

# System Requirements Specification Document – Version 1.0



Project: **Counting Simulator** -

Author: **Ivan Belov** @ [ivan.belov@gmx.com](mailto:ivan.belov@gmx.com)

Date: **2019**

- ❖ Project Description
- ❖ Domain Dictionary
- ❖ Game Rules
- ❖ Functional Requirements
- ❖ Non – Functional Requirements
- ❖ Hardware Requirements
- ❖ Software Requirements
- ❖ Release Plan
- ❖ Interface design
  - Sample G.U.I. #1
  - Sample G.U.I. #2
  - Sample G.U.I. Controls #1
- ❖ Application's Candidate Classes and Relationships
  - Class Interactions.
  - Card Class
    - Card's Design
  - Deck Class
  - Side Deck Class
  - Hand Class
  - Player Class
  - Game Class
  - Match Class
  - Lane Class
  - Game Engine Class
  - Statistics
  - Achievements
  - Settings
- ❖ Software Architecture
- ❖ Design Constraints
- ❖ Main Activity
- ❖ Create New Profile Use Case and Sequence Diagrams.
- ❖ Select Profile Use Case and Sequence Diagrams.
- ❖ Delete Profile Use Case and Sequence Diagrams.
- ❖ Play Single Player Game Use Case and Sequence Diagrams.
- ❖ Play Cooperative Game L.A.N. Use Case and Sequence Diagrams.
- ❖ Play Cooperative Game W.A.N. Use Case and Sequence Diagrams.
- ❖ Record Game Score Use Case and Sequence Diagrams.
- ❖ See Game Scores Use Case and Sequence Diagrams.

## **Project Description**

The Developer's intention for this project to be a stand-alone application distributed via public software distribution services such as Steam, Chrono.gg or any other similar systems.

Counting Simulator is a computer Card Game that simulates a set of Card Game Matches between two Opponents to determine the Winner of a Game. An Opponent can be either a pre-programmed Computer Logic or another Player joining via Local Area Network (L.A.N.) or Wide Area Network (W.A.N.) connections.

This document has Bell MT font of size 14 throughout. Title names and table of contents items are bolded for visibility. Wingdings 0x0076 type of a bullet used for listing details in the document.

The purpose of this program is to provide lasting entertainment to its users, coding experience and monetary gain to the creator.

Assumed factors such as the presence of the market or monetary support for the project could affect its requirements, development or release.

### Domain Dictionary

<b>Term:</b>	<b>Type:</b>	<b>Description:</b>
Card	Entity	A card is the smallest entity in this program.
Unique Card	Entity	A Card that is of any of three Suits but of higher grade.
Basic Deck	Entity	A Basic Deck is a set of eighteen Cards.
Side Deck	Entity	A Side Deck is a set of ten Cards randomly selected from a user's Basic Deck.
Hand	Entity	A Hand is a set of Four Cards randomly selected from a user's Side Deck.
Shuffled Reject Pile (S.R.P.)	Entity	Shuffle Reject Pile is a Basic Deck minus the Cards from the Basic Deck picked for a Side Deck and subsequently for a Hand. The S.R.P. is used to provide a Buffer Card during a Match. Both Players have their own S.R.P.s.
Buffer Card	Entity	A Card that facilitates extra point delivery towards the twenty-one goal during each Match and for both Playeres.
Game	Entity	A Game consists of three Matches, involves two Players and Credit Pot that is awarded to a winning player upon the Game completion.
Match	Entity	A Match is a set of a single Turn(s) for each Player (Opponent / Player) involved in a Game.
Bonus Round	Entity	A Round to determine the Winner of the Game if three Match draws is detected.
Lane	Entity	A Lane holds nine Cards' Spots. Each Player has a Lane. A Lane is filled with Cards from S.R.P. or a Player's Hand. A sum of Cards' values on each Lane determines that Lane's Score.
Card's Spot	Entity	There are nine Card's Spots on a Player's Lane.
Turn	Event	Turn is the gameplay event. There can be any amount of Turns during a Game. During a Turn, an opponent makes a Move.
Move	Event	A Move is an action taken by a player during a game with another opponent. A user can either Deal, Hit or Stand.
Player/ Profile /Opponent	Entity / Menu	An entity that brings together a Deck and a Game Score.
Computer Logic	Entity	It is a set of pre-defined rules that a computer program shall follow while simulating a Game with a Player.

<b>Term:</b>	<b>Type:</b>	<b>Description:</b>
Winner	Entity	The Winner is a Player who wins the most Matches during a Game.
Looser	Entity	A Looser is a Player who wins the least Matches during a Game.
Transaction History	Entity	Transaction History is a stored set of Player's in-game currency purchases.
User Manual	Document	User Manual is a set of descriptive articles explaining the use of Counting Simulator software in detail.
CS_DB	Database	A local or remote database that stores Player's data.
Achievement	Entity	Achievement is single entity representing a Game Goal completed by a Player.
Game Goal	Entity	There shall be several Game Goals that have different probability and aim to achieve.
Settings	Menu / Control	A set of menu options available for a Player.
Dealer	Entity	A Dealer is a logic that distributes a card to each Player during the start of their every Turn.
Registered Profile Owner (R.P.O.)	Entity	R.P.O. is the one who registered a Game Profile with a unique login and a robust and secure password.
Licensed Software Owner (L.S.O.)	Entity	L.S.O. is a person who has purchased a software copy verified by a unique ID combination.
Credit(s)	Entity	Credit is a unit of the in-game currency earned during gameplay or purchase via real-money transactions.
Credit Pot	Entity	Credit Pot is a cumulative amount of Credits invested by both Players in a Game.
Game Score / Record	Entity	Game Score is a set of Betting Pot values and Matches' outcomes. A Player's Game Score is derived from three positive Match Scores. For a Player to win a Game, at least two out of three Matches shall be won. In case of a draw, Bonus Round is introduced.
Match's Score	Entity	The sum of cards on a Lane. A Player needs to come as close to twenty-one points as possible. If a Player goes over twenty-one points and Stands. That Player loses that Match. The only available Score values are twenty-one, twenty, nineteen and eighteen. The higher value trumps the lower one. The Score is updated every time a new Card is placed on a Lane.
Stand	Control	This action gives your Opponent a right to take a turn.

Hit	Control	This action is requesting the top Card from a Player's S.R.P.
Suit	Entity	There are three suits in this game. Positive: +; Negative: -; Combo: +/-.
Card's Reverse Image (C.R.I.)	Entity	Ability to change the default Reverse Image to a custom one based on the in-game progress of the Player.
In-Game Progress / Statistics	Entity / Menu	Statistics is derived from the probability of various events during the game play.

## Game Rules and Process

There shall be two Opponents during a Game.

Both Opponents to have one single basic game deck of eighteen cards. There are three Suits: Positive – represented with the sign of plus - '+,' Negative – the symbol of minus - '-', Combo – positive and negative signs together - '+/-' A Player has a choice of selecting an appropriate sign during the gameplay for any combo Card. The cards in a Basic Deck of all Suits range from one to six.

Unique Cards could be added to the Deck by purchasing or winning them throughout the gameplay — additional cards to extend decks with ranging values from seven to ten.

Unique Cards shall represent the same Suits and a different range of values. Any card could be present in the replicated state. Two copies of any Unique Card could occur in one Deck, Side Deck, and a Hand.

Each Opponent shall have a Hand of Cards.

A Hand consists of four randomly derived Cards from a Side Deck. A Deck consists of 18 Cards of the Basic Deck that comes with every Profile and several Unique Cards purchased by the same Player.

