<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Gamify piano project</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Lai, Tsz Kit (黎子杰)</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>Lai, T. K. (2016). Gamify piano project (Outstanding Academic Papers by Students (OAPS)). Retrieved from City University of Hong Kong, CityU Institutional Repository.</td>
</tr>
<tr>
<td><strong>Issue Date</strong></td>
<td>2016</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/2031/8730">http://hdl.handle.net/2031/8730</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>This work is protected by copyright. Reproduction or distribution of the work in any format is prohibited without written permission of the copyright owner. Access is unrestricted.</td>
</tr>
</tbody>
</table>
Student Name: LAI Tsz Kit
Student No.:
Programme Code: BSC2
Supervisor: NGO, Chong Wah
1st Reader: WONG, Tsui Fong Helena
2nd Reader: CHAN, Edward
Student Final Year Project Declaration

I have read the project guidelines and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I hereby declare that the work I submitted for my final year project, entitled:

Gamify Piano Project

does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Project Guidelines.

Student Name: Tsz Kit LAI  Signature: ____________________________

Student ID: ____________________________  Date: 11 July 2016
1. Abstract

The traditional ways to practice piano is no longer attractive and interesting enough for the current generation. There are numbers of new piano learning or practicing method is introduced, but none of them can really replace the traditional piano practicing method. In order to enhance the interest of piano practice, a piano video game is proposed. It will allow players to practice their piano skill when they are playing this game.

In this project, this idea will be implemented as a real application on iOS platform. Player has to connect their electronic piano with iPad through MIDI port. After establish the connection successfully, player can select any pieces of song they want to play or practice and start the game. Two game mode will be provided in this game, first one is online battle mode, player will be able to play the same song with someone who want to want to play it at the same moment and have a piano battle to see who play better. Also, an offline practice mode is provided for those who want to focus on practicing a song.

To keep the game being attractive to the old player, a customize system is designed. Player have to earn game points to buy a customize item to customize their own gaming environment. With designing different kind of customize item, it will enhance the motivation of player to play this game. And it will be a way for making them feels flesh even if they have played this game for a long time as there will be more and more customize item time by time so as to enhance the life-cycle of the game.

This piano video game project is not just providing an innovative and interesting way for piano practicing, but also establishing a new pianist's network for them to support and help each other, since this game support friend and chatting system. Thus, they can communicate with each other and know a new friend after having a piano battle.
2. **Acknowledgement**

I wish to express my sincere gratitude to City University of Hong Kong, especially, Department of Computer Science for permitting me to take such an interesting topic as my final year project. I express my deep sense of gratitude to Prof. NGO for encouraging me during the whole project. This project will not be completed without his guidance and encouragement. Finally, I want to thanks all the interviewees for spending their treasure time to attend my interview to give me their option and suggestions.

Tsz Kit LAI
## 3. Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ABSTRACT</td>
<td>1</td>
</tr>
<tr>
<td>2. ACKNOWLEDGEMENT</td>
<td>2</td>
</tr>
<tr>
<td>3. TABLE OF CONTENTS</td>
<td>3</td>
</tr>
<tr>
<td>4. INTRODUCTION</td>
<td>5</td>
</tr>
<tr>
<td>4.1 Problem Definition</td>
<td>5</td>
</tr>
<tr>
<td>4.2 Project Scope</td>
<td>6</td>
</tr>
<tr>
<td>4.3 Project Significance and motivation</td>
<td>7</td>
</tr>
<tr>
<td>5. LITERATURE REVIEW</td>
<td>7</td>
</tr>
<tr>
<td>5.1 Major alternatives of the problem</td>
<td>7</td>
</tr>
<tr>
<td>5.1.1 Suzuki piano teaching method</td>
<td>8</td>
</tr>
<tr>
<td>Limitation of Suzuki piano teaching</td>
<td>8</td>
</tr>
<tr>
<td>5.1.2 Piano video game</td>
<td>8</td>
</tr>
<tr>
<td>Limitation of Synthesia</td>
<td>9</td>
</tr>
<tr>
<td>Limitation of Piano Dust Busters and Piano Maestro</td>
<td>10</td>
</tr>
<tr>
<td>6. PROPOSED DESIGN, SOLUTION, SYSTEM</td>
<td>11</td>
</tr>
<tr>
<td>6.1 Gameplay Design</td>
<td>12</td>
</tr>
<tr>
<td>6.1.1 Game modes provided in this game</td>
<td>13</td>
</tr>
<tr>
<td>Auto play game mode</td>
<td>13</td>
</tr>
<tr>
<td>Practice mode</td>
<td>14</td>
</tr>
<tr>
<td>Challenge mode</td>
<td>14</td>
</tr>
<tr>
<td>6.1.2 Online battle</td>
<td>14</td>
</tr>
<tr>
<td>6.2 User Interface Design</td>
<td>15</td>
</tr>
<tr>
<td>6.2.1 Starting Screen User Interface</td>
<td>15</td>
</tr>
<tr>
<td>6.2.2 Main Menu User Interface</td>
<td>16</td>
</tr>
<tr>
<td>6.2.3 Gameplay User Interface</td>
<td>16</td>
</tr>
<tr>
<td>6.2.4 Chatting and friend system</td>
<td>19</td>
</tr>
<tr>
<td>6.2.5 Shop System</td>
<td>19</td>
</tr>
<tr>
<td>6.3 System</td>
<td>20</td>
</tr>
<tr>
<td>6.3.1 Major technical components</td>
<td>20</td>
</tr>
<tr>
<td>Server sides – Database</td>
<td>20</td>
</tr>
<tr>
<td>Server sides – Socket Program</td>
<td>20</td>
</tr>
<tr>
<td>Client sides – MIDI Unified 5</td>
<td>21</td>
</tr>
</tbody>
</table>
Client sides – Music XML ................................................................. 21

6.3.2 Use case diagram for gamify piano project ........................................... 22

7. DETAILED METHODOLOGY AND IMPLEMENTATION ........................................ 28

7.1 Server methodology .............................................................................. 28
  7.1.1 Data retrieval and management on server side database ......................... 29
  7.1.2 Command forwarding between users .................................................. 29
  Connection management ........................................................................ 29
  Battle pairing ......................................................................................... 30

7.2 Server Implementation ............................................................................ 32
  7.2.1 Database Design - Entity Relationship Diagram ..................................... 32
  7.2.2 Class Diagram .............................................................................. 35

7.3 Client Methodology ............................................................................... 37
  7.3.1 Level generation based on MusicXML .................................................. 37
  7.3.2 Note(Color bar) generation ................................................................. 39
  Object pooler approach ........................................................................... 40
  7.3.3 Hit Determination ........................................................................... 41
  7.3.4 Keyboard Input ............................................................................... 41

7.4 Client Implementation ............................................................................ 42
  7.4.1 Summary of software design ............................................................... 42
  7.4.2 Class diagram .................................................................................. 43
  Class description .................................................................................... 44
  7.4.3 Unique programming approach – Event System .................................... 48
  Advantage of using this approach ............................................................ 49
  7.4.4 Unique programming approach – Communication System ................... 49
  Problem encountered: ........................................................................... 49
  Delegation design pattern: .................................................................... 49

8. USER EXPERIENCE EVALUATION .......................................................... 51

8.1 Surveys .................................................................................................. 51
  First interview ......................................................................................... 52
  Interview background information: .......................................................... 52
  Result: .................................................................................................... 52
  Second Interview ................................................................................... 53
  Background information: ...................................................................... 53
  Result: .................................................................................................... 54

9. SUMMARY OF ACHIEVEMENTS .............................................................. 55

10. LIMITATIONS ......................................................................................... 56

Final Year Project – Final Report
4. **Introduction**

4.1 **Problem Definition**

Piano practicing is a time consuming and boring activity to most of the children and teenagers, since it is a must for them to spend tons of time to do lots of boring exercises to improve their sight-reading skill, keeping even beats and the finger flexibility.

According to “10 teachers viewpoints on Suzuki piano”, the dropout rate of traditional piano program in UK is about 80%, after the students have learnt piano two or three years[1]. In Hong Kong, in order to force or encourage their children to practice piano, some parents will hire someone who accompany their children to practice piano[2]. These are signals telling us that the traditional way to practice piano skill is no longer suitable for the children and teenager anymore.

