User Requirements Specifications

Mastermind 21-05-2014

Contents

[**Introduction** 2](#_Toc391022712)

[**Use cases** 3](#_Toc391022713)

[**Add guess** 3](#_Toc391022714)

[**Start Game** 4](#_Toc391022715)

[**Create New Room** 4](#_Toc391022716)

[**Join Game** 5](#_Toc391022717)

[**Add Account** 5](#_Toc391022718)

[**Functional Requirements** 6](#_Toc391022719)

[**Non-functional Requirements** 7](#_Toc391022720)

[**Forms** 7](#_Toc391022721)

[**Login Form** 7](#_Toc391022722)

[**Register Form** 7](#_Toc391022723)

[**List of Rooms Form** 8](#_Toc391022724)

[**Room Form** 9](#_Toc391022725)

[**Architecture Diagram** 11](#_Toc391022726)

[**MoSCoW List** 11](#_Toc391022727)

[**Sequence diagrams** 12](#_Toc391022728)

[**Playing a Game** 12](#_Toc391022729)

[**Login** 13](#_Toc391022730)

[**Players join same Room** 14](#_Toc391022731)

# **Introduction**

The MDW project consists in creating a multi-player game application.

The following document consist of the Uses cases needed to analyse, design and develop the Mastermind multi-player game application.

We are doing this project under guidance of Mr. Michael Franssen.

# **Use cases**

## **Add guess**

**Goal:** To Add a new guess.   
**Actors:** User

**Precondition:** Program is open.

**Post-condition:** Desired guess is added

**MSS:**

1. User types the desirable’s four different digits that compounds a guess.

2. System checks whether the guess is valid.

3. System compares the guess with the hidden numbers and prints on the list box in the same row the guess with the respective feedback:

* A checkmark means that the user guess has one digit of the hidden numbers in the right position.
* A dot means that the user guess has one digit of the hidden numbers, but in the wrong position.
* The above feedback will repeat as needed, and if there is one or more digits from the guess that are not in the hidden numbers then there is no feedback.

4. Use case ends.

**Ext:**

2.1. If the four digits are not different, system prints a message to the user indicating that different numbers should be typed and user case ends without adding the guess.

## **Start Game**

**Goal:** To start game.   
**Actors:** User

**Precondition:** Program.

**Post-condition:** Game is started.

**MSS:**

1. User logged in.

2. System displays a message with two options:

* “Create New Room”, if user choose this option, go to Case Create New Room step 2 MSS.
* “Join Game” go to Case Join Game step 2 MSS.

3. Use case ends.

**Ext:**

2.1 User chooses button cancel, use case ends.

## **Create New Room**

**Goal:** To create a new Room.   
**Actors:** User

**Precondition:** Program is open and use case Start Game was chosen by the user.

**Post-condition:** anew Room is created.

**MSS:**

1. User chooses “Create New Room”

2. System creates a new Room with the name of the user and generates a new random hidden number (4 different digits).

3. User enters number of players and clicks “confirm” button.

4. System shows the form with attending player’s list.

5. System checks the number of players in the list if it is equal to the demanding number.

If it is full (number of players is equal to the demanding number), the game starts.

6. Use case ends.

**Ext:**

2.1 If User has already a personal room, system will generate a new random hidden number.

5.1 If the Room is not full but Room owner want to start, only owner can click “start” button to start game.

5.2 If user click “cancel” button, system will delete the Room and user will back to prevent form.

## **Join Game**

**Goal:** To connect the user with an existing Room.   
**Actors:** User

**Precondition:** Program is open and use case Start Game was chosen by the user.

**Post-condition:** User is connected with an existing Room.

**MSS:**

1. User chooses “Join Game”

2. System shows a list with the name of the existing lobbies and the amount of players in each of it.

3. User chooses one of the listed rooms and use case ends.

**Ext:**

2.1 If there is not lobbies listed, go to use case Start New Game with personal Room step 2 MSS.

## **Add Account**

**Goal:** To create a new user account.   
**Actors:** User

**Precondition:** Program is open.

**Post-condition:** anew user account is created.

**MSS:**

1. User chooses Add Account from the starting form

2. Users types his/her information.

3. System checks if the information is valid.

4. Account is added and use case ends.

**Ext:**

3.1 If the information is not valid, system shows the message “please, enter all information required”, go to step 2 MSS.

3.2 If the Account Name already exist, system shows the message “The Account xxxx ha’s been created, please, enter another Account Name”, go to step 2 MSS.

