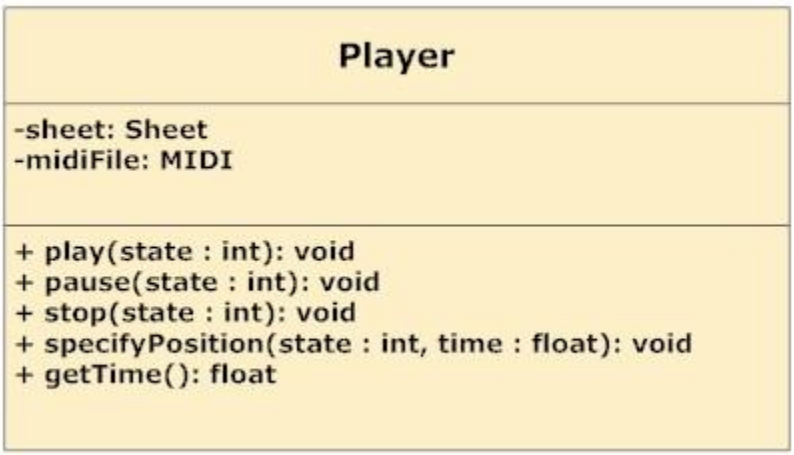


Figure 4.4. Player Class Diagram

Attributes:

sheet: Contains the sheet object of the selected image object or MIDI file.

midiFile: Constructed built-in MIDI object from the specified MIDI file that is given or produced via Image object.

Methods:

play: This method will start to play current MIDI file.

pause: This method will pause the currently playing MIDI.

stop: This method will stop the currently playing MIDI.

specifyPosition: This method will be used to specify starting point for player.

getTime: This method will return the instantaneous position of player over the file.

4.1.4. Sheet Class

This class is the base for all musical related work. It consists of Bar objects, and specifications of the specific sheet music represented as the attributes.

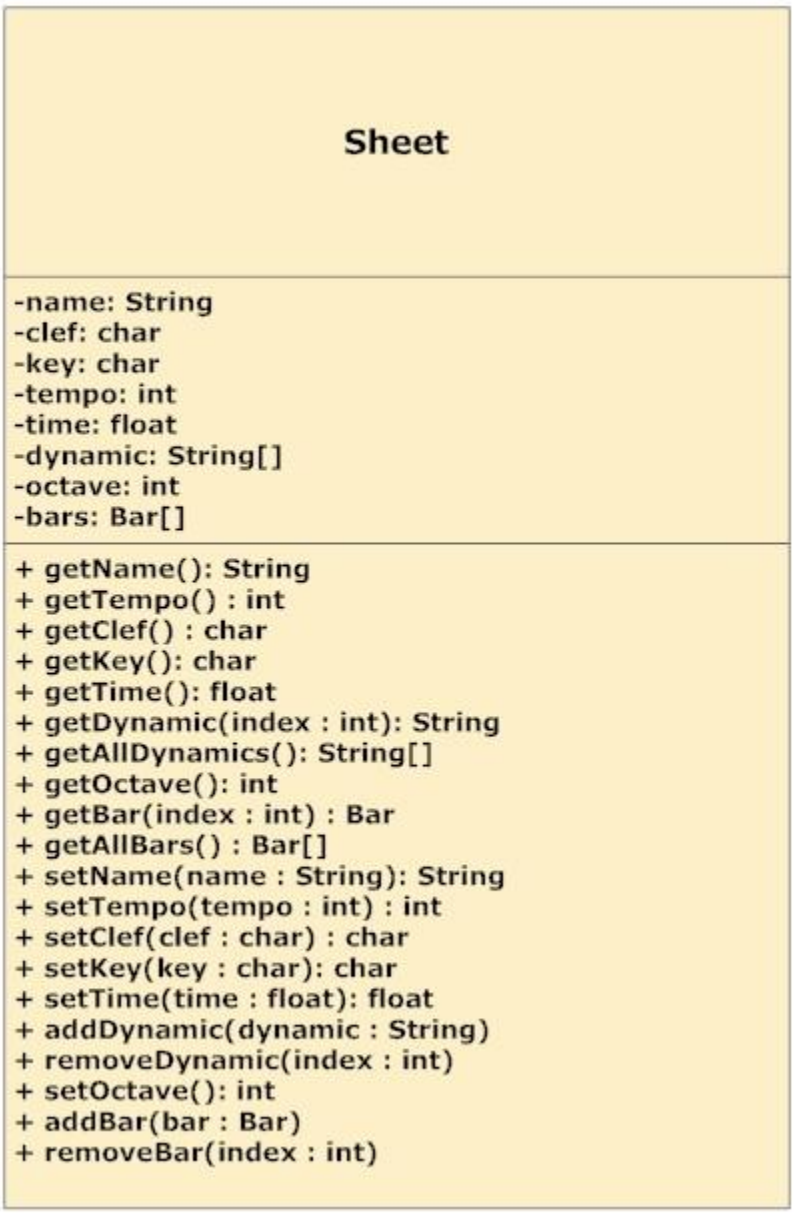


Figure 4.5. Sheet Class Diagram

Attributes:

name: Unique name of the sheet

clef: Clef attribute defines the pitch range. It can be G, F or C.

Key: Key of the sheet music defined in the start

tempo: Metronome of the music. Default speed is valid unless specified.

time: Represent the specific beat.

dynamic: Indicates the relative intensity or volume of a musical line. i.e. piano, forte, fortissimo etc.

octave: Octave of the sheet music. Default positions are valid unless specified.

bars: Bar object array contains the notes in the sheet.

Methods:

There are only getter and setter methods of all attributes.

4.1.5. Bar Class

This class is the container of basic musical components, notes. It has the attributes on its own to enable repetition facilities between bars.

