Technical Report I



Lecturers: Reto Ferri, Johanna Decurtins

Subject: PM3, ZHAW

Team member: Erman Zankov,   
 Gökhan Bag,   
 Joël Plambeck,   
 Lukas Zoss,   
 Nikita Smailov,   
 Tobias Ritscher,   
 Samuel Stalder

Date: 27.10.2020

Content

[1 Use-Case Model 2](#_Toc55706002)

[1.1 System Sequence Diagram (Play Thief Scenario) 10](#_Toc55706003)

[1.2 GUI 11](#_Toc55706004)

[2 Supplementary Specification 13](#_Toc55706005)

[2.1 Additional Card Rules 13](#_Toc55706006)

[3 Domain Model 17](#_Toc55706007)

[4 Software architecture 18](#_Toc55706008)

[4.1 Package diagram 18](#_Toc55706009)

[4.2 State Machine diagram 19](#_Toc55706010)

[5 Design artifacts 20](#_Toc55706011)

[6 Implementation 22](#_Toc55706012)

[7 Project management 23](#_Toc55706013)

[7.1 Team structure 23](#_Toc55706014)

[7.2 Effort estimation 23](#_Toc55706015)

[7.3 Risks 25](#_Toc55706016)

[8 Glossary 26](#_Toc55706017)

# Use-Case Model

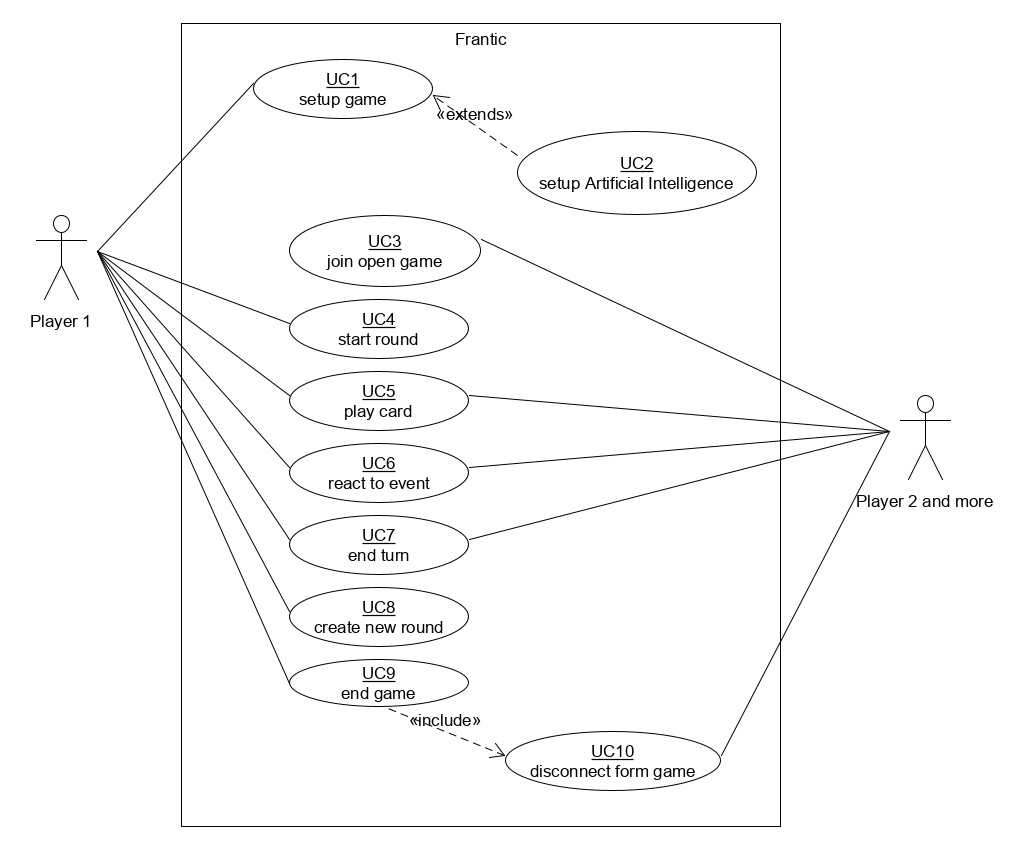


Figure 1: Use-Case-Model of Frantic

|  |  |
| --- | --- |
| UC1 | setup game |
| Scope | This use-case describes the game setup, which the first player undertakes to create a singleplayer game or multiplayer game. In a singleplayer game, the player can choose the amount of Artificial Intelligence Competitors (UC2). The multiplayer mode will create a server connection for other players to connect to and play (UC3). |
| Level | User-function that initiates the game for a player or many players to play. |
| Primary Actor | Player 1 |
| Stakeholders and Interests | The Player wants a smooth and easy user-interface to start a game, without any difficulties. |
| Preconditions | Game executable has to be started in order to be able to set up a game. |
| Success Guarantee / Postconditions | The Game can be started and played, either alone against AI or with friends. (UC3 – UC10) |
| Main Success Scenario /  Basic Flow | 1. Player starts a new game. 2. System presents the player with three options: Single Player, Multiplayer, Join a Game. 3. Player chooses Single Player. 4. System presents the player with the settings: number of AI Players (UC2), end of game condition. 5. Player chooses a number of AI players between 2 and 6 and the end of game condition as a number of points between 113 and 179. 6. Player confirms the choice. 7. System starts a game according to the player’s choice. |
| Extensions / Alternative Flows | \*a. At any time, Player closes the window or System fails:  (System logs after each completed move)   1. Player restarts the game, requests recovery of prior state. 2. System reconstructs prior state.   3a. Player chooses Multiplayer:   1. System presents the player with the settings: number of other players that can join, end of game conditions. 2. Player chooses a number of players between 2 and 6 and the end of game condition as a number of points between 113 and 179. 3. Player confirms the choice. 4. System starts a game according to the player’s choice and opens a connection for other players to join the game.   3b. Player chooses Join a Game: see UC3 |
| Special Requirements | The setup has to be easy, readable and understandable in basic languages. Depending on success, the game has to be translated into different languages and made available. |
| Technology and Data Variations List | None |
| Frequency of Occurrence | Every time a game has to be created, this use-case will be called up. |
| Open Issues | None |

Table 1: Use-Case description for set game

|  |  |
| --- | --- |
| UC2 | setup Artificial Intelligence |
| Description | This use-case describes the setup of Artificial Intelligence for the Singleplayer mode of the game. When the user chooses the amount of enemy AIs and starts the game. The number of AIs will be represented in the game itself with AI players. |
| Level | System-function that reacts to the user’s choice of having AI players as enemies. User only chooses amount in UC1. |
| Primary Actor | Player 1 |
| Stakeholders and Interests | The player wants the right number of AIs created to play against. In addition, the player wants the AI to play like a real player with a certain tactic and form. |
| Preconditions | UC1 has to be initiated and Singleplayer has to be chosen. |
| Guarantee of Success / Post-Conditions | The game starts successfully with the chosen amount of AI player to face off in the rounds to come. |
| Standard Procedure | After the player clicked on Singleplayer and chooses the amount of AI players to face off in the game, the System will react on the Start Game click and undertakes the necessary steps to ensure the game has the chosen amount of AI players and runs smoothly without a hitch. |
| Extensions | If the player closes the window, the game will automatically end. |
| Special Requirements | An Artificial Intelligence has to be programmed that plays with a certain tactic and decision making, so as to make the game interesting for the player. |
| Lists of Technology and Data Variations | None |
| Frequency of Occurrence | Every time a game has to be created and a singleplayer game is started, this use-case will be called up. |
| Various | None |

Table 2: Use-Case description for setup Artificial Intelligence

|  |  |
| --- | --- |
| UC3 | join open game |
| Description | This use-case describes the third option the player encounters when starting the game. A player looking to join a Multiplayer game already created by his or her friend. |
| Level | User-function that joins an already existing game. |
| Primary Actor | Player 2 and more |
| Stakeholders and Interests | The player wants to connect to the correct game without any connection issues or problems. The player wants a stable game without long waiting times to see other players’ actions. |
| Preconditions | A multiplayer game has to be created with conditions and the system has to create a server, for which Player 2 and more can connect to. |
| Guarantee of Success / Post-Conditions | The game starts for all participating players and runs without any major inconveniences. |
| Standard Procedure | Player 1 creates a Multiplayer game with the player’s personal configuration. After that the connecting face will start, where Player 2 and more can connect to the hosted game via a code. As of this point the game has not started yet. As soon as everybody joined the game. The host, Player 1, can start the game. |
| Extensions | If the host player, Player 1, closes the window, his or her game will end. The other player will get a notification that the hosted game does not exist anymore. If one of the other players closes their window, the connection to the host game will cease to exist. For the rest, everything will continue as per usual. |
| Special Requirements | The host player should be able to see who joined the game. The joining player should be notified if the connection worked and that they are connected. |
| Lists of Technology and Data Variations | An optional way to implement, instead of a code, is an invite of the host player himself or herself.  A kick function can be implemented when a player joined that should not belong to the current game. |
| Frequency of Occurrence | Every time a player connects to a hosted game, that has not started yet, this use-case will be called up. |
| Various | How will connection-error be handled? How stable will the connection be between players? |