At the meantime, children and teenagers are willing to practice their skill for playing music game such as “Taiko on Tatsujin”, “《jubeat》”, “Groove Coaster”, “Guitar Hero” etc… And they also have to spend tons of time on practicing in order to become a master of the game. It proofs that the reason behind children not willing to practice piano is not about taking too much time to practice, it is because the practicing method is just too boring.

Therefore, I believe that gamify piano can help to solve this problem, as it can enhance the interest in piano practicing.
4.2 Project Scope

This project will include the following items:

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gameplay design</td>
<td>Studying the way to gamify a piano and design a totally new interesting gameplay which can help player to practice their piano skill.</td>
</tr>
<tr>
<td>Convert MusicXML to the piano game</td>
<td>Reading the MusicXML file and render all the corresponding music notes to the game scene.</td>
</tr>
<tr>
<td>Rate for the player performance</td>
<td>This game should be able to analysis the accuracy of the piano playing. The analysis will be based on the accurate on pitch and beats so 5 as to give the player a score.</td>
</tr>
<tr>
<td>Online piano battle</td>
<td>The system will be able to match up the player who are going to playing the same song at the same time and let them to have a battle. In the battle, player will be able to see how the opponent play this song in real-time and they can view the opponent score as well.</td>
</tr>
<tr>
<td>Leaderboard</td>
<td>In this game, there will be two kinds of leaderboard for each song. First one is “World Rank leaderboard”. It will show the top ten players for this song. Also, your world ranking will be displayed as well. The second one is the personal leaderboard, it will display your top records for playing this song.</td>
</tr>
<tr>
<td>Friend System</td>
<td>In this game, player will be able to add someone as his/her friend by clicking a “Add friend” button. Once the other one accept the request, they will be recorded as a friend in this game. And, they can chat with others anytime.</td>
</tr>
</tbody>
</table>
Able to connect with the digital piano | This game will be able to connect to a digital piano using a MIDI port to lightning connector. And the game will be able to handle the signal sent from the MIDI port.
---|---
Chatting | In this game, a simple chatting system is provided for player to chat with his or her friend any time.

4.3 Project Significance and motivation

This project would be a review on the existing piano practicing method present, and to explore the possibility of improving piano practicing method with information technology. Further, it is expected to provide a totally innovative and interesting gamify piano practicing method that attractive for most of the children and teenagers. This project would be beneficial to all the children and teenagers who are learning piano and think that piano practicing is boring, as this project will make piano practicing become much more fun than before. Furthermore, it would also beneficial to those parents who want their children to practice piano more, as it will enhance the motivation of children to practice piano. This would expectedly to make more children in learning piano and establishing a pianist's network to make them support and help each other's.

5. Literature review

5.1 Major alternatives of the problem

Despite the fact that most of the people learning piano in the traditional way, there are mainly two alternatives of this problem is introduced. Following are the alternatives of this problem:

1. Suzuki piano teaching
2. Piano video game
5.1.1 Suzuki piano teaching method

According to “Traditional & Suzuki Teaching”, Suzuki teaching is introduced in 1945 by “Shin'ichi Suzuki ”, which named “Talent Education” at that time. This method is about how to creating stimulating musical surroundings and excellent training. And there is some statistics [1] shows that comparing with the traditional piano teaching method, the dropout rate of Suzuki piano teaching program is very low. The main differences between traditional piano teaching method and Suzuki piano teaching method are as follow [3]:

1. Suzuki-trained students begin at very young ages (3-10 years old)
2. Suzuki teacher have to establish a partnership with the parent.
3. Suzuki instruction teaches children to play the instrument before introducing note-reading
4. Suzuki-trained students perform frequently and from memory with ease.

Limitation of Suzuki piano teaching

With these differences, we could figure out some of the limitations and disadvantages of using this method. Firstly, it is suitable for children only. For those who are teenager or adult, they cannot be beneficial from this teaching method. Secondly, this method is highly relied on the partnership between Suzuki teacher and parent. However, in our real life, most of the parent will have their own job to do, it may not be possible for them to take the piano lesson with their children every time. Therefore, it may not a really good alternative to solve this problem, especially in Hong Kong

5.1.2 Piano video game

On the present market, there are three famous piano video games that can help people to learn and practice piano skill. Following is the list of those piano video game:

1. Synthesia
2. Piano Dust Busters and Piano Maestro
Synthesia

Synthesia is a piano video game which does not required player to have any music theory knowledge (including note-reading skill). It supports Windows, Mac OSX, iOS and Android platform and allow user to connect it to the electronic piano with MIDI-port. The game style is similar to “DJMax” and “Guitar Hero”.

Fig 5.1.2a. Screen Capture from Synthesia

[Source: https://itunes.apple.com/hk/app/synthesia/id579518445?mt=8]

Fig 5.1.2a. shows as that the gameplay of Synthesia. It is designed to pack every music note as a falling block and the height of the block representing the beat of the note and the position of the black representing when and which key should the player press. Therefore, player will be able to play this game without reading notes from the music sheet.

Limitation of Synthesia

Synthesia is the most famous piano video game on the market. It has more than 500,000 downloads from Play Store on Android platform [4]. However, it has some limitations of the current version of Synthesia. Firstly, it only supports MIDI-port input or using computer keyboard to play the game, it means that it cannot be used when player is playing song with traditional piano. Secondly, it
cannot help to train player's note-reading skill. Player will not be able play piano without using this software. Thirdly, this system does not support interaction between player and player, which will make the game less interesting.

**Piano Dust Busters and Piano Maestro**

Piano Dust Busters and Piano Maestro are developed by Joytunes and support the iPad platform. They are both targeted on the child. For Piano Dust Busters, it is designed for those children who have no experiences in playing piano before. The gameplay style is similar to Synthesia, but less complicated. For Piano Maestro, it is basically put the while music sheet on it with some animation and cartoon background to make it as a game, using this approach the sight-reading skill can be trained when child playing this game. These two game is built with a “Music Sense Engine”, this technology allows the applications listen to what player is playing and identify the notes being played [5]. Therefore, no MIDI adapter and wire is needed to connect between piano and iPad.

**Limitation of Piano Dust Busters and Piano Maestro**

The first one is that Piano Dust Busters and Piano Maestro cannot support the song which is too complicated. According to my testing result, “Music Sense Engine” can only identify one note above middle C and one notes below middle C at the same time, which means that any song which required piano player to press three or more key at the same time, these two games cannot support it. And that is the reason of “Piano Maestro” only support the grade to grade 3, but not higher. As most of the song above grade 3 will require piano player to press more than two key at the same time.

| Piano Dust Busters | Piano Maestro |
The second limitation is that the user interface design is really attractive to the very young age of child as shown in Fig 5.1.2b. and 5.1.2c. but less attractive to teenagers and adult. It means that it will be not that effective to encourage teenagers or elder kids to practice their piano skill.

6. Proposed Design, Solution, System

To solve this problem, “Gamify piano project” is proposed. Although there are already some piano video game is announced on the market, because of the limitations of those games, they cannot really train or encourage player skill for a long time. This project is to proposed a piano game, which can encourage people to practice their piano skill and they can really their piano skill can be improved when they are playing this game. Furthermore, by peer 10 effects, we should establish an online platform to allow more interaction between player and player to influence them to practice more so as to enhance the motivation of piano practicing.
6.1 Gameplay Design

To make this game much more attractive and interesting, the gameplay had been redesigned after conducting a series of interview to teenagers.

Fig 6.1.a The basic user interface of gamify piano project (before redesign)

Fig 6.1.b The basic user interface of gamify piano project (after redesign)

Fig 6.1.b demonstrates the idea of the gameplay design. As you can see, the stave are kept in the game. The red stave on the right hand side are representing high pitch stave, and blue stave on the left hand side are representing lower pitch stave.

All the musical notes are redesigned as a color bar. The color bar will drop down from the top to bottom. Then, user has to press the right key on their keyboard, when the bar drops to the position on the stave which is
representing its pitch.

For example, the green bar in fig 6.1.c is a note representing A# key with octave 4. That means player have to press the A# key when the bar is at the A# position at the stave. To make the game easier to play, a reminder circle is provided. It can tell the player when to press the key by scaling down itself.

Besides, the horizontal position of the note is also representing the pitch of the note, player will know which key to press by identifying the note is on top of which key on the virtual keyboard displayed in the game.

