



University
of Glasgow

MSC IT+ MASTERS TEAM PROJECT
TEAM NAME: End Game

2414816H, Junhao Huang
2431298Y, Guodong Yang
2431374H, Junjia Huang
2485966J, Jiazhao Yan
2431051J, Boren Jing

Table of contents

| | |
|--------------------------------------|-----------|
| Introduction | 1 |
| Method | 1 |
| MVC Diagrams | 2 |
| Model UML | 5 |
| Users | 5 |
| User Stories | 6 |
| Story Cards | 7 |
| Development Sprints | 13 |
| Burndown Chart | 14 |
| Website Design | 15 |
| Assumptions | 20 |
| Testing | 20 |
| Deficiencies | 22 |
| Screenshots | 23 |
| Conclusion and Recommendation | 25 |
| Appendix | 26 |

Introduction

The following PDF document is an overview report of Our “Top Trumps” Team project, the report consists of the current status of our program, web design, development sprints, testing, etc.

The following applications and tools were used in our team project:

1. Github, offers the distributed version control and several collaboration features, allowing every team member to participate in the project.
2. WeChat, for team communication and individual work and distribution.
3. Visio, for complete burndown charts and MVC diagrams.
4. Google Drive, for documentation work and final project report.
5. VSCode, for main project testing and modification, also for personal log record and upload.
6. Eclipse, for command-line version and online version of the project building and testing.

Method

This project will follow the Scrum Agile framework started on Jan 15, 2020, and ended on Feb 17, 2020. There were two Sprints. Sprint 1 focused on completing the command line mode, and Sprint 2 was to design the online mode.

The Database in this project used IP address: 52.24.215.108 which is held by Amazon. However, the connection was slow sometimes due to network conditions. This means that the statistic screen might be slowly loaded. The database structure has shown in Figure 1.

The statistical data will use PostgreSQL language to calculate the average of the game, longest game, etc.

In the table, the column of game-winner has value 1 and 0. value 1 means the user wins the game, and 0 means AI achieves the 40 cards finally.

In Sprint 1, the main technology of achieving command line mode was JAVA. It used an MVC structure to manage the game. In details, the Model_GameManager was the main model which can make communication with the controller.

In Sprint 2, it used knowledge of JAVA, AJAX, JAVAScript, HTML and CSS. HTML and CSS were used to show the interface of the game. JAVAScript and AJAX were to send the request to the backend and make a response to the frontend.