Table 3: Use-Case description for join open game

|  |  |
| --- | --- |
| UC4 | start round |
| Description | This use-case describes the starting of a round by the host player. The system has to create decks and hands for the player and load it to all the participants simultaneously. In a Singleplayer game, the player and the AI will receive information of hand cards and enemy statuses to work with. The same thing happens with a Multiplayer game. |
| Level | User-function that executes a game. |
| Primary Actor | Player 1 |
| Stakeholders and Interests | The player wants the User Interface to adept to game format. In addition, the players expect the display of hand cards and other player information to think out the best strategy. |
| Preconditions | A Singleplayer or Mulitplayer game has to be created with all necessary configurations. AI or player connected to the game. |
| Guarantee of Success / Post-Conditions | The game starts for all participating players and runs without any major inconveniences. |
| Standard Procedure | As soon as Player 1 creates a Singleplayer or Multiplayer game and presses on Start Game, the system will create a different User Interface and start the round for all participants to play. |
| Extensions | If the host player, Player 1, closes the window, his or her game will end. The other player will get a notification that the hosted game does not exist anymore. If one of the other players closes their window, the connection to the host game will cease to exist. For the rest, everything will continue as per usual. |
| Special Requirements | None |
| Lists of Technology and Data Variations | None |
| Frequency of Occurrence | As soon as a game is created, this step has to be taken to continue. |
| Various | None |

Table 4: Use-Case description for start round

|  |  |
| --- | --- |
| UC5 | play card |
| Description | This use-case describes the player playing a card from his or her hand. This use-case includes the display of handcards, selection and the playing of a card. It marks an important step in the game. |
| Level | User-function that is a key role of the game. |
| Primary Actor | Player 1, Player 2 or more |
| Stakeholders and Interests | The player wants an easy and self-explaining game interface for him or her to use. In addition, the players expect the display of hand cards and other player information to think out the best strategy. |
| Preconditions | A Singleplayer or Mulitplayer game has to be created with all necessary configurations. AI or player connected to the game. The game has to be started with the round system implemented. |
| Guarantee of Success / Post-Conditions | The player can choose a card from his hand and double click it to play it. |
| Standard Procedure | A player (Player 1, Player 2 or more) will be notified that it is his or her turn. The player will study his cards and decide on what his move will be. Either playing a card or moving to an event phase, like drawing a card. As soon as he played a card the event phase will start. |
| Extensions | If the host player, Player 1, closes the window, the game will end for everybody. The same thing happens when any of the other player disconnect from a game.  When the player does not play a card, because it is not possible or not expected from a tactical standpoint, a drawing event will occur. |
| Special Requirements | An interchangeable design for the user interface is possible to implement.  The execution of playing cards can be either through double click or a play cards button.  Different languages for the user interface can be implemented. |
| Lists of Technology and Data Variations | None |
| Frequency of Occurrence | Every turn of a player, the play card case will be called up to handle an important basic game function. |
| Various | None |

Table 5: Use-Case description for play card

|  |  |
| --- | --- |
| UC6 | react to event |
| Description | This use-case describes the reaction to an event by either the player or an enemy player. This event could be drawing a card, reacting to a special card, interdicting with a special card or activating a special event card. |
| Level | User- or System-function that react to an event. |
| Primary Actor | Player 1, Player 2 or more |
| Stakeholders and Interests | The player wants to be notified if he or her will be affected from an event or can counter an active event. The player wants to be notified what kind of event is happening. The player wants to know what the effects of the played cards are. |
| Preconditions | A game, Singleplayer or Multiplayer, has to exist with its configurations. The game has to be started and the first player will have played or drawn a card. The event will be kick-started by the card the player laid on the deck. |
| Guarantee of Success / Post-Conditions | The system notifies the next player or other effected players if and how they want to react to a given event. After the event has been resolved, the next turn will be called up. |
| Standard Procedure | A player does not play a card, which means a draw event will be called up, for the player to draw a card.  When the player plays an event card like Fantastic Four, the player will be able to react to the color choosing event.  When a player plays 2nd Chance, a second card event occurs.  When the player plays a special card that effects another player, the other player will be asked if he or she has Counterattack to counter the effects.  When a special card even is activated, all players will be notified and included to the drawbacks of the event, for example: DOOMSDAY, EXPANSION, SEPPUKU, TORNADO, etc. These special events cannot be countered or avoided except with “Lucky Bastard”.  These are the basic procedures that can be encountered in this use-case. |
| Extensions | If the host player, Player 1, closes the window, the game will end for everybody. The same thing happens when any of the other player disconnect from a game.  When a player does not wish to react to an event, he or she can decline, and the game continues. |
| Special Requirements | Every event calls up for a different reaction to different players. There are 18 special card types an event can be called up to. In addition, when a special event is activated, the player are notified that it will be unavoidable. |
| Lists of Technology and Data Variations | None |
| Frequency of Occurrence | Every time a player played a card that effects the game and/or other players this use-case occurs. |
| Various | How will the event call be made fair?  How much time is given for the player to react? |

Table 6: Use-Case description for react to event

|  |  |
| --- | --- |
| UC7 | next turn |
| Description | This use-case describes the end of a player’s turn and the transition to the next player’s turn. |
| Level | User- or System-function that moves to the next turn. |
| Primary Actor | Player 1, Player 2 or more |
| Stakeholders and Interests | The player wants a smooth transition from his or her turn to the next players turn. |
| Preconditions | A player finished his turn by player or drawing a card. The event phase is resolved by all players involved. |
| Guarantee of Success / Post-Conditions | The system notifies the next player that he or she can play her turn. |
| Standard Procedure | After a player has played or reacted to an event, his turn will end automatically, and the next player will be notified of his or her turn. |
| Extensions | If the host player, Player 1, closes the window, the game will end for everybody. The same thing happens when any of the other player disconnect from a game.  The player cannot interfere with the system giving making the next turn available. |
| Special Requirements | None |
| Lists of Technology and Data Variations | None |
| Frequency of Occurrence | Every time a player played a card and reacted to an event, the next turn will be called up. |
| Various | None |

Table 7: Use-Case description for next turn

|  |  |
| --- | --- |
| UC8 | create new round |
| Description | This use-case describes the creation of a new round when the previous round has been ended. If a player has played all his cards unto the discard pile or the draw deck is empty the round is completed. As long as the victory conditions are not met, a new round will be started. |
| Level | User- or System-function that decide if a new round is necessary. |
| Primary Actor | Player 1 |
| Stakeholders and Interests | The player wants the system to create a new round after the previous one has ended. The player wants to continue to play until the victory or defeat conditions are met. |
| Preconditions | One of the participating players emptied his hand cards unto the discard pile or the draw card deck is empty the round ends and the player count points. |
| Guarantee of Success / Post-Conditions | When the victory or defeat conditions are not met, the system saves the players’ points and creates a new round to continue playing. |
| Standard Procedure | A player has played all his cards unto the discard pile or the draw deck is empty the round is completed. All the remaining players’ hand cards will be added up and saved in their name. The system compares the players’ points and starts a new round if the point limit has not been reached. If the point limit is reached, use-case 9 will be activated. |
| Extensions | If the host player, Player 1, closes the window, the game will end for everybody. The same thing happens when any of the other player disconnect from a game. |
| Special Requirements | None |
| Lists of Technology and Data Variations | None |
| Frequency of Occurrence | Every time a player empties his hand cards or the draw deck is emptied, the points are compared and the system decides for a new round. |
| Various | Is it possible to vote to end the game early?  Can the victory/defeat conditions be changed during the game? |

