Oren Farber - 2017

ID: 024910044

Matix Game

Object-oriented programming workshop - Object-oriented Analysis & Design Document

Table of Contents

[Introduction 3](#_Toc493971742)

[Matix Game – Concept and Rules 3](#_Toc493971743)

[Matix System Requirements Analysis 3](#_Toc493971744)

[Game Client Requirements 3](#_Toc493971745)

[Game Server Requirements 4](#_Toc493971746)

[Database 4](#_Toc493971747)

[Player Information 4](#_Toc493971748)

[Game Information 4](#_Toc493971749)

[User Stories 4](#_Toc493971750)

[Login and Registration to the Game Server 5](#_Toc493971751)

[Update Player Details 6](#_Toc493971752)

[Change Player's Password 6](#_Toc493971753)

[Multi Players Game 7](#_Toc493971754)

[Single Player Game 8](#_Toc493971755)

[Start a new game 8](#_Toc493971756)

[Playing the Game 9](#_Toc493971757)

[Get player statistics. 9](#_Toc493971758)

[User activities 10](#_Toc493971759)

[Login and registration to game server 10](#_Toc493971760)

[Update player details 11](#_Toc493971761)

[Multi Player Game 11](#_Toc493971762)

[Single Player Game 12](#_Toc493971763)

[Start a New Game 12](#_Toc493971764)

[Playing the game 13](#_Toc493971765)

[Get Player Statistics 13](#_Toc493971766)

[Matix System Architecture Design 14](#_Toc493971767)

[Game Server 14](#_Toc493971768)

[Game Management 14](#_Toc493971769)

[Matix Business Interface 14](#_Toc493971770)

[Business Layer Main Classes 18](#_Toc493971771)

[Communication 20](#_Toc493971772)

[Matix Service Interface 20](#_Toc493971773)

[Matix Service Callback Interface 23](#_Toc493971774)

[Database & Data Access Layer 25](#_Toc493971775)

[Database Tables and Views 25](#_Toc493971776)

[Data Layer Interface 27](#_Toc493971777)

[Sequence Diagrams 31](#_Toc493971778)

[Login and Registration to the Game Server 32](#_Toc493971779)

[Update User Details 33](#_Toc493971780)

[Change User Password 34](#_Toc493971781)

[Multi Player Game 35](#_Toc493971782)

[Start a New Game 36](#_Toc493971783)

[Playing the Game 37](#_Toc493971784)

[Start Playing with Server 38](#_Toc493971785)

[Playing with Server 39](#_Toc493971786)

[Get Player Statistics 40](#_Toc493971787)

[Game Client Design 41](#_Toc493971788)

[Pages and Controls 41](#_Toc493971789)

[Client components 51](#_Toc493971790)

[User Experience 53](#_Toc493971791)

[Board elements 53](#_Toc493971792)

[Matix System Installation and Operation 55](#_Toc493971793)

[Logging Tool 55](#_Toc493971794)

[Matix Game Server Installation 55](#_Toc493971795)

[System Database Installation 55](#_Toc493971796)

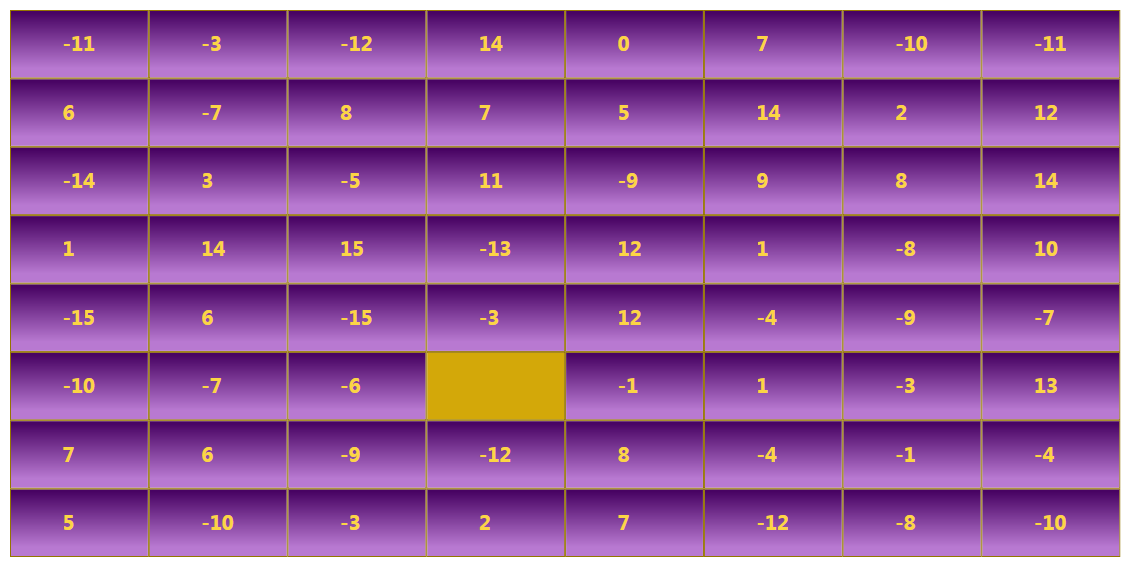
[Game Client Installation 55](#_Toc493971797)

# Introduction

This document is an object-oriented analysis and design document for a board game named Matix. The document describes the game concepts and rules, analyze and identify the software requirements and specifications. Also it describes the implementation of the conceptual model produced during object-oriented analysis.

# Matix Game – Concept and Rules

The Matix game is a board game for two players. It has 8\*8 cells with random integer values between -15 to 15 and one token. The token is a cell marked with a different color. A player plays by moving the token horizontally or vertically by double clicking on a non-used cell on the board, one player can move the token horizontally and the second can move the token vertically. The purpose of the game is to collect the highest score which is the summary of all the cell values a payer collected. The game ends when there is no free cell for the token to move.



# Matix System Requirements Analysis

These sections describe the requirements for this game system. It describes the required parts and the use cases a player can have.

The system should be a client server system and should allow several clients to connect the server at the same time. The system should keep information about the users, the progress and the result of the games.

## Game Client Requirements

The client should be a windows application with a graphical user interface that allow the user to connect the server and play the game. The application should be installed on the user client machine. The client application shows the user its game board as it receives from the server and allows him to play the game according to the games rules. The client should send the game changes to the server and to reflect the second player actions as it received from the server. The client application should supply help and instruction for the client.

## Game Server Requirements

The game server should be a windows application without user interface. The server is the responsible for managing the games of the clients connected to it. The server should support several games and several clients connected to it at the same time. The server should manage players and games information and save it in a database. The server is the game manager and responsible for generating a new board for a new game and enforce the game rules on player's actions that sends to it.

## Database

The database should save all the relevant information for players and games. It should have connection only to the server. The database should allow us to truck the process of a game by saving all the actions players do during the game.

### Player Information

For players, we should use email address as an identification key and should be presented to other players with his nickname.

* First name
* Last name
* Nick name
* Email
* Password
* Type

### Game Information

For each game, we must save the it's initiate time the players of the game and its generated board. During the game, we should keep the activity of the games so we can reconstruct the process of the game.

## User Stories

The section describes the following user stories

* Log in and registration to the game server
* Update player's details
* Change player's password
* Multi player game
* Single player game
* Start a new game
* Playing the game
* Get player statistics.

### Login and Registration to the Game Server

While a player starts the client application he needs to connect to the server. To identify the player, the system needs a login and registration process. The client should allow the player to save its login properties locally. A user should register to the server so it can save its games data. Registration & login should base on player email address. The password the user uses must not be kept in the database as a plain text. The system can save a hash value generated from that password. For presentation, a player must use a unique nickname that will identity him among other players.