MVC Diagrams

- **CommandLine Mode:**

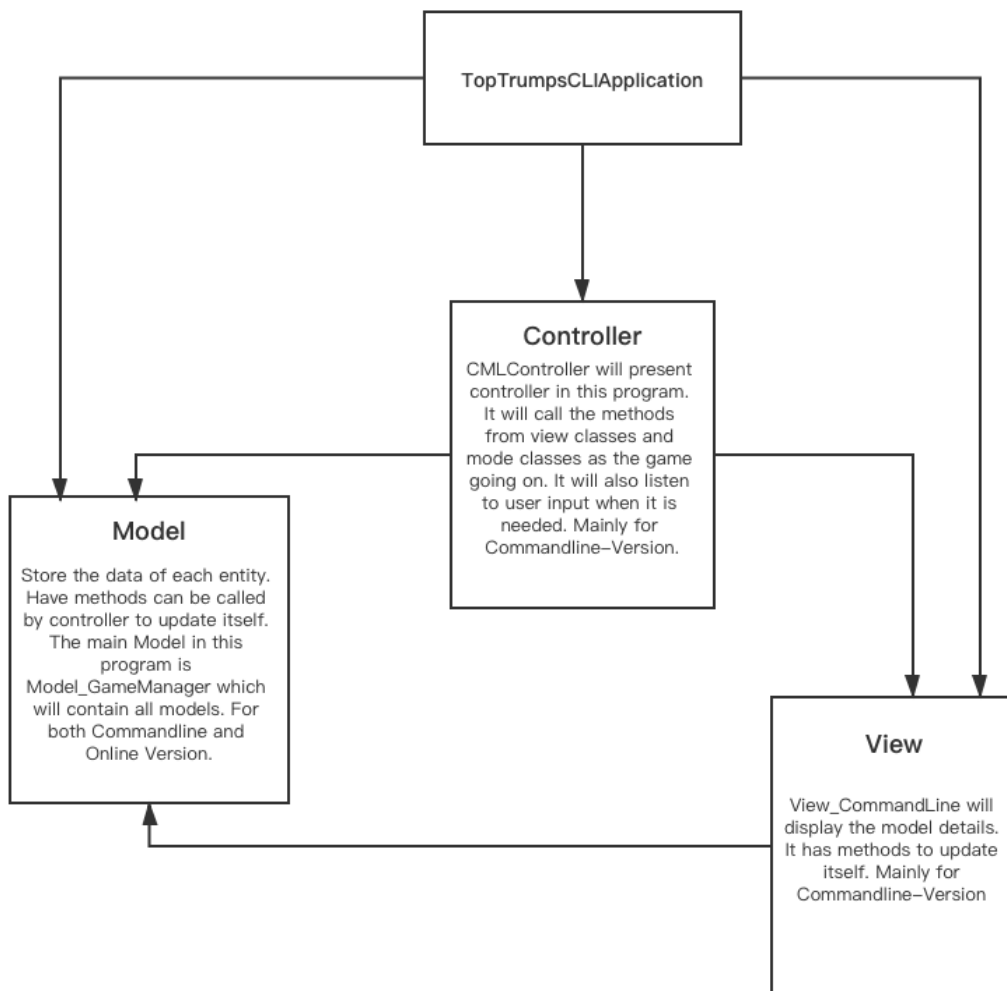


fig 1. MVC diagrams for command line Mode

- **Online Mode:**

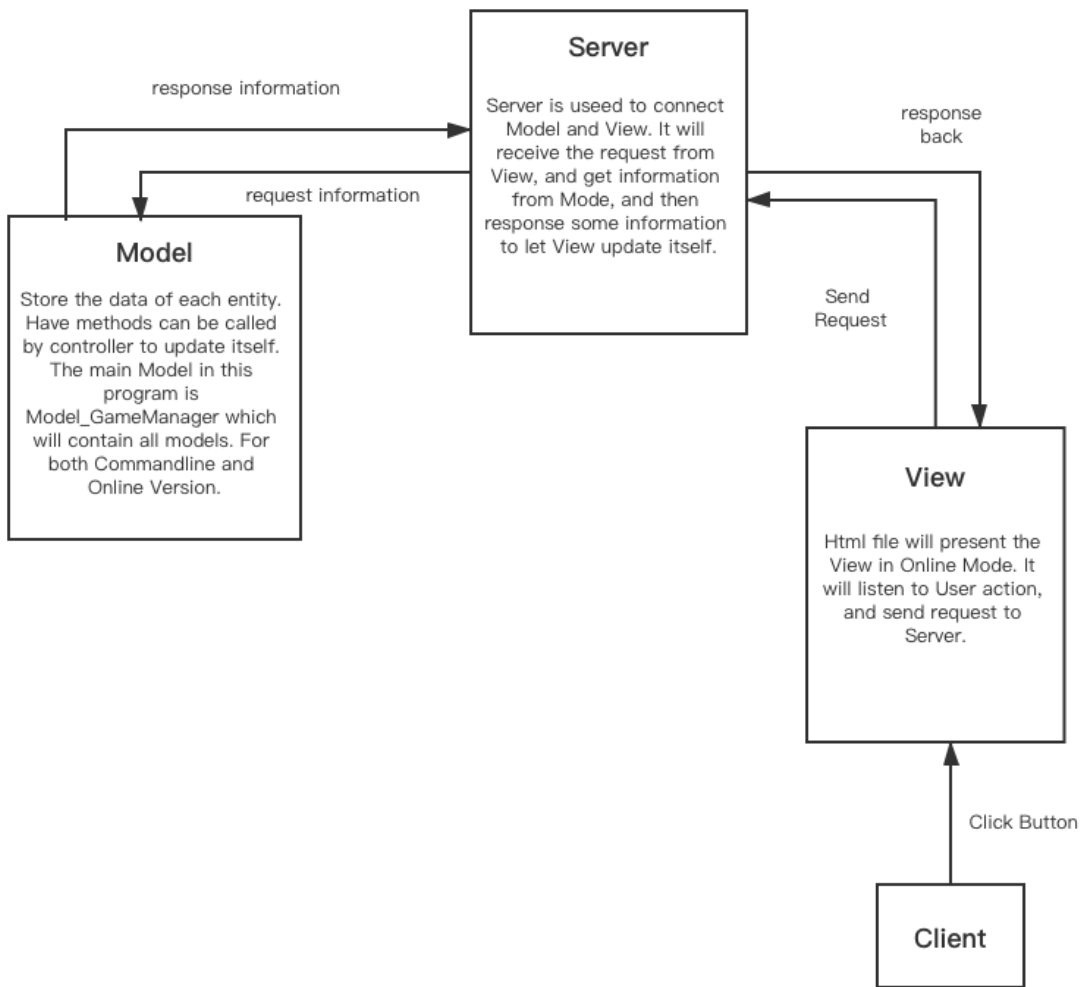


fig 2. MVC diagrams for online Mode

Model UML

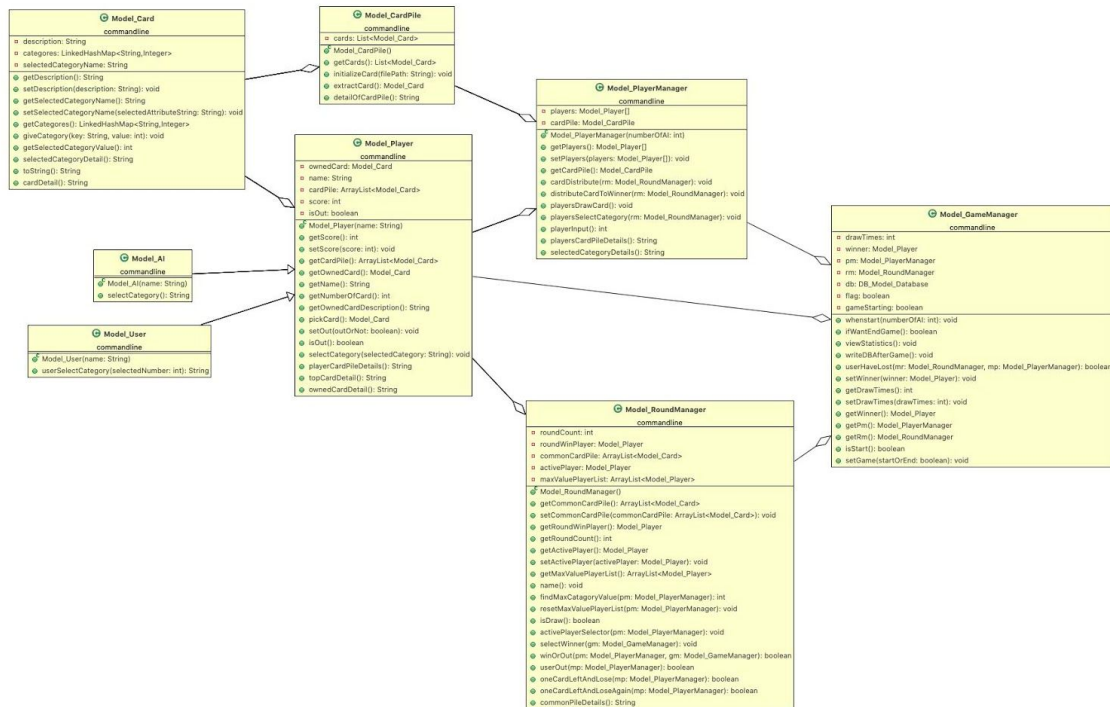


fig 3. Model UML

Users

The users in these projects might be game players, program developers, customers, and card image painters. Game players will focus on the experience during the game. It is necessary to they are likely to exit the game each time. Program developers will consider the bugs, so it is important to fix the bugs. Customers might be interested in the game data. They might give some suggestion that how the game will go into the market. Card painters might not care about who wins the game, so the images should show correctly at each round.

User Stories

The user stories were beaked down by several points:

Must-have

- The user must play the game with the command-line version.

Estimated costs: 20 points

Actual costs: 23 points

(The command line version costs more effort than expected, which mainly results in the card ArrayList built and database knowledge review.)

- The user must play the game with the online version.

Estimated costs: 18 points

Actual costs: 18 points

(The online version costs significant effort as expected which mainly because of the use while learning how to use JavaScript and link it to the command-line version.)

Should-have

- The user should be able to access the scoreboard when one round of the game was over.