Each Opponent gets ten Deck's Cards to form a Side Deck before deriving a Hand for a Player and a distinct Game. Side Deck selection could be performed manually or randomly – automatic. After a Side Deck selection, a random four Cards are selected as a Hand for each Opponent.

Each Game consists of three Matches. During each Match, a Player needs to collect as close to twenty-one points as possible to win.

A Match is divided into Turns.

The first Opponent to take the first Turn is decided via a random draw.

During a Turn, the top Card is drawn from the S.R.P., which belongs to the Turn's owner, and is placed on one of the nine Card Spots available for each Lane. The face value of the Card is either added to or subtracted from the Match's Score. After S.R.P.'s Card is placed on a Lane, A Player has a choice of actions.

Choice A: Select any number of Cards from a Player's Deck to be placed on a Lane. There are only four Cards available for that. After the selection, a Player must select Stand to let the Opponent proceed with her or his Turn.

Choice B: Select Stand without selecting any Card from a Hand.

If you are not the first one to start a Match, then you choose either **Hitting** for a top Card out of S.R.P., **Standing** and giving an Opponent a Turn, or to select a Card from his or her Hand.

The Game goes as many rounds as needed to determine the Winner. A Player who wins the most Rounds out of three is a Winner. Bonus Round is introduced if a tie Score is detected.

When a Winner is determined, the combined Pot with Bets from both Players is awarded to a Winner.

The Record of outcomes for each Match of a Game is recorded in Player's Statistics.

If any Achievement is completed, it is awarded to a Player. A screen notice and an available message from a software distribution system might be implemented.



## Functional Requirements

<b>Code:</b>	<b>Description:</b>
<b>FR-1</b>	This is a windowed application, implements full screen capabilities.
<b>FR-2</b>	Display the User's Hand (Four Cards).
<b>FR-3</b>	Display the Opponent's Hand C.R.I. up (Four Cards).
<b>FR-4</b>	Display the change of Cards at each Turn.
<b>FR-5</b>	The application shall display the changing Score of the current Match for both Opponents and an overall Game.
<b>FR-6</b>	The Basic Deck shall be allowed to run out, yet it never should since only nice places are assigned for each Player, with at most eighteen Cards.
<b>FR-7</b>	A User shall be able to see her / his Game Score history or Statistics.
<b>FR-8</b>	This application let users have random Side Decks.
<b>FR-9</b>	The Side Deck must have ten Cards selected randomly from the Basic Deck.
	The Hand must have four Cards selected randomly from the Side Deck.
<b>FR-10</b>	A Credit Pot is to be awarded to the Winner when one is determined.
<b>FR-11</b>	A Game Score / Record shall be recorded after the end of any Match or a Game.
<b>FR-12</b>	This application shall support cross-PC functionality between all stationary deployments.
<b>FR-13</b>	Financial in-game transactions shall be performed securely and seamlessly.
<b>FR-14</b>	A Player shall be able to purchase an in-game Credits or DLC items via the software distribution service.
<b>FR-15</b>	Statistics are selected by the Developer and are permanent. No Player can add or filter out a data line from the Statistics tab.
<b>FR-16</b>	Face of any Card on the Lane, and a C.R.I. of each Opponents' Deck can be magnified for view during the gameplay.

### Non – Functional Requirements

<b>Code:</b>	<b>Description:</b>
<b>NFR-1</b>	This application shall have a unified graphical interface.
<b>NFR-2</b>	User Manual shall be available as a digital copy.
<b>NFR-3</b>	User account passwords shall be a minimum 8 characters in length and must contain a lowercase letter, an uppercase letter, a number, and a symbol.
<b>NFR-4</b>	Users should be able create their own deck suits.
<b>NFR-5</b>	The payment system shall be available 24 hours a day, 7 days a week, and shall have a 99.5% uptime.
<b>NFR-6</b>	The startup time for the application shall not exceed 5 seconds.
<b>NFR-7</b>	Monetary amounts shall be accurate to 2 decimal places, rounding where necessary.
<b>NFR-8</b>	user shall be able to top up their Credit amount with any currency at any time.
<b>NFR-9</b>	The application shall be simple and intuitive such that a new user with no training shall be able to use the service immediately.
<b>NFR-10</b>	The system shall be scalable to support unlimited growth in the number of concurrent users.
<b>NFR-11</b>	A user shall be able to select a custom design of the back of player's deck.
<b>NFR-12</b>	A user shall accept terms of services during the installation of the Counting Simulator.
<b>NFR-13</b>	A betting amount provided by both Players could be of various amounts.
<b>NFR-14</b>	Both Players can select a custom C.R.I. for their Decks upon the Game progress.
<b>NFR-15</b>	Transaction History shall be stored alongside the Player's Statistics data.
<b>NFR-16</b>	The Graphical User Interface (G.U.I.) of the application has Orator Std Bold 18 font.
<b>NFR-17</b>	Other menus are of the same font and blown up to 36 points.

## Hardware Requirements

- ❖ Processor: 1.0 GHz
- ❖ Memory: 512 MB
- ❖ Storage: 300 MB
- ❖ Sound Card: DirectX compatible

## **Software Requirements**

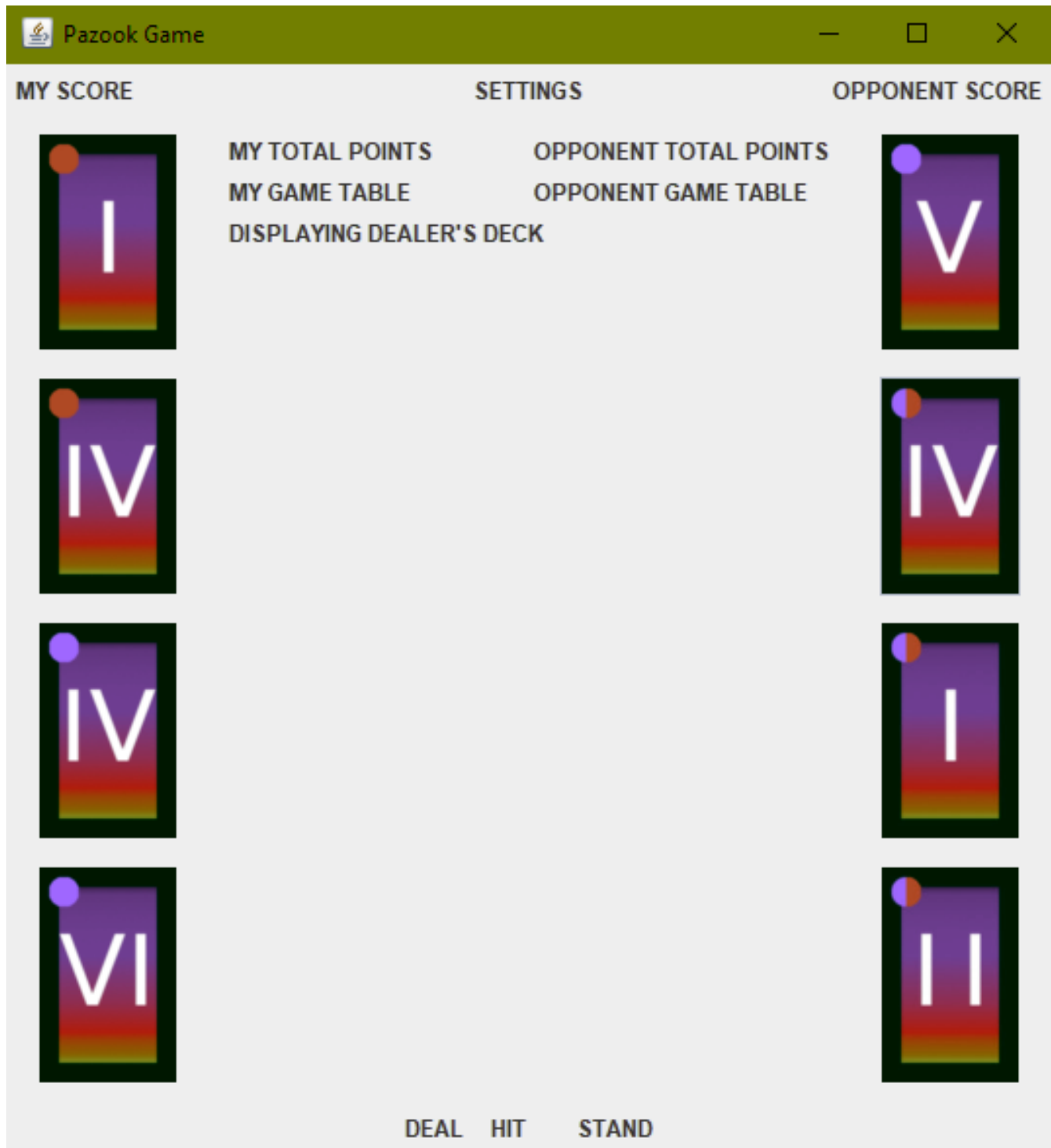
- ❖ Operating System: Windows 7 or higher

