

**COEN 6312-** Model Driven Software Engineering

**(Fall 2016)**

**Project Name: Learn the Nodes**

**Project Team: Music Geeks**

**Deliverable No. 2**

**Submitted by:             Submitted to:**

Team: Music GeeksAbdelwahab Hamou Lhadj

Submission Date: Oct 12, 2016

Table of Contents

[1.Textual description of Domain](#_30j0zll)

[1.1 Domain Model:](#_1fob9te)

[1.2 Domain model description:](#_3znysh7)

[2. Use case analysis.](#_2et92p0)

[2.1 Use case model.](#_tyjcwt)

[2.2 List of use cases:](#_3dy6vkm)

[2.3 Use Case Description](#_1t3h5sf)

[3.Requirements](#_2s8eyo1)

[3.1 Functional Requirements](#_3rdcrjn)

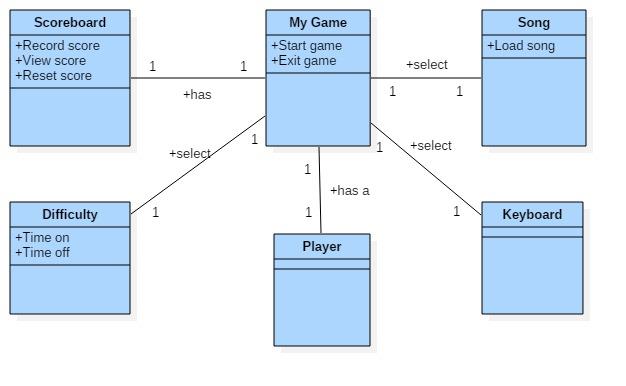
[3.2 Non-functional requirements](#_26in1rg)

[3.3 Software Requirements-](#_35nkun2)

[3.4 Hardware requirements](#_1ksv4uv)

**1.Textual description of Domain**

**1.1 Domain Model:** Domain model is used to describe the problem domain space. It is a way to describe, model and to find the relationships between the real world entities. It is conceptual model of the domain which includes both data and behavior. It is visual presentation of separation of domain into individual conceptual classes and objects.



**1.2 Domain model description:**

**Player:** The one who is playing the game. Player has the option of learn the nodes from where he can choose either start the game if he wants to play it or exit the game if he wants to close the game at any point of time.

**Difficulty:** It is the level preferred by the player to continue his/her game. It has two attributes.

Time on: It will keep the record of time taken by the user to play the selected song using interface.

Time off: No time constraint on the user while playing the game.

**Scoreboard:** will track scores achieved by the player while playing the game. It has following three attributes.

Record score: It will keep the record of player’s scores updating while he/she is playing the game.

View score: It will display the scores to the user scored by him.

Reset Score: It will allow the user to reset his/her score according to preference.

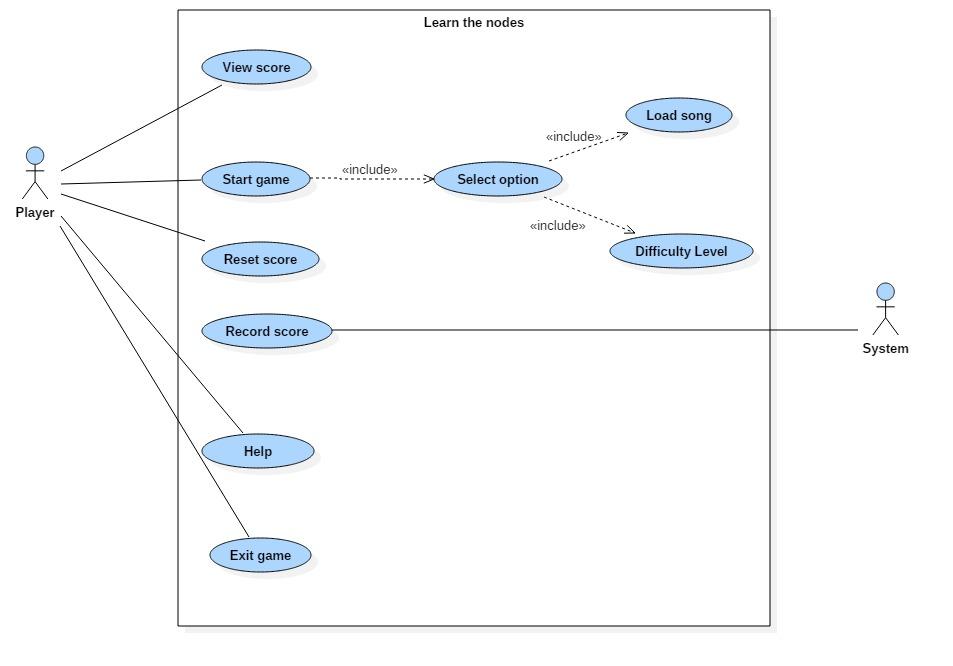
**Song:** It contains the songs already present in the database. It has further one attribute

Load: It will allow the user to load the song of his/her preference.