Estimated costs: 8 points

Actual costs: 7 points

(The scoreboard should record the data was generated from the game every round and appear while the game was finished)

- The user should be able to quit the game at any time and both the command-line version and online version of the game.

Estimated costs: 8 points

Actual costs: 8 points

(The 'Quit' function mainly activated by 'keywords' or bottom according to whether the user is playing the game on the command-line version or the online version.)

Story Cards not implemented

- The user could mute the background music if they wish to.
- The Manual Mode allows the user to control AI's selection.
- The Developer Mode allows the user to modify cards attribute value by user-interface.

Story Cards

#1

Front Side:

- **Story Name: Command line version playable**
"As a player, I could play the game on the command line."
- **Sprint: 1**
- **Priority: Must**
- **Estimated Cost: 20 Points**
- **Actual Cost: 23 Points**

Back Side:

- **The game function could run properly including start and exit.**
- **The game could run on any environment with setup version.**

#2

Front Side:

- **Story Name: The number of AI player**
"As a player, I could choose how many players in one game"
- **Sprint: 1**
- **Priority: Should**
- **Estimated Cost: 11 Points**
- **Actual Cost: 9 Points**

Back Side:

- **The player number could be chosen at the begging of the game.**
- **Relevant log function should be updated according to the actual player number.**

#3

Front Side:

- **Story Name: Scoreboard**
"As a player, I wish to see the scoreboard after the game finished."
- **Sprint: 1**
- **Priority: Should**
- **Estimated Cost: 8 Points**
- **Actual Cost: 7 Points**

Back Side:

- The scoreboard should appear after the player was winning or losing.
- The scoreboard should record the data that was generated every round.
- The data storage should be in order by the timeline.

#4

Front Side:

- Story Name: Quit the game
“As a player, I wish to quit the game at any time I would like to.”
- Sprint: 1
- Priority: Must
- Estimated Cost: 7 Points
- Actual Cost: 8 Points

Back Side:

- Specific 'quit' key should be activated at any time if the user wants to terminate the game.
- The instruction of quit game hotkey should be shown at the begging of the game interface.

#5

Front Side:

- **Story Name: Online version**
“As a player, I wish to play this game on the web page.”

- **Sprint: 2**
- **Priority: Must**
- **Estimated Cost: 18 Points**
- **Actual Cost: 18 Points**

Back Side:

- **Relevant JavaScript and Database should be linked.**
- **The interface of the web page for a human player should be clear and easy to use.**

#6

Front Side:

- **Story Name: Web page operation**
“As a player, I wish to play online version game easily by clicking the bottom.”
- **Sprint: 2**
- **Priority: Must**
- **Estimated Cost: 10 Points**
- **Actual Cost: 9 Points**

Back Side:

- **The function of the command line version should be activated by tracking the action of the mouse.**

- The interface should give the feedback of player mouse action including the error warning.

#7

Front Side:

- Story Name: Web page result

“As a player, I wish to see the game result of the online version.”

- Sprint: 2
- Priority: Should
- Estimated Cost: 8 Points
- Actual Cost: 7 Points

Back Side:

- The web page interface should be showing a window of the result of the game after finished.
- The storage of game data should be pulled out once the game finished.

#8

Front Side:

- Story Name: Card picture

“As a player, I wish to have a unique card image.”

- Sprint: 2
- Priority: Should

- **Estimated Cost: 10 Points**
- **Actual Cost: 11 Points**

Back Side:

- **Card image should have an individual storage database linked to the card ArrayList.**
- **Each unique card image should link to the corresponding card (card name).**

#9

Front Side:

- **Story Name: Web game quit and restart**
“As a player, I wish to quit the game and restart at any time.”
- **Sprint: 2**
- **Priority: Must**
- **Estimated Cost: 8 Points**
- **Actual Cost: 8 Points**

Back Side:

- **Online version game should have a 'Quit' bottom which linked a 'break' function can quit the game at any time.**

Development Sprints

| Sprint | Date | Sprint Backlog |
|---------|-------------------------|---|
| Sprint1 | 16/01/2020 - 02/02/2020 | <ol style="list-style-type: none"> 1. Command Line Model 2. The number of players 3. Get game statistics 4. Quit the game |
| Sprint2 | 02/02/2020 - 16/02/2020 | <ol style="list-style-type: none"> 1. Online version 2. User control 3. Access to game statistics 4. Add card image 5. Quit and restart web game |

Table 1. Development sprints 1.

| Sprint | Top Trumps Game | | |
|---------|-----------------|--------|-------|
| Sprint1 | Day | Remain | Ideal |
| | 1 | 100 | 100 |
| | 11 | 80 | 80 |
| | 18 | 70 | 60 |
| Sprint2 | 25 | 60 | 40 |
| | 32 | 18 | 20 |
| | 33 | 0 | 0 |

Table 2. Development sprints 2.