Comparing to the previous design, the new designed gameplay, will allow those players who don’t have any music knowledge to play this game. Also, when the player plays this game more and more times, player should be able to have a basic idea on reading notes on the stave section.

Also, the new designed gameplay is more attractive and seems to be much more interesting to player, according to my researching result.

6.1.1 Game modes provided in this game

There are totally 3 game modes are provided in this game:

1. Auto play mode
2. Practice mode
3. Challenge mode

Auto play game mode

In this mode, the game will play all the note automatically, in order to shows what is the song sound like and give a demo to player how to play this game. Although player can always obtain full combo in this game mode, the record
will not be saved into database.

**Practice mode**

In this game mode, note (Color bar) will pause before they have been pressed correctly. Therefore, player will have enough time to find out which key to press so that they can learn or practice how to play this song.

**Challenge mode**

In this game mode, note (Color bar) will not pause. They will drop down at the speed given by the MusicXML file of that song. The note will disappear when player forget to press it at the right time. Also, combo will be reset, when player miss a note.

6.1.2 Online battle

Player will be able to join online battle to have a piano battle with other player. In online battle, the auto play mode will be disabled because it is meaningless for a computer to join online battle.

In online battle, the virtual keyboard will highlight the key in blue to show what are the opponent is pressing on his or her keyboard. Also, the music visualizer will show not only player progress in yellow color, but also show the opponent progress in blue color. Therefore, player will be able to know the opponent progress in real time. Also, opponent score will be displayed at the top left hand corner as well.
6.2 User Interface Design

In this game, there are several main screens for player to play with and use the functions provided in this game. The following are the screens list:

1. Starting screen
2. Main menu
3. Main gameplay
4. Chatting and friend system
5. Shop system

6.2.1 Starting Screen User Interface

This screen is for player to start the game and for the system to login to the server as well. And it clearly states the name of this game “Meteor”, which implies the music note falling like a “Meteor”.

Fig 6.2.1 Starting screen
6.2.2 Main Menu User Interface

Fig 6.2.2 Main Menu

Fig 6.2.2 shows the main menu for player to select song to play with. In this scene, there are two bars, the bottom bar is response to display all related information of the selected song. For example, the name of the song, the highest score, grade which the player got, the difficult level of the song will be displayed as well.

At the top bar, it includes two button, both of them is response to start a game. For the “SINGLE” button, that’s mean player will play the game by himself/herself. For the “Online” button, that’s mean player want to join an online battle. The system will try to match the player with another who wants to play the same song at the same time.

6.2.3 Gameplay User Interface

In the main game, there are mainly five components to construct the main gameplay screen. Following is the components list:

1. Scoreboard
2. Stave
3. Reminder
4. Virtual keyboard
5. Musical notes
6. Music visualizer and progress display

Stave

In the real music sheet, a stave must begin with a "clef", "time signature" and "key signature". However, in an interactive game, they are not a must anymore. As a matter of fact when a music sheet translates to an interactive game, player can know when to press the key by the speed of the musical notes coming out. As previously mentioned, an innovative sight-reading practicing gameplay mechanism is added into this game. This mechanism is that player has to press the right key on the piano, when the color bar drop down to the corresponding position on the stave that representing the pitch of that note.

Reminder

To make the game easier to play, a reminder will appear on the corresponding position on the stave, when the note has to be hit in a few seconds. At this point, the reminder is set to be a scaling circle, when the note has to be hit in a few seconds, the circle will appear, and scaling down. When the circle getting smaller, mean that player have to press the key.

Virtual Keyboard

This virtual keyboard is playing two roles in this game. Firstly, it is a helper for player to know which key to press, since the note will be aligned above the corresponding key. For example, if the note is a Middle C note, then the note will be placed above the Middle C key in the virtual keyboard. Moreover, the virtual keyboard also help player to know which key they are pressing on the keyboard. When the player pressed a key on keyboard, the corresponding key on virtual keyboard will turn to red to shows that it is pressed.
**Musical Note**

As previously mentioned, the musical note is redesigned as a color bar. At this point, there are four types of color bar in the system, which are blue, red, orange and green. Blue bar means that it is a note should be pressed by left hand. Red bar means it is a note should be pressed by right hand. Orange bar means it is a sharp note. Green bar means it is a flat note.

**Music visualizer and progress display**

![Image of music visualizer and progress display](image)

**Fig 6.2.3 Music visualizer and progress display**

As shown in Fig 6.2.3, there is a music visualizer in the background of the game. The music visualizer will give response when player press a key on their keyboard. Also, music visualizer will help to display the progress of the game play. At the beginning of the song, all the blocks which construct the music visualizer is black, then it will change to yellow based on the progress. Therefore, when the song is finish, the music visualizer will turn to yellow.
6.2.4 Chatting and friend system

As shown in Fig 6.2.4, player will be able to chat with friend who is online in real time. Player can type word and press the send button at the bottom right corner to send the message. In the left hand sides, player can browser his/her friend who is online through this browser.

6.2.5 Shop System

The number of “Meteors” (money), player have will be shown in the top bar. If player have enough “Meteors”, he/she can easily buy the item by pressing the button at the top right corner.
6.3 System

This part will include the major technical components of this system and the use case diagram to describe the functionality provided in this game.

6.3.1 Major technical components

This project will eventually include 6 major technical components. The following is the major technical components lists for this project.

1. Server sides
   A. Database
   B. Socket programming

2. Client sides
   A. MIDI Unified 5
   B. Music XML

Server sides

A server is a necessary component for this project, since this project is not only about making players play piano alone, but also encourage them to have more interaction between players and players. Therefore, server will play an important role on establishing connection between players and also storing some essential data of players so as to make the interaction become possible.

Server sides – Database

Server sides database will be responsible for storing basic player information, Music XML for all the available songs, the score of each play and the playing progress for each player. To cut the cost of developing this project, I will use MySQL database management system to implement this part.

Server sides – Socket Program

To make online piano battle become possible, socket program is a must. With socket program, server will be able to communication with client immediately. The following figure shows how the socket program works in this project.
Fig 6.3.1. Communication between client and server

Fig 6.3.1 shows the concept of how online piano battle will work. In Fig 6.3.1, the entire message between iPad and Server will be sent thought socket. And I will define a self-defined protocol later for Client sides and server sides to communicate with each other's. In this way, we can send a string instead of sending JSON to server so as to reduce the package size. Therefore, the communication performance will be better.

Client sides

Client sides are composed by two hardware, the first one is a iPad and second one is the digital piano. And they will connected by a midi to lightning connector. And it is responsible for dealing with the user input and interacts with the player. Also, to minimize the workload on the server, some of the data will be stored in client sides.

Client sides – MIDI Unified 5

MIDI Unified 5 is a third-party library for developer to connect to all the devices which support MIDI. In this project, MIDI Unified 5 will be used for connecting with the digital piano and retrieve what player is pressing on the digital piano.

Client sides – Music XML

MusicXML is the standard for sharing interactive sheet music. In this project, MusicXML file will play the music sheet role. It will be used to record the pitch, beats, the major/minor scale used in this song etc.. Every time, the game is about to start, the musicXML will be loaded to generate the game.
6.3.2 Use case diagram for gamify piano project

In this project, there are mainly 8 functions are provided to the player and 1 function is provided for the administrator to update the game. Following are the list of the function, this game will provided and the use case diagram for this project.

1. Player
   A. Create new account
   B. Add new friend
   C. Playing song (online battle)
   D. Playing song in practice mode
   E. Buy customize item
   F. Customize scoreboard
   G. Chat

2. Administrator
   A. Add new song

Fig 6.3.2 Use case diagram for gamify piano project
### Create new account

**Use case: Create new account**

<table>
<thead>
<tr>
<th>Actors</th>
<th>Player</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Brief description</strong></td>
<td>Creating a account for the player who are playing this game for the first time. Account information will be stored within the client sides, the system will automatically login, when player open this application later.</td>
</tr>
</tbody>
</table>
| **Preconditions** | Must be able to connect to server  
There is no account record found in the iPad |
| **Basic flow of the event** | The use case begin when the player open the application for the first time on the device.  
Game will ask player to enter his/her name.  
Player enters his/her name  
System display account create successfully message |
| **Alternative flow** | iPad is not able to connect to game server, game will display a error message to player. |

### Add new friend

**Use case: Add new friend**

<table>
<thead>
<tr>
<th>Actors</th>
<th>Player</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Brief description</strong></td>
<td>Player will be able to add new friend in the game, by inputting their player ID.</td>
</tr>
<tr>
<td><strong>Preconditions</strong></td>
<td>Must be able to connect to server</td>
</tr>
</tbody>
</table>
| **Basic flow of the event** | The use case begin when the player press the add friend button  
System will ask player to enter the friend ID, he/she want to invite.  
Player enters the player ID. |
| System display the corresponding player name and asking for confirmation  
| Player press confirm  
| System add him/her as the players friend  

**Alternative flow**

Enter wrong player ID, system will display an error message and loop back to step 2.  
Do not press “Confirm” in step 5, system will loop back to step 2.

