|  |
| --- |
|  |
| User requirement specifications |
|  |
| MDW – Ludo game |

**Fontys university of applied sciences**

May 11, 2016

Authored by: Monica Stoica , Dimitar Vikentiev, Rosen Danev

Table of Contents

[Introduction 2](#_Toc450769338)

[About the game 3](#_Toc450769339)

[Functional requirements 4](#_Toc450769340)

[Moscow 4](#_Toc450769341)

[Use Cases 5](#_Toc450769342)

[Use case diagram 10](#_Toc450769343)

[User Interface 11](#_Toc450769344)

[Non-functional requirements 12](#_Toc450769345)

# Introduction

Our group consists of three members: Rosen Danev, Monica Stoica and Dimitar Vikentiev, students of class EI8S2.

The following document describes the implementation of an object-oriented software product using UML techniques.

The goal of this software system is to allow users to play the Ludo game. In addition, the User Requirements Specification (URS) will be described such as functional and non-functional requirements and user interface.

The functional requirements are represented by use cases and MoScoW. We have chosen the most suitable use cases so that the most functionality of our system will be covered. In this way, we were able to determine the most appropriate user-friendly interface.

# About the game

Ludo, is a board game for two to four players in which players race their four pawns from start to finish according to dice rolls.

Before the beginning of the game, each player’s tokens are out of play staged in one of the large corner areas of the board (called the player’s yard).

When able to, the players will enter their tokens one per time on their respective starting squares, and proceed to race them anticlockwise around the board along the game track (the path of squares not part of any player's home column). When reaching the square below his home column, a player continues by racing tokens up the column to the finishing square. The rolls of a cube die control the swiftness of the tokens, and entry to the finishing square requires a precise roll from the player. The first to bring all their tokens to the finish wins the game. The others often continue play to determine second-, third-, and fourth-place finishers.

To enter a token into play from its staging area to its starting square, a player must roll a 6. If the player has no tokens yet in play and does not roll a 6, the turn passes to the next player. Once a player has one or more tokens in play, he selects a token and moves it forward along the track the number of squares indicated by the die roll. Players must always move a token according to the die value rolled, and if no move is possible, pass their turn to the next player.

When a player rolls a 6 he may choose to advance a token already in play, or alternatively, he may enter another staged token to its starting square. The rolling of a 6 earns the player an additional ("bonus") roll in that turn. If the additional roll results in a 6 again, the player earns an additional bonus roll. If the third roll is also a 6, the player may not move a token and the turn immediately passes to the next player.

A player may not end his move on a square he already occupies. If the advance of a token ends on a square occupied by an opponent's token, the opponent token is returned to its owner's yard. The returned token may only be re-entered into play when the owner again rolls a 6. There are no "safe" squares on the game track which protect a player's tokens from being returned. A player's home column squares are always safe, however, since no opponent may enter them.

# Functional requirements

## Moscow

|  |  |  |
| --- | --- | --- |
| Nb. | Requirement | MoSCoW |
|  | The player can throw the dice | M |
|  | The game will start automatically after all 4 players have joined. If there are less than 4 players but more than 2, then any of them can start the game. | M |
|  | A player can choose one of the four colours to represent him/her | M |
|  | The players can exchange messages | S |
|  | A player can log in with a username and password | S |
|  | A player can log out | S |
|  | A player can quit the game at any time without affecting the game | M |
|  | Only one player per square | M |
|  | Ranking | S |
|  | A player can sign in | S |
|  | Other players will be informed when a player choses to exit the game or is kicked out of the game | M |
|  | Players can invite other players to join | C |
|  | A player can pause and resume the game | S |
|  | At the end of the game, an overview is provided | S |
|  | Allow other users to watch the game | C |
|  | Replay of the game | C |
|  | The player has to move the pawn according to the dice | M |
|  | If rolled a 6, the player has the opportunity to start with a new pawn and the user can roll the dice again | M |
|  | When a player wins, the game will end for him/her. However, he will have the opportunity to watch the other players compete for the second and third place | M |
|  | Ranking of the players | S |
|  | Visitors can watch the game | C |

## Use Cases

*The following use-cases represent the functional requirements that Ludo game will be providing.*

***Goal:*** Throw the dice

***Pre-condition:*** The game has been started

***Actor:*** Player

***MSS:***

1. The actor presses on the ‘Roll the dice’ button
2. The system generates a random number between 1 and 6 and shows it on the screen.
3. The system allows the actor to move the pawn according to the number rolled before.

