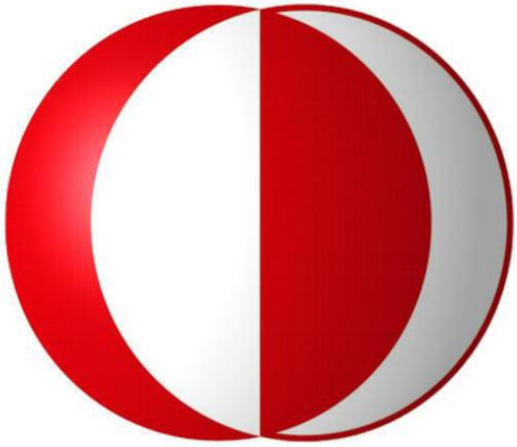
Middle East Technical University



CENG 491

Software Requirements Specifications
for
“DigiMuse”

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

Table Contents

1 Introduction

This document contains the software requirements for “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 software requirements specification intends to provide complete description of all requirements of “DigiMuse”. The requirements 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 to ensure the quality of the software produced.

1.3 Scope

This document addresses the functionality, external interfaces, performance, attributes and design constraints “DigiMuse” to be developed. However, the requirements presented in this document do not impose any design or implementation details.

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
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.creatly.com
[4] http://agutie.homestead.com/files/Inca_Music.htm
[5] http://www.lib.virginia.edu/artsandmedia/dmmc/Music/UnicodeMusic/

1.7 Overview

The report contains seven sections. First of alli it starts with the purpose and scope of the application. After the introduction part, there is an overall description including product perspective, product functions and constraints, assumptions and dependencies. Then, specific requirements of the software will be explained. In this part, interface requirements and both the functional and non-functional requirements will be addressed. As the fourth part, data model and description, information domain for the software will be described. Then, behavior of the software will be explained by introducing the major states and using state transition diagrams. After the planning part, the report will conclude with Conclusion part.

2 Overall Description

2.1 Product Perspective

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 realatively simple structure. It will be consisting of two main modules:

- **Optical Music Recognition(OMR) Module.**

  The user will need a piece of sheet music, like:

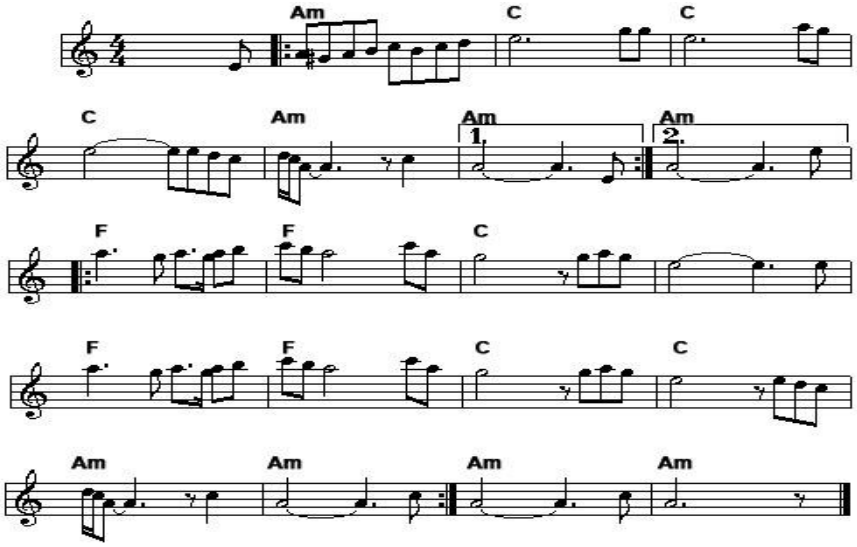


Figure:1

Example of Sheet Music
El Condor Pasa
by Daniel Alomia Robles

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

Then the user will take picture of sheet music with her/his mobile phone runnin on Android:

It is our Optical Music Recognition(OMR) Module that enables the user to take a photograph of the printed sheet music with his/her mobile device and then converts this music to MIDI format.

This feature will also be very handful to music libraries and publishers which want to replace their printed sheet music with their digital verisonsd, since MIDI format is easily convertible to digital sheet music or tablature.

- **MIDI Reader Module**

  MIDI Reader Module takes the MIDI formatted music and provides MIDI reading facilities and in order to enable MIDI to convert sheet music to its digital equivalence.

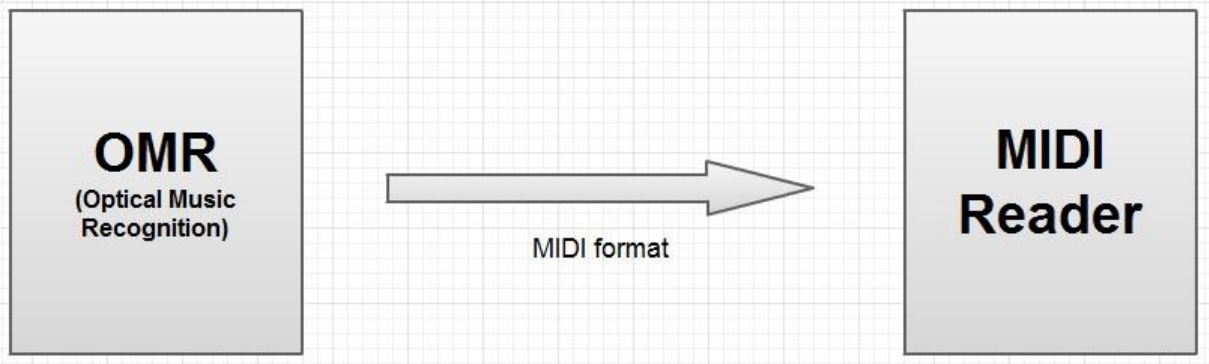


Figure:2

From this step on our users will able to play their sheet music on a mobile device that has to be operating on Android. The MIDI Reader Module of our project, will also provide the user a chance to visualize a MIDI file in digital sheet music format or tablature format.

2.2 Product Functions

The porduct will be providing the set of functions in order to supply the user with the facilities explained throughout this document. We plan to keep funcitonality design as simple as possible for the user to use the application most effectively. The informations given in this section will cover the the brief overview of the product funcitons:

- ***Save as MIDI***

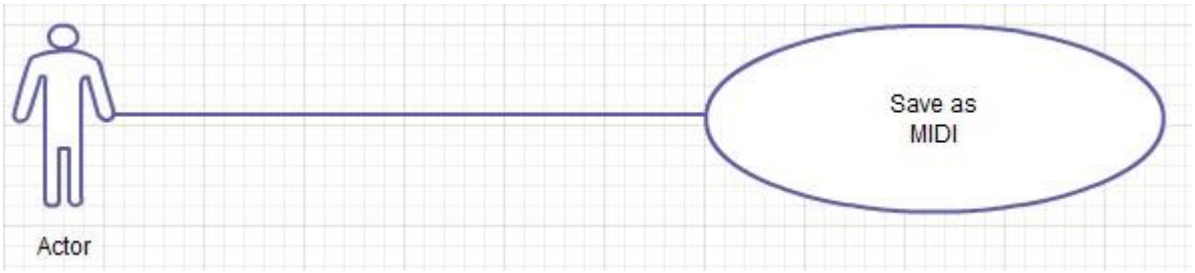


Figure:3

The file will be saved as a MIDI file.

- ***Play***

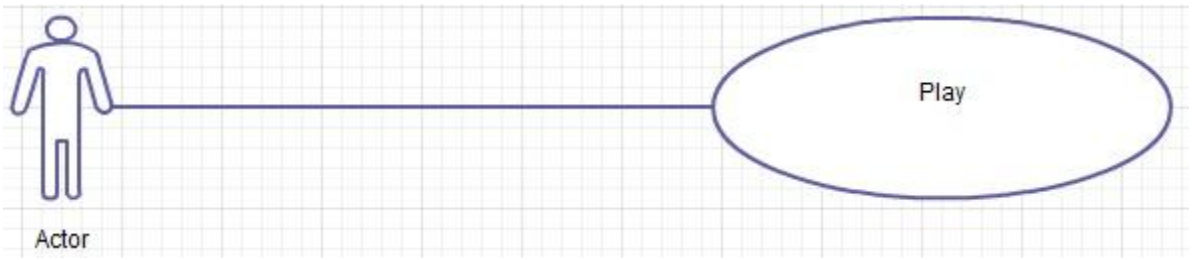


Figure:4

This funciton has the straightforward meaning, it plays the music that is now in MIDI format.

- ***Stop***

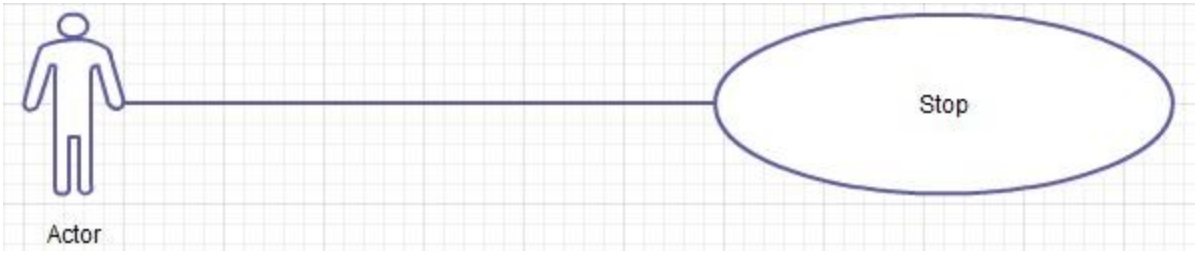


Figure:5

Stops playing the music.

- ***Pause***

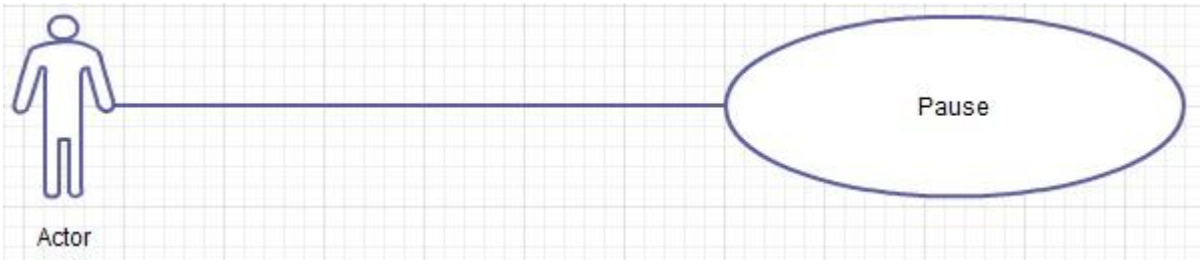


Figure:7

- ***Note Edit***

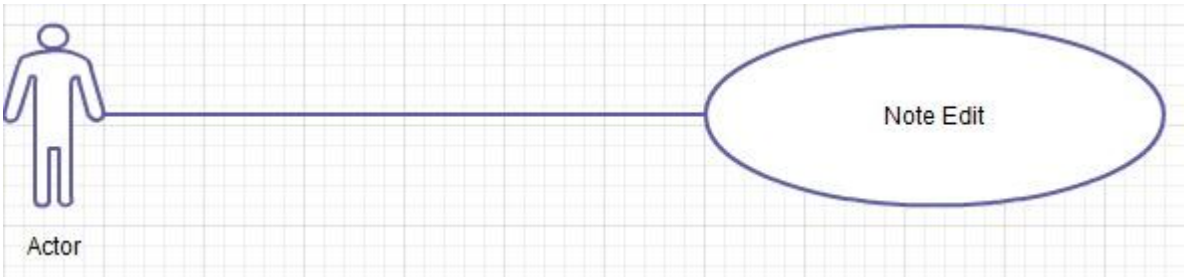


Figure:8

The user will be able to perform simple note editing features if she/he wishes to do so. One of the examples for such a need could be the case when user finds the mistake in notes' appearence due to some sort of probable inaccuracy during the image proccessing state of sheet music emage.