### Update Player Details

A registered player can update its details on the game server. The player must be logged in at that time. The process should allow the player to update all its properties except email address and password.



### Change Player's Password

A player can change it password to a new one as a separate process.



### Multi Players Game

A player can choose to play with other players by selecting multi player game. The player should be added to a waiting list and to receive a list of other waiting players so he can choose one to play with.



### Single Player Game

The player can choose to play the game with the server as a second player.



### Start a new game

When a second player is selected, a new game can be started. The server should generate a new game and sends it to the players. The players receive the board generated by the server and show it with the player's details.



### Playing the Game

On its turn, a player can move the token on board to a free cell. The client application should enforce the game rules and instruct the player with current turn and playing direction.



### Get player statistics.

A player can receives statistics information about his previous games and score.



## User activities

The section lists the main activities of the system and show an activity diagram for each activity.

* Log in and registration to the game server
* Update player details
* Multi Player Game
* Single Player Game
* Start a new game
* Playing the game
* Get player statistics.

### Login and registration to game server

The diagram describes the login and registration process. First a user must register and then he can be logged in to the server. The registration process should be done only once.



### Update player details

A registered player can update its details on the game server.



### Multi Player Game

While a player selects to play with a human player the server adds him to a waiting players list and send back a list of waiting players.



### Single Player Game

While a player selects to play with the server. The server generates a new game and sends it details to the client



### Start a New Game

While a game is started the server generates a random board, select for each player its direction. For multi players game the server chooses who will start the game. For single player game, the human player always starts.



### Playing the game

On its turn, the player can move the token to a new cell by double clicking the selected one. The game page should inform the player who's should play. The server should verify that the action is valid, update the database with the action and check that the game is not ended yet.



### Get Player Statistics

User can show its playing results



# Matix System Architecture Design

These following sections describe the system architecture, its components and how the system should be implemented.

## Game Server

The game server should be created as a windows service application and implement WCF service for communication between the clients and the server. The server will be connected to a dedicated database that saves all the relevant players and games information.

All three parts, communication, business and data access should be implemented as separated class libraries so we can handle the software as separated layers.

### Game Management

This section describes the business layer that responsible for managing the games, create and update boards, and managing player's data.

#### User login and registration

The business layer handles the users and allows the client to register a new user. Each new user should be saved in the database. The layer manages the login process for existing players and give the interface for update details and change password.

#### Wait for second player

While a player connects to the server and select to play with another player, he should receive a list of waiting players if exist. The server should add the player to a waiting players list and send notification to all other waiting clients with the updated list.

#### Select a player and start a new game.

While a player selects a second player from the waiting list. The server should start a new game and sends the generated board to the clients.

#### Playing the game

The server notifies each player when its turn and wait to receive a game action. When that message receives, the server check that it is a valid action, update the board of this game and add the information to the database. When updating is ended the server notifies the first client with acknowledge and update the second client with the board and player changes and that his turn to play.

#### Ending the game

When the server receives a game action it should check whether the second player can move the token, on its turn, to an unused cell. If there is no free cell the game is ended and the server should calculate the score for each player, update the database and notify the clients who is the winner.

### Matix Business Interface

The IMatixBusinessInterface expose the methods other layers can access in the business layer. The Communication layer that expose to the clients is using that interface to inform the business layer with messages from the client.

#### Set Matix WCF Service

Allow to set an instance of the communication layer.

##### Parameters

* Matix WCF Service (instance reference)

#### Client Disconnected

Notification that a user disconnected.

##### Parameters

* Email (string)

#### Add Player

Add a new user to the system

##### Parameters

* First Name (string)
* Last Name (string)
* Nickname (string)
* Email (string)
* Password (string)

##### Reply

* RegistrationResult (structure)

#### Update Player

Update player details

##### Parameters

* Email (string)
* First Name (string)
* Last Name (string)
* Nickname (string)

##### Reply

* Operation Status (enumerator)

#### Change Password

Change the current user password

##### Parameters

* Email (string)
* Old Password (string)
* New Password

##### Reply

* Operation Status (enumerator)

#### User Login

Allow the user to login to the system. On reply the user receives its nickname. The email and password compared to those saved in the database. The IP address is saved for later use.

##### Parameters

* Email (string)
* Password (string)
* IP Address (string

##### Replay

* Login Result (structure)

#### User Logout

Allow the user to logout the system and report a reason. Allow the server to remove the client from its internal dictionaries.

##### Parameters

* Email (string)
* Reason (string

##### Replay

* Operation Status (enumerator)

#### Get Waiting Players List

Allow the user to register as a waiting player and to receive a list of other waiting players so he can select one to play with.

##### Parameters

* Email (string)

##### Replay

* Waiting Player Result (structure)

#### Start Playing with Player

Allow the user to select a player to play with

##### Parameters

* Email (string)
* Nickname (string

##### Replay

* Operation Status (enumerator)

#### Start Playing with Robot

Allow the user to select a player to play with the server. This method starts a task of creating a new game.

##### Parameters

* Email (string)

##### Replay

* Operation Status (enumerator)

#### Set Game Action

Allow the game client to report a new selected token. After validation, the method updates the game instance the database and start a task to notify the other player with the change.

##### Parameters

* Email (string)
* Row (integer)
* Column (integer)

##### Replay

* Operation Status (enumerator)

#### Get Players Statistics

Allow the user to get its statistics information.

##### Parameters

* Email (string)

##### Replay

* Players Statistics Result (structure)

#### Remove from Waiting Players

Allow the game client to report that the player is not waiting to play.

##### Parameters

* Email (string)

#### Quit the Game

Allow the game client to report that the player choose to quit the game.

##### Parameters

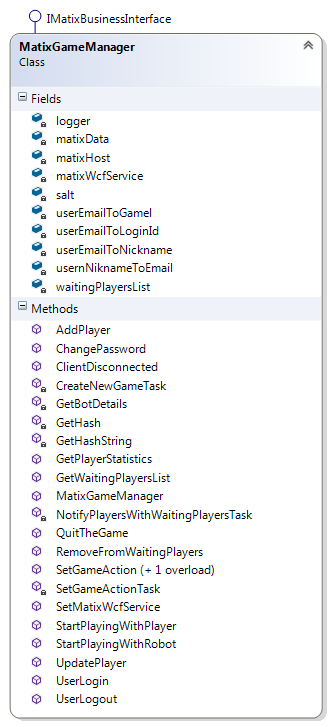
* Email (string)

### Business Layer Main Classes

This section describes classes of the server business layer. This layer responsible for managing the games.

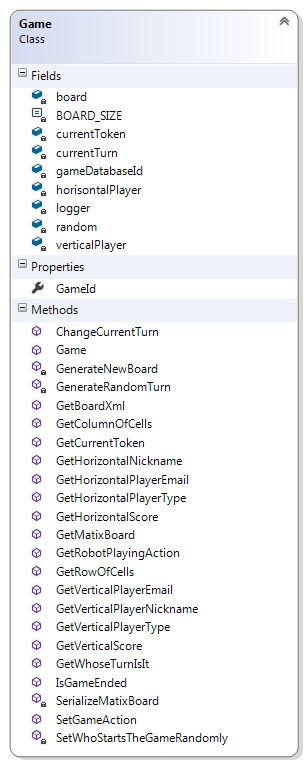
#### Matix Game Manager

The class MatixGameManager is the main business layer class and implements the IMatixBusinessInterface. The class initialize all the other server layers communication and data access.