Burndown Chart

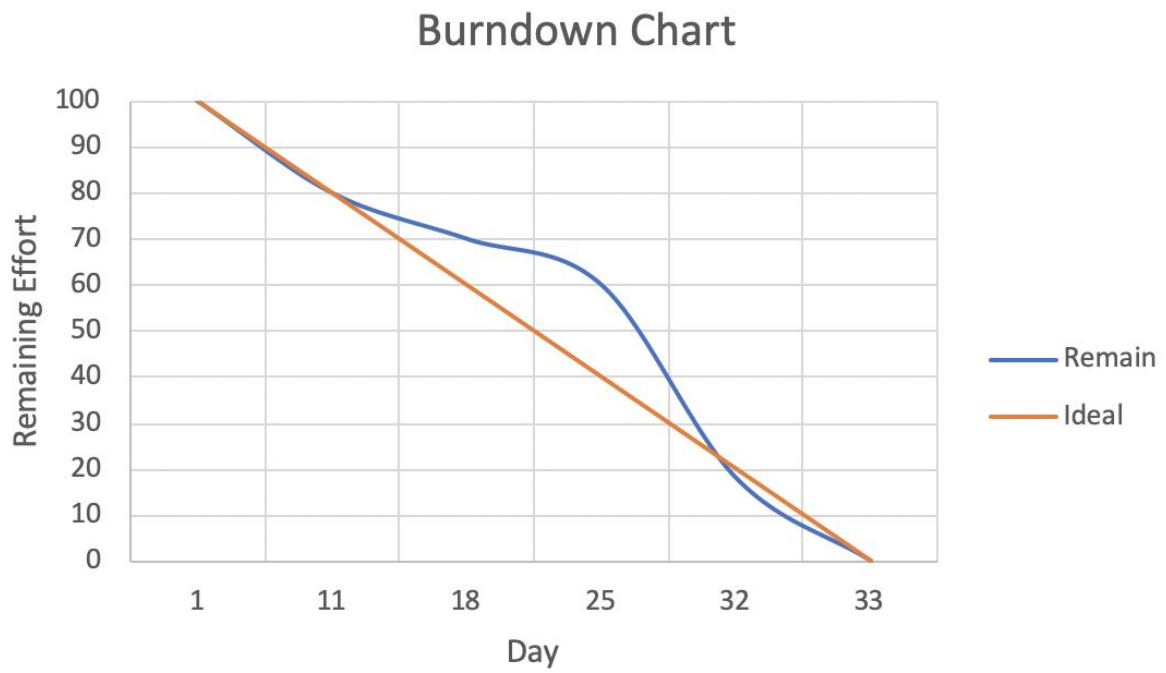


fig 4. Burndown Chart

Website Design

The Selection Screen page, also the home page, consists of the main title and two buttons. The main page features a simple design and a friendly UI. Meanwhile, the plane in the background highlights the theme of the game. (The main page design is implemented through CSS and HTML)

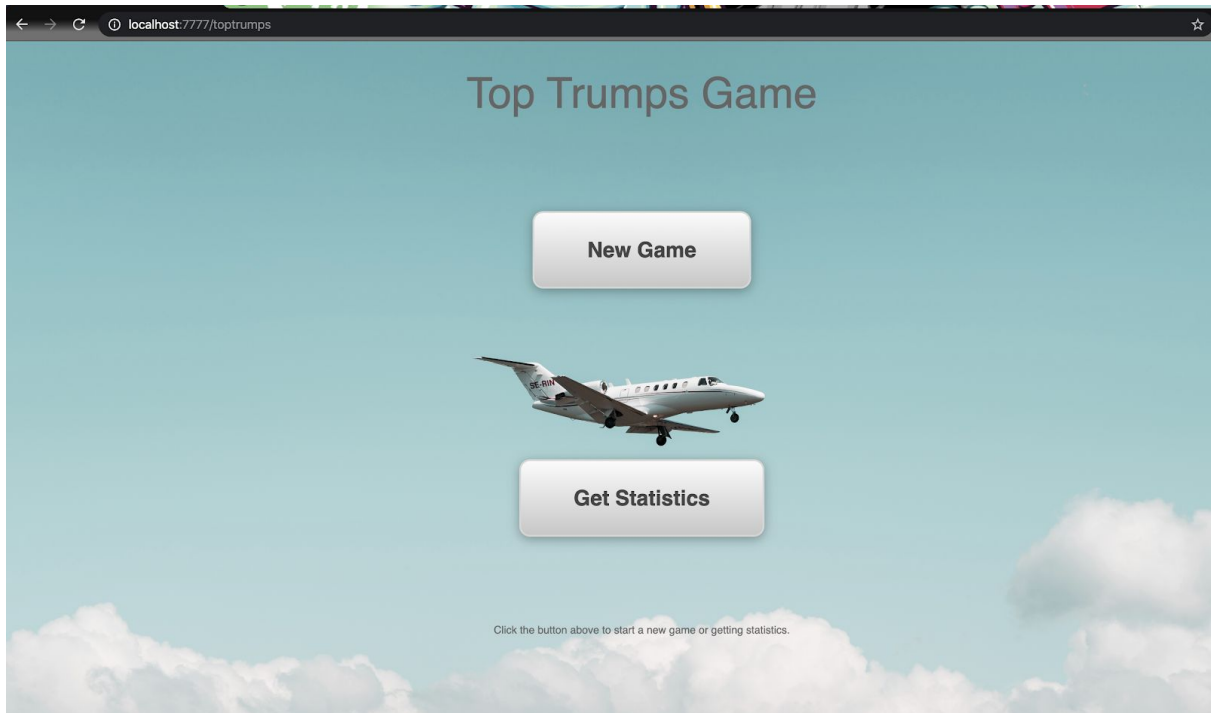


fig 5. Home Page

The Game Screen page also follows the theme design of the selection page, according to the game order to display the page design. First of all, Click the new game button on the home page and enter the game page. Moreover, if the user chooses the EXIT button which will interrupt the game. The game will display the player's card (Including the hover function), the number of rounds and the active Player (The red highlighted text) when the game is initialized. The contents of the card include the name of the card, the number of cards left in the player's deck, and the corresponding attributes of the card.