**Release Plan**

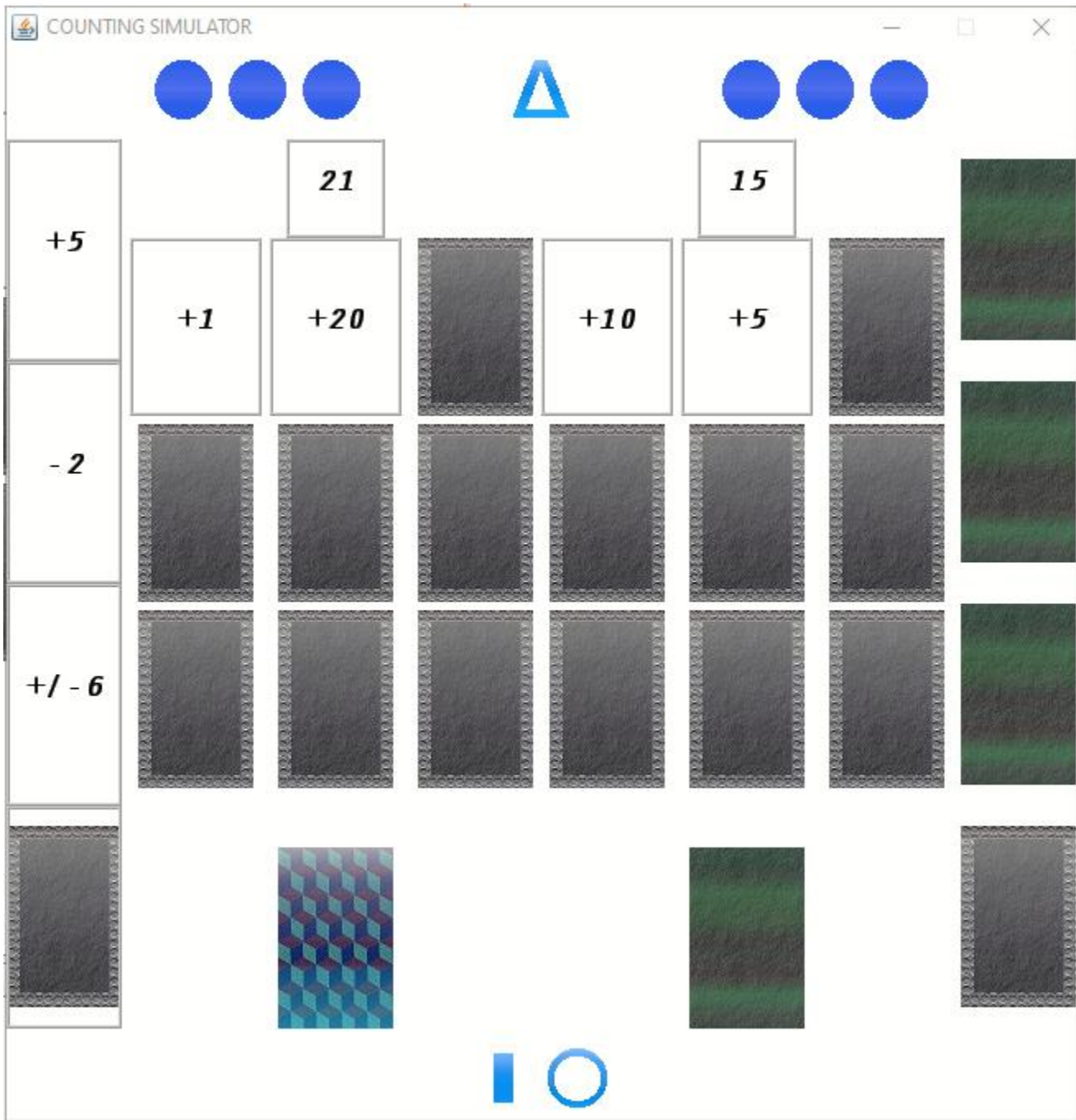
<b>Version:</b>	<b>Description:</b>	<b>Release Date:</b>
CS_SRS_V01.pdf	First release of System Requirements Specification Document.	October 2020
CS_v01.exe	Constructed classes, implemented logic, first revision of System Requirements Specification Document.	December 2020
CS_SRS_V02.pdf	Second release of System Requirements Specification Document based on previous revision.	April 2021

## Interface Design













### ❖ Sample G.U.I. #1



❖ Sample G.U.I. #2



## ❖ Sample G.U.I. Controls #1

Control Name and Extension	Size	Visual Representation
Settings.gif	36 x 36	
Hit.gif	36 x 36	
Stand.gif	32 x 32	
CountingSimulatorIconS.ico	16 x 16	
CountingSimulatorIconM.ico	24 x 24	
CountingSimulatorIconL.ico	32 x 32	
CountingSimulatorIconXL.ico	48 x 48	
UnfinishedRound.gif	36 x 36	
WonRound.gif	36 x 36	
LostRound.gif	36 x 36	
CardSpot.gif	70 x 110	
CardSpotL.gif	140 x 220	