***Ext***: 1.a. It is not the actor’s turn. The system displays a message informing the actor that he has to wait for his turn to come.

***Goal:*** Start the game

***Pre-Condition:*** At least 2 players have joined the game

***Actor:*** Player

***MSS:***

1. The actor presses the ‘Start game’ button.
2. The system checks if there are at least two players and starts the game by enabling one of the players to throw the dice

***Ext:*** 1.a There are exactly four players. The system will automatically start the game and will allow one of the players to throw the dice.

1.b .There are already 4 players playing the game. The actor will join the game as a visitor.

***Goal:*** Quit the game

***Pre-condition:*** The game is running

***Actor:*** Player

***MSS:***

1. The actor presses the ‘Quit game’ button
2. The system displays a dialog box asking the actor to confirm his/her choice.
3. The actor presses the ‘Quit’ button
4. The system kicks out the actor from the game and informs the other players that the actor has quitted the game.

***Ext:*** 4a. There are only two players in the game. The system will end the current game and both players will be kicked out.

***Goal:*** Move the pawn

***Pre-condition:*** The dice has been rolled

***Actor:*** Player

***MSS:***

1. The actor clicks on the pawn he/she wants to move.
2. The system calculates and moves the pawn on the proper square and disables the current actor and enables the next actor to roll the dice.

***Ext:*** 1.a. The actor has no pawns out yet. The system checks if the thrown number is a 6. If yes, the actor clicks on one of the paws. The system draws the pawn on the screen. In both cases, the system disables the current actor and enables the next actor to roll the dice.

3.c There is a pawn on the selected square. The already existent pawn will be put back on the starting spot and the new pawn will take its place.

***Goal:*** Players exchange messages

***Pre-Condition:*** The game has been started

***Actor:*** Player

***MSS:***

1. The actor types a message in the message text box.

2. The actor presses the ‘Send’ button.

3. The system updates the chat window with the new message for all users of the system.

***Goal:*** Log in

***Pre-Condition: -***

***Actor:*** Player

***MSS:***

1. The actor presses the log in button
2. The system displays two textboxes for username and password respectively.
3. The actor enters the username and password.
4. The actor presses the ‘Login’ button.
5. The system checks if the account exists and if the username and the password match.
6. The system displays the main menu of the game.

***Ext:***  3a. The username and password are not in the database. The system notifies the actor and the actor is sent to step 3.

3.b. The username and password combination does not match. The system notifies the actor and the actor is sent to step 3.

***Goal:*** Log out

***Pre-Condition:*** The game has been started and the player has already logged in.

***Actor:*** Player

***MSS:***

1. The actor presses the ‘Log out’ button.

2. The system logs out the actor and displays the start screen of the game.

***Goal:*** Pause game

***Pre-Condition:*** The game is already running in a game phase.

***Actor:*** Player

***MSS:***

1. The actor presses the ‘Pause’ button.

2. The system will pause the game for all the players.

***Goal:*** Overview

***Pre-Condition:*** The game has ended.

***Actor:*** Player

***MSS:***

1.The system displays overview of the game.

