Project: **Pazoock – System Requirements Specification**

Author: **Ivan Belov @ ivan.belov@gmx.com**

Date: **2019**

Domain Dictionary

Term:	Type:	Description:
Card	Entity	A card is the smallest entity in the Pazook Card Game.
Deck	Entity	A Deck is a set of eighteen Cards.
Side Deck	Entity	A Side Deck is a set of ten Cards.
Hand	Entity	A Hand is a set of Four Cards.
Game	Entity	A Game consists of Matches, involves two Players, and Credit Pot that is awarded to a winning player.
Match	Entity	A Match is a set of a single Turn for each Player (Opponent / Player) involved in a Game.
Score	Entity	A Score of a Player is a combination of Cards' values that are present on a Lane associated with that Player. The score is updated every time a new Card is placed on a Lane.
Lane	Entity	A Lane holds Cards that are placed on it from a Dealer or by a User during each Turn. A set of Cards on each Lane determines the Score standings of both Players during a Game.
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	An entity that brings together a Deck, Score History, Amount of Credits won throughout previously played Games.
Computer Logic	Entity	It is a set of pre-defined rules that a computer program shall follow while simulating a Game with 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	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	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 an object that has a value of credits waged during a game, players who took part in it, the winner and the loser.

Term:	Type:	Description:
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 Pazook software in detail.

Project Description

Pazack is the Star Wars Universe card game.

Pazook is a personal representation of the intellectual entertainment product mentioned above.
System Requirements Specification Document describes Pazook_SRS_1 version.

Pazook is a computer game that helps simulate a set of Card Game Matches between two Opponents.

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 intended audience of this paper is a future development team, marketing staff, and investors. The suggested sequence for reading this document is to go through the table of contents in a sequential order.

The purpose of this program is to provide entertainment to its users.

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

Game Rules

There shall be two opponents in this game.

Each of the opponents to have one single basic game deck of eighteen cards. There are three groups of cards: Positive – represented with the sign of plus - '+,' Negative – the symbol of minus - '-,' Combined Group – positive and negative signs together - '+/-.' The cards in all three groups range from one to six.

Extra cards could be added to the deck by purchasing or winning them though out the gameplay — additional cards to extend decks with ranging values from seven to ten.

Each card shall have a sign, a value, a unique face representation and a custom deck suite. Any card could be present in the replicated state. Multiple copies of any card could occur in one deck, side deck, and a hand.

Each opponent has to select a side deck consisting of ten cards before each match. Side deck selection could be performed manually or randomly – automatic. After a side deck selection, a random four cards are selected as a hand of each opponent.

The game goes as many rounds as needed to determine the winner of three rounds. Each round, both opponents are trying to bring their own combined pot value as close to twenty-one as possible.

Each round consists of turns. Turns are taken by opponents one after another. Each turn, a single card is drawn from a third basic shuffled deck and placed in a player's pot. After an automatic card placement by a dealer, a player has a chance to perform an action. A player during his / her turn can select any number of cards from their hand to be added to own pot, can hold or remain as is, can withdraw from a game.

A player who wins the most rounds out of three is a winner; tie round or games are not taken into consideration, but still recorded.

Functional Requirements

Code:	**Description:**
FR-1	Full-Screen ability shall be implemented.
FR-2	Display the user's hand (Four Cards).
FR-3	Display the opponent's hand suite up (Four Cards).
FR-4	Display the change of hand's cards after each turn.
FR-5	The application shall display the changing score of the current hand for both opponents and an overall game.
FR-6	The general deck shall be allowed to run out.
FR-7	A user shall be able to see her / his previous game scores.
FR-8	This application shall let users choose between a random side deck and a custom one (Ten Cards).
FR-9	The Side Deck must have ten Cards selected randomly or manually.
FR-10	A Credit Pot is to be awarded to the Winner when one is determined.
FR-11	A Game Score / Record shall be recorded after the end of every Match and Game.
FR-12	This application shall support cross-platform functionality between all mobile and stationary deployments.
FR-13	Financial in-game transactions shall be performed securely.
FR-14	A Player shall be able to purchase in-game currency: Credits, at own leisure.
FR-15	Transaction History shall be stored alongside the Player's data.
FR-16	The application shall self-logout after a period of inactivity.
FR-17	User Manual shall be available as a digital copy.
FR-18	
FR-19	
FR-20	

Non – Functional Requirements

Code:	Description:
NFR-1	This application shall have a unified graphical interface.
NFR-2	Face of any card and a suite of each opponents' deck shall be displayed and available for view during the gameplay.
NFR-3	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.
NFR-4	Users should be able create their own deck suits.
NFR-5	The payment system shall be available 24 hours a day, 7 days a week, and shall have a 99.5% uptime.
NFR-6	The startup time for the application shall not exceed 5 seconds.
NFR-7	Monetary amounts shall be accurate to 2 decimal places, rounding where necessary.
NFR-8	user shall be able to top up their Credit amount with any currency at any time.
NFR-9	The application shall be simple and intuitive such that a new user with no training shall be able to use the service immediately.
NFR-10	The system shall be scalable to support unlimited growth in the number of concurrent users.
NFR-11	A user shall be able to select a custom design of the back of player's deck.
NFR-12	A user shall accept terms of services during the installation of the Pazook.
NFR-13	A betting amount provided by both Players shall be equal.