## Application's Classes

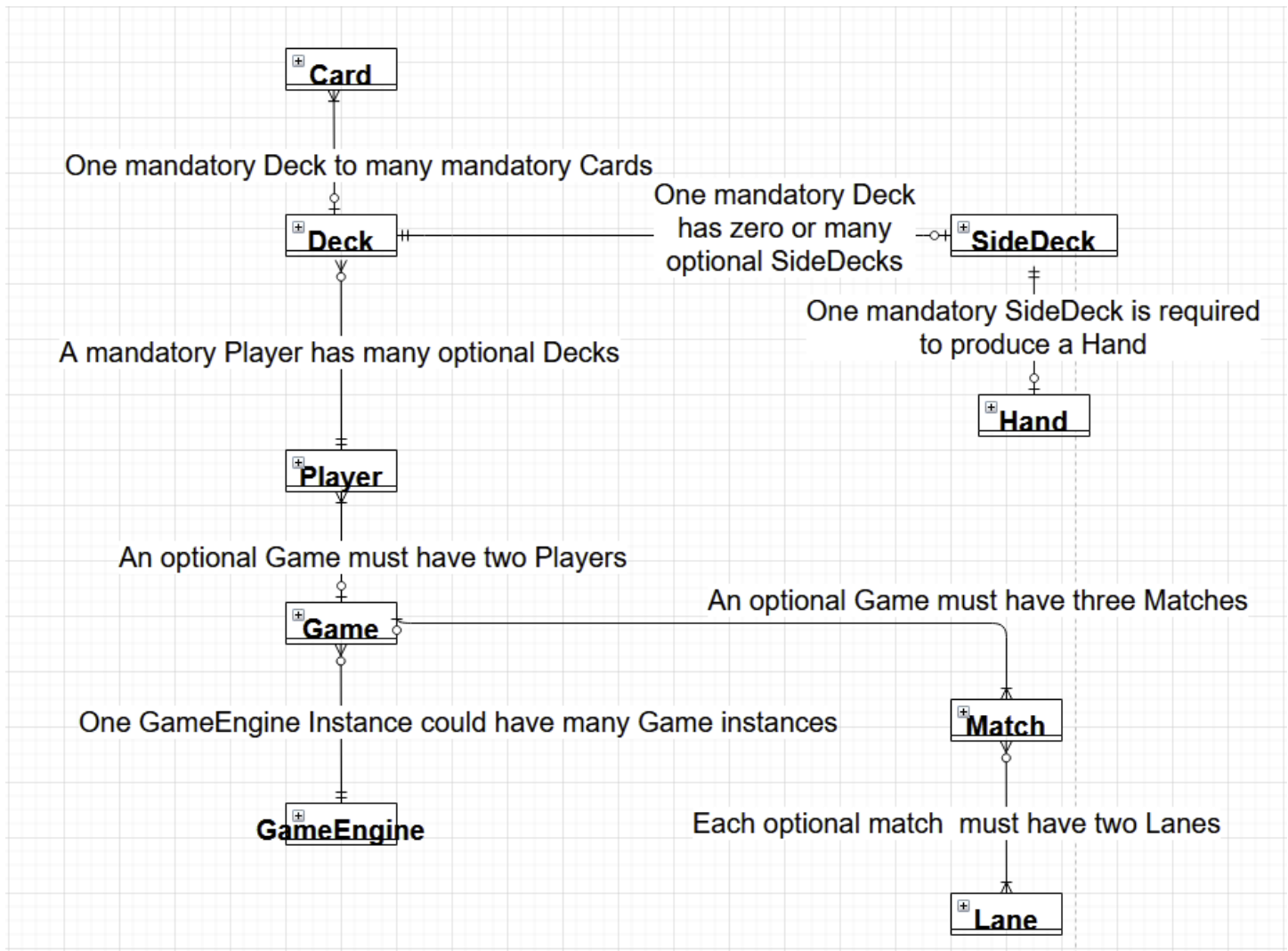
### Candidate Classes:

- ❖ Card
- ❖ Deck
- ❖ Side Deck
- ❖ Hand
- ❖ Player
- ❖ Game
- ❖ Match
- ❖ Lane
- ❖ Game Engine
- ❖ Statistics
- ❖ Achievements
- ❖ Settings

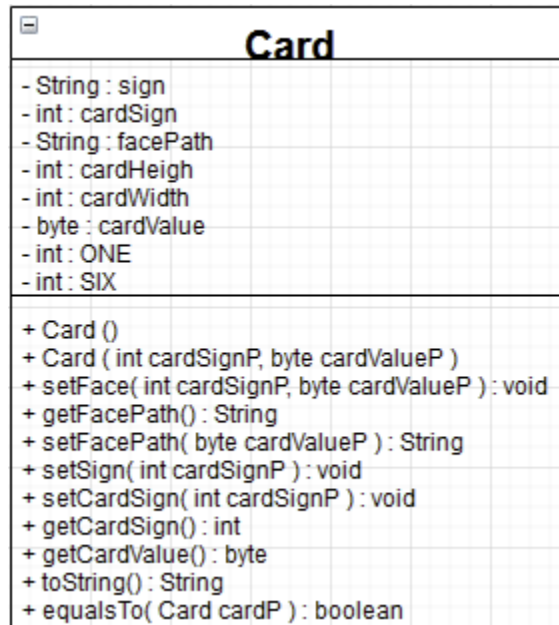
### List of Relationships:

- ❖ A Deck has eighteen or more Cards.
- ❖ A Player could have many Decks.
- ❖ A Side Deck consists of ten Cards selected randomly from a Deck.
- ❖ A Hand consists of four Cards selected randomly from a Side Deck.
- ❖ A Game consists of three Matches and must include two Players.
- ❖ A Match has two distinct Players.
- ❖ A Match much have two Lanes.

## Class Interactions

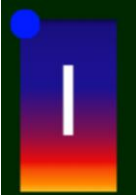


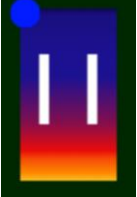
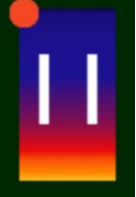
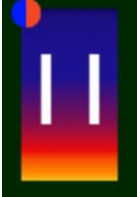

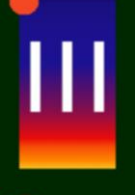















## Card Class



A Card is a class representing a card entity. A card is one of the **three** in-game **suits**: positive, negative, and combo. A card can be of one of **six ranks**. These ranks include the roman numerals from one through six. Wild cards might be implemented to be purchased with in-game currency. Each card has a face and back. A card's face describes a suit and a rank.

Cards' Design

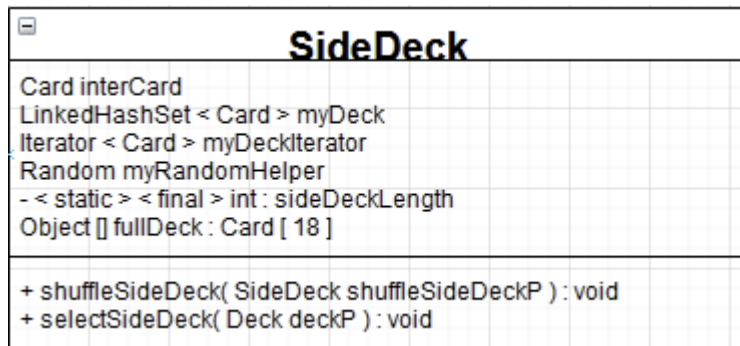
	Negative Suit	Positive Suit	Combo Suit
Rank One			
Rank Two			
Rank Three			
Rank Four			
Rank Five			
Rank Six			
Default Back Design			

## Deck Class

Deck	
Card interCard	
LinkedHashSet < Card > myDeck	
Iterator < Card > myDeckIterator	
Random myRandomHelper	
- < static > < final > int : deckLength	
- boolean : tripValue	
+ shuffleDeck( Deck shuffleDeckP ) : void	
+ createFullDeck() : void	

A Deck is a representation of a deck of playing cards. There are eighteen cards in each standard deck. A standard deck of cards is required for a player to play a game. The back of the deck of cards can have custom graphics set by a user. Ultimately a gaming deck could consist of a standard deck and a deck of random wild cards. A user could have multiple card decks.

## Side Deck Class



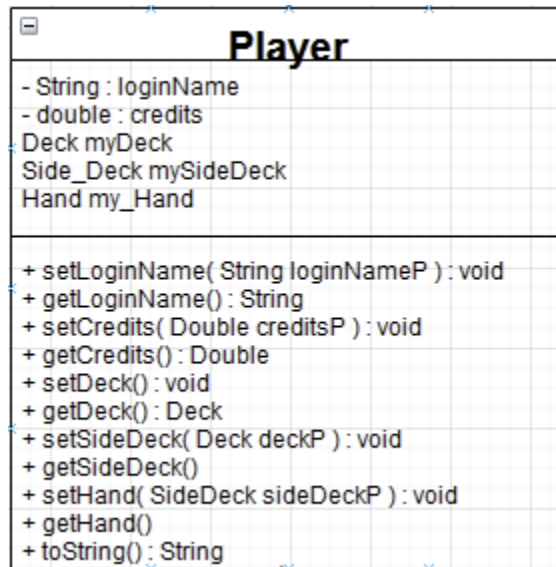
A Side Deck is a representation of a randomized, shuffled and shortened deck of cards. Ten cards shall be selected by a user or randomly by computer logic. Side Deck has ten cards.