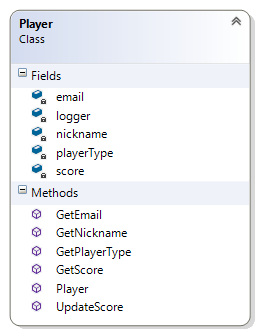
#### Game

The Game class encapsulates the properties of a Matix Game and allows to have relevant methods



#### Player

The Player class encapsulates the properties of a player.



## Communication

This section describes the communication layer. This layer is implemented as a separated library using WCF technology. The communication protocol supports Duplex type and allows sending messages from the client to the server and to use callback method for notifying the client on changes from the server and other players.

### Matix Service Interface

That communication WCF interface is implemented in the server and contains the following methods.

#### Operation Status Enumeration

Enumerate the available result status of operation

* Success,
* Failure,
* Invalid Email,
* Invalid Password,
* Operation time Out,
* Rejected,
* Invalid Action

#### Login Message

The message sent from the client with login parameters

##### Parameters

* LoginData (structure)

##### Reply

* LoginResult (structure)

#### LoginData

The class contains the player information for login message.

* Email (string)
* Password (string)

#### LoginResult

The class contains the result information for login message

* Operation Status (enumeration)
* Player's Nickname (string)

#### User Registration

The message sends registration data from a player to the server.

##### Parameters

* UserInformationData (structure)

##### Reply

* RegistrationResult (structure)

#### UserInformationData

* Email Address (string)
* First Name (string)
* Last Name (string)
* Nickname (string)
* Password (string)

#### RegistrationResult

* Operation Status (enumeration)
* Message (string)

#### Update User Details

The message allows the user to update its first name last name and nick name.

##### Parameters

* UserInformationData (structure)

##### Reply

* Operation Status (enumeration)

#### Change player password

The message allows the user to change its password in the database.

##### Parameters

* Email Address
* Old Password
* New Password

##### Reply

* Operation Status (enumeration)

#### Get Waiting Player List

The message returns a list of current waiting players with some statistics information

##### Parameters

* The current players email address (string)

##### Reply

* List of WaitingPlayerResult structure

#### Waiting Player Result

The class contains the information of the current waiting players and the operation status of the message.

* List of WaitingPlayer (structure)
* Operation Status (enumerator)

#### WaitingPlayer

The class contains information of waiting players that can be shown to the player.

* Nickname (string)
* Total Games (integer)
* Number of winnings (integer)
* Total Score (integer) – Can be negative value

#### Select Player to Play

The message sent from the first player to the server after the waiting players received and the user select one of the players to play with.

##### Parameters

* Email Address (string) – The email address of the sender
* Nickname (string) – The nickname of the second player

##### Reply

* Operation Status (enumerator)

#### Select Robot to Play

The player selects to start a new game with the server.

##### Parameters

* Email Address (string) – The email address of the sender

##### Reply

* Operation Status (enumerator)

#### Quit the Game

Notify the server that a player quite the current game he is playing

##### Parameters

* Email Address (string)

##### Reply

* None

### Matix Service Callback Interface

That WCF call back interface defined in the server and implemented in the client. The interface contains the following methods.

#### Update Waiting Player

The method allows the client to be notified while the waiting players list is changed.

##### Parameters

* WaitingPlayerResult (structure)

#### Starting a New Game

The method allows the client to be notified when a game is starting and to receive the generated game board.

##### Parameter

* Matix Board (structure)
* Horizontal Nickname (string)
* Vertical Nickname (string)
* Who Is Starting (enumerator)

#### Update Game Action

The method allows the client to be notified with the game action the other player did.

##### Parameters

* New Token Row (int)
* New Token Column (int)
* The Selected Token Score {int)

#### Update Game Ended

The method allows the client to be notified that the game is ended.

##### Parameters

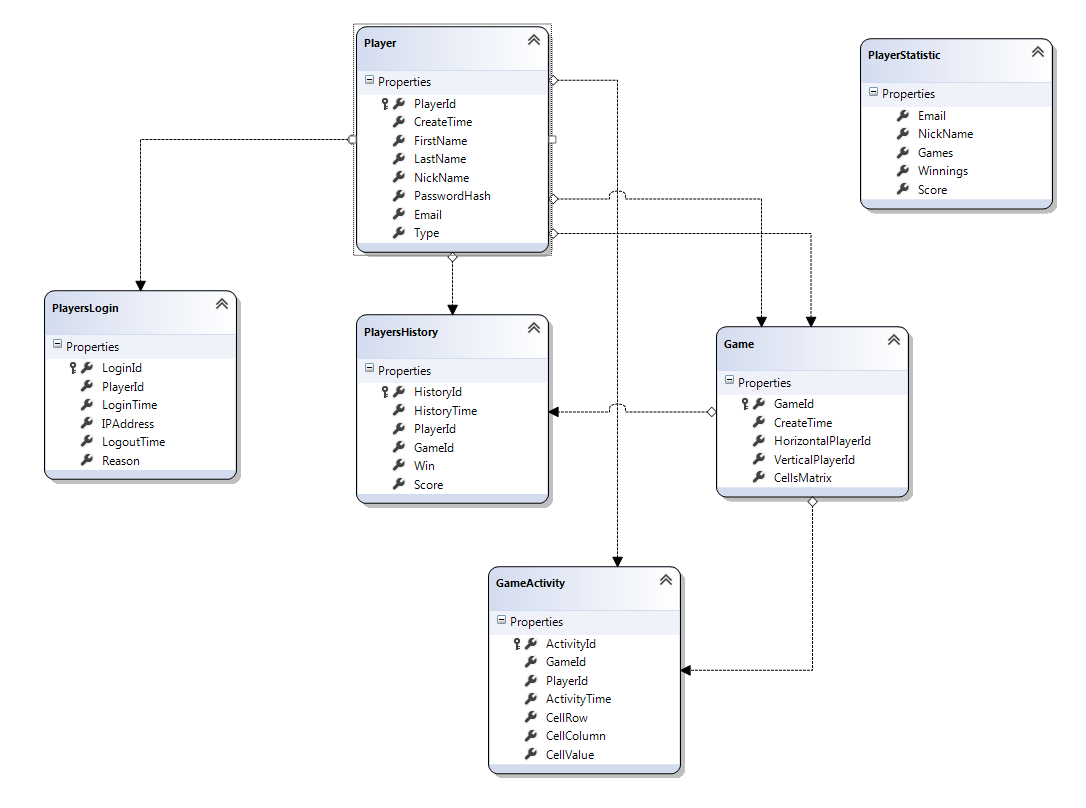
* Winner Nickname (string)
* Winner Score (int)

## Database & Data Access Layer

This section describes the data access layer that contains database interface and the database tables. The layer use LINQ to implement the queries to the database.

### Database Tables and Views

This section describes the database tables. The purpose of the database is to store players and game information.



#### Players

This table contains player's unique information. The table contains one record for each player register to the system.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Player Id | Create Time | First Name | Last Name | Nick Name | Password Hash | Email | Type |
|  |  |  |  |  |  |  |  |

* Player Id - Unique player id – Primary key
* Create Time – The first time a player registers to the game
* First Name – Player first name
* Last Name – Player last name
* Nick Name – Players nick name - must have a unique value
* Password Hash - A generated hash string from the password.
* Email – Players unique email address – must have a unique value
* Type – Whether the player is a Human or a Robot

#### Players Login

The table contains information of player's logins.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Login id | Player Id | Login Time | IP Address | Logout Time | Reason |
|  |  |  |  |  |  |

* Login id – Unique id for this record – Primary key
* Player Id – Unique player id – Foreign key from Players table
* Login Time – Login event time
* IP Address – Players client IP address
* Logout Time – Log out event time
* Reason – A string for short description

#### Games

This table stores game unique information.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Game Id | Create Time | Horizontal Player ID | Vertical Player ID | Cells Matrix |
|  |  |  |  |  |

* Game Id - Unique id for this record – Primary key
* Create Time – Create time event
* Horizontal Player Id – The Id of the horizontal player – Foreign key from Players table
* Vertical Player id – The Id of the vertical player - – Foreign key from Players table
* Cell Matrix – An XML matrix that contains the generated game board.