Hardware Requirements

- Dedicated desktop Windows machine with support of .Net - 4.0
- CPU 1.4 GHz 32-bit (x86)
- RAM 4 GB
- 20 GB local hard disc space
- Broadband Internet connection if remote data storage and messaging modules are utilized
- Input and pointer devices such as keyboard and mouse
- Output device such as VGA or HDMI supported monitor
- Portable device with Android 6.0 operating system

Software Requirements

Software needed to operate:	Description:
Windows OS – 10 Android – 6.0	Chorez – Biz is supported by Windows for products running on stationary machines, and Android for those who prefer the portable version of our line of products. The ability for cross-platform software communication is one of the leading development activities currently pursued by the development team.
Microsoft SQL Server 2019 My SQL – 5.0	Those who wish to store their data redundantly on off-site servers may decide to utilize Microsoft SQL Server implementation of their database. Those who look for a cheap and reliable solution for data storage might prefer hosting information on a local machine.

Release Plan

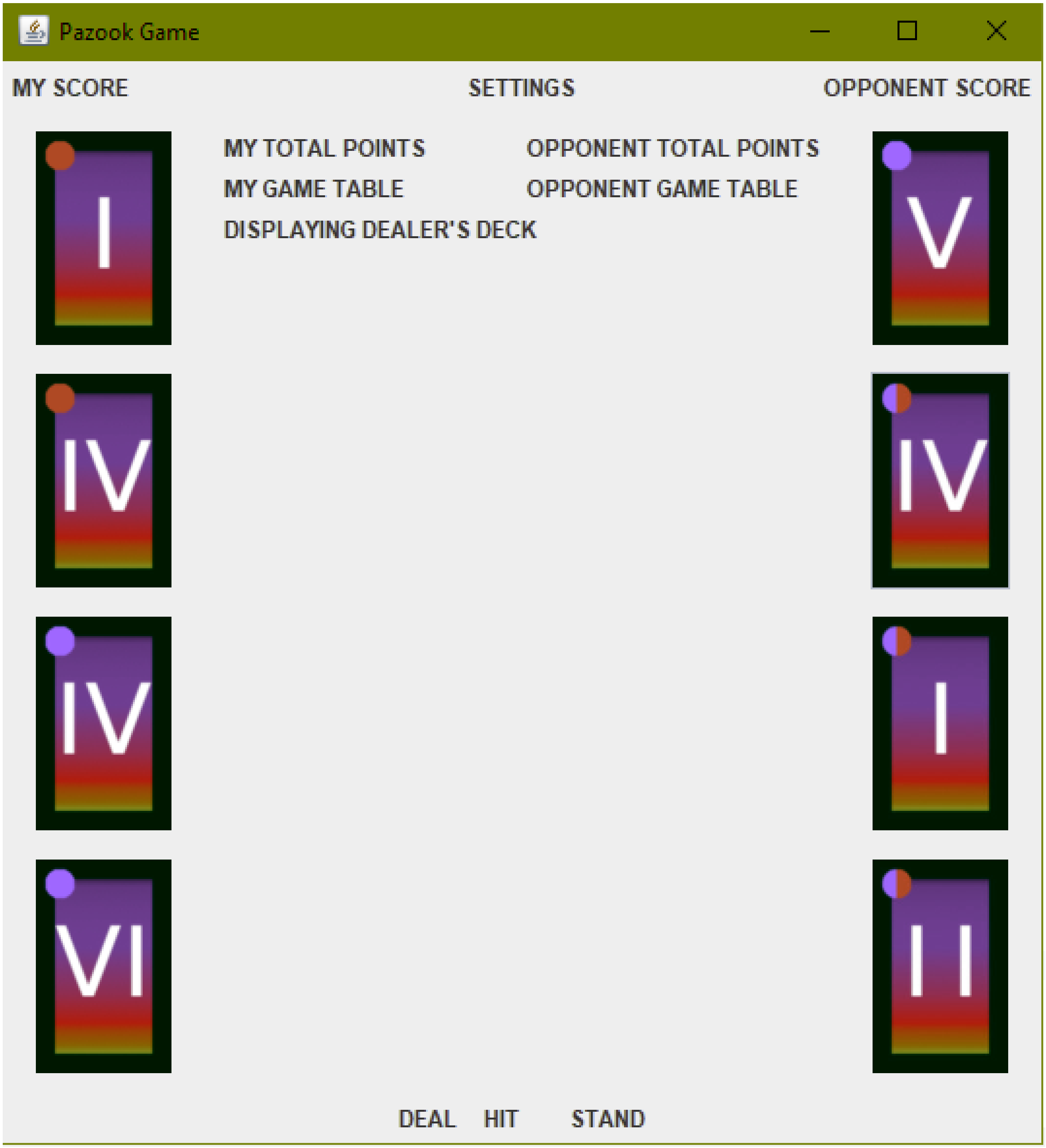
Version:	Description:	Release Date:
Pazook_SRS_1	First release of System Requirements Specification Document.	January 2020
Pazook_v01	Constructed classes, implemented logic, first revision of System Requirements Specification Document.	April 2020
Pazook_SRS_2	Second release of System Requirements Specification Document based on previous revision.	August 2020
Pazook_v10.exe	This version features an ability to play a game of Pazook with computer logic and save data locally and securely. Update of SRS Documentation.	November 2020
Pazook_v20.exe	This version features an ability to play a game of Pazook with another user over a secure connection on a local network. Update of SRS Documentation if required.	March 2021
Pazook_v30.exe	This version provides an ability to play a game of Pazook with another player over a secure connection on a remote game server.	June 2021

Training Plan

Needs and Skills Analysis:	Anticipated users are private citizens - contract initiators, and private, small businesses owners- contractors, as well as those with necessary computer skills: data entry clerk, and other various clerical staff members. Tasks to be taught are: ❖ originate contract ❖ manage contract ❖ select a contractor ❖ track work progress ❖ verify completed contract ❖ pay contractor ❖ leave contractor review Skills to be learned: ❖ Data input utilizing computer input devices ❖ Basic troubleshooting and ability to find help or identify an issue
Development Approach:	A prototype approach is to be used when creating a recorded set of correct instructions needed for the execution of basic application tasks. This development work is dependent on constant end-user feedback to provide a high retention training product.
Training Curriculum:	A document describing a how-to of a feature is to be developed as soon as a given software part is implemented and published. The questionnaires and feedback pop-up windows are provided based on user needs and requests to facilitate the user-developer communication.