## Hand Class

Hand	
Card interCard	
LinkedHashSet < Card > myHand	
Random myRandomHelper	
- < static > < final > int : handLength	
Object [] sideDeck : Card [ 10 ]	
myCardFace : Image	
+ selectHand( SideDeck sideDeckP ) : void	

A Hand class represents a set of cards. There are four cards in hand. Those cards are selected randomly for a side deck before each match.

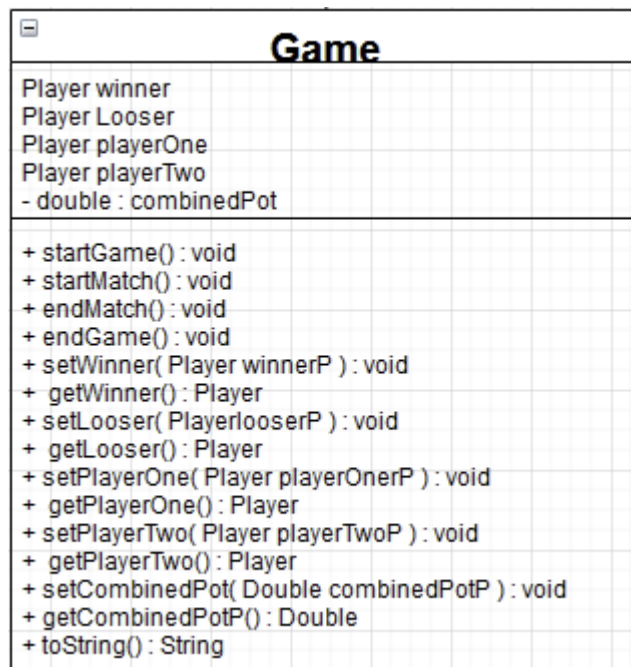
## Player Class



Player class hold variables and features that describe a game user in detail. The player is also an account holder for a network game. A player has login credentials, credits that he or she earned up to date, a deck of cards.

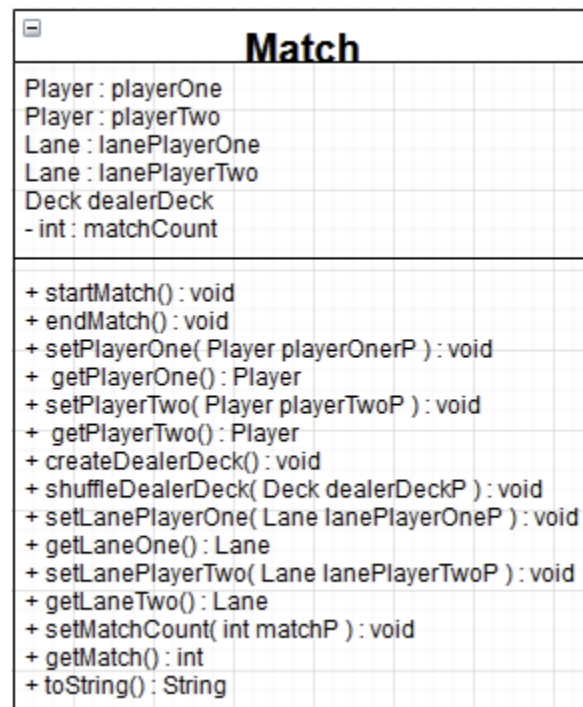


## Game Class



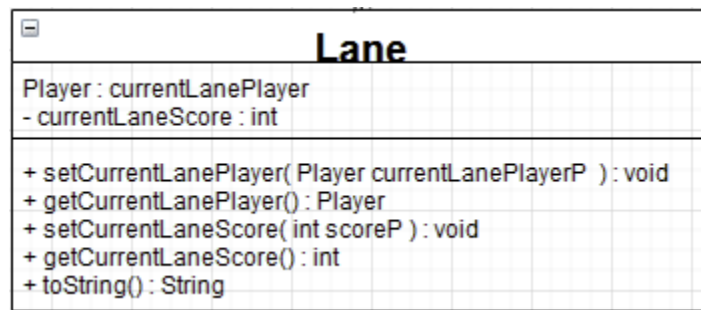
A Game is an instance of a Pazook game. It brings two players with two decks together for few matches to determine a winning party. A game shall have a pot of credits that players are betting their win.

## Match Class



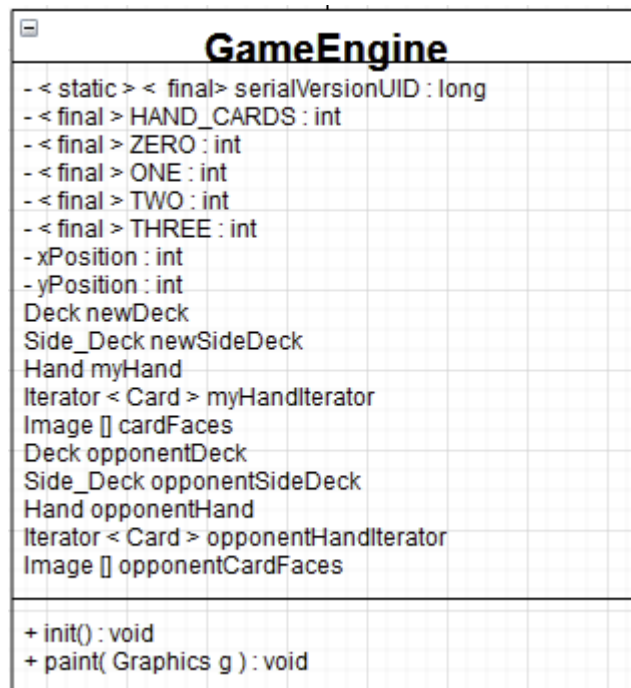
A Match is the inner work of a game. There are three matches in each game. The winner of the most matches is the winner of that game. Overall bet pot is awarded to a winning player. The statistics of each match and a game is recorded locally or remotely.

## Lane Class



A Lane is a representation of a set of nine cards. This set consists of cards being dealt by a dealer, in addition to cards selected from a player's hand.

## Game Engine Class



A Game Engine class provides Graphical User Interface and the logic behind the application flow.

