****

**SOEN 6011**

**Software Engineering Processes**

**Summer 2016**

**Assignment #2**

**Requirements Document and Use Case Analysis**

**By:**

**Group-8 (LIONS)**

**Team Members**

|  |  |  |
| --- | --- | --- |
| Mona | Movahedi | 26979327 |
| Supreetha | Mudarajappa | 27682417 |
| Heya | Na | 27208243 |
| Amandeep | Narula | 27359950 |
| Dhruv | Ohri | 27149220 |
| Parth | Patel | 40014231 |
| Prashanth | Ramesh | 27382006 |
| Sukhjit | Singh | 40009670 |

**Table of Content**

[1. Background information: 3](#_Toc450924342)

[2. Problem: 3](#_Toc450924343)

[3. Functional Requirements 3](#_Toc450924344)

[**3.1 Project deliverable-1** 4](#_Toc450924345)

[**3.2 Project deliverable-2** 4](#_Toc450924346)

[**3.3 Project deliverable-3** 5](#_Toc450924347)

[4. Non-functional requirements 5](#_Toc450924348)

[**4.1 Project deliverable-1** 5](#_Toc450924349)

[**4.1.1 Performance Requirement** 5](#_Toc450924350)

[**4.1.2 Software Quality Attributes** 5](#_Toc450924351)

[**4.2 Project deliverable-2** 5](#_Toc450924352)

[**4.2.1 Performance Requirement** 5](#_Toc450924353)

[**4.2.2 Software Quality Attributes** 6](#_Toc450924354)

[4.3 Project deliverable-3 6](#_Toc450924355)

[**4.3.1 Performance Requirement** 6](#_Toc450924356)

[**4.3.2 Software Quality Attributes** 6](#_Toc450924357)

[5. Use Case 6](#_Toc450924358)

[5.1 Use Case Diagram for Project deliverable-1 7](#_Toc450924359)

[5.2 Use Case Scenarios for deliverable 1 7](#_Toc450924360)

[5.3 Use Case Diagram for Project deliverable-2 10](#_Toc450924361)

[5.4 Use Case Scenarios for deliverable 2: 11](#_Toc450924362)

[5.5 Use Case Diagram for Project deliverable-3: 14](#_Toc450924363)

[5.6 Use Case Scenarios for deliverable 3: 14](#_Toc450924364)

# Background information:

The tic-tac game is played on a 3\*3 grid that consists 9 cells. The game is played by two players. One player can also be a computer. All the cells in the game will initially be loaded empty. One player will choose X and the other player will choose O. whatever symbol user initially selects, he can draw only that symbol on an empty cell. Both the users can draw until

1. The board is full and the game is resulted in a draw
2. One of these two players had drawn three X’s or O’s in a row, either horizontally, vertically or diagonally.

The application is designed in such a way that only two players can play the game at a particular instance.

# Problem:

To develop an application to play a 3\*3 Grid Tic-tac-toe game. The game developed will work as desktop application also as well as mobile (android) application. The game will give an option for human player to choose either ‘X’ or ‘O’. The application will be played through a suitable interface, to play against a human player.

In the first deliverable a graphical user interface (GUI) will be designed, which displays the 3\*3 board and player can enter either ‘X’ or ‘O’.

In the second deliverable, the gameplay of Tic-tac-toe is implemented in the following way. Two players can mark X’s or O’s in the 3\*3 board. If the player who succeeds in getting three X;s or O’s consecutively across horizontally, vertically or diagonally is declared as the winner of the game. If none of them gets three consecutive X’s or O’s the game is declared as Draw. Later option will give a choice for players to restart the game.

In third deliverable, the game is designed for human player to play against the computer. The gameplay should be implemented in such a way that the different difficulty levels or heuristics levels where the computer will always aim to build the human player by implementing advanced algorithm for different difficulty levels. If none of them gets three consecutive X’s or O’s the game is declared as Draw. Later option will give a choice for players to restart the game.

We will develop the game by considering each functional and non-functional requirements for each of the deliverable as setting our goal.