Table 8: Use-Case description for create new round

|  |  |
| --- | --- |
| UC9 | end game |
| Description | This use-case describes the games end. This occurs when the victory/defeat conditions are met. In this case, a player reached the maximum amount of points. |
| Level | User- or System-function that decide if a new round is necessary. |
| Primary Actor | Player 1 |
| Stakeholders and Interests | The player wants the system to end the game when a player has met the victory/defeat conditions. The host player wants to be returned to the main menu. The other players too will be returned to the main menu via use-case 10. |
| Preconditions | A round has ended, and the system adds the points earned to the already existing points. A player’s points will be compared to the victory/defeat condition points. |
| Guarantee of Success / Post-Conditions | If a player’s points are exactly or more than the victory/defeat condition amount, the game ends for all the participants. |
| Standard Procedure | A round has ended when a player emptied his hand cards, or the draw deck has been depleted. The points are added to the already existing player points and compared with the condition points. If the condition is met, the game is ended. All player receives a notification on who lost the game and who won. Then the game ends, and the host is returned to the main menu. The other players are confronted with use-case 10. |
| Extensions | If the host player, Player 1, closes the window, the game will end for everybody. The same thing happens when any of the other player disconnect from a game. |
| Special Requirements | None |
| Lists of Technology and Data Variations | None |
| Frequency of Occurrence | Every time a player meets the victory/defeat condition the game ends. |
| Various | None |

Table 9: Use-Case description for end game

|  |  |
| --- | --- |
| UC10 | disconnect form game |
| Description | This use-case describes the disconnecting of the participating players from the host player’s game. |
| Level | User- or System-function to disconnect. |
| Primary Actor | Player 2 or more |
| Stakeholders and Interests | The player wants to disconnect from the game after a successful game or a “rage quit”. |
| Preconditions | A game has ended and the victor and looser is determined. |
| Guarantee of Success / Post-Conditions | Every player is returned to the main menu and the system managed to shut down connection and other unnecessary functions from this point out. |
| Standard Procedure | A game has counted the points of every player and determined the victor and looser. This information is displayed for all players to see. After a given time period the players are returned to their respective main menu. Players connected to the host player will be disconnected from the hosted game. |
| Extensions | none |
| Special Requirements | None |
| Lists of Technology and Data Variations | None |
| Frequency of Occurrence | Every time a game is ended. |
| Various | None |

Table 10: Use-Case description for disconnect form game

## System Sequence Diagram (Play Thief Scenario)

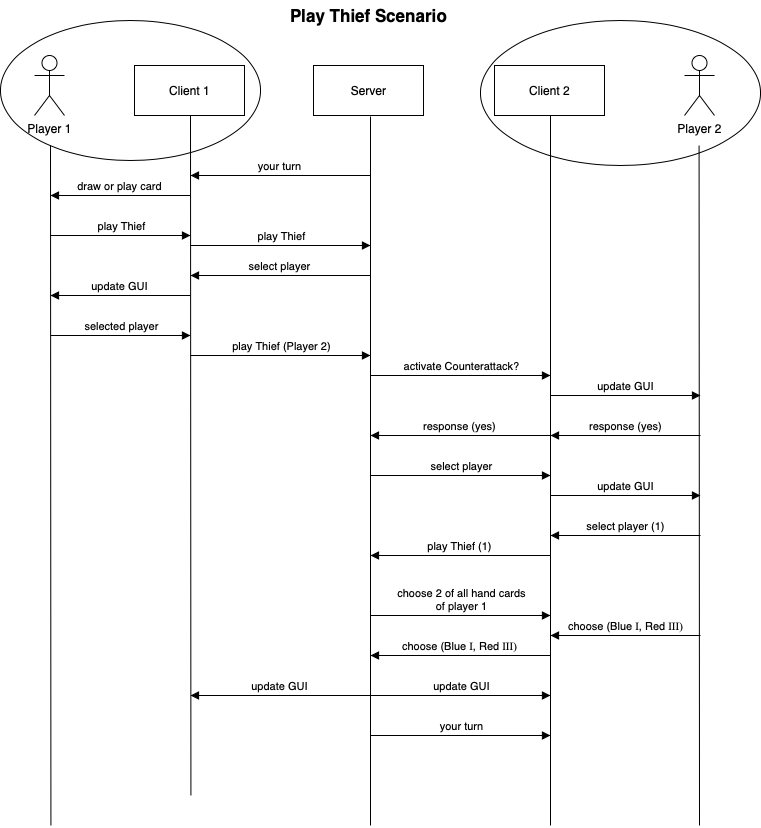


Figure 5: System Sequence Diagram, Play Thief Scenario of the card game Frantic

There are two players (Player 1 and Player 2) who are both connected to the server as a client.

In this scenario we are simulating that Player 1 plays the card Thief against Player 2 and Player 2 counters with a counterattack.

1. Server sets the current Player to Player 1 and sends this information to Client 1.
2. Player 1 has to choose between playing a card or drawing one.
3. Player 1 chooses to play Thief.
4. Client 1 tells the Server, that Thief has been played.
5. Client 1 has to ask the player for the target and the GUI is updated.
6. Player 1 chooses Player 2 as target.
7. Client 1 sends this information to the server.
8. Server checks if Player 2 has a counterattack (true).
9. Server asks over Client 2 if Player 2 wants to play a counterattack.
10. Player 2 responses yes, this is sent to the server.
11. Player 2 has to choose a target.
12. Player 1 is selected.
13. Server checks if Player 1 has a counterattack (false).
14. Server sends all handcards of Player 1 to Client 2.
15. Client 2 shows the Cards to Player 2, who has to select two cards.
16. Blue 1 and Red 3 is chosen.
17. This information is passed to the Server.
18. The GUI of all players is updated, they both see the current gameboard.
19. Server sets the current Player to Player 2 and sends this information to Client 2.

## GUI

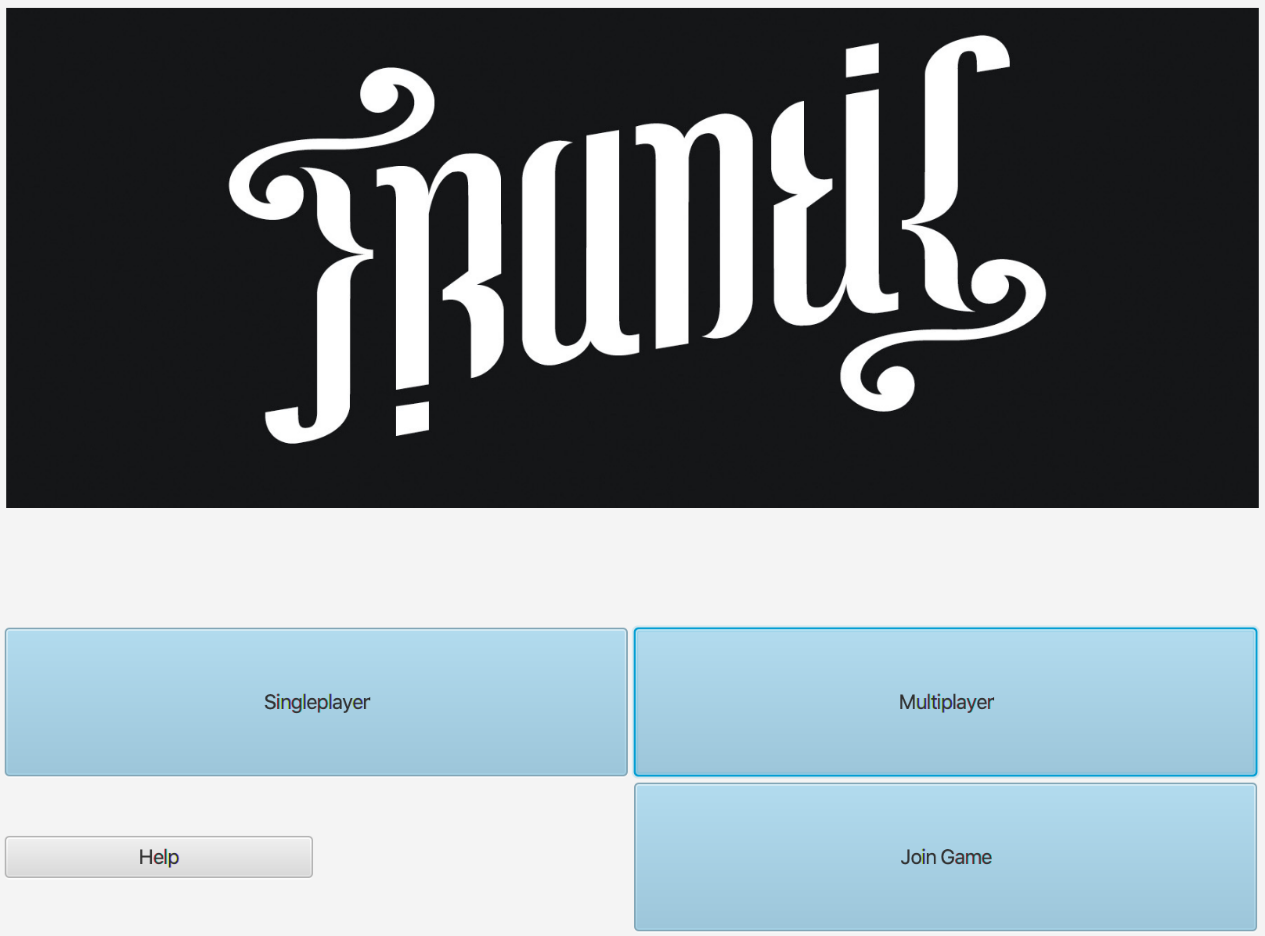


Figure 8: Screenshot of the main menu

The main menu of the game is the entry point of Frantic. In the main menu you can choose between the two different game modes: Singleplayer and Multiplayer. With the Singleplayer mode you can get to know the game and play against computer controlled opponents to practice your tactics. The Multiplayer mode allows you to test your tactics against your friends from all over the world!