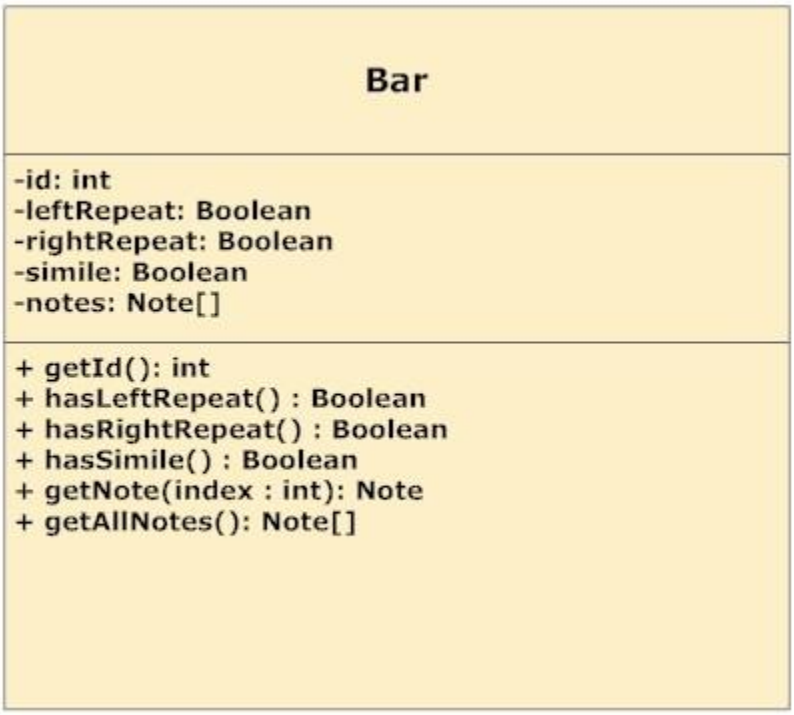


Figure 4.6. Bar Class Diagram

Attributes:

Id: Unique ID of the bar.

leftRepeat: Indicator of whether the bar object contains a left repeat mark or not.

rightRepeat: Indicator of whether the bar object contains a right repeat mark or not.

simile: Indicator of whether the bar object contains a simile mark or not.

notes: Note object array contains the notes in one bar.

Methods:

There are only getter and setter methods of all attributes.

4.1.6. Note Class

This class basically defines the note concept adapted directly from the real life. A note object has a lot of attributes to have the capability of representing each musical nuance. The attributes are defined below.

Figure 4.7. Note Class Diagram

Attributes:

Id: Unique ID of the note.

value: Simply defines the length of the note

tone: Defines the pitch value of the note. i.e. A, B, C, D, E, F, G

accidental: Defines if there is a pitch change by a semitone in the note, i.e. flat, sharp and natural.

isRest: Indicator of whether the note is a rest or not.

tie: Indicator of whether the note has a tie or not. If yes, enumerates all the tied notes with the same value.

slur: Indicator of whether the note has a slur or not. If yes, enumerates all the slurred notes with the same value.

tuplet: Indicator of whether the note is played as a tuplet or not. If yes, enumerates all the notes in the tuplet with the same value.

accent: Indicator of whether the note is played with an accent or not. If yes, the corresponding enumeration value of the appropriate accent is contained as the value.

relation: Indicator of whether the note is connected with other notes or not. If yes, enumerates all the connected notes with the same value.

Methods:

There are only getter and setter methods of all attributes.

4.1.7. MainActivityClass Class

This class is the main class of the application. Application starts with this class and all UI activities are operated by MainActivityClass. Moreover, main features of the application such as image processing and playing MIDI is controlled here.

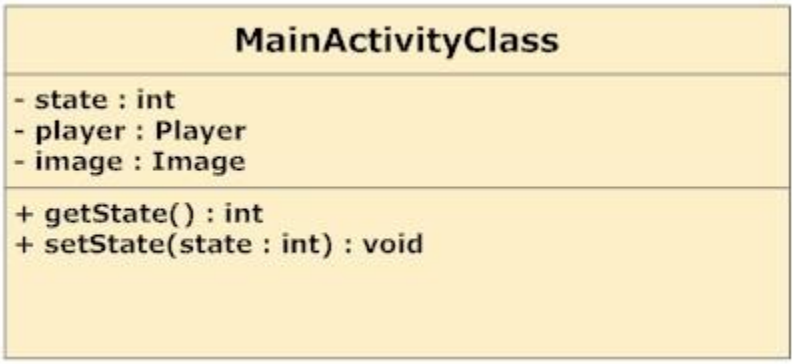


Figure 4.8. MainActivityClass Class Diagram

Attributes:

state: A state information for GUI.

player: Player class instance. It is used to manipulate all operations over the MIDI file.

image: An image instance which is taken by user. This image is processed by here and used to create sheet object.

Methods:

There are only getter and setter methods of state attribute.

5 SYSTEM ARCHITECTURE

In this section, software system architecture for DigiMuse is presented in a top-down manner.

5.1 Architectural Design

There are two main modules interacting with each other in the project, “DigiMuse”. These are discussed in the coming sections and their roles are indicated by using sequence diagrams.

5.2 Description of Components

This section describes the major components of “DigiMuse”. It has two main modules which are Optical Music Recognition Module and MIDI Reader Module.

5.2.1 Optical Music Recognition Module

5.2.1.1. Processing narrative

The optical music recognition module provides the necessary functionality for converting the photo of the printed sheet music which is taken by a mobile device into a Sheet instance.

5.2.1.2. Interface description

This component gets a photo of a printed image as an input and it returns an instance of a Sheet class.

5.2.1.3. Processing detail

The optical music recognition module is responsible for converting the photos of printed musical sheet into a digital one as accurate as possible. To define all musical characters necessitates the usage of image processing methods. After all the characters are recognized, corresponding attributes are specified in a Sheet instance.