Interface Design

❖ Sample Graphical User Interface



Application's Classes

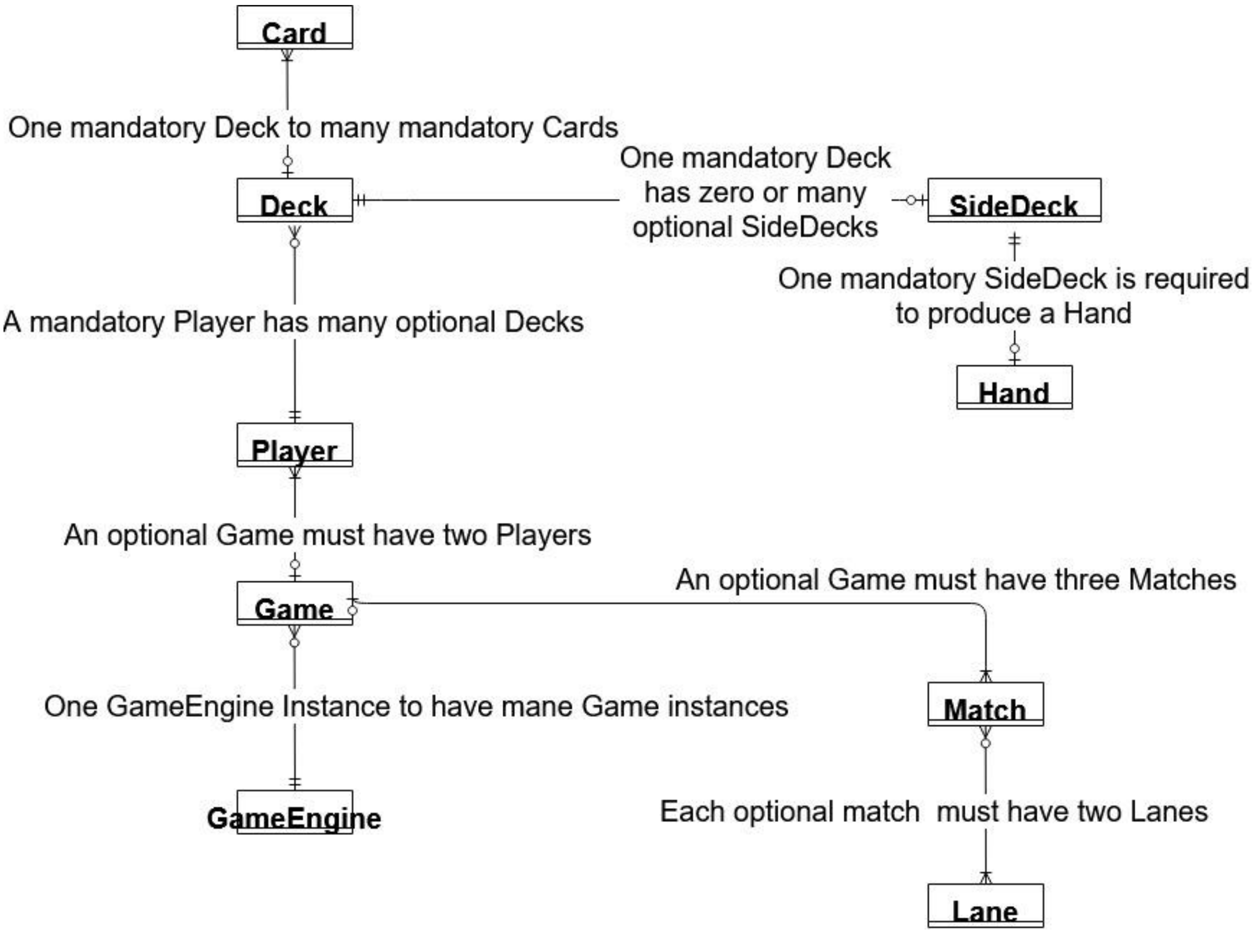
Candidate Classes:

- **Card**
- **Deck**
- **Side Deck**
- **Hand**
- **Player**
- **Game**
- **Match**
- **Lane**
- **Game Engine**

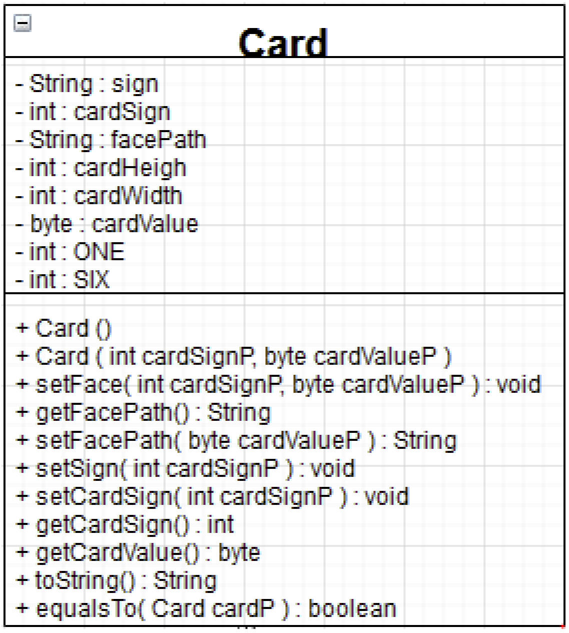
List of Relationships:

- **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
One mandatory Deck to many mandatory Cards
Deck
One mandatory Deck
has zero or many
optional SideDecks
SideDeck
One mandatory SideDeck is required
to produce a Hand
Hand
A mandatory Player has many optional Decks
Player
An optional Game must have two Players
Game
An optional Game must have three Matches
One GameEngine Instance to have mane Game instances
Match
GameEngine
Each optional match must have two Lanes
Lane

Card Class



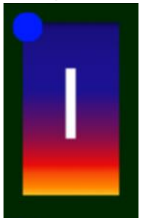
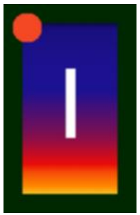
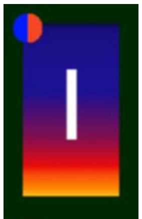
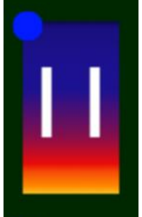
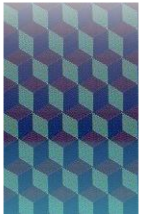
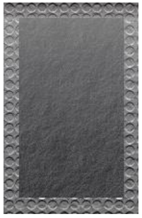
Card

- String : sign
- int : cardSign
- String : facePath
- int : cardHeigh
- int : cardWidth
- byte : cardValue
- int : ONE
- int : SIX

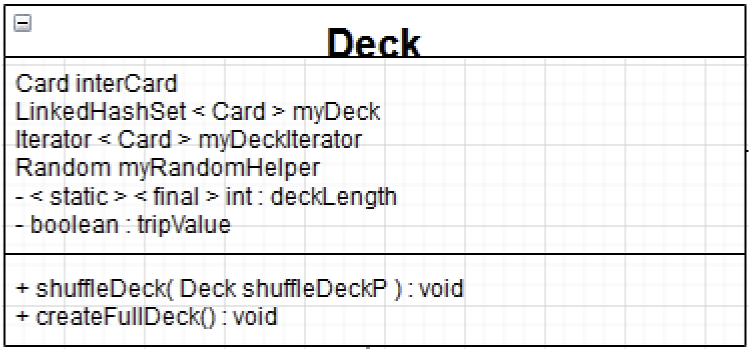
```
+ Card ()
+ Card ( int cardSignP, byte cardValueP )
+ setFace( int cardSignP, byte cardValueP ) : void
+ getFacePath() : String
+ setFacePath( byte cardValueP ) : String
+ setSign( int cardSignP ) : void
+ setCardSign( int cardSignP ) : void
+ getCardSign() : int
+ getCardValue() : byte
+ toString() : String
+ equalsTo( Card cardP ) : boolean
```

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			
	Player	**Opponent**	**Card's Spot**
Default Back Design			

Deck Class



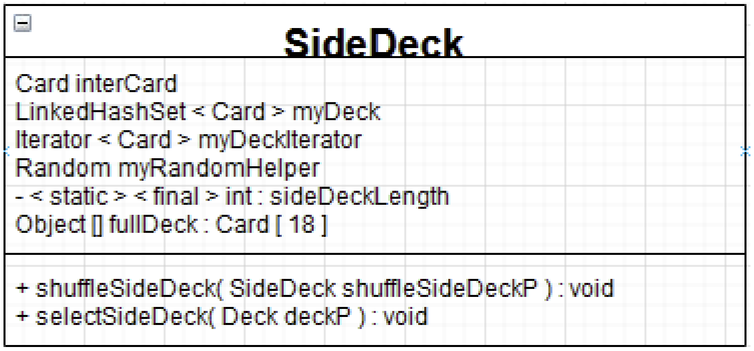
Deck

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



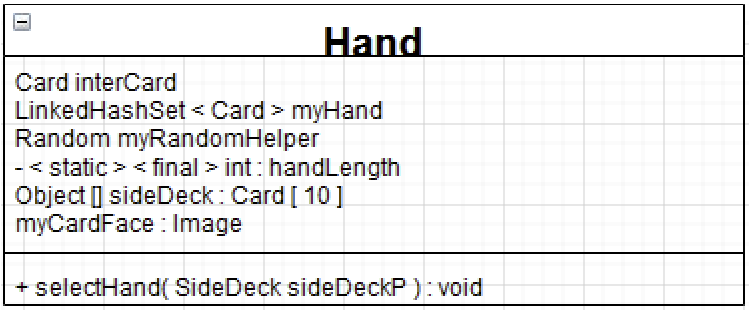
SideDeck

Card interCard
LinkedHashSet < Card > myDeck
Iterator < Card > myDeckIterator
Random myRandomHelper
- < static > < final > int : sideDeckLength
Object [] fullDeck : Card [18]

+ shuffleSideDeck(SideDeck shuffleSideDeckP) : void
+ selectSideDeck(Deck deckP) : void