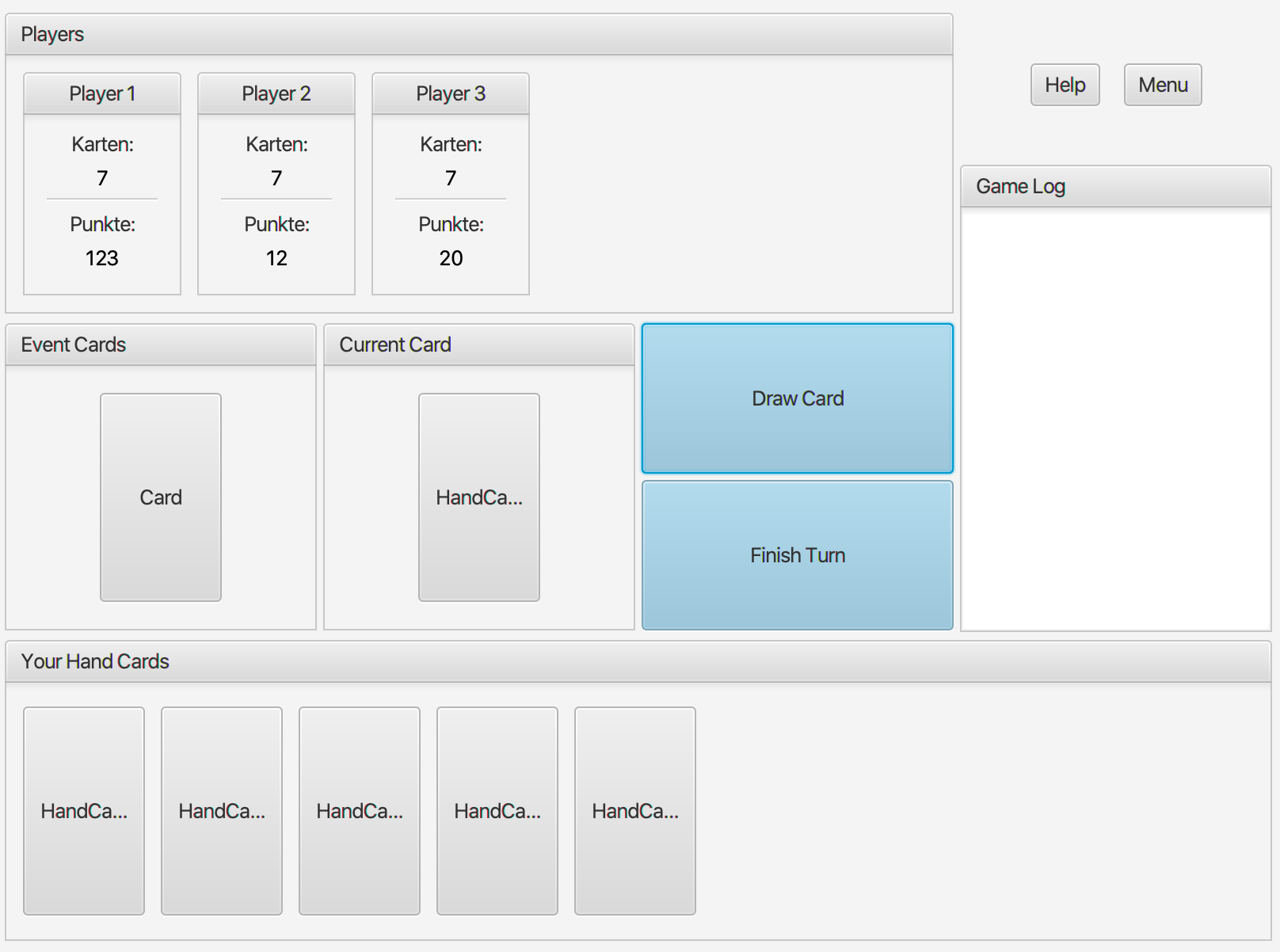


Figure 9: Screenshot of the main GUI

Once the game mode is selected the game is presented with the players hand displayed at the bottom. The last played card is displayed right above the hand cards and between the event cards and the buttons to draw a card or finish your turn. On the right hand side the past events are written in the game log. At the very top, next to the menu and help button, the opponents are displayed with their key metrics like total points and cards left in their hand. With this game view it is easy to make well informed decisions and play according to your tactics.

# Supplementary Specification

**Functionality**

* All moves and errors are logged.
* Illegal moves should be detected and prevented.
* In multiplayer mode the program should create a virtual room with all participants.
* A multiplayer game must comply with industry security standards to prevent data manipulation and unauthorized access.

**Usability**

* The program must be executable on the current computer systems (MacOS, Windows).
* The player should be able to enter a game within 30 seconds of starting the application.

**Reliability**

* It must be possible to interrupt the game at any time without losing any data.
* If the connection to the other players is lost, the program must try to reconnect.

**Performance**

* Creating a connection to a virtual room may take a maximum of five seconds.
* The move of a computer-controlled opponent must be calculated within 0.5 seconds.
* The transfer of a player's move to the other players may take a maximum of one second and must be successful in 99.9% of cases.

**Supportability**

* There is a chat function that can be used by all players in the same virtual room.

**Implementation Constraints**

* The use of Java technology is already stipulated.

**Free Open Source Components**

* Free and open source Java components should be used in building the game. That makes JavaFX a strong contender for building the GUI.

**Legal Issues**

* The game’s implementation was greenlighted by the original creators. We must plan for possible licensing restrictions.

## Additional Card Rules

* **Ein Bild, das Text, Objekt, Schild, Ende enthält.

  Automatisch generierte Beschreibung2nd Chance:** The player has to play another card on top of 2nd Chance, either color on color or colorless Special Cards. If a player cannot play a card from his or her hand, he has to draw one from the deck.
* **Ein Bild, das Text enthält.

  Automatisch generierte BeschreibungColor Swap:** Can be played on two different colors and swaps to the other color.
* **Ein Bild, das Zeichnung, Schild, Ende enthält.

  Automatisch generierte BeschreibungExchange:** The player of this card gives another player two cards of his choice from his hand and in exchange has to blindly draw two cards from his chosen opponent.
* **Gift:** The player of this card gives two cards from his hand to another player.
* **Skip:** The player of this cards chooses a fellow player, who is suspended for one turn.
* **Ein Bild, das Schild, Zeichnung, Straße enthält.

  Automatisch generierte BeschreibungThief**: The player of this card has to look into the cards of another player and steal two of his/her hand cards.
* **Ein Bild, das Schild, Kühlschrank, Zeichnung, Ende enthält.

  Automatisch generierte BeschreibungTroublemaker:** Activates a special event card.
* **Fantastic:** This card can be played on any card. You can choose a number or a color.
* **Fantastic Four:** The player of this card chooses a person, who in return has to draw four cards form the deck. It is also possible to determine multiple players and divide the four cards between them.
* **Ein Bild, das Text enthält.

  Automatisch generierte BeschreibungCounterattack:** As soon as a Special Card is played against a player, this card can be thrown in instantly by the victim. The effect is cancelled, and the player of this card can redirect it to another player. The target of the freshly obtained effect is freely choosable.
* **Equality:** The player of this card chooses a fellow player who holds fewer cards than himself in his hands. This player has to draw as many cards until their number of cards is equal.
* **Inequality:** The player of this card can choose a victim, which has to draw as many cards form him/her until the victim holds more cards in its hand.
* **Ein Bild, das Schild, Straße, Ende, Bus enthält.

  Automatisch generierte BeschreibungLucky Bastard:** This card can be thrown in right before a special event is activated. The player of this card is then excluded from the event and its effect is not applied to him/her.
* **Ein Bild, das Text enthält.

  Automatisch generierte BeschreibungNice Try:** As soon as a player got rid of all his hand cards and therefore ends the current round, this card can be thrown in immediately. The player who ended the round then needs to draw three cards and the game round continues.