# **Functional Requirements**

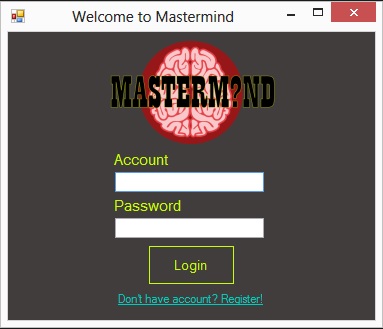
* The system must store the information of a created account.
* The system must compare the login information of a user with the stored accounts, if it found it then should let the user go to the List of Rooms Form.
* The system should be able to create a Room with the user name.
* The system should create a list with the users added to a Room (Join game) in the **XXXX FORM.**
* If a Room is created, the system must generate a new hidden number (four different digits).
* The system must check if the digits that user types are different and four.
* The System must compare the value and position of each of the four digits added by the user with the position and value of each of the digits of the hidden numbers.
* The system must return a check mark when the position and value of a digit added by the user is equal to the position and value of one of the hidden numbers.
* The system must return a dot if the value of a digit added by the user is equal to the value and not its position of one of the hidden digits
* If the user enter the four digits with the same position and value than the hidden value, system must show the message box that inform the user win the game and other users lose the game.
* System must be able to show the chat messages for all users in the same Room.

# **Non-functional Requirements**

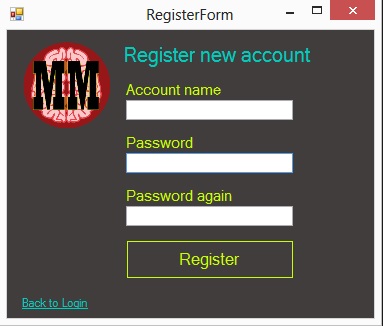
* System cannot let user guess character different than digits. System can only let user guess numbers.
* System cannot provide more or less than four random digits number.
* System cannot let user determine the hidden number.
* System cannot work without Internet.
* System cannot determine the time of game.
* System cannot let user who have no account to play the game.
* System cannot record the number of the user’s winning game.
* System cannot store the message of the guessing and chatting history after user exit the form.

# **Forms**

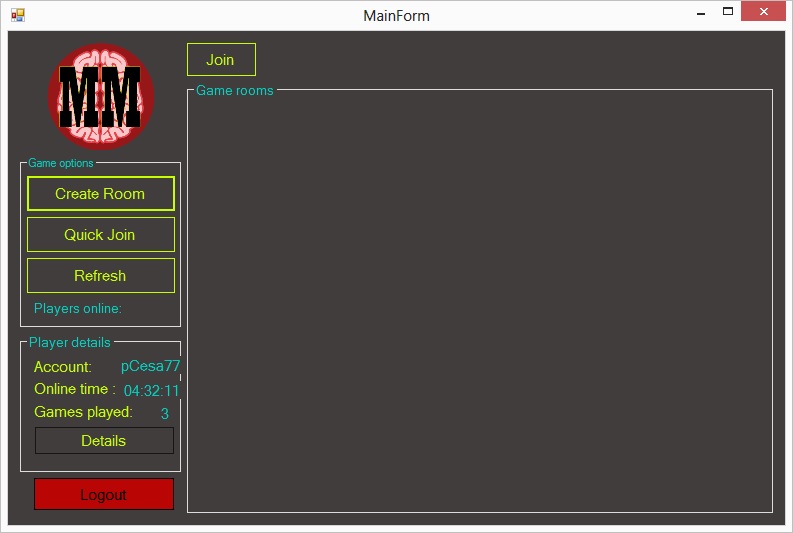
## **Login Form**



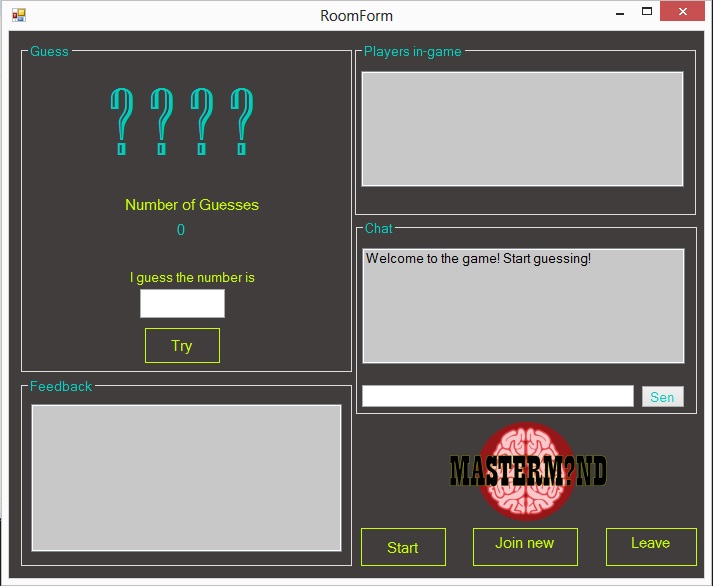
## **Register Form**



## **List of Rooms Form**



## **Room Form**



# **Architecture Diagram**

Mastermind

Server

Mastermind

Client

IAccount

IRoom

IGame

IChat

IGameCallback

IChatCallback

IAccount: CheckAccount(int ID,int Pas); AddNewAccount(int ID,int Pas);