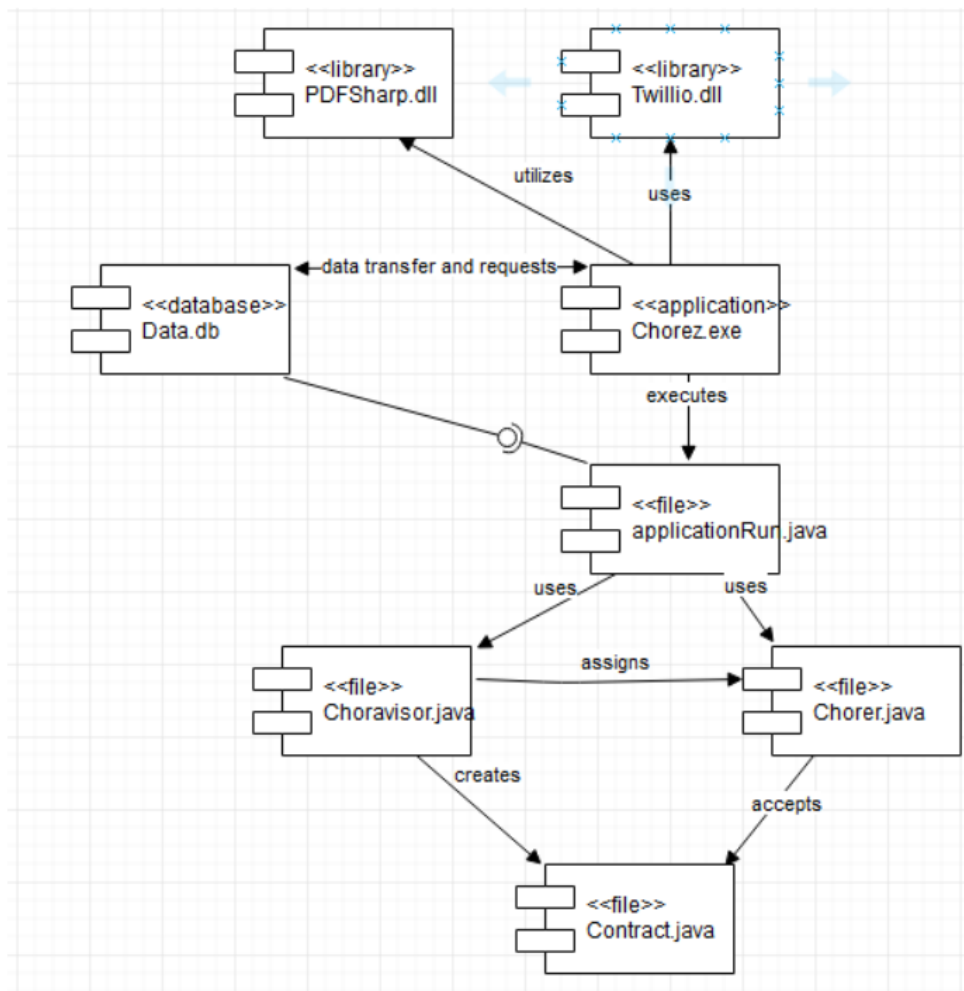
## Software Architecture

Pazook is developed using Service-Oriented architecture with the implementation of the top-down approach of development.

The software is a three-tier business application to be accessed from any operating system running Java Virtual Machine. Locally and remotely stored databases are used for data storage. Microsoft Azure Servers can host the application's logical layer.

Component diagram below describes modules of the software.

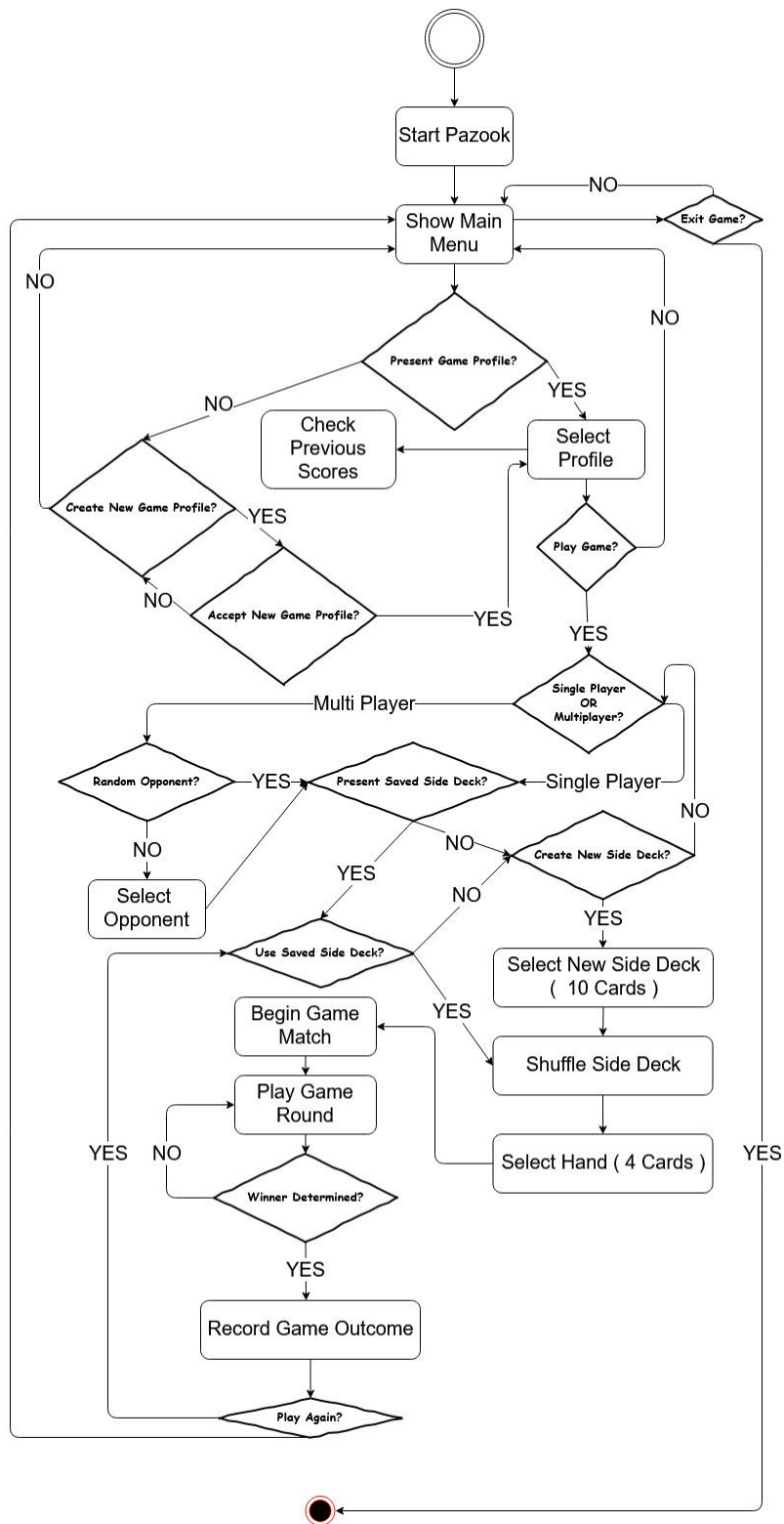
Component diagram:



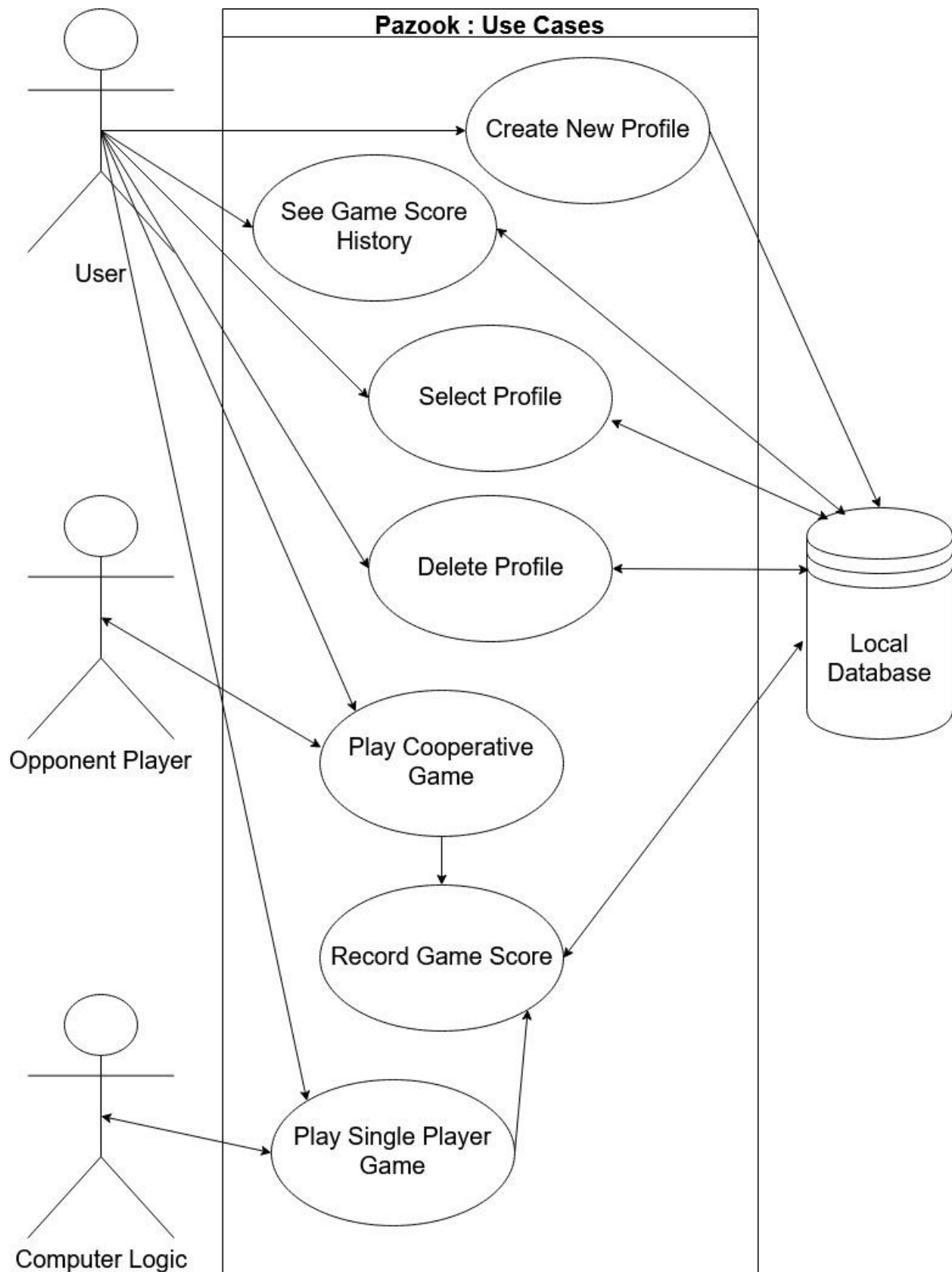
### Design Constraint

Constraint:	Type:	Description:
Commercial	Self - Imposed	Time and budget
Legal Compliance	Imposed	Applicable laws, regulations, standard
Design	Self - Imposed	Identical Graphical User Interface features, predetermined syntax format and design pattern.
Localization	Self – Imposed	Providing an ability to switch or change User Interface or modules based on cultural or geographical differences
Integration	Self - Imposed	Usage of various third-party modules, systems, and products with Pazook conjunction.
Security	Self - Imposed	Production of streamlined and effective database queries to reduce resource usage and prevent data leaks. Database duplication for redundant local and hosted data storage. Encryption to mask any remote or internal communication.
Ease of Access	Self - Imposed	Screen optimization, clean, consistent and simple User Interface.

## Main Activity



## Use Case Diagram



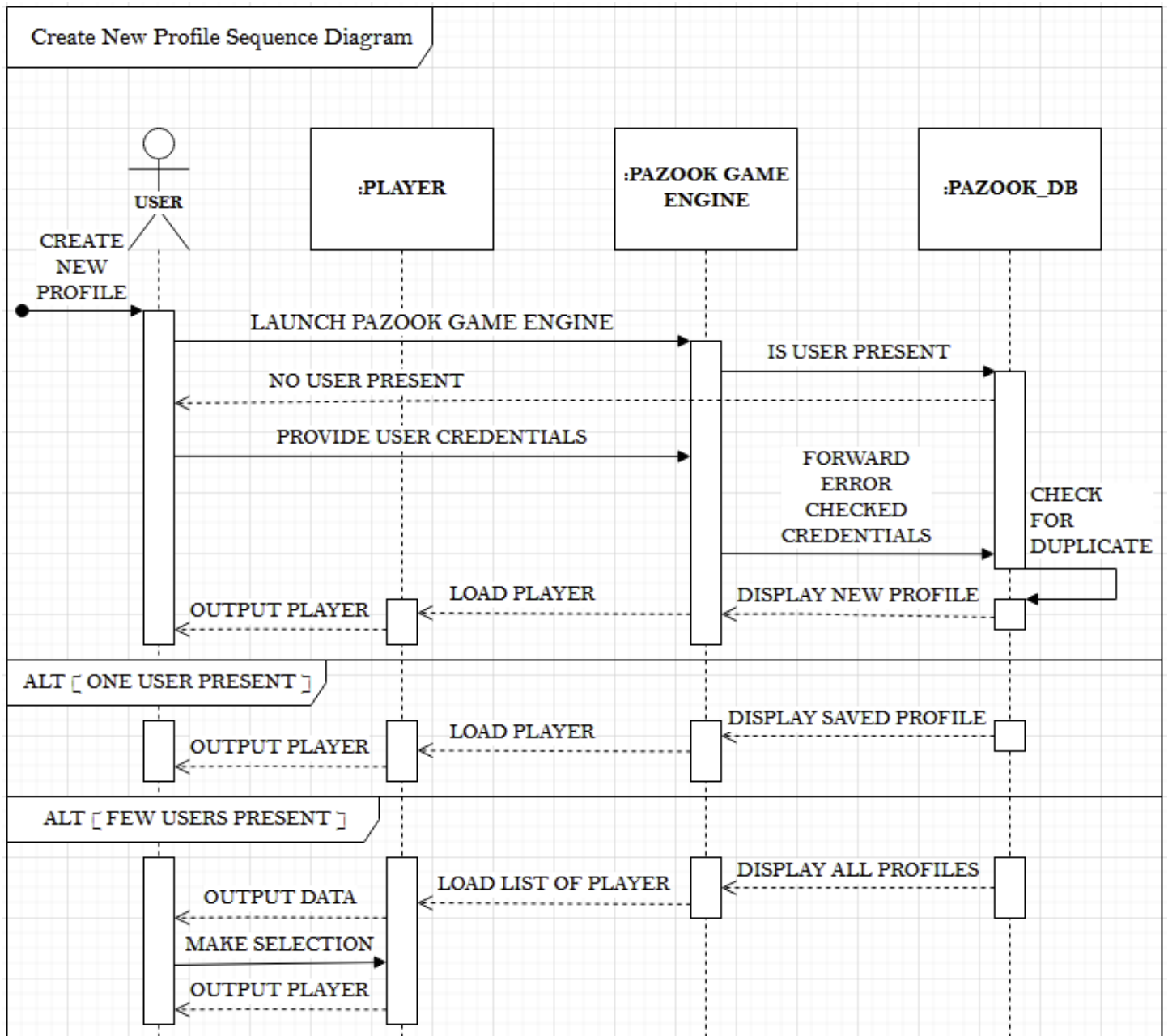




### Create New Profile Use Case

Name:	Create New Profile.
Description:	Any person who paid for and downloaded Pazook application shall be able to create a game account. The account is required for cross-device game data transfer.
Primary Actors:	Purchased Software Copy Owner.
Secondary Actors:	None.
Preconditions:	Purchased, Downloaded and Installed Licensed Copy of Pazook.
Main Flow:	<ol style="list-style-type: none"> <li>1. Start Pazook Execution File.</li> <li>2. Accept Usage Rights on first launch of the application.</li> <li>3. Application Scans Local Storage for a Game Profile.</li> <li>4. If Profile is Detected, it is Loaded.</li> <li>5. If Multiple Profiles are detected, they are loaded onto a screen for a user to make selection.</li> <li>6. If No Profile Detected, Profile Creation Page is Opened.</li> <li>7. Upon completion of the Profile Creation Page, Main Page loaded.</li> </ol>
Postconditions:	<p>Creating an account to record game scores and keep game associated data for a given user in one place.</p> <p>Selected and Booted Up Account to a current game session.</p>