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



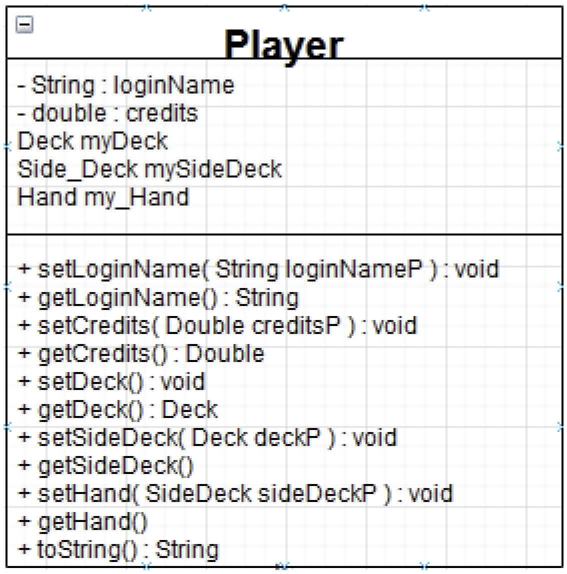
Hand

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



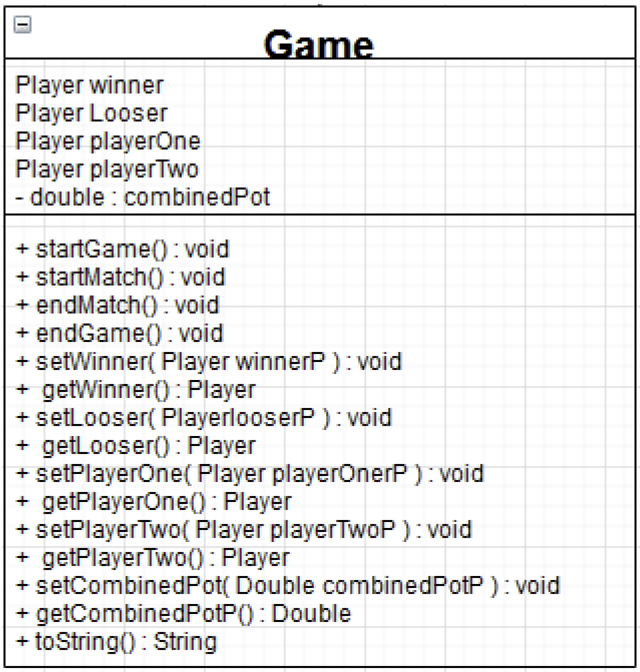
Player

- String : loginName
- double : credits
Deck myDeck
Side_Deck mySideDeck
Hand my_Hand

+ setLoginName(String loginNameP) : void
+ getLoginName() : String
+ setCredits(Double creditsP) : void
+ getCredits() : Double
+ setDeck() : void
+ getDeck() : Deck
+ setSideDeck(Deck deckP) : void
+ getSideDeck()
+ setHand(SideDeck sideDeckP) : void
+ getHand()
+ toString() : String

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



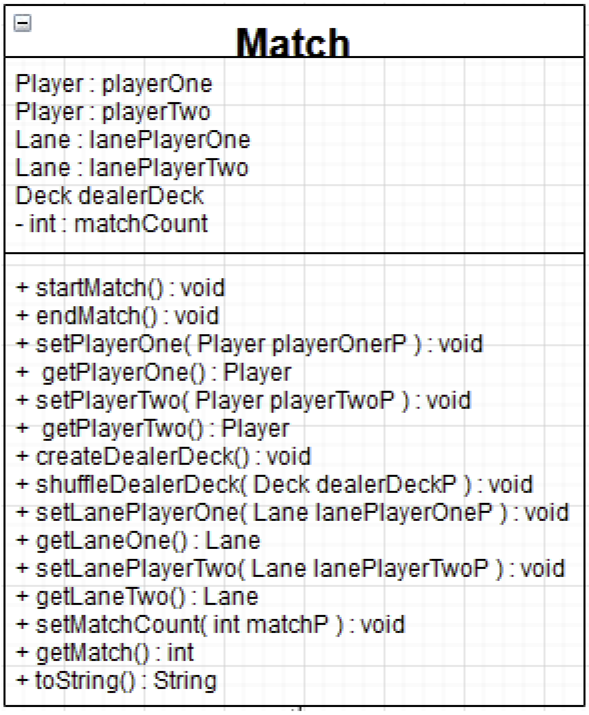
Game

Player winner
Player Looser
Player playerOne
Player playerTwo
- double : combinedPot

+ startGame() : void
+ startMatch() : void
+ endMatch() : void
+ endGame() : void
+ setWinner(Player winnerP) : void
+ getWinner() : Player
+ setLooser(PlayerlooserP) : void
+ getLooser() : Player
+ setPlayerOne(Player playerOnerP) : void
+ getPlayerOne() : Player
+ setPlayerTwo(Player playerTwoP) : void
+ getPlayerTwo() : Player
+ setCombinedPot(Double combinedPotP) : void
+ getCombinedPotP() : Double
+ toString() : String

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

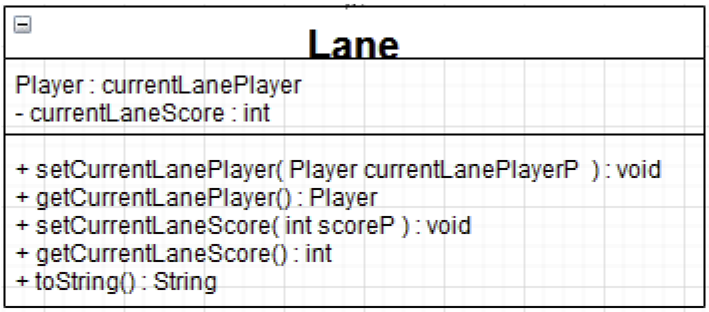


```
Match
Player : playerOne
Player : playerTwo
Lane : lanePlayerOne
Lane : lanePlayerTwo
Deck dealerDeck
- int : matchCount

+ startMatch() : void
+ endMatch() : void
+ setPlayerOne( Player playerOnerP ) : void
+  getPlayerOne() : Player
+ setPlayerTwo( Player playerTwoP ) : void
+  getPlayerTwo() : Player
+ createDealerDeck() : void
+ shuffleDealerDeck( Deck dealerDeckP ) : void
+ setLanePlayerOne( Lane lanePlayerOneP ) : void
+ getLaneOne() : Lane
+ setLanePlayerTwo( Lane lanePlayerTwoP ) : void
+ getLaneTwo() : Lane
+ setMatchCount( int matchP ) : void
+ getMatch() : int
+ toString() : String
```