- ***Tempo***

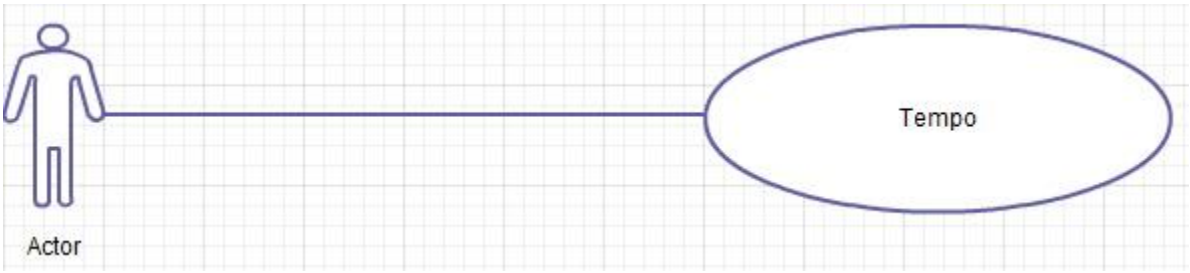


Figure:9

The user will be able to adjust the tempo of the music being played.

- ***Transpose***

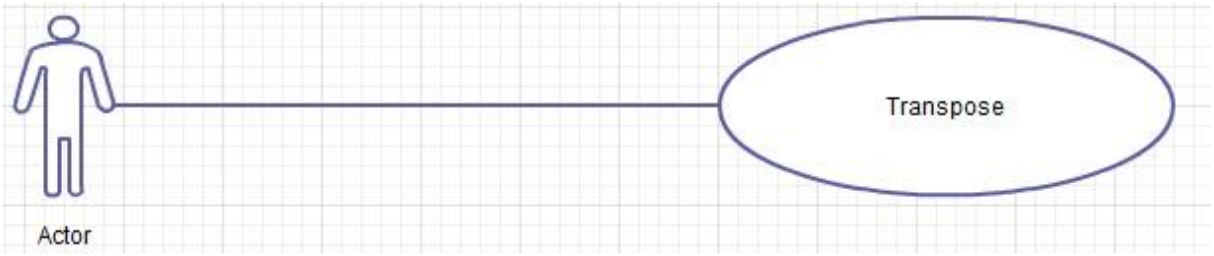


Figure:10

This funtion will provide the user with notes transpose facilities.

- ***Clef Change***

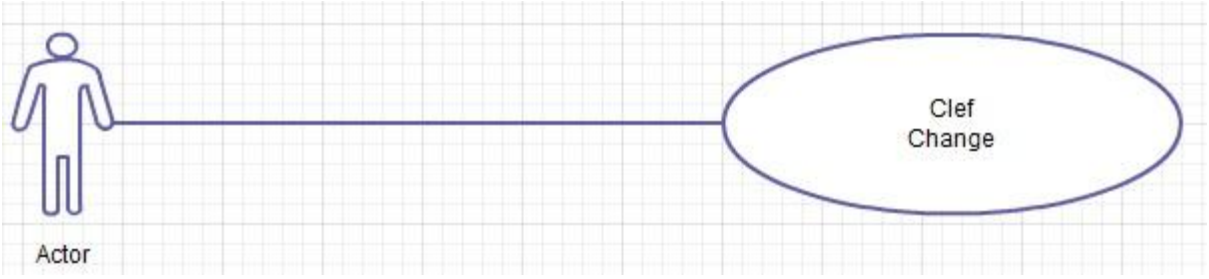


Figure:11

The user will be able to change the clef with the help of this function.

- ***UI Change***

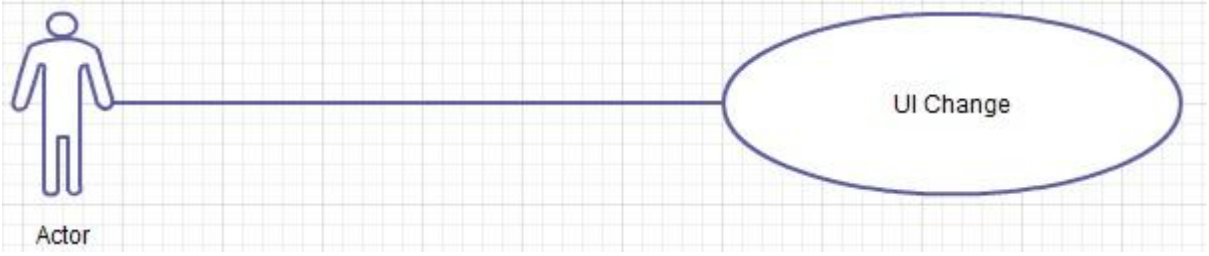


Figure:12

The screen of the MIDI Reader may be differentiated into 3 different parts: digital sheet music part, tablature format and keyboard parts. So the user interface will be based on which one (or more) of the those 3 parts the user want to be viewing.

- ***Export***

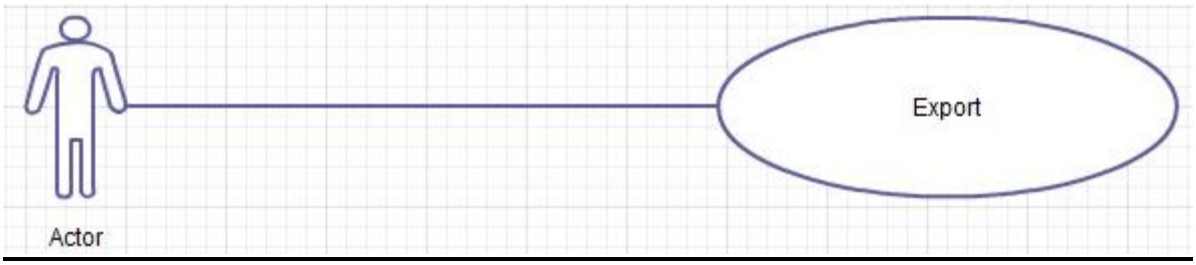


Figure:13

Using this feature will import the MIDI formatted file.

- ***Load MIDI***

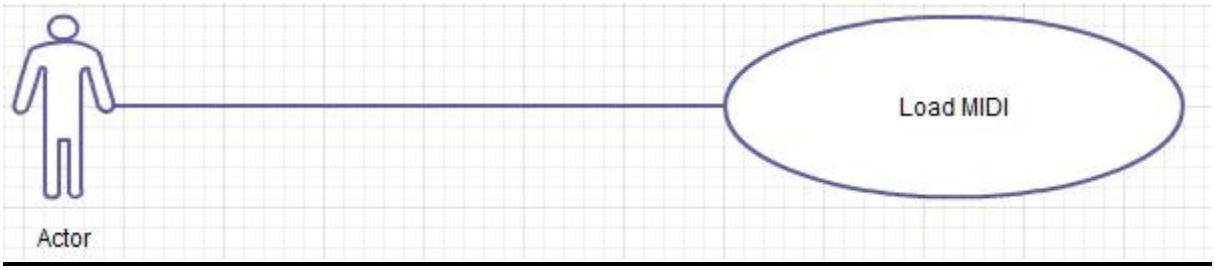


Figure:14

This feature will be used for loading the MIDI formatted files.

2.3 Constraints, Assumptions and Dependencies

It has already been metioned that DigiMuse application will be used on mobile devices. The mobile device using our application will have to be running on Android OS. So the range of our potential user set will cover everybody who have the Android supporting cellphones.

Due to the limitaions on the sceen size of typical mobile devices we decided to provide our users with the abilitiy of choosing one option out of several for screen view. Specifically, users will be able to view different parts of screen content separately instead of leaving them with one screen view option only, which would show all of three subparts (digital sheet music part, tablature format and keyboard parts) simultaneously. This will allow the users to get the best out of the the screen size of their mobile devices.

3 Specific Requirements

This section describes all the software requirements to a level of detail sufficient to enable designers to design and testers to test a system which satisfies this specification.

3.1 INTERFACE REQUIREMENTS

When the application is run, a menu appears asking the user to take a photo to be converted into MIDI format or to use an existing MIDI file. Then, all the options are chosen through one interface which provides the user to choose what should be done next. User can choose Load MIDI option just to load a MIDI file or choose Export option to export a MIDI formatted file or Save as MIDI option to save a MIDI file. He/she can also determine the user interface by choosing what will appear in the mobile device's screen. Furthermore, there is an option to change the clef and tempo, and to transpose the notes and note edit option provides the user to play with the musical symbols. There are also Pause, Play and Stop options.

3.2 Functional Requirements

3.2.1 Image to MIDI Conversion

3.2.1.1 BACKGROUND INFORMATION

Converting printed sheet music to a digital format is our product's main function. The main aim of this project is to enable the musicians to create their digital sheet music archive and play it easily with illustrations, tabs and vocalizing abilities with different instruments in a mobile device. Traditional printed sheet music is easily reachable by musicians in libraries or musical method books. Our project's image to MIDI conversion feature simply enables the user to take a photograph of the printed sheet music with his/her mobile device operating on Android and convert it to a digital format i.e. MIDI. MIDI (Musical Instrument Digital Interface) is an industry-standard protocol that enables electronic musical instruments (synthesizers, drum machines), computers and other electronic equipment (MIDI controllers, sound cards, and samplers) to communicate and synchronize with each other. MIDI format is easily convertible to digital sheet music or tablature. In addition to those, our project will solve any musician's issue by simply digitalizing a musician's sheet music library and letting him/her to carry them with a simple mobile device.

To enable this feature we intend to use some image processing techniques on the image of sheet music. With these techniques, the program will determine that exact position of each note on the stave (the set of five horizontal lines and four spaces that each represents a different musical pitch). After applying image processing to the image we will have the value of each note and then we can simply create MIDI formatted file with this information. Our application will then open this MIDI file by its MIDI reader module which will be explained later on this document.