#### Game Activities

This table describe the process of the game. It contains a record for each move a player did during the game.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Activity ID | Game ID | Player ID | Activity Time | Cell Row | Cell Column | Cell Value |
|  |  |  |  |  |  |  |

* Activity Id - Unique id for this record – Primary key
* Game Id – The id of the game in Games table - Foreign key
* Player Id – The Id of the player that do the game activity - Foreign key from Players table
* Cell Row – The row number of the new selected cell
* Cell Column – The column number of the new selected cell
* Cell Value – The value of the selected cell

#### Players History

This table is a connection table to record the game result for each player and game.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| History ID | History Time | Player ID | Game Id | Win | Score |  |
|  |  |  |  |  |  |  |

* History Id - Unique id for this record – Primary key
* History Time – The time of the event, should be the time the game ended
* Player Id – The player Id this information refers to - Foreign key from Players table
* Game Id – The game Id this information refers to - Foreign key from Games table
* Win – True whether this player wins this game
* Score – The score of the player in the game- This value should be the summary of all the cell values selected by that user during the game.

#### Player Statistic

This view summarizes players statistics information and contains the following information. It can be used to receives player statistics information.

* Email – The players email address
* Nickname – The players nickname
* Games – The total number of games the player plays
* Winnings – The number of winnings the player has
* Score – The average score per a game

### Data Layer Interface

This section describes the interface the DAL expose to update or retrieve information to and from the database.

#### Is Player Email Exist

Check whether an email address is already used.

##### Parameters

* Email (string)

##### Return

* True if exist otherwise false

#### Get Player Nickname

Get player's nickname based on his email

##### Parameters

* Email (string)

##### Return

* Player nickname (string)

#### Check Email And Password Hash

Check that the email with its password are the same in the database.

##### Parameters

* Email (string)
* Password Hash (string)

##### Return

* True if OK otherwise false

#### Add Player

Add a new player to the database and return its id.

##### Parameters

* First name (string)
* Last name (string)
* Nick Name (string)
* Email address (string)
* Password Hash (string)

##### Return

* Throws exception on error

#### Update Player Information

Update player information

##### Parameters

* Email (string)
* First name (string)
* Last name (string)
* Nick Name (string)

##### Return

* Throws exception on error

#### Change Password

Update player password in the database

##### Parameters

* Email (string)
* New Password Hash (string)

##### Return

* Throws exception on error

#### Payer Login

Save a record that a player login to the system

##### Parameters

* Email address – The user registered email
* Password Hash – The generated password hash
* IP Address – The client IP address

##### Return

* Login id (long)

#### Player Logout

Update that player logout from the system. The method updates the login record with the current time and reason.

##### Parameters

* Login Id (long)
* Email (string)
* Reason (string)

##### Return

* Throws exception on error

#### Get Player Statistics

Get player statistics information

##### Parameters

* Email (string)

##### Return

* PlayerScoreData (structure)

#### Get Waiting Player Data

##### Parameters

* Email (string)

##### Return

* PlayerScoreData (structure)

#### Create New Game

Creates a new game record in the database. The method returns the new created game id.

##### Parameters

* Horizontal Player Email (string)
* Vertical Player Email (string)
* Board cells as XML (string)

##### Return

* Game Id (long)

#### Add Game Action

Update database with game activities.

##### Parameters

* Email (string)
* Game Id (long)
* Row (int)
* Column (int)
* Value (int)

##### Return

* Throws exception on error

#### Add Player History

Update the database that a game was ended and that we have a winner. The record is added for each player

##### Parameters

* Email (string)
* Game Id (long)
* Win (bool)
* Score (int) – Can be a negative value.

##### Return

* Throws exception on error

## Sequence Diagrams

The next sections describe the main actions int the application and show a sequence diagram for each one of those actions.

* [Login and Registration to the Game Server](#_Login_and_Registration)
* [Update Player Details](#_Update_User_Details)
* [Change User Password](#_Change_User_Password)
* [Multi Player Game](#_Multi_Player_Game)
* [Start a New Game](#_Start_a_New)
* [Playing the Game](#_Playing_the_Game)
* [Start Playing with Server](#_Start_Playing_with)
* [Playing with Server](#_Playing_with_Server)
* [Get Player Statistics](#_Get_Player_Statistics)

### Login and Registration to the Game Server

The section describes the sequence of operations between application layers for registration and login scenarios.



### Update User Details

The section describes the sequence of operations between application layers for updating user details.



### Change User Password

The section describes the sequence of operations between application layers while the user changes its password.



### Multi Player Game

The section describes the multi-player game till the game is starting. In this process, the player registers as a waiting player and receives a list if all other waiting players listed in the system to play with.



### Start a New Game

The section describes the sequence of starting a new game with another player



### Playing the Game

The section describes the sequence of operations between application layers while a player plays the game.



### Start Playing with Server

The section describes the sequence of operations between application layers while starting a game with the server.



### Playing with Server

The section describes the sequence of operations between application layers while playing the game with the server. On that scenario, the server selects its action instead of notifying the other player.



### Get Player Statistics

The section describes the sequence of operations between application layers while player select to show its statistics or while a game is ended.



## Game Client Design

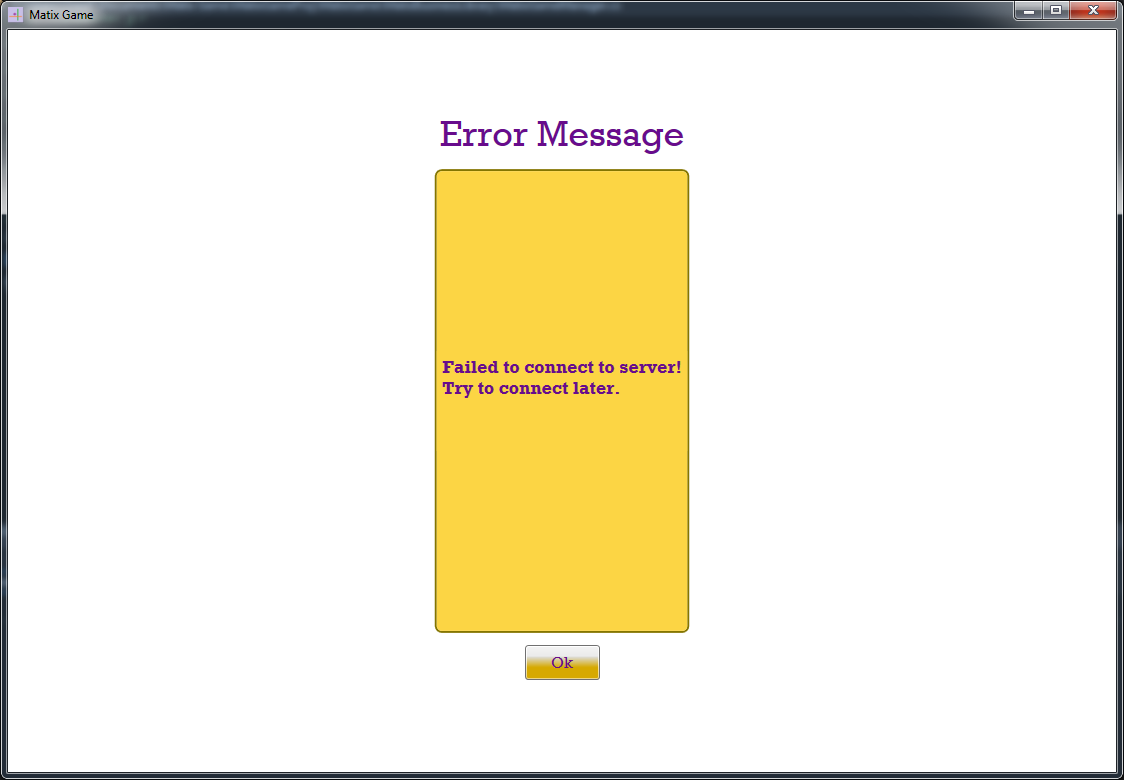
The game client application should be a WPF windows application. The application should allow the player to connect and login to the game server and to allow him to play the game.