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



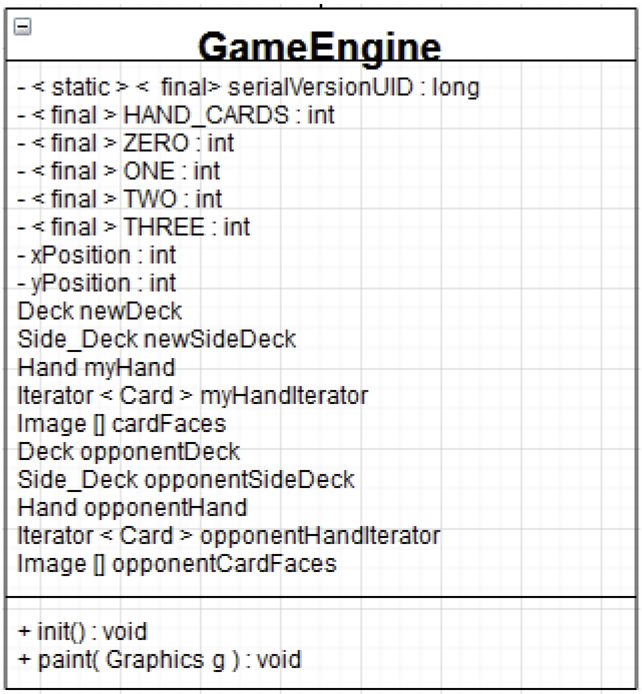
Lane

Player : currentLanePlayer
- currentLaneScore : int

+ setCurrentLanePlayer(Player currentLanePlayerP) : void
+ getCurrentLanePlayer() : Player
+ setCurrentLaneScore(int scoreP) : void
+ getCurrentLaneScore() : int
+ toString() : String

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



GameEngine

```
- < static > <  final> serialVersionUID : long
- < final > HAND_CARDS : int
- < final > ZERO : int
- < final > ONE : int
- < final > TWO : int
- < final > THREE : int
- xPosition : int
- yPosition : int
Deck newDeck
Side_Deck newSideDeck
Hand myHand
Iterator < Card > myHandIterator
Image [] cardFaces
Deck opponentDeck
Side_Deck opponentSideDeck
Hand opponentHand
Iterator < Card > opponentHandIterator
Image [] opponentCardFaces

+ init() : void
+ paint( Graphics g ) : void
```

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

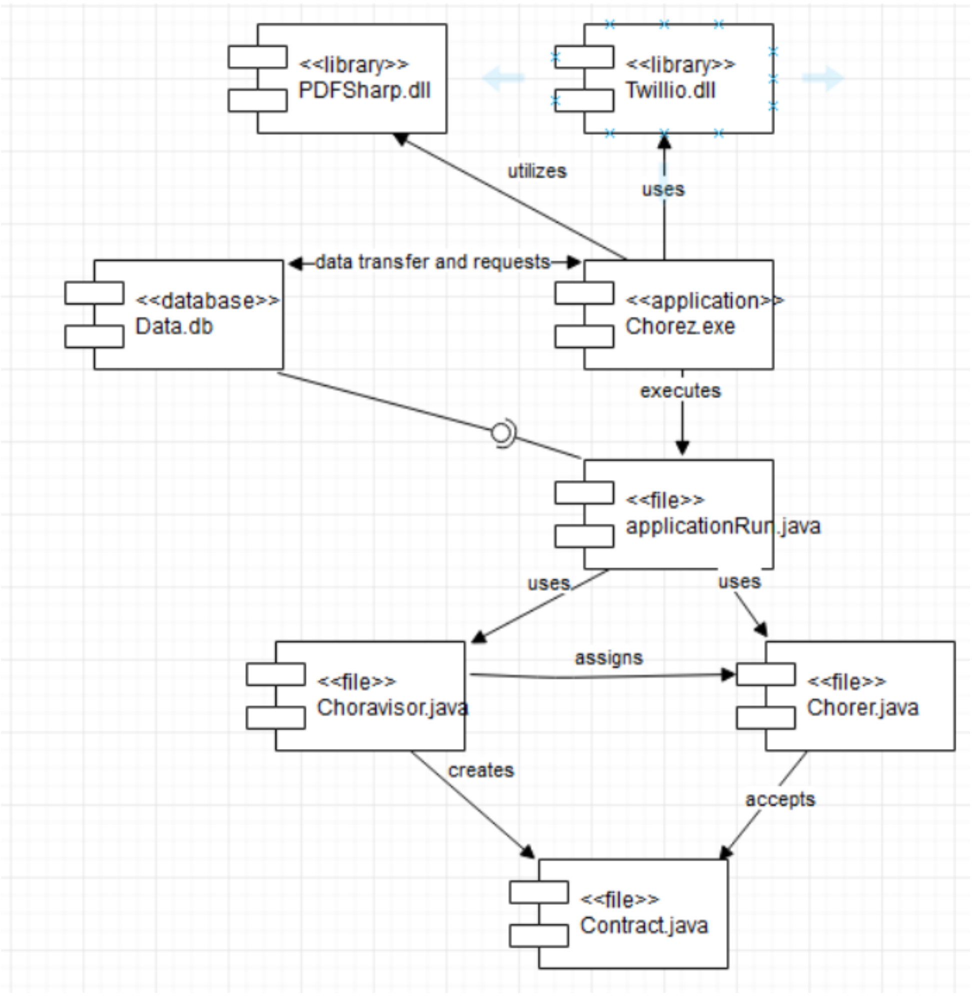
Software Architecture

The top-down approach as an architectural style and an iterative development model are used in the research and development phase.