***Important:*** If multiple players finish off their cards from an event and Nice Try is played, every player who finished has to draw three cards.

* **Special Favours:** The player of this cards can exchange all special cards with another player.
* **Curse:** This card cannot be played or discarded. But it can be passed on or exchanged at any given opportunity. If a player holds this card at the end of a round, it counts as 13 points.
* **Fuck You:** You can only dismiss this card, when you have exactly then cards in your hand, including the Fuck You cards itself. The round continues with the card played before Fuck You.

***Important:*** This card can only be blindly obtained by an opponent and not be willingly given.

***Exception:*** During Special Event cards it can be thrown away or passed on.

* **Mimicry:** When being played, this card can become any existing Special Cards.  
  Important: If this card imitates a colored special card, it can only be played on said color.
* **Black Hole:** All Black Cards in the hands of the players go to the player, who activated the Black Hole event.
* **Capitalism**: He that has plenty of goods shall have more. All players double their hand cards.
* ****Charity: Every player has to pick one card from the player with the most hand cards.

***Important:*** If two or more player have equal amount of most cards, cards are picked form all of them. These players don’t have to draw cards form each other.

* **Communism:** Everyone has to draw as many cards to equal the player, who holds the most cards in his hands.
* **Crowdfunding:** Every player has to give one card to the player with the least hand cards.

***Important:*** If two or more players have an equal amount of least cards, cards are given to all of them. There players don’t have to give cards to each other.

* **Distributor:** All players give their hand cards in a small deck face down to the player who played the Black Card. This player can look at each small deck and distribute them between the players, without shuffling them. The player who played the Black Card has to distribute his/ her hand cards as well and keep a deck form another player.
* **Doomsday:** The game round is immediately over. Every player receives 50 points.
* **Double Taxation:** The player who holds the least points with the three highest cards has to draw one card, the player with the second least points has to draw 2 and so on. In case of equal score, the affected players have to draw the equal amount of cards.
* **Earthquake:** Every player gives his cards to the player to his right.
* **Event Manager:** The next three Event Cards from the deck are flipped over. The player of the Black Card then chooses one of these events which will be executed.
* **Expansion:** The players have to draw cards from the deck accordingly: The 1st player draws one card, the 2nd draws two, the 3rd three and so on.
* **Finish Line**: The game round is immediately over and the players count their points according to their hand cards.
* **Friday The 13th:** Nothing happens.
* **Gambling Man:** Every player has to place a preferably low Numeral Card of the last played color face down. All cards are simultaneously turned around. The player with the highest digit has to take the other cards in. Players without Numeral Cards of said color have to draw two cards as penalty.

***Important:*** If no color has been played so far, the event is ineffective.

* **Identity Theft:** The player with the lowest amount of points swaps the score with the player who holds the most points.

*If the points are not written down: The player of* *the Black Card chooses two players (also themselves* *possible) who swap their place at the table, and with* *it their cards.*

* **Last Chance:** Every time a player has discarded the last card an event is activated as a last resort.
* **Market:** As many cards as there are players, the top cards from the deck are turned face up in front of the players. The players then pick in turn one card to take in their hands.
* **Mating Season:** Every combination that can be achieved with Numeral Cards have to be disposed of. Combinations would be pairs, three of a kind, four of a kind and so on. The color of the cards doesn’t matter.
* **Merry Christmas:** Every player has to give all of their hand cards to other players. They can divide them as they please.
* **Mexican standoff:** All players dispose of their cards and draw in turn three new cards from the deck.
* **Plague:** There are immediately two additional event cards executed, one after the other. Even if the first event would end the current round or a player has discarded all cards, the second event will be executed as well as possible.
* **Plus One:** New rule for the rest of the round: Every time the players have to draw cards from the card deck, they have to draw one more.
* **Recession:** The players have to dispose of cards from their hands accordingly: The 1st player has to dispose of one card, the 2nd disposes of two, and the 3rd of three and so on.
* **Repeat:** The player of the Black Card has to execute an event that has already been uncovered.

***Important:*** If no other event has been uncovered so far, the event is ineffective.

* **Robin Hood:** The player with the smallest amount of hand cards swaps his cards with the player who holds the most.
* **Russian Roulette:** All players put one card from their hand together. These cards get shuffled and put on top of the card deck and the game continues.
* **Seppuku:** The players decide in turn if they want to commit Seppuku. If they risk it, they have to draw the top card from the card deck. If it is a numeral card they get credited 21 points, if it’s any other card they get an additional 42 points. If they decide not to commit Seppuku then they get 21 additional points.

*If the points are not written down: If a numeral* *card is drawn they can dispose of two cards from* *their hand, if it’s any other card they have to draw* *four cards from the deck. If they decide not to commit Seppuku, then they have to draw two cards from* *the deck.*

* **Surprise Party:** Every player must give one of their cards to a player of their choosing.
* **The All-Seeing Eye:** Every player has to show their cards. The cards are exposed until every player gives his OK to continue.
* **Third Time Lucky:** Every player has to draw three cards.
* **Time Bomb:** Every player has only three turns left. The round ends when a player would reach his fourth turn. If a player can dispose of all his cards before the fourth turn, he gets credited ten points. The other players get a penalty of ten points. If no one is able to diffuse the bomb, the round is over and the points in this round get doubled.
* **Tornado:** The hand cards of all players are put together, shuffled and one at a time newly distributed by the player of the Black Card.
* **Trust Fall:** Every player has to choose in turn a fellow player who has to give him/her two cards from his hand.
* **Vandalism:** Every player has to dispose of every card (Numeral and Special Cards) of the last played color.

***Important***: If no color has been played so far, the event is ineffective.

# Domain Model

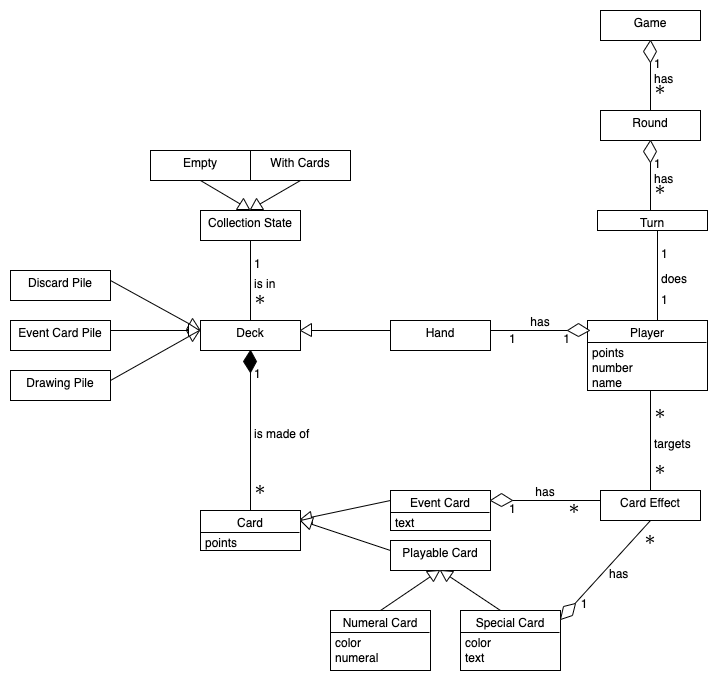


Figure 2: Domain model of the card game Frantic

The Domain Model represents the first step on designing the big and important functionalities. This process displays the first thought flow made by the team designing a card game into a program. The Domain Model includes the basic principles found in a card game, players and cards. Obviously, there can’t only be players and cards, because those terms cover a broad spectrum. For example, a card is specified as a playable card and an event card. Then there is a subcategory of playable cards, special cards and numeral cards. Further details of cards would have to be specified in the game rules and further requirements and don’t really impact this model. Another important factor in the model is the player. The player is a key element in a card game. A player has a hand with cards in it, plays his or her given turn and keeps the game going many rounds until it is finished. Now comes the major factor to connect the player to cards and that is the deck. The player will have to draw cards from the deck to be able to hold hand cards. In addition, the deck would be divided in three categories. A drawing pile, from which the player can draw his or her cards from, a discard pile unto which the player lays down his or her hand cards and an event cards pile. The event card pile is mostly called up when a player activates an event through playing special cards. These cards do not mix with the normal game play, but they effect it for the better or worse. This is the basic thought process we have gone with to design Frantic.