### Pages and Controls

The section describes the all the pages and user control we should use.

#### Error Page

The Error page shows an error in case the client could not connect to the server.

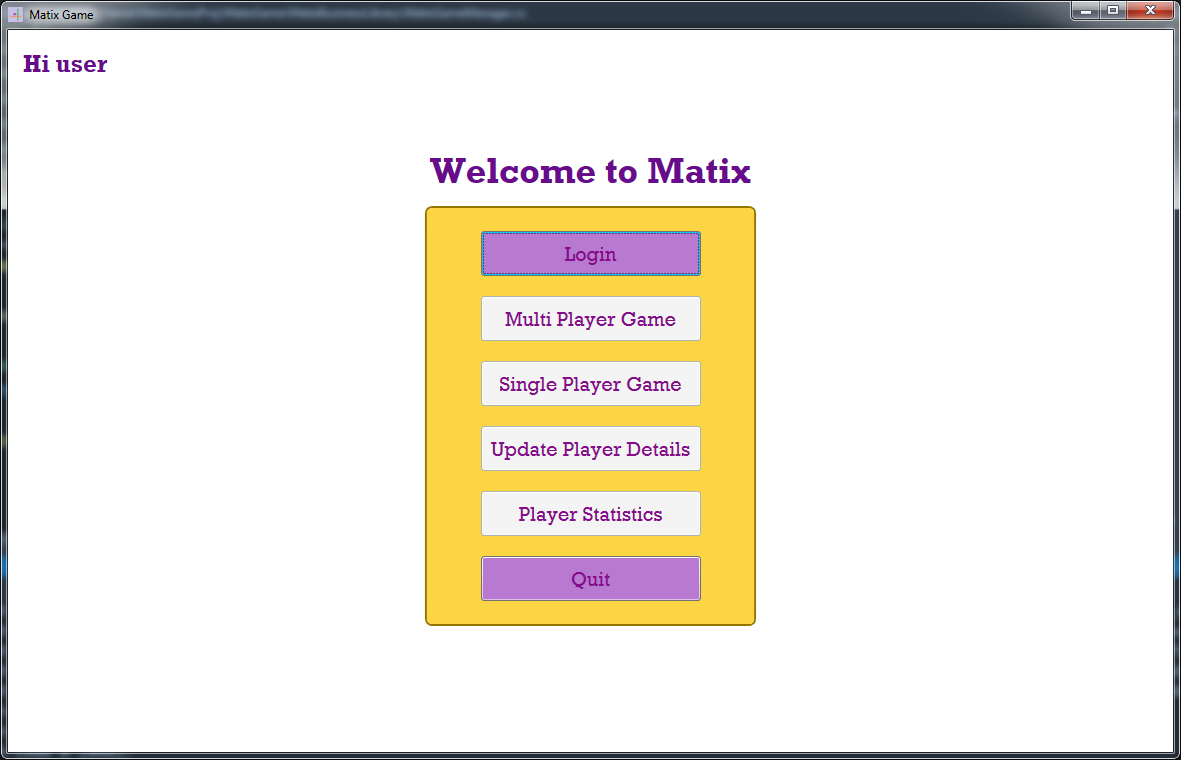
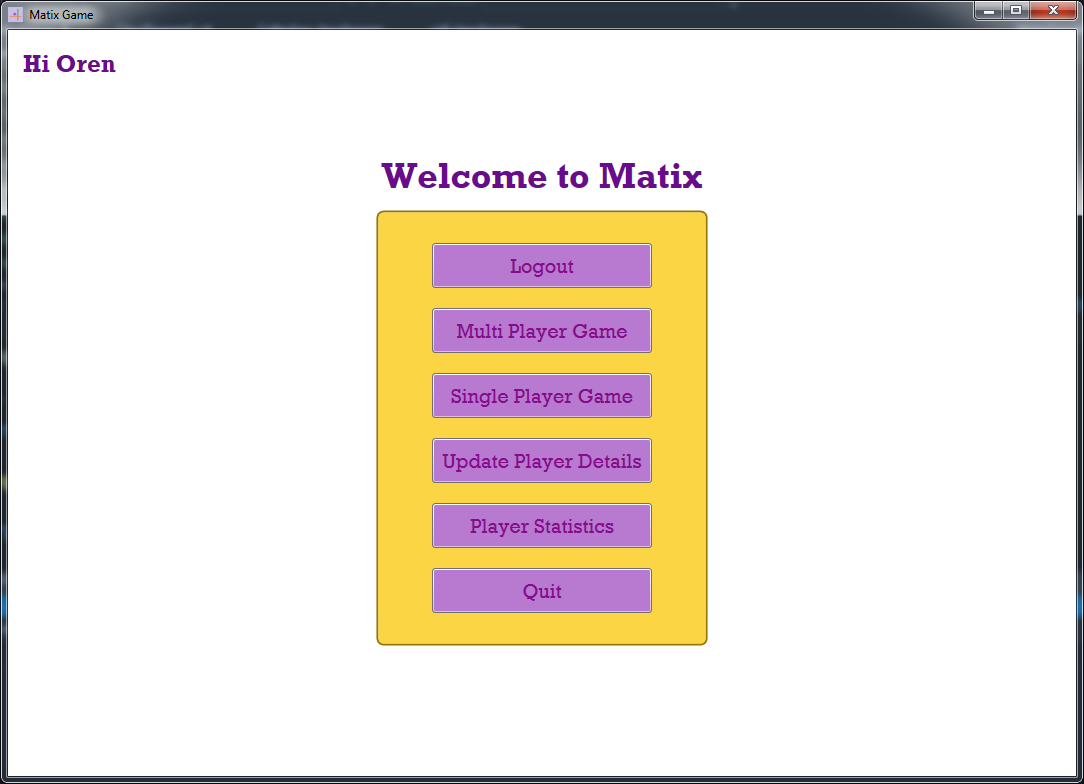


#### Welcome Page

The welcome page opens at client application start up and allows the user to select the requested operation.

* Login / Logout
* Multi Player Game
* Single Player Game
* Update Player Details
* Player Statistics
* Quit

The page has two forms the first one is when there is no logged in user. In that case the user must login to select another option. In the second form, the user is already logged in so the button changed to logout and all other option are available.

#### Login Page

The login page allows the user to login into the system using his email address and password or selecting to register in case of a new user. The page saves user email and password in the system settings to connect the user automatically when its available.