The software is a three-tier business application to be accessed from Windows, Android or Mac operating systems. Locally and remotely stored databases are used for data storage. Microsoft Azure can host the application's logical layer. A printer is required to print transaction statements but not needed for the Chorez - Biz performance or functional operation.

Component diagram below describes modules of the software.

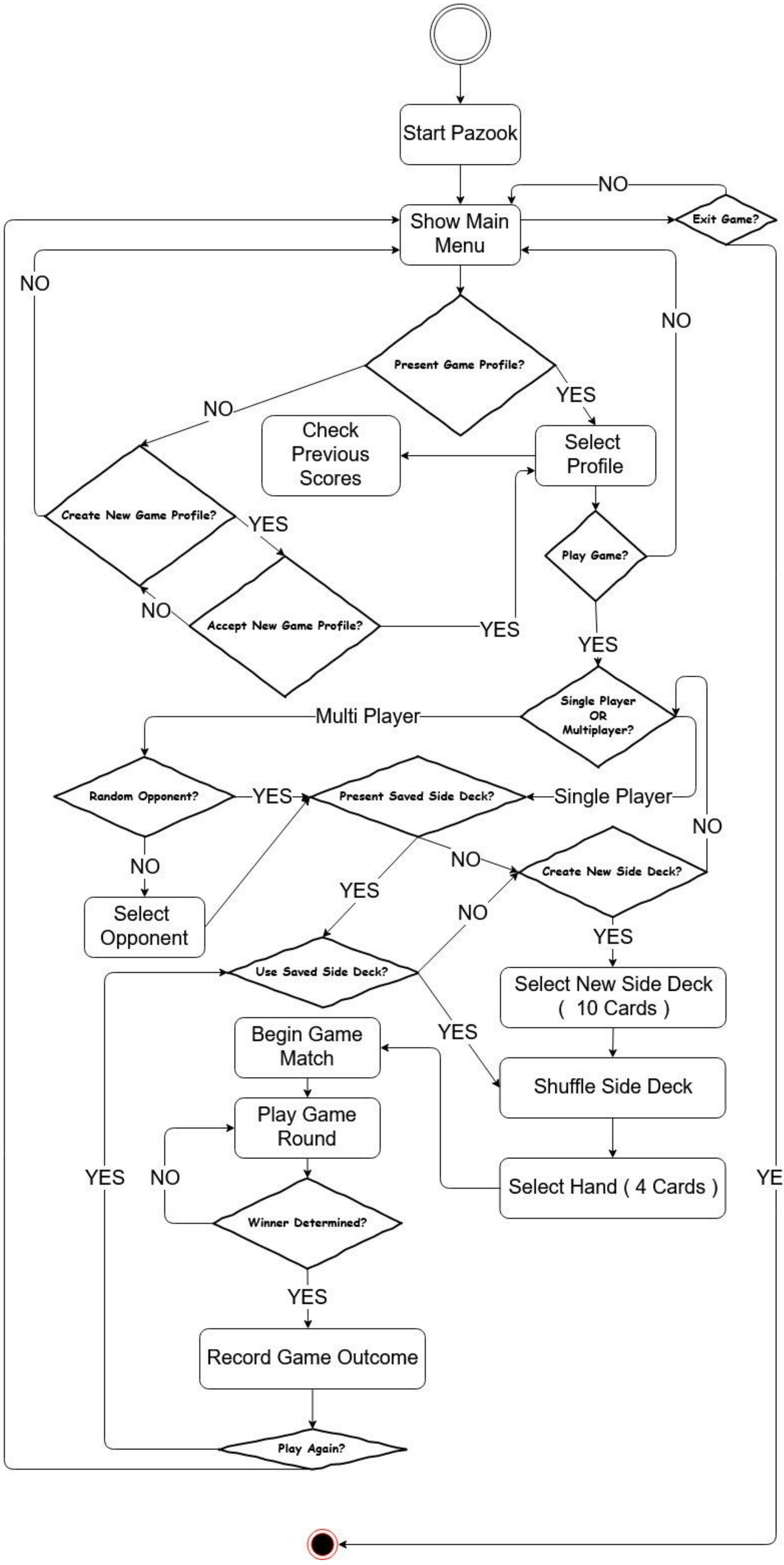
Component diagram:



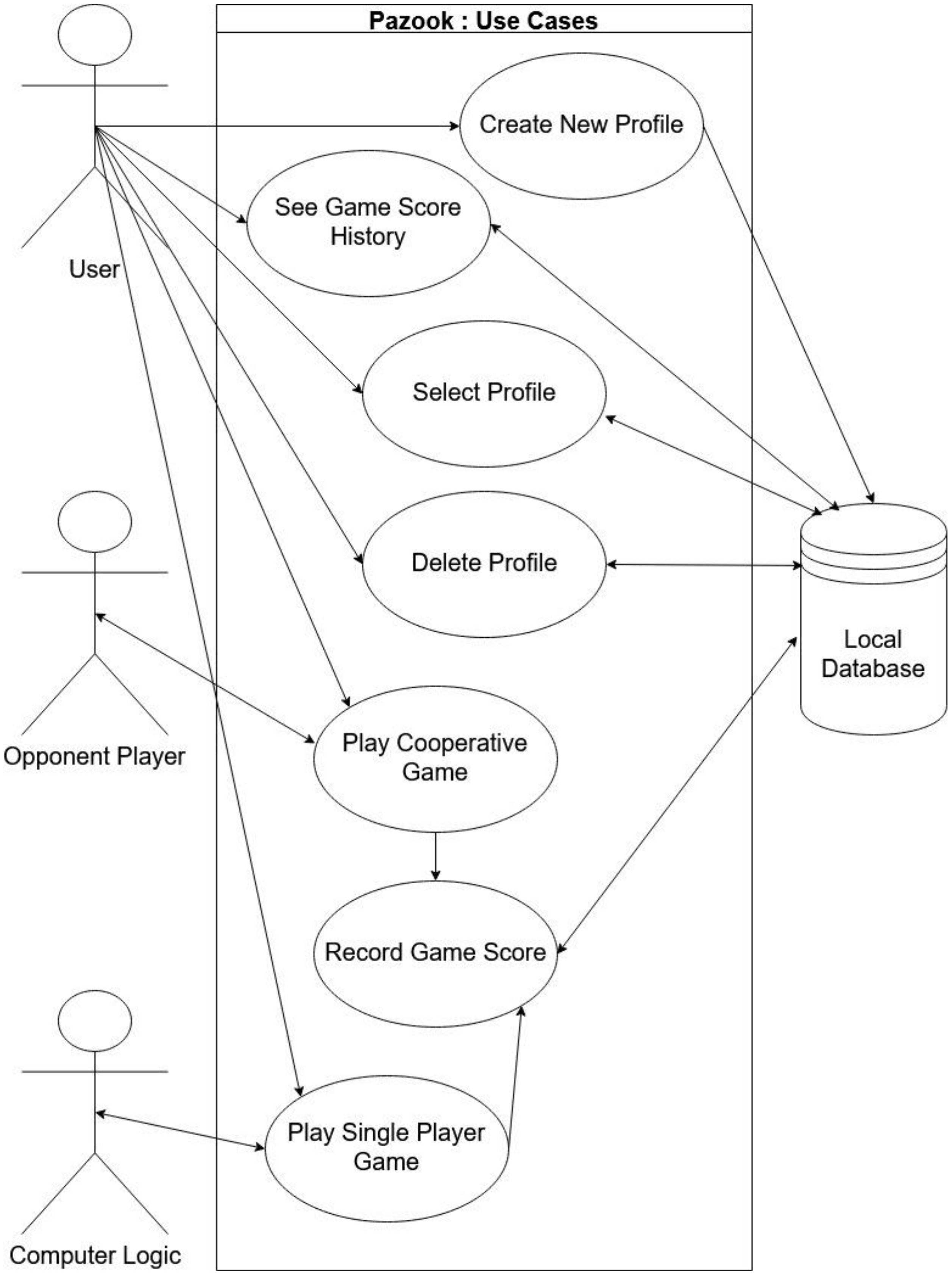
Design Constraint

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 modules, systems, and products with Chorez – Biz 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:	1. Purchase Pazook. 2. Accept Usage Rights. 3. Install Pazook. 4. Start Pazook Execution File. 5. Application Scans Local Storage for a Game Profile. 6. If Profile is Detected, it is Loaded. 7. If No Profile Detected, Profile Creation Page is Opened. 8. Upon completion of the Profile Creation Page, Main Page loaded.
Postconditions:	Creating an account to record game scores and keep game associated data for a given user in one place. Selected and Booted Up Account to a current game session.

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:	1. Accept user login credentials 2. Check any account presence against provided login credentials on a local or remote server containing a database. 3. If a single Profile is detected, Load the Profile. 4. If multiple Profiles are detected, load a selection screen. 5. After the selection has been made, output relevant Player data on the screen. 6. Continue to the Main Menu.
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:	1. Check if more than one Profile belongs to the R.P.O. 2. Provide a User with a list of available Profiles 3. Erase all associated data with a selected Profile from local or remotely stored database(s).
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:	1. Select Single Player Game from the Main Menu. 2. Bet Credits. 3. Select Random or Custom Side Deck. 4. The Game logic determines the first Player to take a Turn. 5. Each Player in a repeating sequence takes Game Turns. 6. If the Winner is determined, Credit Pot is awarded to that Player. 7. Game Record is saved.
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:	
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:	1. Select Cooperative Game from the Main Menu. 2. Select an Opponent to play a Game. 3. Bet Credits. 4. Select Random or Custom Side Deck. 5. The Game logic determines the first Player to take a Turn. 6. Each Player in a repeating sequence takes Game Turns. 7. If the Winner is determined, Credit Pot is awarded to that Player. 8. Game Record is saved.
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:	
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:	1. Select Cooperative Game from the Main Menu. 2. Select an Opponent to play a Game. 3. Bet Credits. 4. Select Random or Custom Side Deck. 5. The Game logic determines the first Player to take a Turn. 6. Each Player in a repeating sequence takes Game Turns. 7. If the Winner is determined, Credit Pot is awarded to that Player. 8. Game Record is saved.
Postconditions:	A Finished Game with Game Record data saved, Winner and Looser identified. Credit Pot is awarded to the Winner.

Play Cooperative Game W.A.N. Sequence Diagram

Record Game Score Use Case

Name:	Record Game Score
Description:	
Primary Actors:	A Player.
Secondary Actors:	An Opponent.
Preconditions:	Completed Game with Winner present.
Main Flow:	
Postconditions:	Display Main Menu.

Record Game Score Sequence Diagram

See Game Scores Use Case

Name:	See Game Score Player Game
Description:	
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:	
Postconditions:	

See Game Scores Sequence Diagram