# Software architecture

## Package diagram

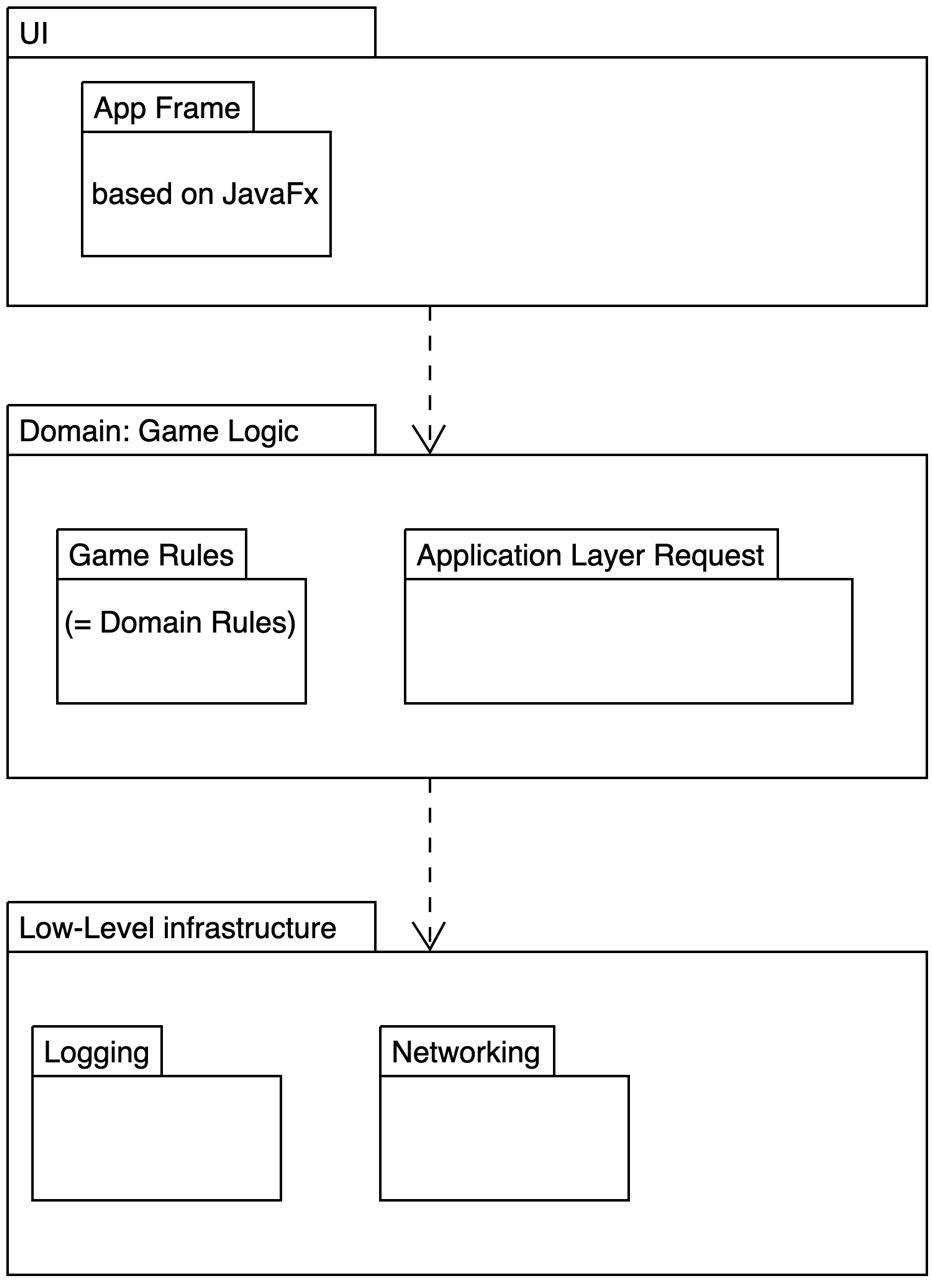
There are three packets containing in our project. UI, Game Logic and low-level infrastructure. The UI packet contains the Game and Menu Interface as well as all the Controller and View classes necessary to build the GUI, which are contained in a subpacket AppFrame. The game logic packet has all classes necessary to run the UI packet such as the interfaces Card and EventCard as well as the classes HandCard, SpecialCard and all the different event cards classes necessary to create all the types of cards which are stored in the Deck and EventDeck class, as well as Player, ComputerPlayer and Game classes positioned in the subpacket Game Rules. The communication between the UI and the game logic is made possible thanks to the other subpacket in Game Logic called Application layer request. The game logic packet also communicates with the low-level infrastructure packet, which contains the logging and networking.

Figure 3: Package diagram of the card game Frantic

## State Machine diagram

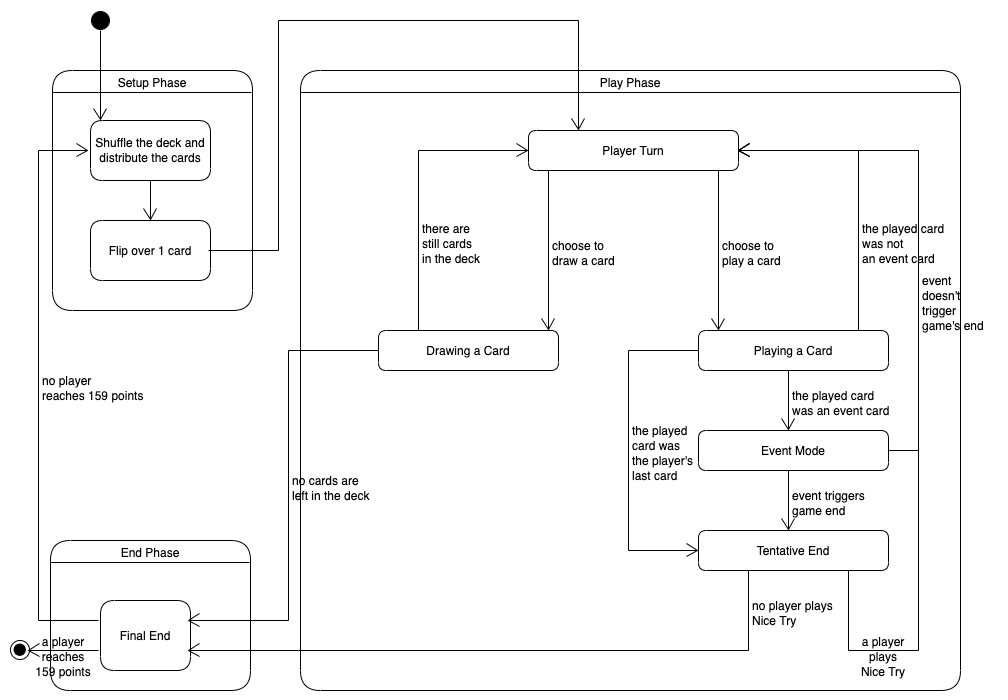


Figure 4: State machine diagram of the card game Frantic

The state diagram represents the general course of the game. It must be said that special functionalities of event and special cards are excluded here. There are three composite state, the first is Setup Phase, the second is Play Phase and the last one is End Phase. The main part is all covered in Play Phase. The inner cycle is based on AI or player decisions. At first weather drawing a card or playing a card. Playing a card may activate an event. After a player finished his turn, the next one is in line. This loop ends if one player has played his last card and nobody plays Nice-Try card. Then a new round is started, and the Setup Phase occurs again. This bigger cycle repeats itself until one player reached over 159 points.