5.2.1.4. Dynamic behavior

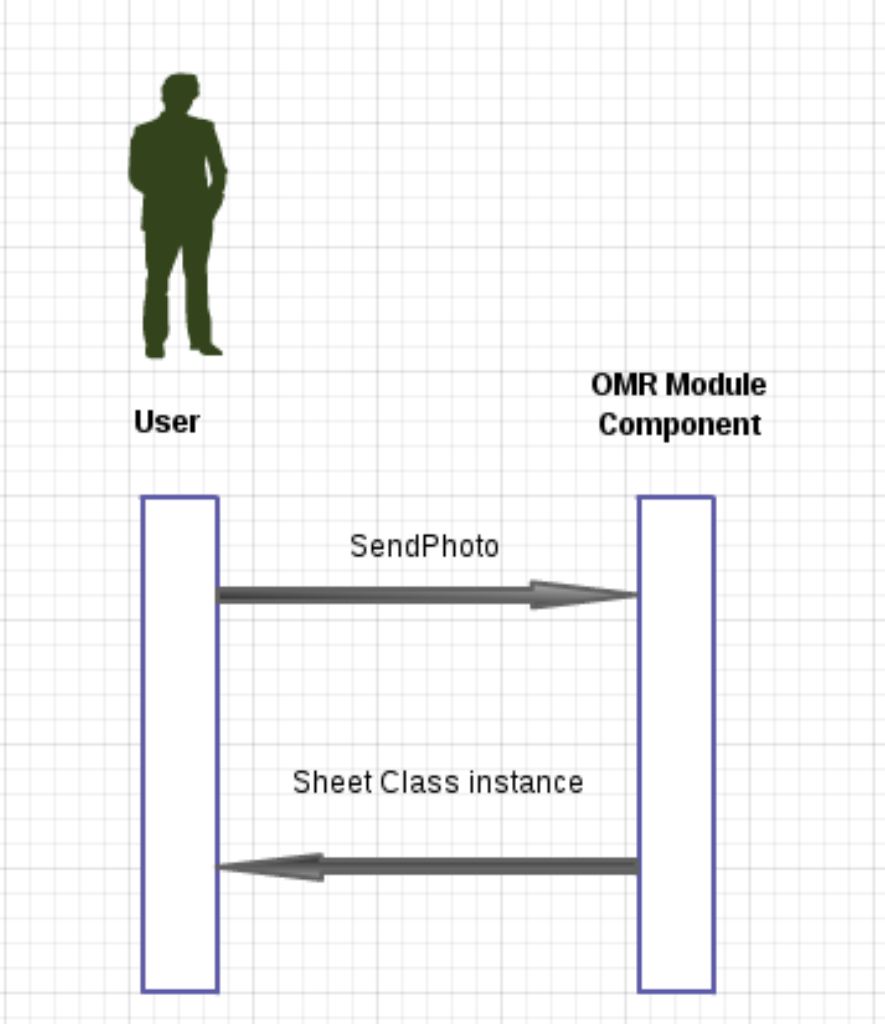


Figure 5.1. Sequence Diagram for OMR Module

Optical music recognition can interact with the MIDI reader module. After it outputs a Sheet instance, the MIDI reader module has the capability to convert into MIDI format.

5.2.2 MIDI Reader Module

5.2.2.1. Processing narrative

The MIDI reader module takes a Sheet instance and converts in into a MIDI file. On the other hand, it has the capability to make the reverse operation which is changing the MIDI file into a Sheet instance.

This module has two components which are explained below:

5.2.2.1.1 MidiMod

One of the components of MIDI reader module is MidiMod. This component takes a Sheet class instance and it returns a Midi File as output.

5.2.2.1.2 Player

The component, Player, takes the MIDI file which is the output of the MidiMod components, or it takes the Sheet class instance from the optical music recognition module. Then, it outputs a Sheet class instance.

5.2.2.2. Interface description

This module can get a Sheet instance or a MIDI file as input. As output, it can create a MIDI file for a Sheet instance input, or a Sheet instance for a MIDI file input.

5.2.2.3. Processing detail

MIDI reader module has a variety of functions to read MIDI files and apply edit functions on them. All of these functions are given with their classes in section 4. Using the two components, MidiMod and Player, MIDI reader module gets a Sheet class instance as input and it outputs a sheet class instance.