#### Registration Page

The registration page allows the user to register its basic information to the server. On registration, the user must update the following parameters

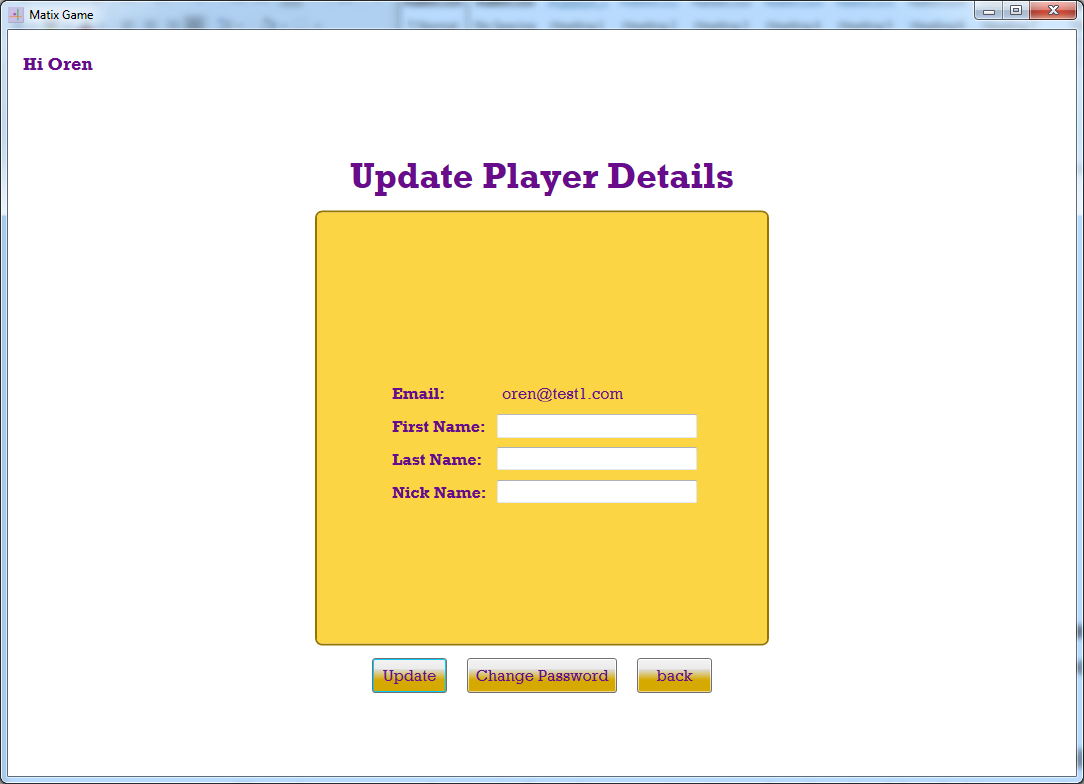
* First Name
* Last Name
* Nick Name
* Email address



#### Update player details

The update page allows the user to update the following fields

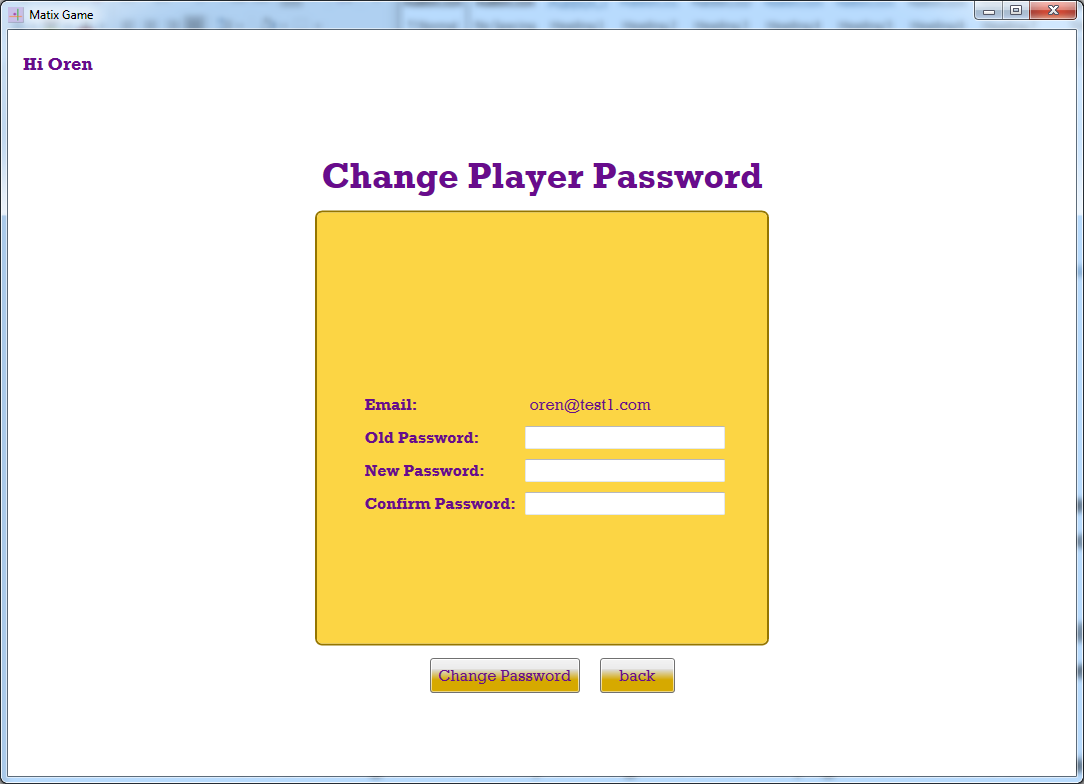
* First Name
* Last Name
* Nick Name



#### Change Password

The page allows the user to change its current password. The email is a key and can't be changed. The page should verify that all the parameters are available and that new and confirmed password fields contains the same value.

* Old Password
* New Password
* Confirm Password



##### Board User Control

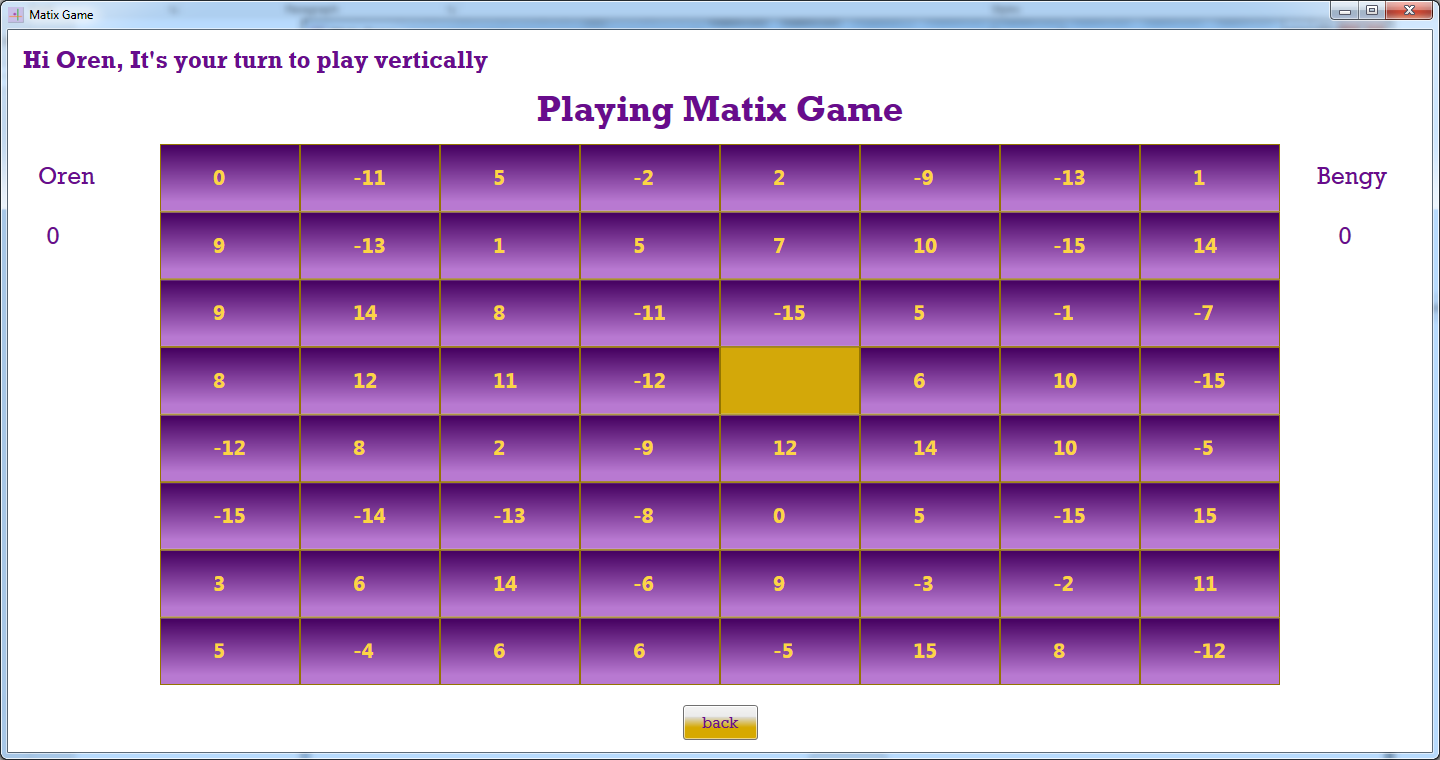
The board should be implemented as a user control. The board shows 64 cells with integer values. The player can select a cell and to receive its value by double clicking on the cell.