# Design artifacts

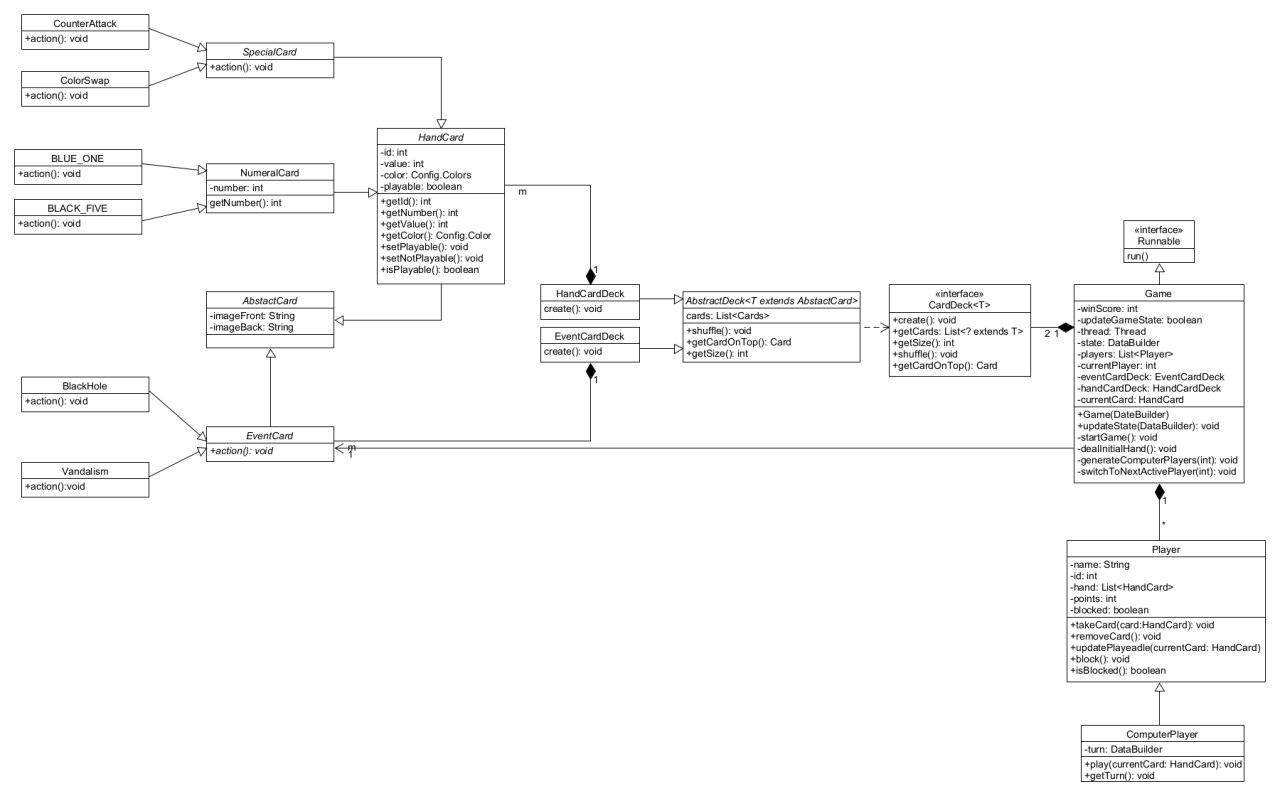


Figure 6: UML class diagram of the card game Frantic, part 1

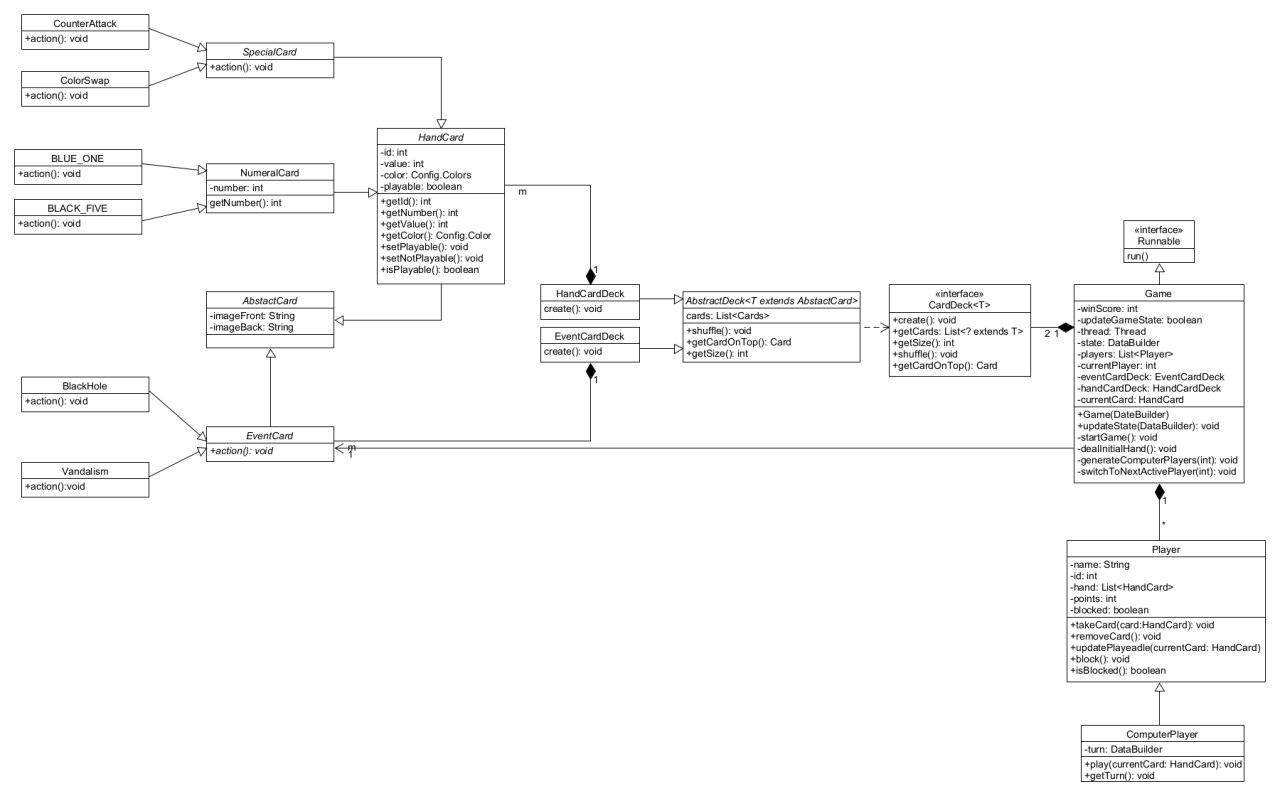


Figure 7: UML class diagram of the card game Frantic, part 2

The class diagram represents our implementation of Frantic. The Game class, which implements the interface Runnable, is the start of the program. It contains the logic of the game, and it’s a storage for all needed information like the players and the deck. Next is the Player class, which is needed to save all information of the player. The two Decks (event and normal) are defined by an Interface and an abstract class. The Cards are made with the Card class, which is based on the AbstractCard class. They are divided into Event Cards, Numeral Cards and Special Cards and all of them have an action which is called when the card is played.

# Implementation

When planning the software architecture for the implementation, not everything was always clear, and we had to decide on certain structures. The decided to choose Java. It is a widely used cross-platform language. Everyone in our team already has Java experience. In this way, the training effort can be reduced, and we can spend more energy on the implementation of the specific product. We decided to use Java sockets for the multiplayer mode. This library offers an extensive range of functions. Communication using Java sockets is relatively low level. A high-level library would have been a large loss of control.

No external server is used in multiplayer mode. A temporary server will be created on the computer that created the game. Other computers on the same network can join this server as clients. With a game in the local network, the error elimination of network problems can be limited.

In single player mode you play against one to seven computer opponents. Machine learning by the computer opponents would be an interesting thing, but not necessary. Even without this method, a strong artificial intelligence can be built up.

The GUI is implemented with JavaFX rather the Canvas. With canvas, animations would be easier to implement but JavaFX offers better and clearer elements for designing and implementing a large project.

[The Code is uploaded to Moodle]

# Project management

Code management: GitHub

Time recording tool: Excel

## Team structure

Project manager: Samuel Stalder

Vice project manager: Lukas Zoss

Team-GUI: Joël Plambeck (head) & Lukas Zoss

Team-Logic: Nikita Smailov (head) & Erman Zankov

Team-AI: Gökhan Bag

Team-Multiplayer: Tobias Ritscher

## Effort estimation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tasks** | **Planned effort per person** | **Planned effort as team** | **Effort** | **SW** |
| Introduction, team building | 5 | 35 | - | 01 |
| Input project management | 5 | 35 | - | 02 |
| Topic development Frantic | 6 | 42 | 40 | 02 |
| Project sketch | 7 | 49 | 40 | 03 |
| Presentation project sketch | 2 | 14 | 10 | 03 |
| Subtask Iteration-Review #2 | 10 | 70 | 66 | 05 |
| Team meeting Iteration-Review #2 and debriefing | 5 | 35 | 35 | 05 |
| Subtask Iteration-Review #3 | 10 | 70 | 89 | 07 |
| Team meeting Iteration-Review #3 and debriefing | 5 | 35 |  | 07 |
| Subtask Iteration-Review #4 | 10 | 70 | 63 | 09 |
| Team meeting Iteration-Review #4 and debriefing | 5 | 35 |  | 09 |
| Subtask Iteration-Review #5 | 10 | 70 |  | 11 |
| Team meeting Iteration-Review #5 and debriefing | 5 | 35 |  | 11 |
| Technical Report | 8 | 56 |  | 13 |
| Presentation M3 | 2 | 14 |  | 13 |
| Beta-Release | 6 | 42 |  | 13 |
| Feedback Beta-Release and technical report II | 1 | 7 | - | 14 |