---

**Play song**

**Use case: Play song**

<table>
<thead>
<tr>
<th>Actors</th>
<th>Player</th>
</tr>
</thead>
</table>

| Brief description | Player will be able to have an online battle with someone on the internet who wants to play the same song at the same moment. If there is no such a player, then player will play the song without an online battle. |

| Preconditions | NA |

| Basic flow of the event | The use case begin when the player press the play song button.  
| System will display a list of song that player can select one of them to play.  
| Player select one of the song  
| System automatically matches the player with other player who wants to play the song at that moment.  
| Start the gameplay.  
| System calculates the score he/she got in that game.  
| System shows the winner of the game  
|<<include use case: gain experiences and points>>> |

| Alternative flow | If the system cannot matches up the player for 20 seconds, the game will start and player will play it without a online battle |
If the client cannot connect to server, the game will start and player will play it without an online battle.

### Play song in practice mode

**Use case: Play song in practice mode**

<table>
<thead>
<tr>
<th>Actors</th>
<th>Player</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Brief description</strong></td>
<td>Playing song in practice mode, the musical notes will not move until the player press the right key on the piano.</td>
</tr>
<tr>
<td><strong>Preconditions</strong></td>
<td>NA</td>
</tr>
<tr>
<td><strong>Basic flow of the event</strong></td>
<td>The use case begin when the player press the play in practice mode button. System will display a list of song that player can select one of them to practice. Player select one of song Start the game in practice mode. System calculates the score he/she got in that game. &lt;&lt;include use case: gain experiences and points&gt;&gt;</td>
</tr>
<tr>
<td><strong>Alternative flow</strong></td>
<td>NA</td>
</tr>
</tbody>
</table>

### Buying customize background item

**Use case: Buying customize background item**

<table>
<thead>
<tr>
<th>Actors</th>
<th>Player</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Brief description</strong></td>
<td>Using the game points gained from the playing song, player will be able to buy customize background item to customize his/her background in the game</td>
</tr>
<tr>
<td><strong>Preconditions</strong></td>
<td>NA</td>
</tr>
</tbody>
</table>
Basic flow of the event

The use case begins when the player presses the shop button. System will display a list of customize background items for player to buy. Player selects one of the items to buy. System reduces the game point in the player account and adds the corresponding item in the player account.

Alternative flow

If the player's game point is not enough to buy the item, the system will show an error message and loop back to step 2.

Customize background

Use case: Customize background

<table>
<thead>
<tr>
<th>Actors</th>
<th>Player</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief description</td>
<td>Using their customize background item, player can change his/her background.</td>
</tr>
<tr>
<td>Preconditions</td>
<td>NA</td>
</tr>
</tbody>
</table>

Basic flow of the event

The use case begins when the player presses the shop button. System displays a list of customize background items which belong to the player and asks him/her to select anyone of them to use. Player selects one of the customize background items. System updates the properties of player's background.

Alternative flow

NA

Chat

Use case: Chat

<table>
<thead>
<tr>
<th>Actors</th>
<th>Player</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief description</td>
<td>Player will be able to chat to their friend, using this function.</td>
</tr>
<tr>
<td>Preconditions</td>
<td>Must be able to connect to the server</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td><strong>Basic flow of the event</strong></td>
<td>The use case will begin when player press the “chat” button. System shows up player friend list and waiting for player to pick anyone of them to chat. Player selects a friend to chat. System opens a channel for them to chat. Player enter a message and press “send” button System sends the message to the friend.</td>
</tr>
<tr>
<td><strong>Alternative flow</strong></td>
<td>if iPad is disconnected to the server, during chatting. System will shows up a error message to tell the player, the iPad is disconnect to the server.</td>
</tr>
</tbody>
</table>

**Gain experiences and game points**

<table>
<thead>
<tr>
<th>Use case: Gain experiences and game points</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors</strong></td>
<td>Player</td>
</tr>
<tr>
<td><strong>Brief description</strong></td>
<td>Player experiences and game points will be increased.</td>
</tr>
<tr>
<td><strong>Preconditions</strong></td>
<td>NA</td>
</tr>
<tr>
<td><strong>Basic flow of the event</strong></td>
<td>The use case begins when the game is finished. System will calculates the gained experiences and game points based on the game mode, song, and the score of that game. System will update the record to the local database. System will show the gained experiences and game points to the player.</td>
</tr>
<tr>
<td><strong>Alternative flow</strong></td>
<td>If the game can connect to the game server at step 3, the record will be synchronizing to the server database.</td>
</tr>
</tbody>
</table>
Add new song

<table>
<thead>
<tr>
<th>Use case: Add new song</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors</strong></td>
</tr>
<tr>
<td><strong>Brief description</strong></td>
</tr>
<tr>
<td><strong>Preconditions</strong></td>
</tr>
<tr>
<td><strong>Basic flow of the event</strong></td>
</tr>
<tr>
<td><strong>Alternative flow</strong></td>
</tr>
</tbody>
</table>

7. **Detailed Methodology and Implementation**

In the following sections, I will explain the mechanisms and logic behind my application in methodology part. And explaining the software architecture and programming approaches, I have used to implement those logic and mechanisms, in the implementation section.

7.1 **Server methodology**

In this system, server side program is mainly responsible for two responsibilities.

1. Data retrieval and management on server side database
2. Command forwarding between users
7.1.1 Data retrieval and management on server side database

Since the database is not stored on the local storage, server side, client side are using different programming language and framework as well. Therefore, there will not have a ready-made method for client to connect to the server side database directly. I have to develop a server side program to receive the command sent from client side, to execute the corresponding SQL query and send back the query result through an open-standard format (JSON - JavaScript Object Notation).

7.1.2 Command forwarding between users

To achieve the command forwarding, there are two important mechanisms have been designed.

1. Connection management
2. Battle pairing

Connection management
Unlike most of the data management application, server will not only send message to client after user making query request, but message could be sent anytime. For example, if user B want to send a message to user A, server
should be able to forward the message to user A initiatively, even user A have not made a request to get message.

Therefore, server side program should be able to store the connection after user have logged in, so that server side program can send message to any online client anytime.

Fig 7.1.2a Server side - connection management [see Appendix II.]

Fig 7.1.2a shows the concept of connection management. It is basically, pack the connection (socket connection) and its user id as an object, and store it into an array list so that server will be able to send message to any client anytime through the stored connection, when it has the client user ID.

Also, when a user has left the game, a disconnected signal must be sent to the server. Therefore, server can remove the invalid connection from the connection pool.

**Battle pairing**

As previously mentioned, this game allows player to have piano battle with other online players who want to play the same song at the same time. To make this function possible, we need mechanisms to do a player matching.
Fig 7.1.2b Battle pairing procedure [see Appendix III]

Fig 7.1.2b shows the procedure of battle pairing. Server will have a matrix to store all the room for battle pairing. For columns, it will represent the song ID, user want to play, for the rows, it will represent the rooms ID. Once a room has two players, then it will marked as unavailable and send the opponent user ID to each other. Therefore, they can forward command to each other. If the room has only one player, then the server will send a wait signal for user to wait for pairing.
7.2 Server Implementation

7.2.1 Database Design - Entity Relationship Diagram

Fig 7.2.1 Entity Relationship Diagram [see Appendix IV]

In order to avoid the chance of having data inconsistent, data redundancy, and storage waste problem, I designed this database in third normal form (3NF), after normalized the database, there are 6 entities are designed in this system, Following are the list:

1. Items
2. Users
3. ShopRecord
4. Friends
5. Scoreboard
6. Songs
Items

This entity is used to store the item that can be sold in the game store, user will be able to using “Meteors” (Currency in this game) to customize their gaming background. In this entity, there are 4 columns:

<table>
<thead>
<tr>
<th>items_id</th>
<th>The identify number of the items so as to make every record unique and identical so it will be a primary key of this entity.</th>
</tr>
</thead>
<tbody>
<tr>
<td>items_name</td>
<td>The name of item</td>
</tr>
<tr>
<td>items_score</td>
<td>It is stated the required number of “Meteors” to buy this item</td>
</tr>
<tr>
<td>items_file</td>
<td>The file path of this item. Client will load the resource (Sprite) based on this path.</td>
</tr>
</tbody>
</table>

Users

Users entity is stored every user record in the game, it will be used to check if the login is validate or not. And give a identity to every player.