**Keyboard:** It will provide an option of playing using one hand or use both the hands to the user who is playing the game.

**2. Use case analysis.**

**2.1 Use case model.** Use case model for learn the nodes is inspired by the interaction of player with interface and is presented below



**2.2 List of use cases:**

|  |  |
| --- | --- |
| *UC-#* | *UC Name* |
| *UC-1* | *Record Score* |
| *UC-2* | *View Score* |
| *UC-3* | *Start game* |
| *UC-4* | *Select option* |
| *UC-5* | *Load song* |
| *UC-6* | *Time on/off* |
| *UC-7* | *Help* |
| *UC-8* | *Reset score* |
| *UC-9* | *Exit Game* |

Table 1 - List of Use Cases

**2.3 Use Case Description**

#### *Use Case #2: View Score*

|  |  |  |
| --- | --- | --- |
| **Use Case UC-2** | View Score | |
| **Related Requirements:** | RQ-3 | |
| **Priority** | High | |
| **Pre Conditions** | User already playing the game. | |
| **Post Conditions** | The current score of user is displayed. | |
| **Initiating Actors** | User | |
| **Actor’s Goal** | To see the current score | |
| **Trigger** | User clicks on ‘View score’ button | |
| **Main Scenario** | **Step** | **Action** |
| 1. | User clicks on the view score button. |
| 2. | Application displays a current score. |

#### *Use Case #3: Start Game*

|  |  |  |
| --- | --- | --- |
| **Use Case UC-3** | Start Game | |
| **Related Requirements:** | RQ-1 | |
| **Priority** | High | |
| **Pre Conditions** | Application should be opened. | |
| **Post Conditions** | New window open to allow user to select further option. | |
| **Initiating Actors** | User | |
| **Actor’s Goal** | To start playing game | |
| **Trigger** | User clicks on Start button | |
| **Main Scenario** | **Step** | **Action** |
| 1. | User opens ‘learn the nodes’ application |
| 2. | Application displays homepage. |
| 3. | User clicks on ‘Start Game’ button. |

#### *Use Case #4: Select options*

|  |  |  |
| --- | --- | --- |
| **Use Case UC-4** | Select options | |
| **Related Requirements:** | RQ-4 | |
| **Priority** | High | |
| **Pre Conditions** | User already clicks on start button. | |
| **Post Conditions** | 1. Song is loaded. 2. Timing is on or off according to user’s preference. | |
| **Initiating Actors** | User | |
| **Actor’s Goal** | To select the option. | |
| **Trigger** | User clicks on ‘select option’ button | |
| **Main Scenario** | **Step** | **Action** |
| 1. | User clicks on the select option button. |
| 2. | Application displays further options. |
|  | 3. | Required option is selected. |

#### *Use Case #5: Load song*

|  |  |  |
| --- | --- | --- |
| **Use Case UC-5** | Load song | |
| **Related Requirements:** | RQ-4 | |
| **Priority** | High | |
| **Pre Conditions** | User should click on select option button. | |
| **Post conditions** | Selected song is loaded. | |
| **Initiating Actors** | User | |
| **Actor’s Goal** | To load the desired song. | |
| **Trigger** | User clicks on ‘Load song’ button. | |
| **Main Scenario** | **Step** | **Action** |
| 1. | User clicks ‘load song’ button. |
| 2. | Application displays a window showing list of songs. |
|  | 3. | User selects particular song. |
|  | 4. | Selected song is uploaded. |

#### *Use Case #6: Time on/off*

|  |  |  |
| --- | --- | --- |
| **Use Case UC-6** | Time on/off | |
| **Related Requirements:** | RQ-4, RQ-5 | |
| **Priority** | High | |
| **Pre Conditions** | User should click on select option. | |
| **Post conditions** | Time displayed according to user’s preference otherwise not. | |
| **Initiating Actors** | User | |
| **Actor’s Goal** | To include timing option or not . | |
| **Trigger** | User clicks on ‘time on/off’ button. | |
| **Main Scenario** | **Step** | **Action** |
| 1. | User clicks ‘time on/off’ button. |
| 2. | Time is displayed if user clicks on ‘time on’ button |
|  | 3. | If user selects ‘time off’ game continues without score. |