# Functional Requirements

|  |  |
| --- | --- |
| Deliverables | Requirement |
| Deliverable 1 | |
| 1.1 | 3 X 3 Board displayed on the screen |
| 1.2 | Exit the game |
| 1.3 | Choose X or O for display on the gameboard |
| 1.4 | Draw "X" and "O" |
| 1.5 | Reset the board |
| 1.6 | A help button for Guidelines |
| Deliverable 2 | |
| 2.1 | Start a new game |
| 2.2 | How many games a player have won |
| 2.3 | Display best of 3 or best of 5 game series |
| 2.4 | Display status of whose turn it is |
|  |  |
| Deliverable 3 | |
| 3.1 | Different levels of game play |
| 3.2 | Background music that is different when playing vs the computer |
| 3.3 | Able to start as an X or as an O |
| 3.4 | Reward or a gift for the winner |
| 3.5 | Save game |

Table 1 Functional Requirements

## **3.1 Project deliverable-1**

**3.1.1. Displaying a 3x3 grid:**

This grid is required to play the Tic-Tac-Toe game. It is on this grid that the two players take turns marking the spaces as X or O.

**3.1.2. Exit the game:**

The grid should have an option to exit the game whenever a player wants.

**3.1.3. Alternating between X and O Players:**

This game requires two players, one playing as X and one playing as O (with X usually starting).To ensure that a player doesn’t play twice, switching between them is required.

**3.1.4. Reset the board:**

The reset button will let user to rest the board at any point of the game.

**3.1.5. Help button:**

The button will show guidelines to the users of how to play the game.

## **3.2 Project deliverable-2**

**3.2.1. Start a new game:**

This option will let user to start a new game at any point of time.

**3.2.2 Display of number of moves user has won:**

The system must displays that after how many moves the player can win

**3.2.3 Display best game in last 3 games:**

The system must displayed the best game in last 3 games.

**3.2.4 Display whose turn it is:**

The system must display which player’s turn it is during the game.

## **3.3 Project deliverable-3**

**3.3.1 Different levels of game:**

The game will have three levels as easy, medium and hard. A user can choose what level he/she wants to choose to play with the heuristic.

**3.3.2 Background music for the game:**

When the user plays the game against the heuristic there will a background music playing.

**3.3.3 Able to start as an X or an O:**

A player can choose whether he/she will be able to choose X or O to make the moves on the board.

**3.3.4 Reward or a gift for the winner:**

When the player wins the game he/she will get a virtual gift from the game center.

**3.3.5 Save game:**

When a player wants to save the current game in the middle, he/she can save it and close the game. When the user reopens the game it will continue from the saved position.

# Non-functional requirements

## **4.1 Project deliverable-1**

### **4.1.1 Performance Requirement**

1) **Responsiveness:** The game application must respond quickly to players and changing environment.

### **4.1.2 Software Quality Attributes**

1) **Usability:** The game application must be easy to use so that even the new players can adapt easily.

2) **Reliability:** Ability to retrieve the scores even after the game application crashes.

3) **Portability:** The application must be portable so that it can run on any platform.

4) **Robustness:** The application must be robust to allow the scope of making mistakes by the players.

## **4.2 Project deliverable-2**

### **4.2.1 Performance Requirement**

**1) Responsiveness:** The game application must respond quickly to players and changing environment.

### **4.2.2 Software Quality Attributes**

1) **Usability:** The game application must be easy to use so that even the new players can adapt easily.

2) **Reliability:** Ability to retrieve the scores even after the game application crashes.

3) **Portability:** The application must be portable so that it can run on any platform.

4) **Robustness:** The application must be robust to allow the scope of making mistakes by the players.

## 4.3 Project deliverable-3

### **4.3.1 Performance Requirement**

1) **Responsiveness:** The game application must respond quickly to players and changing environment.

### **4.3.2 Software Quality Attributes**

1) **Usability:** The game application must be easy to use so that even the new players can adapt easily.

2) **Reliability:** Ability to retrieve the scores even after the game application crashes.

3) **Portability:** The application must be portable so that it can run on any platform.

4) **Robustness:** The application must be robust to allow the scope of making mistakes by the players.

# Use Case

## 5.1 Use Case Diagram for Project deliverable-1

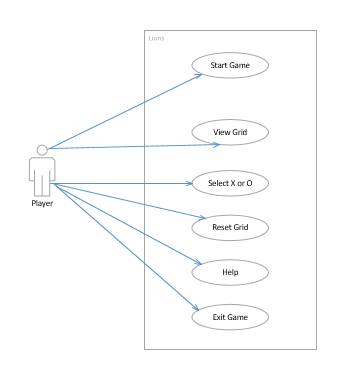


Figure1 Use Case Diagram for deliverable-1

## 5.2 Use Case Scenarios for deliverable 1

|  |  |
| --- | --- |
| **Use Case ID:** | UC1.1 |
| **Use Case Name:** | Play Game |
| **Priority:** | High |
| **Primary Actors:** | Player |
| **Description:** | This use case describe how the Player draws either X or O |
| **Trigger:** | Player clicks the cell where he wants to draw |
| **Pre-condition:** | Gameboard should be displayed |
| **Post-conditions:** | The X or O is drawn on the board based on Player choice |
| **Normal Flow:** | 1. Player starts the game by launching the jar file 2. The system displays A 3x3 board with 9 cells on the screen 3. User clicks a cell . 4. The system displays X in selected cell. 5. User clicks again in the same cell or any other cell to draw O 6. System displays in selected Cell. |
| **Alternate Flow:** | None |
| **Exceptional Flow:** | None |
| **Assumptions:** | * System is java enabled * Jar file is not corrupt * Images used to draw X and O are not corrupt. |