## Create New Profile Sequence Diagram



### Select Profile Use Case

Name:	Select Profile.
Description:	The application automatically checks local or remote data storage for any present Game Profile associated with login credentials. When identified, the profile would be loaded. If several locally or remotely stored profiles that are associated with one account are present, the choice shall be given in the form of a pop-up window or other fun and interactive way.
Primary Actors:	Registered Profile Owner, who is a Licensed Software Owner.
Secondary Actors:	None.
Preconditions:	Purchased, Downloaded and Installed Licensed Copy of Pazook. Registered Profile Owner with a Licensed Software Copy is present.
Main Flow:	<ol style="list-style-type: none"> <li>1. Accept user login credentials</li> <li>2. Check any account presence against provided login credentials on a local or remote server containing a database.</li> <li>3. If a single Profile is detected, Load the Profile.</li> <li>4. If multiple Profiles are detected, load a selection screen.</li> <li>5. After the selection has been made, output relevant Player data on the screen.</li> <li>6. Continue to the Main Menu.</li> </ol>
Postconditions:	Selected and Booted Up Account to a current game session.

## Select Profile Sequence Diagram

### Delete Profile Use Case

Name:	Delete Profile.
Description:	Delete Profile Use Case describes steps performed during the deletion of a selected Profile.
Primary Actors:	Registered Profile Owner, who is a Licensed Software Owner.
Secondary Actors:	None.
Preconditions:	Purchased, Downloaded and Installed Licensed Copy of Pazook. Registered Profile with a Licensed Software Copy is present.
Main Flow:	<ol style="list-style-type: none"> <li>1. Check if more than one Profile belongs to the R.P.O.</li> <li>2. Provide a User with a list of available Profiles</li> <li>3. Erase all associated data with a selected Profile from local or remotely stored database(s).</li> </ol>
Postconditions:	Selected Profile is erased from the data storage. Previous Game History, amount of Credits earned, Login Credentials associated with the Selected Profile are erased.

## Delete Profile Sequence Diagram

### Play Single Player Game Use Case

Name:	Play Single Player Game.
Description:	Play Single Player Game Use Case describes steps needed to simulate a Card Game between a Player and a Computer Logic.
Primary Actors:	Registered Profile Owner, who is a Licensed Software Owner.
Secondary Actors:	Computer Logic.
Preconditions:	Purchased, Downloaded and Installed Licensed Copy of Pazook. Registered Profile with a Licensed Software Copy is present.
Main Flow:	<ol style="list-style-type: none"> <li>1. Select Single Player Game from the Main Menu.</li> <li>2. Bet Credits.</li> <li>3. Select Random or Custom Side Deck.</li> <li>4. The Game logic determines the first Player to take a Turn.</li> <li>5. Each Player in a repeating sequence takes Game Turns.</li> <li>6. If the Winner is determined, Credit Pot is awarded to that Player.</li> <li>7. Game Record is saved.</li> </ol>
Postconditions:	A Finished Game with Game Record data saved, Winner and Looser identified. Credit Pot is awarded to the Winner.



## Play Single Player Game Sequence Diagram

### Play Cooperative Game L.A.N. Use Case

Name:	Play Cooperative L.A.N. Game
Description:	This Use Case describes steps needed to simulate a Card Game between two Players located on the same Local Area Network.
Primary Actors:	Registered Profile Owner, who is a Licensed Software Owner.
Secondary Actors:	Another Player.
Preconditions:	Purchased, Downloaded and Installed Licensed Copy of Pazook. Registered Profile with a Licensed Software Copy is present. Both Players are present on the same Local Area Network.
Main Flow:	<ol style="list-style-type: none"> <li>1. Select Cooperative Game from the Main Menu.</li> <li>2. Select an Opponent to play a Game.</li> <li>3. Bet Credits.</li> <li>4. Select Random or Custom Side Deck.</li> <li>5. The Game logic determines the first Player to take a Turn.</li> <li>6. Each Player in a repeating sequence takes Game Turns.</li> <li>7. If the Winner is determined, Credit Pot is awarded to that Player.</li> <li>8. Game Record is saved.</li> </ol>
Postconditions:	A Finished Game with Game Record data saved, Winner and Looser identified. Credit Pot is awarded to the Winner.

## **Play Cooperative Game L.A.N. Sequence Diagram**

### Play Cooperative Game W.A.N. Use Case

Name:	Play Cooperative W.A.N. Game
Description:	This Use Case describes steps needed to simulate a Card Game between two on-line Players.
Primary Actors:	Registered Profile Owner, who is a Licensed Software Owner.
Secondary Actors:	Another Player.
Preconditions:	Purchased, Downloaded and Installed Licensed Copy of Pazook. Registered Profile with a Licensed Software Copy is present. Both Players are On-Line.
Main Flow:	<ol style="list-style-type: none"> <li>1. Select Cooperative Game from the Main Menu.</li> <li>2. Select an Opponent to play a Game.</li> <li>3. Bet Credits.</li> <li>4. Select Random or Custom Side Deck.</li> <li>5. The Game logic determines the first Player to take a Turn.</li> <li>6. Each Player in a repeating sequence takes Game Turns.</li> <li>7. If the Winner is determined, Credit Pot is awarded to that Player.</li> <li>8. Game Record is saved.</li> </ol>
Postconditions:	A Finished Game with Game Record data saved, Winner and Looser identified. Credit Pot is awarded to the Winner. Display Main Menu.

## **Play Cooperative Game W.A.N. Sequence Diagram**

### Record Game Score Use Case

Name:	Record Game Score
Description:	Record Game Score Use Case describes steps taken to record the outcome of any Game.
Primary Actors:	A Player.
Secondary Actors:	An Opponent.
Preconditions:	Completed Game with Winner present.
Main Flow:	<ol style="list-style-type: none"> <li>1. Assign a Player with the most Mathes won as a Winner.</li> <li>2. Assign a Player with the least</li> <li>3. Matches won as a Looser.</li> <li>4. Record the Winner and the Looser.</li> <li>5. Record the size of the Credit Pot.</li> <li>6. Send recorded data to the Pazook_DB.</li> <li>7. Recorded data is associated with both Players of a Game.</li> <li>8. Display Main Menu.</li> </ol>
Postconditions:	Display Main Menu.

## Record Game Score Sequence Diagram

### See Game Scores Use Case

Name:	See Game Score Player Game
Description:	This Use Case describes steps taken to show a history of Games linked to a Player.
Primary Actors:	A Player.
Secondary Actors:	None.
Preconditions:	Purchased, Downloaded and Installed Licensed Copy of Pazook. Registered Profile with a Licensed Software Copy is present.
Main Flow:	<ol style="list-style-type: none"> <li>1. Check Player's Game Score records for any data present.</li> <li>2. If no data is present, output an informative message describing such an event.</li> <li>3. If data is present, output it in a separate window.</li> </ol>
Postconditions:	Display Game Score record of a Player. Display Main Menu with a notification message if no Game Score data is present.



## See Game Scores Sequence Diagram