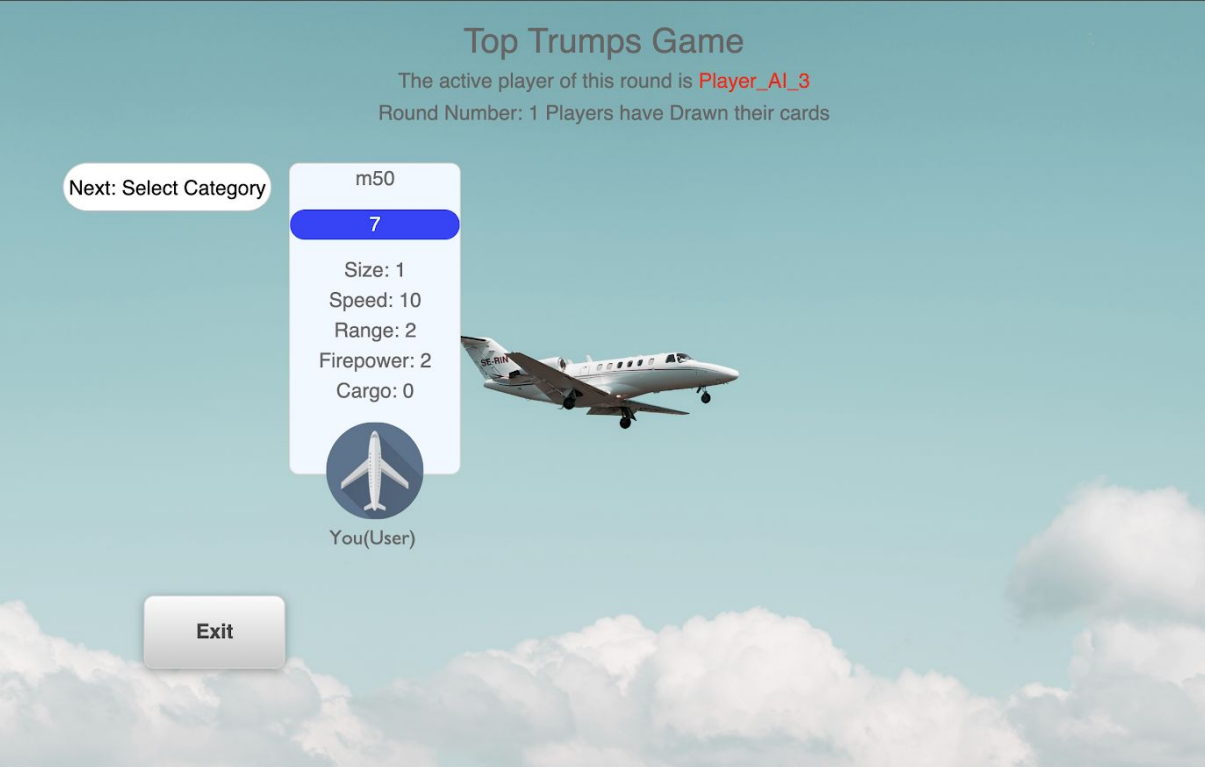


fig 5. In-game screen 1

Moreover, in the game comparison screen, players can visually judge the attribute comparison selected for this round by the above text. The active player card is highlighted in yellow.

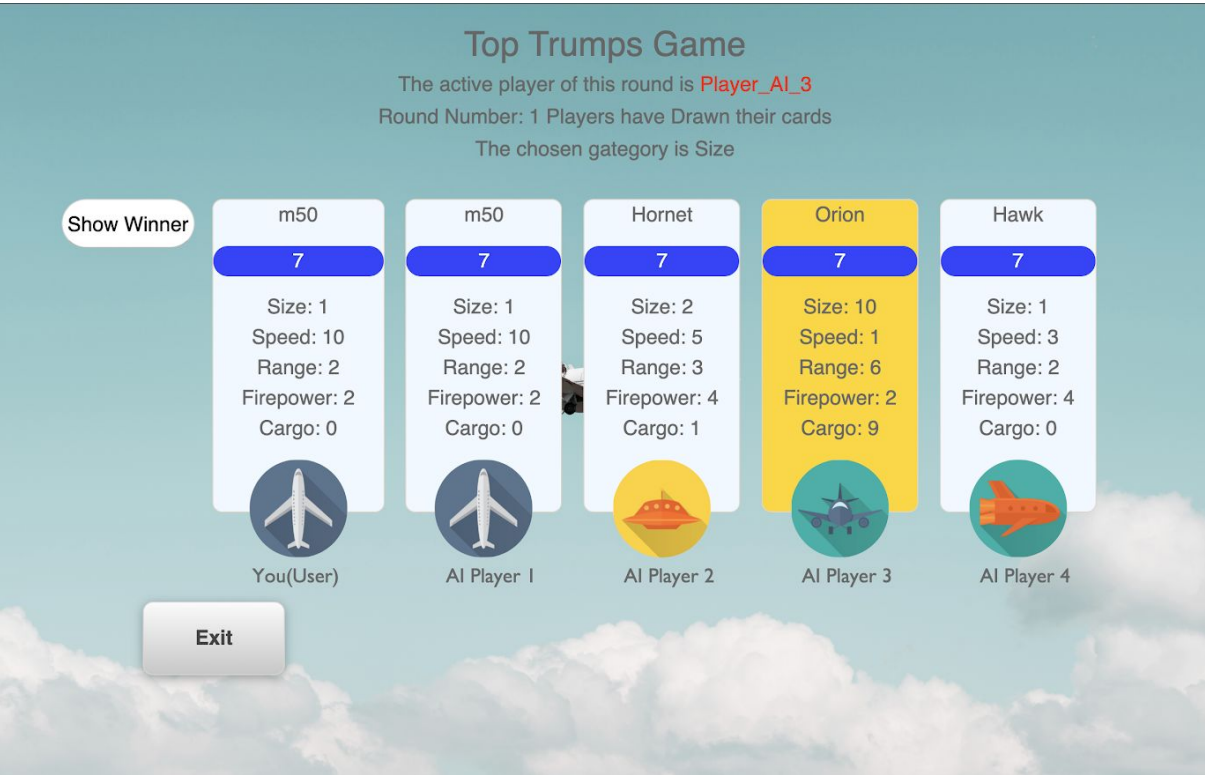


fig 6. In-game screen 2

The next page will show the winner result per round the winner of this round can be seen in the show results screen (The yellow highlighted text).