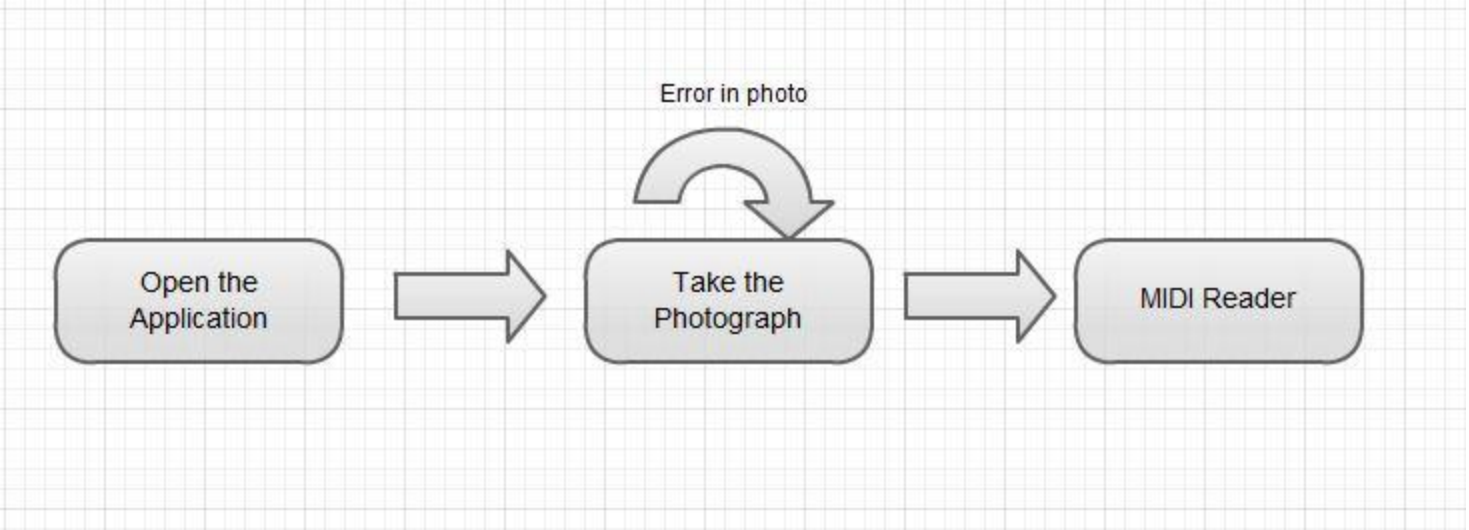


Figure:15

3.2.1.2 STIMULUS/RESPONSE SEQUENCES

3.2.1.2.1 Basic Data Flow

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to take a photograph of a printed sheet music.
4. User points and shoots the photograph.
5. Photograph is processed by the application and MIDI file is created.
6. MIDI reader is opened.

3.2.1.2.1 Alternative Data Flow 1

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to take a photograph of a printed sheet music.
4. User points and shoots the photograph.
5. Photograph is processed and not enough information is found to create a MIDI file.
6. User gets an error message and is asked to take the photograph again.

3.2.1.3 FUNCTIONAL REQUIREMENTS

- The photograph may not be a photograph of a printed sheet music or image processing may not be so successful on that image to extract enough information. The system should check if there is enough information to create a MIDI file. If not it should give an error message.

- The quality of the image should be checked by the system. If image is too dark or too light to process, user should be warned.
- The system should check the image format. If the format is not supported, an error massage should be given.
- If the image is broken or in a bad format, an error massage should be given.

3.2.2 Changing the User Interface

3.2.2.1 BACKGROUND INFORMATION

The user can visualize a MIDI file in digital sheet music format or tablature format thanks to MIDI reader module of the application. The main screen of the MIDI reader consists of three main parts; digital sheet music part, tablature format part and keyboard. This screen is explained in detail in the interface requirements part of this document. The application will give users the option to change the appearance of this MIDI reader screen. It will basically give the opportunity to select which of three features user want to see on this screen. Users can select only one, two or all three parts to see. The application will adjust the size and scale the parts properly for a good view. The needs of the user who own a device with a small display or large display may change in terms of user interface. Changing the user interface will give the opportunity to customize the application and have a better view for users.

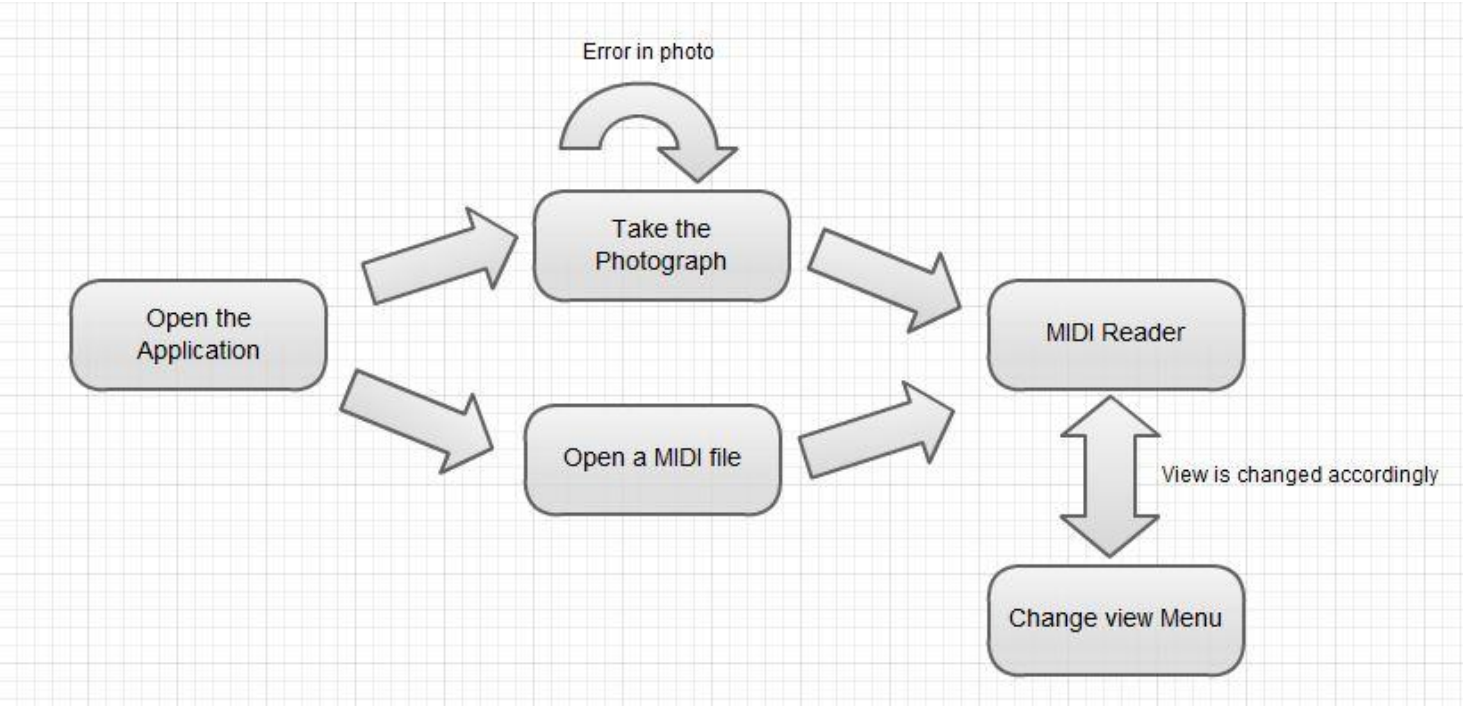


Figure:16

3.2.2.2 STIMULUS/RESPONSE SEQUENCES

3.2.2.2.1 Basic Data Flow 1

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to take a photograph of a printed sheet music.
4. User points and shoots the photograph.
5. Photograph is processed by the application and MIDI file is created.
6. MIDI reader is opened.
7. User opens the menu and selects change view.
8. User selects which part he/she wants to see.
9. View is changed accordingly.

3.2.2.2.1 Basic Data Flow 2

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to open a MIDI file to read.
4. MIDI reader is opened.
5. User opens the menu and selects change view.
6. User selects which part he/she wants to see.
7. View is changed accordingly.

3.2.2.3 FUNCTIONAL REQUIREMENTS

- The system should use the screen properly. All items should fit in the screen and must be scaled to fill the empty spaces.
- User should have full freedom on objects. All combinations of three items should be possible to select.
- System shouldn’t let user to select none of the three items. At least one item should be selected at all times.

3.2.3 Instrument Selection

3.2.3.1 BACKGROUND INFORMATION

Instrument selection is one of the important features of the application. After processing the image or opening a MIDI file to read user can listen a part of the song or the whole song. In this part it is quite important to have a realistic sound. Thanks to instrument selection feature of the application, user can select one of the instruments which are supported by the application to listen the part with the sound

of that instrument. There may be various pre-defined instruments such as; guitar, bass guitar, piano etc.