IRoom: AddNewRoom(int OwnerID,int Num,list<Account> people); DeleteRoom(int OwnerID); AddRoomPeople(Account people); RemoveRoomPeople(Account people);

IGame: getFeedback(string n); getGuessNum(string name,int num);[checkNumCorrect(int num)]

IChat: getMessage(string n);

IGameCallback: sendFeedback(string n);

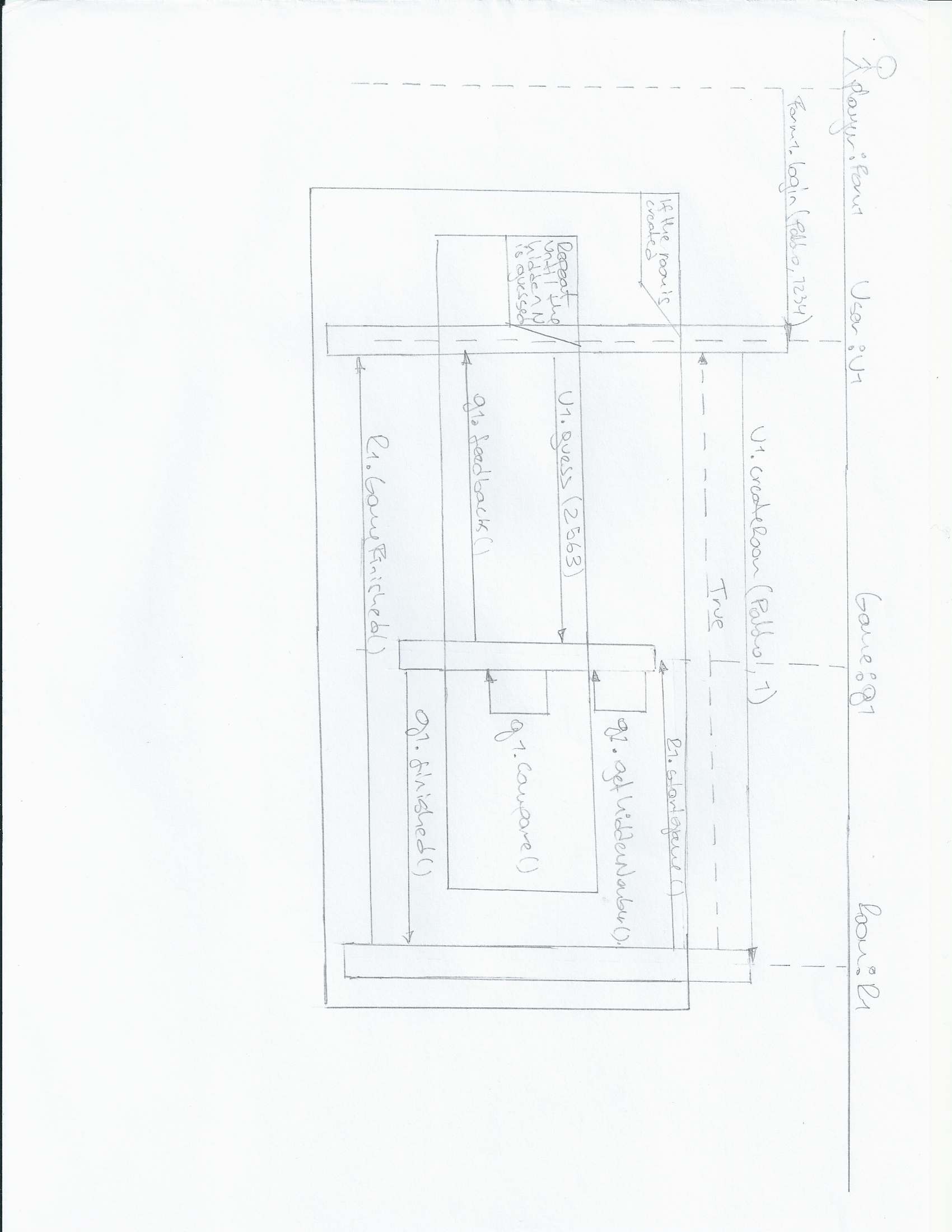
IChatCallback: sendMessage(string n);