<table>
<thead>
<tr>
<th>users_id</th>
<th>The identify number of users, it is also a primary key of this entity.</th>
</tr>
</thead>
<tbody>
<tr>
<td>users_name</td>
<td>The name of the user</td>
</tr>
<tr>
<td>users_password</td>
<td>The password of the user.</td>
</tr>
<tr>
<td>users_money</td>
<td>The remaining “Meteors” in his account</td>
</tr>
<tr>
<td>users_item_id</td>
<td>The background ID that user is using at this moment.</td>
</tr>
</tbody>
</table>

Shop Record

Shop Record is a weak entity, that composed by Users and Items. The usage of this table is to record the purchase record of different users.

| shopRecord_users_id | the users ID of the purchase record |
shopRecord_items_id | the ID of the item had been purchased

Friends

“Friends” is a weak entity, that composed by two user ID. The usage of this table is to keep the friendship record.

<table>
<thead>
<tr>
<th>user_id</th>
<th>the ID of the player</th>
</tr>
</thead>
<tbody>
<tr>
<td>friend_id</td>
<td>the ID of the friend</td>
</tr>
</tbody>
</table>

Scoreboard

“Scoreboard” is a weak entity, it is composed by the song ID, user ID, the score user obtained, and also the mode number. The usage of this table is to keep the score record.

<table>
<thead>
<tr>
<th>scoreboard_score</th>
<th>The score, player obtained in this play.</th>
</tr>
</thead>
<tbody>
<tr>
<td>scoreboard_songId</td>
<td>The song ID of this scoreboard record.</td>
</tr>
<tr>
<td>scoreboard_userId</td>
<td>The ID of the owner of this record.</td>
</tr>
<tr>
<td>scoreboard_mode</td>
<td>The mode number of this record. Mode number 0 means that it is played in challenge mode, Mode number 1 means that it is played in practice mode.</td>
</tr>
</tbody>
</table>

Songs

This entity is stored all the necessary information of available songs in this game. Therefore, if game administrator need to add a new song, then administrator has to update the database by adding a new song record to this entity.

| songs_id   | The identical number of the song record, which will make every song record become unique. It is also a primary key of this |

Final Year Project – Final Report
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>songs_name</td>
<td>The name of this song.</td>
</tr>
<tr>
<td>songs_level</td>
<td>The difficulty level of this song. 1 is easiest and 10 is hardest.</td>
</tr>
<tr>
<td>songs_file</td>
<td>The file path of this song. Client will load the resource (including the MusicXML and Song Photo) based on this path.</td>
</tr>
</tbody>
</table>

7.2.2 Class Diagram

Fig 7.2.2. Class diagram for server side [see Appendix V. or https://drive.google.com/file/d/0B8z8xfxPXBRnJVmxMN1pmU0zxZ0E/view?usp=sharing]

The server side program is coded in JavaScript using a Node.JS engine. Because of the fact that JavaScript is not a Object-Oriented language, the inheritance mechanism is missing. Therefore, these classes will only have composition, aggregation or association relationship between each other.

As previously mentioned, all the connection (socket connection) will be saved into an array list in this server program. And the array list is stored in “UsersSystem”, other classes will be able to retrieve the connection by calling the "getUserById" function.

For the battle pairing part, all the room matrix is stored in the “RoomsSystem”, and “RoomsSystem” class will assign a available room, when the “getRoomBySongId” function is called. After the available room object is retrieved, then server can call the “Pair” function provided by room class so as to complete the battle pairing function.
## Class Description

<table>
<thead>
<tr>
<th>Class name</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerGateway</td>
<td>It act like a gateway or hubs in this system. When the server received a command, it will call the corresponding class to handle it.</td>
</tr>
<tr>
<td>Friend System</td>
<td>This class is help to retrieve all the friend related information from the database. Also, it will response to check the online status of those friend.</td>
</tr>
<tr>
<td>RoomsSystem</td>
<td>This class is help to pair the player who want to have a online battle on the same song at the same time.</td>
</tr>
<tr>
<td>DirectCommunicationSystem</td>
<td>This class is responsible for forwarding the socket message to the specific player without any process.</td>
</tr>
<tr>
<td>ScoreboardSystem</td>
<td>This class is responsible for handling retrieve scoreboard request. Also, it will responsible for updating the score records.</td>
</tr>
<tr>
<td>UsersSystem</td>
<td>This class is responsible for managing the online user. Also, a login function is provided to handle player to login to the server.</td>
</tr>
<tr>
<td>ShopSystem</td>
<td>This class is responsible for retrieve the product list that player can buy in the game store. Also, it will responsible to handle the deal between game store and player.</td>
</tr>
<tr>
<td>Room</td>
<td>This class will responsible for pairing up player, when there are two players in this room, the status of this room will be changed to ready.</td>
</tr>
<tr>
<td>Users</td>
<td>This class will keep the player basic information and most importantly keeping his/her socket ID. Therefore, system can easily send a message to specific user any time.</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DatabaseManager</td>
<td>This class is to provide basic database operation function to others classes to use. For example, making query to database, or execute record update etc..</td>
</tr>
<tr>
<td>SQLDictionary</td>
<td>This class is stored all the SQL, other classes can easily use the SQL provided by this class. Using this method, can improve the reusability. And it will make SQL easier to manage in the future.</td>
</tr>
</tbody>
</table>

### 7.3 Client Methodology

There are 4 core mechanisms are designed in this game, in order to make this game run smoothly. Following is the list of those mechanisms:

1. Level generation based on Music XML
2. Note (Color bar) generation
3. Hit determination
4. Keyboard (Electronic piano) input system

#### 7.3.1 Level generation based on MusicXML

As previously mentioned, the game level is generated based on music sheet (in MusicXML format). Therefore, it is a must for this application to have mechanisms to read and convert the music sheet to something readable to the game.
Fig 7.3.1 MusicXML conversion [see Appendix VI]

Fig 7.3.1 shows the concept of this conversion. This conversion basically is to convert the music sheet as an array. The array will contain all the notes in the music sheet (including rest note, and normal note). The algorithm to assign note into array is:

1. Define how many array blocks would a specific music unit take. (In Fig 7.3.1 example, for every 16th note ($\frac{1}{16}$), it would take 1 array block. Therefore, for some music note which has double or even 4 times duration then 16th note ($\frac{1}{16}$), they would take 2 array blocks or 4 array blocks)

2. Assign note based on the array pointer, then update the array pointer based on the duration of the assigned note and the assumption on step 1.

Repeat step 2 until end of the song.

Using this algorithm, we can construct an array which contains all the notes in the music sheet. Also, the index of the array will implies the timing of the note should be play.

When we want to play the song, we only have to make a pointer to read the array from the beginning to the end. The speed of moving the pointer, is depends on how fast the song is. In other words, it should depend on the tempo of the song.
7.3.2 Note (Color bar) generation

Fig 7.3.2a. Concept of the first approach to generate note

Fig 7.3.2a shows the simplest approach to implement this function. The red box is the camera, which is what the player’s sight view. And the blue bar, above the camera, is representing the note to be generated at the beginning of the game. All the notes will drop down at the same speed, so that the note with higher Y coordinate will appear later, so it will work like a music box. However, using this approach will cause “Lag” and consume too much memory which mobile device may not have.

Fig 7.3.2b Concept of second approach to generate note
To solve this problem, I update the program to generate only the notes in the next measure as shown in Fig 7.3.2b, instead of generating all the notes in the song at the beginning.

With the second approach, the usage of memory are reduced in a huge extend, but it leads another problem. Because of the fact that instantiate a Game Object needs lots of computing power. It will cause “Lag” as well.