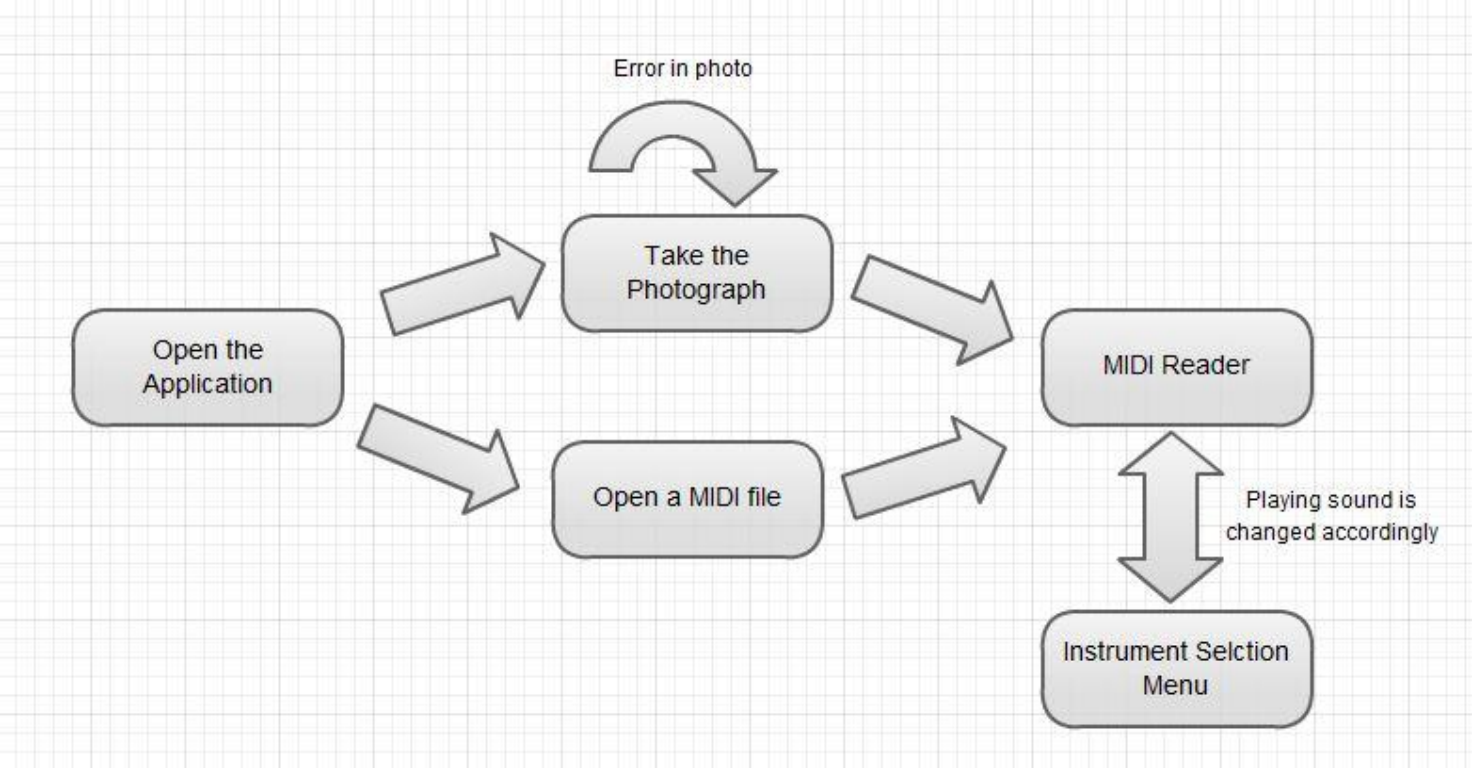


Figure:17

3.2.3.2 STIMULUS/RESPONSE SEQUENCES

3.2.3.2.1 Basic Data Flow 1

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to take a photograph of a printed sheet music.
4. User points and shoots the photograph.
5. Photograph is processed by the application and MIDI file is created.
6. MIDI reader is opened.
7. User opens the menu and selects instrument selection menu item.
8. User selects one of the possible instruments.
9. Play sound is changed accordingly.

3.2.3.2.1 Basic Data Flow 2

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.

3. User selects to open a MIDI file to read.
4. MIDI reader is opened.
5. User opens the menu and selects instrument selection menu item.
6. User selects one of the possible instruments.
7. Play sound is changed accordingly.

3.2.3.3 FUNCTIONAL REQUIREMENTS

- If keyboard view is selected, after changing the instrument, keyboard image should be changed accordingly.
- Automatic instrument initialization should be done according to the clef. If there is more than one defined instrument for a clef there should be a default value.

3.2.4 Note Edit

3.2.4.1 BACKGROUND INFORMATION

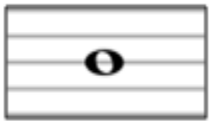
Note Edit function enables the users to edit the musical symbols and notes as they want. After processing the image or opening the MIDI file, selecting the note edit mode provides the user with a menu. Notes will be indicated by the symbols “A”, “B”, “C”, ”D”, “E”, “F” and “G”. The note and rest values will be changed by using a button which increases or decreases the value of the specified symbol. After choosing the edit mode and what to change by just pointing it on the screen, user can also change clefs, accidentals, time signatures, note relationships, dynamics, articulation marks, octave marks, repetitions and codas. This feature enables user to play the music in a way he/she prefers.

In order to explain the meanings of the each symbol read, musical symbols which are recognizable are given with their roles below.

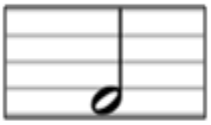
<u>Notes recognizable by the application:</u>

These notes differ from each other by their values. While processing the images corresponding to these notes, each one is vocalized according to their duration which is indicated below.

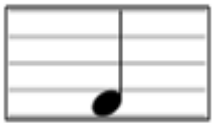
Double Whole Note



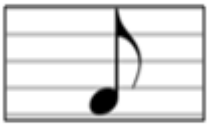
Whole Note



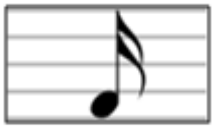
Half Note



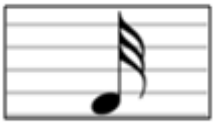
Quarter Note



Eighth Note

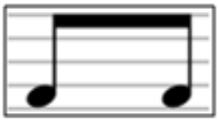


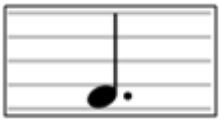
Sixteenth Note

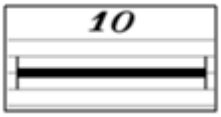


Thirty-Second Note

Sixty-Fourth Note


Beamed Note


Dotted Note



Multi-Measure Rest

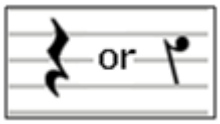
Rests recognizable by the application:

Similar to the notes, rests also differ from each other by their values. While processing the images corresponding to the rests, each one is vocalized according to their duration which is indicated below.

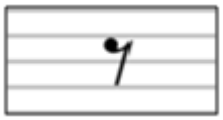

Double Whole Note

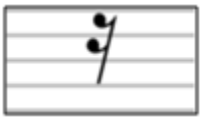

Whole Note

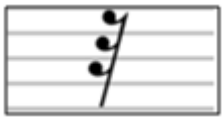

Half Note

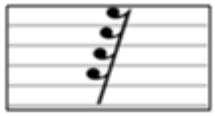


Quarter Note


Eighth Note


Sixteenth Note


Thirty-Second Note

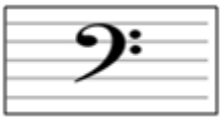

Sixty-Fourth Note

Clefs recognizable by the application:

There are five clefs which are supported by the application. They are the leftmost symbol which appears on the staff. It is important to recognize them also because the pitch range of the staff is determined according to type of the clef used.


G Clef


C Clef

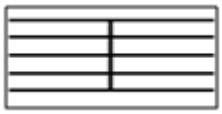

F Clef

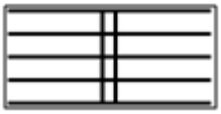
Lines recognizable by the application:

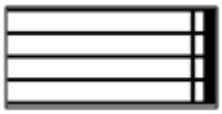
All the musical symbols are placed on the staffs indicated below. Therefore, in order to read all the notes correctly, it is necessary to recognize the lines. The types shown below have different meanings. While the first one is an ordinary staff which is used just to place the musical symbols, *ledger lines* are used to extend the pitches, *accolade* indicates the portion of the music which is played simultaneously, *dotted barline is used to divide* the long measures into shorter segments just for the ease of reading, bar *line* is used to separate the measures, *double barline* separates the two sections of the music and *bold double barline* is used to indicate the end of the entire composition.


Staff


Ledger Lines

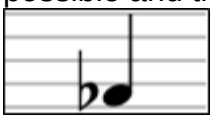

Bar Line

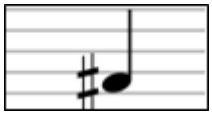

Double Barline

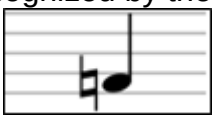

Bold double Barline

Accidentals recognizable by the application:

Accidentals are used to change the pitch of the notes which follows them. Using a few of them is also possible and this can also be recognized by the software.

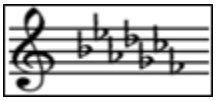

Flat

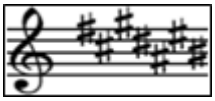

Sharp


Natural

Key Signatures recognizable by the application:

Key Signtures allows the composer not to use the accidentals for many notes. They determine the prevailing key of the music.

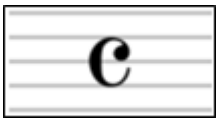

Flat Key Signature

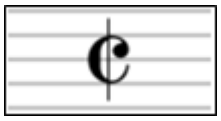

Sharp Key Signature

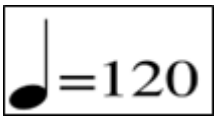
Time signatures recognizable by the application:

Music is divided into parts uniformly. *Time signatures* determine the number of beats in these sections.


Specific Time


Common Time


Cut Time

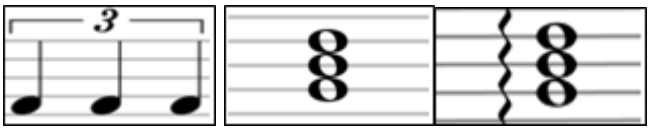


Metronome Mark

Note Relationships recognizable by the application:

Note relationships determine the way in which the notes are played. It is also important to extract these symbols from the whole image correctly to play the printed music sheet as accurate as possible.

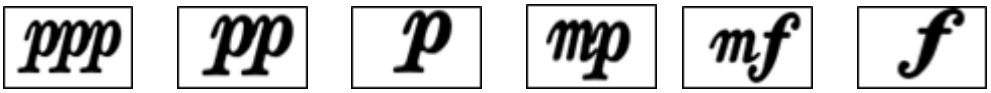
Tie Slur Slur



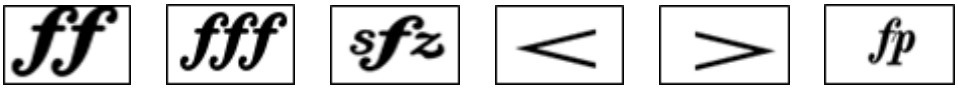
Tuplet Chord Arpeggiated chord

Dynamics recognizable by the application:

Dynamics indicate the intensity of the music. They determine whether the notes will be played loudly, softly, etc.



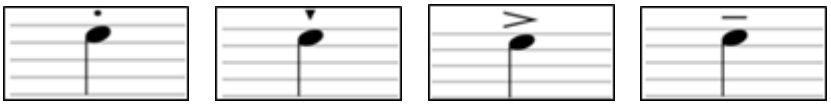
Pianississimo Pianissimo Piano Mezzo Pian Mezzo Forte Forte



Fortissimo Fortissisimo Sforzando Crescendo Diminuendo Forte-Piano

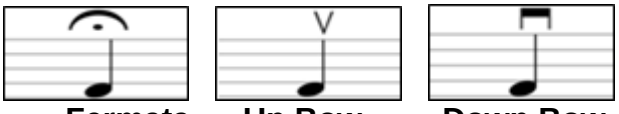
Articulation marks recognizable by the application:

Articulation mark determines the way in which the individual notes are played within a phrase or passage.