#### *Use Case #7: Help*

|  |  |  |
| --- | --- | --- |
| **Use Case UC-8** | Help | |
| **Related Requirements:** | RQ-7 | |
| **Priority** | Low | |
| **Pre Conditions** | Application should be opened. | |
| **Post conditions** | New window open to provide help to the player. | |
| **Initiating Actors** | User | |
| **Actor’s Goal** | To get help regarding doubts. | |
| **Trigger** | User clicks on ‘Help’ button. | |
| **Main Scenario** | **Step** | **Action** |
| 1. | User clicks ‘Help’ button. |
| 2. | Application displays a window showing instructions of game. |

#### *Use Case #8: Reset Score*

|  |  |  |
| --- | --- | --- |
| **Use Case UC-8** | Reset Score | |
| **Related Requirements:** | RQ-7 | |
| **Priority** | Medium | |
| **Pre Conditions** | User already playing the game. | |
| **Post Conditions** | The screen is cleared of any symbols. | |
| **Initiating Actors** | User | |
| **Actor’s Goal** | To reset the current score | |
| **Trigger** | User clicks on ‘Reset’ button | |
| **Main Scenario** | **Step** | **Action** |
| 1. | User clicks on the reset button. |
| 2. | Application displays a confirmation message. |
|  | 3. | Score resets to zero. |

*Use Case #9: Exit Game*

|  |  |  |
| --- | --- | --- |
| **Use Case UC-9** | Exit Game | |
| **Related Requirements:** | RQ-8 | |
| **Priority** | Medium | |
| **Pre Conditions** | Application is running. | |
| **Post Conditions** | Application is terminated. | |
| **Initiating Actors** | User | |
| **Actor’s Goal** | To quit the application. | |
| **Trigger** | User clicks on ‘Exit’ button | |
| **Main Scenario** | **Step** | **Action** |
| 1. | User clicks on the ‘Exit’ button. |
| 2. | Application displays a confirmation message. |
| 3. | 3.1 Exit application if user clicks on ‘Yes’ option. |
|  | **3.2** Resume application if user clicks on ‘No’ option. |

**3.Requirements**

A requirement is a document containing physical as well as functional needs that a system must perform. These represent elements and functions that are necessary for the particular project.

We normally classified requirements as functional and nonfunctional requirements.

* Functional requirements specify the needs of the customers i.e. how the users want their system should be.
* Non-functional requirements, which defines the standards that can be used to conclude the quality of the system.

**3.1 Functional Requirements**

Functional requirements are the desired functionality that the user wants in their system. It specifies the operations and activities that a system must be able to perform. Requirements i.e. functions that our game should support are as follow

* User can start a new game in the beginning.
* User can record his/her score.
* User can view his/her score either he is playing the game right now or not.
* User should be able to have an access to select option in order to load a song of his/her choice.
* An option of timing should be optional for the user i.e. either it has to be on or off, depends upon user’s requirements.
* User should be able to access the help option in case of any doubt.
* User can also reset his/her score to the zero if required.
* User should be able to exit the game whenever desired.

**3.2 Non-functional requirements**

Non-functional requirements place constraints on how the system will provide solution to the problem. Non-functional requirements for our system are as follow

* **Modifiability:** The game should be effectively modifiable i.e it should be easy to add new components later and to modify the existing components.
* **Operability:** The game would be easy to utilize i.e it should be well-formed which makes the game easy to learn and operate.
* **Responsiveness:** The game should be able to respond quickly to the player. (Responsive comes under Performance Requirements)
* **Robustness:** The game should be able to cope with errors made by the players while playing.
* **Usability:** The game should be easy to use so that even the new player can adjust easily. Moreover, the game should provide a “help” option for the user to better understand the game and play accordingly.
* **Interface:** The interface of the game developed is easily understandable by any user.
* **Performance Efficiency:** In order to keep the efficiency high, the game should be able to perform the functions under limited memory and graphic options.

**3.3 Software Requirements-**

* Eclipse 4.5.2
* Papyrus
* JDK 8
* Star Uml
* Microsoft Visio
* Microsoft Office
* Microsoft windows 7 or higher

**3.4 Hardware requirements**

* RAM-512 mb or higher
* Android Device
* Hdd space- Minimum 2gb free
* Cpu-2.0 ghz or higher