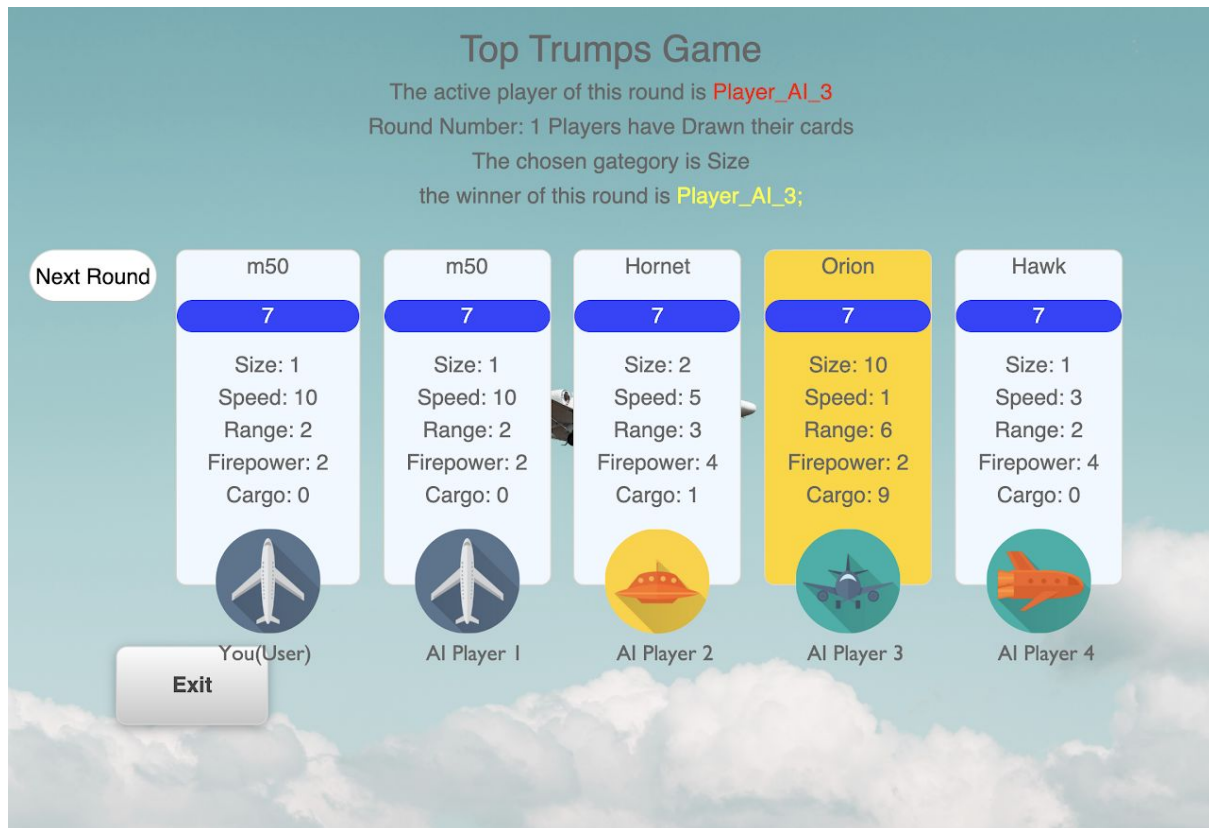


fig 7. Results screen

In addition, the category button is displayed to remind the user to make a choice only when the user is active. The category button is set to hide when the AI is active.

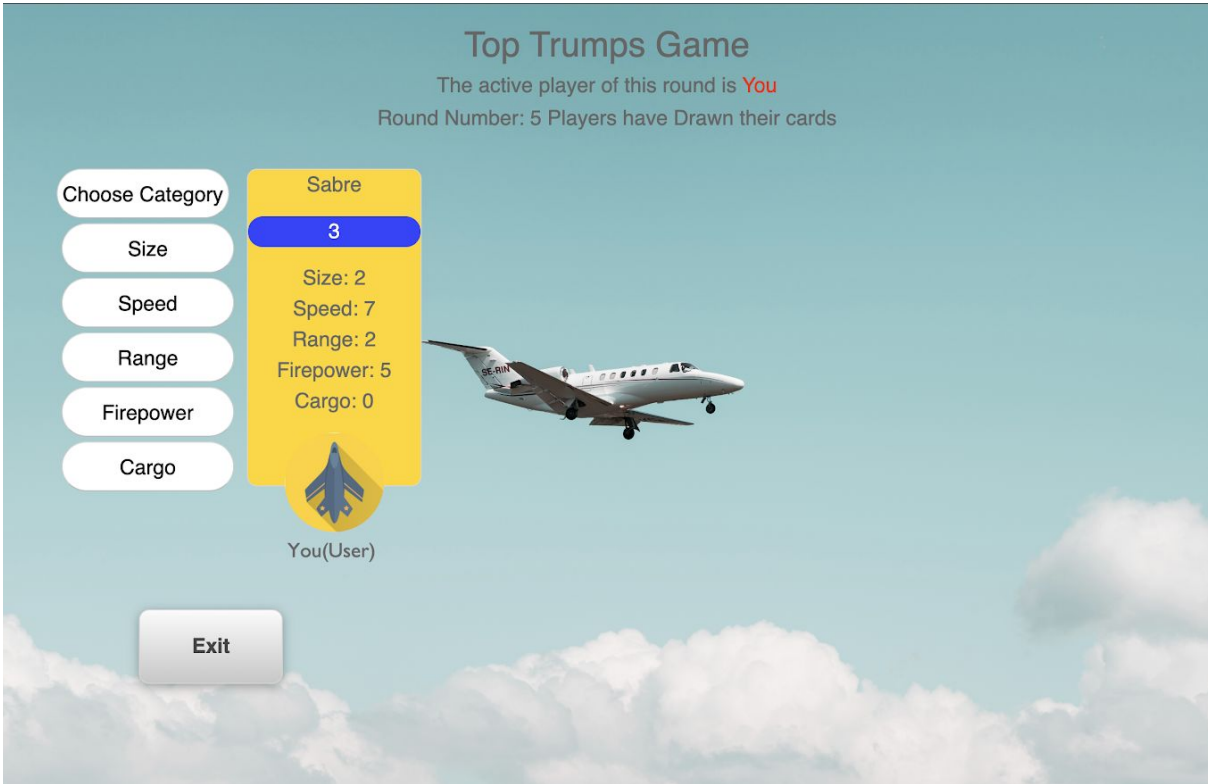


fig 8. In-game screen 3

The Final game-winner result is shown as follows. The winner card displays the final winner and each person win score.