Staccato Staccatissimo Accent Tenuto

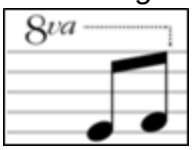
Marcato Stopped Note Open Note Pause



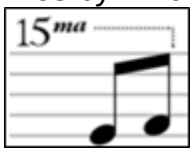
Fermata Up Bow Down Bow

Octave signs recognizable by the application:

Octave signs determines by which octave higher the passage should be played.



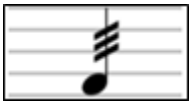
Ottava



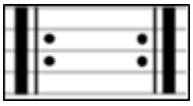
Quindicesima

Repetition and codas recognizable by the application:

Repetitions and codas determine how many times a specific part of the composition should be played. Interpreting them correctly has a significance to make the repetitions at correct time.

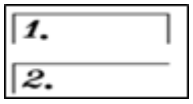


Tremolo

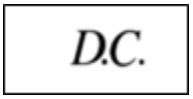


Repeat Signs

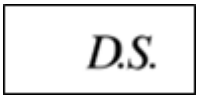
Simile Marks



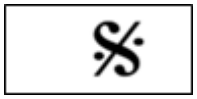
Volta Brackets



Da Capo



Dal Segno



Segno

Coda

By choosing the note edit option, each musical symbols can be changed into another symbol.

All the figures are taken from: http://en.wikipedia.org/wiki/List_of_musical_symbols

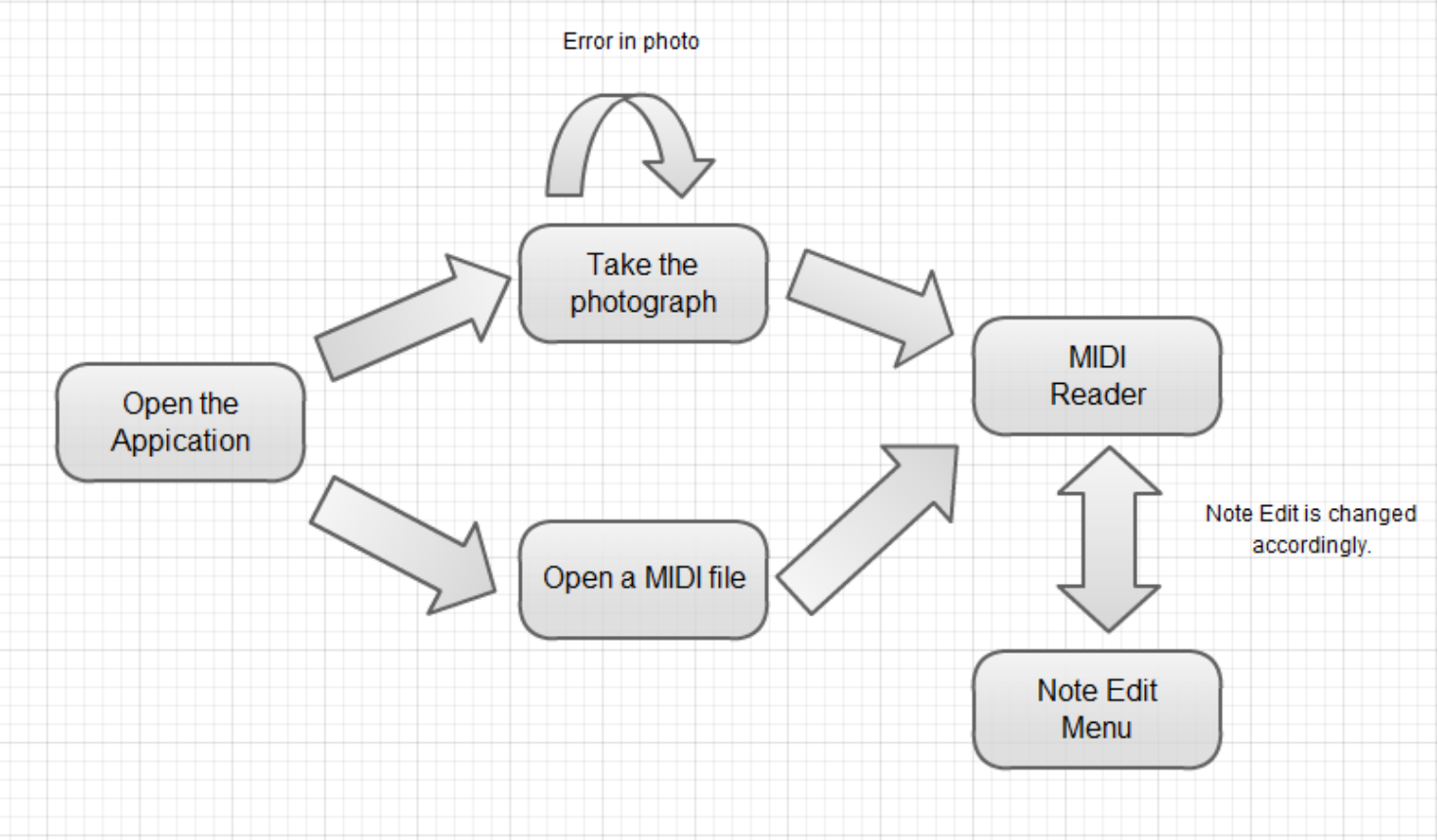


Figure:18

3.2.4.2 STIMULUS/RESPONSE SEQUENCES

3.2.4.2.1 Basic Data Flow 1

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to take a photograph of a printed sheet music.
4. User points and shoots the photograph.
5. Photograph is processed by the application and MIDI file is created.
6. MIDI reader is opened.
7. User opens the menu and selects the note edit menu item.
8. Selected notes or musical symbols are changed accordingly.

3.2.4.2.2 Basic Data Flow 2

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to open a MIDI file.
4. MIDI reader is opened.

5. User opens the menu and selects the note edit menu item.
6. Selected notes or musical symbols are changed accordingly.

3.2.4.3 FUNCTIONAL REQUIREMENTS

- If a note or a rest is changed and after this change measure of a bar or more than one bars is broken than user gets an error message but the part is played anyway.

3.2.5 Play/Pause

3.2.5.1 BACKGROUND INFORMATION

Play/Pause is one of the most essential functions of the application.It simply plays the created MIDI track with chosen visual augmentations. This feature enables the user to play the MIDI track, see it on the digital sheet with an indicator of the current note and/or follow the keyboard fret or positions at the same time. In addition to these, the user can set the starting point with the finger move on the sheet. If the track is not being played that moment play/pause procedure starts the process from the starting point which is specified by the user. If the track is being played, play/pause procedure stops the track both in visual and audio side.

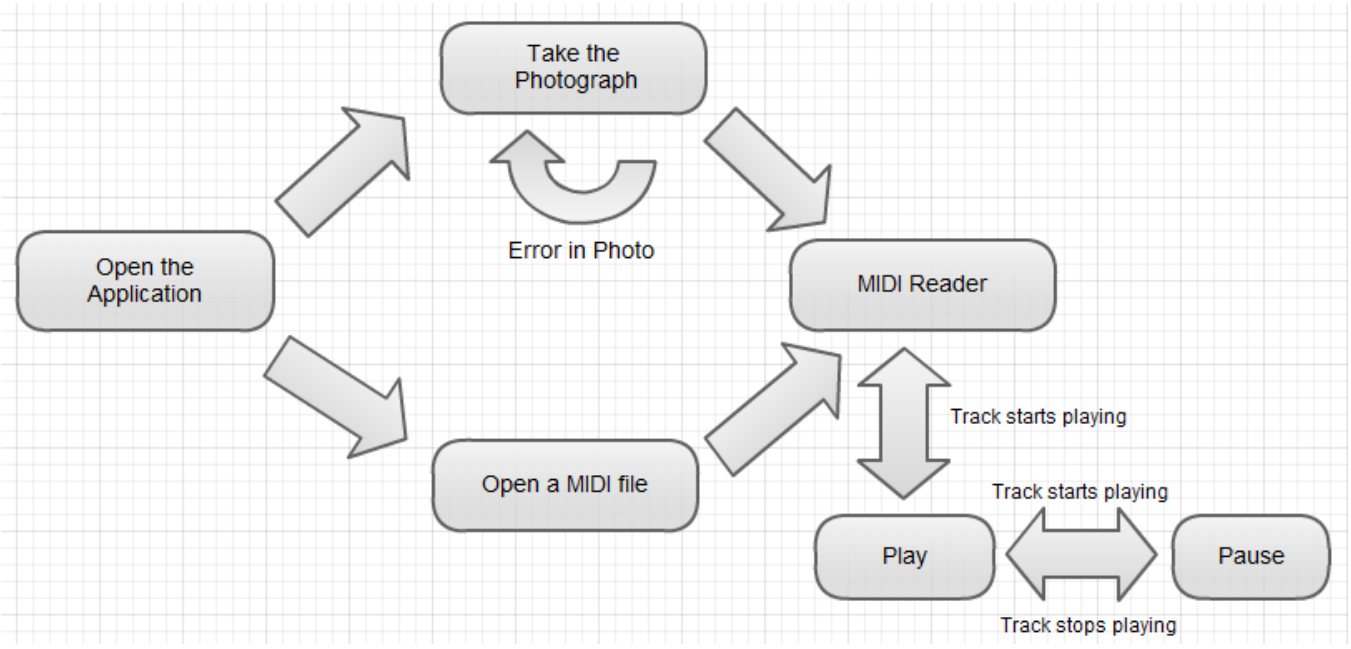


Figure:19

3.2.5.2 STIMULUS/RESPONSE SEQUENCES

3.2.5.2.1 Basic Data Flow 1

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to take a photograph of a printed sheet music.
4. User points and shoots the photograph.
5. Photograph is processed by the application and MIDI file is created.
6. MIDI reader is opened.
7. User specifies the starting point with the selection through sheet.
8. User presses play/pause button on the screen.
9. The track starts playing to track both in audio and visual frames.

3.2.5.2.2 Basic Data Flow 2

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to open a MIDI file to read.
4. MIDI reader is opened.
5. User specifies the starting point with the selection through sheet.
6. User presses play/pause button on the screen.
7. The track starts playing to track both in audio and visual frames.

3.2.5.2.2 Alternative Data Flow 1

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to take a photograph of a printed sheet music.
4. User points and shoots the photograph.
5. Photograph is processed by the application and MIDI file is created.
6. MIDI reader is opened.
7. User specifies the starting point with the selection through sheet.

8. User presses play/pause button on the screen.
9. The track starts playing to track both in audio and visual frames.
10. User presses play/pause button on the screen.
11. The track stops playing the song.

3.2.5.2.2 Alternative Data Flow 2

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to open a MIDI file to read.
4. MIDI reader is opened.
5. User specifies the starting point with the selection through sheet.
6. User presses play/pause button on the screen.
7. The track starts playing to track both in audio and visual frames.
8. User presses play/pause button on the screen.
9. The track stops playing the song.