5.2.2.4. Dynamic behavior

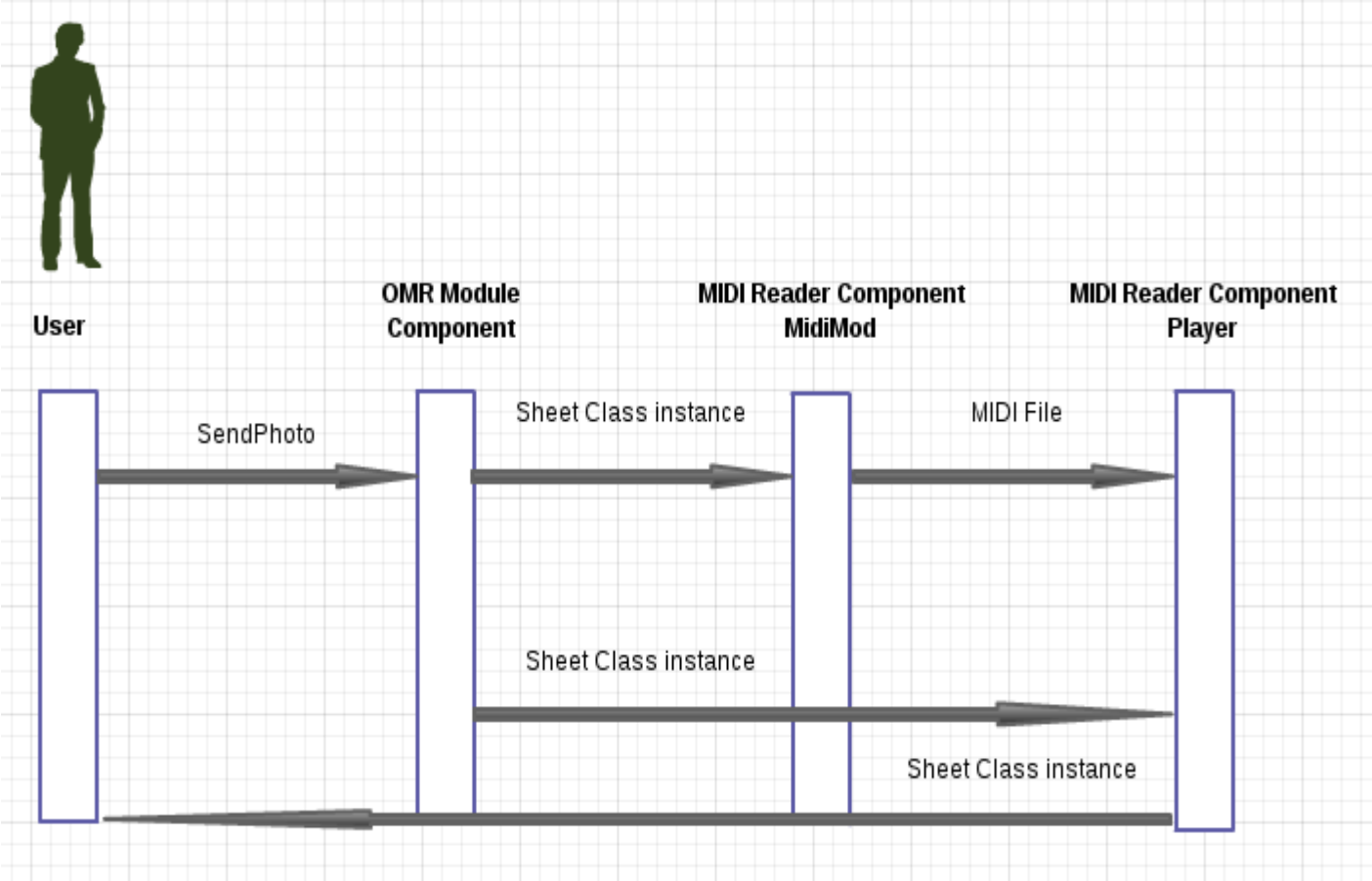


Figure 5.2. Sequence Diagram for MIDI Reader Module

MIDI reader module can interact with the optical music recognition module. If the user does not choose the option to read an already existing MIDI file, then this module takes the output of the optical music recognition module.

5.3 Design Rationale

After deciding on the project, the project group, GOBIT, made a detailed search on the shortages of the existing applications about optical music recognition. First of all, the product is decided to be a mobile application and in addition to recognizing printed symbols, our software can also read MIDI files.

System composition is planned in the most efficient way to satisfy what is required. The recognizable characters are chosen to be as many as possible. While designing, all group members discussed on the usage area of the application and it is decided that it would be better at reading classical music characters.

5.4 Traceability of requirements

For both modules, optical music recognition module and MIDI reader module, there is one to one mapping for all the components with the uses cases defined in the software requirements report.

6 User Interface Design

6.1 Overview of User Interface

DigiMuse will start with the screen that has two buttons for two different functionalities. Specifically:

- Take Photograph
- Open MIDI

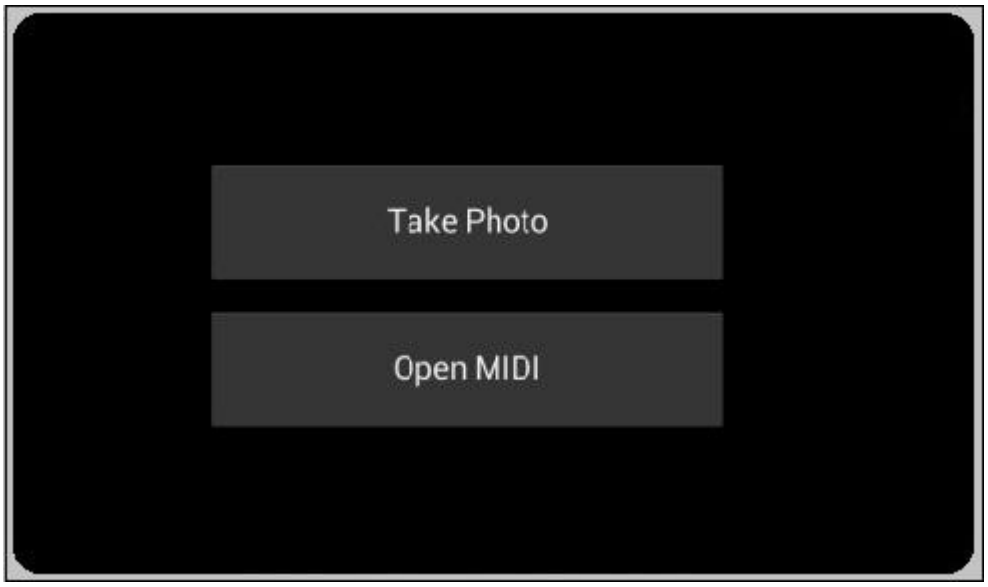


Figure 6.1. Entrance Screen

At this point the user chooses one of two options, in particular the user may choose to take a picture or load a MIDI file. In either case the user will be directed to the appropriate screen.
In case of “Open MIDI” option, the user will be able to load the file which has to be in MIDI format. If the users want to take a picture of some sheet note, he/she does so using the camera of the mobile device. Which will display the following:

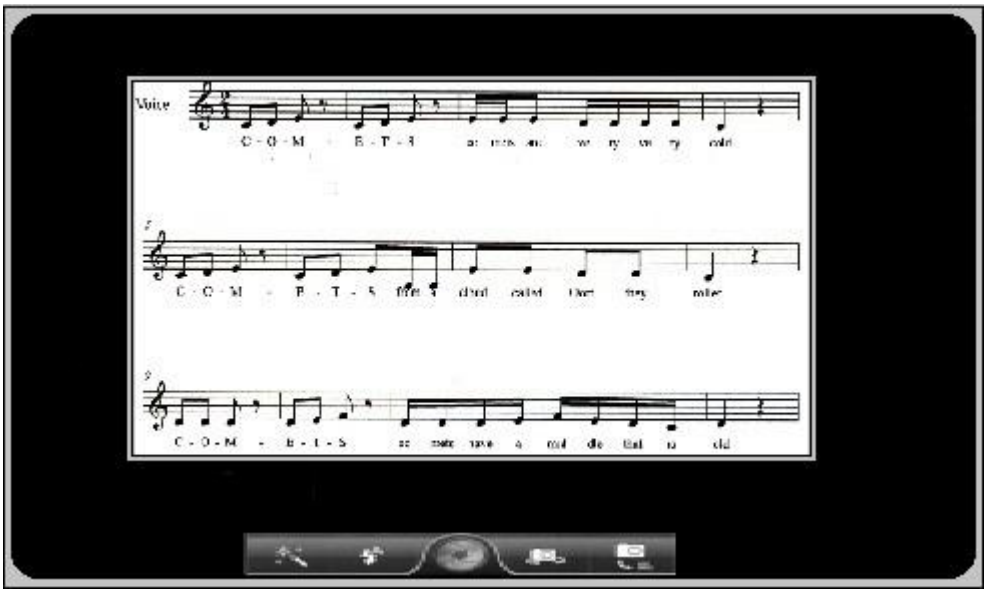


Figure 6.2. Taking the Picture

Once the user has taken the picture, it is now ready to be processed by OMR Module.