Object pooler approach

To solve the lagging problem, I create a pool which is instantiated a specific amount of inactive note (Color bar) inside the pool at the beginning of the game. When the system needs to generate the note in next measure, system can request note from the pool so that system will not need to instantiate note (Color bar) at run time. After the note is pressed or missed, the note will back to object pooler again, so that it can be reused.
7.3.3 Hit Determination

In this game, when player press a key on their keyboard to hit the note, system will determine whether it is a “Perfect hit”, “Great hit”, “Normal hit” or “Miss”.

![Collision detector on stave](image1.png)

Fig 7.3.3a. Collision detector on stave

![Hit determination section on each note](image2.png)

Fig 7.3.3b. Hit determination section on each note

Fig 7.3.3a shows that the collision detector is attached on the stave, when a note is over it, a collision will occur. When a user pressed key on their keyboard, system will check if there is any collision occurs in the game. If there is, it will determine if the hit is a “Perfect hit”, “Great hit”, “Normal hit” or “Miss”, depends on which section is the collision occur, detailed shown in Fig 7.3.3b.

7.3.4 Keyboard Input

Although I have used MIDI Unified 5 library to help me handle the input part, some process is required to make it fit my game perfectly.

After obtaining the MIDI input signal from MIDI Unified 5, it will update the state on the abstract keyboard in the game. The abstract keyboard is a model class, which is absolutely invisible to player. However, it will be used to record which key is pressed right now, and which key is not.

Therefore, other classes can easily know what the current state of the keyboard is so as to helping the program to give response and make determination.
7.4 Client Implementation

7.4.1 Summary of software design

![Layer of classes in this game](image)

Fig 7.4.1 Layer of classes in this game

There are totally 58 classes in this client-side application. All those classes can be categorized into three layers of classes.

The bottom layer is helper classes. They are response to do calculation and storing global constant to make other classes can easily access them. To ensure it can be access by other classes easily, all the method will be set to be a static method, or the class will adopt the singleton design pattern.

The classes in second layer are some logic or event handling classes. These classes are response to generate game object, storing game state or handle user input. These classes will directly call the help classes function to delegate the calculation task or get global constant.

The classes in the top layer are controller classes. They are responsible for controlling the game object on the screen. Moreover, when there are some collision occurs, they will delegate the task to the event handling classes to handle it.
7.4.2 Class diagram

Fig 7.4.2 Client side class diagram [Appendix VII or https://drive.google.com/file/d/0B8z8xfxPXBNjJmtMb3IWXJQT28/view?usp=sharing]

To make it easier to read, the classes are highlighted in different color by layer.

- For the helper classes’ layer, they will be highlighted in gray.
- For the logic / event handler layer, they will be highlighted in blue, purple, white or orange.
- For the controller layer, these classes will be highlighted in red.
- For auxiliary view of existing classes, they will be highlighted in green, these auxiliary view can make the class diagram have a higher readability.
- For the classes in orange, they are the classes contain thread programming.

**Class description**

<table>
<thead>
<tr>
<th>Class name</th>
<th>Class responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationWindows</td>
<td>It is responsible to get the current width and height of the gaming device. Also, 4 functions, to help others class to get the coordinate of the bottom, top, left and right of the screen.</td>
</tr>
<tr>
<td>PositionCalculator</td>
<td>It is responsible to calculate all the position of each game object used in the game.</td>
</tr>
<tr>
<td>AnimationCalculator</td>
<td>It is responsible to calculate anything related to animation that will be used in the game.</td>
</tr>
<tr>
<td>TempoCalculator</td>
<td>It is responsible for calculate speed based on the tempo given by the musicXML file.</td>
</tr>
<tr>
<td>SystemDictionary</td>
<td>It store all the self defined ID, that will be useful in the system</td>
</tr>
<tr>
<td>GameUIDictionary</td>
<td>It store all the constants related to UI.</td>
</tr>
<tr>
<td>MusicDictionary</td>
<td>It store all the music related constants, and provided some basic convert function to convert string to those self defined music constants.</td>
</tr>
<tr>
<td>SystemSettings</td>
<td>It is responsible to store some global variable that can be changed in the future (for example, fps(Frames per seconds) of this game)</td>
</tr>
<tr>
<td>MapSystem</td>
<td>It is responsible for generating note, stave, reminder and detectors in the game.</td>
</tr>
<tr>
<td>KeyboardSystem</td>
<td>It is response to handle the keyboard input,</td>
</tr>
</tbody>
</table>
and manage the AbstractKeyboard class.

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComboSystem</td>
<td>It is response to shows and update the combo number.</td>
</tr>
<tr>
<td>ScoreSystem</td>
<td>It is response to show and update the score on the top left corner.</td>
</tr>
<tr>
<td>EventSystem</td>
<td>It is response to handle different event in the system, and act like a switch to assign task to the corresponding event handler.</td>
</tr>
<tr>
<td>OnReminderEventHandler</td>
<td>It is response to handle that when the system found that the note is close enough that the system should generate the reminder to remind player to press the key, then this event handler, should be able to call the map system to generate the reminder.</td>
</tr>
<tr>
<td>OnMeasureEventHandler</td>
<td>It is response to handle that when one measure is finish, this event handler should call the map system to generate next measure.</td>
</tr>
<tr>
<td>OnBackgroundPlayEventHandler</td>
<td>It is response to output the corresponding sound, just before the player should press the key.</td>
</tr>
<tr>
<td>OnCompleteListener</td>
<td>It is response to handle what should the system do, when a note is finish.</td>
</tr>
<tr>
<td>OnNormalEventHandler</td>
<td>It is response to handle the normal hint event. It will call combo system and score system to update their combo and score.</td>
</tr>
<tr>
<td>OnGreatEventHandler</td>
<td>It is response to handle the great hint event. It will call combo system and score system to update their combo and score.</td>
</tr>
<tr>
<td>OnPerfectEventHandler</td>
<td>It is response to handle the perfect hint event. It will call combo system and score system to update their combo and score.</td>
</tr>
<tr>
<td>Event Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OnMissEventHandler</td>
<td>It is response to call combo system to reset, if there is a miss occur.</td>
</tr>
<tr>
<td>MusicNote</td>
<td>It records all the necessary attributes of a music note in the game.</td>
</tr>
<tr>
<td>SystemNote</td>
<td>There are two types of system notes: one is a “measure” note, the other one is a “ending” note, they are telling the application when is the end of a measure, and when the song is end.</td>
</tr>
<tr>
<td>NoteInterface</td>
<td>A interface for both music note and system note.</td>
</tr>
<tr>
<td>Poolable</td>
<td>Poolable interface is designed for those game object that have to use the object pooler to implemented.</td>
</tr>
<tr>
<td>Colliable</td>
<td>A interface for a game object that will be attached the collision listener to implemented.</td>
</tr>
<tr>
<td>AbstractKeyboard</td>
<td>This class is manage 88 AbstractKey instance to simulate a keyboard in the system.</td>
</tr>
<tr>
<td>AbstractKey</td>
<td>This class is mainly keep the state of the key so that the system can tell if the key is pressed or not.</td>
</tr>
<tr>
<td>MusicXMLDocumentHeader</td>
<td>This class is stored all the beat attributes of the selected song.</td>
</tr>
<tr>
<td>MusicXMLDocumentBody</td>
<td>This class is response to read the musicXML file and convert it into a list of music note and store them.</td>
</tr>
<tr>
<td>ReminderController</td>
<td>It is response to control the reminder to</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LengthBarController</td>
<td>It is response to control the color bar height. When the note is pressed, the color bar height should be scale down.</td>
</tr>
<tr>
<td>NoteController</td>
<td>It is response to control the note (color bar) to drop or pause, when different event happen.</td>
</tr>
<tr>
<td>CollisionListener</td>
<td>It is a class that will be response to call its parent in Unity with a ID, when a collision occur. Therefore, different Collidable object can reuse this collision listener, to implement the collision detection function.</td>
</tr>
<tr>
<td>KeyController</td>
<td>It is response to control the key color, when user press the key on the physical keyboard, then the key controller will change the corresponding key color.</td>
</tr>
<tr>
<td>StartSystem</td>
<td>It is response to control the starting screen and also helping user to login to the system</td>
</tr>
<tr>
<td>MenuSystem</td>
<td>It is response to control the menu, so as to user will able to select the song, they want to play in either “Auto”/ &quot;Challenge”/ &quot;Practice” mode.</td>
</tr>
<tr>
<td>ChattingSystem</td>
<td>It is response to send message and display the chatting history.</td>
</tr>
<tr>
<td>FriendBrowser</td>
<td>It is response to shows all the online friend that player can chat with.</td>
</tr>
<tr>
<td>ResultSystem</td>
<td>It is response to display the result page and also send the gaming result to server to update the record.</td>
</tr>
<tr>
<td>FriendController</td>
<td>It is response to control the friend name label inside the friend browser.</td>
</tr>
</tbody>
</table>
Container

It is a controller class to control selectable item, for example, the selling item in game shop, and the song in menu.