3.2.5.3 FUNCTIONAL REQUIREMENTS

- If the keyboard view is selected, after changing the key of the track with transpose feature, indicators appear on the keyboard should be altered appropriately.
- If the sheet music and/or tab view is selected, after changing the key of the track with transpose feature, notes/tabs should be altered with the new ones corresponds to the transposed notes.
- If transpose feature is not used, the track should keep the original key.

3.2.6 Transpose

3.2.6.1 BACKGROUND INFORMATION

Transpose feature is a very useful tool of the application. After obtaining the MIDI track and its sheet with preprocessing, in the player interface, user can define the transposed key of the track. This feature enables the user to hear the song with different tonalities or allows him/her to study with the tuning of his/her instrument.

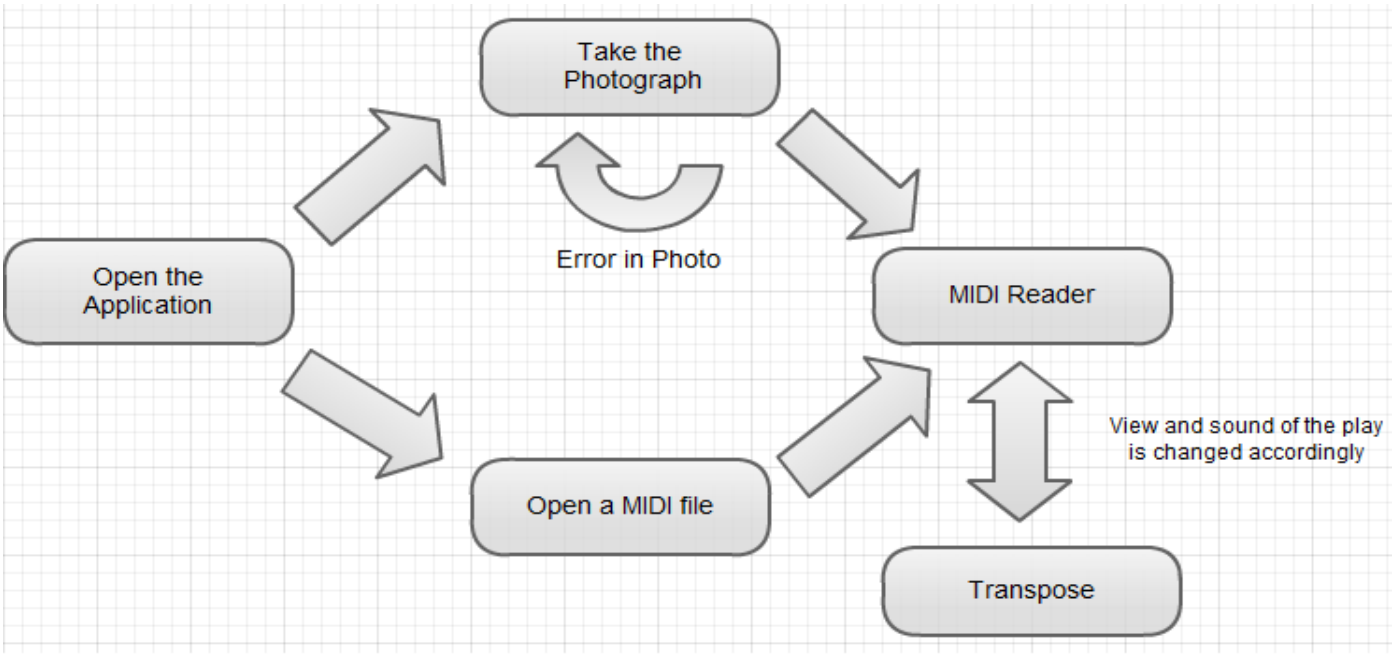


Figure:20

3.2.6.2 STIMULUS/RESPONSE SEQUENCES

3.2.6.2.1 Basic Data Flow 1

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to take a photograph of a printed sheet music.
4. User points and shoots the photograph.
5. Photograph is processed by the application and MIDI file is created.
6. MIDI reader is opened.
7. User opens the menu and selects transpose menu item.
8. User selects the new key of the song which he/she desires.
9. Play sound and view sheet is changed accordingly.

3.2.6.2.2 Basic Data Flow 2

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.

3. User selects to open a MIDI file to read.
4. MIDI reader is opened.
5. User opens the menu and selects transpose menu item.
6. User selects the new key of the song which he/she desires.
7. Play sound and view sheet is changed accordingly.

3.2.6.3 FUNCTIONAL REQUIREMENTS

- If the keyboard view is selected, after changing the key of the track with transpose feature, indicators appear on the keyboard should be altered appropriately.
- If the sheet music and/or tab view is selected, after changing the key of the track with transpose feature, notes/tabs should be altered with the new ones corresponds to the transposed notes.
- If transpose feature is not used, the track should keep the original key.

3.2.7 Clef Change

3.2.7.1 BACKGROUND INFORMATION

Transpose feature is a very useful tool of the application. After obtaining the MIDI track and its sheet with preprocessing, in the player interface, user can define the transposed key of the track. This feature enables the user to hear the song with different tonalities or allows him/her to study with the tuning of his/her instrument.

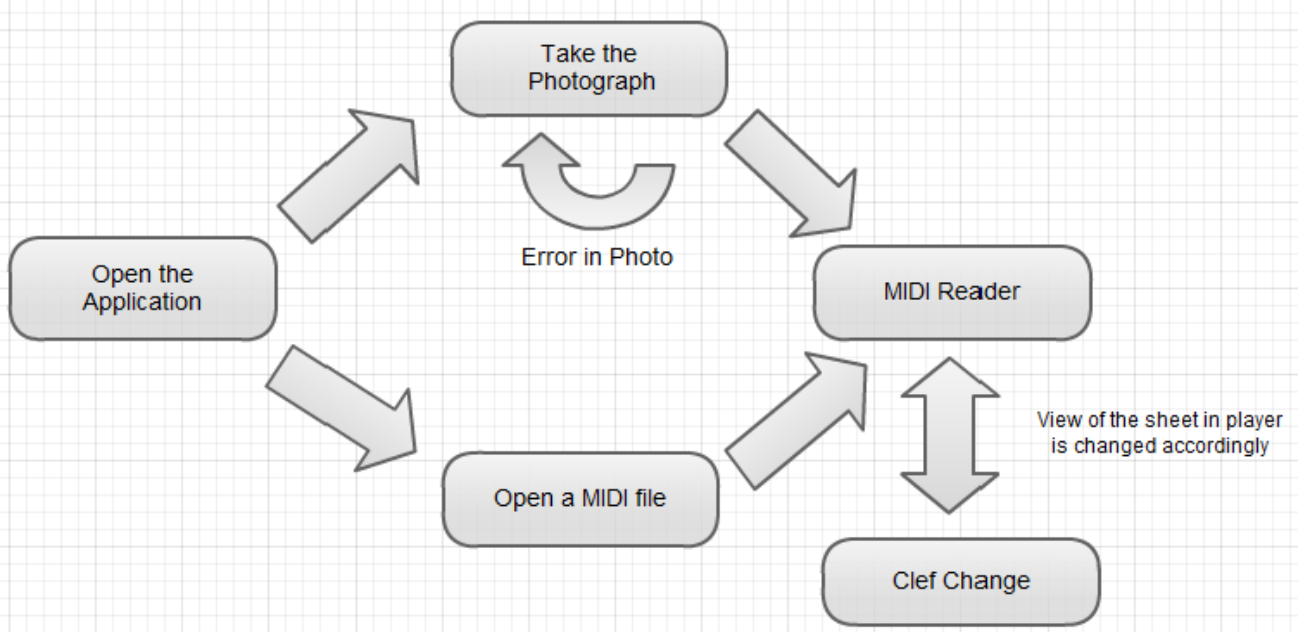


Figure:21

3.2.7.2 STIMULUS/RESPONSE SEQUENCES

3.2.7.2.1 Basic Data Flow 1

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to take a photograph of a printed sheet music.
4. User points and shoots the photograph.
5. Photograph is processed by the application and MIDI file is created.
6. MIDI reader is opened.
7. User opens the menu and selects transpose menu item.
8. User selects the new key of the song which he/she desires.
9. View sheet is changed accordingly.

3.2.7.2.2 Basic Data Flow 2

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to open a MIDI file to read.
4. MIDI reader is opened.

5. User opens the menu and selects transpose menu item.
6. User selects the new key of the song which he/she desires.
7. View sheet is changed accordingly.

3.2.7.3 FUNCTIONAL REQUIREMENTS

- If the sheet music view is selected, after changing the key of the track with transpose feature, notes should be altered with the new ones corresponds to the new clef.
- If clef change feature is not used, the track should keep the original clef of the sheet.
- Clef change only affects the way of writing the notes on the sheet. It should not affect the tonality and sound at all.

3.2.8 Load Midi

3.2.8.1 BACKGROUND INFORMATION

Standard MIDI files provide a common file format used by most musical software and hardware devices to store song information including the title, track names. Almost every software music sequencer is capable of loading and saving standard MIDI files. Importing the MIDI file loads MIDI region data (note, controller, pitchbend, specific meta events) into an opened project with global data (such as tempo tempo events, signature, chords, tracj names and so on) being ingnored.

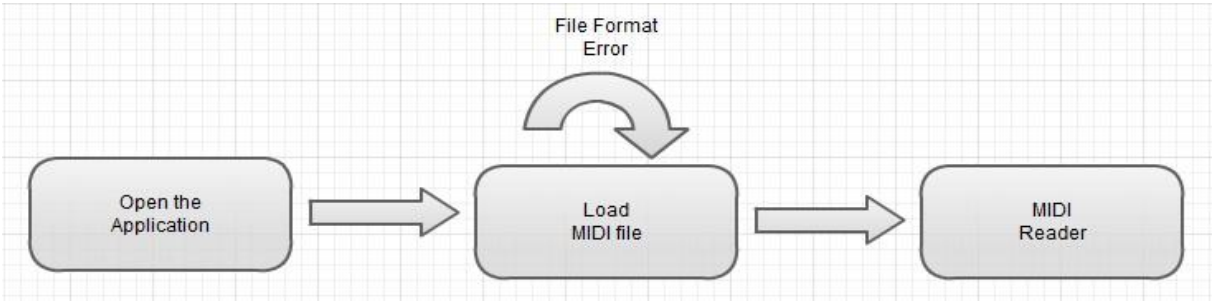


Figure:22

3.2.8.2 STIMULUS/RESPONSE SEQUENCES

3.2.8.2.1 Basic Data Flow

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. User selects to load a MIDI file.
4. MIDI reader is opened and the MIDI file is recognized.

3.2.8.2.2 FUNCTIONAL REQUIREMENTS

- Any MIDI formatted file (.mid) should be easily processed and used by our application, thanks to the application's MIDI Reader Module.
- The system will have to check if the appropriate file format (.mid) is being loaded If the file fomrat is not suported then the error message should be given.