The user will see the following screen, while the image processing is being processed:

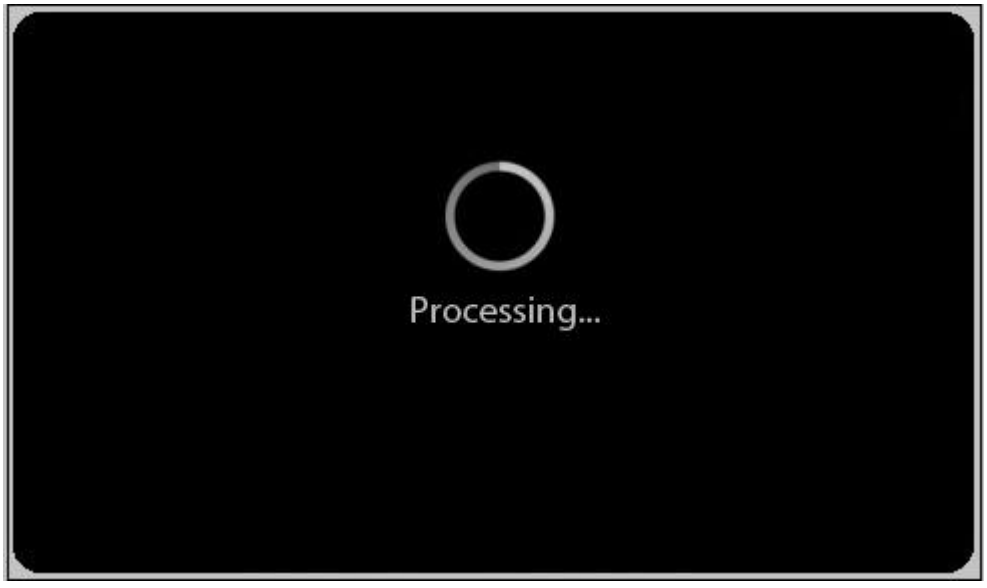


Figure 6.3. Waiting for Image Processing

Once the image has been processed, DigiMuse is now ready to be used and the following screen will be displayed:

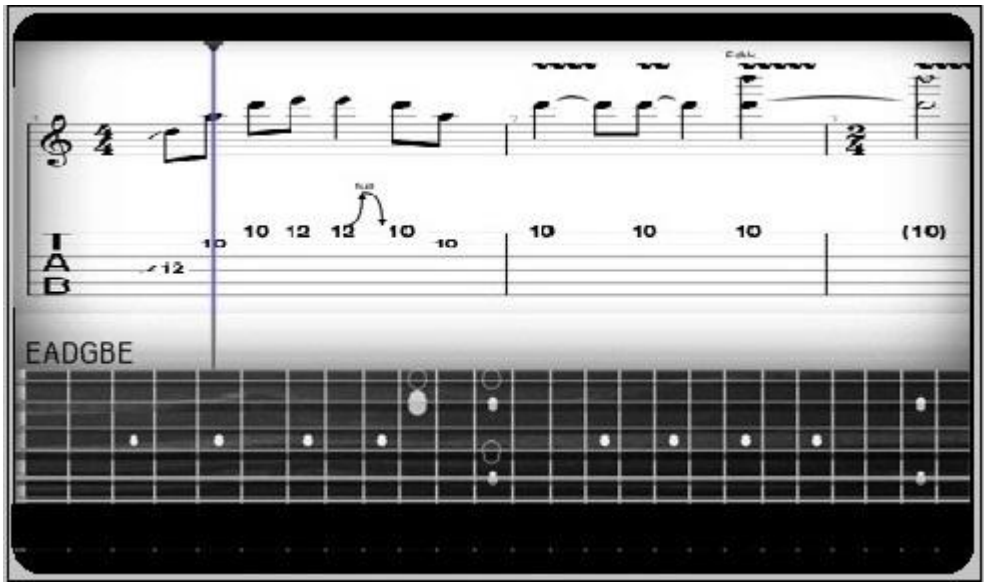


Figure 6.4. Main Screen View

This screen option views the sheet note, tab and keyboard sections all in one screen. However, if we take into consideration the fact that mobile devices have relatively small screen size, compared to laptops for example, we decided to provide the user with the chance to be able to separate the above screen into several parts.

There three different screen options consist of different combinations of:

- Tab and Keyboard:

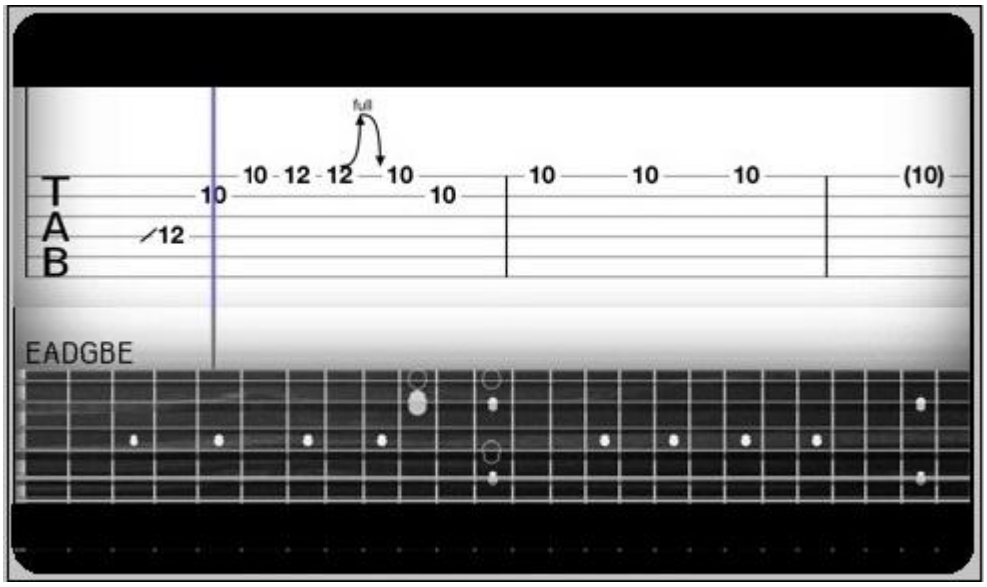


Figure 6.5. Main Screen View Variation

- Sheet Note and Keyboard:

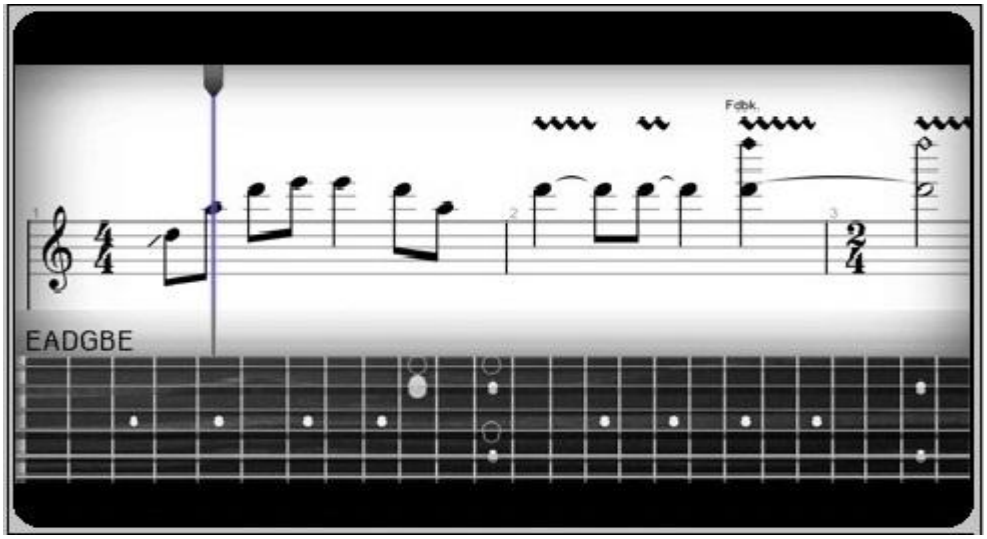


Figure 6.6. Main Screen View Variation

After the user finishes playing the sheet music, he/she may quit the application using the "Exit" option from the main menu, which is going to be referred to and explained in more details in section 6.3.

- Sheet Note and Tab:

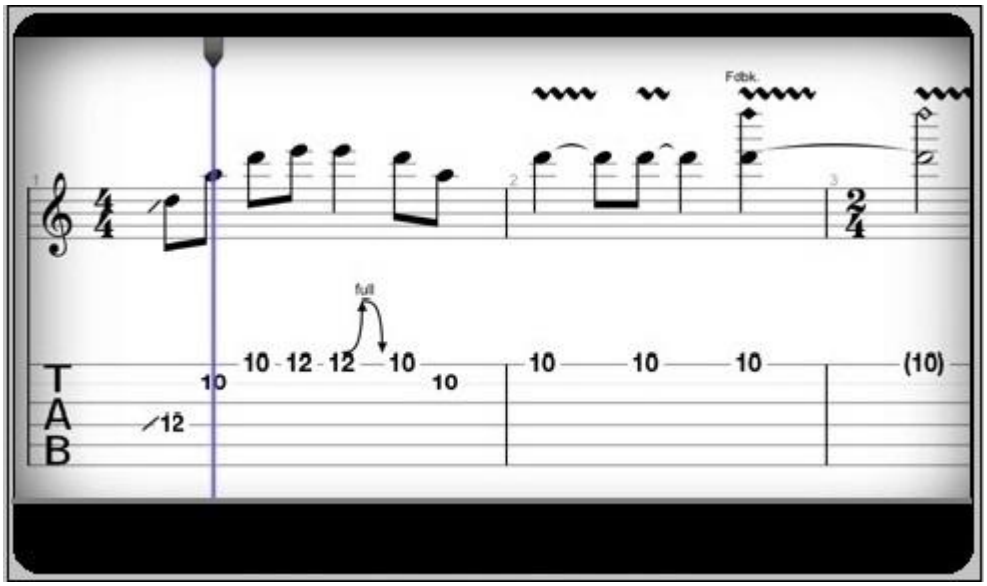


Figure 6.6. Main Screen View Variation

We decided to give the main screen view in three different options in order to let the user feel comfortable viewing the content in terms of screen size and visibility.