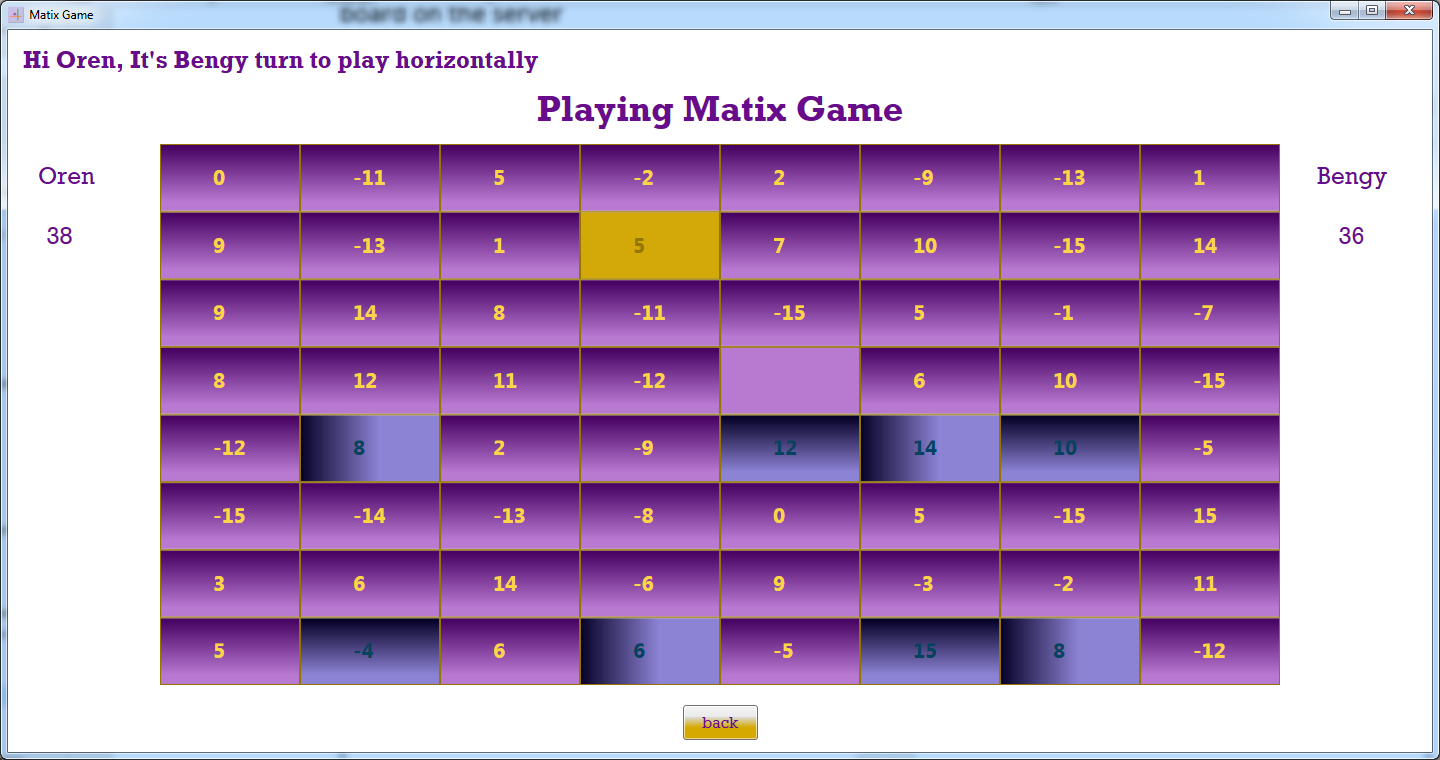
##### Circular Progress Bar User Control

Use a waiting control on top of the game after the page is shown and still waiting to the generated board on the server

#### Game Page

The game page allows the player to play the game. It shows the board and players nickname and score. The page allows the player on its turn to move the token to a valid position. The page embedded the Board UserControl to show and to allow playing. The first image shows the beginning of a game and the second one shows the game during playing.





#### Players List Page

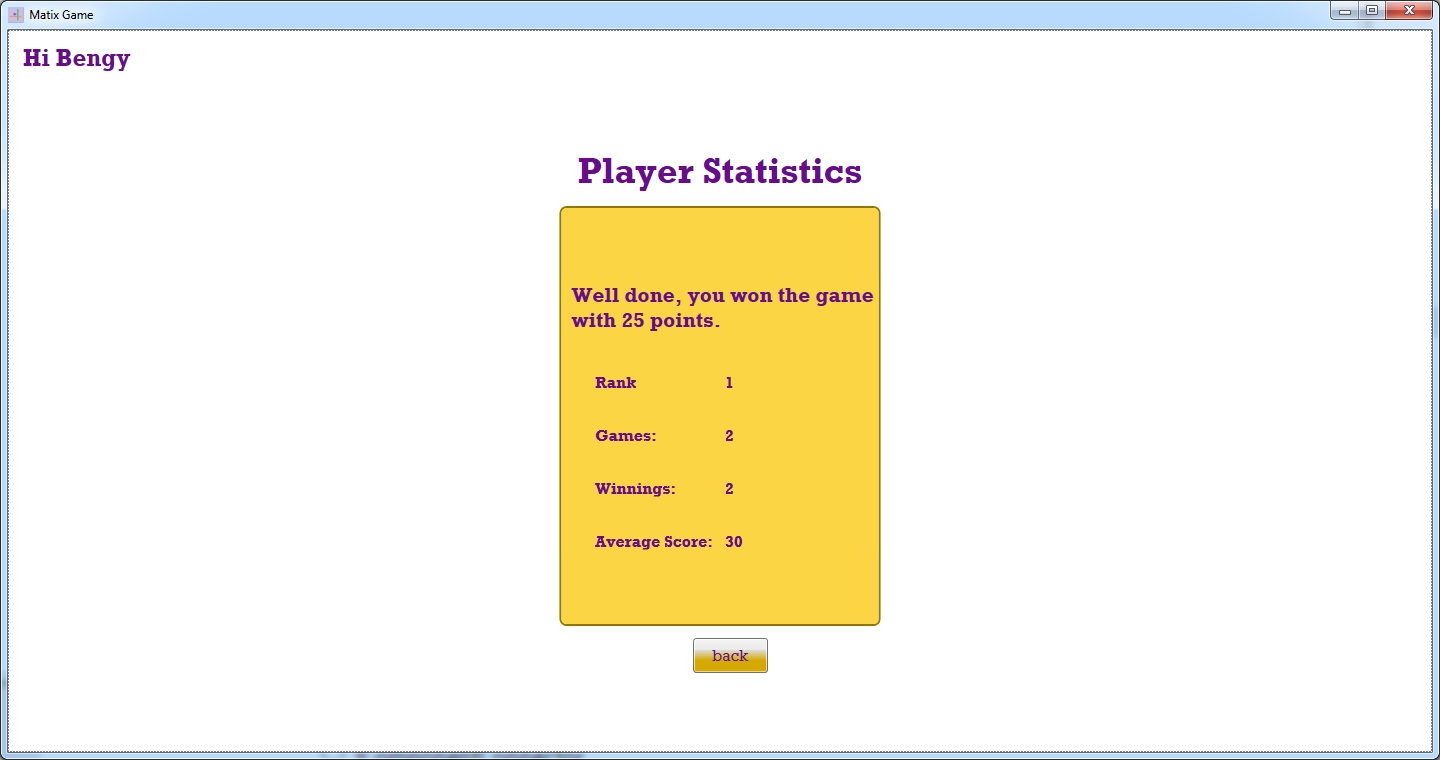
While a user selects to play with another player he should receive the list of currently waiting players. The page shows the players details and allows the user the choose a player to play with.