3.2.9 SAVE AS

3.2.9.1 BACKROUND INFORMATION

Saving files in MIDI format will obviously be one of the crucial features. Usually the MIDI files are loaded into some form of 'player' (software or hardware), which is MIDI Reader in our case, and the final *sound* is then produced by a sound-engine that is connected to or that forms part of the player. Musical performances are not usually created as .mid files; rather a composition is recorded using some sequencer that saves MIDI data in it's own format. However, most if not all sequencers an 'Save As' or 'Export' as a Standard MIDI File.

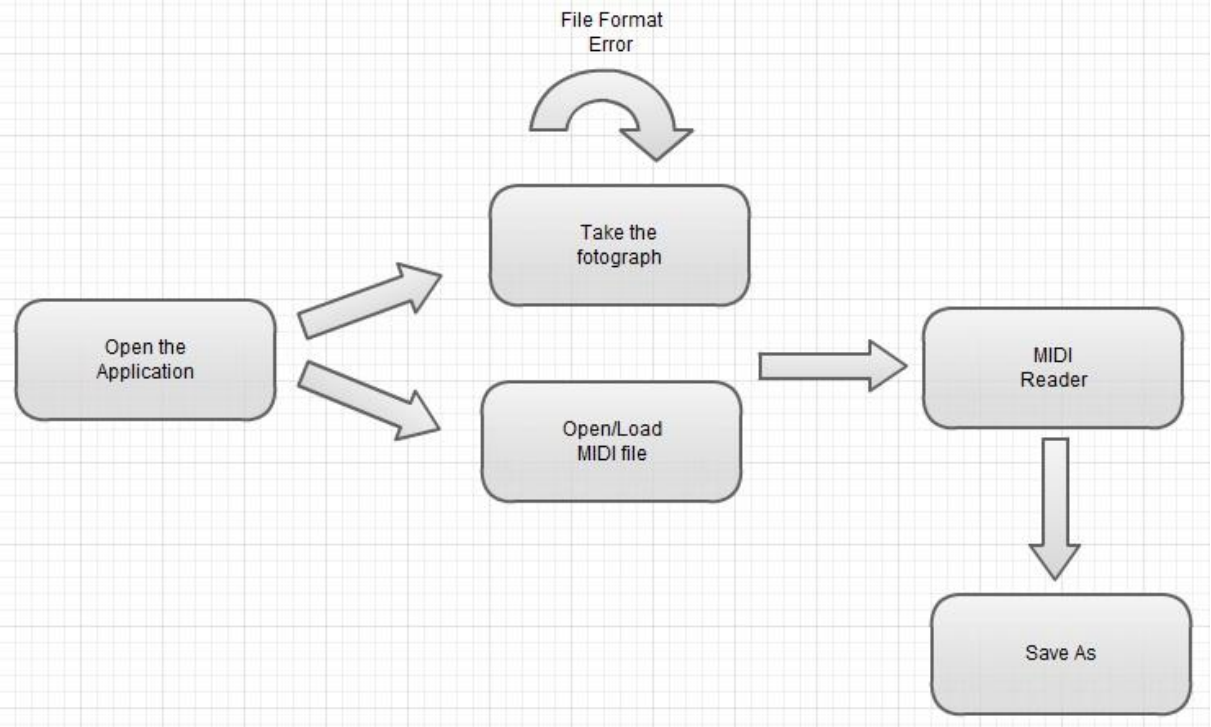


Figure:23

3.2.9.2 STIMULUS/RESPONSE SEQUENCES

3.2.9.2.1 Basic Data Flow

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. 3. If user chooses to take a photograph then she/he points and shoots the photograph.
4. Photograph is processed by the application and MIDI file is created.
5. MIDI reader is opened.
6. User opens the menu and selects ‘Save As’ to save the file.

3.2.9.2.2 FUNCTIONAL REQUIREMENTS

- User should be able to save the file that was loaded.
- User should be also able to save the file if he/she has edited it.

3.2.10 EXPORT

3.2.10.1 BACKGROUND INFORMATION

The files created in our application will be saved and exported in order to be played on other devices.

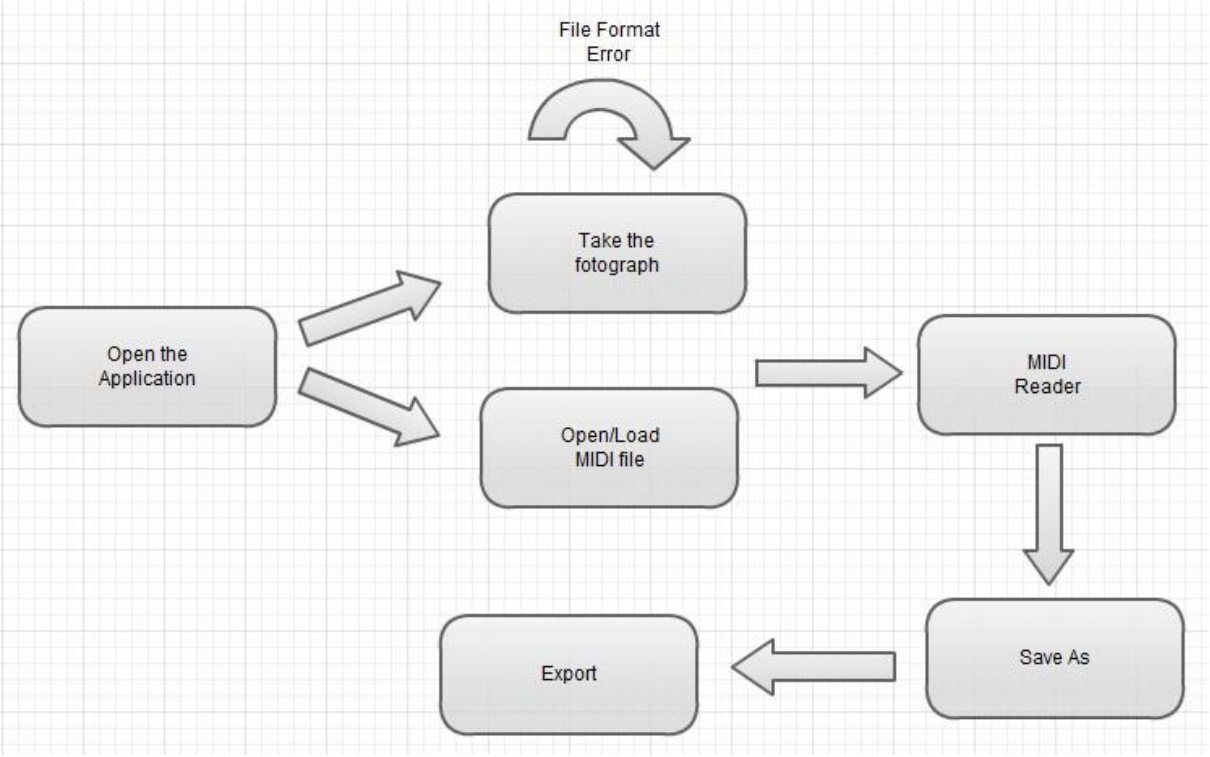


Figure:24

3.2.7.2 STIMULUS/RESPONSE SEQUENCES

3.2.7.2.1 Basic Data Flow

1. User opens the application on their mobile device.
2. User is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
3. If user chooses to take a photograph then she/he points and shoots the photograph.
4. Photograph is processed by the application and MIDI file is created.
5. MIDI reader is opened.
6. User opens the menu and selects ‘Save As’ to save the file.
7. User chooses ‘Export’ to export the file.

3.2.7.2.2 FUNCTIONAL REQUIREMENTS

- User should be able to export the saved file by choosing ‘Export’ from the menu.
- The file should be exported in portable and readable format, such as pdf.

3.3 Non-functional Requirements

3.3.1 Performance requirements

Since the application will run on mobile devices, performance of the algorithms used becomes an important issue. Most critical part is designing a very efficient image processing algorithm on images. Image processing techniques are usually very costly operations. They may need high CPU power. However, mobile device that runs the application may not have a very powerful CPU. This scenario is quite expected. Thus, designing a very efficient image processing algorithm is crucial. Otherwise, processing the may take too long and program becomes useless.

Second important issue on performance is related with MIDI player module. Our application can play MIDI formatted sound and at the same time it shows the note sequence corresponding that sound. It is very important that these two streams are synchronized. So, there shouldn't be any delays on playing the sound and displaying notes. Implementation of these parts must satisfy this requirement.

Possibility of low CPU powered device should be considered all the time.

Furthermore, lack of enough available RAM space may be a problem. Mobile device may not have large amount of RAM storage and the application may need a big available space on memory, because it processes relatively high resolution images. Available RAM space should be checked all the times.

Number of simultaneous users and amount of the data processed are not big issues for the application since, there can be only one user using the program at any times and the amount of the information to be processed can not exceed the size of an A4 page full of notes.

3.3.2 Design constraints

The application can only run on Android OS.
Only Java programming language can be used since Android systems only allows Java written codes.
To use the application the device must have a built-in camera. Application should access the camera to take a photograph and to load it.
Mobile device must have an SD card inserted to it and the application must have privileges to access (read/write) to SD card of the mobile device, since it will read and write MIDI files to SD card.

4 Data Model and Description

4.1 Data Description

In this section, the data objects and the relations between them are explained briefly. In addition to those complete data model and a data dictionary will be provided. Each class of our application and usage of these will be described.

4.1.1 Data objects

4.1.1.1 Main Class

Main Class is responsible for executing the main procedure and forming a roof for the other classes defined in order to operate main features of the application. It will contain an Image and Player Class instances as attributes which will be described later on this document. Since the application requires a lot of interaction with the user via GUI, Main Class carries out this continuing operation.

state: Indicator of the state of the application.

4.1.1.2 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.

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

4.1.1.3. Player Class

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

sheet: Contains the sheet object of the selected image object or MIDI file.
midi: Constructed built-in MIDI object from the specified MIDI file that is given or produced via Image object.

4.1.1.4. 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.

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.

4.1.1.5. 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.
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.

4.1.1.6. 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.
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.

4.1.2 Relationships

Main - Image: There will be one main class for every run of the application. Main class will have one Image object as an attiribute. This object may be empty if user selects to open a MIDI file directly. Main class carries out image processing operation on this Image object.

Main - Player: Main class will have one player object as an attribute. This object can not be empty. Player objects operates on notes. Shows the required note stream on the screen and plays/pauses/stops the MIDI file.
Player - Sheet: Every Player object (one for each application run) will have one Sheet object as an attribute. Sheet object represents real-world printed sheet item. It contains every information about this particular music passage. Player object will use this information to play the music properly.
Sheet - Bar: Every Sheet object has one or more Bar objects. It depends the length of the part user wants to listen. Sheet basically consists of Bars.
Bar - Note: Every Bar object has one or more Note objects. Number of the Note objects depends on the measure of the song. Bar consists of Notes.