6.2 Screen Images

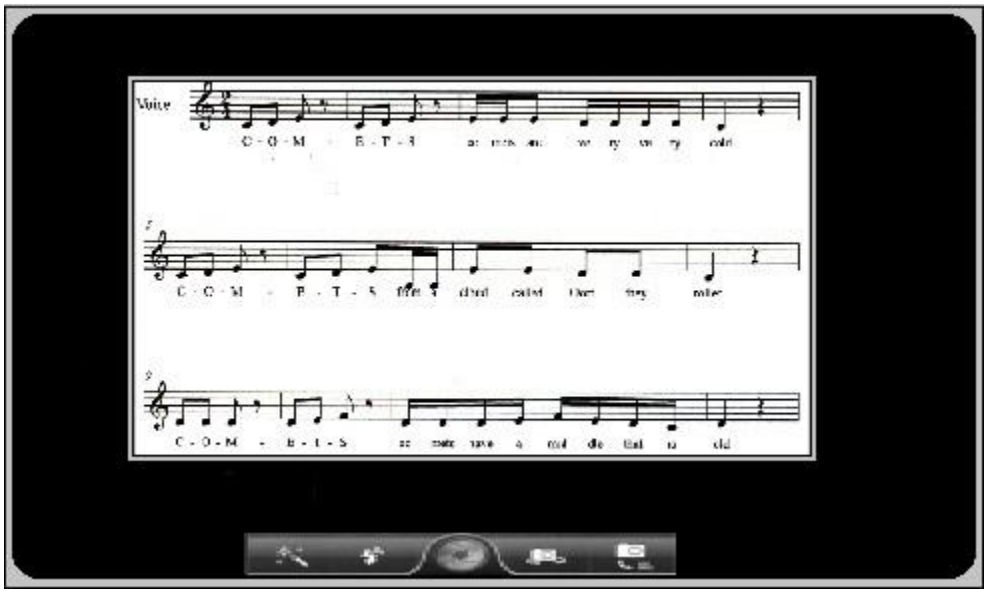


Figure 6.7

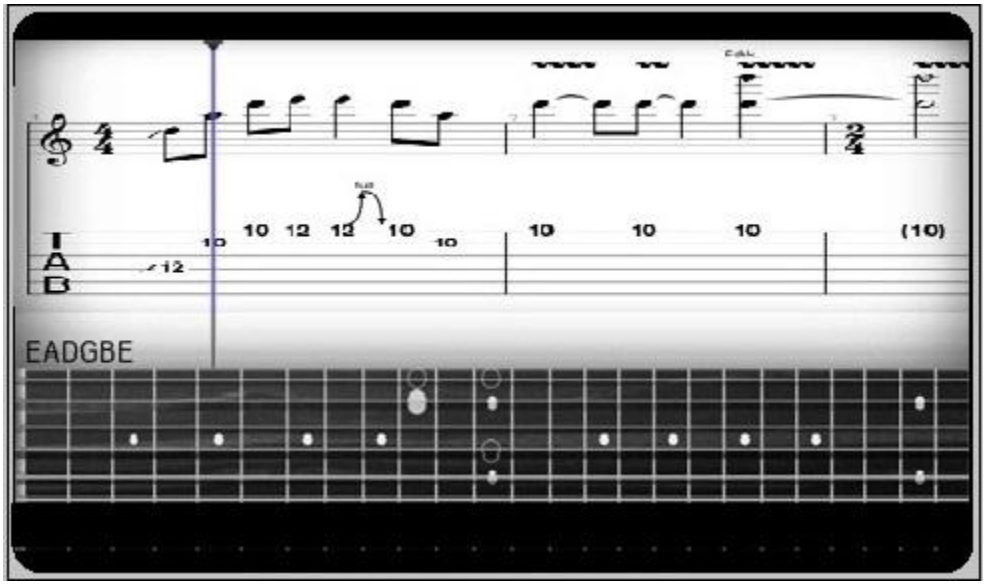
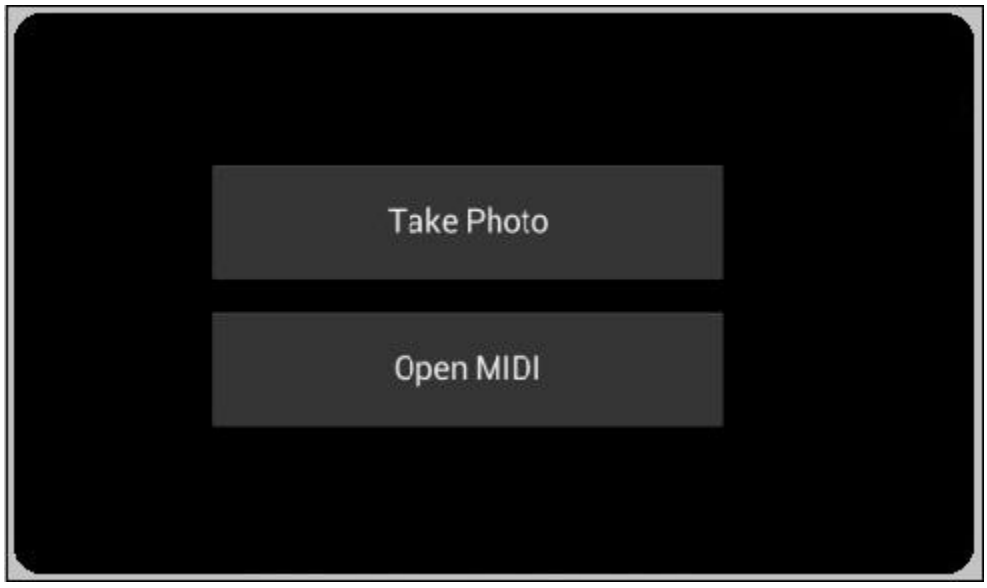


Figure 6.8

6.3 Screen Objects and Actions

6.3.1. Entrance Menu

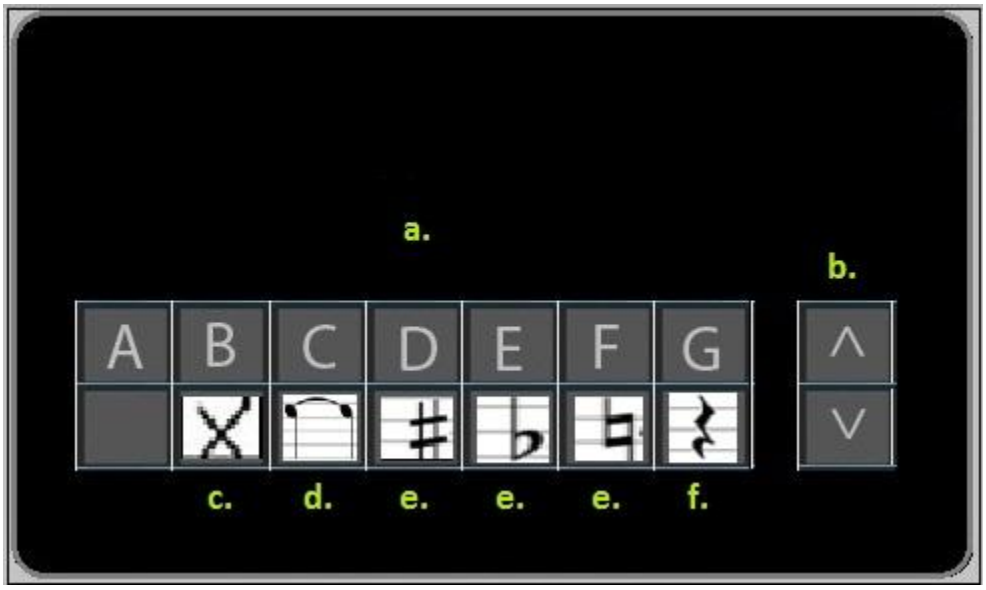


a. Take a photo to be processed. After this is selected camera of the device will be opened.
b. Open a MIDI file to listen and visualize the notes.