***Goal:***  Ranking

***Pre-Condition:*** The player has an account

***Actor:*** Player

***MSS:***

1. The player presses the ‘See Ranking’ button.

2. The system displays player’s ranking.

***Goal:*** Sign in

***Pre-Condition:*** The game has been started

***Actor:*** Player

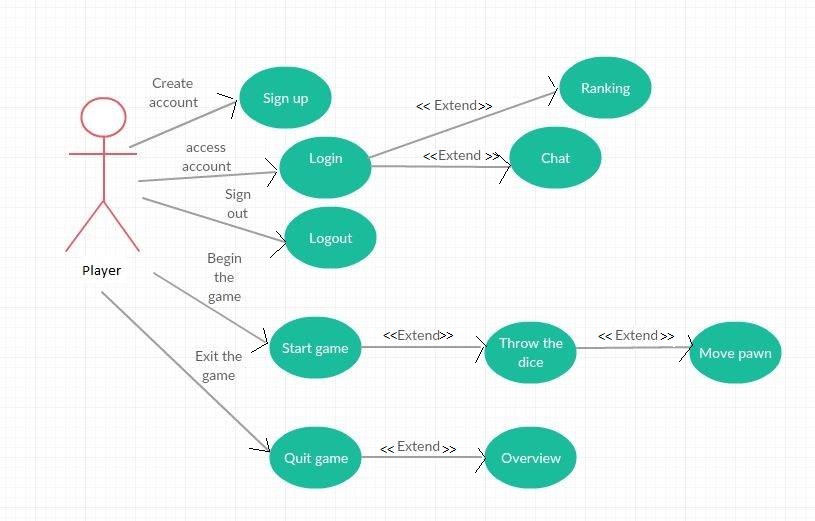
***MSS:***

1. The actor presses the ‘Sign in’ button.

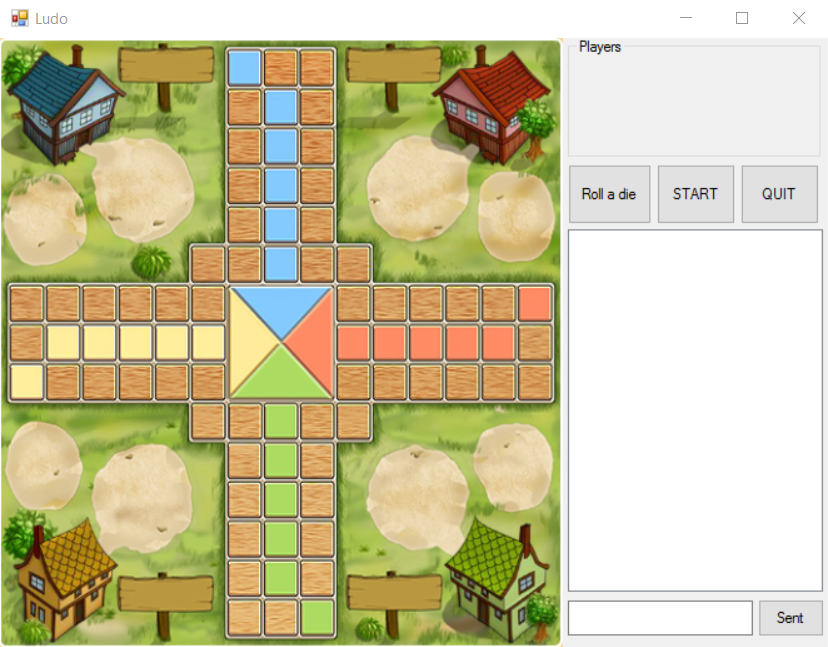
2. The system displays the Sign in form.

1. The actor enters the needed information and presses ‘Submit’ button.
2. The system creates account.
3. The system closes the form and opens the main menu.

## Use case diagram



# User Interface



The picture above represents the main screen of the game. However, it represents only the ‘Must’ requirements, meaning that the interface will suffer some updates. At the beginning, when the user starts the game, he/she will need to choose a colour to represent him/her - by clicking on designated area.

In the upper right corner, the user will see all players who have joined the game. Slightly down there are 3 buttons (roll a die, start, quit). Button roll a die will be available for click only if it’s the turn of the user. When the user click on that button will be shown what die received.

As is mentioned in the description of the game at least 2 players (users) must have entered the game to be launched by clicking on the start button.   
By clicking the quit button, the user will be able to leave the game.

The players can communicate with each other on the chat displayed on the right of the screen. In order to send a message, a user has to type some text in the indicated text box and press the send button.

# Non-functional requirements

Due to its target audience, the game is aiming for a user friendly interface. Therefore, it does not matter how experienced a user is with the game. The only aspect that the user has to be concerned with is represented by the rules of the game. We will deliver test plans in order to achieve reliability for preventing unexpected error which might lead to unwished crashes. The final product will be an installation file.

1. The game will be written in C#
2. The game can only be run on a computer with Windows.
3. The game is running on a server.
4. Minimum two players can start a game.
5. Maximum four players can play.
6. The game will store a player’s information in the database (such as Username, Password, ranking)
7. Once logged in, the player will be added to chat.