OnClickListener

This is a class to listen to the click event, so every object which is clickable can use this script.

CommunicationSystem

It is response to send and receive message from server in a unique method. Please see section x.x in detail.

7.4.3 Unique programming approach – Event System

Fig 7.4.3 Class diagram – Event System [Appendix VIII]

Fig 7.4.3 shows that there is a Collision Handler Interface, and all the event handler will implement this interface. This interface has two methods. The first one is a accept method which is used to determine whether they are responsible for the event or not. The other method is a handle method, which will be called, when the event handler is responsible for handling that event.

In the Event System class, it will have a list of each event handler. When a event happen, the event system will search for the responsible event handler, by calling the accept method, then the handle method of the responsible event handler will be called. The following code, shows the exact implementation.

```java
//eventHandlers are the list that contains every event handler in the system

public void handleCollisionEvent(NoteInterface target, int eventID){
```
CollisionHandlerInterface handler = null;
for (int i = 0; i < eventHandlers.Count; i++) {
    handler = (CollisionHandlerInterface)eventHandlers[i];
    if (handler.accept(eventID)) {
        break;
    }
}
handler.handle(target);

7.4.3 Event System implementation

**Advantage of using this approach**

With this approach, this system will have a higher extendibility and flexibility. When I need to add a new event in this system, I can easily create a new event handler by implementing the “CollisionHandler” Interface. And add the new event handler to the list in the Event System class. After that, the new event is already added into the system.

This programming approach observed the Open-Closed Principle, since there is not require any medication when I need to add a new event into the system.

7.4.4 Unique programming approach – Communication System

**Problem encountered:**

Communication system is the only class response to communicate with server directly in this application. Other classes have to delegate their communication request to this class very often. However, after communication system received the response from server, it will not have a mechanism to return and do not know where to return the result. Therefore, I have designed a unique programming approach based on a well-known design pattern - Delegation design pattern.

**Delegation design pattern:**
Fig 7.4.4a Delegation design pattern [Appendix IX.]

Fig 7.4.4a shows the design of the communication system. “CommunicationSystem” class is playing as a delegator. And “OnCommunicationSystem” will be a delegatee interface. Other classes like “MenuSystem”, “ShopSystem”, “ProgressSystem”, “ScoreSystem”, “StartSystem” etc… as a concrete delegatee role in the delegation design pattern.

With the use of delegation design pattern, there will be a method to return the execution result from delegator to delegatee. However, when communication system received a response from server, it will not know where to return the result. Therefore, I designed another mechanism to solve this problem.

To solve this problem, communication system will send each request with a task ID, and save the delegatee reference with the assigned task ID together into an array. When server is sending back the response, it will send back the task ID as well. Therefore, communication system will be able to have the task ID to search from the array to find out the delegatee. With the delegation
design pattern, the server response will be able to return to the class which makes the request.

8. User Experience Evaluation

According to “User Experience Evaluation Methods in Academic and Industrial Contexts”, there are totally 5 methods to evaluate user experience [6].

1. Lab Studies
2. Field Studies
3. Surveys
4. Expert Evaluations
5. Mixed Methods

I have chosen surveys method to evaluate user experiences, because of its efficiency and applicability.

8.1 Surveys

Survey is mainly used on game play design evaluation in this project. Interviews were conducted two times at different stage in this project. The first interview was conducted at the very beginning of the project, in order to know what people think about the draft game play design so as to adjust the development direction of this project. The second interview was conducted after two versions of game play design is completed so as to know what people will concern about and which rendering style, would be most acceptable to player.

The reason of conducting traditional face to face interviews instead of conducting an online questionnaires survey is that interviewee is hard to understand the game play design by looking at a screen capture or reading the game play description.

Therefore, I decided to conduct face to face interviews, so as to explain each of the game play design to them.
First interview

*Interview background information:*

Date: October 2015  
Number of interviewees: 4  
Male/ Female Ratio: 1/1

As previously mentioned, the purpose of conducting first interview is to know what people think about the draft gameplay design so as to adjust the development direction of this project. Therefore, I have designed 3 gameplays in different style for interviewees to choose, as shown in Fig 8.1a.

**Result:**

<table>
<thead>
<tr>
<th>Number of gameplay design</th>
<th>Number of people chosen this gameplay design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gameplay 1</td>
<td>4</td>
</tr>
<tr>
<td>Gameplay 2</td>
<td>0</td>
</tr>
<tr>
<td>Gameplay 3</td>
<td>0</td>
</tr>
</tbody>
</table>
Most of the interviewees make a negative response about the rendering style. They think that it is not attractive enough for them to play this game. And the main reason of why they choose the Gameplay 1 is that they can easily understand how to play the game.
For Gameplay 2, interviewees think it is just a music sheet in other rendering style.
For Gameplay 3, 3 of the interviewees said the same comment as Gameplay 2, and there is an interviewee said that it is little bit confusing to render it in 3D style.

Second Interview

*Background information*
Date: December 2015
Number of interviewees: 7
Male/ Female Ratio: 4 / 3

![Fig 8.1b. Gameplay design 4](image-url)
According to the result of first interview, the gameplay has been redesigned based on the Gameplay 1 and improving the 2D rendering style. As a result, there are two versions of gameplay design has been completed, as shown in Fig 8.1b and Fig 8.1c.

**Result:**

<table>
<thead>
<tr>
<th>Number of gameplay design</th>
<th>Number of people chosen this gameplay design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gameplay 4</td>
<td>2</td>
</tr>
<tr>
<td>Gameplay 5</td>
<td>5</td>
</tr>
</tbody>
</table>

**Male prefer gameplay design**

Fig 8.1d Interview result analyses
In the second interview, most of the interviewees make a positive response about the rendering style for both of the gameplay design. According to Fig 8.1d and Fig 8.1e, I can know that male would prefer gameplay design 5 more, while female would prefer gameplay design 4 more. Although this game is designed for both male and female, male player is generally more than female player in 2016, according to “Distribution of computer and video gamers in the United States from 2006 to 2016, by gender” [7]. Therefore, I decided to use gameplay design 5 in this project.

9. **Summary of achievements**

Practicing piano is a boring activity to most of the people, since it will take a ton of time to do boring exercises to improve their sight-reading skill and keeping even beats. Although there is already some piano practicing or learning game on the market, none of them successfully provide a fun way for practicing their sight-reading skill. There are either training their sight-reading skill, but less funny (e.g. Piano Meaestro), or interesting but cannot train player sight-reading skill (e.g. Synthesia). This project provided an innovative approach for player to have an enjoyable gameplay while the sight-reading skill of the player can be trained at the same time.

To make sure the game is attractive to player, a series of surveys were conducted and analysis the result to improve the gameplay design. Besides, different programming approach was evaluated (in methodology section) to choose the most suitable approach to be adopted in this project.
10. Limitations

The biggest limitations of this game, is that it cannot help player to practice the volume control and pedal control.

Since the idea of this game is want to make the piano practicing become interesting, at the same time, it should be able to help player to practice the core techniques which are sight-reading skill and keeping even beat. However, I noticed that the gameplay will be too hard for most of the player, if the player is required to do all the thing (sight-reading, keeping beat, volume control and pedal control) at the same time.

Therefore, I tried to simplify the gameplay by taking out the volume controlling and pedal controlling part.

As a result, this game will be able to help player to train their sight-reading skill and the sense of rhythm, but it could not help player to practice their volume control and pedal control skill.

11. Critical review

During the development of this project, I realized that as a good developer, I should not only think in engineering aspect, but it is really important to think as a user and designer.

User will never care about how difficult or how complex the system is, they will only care about if it is easy to use and the user interface is attractive or not.