Table 11: Effort estimation of all tasks during the semester

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subtask Iteration #2 (13.10.2020) | Responsibility | Planned Effort | Effort | Result |
| Draft domain model | Gökhan | 10 | 8 | finished |
| Draft class diagram | Samuel | 10 | 10 | finished |
| Draft Use-Case diagram | Lukas | 8 | 8 | finished |
| Development of the project structure with Gradle JavaFX and test environment | Erman, Nikita | 10 | 10 | finished |
| Draft GUI | Nikita, Joël | 6 | 7 | Partly done. The game menu is not planned yet. |
| Clarification of machine learning | Erman | 4 | 3 | finished |
| Clarification of artificial intelligence | Erman | 6 | 6 | finished |
| Clarification of multiplayer | Tobias | 10 | 9 | Partly done. It still has to be clarified which data are sent from the server to the clients and whether the clients should also contain logic. |
| Setup GitHub | Samuel | 6 | 5 | finished |

Table 12: List of subtasks of Iteration #2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subtask Iteration #3 (27.10.2020) | Responsibility | Planned Effort | Effort | Result |
| Doc: Fully-Dressed-Use-Case | Lukas | 4 | 7 | finished |
| Doc: Definition of further requirements | Lukas | 4 | 5 | finished |
| Doc: Domain model | Gökhan | 4 | 4 | finished |
| Doc: Software architecture | Gökhan, Samuel, Tobias | 5 | 9 | finished |
| Doc: State diagram: game logic | Samuel | 4 | 5 | finished |
| Doc: UML-Package diagram | Gökhan | 3 | 4 | finished |
| Doc: System sequence diagram: Play Thief Scenario | Tobias | 5 | 5 | finished |
| Doc: Design artifacts: UML-Class diagram | Gökhan | 5 | 5 | finished |
| Doc: Documentation of Implementation | Samuel | 4 | 5 | finished |
| Doc: Documentation of project management | Samuel | 5 | 5 | finished |
| Doc: Glossary | Gökhan | 3 | 4 | finished |
| Code: Implementation of the class diagram without logic | Nikita, Erman | 7 | 7 | finished |
| Code: Simple game play in single player mode against a computer-controlled opponent who discards random cards. Special cards and event cards have no effects | Nikita, Erman | 7 | 9 | started |
| GUI: FXML file for displaying the playground | Joël, Lukas | 3 | 4 | finished |
| GUI: FXML file for displaying the menu | Joël, Lukas | 2 | 3 | finished |
| Implementation of MVC-Patterns | Joël, Lukas | 2 | 2 | finished |
| Multiplayer: Definition Protocol | Tobias | 2 | 4 | finished |
| Multiplayer: Definition of the transferred data | Tobias | 1 | 2 | started |

Table 13: List of subtasks of Iteration #3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subtask Iteration #4 (10.11.2020) | Responsibility | Planned Effort | Effort | Result |
| Code: Implementation of the game sequence according to the state diagram | Nikita, Erman | 10 | 9 | started |
| Code: Implementation of the first special card | Nikita, Erman, Joël, Lukas | 6 | 5 | finished |
| Code: Implementation of the first event card | Nikita, Erman, Joël, Lukas | 6 | 5 | finished |
| Code: Unit tests for game sequence | Nikita, Erman | 3 | 3 | started |
| Code: Implementation of Logger | Tobias | 4 | 2 | finished |
| Doc: adjustments | Samuel, Gökhan | 10 | 7 | finished |
| External test project of the server-client communication | Tobias | 9 | 9 | finished |
| Interface definition of the protocol | Tobias | 5 | 7 | finished |
| GUI: display of the cards | Joël, Lukas | 5 | 4 | Finished |
| Clear interface definition between logic, GUI and server | Tobias | 6 | 8 | Finished |
| AI: Implementation of a simple computer opponent | Erman | 6 | 4 | started |

Table 14: List of subtasks of Iteration #4

Result: Not all tasks could be finishes. The Definition of the interface took more time then estimated and blocked the workflow of the other subtasks. The Implementation of the game sequence and their unit tests are processed in the next Iteration.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subtask Iteration #5 (24.11.2020) | Responsibility | Planned Effort | Effort | Result |
| Code: Implementation of the game sequence according to the state diagram | Nikita, Erman | 10 |  |  |
| Code: Unit tests for game sequence | Nikita, Erman | 4 |  |  |
| Code: Implement more special cards | Nikita, Erman | 5 |  |  |
| Code: Implement more event cards | Nikita, Erman | 5 |  |  |
| Code: Implementation of a more advanced computer opponent | Gökhan | 3 |  |  |
| Code: Unit tests for computer opponent | Gökhan | 3 |  |  |
| Doc: Interaction diagram | Gökhan | 3 |  |  |
| Doc: adjustments | Samuel | 7 |  |  |
| Integration of the external test project | Tobias | 6 |  |  |
| Definition of Server functions | Tobias | 3 |  |  |
| Modification GUI-Controller | Joël, Lukas | 7 |  |  |
| Definition of dataPackage | Joël, Lukas | 4 |  |  |
| Fully covered game sequence which interacts with Logic, Server, Client and GUI | All | 10 |  |  |

## Risks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Impact** | high | Single player-Mode |  | Multiplayer-Modus |
|  | medium | GUI with JavaFX | Game complexity  Artificial Intelligence |  |
|  | low |  |  | Loss of staff (Corona) |
|  |  | low | medium | high |
|  |  |  |  | **Probability of occurrence** |

Table 15: Risk matrix of the project

What has changed:

* Thanks to GitHub and Microsoft-Teams, the effects of a staff failure during the corona pandemic are manageable. Working from home is easy.
* By now all team members know the game. Thus, the likelihood of a misunderstanding decreases.

# Glossary

|  |  |
| --- | --- |
| Word | Meaning |
| Deck | Mixture of all drawable hand cards |
| Hand | Every player has a list of cards on his hand. Game ends if hand is empty |
| Discard Pile | All played drawable hand cards get collected in the discard Pile |
| Drawing Pile | Is a subset of deck |
| Event Card Pile | Set of all event card |
| Card | Playable object which can activated. It may activate an event or a special function |
| Player | Human being who is aware of this card game |
| Turn | The next Player active if the previous player finished his turn |
| Round | A round is over if one player has an empty hand |
| Card Target | The functionality of event or special card can target one or multiple players |
| Event Card | An event card is a passive card which can be activated during game. See 2.1 for a description of each event card. |
| Numeral Card | A numeral card is a hand card which can be thrown on the discard pile. It has no special effects. |
| Special Card | A special card is a hand card which can be thrown on the discard pile. It activates a special effect. See 2.1 for a description of each special card. |
| JavaFX | The framework we use for creating the interactive user interface of Frantic. |

Table 16: Glossary of common Frantic words

List of figures

[Figure 1: Use-Case-Model of Frantic 2](#_Toc54693598)

[Figure 2: Domain model of the card game Frantic 14](#_Toc54693599)

[Figure 3: Package diagram of the card game Frantic 15](file:////Users/ritscher/Desktop/3.%20Sem/PM3/Technischer%20Bericht%201/PM3-HS20-IT19a_ZH-Technischer_Bericht_I-Team03.docx#_Toc54693600)

[Figure 4: State machine diagram of the card game Frantic 16](#_Toc54693601)

[Figure 5: Flowchart, Play Thief Scenario of the card game Frantic 17](#_Toc54693602)

[Figure 6: UML class diagram of the card game Frantic, part 1 19](#_Toc54693603)

[Figure 7: UML class diagram of the card game Frantic, part 2 19](#_Toc54693604)

[Figure 8: Screenshot of the main menu 21](#_Toc54693605)

[Figure 9: Screenshot of the main GUI 22](#_Toc54693606)

List of tables

[Table 1: Use-Case description for set game 3](#_Toc54693607)

[Table 2: Use-Case description for setup Artificial Intelligence 4](#_Toc54693608)

[Table 3: Use-Case description for join open game 4](#_Toc54693609)

[Table 4: Use-Case description for start round 5](#_Toc54693610)

[Table 5: Use-Case description for play card 6](#_Toc54693611)

[Table 6: Use-Case description for react to event 7](#_Toc54693612)

[Table 7: Use-Case description for next turn 7](#_Toc54693613)

[Table 8: Use-Case description for create new round 8](#_Toc54693614)

[Table 9: Use-Case description for end game 9](#_Toc54693615)

[Table 10: Use-Case description for disconnect form game 9](#_Toc54693616)

[Table 11: Effort estimation of all tasks during the semester 23](#_Toc54693617)

[Table 12: List of subtasks of Iteration #2 24](#_Toc54693618)

[Table 13: List of subtasks of Iteration #3 24](#_Toc54693619)

[Table 14: List of subtasks of Iteration #4 25](#_Toc54693620)

[Table 15: risk matrix of the project 25](#_Toc54693621)

[Table 16: Glossary of common Frantic words 26](#_Toc54693622)