|  |  |
| --- | --- |
| **Use Case ID:** | UC1.2 |
| **Use Case Name:** | Reset game |
| **Priority:** | N/A |
| **Primary Actors:** | Player |
| **Description:** | User is able to reset the game board |
| **Trigger:** | Player clicks the reset button |
| **Pre-condition:** | Game board should have some cells filled with either X or O |
| **Post-conditions:** | All the cells on game board are empty |
| **Normal Flow:** | 1. Player chooses the option to reset the board by clicking the reset button under game menu. 2. System makes All the cells are empty 3. System displays A status saying “board is reset”. |
| **Alternate Flow:** | 1. Player chooses the option to reset the board by clicking the reset button under game menu. 2. All the cells are empty 3. System Displays A status saying “board is reset”. |
| **Exceptional Flow:** | None |
| **Assumptions:** | * System is Java enabled * Jar file is not corrupt * Reset button is enabled |

|  |  |
| --- | --- |
| **Use Case ID:** | UC1.3 |
| **Use Case Name:** | Exit game |
| **Priority:** | N/A |
| **Primary Actors:** | Player |
| **Description:** | This use case explain how User is able to exit the game at any point |
| **Trigger:** | User clicks exit game button |
| **Pre-condition:** | Game should be running |
| **Post-conditions:** | The application should close successfully |
| **Normal Flow:** | 1. Player chooses exit option by clicking exit button under the game menu 2. System Displays a message informing that “Are you sure to close the game” 3. Player Chooses yes button. 4. System closed the game. |
| **Alternate Flow:** | 1. Player Choose no button. 2. System close the message boz and displays the board again. |
| **Exceptional Flow:** | 1. Exit button is disabled or not working 2. User tries to force quit the game by clicking the cancel button. |
| **Assumptions:** | * System is Java enabled * Jar file is not corrupt * Exit button is enabled and functional. |

## 5.3 Use Case Diagram for Project deliverable-2

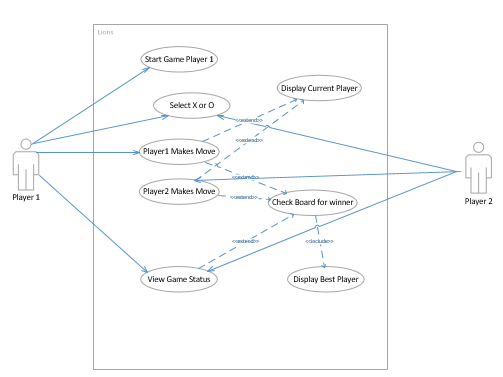


Figure2 Use Case Diagram for deliverable-2

## 5.4 Use Case Scenarios for deliverable 2:

|  |  |
| --- | --- |
| **Use Case ID:** | UC2.2 |
| **Use Case Name:** | Display moves to win |
| **Priority:** | high |
| **Primary Actors:** | Player |
| **Description:** | This use case displays How many moves a player have to won |
| **Trigger:** | The player press the moves number button |
| **Pre-condition:** | The game is started |
| **Post-conditions:** | The number of remaining moves will be displayed |
| **Normal Flow:** | 1. The player press the moves number button 2. The system calculates the remaining moves to player has won. 3. The System displays the remaining moves. |
| **Alternate Flow:** | None |
| **Exceptional Flow:** | None |
| **Assumptions:** | * System is java enabled * Jar file is not corrupt * Images used to draw X and O are not corrupt. |

|  |  |
| --- | --- |
| **Use Case ID:** | UC 2.3 |
| **Use Case Name:** | Display Best Player |
| **Priority:** | high |
| **Primary Actors:** | Player |
| **Description:** | This use case displays who is the best player in last 3 games. |
| **Trigger:** | The player press the best player button |
| **Pre-condition:** | The game is started |
| **Post-conditions:** | The best player in last 3 games will be displayed |
| **Normal Flow:** | 1. The player press the best player button  2. The system checks the history of the game.  3. The System finds the best player in last 3 games depend on their scores  4. The system displays the best player in last 3 games with his score. |
| **Alternate Flow:** | None |
| **Exceptional Flow:** | None |
| **Assumptions:** | * System is java enabled * Jar file is not corrupt * Images used to draw X and O are not corrupt. |