6.3.2. Main Menu

a. New: Opens the entrance menu to create new player.
b. Export: Exports the currently playing MIDI to a readable and portable document.
c. Tempo: Change the playing pace of the MIDI.
d. Edit: Used for entering the editing mode.
e. Transpose: Used for transposing the sheet by desired pitch.
f. Octave Change: Used for changing the octave by desired amount.
g. Clef Change: Used for changing the clef with the desired one.
h. Save As: Used for saving the currently playing MIDI to a location on SD card.
i. Exit: Used for exiting the program.

6.3.3. Editing Keyboard



a. Letters: Corresponds to note tone. They are used for changing the tone of selected note.
b. Arrows: Used for changing the value of the selected note.
c. Ghost Note: Changes the note to a toneless click sound.
d. Note tie: Adds a tie between the selected note and the next note.
e. Accidentals: Used for changing the tone of the selected note by half tone.
f. Rest: Converts the note to a rest (silent) note with specified value by the arrows.

6.3.4. Edit Mode Menu

a. Dynamics: Used for changing dynamics of the selected note.
b. Articulation: Used for changing articulation of the selected note.
c. Octave: Used for changing octave of the selected note.

More information about these features can be found in SRS Section 3.2.4.1.

7. Libraries and Tools

7.1 OpenCV

7.1.1. Description

OpenCV (*Open Source Computer Vision Library*) is a library of programming functions mainly aimed at real time computer vision, developed by Intel. It is free for use under the open source BSD license. The library is cross-platform [6]. OpenCV has C, C++ and Python interfaces so far. It focuses mainly on real-time image processing and the implementations are highly optimized [7]. OpenCV provides lots of functionality for segmentation and recognition.

7.1.2. Usage in Digimuse

Digimuse involves lots of image processing and pattern recognition due to its Optical Music Recognition engine and mobile device's camera usage. First of all, digitalizing a sheet music requires highly an enhanced image in order to achieve high accuracy rates. Also, the image will be saved using mobile device's camera, so it needs aligning and pre-processing. For these, image processing facility of OpenCV will be used. After that, using the enhanced sheet music image, segmentation and pattern recognition functionalities of OpenCV will be used for note detection.

7.2 Android SDK

7.2.1. Description

The Android SDK includes a variety of tools that help you develop mobile applications for the Android platform. The tools are classified into two groups: SDK tools and platform tools. SDK tools are platform independent and are required no matter which Android platform you are developing on. Platform tools are customized to support the features of the latest Android platform.[8]

7.2.2. Usage in Digimuse

Since, DigiMuse operates on Android OS; Android SDK is a must to use.

7.3 Android NDK

7.3.1. Description

The Android NDK is a toolset which enables the users to embed components that make use of native code in your Android applications. Android applications run in the Dalvik virtual machine. The NDK allows the user to implement parts of his/her applications using native-code languages such as C and C++. This can provide benefits to certain classes of applications, in the form of reuse of existing code and in some cases increased speed.[9]

7.3.2. Usage in Digimuse

DigiMuse uses OpenCV as it is explained in Section 7.1. OpenCV is not yet compatible with Java but the finest quality and easiest to use library in Computer Vision. Thanks to Android NDK, the code using OpenCV can be written in C++ and embedded into main java code. The project greatly makes use of this functionality.

7.4 Eclipse

7.4.1. Description

Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system [10]. Eclipse is a free and open source software. The IDE has different versions for different languages such as Eclipse CDT for C/C++, Eclipse JDT for Java and etc.

7.4.2. Usage in Digimuse

Digimuse, runs on devices operating on Android. Using Android Development Tools (ADT) plug-in of Eclipse JDT enables us to implement clean and efficient Java code. With C++ plug-in (CDT), OpenCV usage becomes easier.

8 Time Planning

8.1 Term 1 Gantt Chart

Current Week													
Weeks / Tasks	10/10 to 10/17	10/17 to 10/24	10/24 to 10/31	10/31 to 11/06	11/06 to 11/13	11/13 to 11/20	11/20 to 11/27	11/27 to 12/04	12/04 to 12/11	12/11 to 12/18	12/18 to 12/25	12/25 to 01/01	01/01 to 01/08
Project Selection - PreProposal													
Project Proposal													
Requirement Analysis and SRS Document													
Individual Research													
Initial Design Report													
Preparation of First Basic UI													
Detailed Design Report													
Prototype Demo													
Tasks / Weeks	10/10 to 10/17	10/17 to 10/24	10/24 to 10/31	10/31 to 11/06	11/06 to 11/13	11/13 to 11/20	11/20 to 11/27	11/27 to 12/04	12/04 to 12/11	12/11 to 12/18	12/18 to 12/25	12/25 to 01/01	01/01 to 01/08

Figure 9.1

8.2 Term 2 Gantt Chart

Current Week														
Weeks / Tasks	02/06 to 02/13	02/13 to 02/20	02/20 to 02/27	02/27 to 03/05	03/05 to 03/12	03/12 to 03/19	03/19 to 03/26	03/26 to 04/02	04/02 to 04/09	04/09 to 04/16	04/16 to 04/23	04/23 to 04/30	04/30 to 05/07	05/07 to 05/14	05/14 to 05/21
Implementing Image Processing Module															
Implementing UI of Mobile App															
Conversion to MIDI Format															
Implementing MIDI Reader Module															
Adding Extra Features to MIDI Player															
Testing the Application															
Tasks / Weeks	02/06 to 02/13	02/13 to 02/20	02/20 to 02/27	02/27 to 03/05	03/05 to 03/12	03/12 to 03/19	03/19 to 03/26	03/26 to 04/02	04/02 to 04/09	04/09 to 04/16	04/16 to 04/23	04/23 to 04/30	04/30 to 05/07	05/07 to 05/14	05/14 to 05/21

Figure 9.2

9 Conclusion

Initial design elements of “Digimuse” have been presented in this document. The design for data model, system architecture, and user interfaces have been discussed. A lot of detailing has been done on the project scenario that shows the most of the essential details. This document is the first design of the project in a way that further detailed information is provided. Moreover, it defines all the components with initial details by class diagrams and sequence diagrams, user interface design and how the user interacts with DigiMuse. Further information on the technical design is given with detailed explanations of the modules. Lastly, the progress of the project so far and future plans are stated.