4.1.3 Complete data model

This section describes a complete data model merging data object descriptions with relationships explained in previous sections. Figure below shows the entity relationship diagram of the data model which provides a conceptual representation of data. The relationships between data objects provide both relational information.

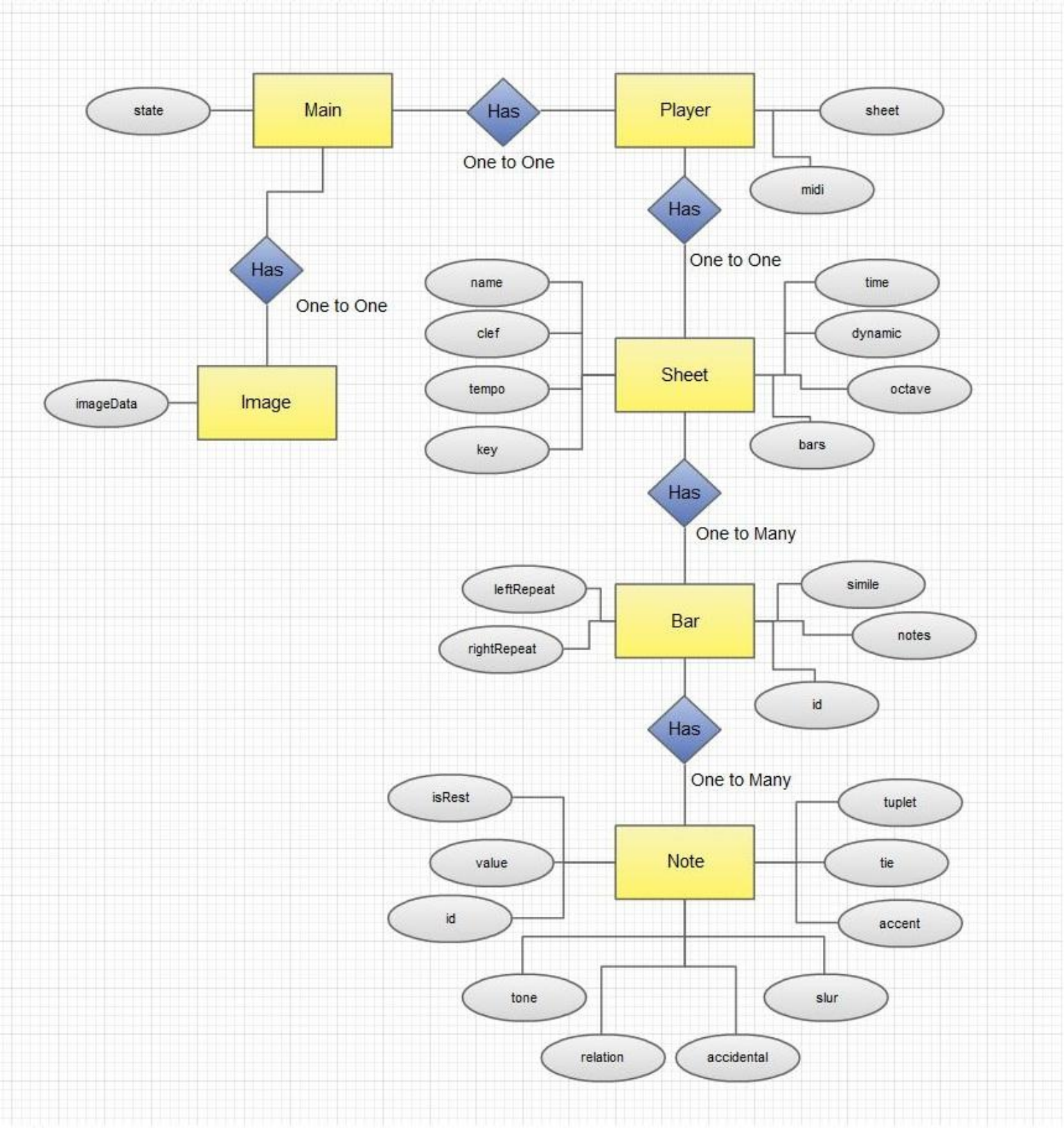

state
Main
Has
One to One
Player
sheet
midi
Has
One to One
Has
One to One
name
clef
tempo
key
Sheet
time
dynamic
octave
bars
imageData
Image
Has
One to Many
leftRepeat
rightRepeat
Bar
simile
notes
id
Has
One to Many
isRest
value
id
Note
tuplet
tie
accent
tone
slur
relation
accidental

Figure:25

5 Behavioral Model and Description

5.1 Description for software behavior

The purpose of the application being implemented is to convert some sheet music piece into digital format. So the major events that could be noticed are as follows:

- The user first opens the application.
- Then the user is asked to select between two options to run; either to open a MIDI file to read or to take a photograph of a printed sheet music.
- In first case, the fotograph of sheet music taken by the user is accessed by the application and loaded.
- The sheet music is converted to MIDI format by Optical Music Recognition(OMR)
- In the second case, the MIDI file is loaded
- MIDI Reader Module takes the MIDI formatted music and provides MIDI reading facilities which allow MIDI to convert sheet music to its digital equivalence.

5.2 State Transition Diagrams

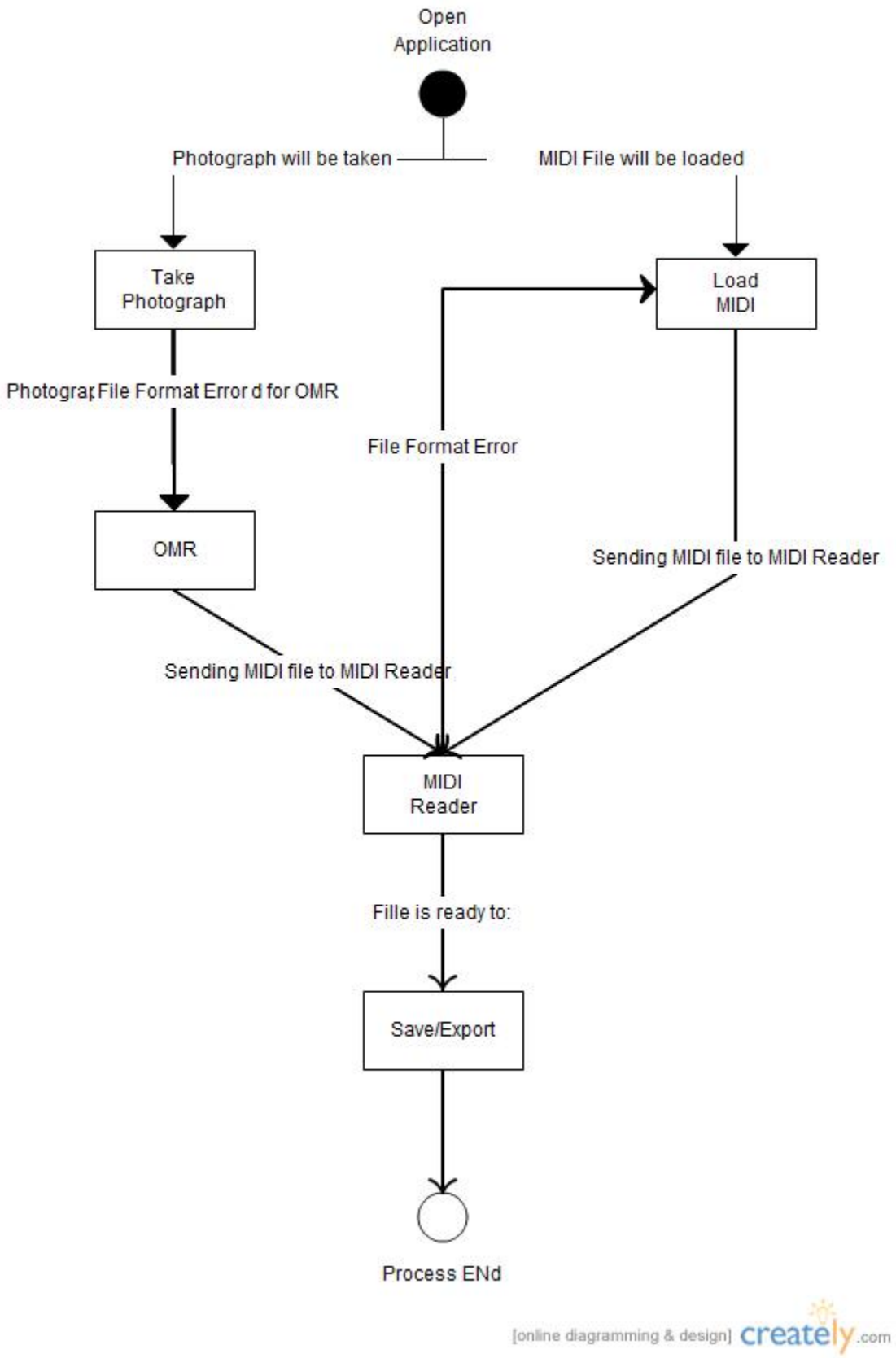

Open
Application
Photograph will be taken
MIDI File will be loaded
Take
Photograph
Load
MIDI
Photograp File Format Error d for OMR
File Format Error
OMR
Sending MIDI file to MIDI Reader
Sending MIDI file to MIDI Reader
MIDI
Reader
Fille is ready to:
Save/Export
Process ENd
[online diagramming & design] creately.com

Figure:26

6. Planning

6.1 Team Structure

The team “GOBIT”, has 2 weekly meetings on Monday and on Friday to discuss about the progress of the project and to provide information exchange. Before starting to work, there is an equal job distribution between team members and after everyone completes the job given, there are meetings in which each team member explains his/her own job to others.

Because this project is not a sponsored one, when needed, academicians in the department of computer engineering in METU will be asked for help. In addition to meetings indicated above, there are weekly meetings with assistant Serdar Çiftçi every Thursday.

6.2 Estimation (Basic Schedule)

“GOBIT”, the project team aims to prepare the detailed design report and demo until the end of the semester.
First term schedule is given in Figure ? and second term schedule is Figure ?+1:).

First term schedule:

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:27

Second term schedule:

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
Inplementing 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:28

6.3 Process Model

As a software development process, spiral model that combines the elements of both the design and prototyping-in-stages. The aim to use that model is to take the advantage of combining top-down and bottom up concepts. [1]

Spiral model used in the project is shown in Figure 29:

Figure:29

[1]http://en.wikipedia.org/wiki/Software_development_methodology

7 CONCLUSION

This analysis report gives information about the approach of team “GOBIT” for the design of “DigiMuse”. After indicating the problem definition and the objective, an overall description is made through product perspective, product functions and constraints of the project. Then, specific requirements and data model and description is explained. After describing the behavioral model, scheduling is also included in the document. This specification will hopefully constitute the basis for design, development, and testing of the project.