|  |  |
| --- | --- |
| **Use Case ID:** | UC 2.4 |
| **Use Case Name:** | Display Current Player |
| **Priority:** | high |
| **Primary Actors:** | System |
| **Description:** | This use case displays who is the current player’s name |
| **Trigger:** | The player makes a move. |
| **Pre-condition:** | The game is started |
| **Post-conditions:** | The name of player is displayed |
| **Normal Flow:** | 1. The player makes a move.  2. The system Find the name of current player  3. The System displays the name of current player. |
| **Alternate Flow:** | None |
| **Exceptional Flow:** | None |
| **Assumptions:** | * System is java enabled * Jar file is not corrupt * Images used to draw X and O are not corrupt. |

## 5.5 Use Case Diagram for Project deliverable-3:

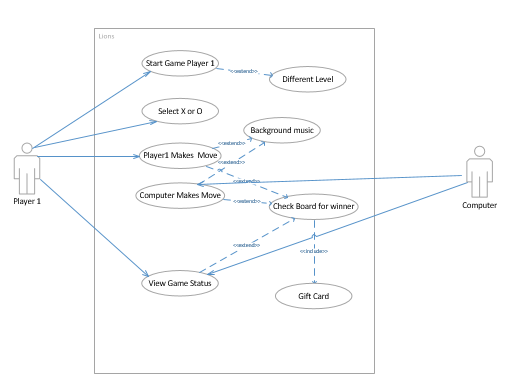


Figure3 Use Case Diagram for deliverable-3

## 5.6 Use Case Scenarios for deliverable 3:

|  |  |
| --- | --- |
| **Use Case ID:** | UC 3.1 |
| **Use Case Name:** | Computer Makes Move |
| **Priority:** | High |
| **Primary Actors:** | Computer |
| **Description:** | If player 1 chooses to draw X then Computer is able to draw O or vice versa. |
| **Trigger:** | Computer clicks any of the empty cells. |
| **Pre-condition:** | Grid is displayed with at least one empty cell where the Computer can make move. |
| **Post-conditions:** | If Computer is playing with X, X is drawn on the cell. |
| **Normal Flow:** | 1. Computer clicks one of the empty cell in the grid. 2. Computer draws symbol on the empty cell that is clicked. |
| **Alternate Flow:** | None |
| **Exceptional Flow:** | None. |
| **Assumptions:** | * System is Java enabled * Jar file is not corrupt * Grid is displayed with at least one empty cell.. |

|  |  |
| --- | --- |
| **Use Case ID:** | UC 3.2 |
| **Use Case Name:** | Background Music |
| **Priority:** | Low |
| **Primary Actors:** | Speaker |
| **Description:** | Background music is played while a player 1 is playing with computer. |
| **Trigger:** | Player 1 or computer makes move. |
| **Pre-condition:** | Grid is displayed and cells are clickable.  Speakers are turned on and function properly. |
| **Post-conditions:** | Speakers play the background music as soon as user or computer makes a move. |
| **Normal Flow:** | 1. Player 1 or computer clicks any of the cells. 2. The background music is played. |
| **Alternate Flow:** | None |
| **Exceptional Flow:** | None. |
| **Assumptions:** | * Grid is displayed and cells are clickable.. * Speakers are turned on and functional. |

|  |  |
| --- | --- |
| **Use Case ID:** | UC 3.3 |
| **Use Case Name:** | Different Level |
| **Priority:** | N/A |
| **Primary Actors:** | Player 1 |
| **Description:** | After player 1 starts the game, he should be able to select the difficulty level i.e. Hard, Easy etc. |
| **Trigger:** | User clicks the start game button  User selects the difficulty level from the drop down menu and click ok. |
| **Pre-condition:** | Start button is enabled and functional.  The drop down menu is selectable and Ok button is enabled and functional.. |
| **Post-conditions:** | The difficulty level is applied. |
| **Normal Flow:** | 1. User clicks on the start button. 2. User selects the difficulty level from the drop down menu and click ok button. 3. The difficulty level is applied and also displayed on the screen. |
| **Alternate Flow:** | None |
| **Exceptional Flow:** | None. |
| **Assumptions:** | * Start button is enabled and drop down menu is selectable. |