#### Player Statistics Page

The page shows the user statistics. The application should show it as a user request or at the end of a game.



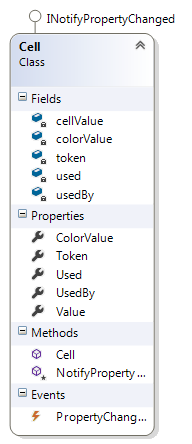


### Client components

This section describes the major components, that are not pages, which are part of the game client

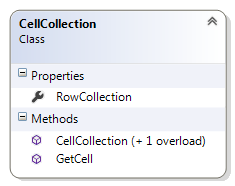
#### Cell

A cell represents a square in the game board.



#### Cell Collection

The class encapsulates a list of lists of cells.



#### Board

The board is a matrix of cells with 8 rows and 8 columns. The class derived from UserControl and embedded in the game page. The board presents the cells and allow the player to perform a game action by selecting a new token location. The class responsible to validate that the action the player act is valid according to the game rules.

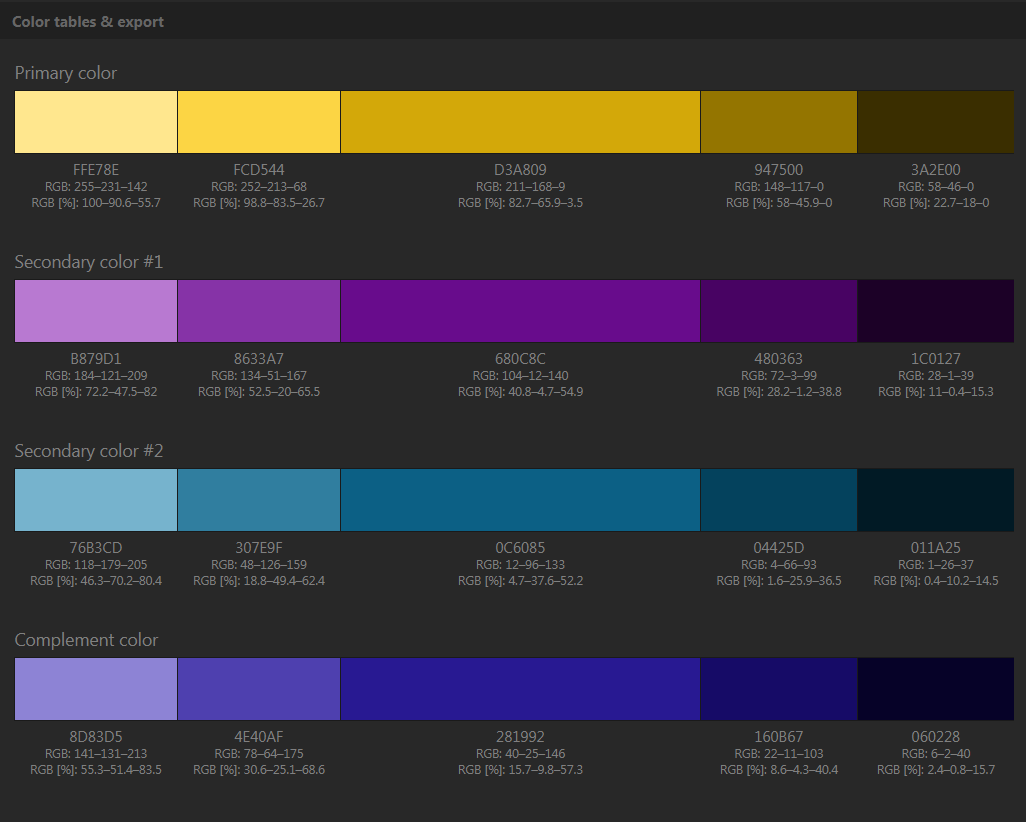
#### 

## User Experience

The section descries a font and color should use for this game client.

#### Colors

This section describes the colors selected for the application. Basically, I choose to use colors from the following palette.



#### Font

The font selected for the client application is 'Rockwell'.

### Board elements

This section describes the board elements and their meaning.

#### Unused Cell

Unused cell is a cell that were not selected yet by a player. On its turn and playing direction, a player can select a such kind of a cell.



#### First Token Cell

On the beginning of the game the server selects the first token, that has no value. At first it has an active token color



After a player select a new token the first token, that doesn't belong to any player chnges to a clean purple color.



#### Token Cell

While a player selects a token, it become the active on and shows the value the player collects to its score.



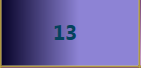
#### Vertical Player Selection

The cell indicates that is was selected by the vertical player.



#### Horizontal Player Selection

The cell indicates that is was selected by the horizontal player.



# Matix System Installation and Operation

This section describes the installation and other operation needed to successfully install the game server as a windows service create the database and install a client.

## Logging Tool

All parts of the system should have log file that describes the flow of the software behavior. We should create one log file for the client and a second log file for the server both can use log4net for implementing the logger. The log folder can be at the current running folder.

## Matix Game Server Installation

The section describes the operations needed to install the game service

* Install the service
* Add access rights to Network Service to the service location folder.

In order to install the game server as a service run the following command. Browse to the bin directory where MatixGameService.exe is located and run the following command.

***Installutil MatixGameService.exe***

If you have modified the service that is already installed, you can uninstall it by using following command:  
***Installutil /u MatixGameService.exe***

## System Database Installation

This section describes the scripts crating the Matix database. The database is based on SQL server and an instance of database service version 2008 or later should be installed. It can be installed on the Game Server machine or on a separated one.

First, we should create the database using the following script.

* [Database\CreateMatixDatabase.sql](Database/CreateMatixDatabase.sql)

After the database created we can add the tables using the following scripts

* [Database\CreatePlayersTable.sql](Database/CreatePlayersTable.sql)
* [Database\CreatePlayersLoginTable.sql](Database/CreatePlayersLoginTable.sql)
* [Database\CreatePlayersHistoryTable.sql](Database/CreatePlayersHistoryTable.sql)
* [Database\CreateGamesTable.sql](Database/CreateGamesTable.sql)
* [Database\CreateGameActivitiesTable.sql](Database/CreateGameActivitiesTable.sql)

## Game Client Installation

Currently use the release folder as in installation folder. Copy the folder to the destination and run the application.