# **MoSCoW List**

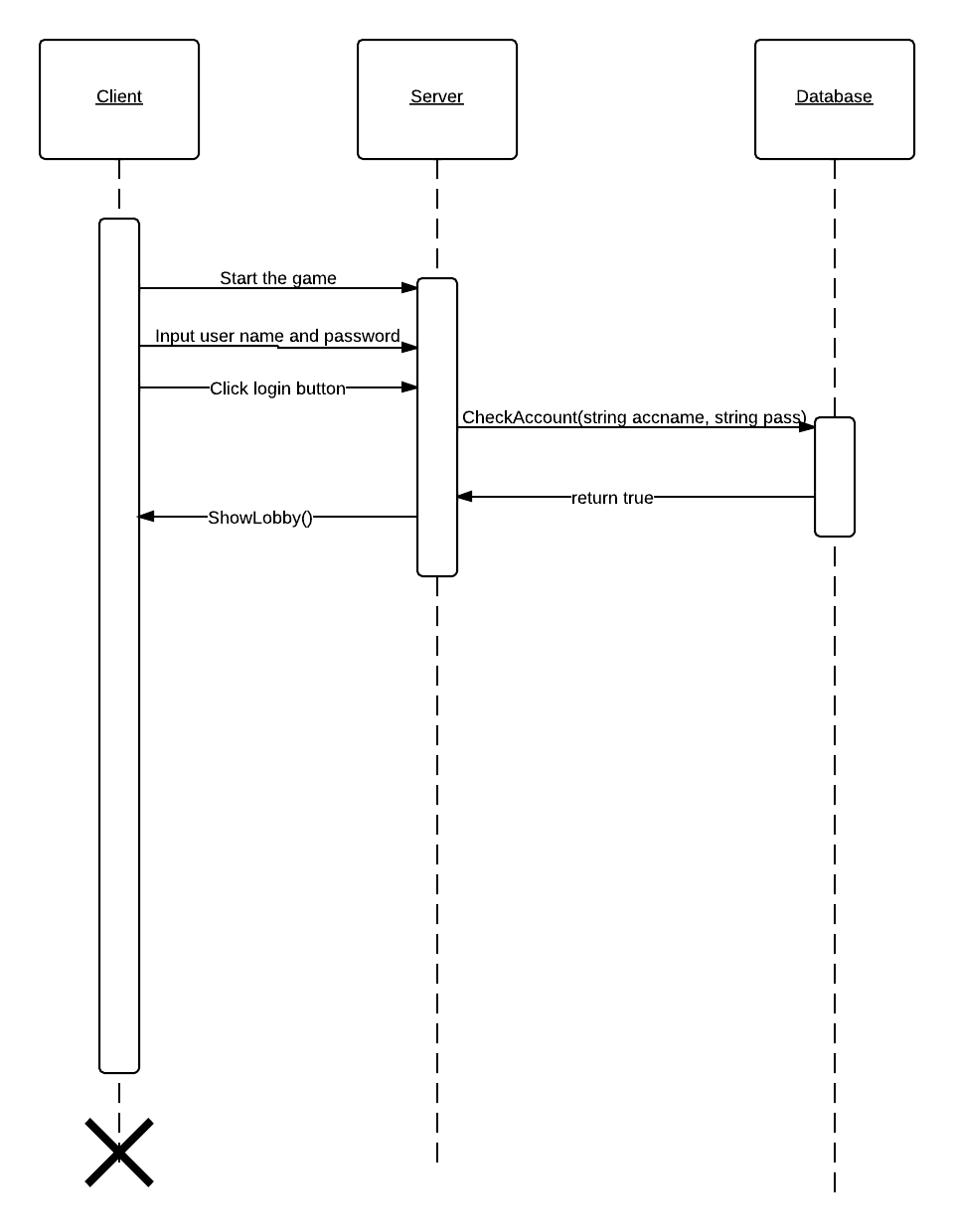
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Must | Should | Could | Will not |
| Add Guess | X |  |  |  |
| Start Game | X |  |  |  |
| Create New Room | X |  |  |  |
| Join Game | X |  |  |  |
| Create Account | X |  |  |  |
| Private rooms |  | X |  |  |
| Login | X |  |  |  |
| Game currency |  |  | X |  |
| Leave game | X |  |  |  |
| Bonuses for earning currency |  |  | X |  |
| Game tricks to interact with other players |  |  | X |  |
| Ranklist |  |  | X |  |

# **Sequence diagrams**

## **Playing a Game**



## **Login**



## **Players join same Room**