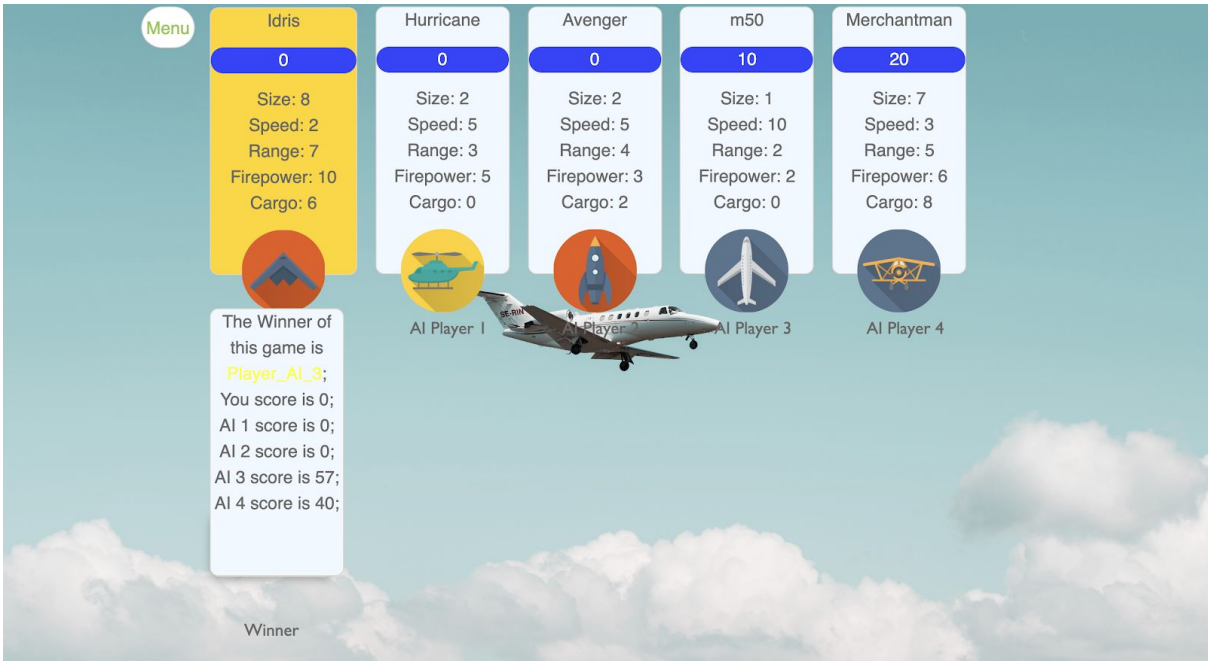
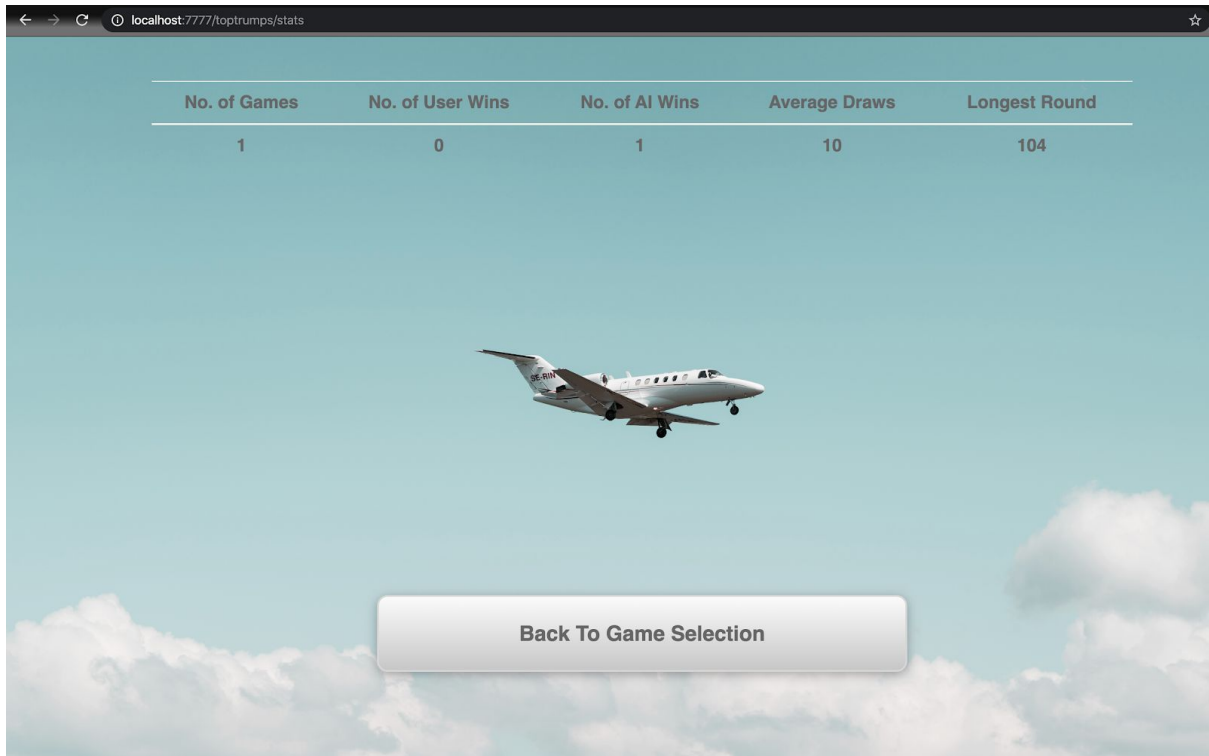


fig 9. Final game-winner result

The Statistics page is Statistics Screen page. This page displays game-related statistics based on past game records through tables.



| No. of Games | No. of User Wins | No. of AI Wins | Average Draws | Longest Round |
|--------------|------------------|----------------|---------------|---------------|
| 1 | 0 | 1 | 10 | 104 |

Back To Game Selection

fig 10. Statistics screen page

Assumptions

Basically, the project application can run normally based on the standard situation. However, there are some bugs or problems still not found in the previous program test. Hence, this part will divide two sections, including Command-Line mode and Online mode, to propose few assumptions of issues raised when the project operating.

In the command-line section, there may be a problem with AI players continue gaming which has too many random turns if the user is out, which will lead to data redundancy problems and troubles of operation. Hence, the ai player is set to play the game automatically after the user is eliminated

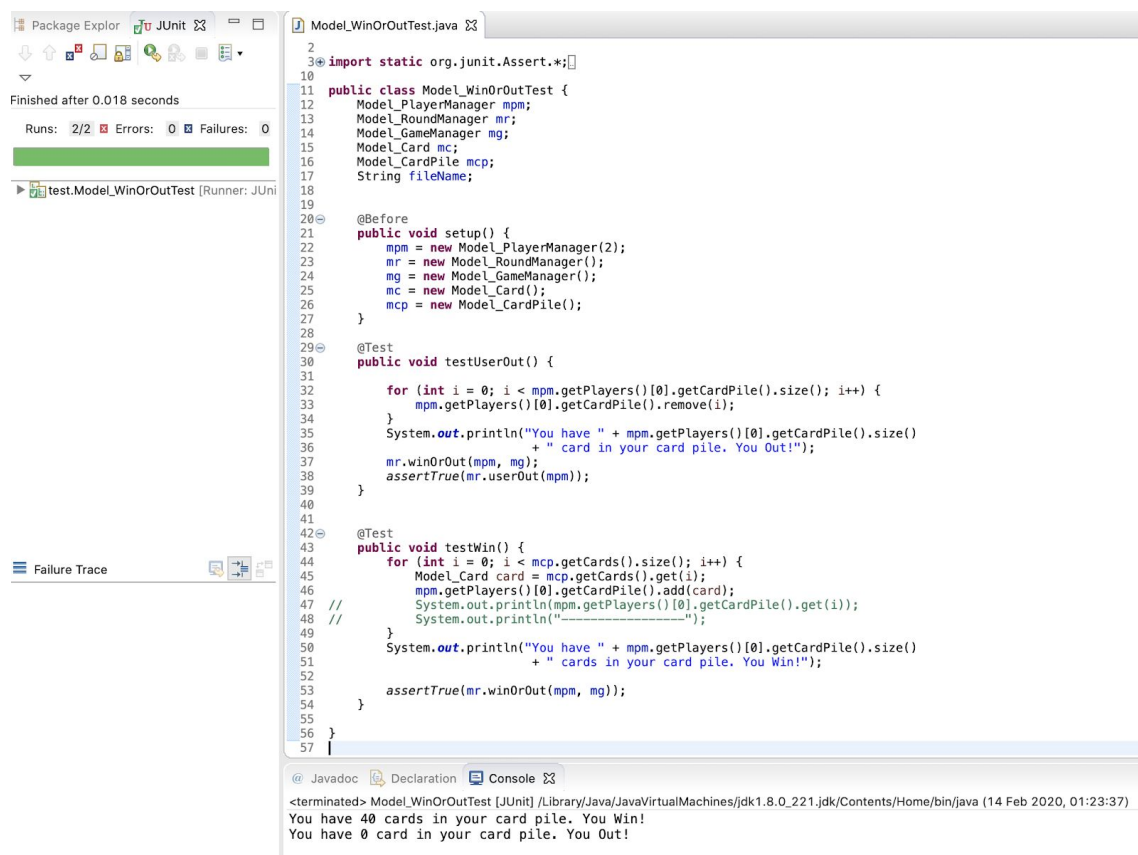
In the online mode, there is an assumption that the game might run on different browsers while different players were playing, so we needed to make sure that it worked on different browsers or different devices.