In order to make the application become attractive to user, I learnt a lot of designing skill and drawing skill in this project. (For example, how to use Photoshop to create some gaming element and how to use After Effect to create some video clips)

Moreover, I have a chance to apply different programming skill in this project. For example, bit mask, delegation design pattern, normalized database etc…
so as to improve the extendibility and efficiency of my game. Also, when I encounter an efficiency problem, I have successfully analysis the time complexity of the problem so as to solve that problem.

12. Future Improvement

At this point, this project can only gamify piano, but not including other instruments, for example, guitar, drum, etc…

However, this game play design approach and the MIDI connection provide a potential to support different instrument. Therefore, once the all the classes related to piano can be redesigned into a abstract instrument interface, all the other student learning other instruments should be able to learn or practice their skill through this game.

Moreover, when the game supports more and more different instruments, the game should have a possibility to support a band to practice their skill online. In this way, band will be more practicing chance by playing this game. To make the band can practice through this game, a room size should be updated, and the computing power of the server should be upgraded as a matter of fact that, it will be a heavy computing task, if there is a ton of message forwarding have to be done at a short period of time.

13. References


14. Appendix

Appendix I:
Appendix III:

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Song ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room Number</td>
<td></td>
</tr>
<tr>
<td>Song A</td>
<td>Song B</td>
</tr>
<tr>
<td>Send User: User ID 1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 2</th>
<th>Song ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room Number</td>
<td></td>
</tr>
<tr>
<td>Song A</td>
<td>Song B</td>
</tr>
<tr>
<td>Send User: User ID 1</td>
<td></td>
</tr>
</tbody>
</table>

User ID 5

User ID 7

User ID 6
Appendix IV:
Appendix VI

Poco moto

\begin{music}
\begin{musicexample}[b]{2.5cm}
\myScore
\myStaffType{bass}
\myStaffType{treble}
\myGrandStaffType{true}
\myScoreTitle{Appendix VI}
\myStaffName{1}{Poco moto}
\myStaffName{2}{Poco moto}
\myMusicInput{myexample}
\end{musicexample}
\end{music}

The duration for every $\frac{1}{2}$ equals to 1 block

\begin{music}
\begin{musicexample}[b]{2.5cm}
\myScore
\myStaffType{bass}
\myStaffType{treble}
\myGrandStaffType{true}
\myScoreTitle{Appendix VI}
\myStaffName{1}{Poco moto}
\myStaffName{2}{Poco moto}
\myMusicInput{myexample}
\end{musicexample}
\end{music}
Appendix X

Monthly Logs

July 2016 Monthly Logs
In this month, I am mainly focusing on the debugging part and implementing the shop system which will allow player to use their game points to buy some item to customize their gaming environment in the game.

June 2016 Monthly Logs
In this month, I am working on the online battle part. To implement online battle function, a server side program is a must. I decided to implement the server side program using NodeJS and web socket. Also, chatting system is implemented in this month.

May 2016 Monthly Logs

2/5/2016 Progress Log - Updating the material designed for the gameplay.
Using Photoshop, to update some of the materials that will be used in the main gameplay. And also, design some more material for lighting effect in the main gameplay. And export them as png image so that it will be easily to use in the future. Also, a updated preview gameplay image has been completed as well.

2/5/2016 to 5/5/2016 Progress Log - Reading the MusicXML file in Unity
Continue to coding a MusicXML reader in C# language, to read a MusicXML file data into and array, in a self defined format. Therefore, I can easily generate a map in the music game in the future.

Using my MusicXML reader, to generate music notes in the game at the right time. Also, having some basic control on those generated music note so that they will be able to move.

12/5/2016 to 15/5/2016 Progress Log - Redesign and rebuild the software architecture
Redesign the whole software architecture, using class diagram. Because of the fact that finding out that the software architecture is not well designed enough for the continue development. After redesign and rebuild the whole system, the system have a higher extensibility and flexibility.
16/5/2016 to 25/5/2016 Progress Log - Implementing the mechanism behind the gameplay.
Implementing different event that will happen in the main gameplay. For example a event when user missing to hit the note at the right time, or a event when user hitting the music note at the perfect timing. And also, implementing a much comprehensive generating map system to make the music note not only can move, but also stop and they will have different length base on the duration of that music note. (For example, a 16th note will have a shorter length than a quarter note.)

25/5/2016 to 31/5/2016 Progress Log - Establishing the connection between the system and piano.
Using MIDI, I will be able to use the keyboard input and output. So I have to implement a system with plugin "Unified MIDI 5" and physical MIDI adapter "iRigi MIDI 2" to obtain MIDI signal in the way I designed.

Jan 2015 Monthly Logs

In this month, the piano game's gameplay has been updated to be much more attractive and interesting. Also, the artwork has been well-designed and saved as png file, for each of the components of the main gameplay. To ensure, the updated gameplay will be interesting and attractive to player, I had conducted 10 interviews by asking interviewees that if they are interested in playing this game after having a look on my pre-view gameplay image. Almost all of the feedback are positive.

For the programming part, I am trying on change the game engine to develop this project as well. As a matter of fact that, the "SceneKit" game engine, provided by Apple, is designed for physical simulation game. It will be good for develop something like 3D version Mario game or war game, but not a good idea for develop a game that totally irrelevant to physics.

Therefore, I am trying to learn and implement this project using another game engine named "Unity", in order to make the game performs better.

Dec 2015 Monthly Logs
In this month, I am working on redesign the game play to make the game more interesting and attractive to the player. Hence, some user interface changes is made. Also, I am trying to working on the MusicXML part, to make iPad can actually read MusicXML file and be able to get some data from it. Therefore, what I am actually doing for this part, is building interface for reading MusicXML file so that I will be able to easily develop the game easily in the coming months.

In the coming month, I will try to finish the interface implementation for reading MusicXML. Also, I will refine the software structure to make it easily to extend and maintenance in the future development.

Oct 2015 Monthly Logs
1/10/2015 Progress Log - Start studying different piano game
Start to study different kind of music game such as "Taiko no Tatsujin", "Jubeat", "Groove Coaster". The studying including how they design their user interface and the way to play the game.

6/10/2015 Progress Log - Designing Piano game (Musical Planet version)
After studying those music games, I got a brief idea on how to design the user interface of this project. The basic idea is that creating a planet at the middle of the game as a scoreboard. And there will be ten rings at the outside of the planet. And five of the rings will at the bottom, and five of them will at the top. Therefore, something like a stave will be constructed. And the musical notes(something like a ball) will came out from a specificity point on those rings. And those musical notes will orbit the scoreboard one round. And user are required to hint the right piano key, when the musical note is arrived to the end of the ring.

15/10/2015 Progress Log - Designing Piano game (Roller Coaster version)
Found out another way to design the game, which is inspired by a video named "Zürich Chamber Orchestra | Roller Coaster" (URL: https://www.youtube.com/watch?v=InYoStzmGo). And I am wondering which approach will be most suitable for designing my piano game (Theme: musical planet /musical roller coaster)

17/10/2015 Progress Log - Decided to try to implement prototype for both theme of piano game.
I decided to implement a prototype for both of the theme to see which one is better. And to if it is possible or not to implement.
19/10/2015 Progress Log - Studying the concrete way to implement this game. Start to learn how to implements 3D games using Swift. Found out that there are two 3D game engines, that "Apple" provided for developer to develop 3D game. The first one is called "SceneKit", the second one is called "Metal". SceneKit is a high level 3D game engine, which means that it is easier to use, but not as flexible as using "Metal". And I decided to have a try on using "SceneKit", to develop the user interface first. Since if I can use "SceneKit" to implement this game. It must be much more simpler than using "Metal" to write this game.

22/10/2015 Progress Log - Implemented static user interface for musical planet version of piano game. Implemented a static user interface for the piano game with musical planet theme.

24/10/2015 Progress Log - Apply for the extension of FYP submission
Because of the full-time job, I noticed that it is not easy to handle the Final Year Project, therefore, I applied for the extension of FYP submission.

25/10/2015 Progress Log - Found out technical issues
Found out that it is not that easy to make the musical notes to orbit the musical planet, since the standard API of "SceneKit", doesn't provided a flexible rotate function. There is some handicap on using the standard rotate function to make the musical notes to orbit the musical planet.

27/10/2015 Progress Log - Technical issues solved
Using the "moveTo()" function with render loop provided by "SceneKit", I have successfully implement the make the musical notes orbit the planet(Scoreboard).