Testing

Testing is the process of checking an application's functionality to ensure that it operates as required. Considering the programming language, Java, used in the Command-Line-Model, this project adopts Junit for testing models. The examples of testUserOut and testWin are shown in Fig-11 Model_WinOrOutTest and the Model_RoundManager class is demonstrated in Fig-12.

The function of winOrOut in Model_RoundManager class is to check the number of cards in player[0] card pile (the human player by default) whether is equal to 0 or 40. If it is 0, the player will be out. Instead, if there are 40 cards in the player's card pile, he will win. Another method called userOut in Model_RoundManager class is used to return a boolean value through an if statement, based on a boolean type method named isOut. The result of which is passed by the same type method, setOut, in winOrOut method.

In testUserOut method, the human player's card pile was cleared and the winOrOut function was called afterwards. Then, assert that the result of isOut function would be true. As the Junit tested, it was true and the player was out. The principle of testWin function is the opposite of testUserOut function. In order to win the game, the player should hold 40 cards in his card pile. Thus, the user's card pile added 40 cards in it and we asserted that the result of winOrOut function would return true, which means the human player won. Also, according to the result of Junit test, it was true and the player won the game.



```

2
3 import static org.junit.Assert.*;
10
11 public class Model_WinOrOutTest {
12     Model_PlayerManager mpm;
13     Model_RoundManager mr;
14     Model_GameManager mg;
15     Model_Card mc;
16     Model_CardPile mcp;
17     String fileName;
18
19
20 @Before
21 public void setup() {
22     mpm = new Model_PlayerManager(2);
23     mr = new Model_RoundManager();
24     mg = new Model_GameManager();
25     mc = new Model_Card();
26     mcp = new Model_CardPile();
27 }
28
29 @Test
30 public void testUserOut() {
31
32     for (int i = 0; i < mpm.getPlayers()[0].getCardPile().size(); i++) {
33         mpm.getPlayers()[0].getCardPile().remove(i);
34     }
35     System.out.println("You have " + mpm.getPlayers()[0].getCardPile().size()
36         + " card in your card pile. You Out!");
37     mr.winOrOut(mpm, mg);
38     assertTrue(mr.userOut(mpm));
39 }
40
41
42 @Test
43 public void testWin() {
44     for (int i = 0; i < mcp.getCards().size(); i++) {
45         Model_Card card = mcp.getCards().get(i);
46         mpm.getPlayers()[0].getCardPile().add(card);
47         System.out.println(mpm.getPlayers()[0].getCardPile().get(i));
48         // System.out.println("-----");
49     }
50     System.out.println("You have " + mpm.getPlayers()[0].getCardPile().size()
51         + " cards in your card pile. You Win!");
52     assertTrue(mr.winOrOut(mpm, mg));
53 }
54 }
55
56 }
57

```

Package Explorer JUnit

Finished after 0.018 seconds

Runs: 2/2 Errors: 0 Failures: 0

test.Model_WinOrOutTest [Runner: JUnit]

Failure Trace

@ Javadoc Declaration Console

<terminated> Model_WinOrOutTest [JUnit] /Library/Java/JavaVirtualMachines/jdk1.8.0_221.jdk/Contents/Home/bin/java (14 Feb 2020, 01:23:37)

You have 40 cards in your card pile. You Win!

You have 0 card in your card pile. You Out!

Fig 11. Model_WinOrOutTest.


```

190     }
191
192
193
194
195     public boolean winOrOut(Model_PlayerManager pm, Model_GameManager gm) {
196         /*
197          * If a player has 40 cards on his pile, the he will win the game.
198          * Game over, recording game information.
199          * If a player got 0 cards on his pile, then he is out.
200          */
201         for (int i = 0; i < pm.getPlayers().length; i++) {
202             if (pm.getPlayers()[i].getCardPile().size() == 0) {
203                 pm.getPlayers()[i].setOut(true);
204             }
205             if (pm.getPlayers()[i].getCardPile().size() == 40) {
206                 gm.setWinner(pm.getPlayers()[i]);
207                 return true; //this means this game will end, and User win the game
208             }
209             endGame();
210         }
211         // if (i != 0 && players[i].getCardPile().size() == 40) {
212         //     endGame();
213         // }
214         roundCount++;
215         return false;
216     }
217
218
219
220
221
222
223
224     // These three methods below are used to judge the user will be lose game or not
225     // They will be actually used in Model_GameManager
226
227     public boolean userOut(Model_PlayerManager mp) {
228         if (mp.getPlayers()[0].isOut()) {
229             return true;
230         } else {
231             return false;
232         }
233     }
234

```

Fig 12. Model_RoundManager

Deficiencies

The deficiencies section is mainly described in two modes, Online mode and Command-line mode.

In the Command-line mode, The whole program is implemented according to the MVC architecture, but the classes of the command line part are all set in the command line package, including model, view and control (distinguished by name). There may be some difficulty in identifying which classes the different functions are in.

In the Online mode, there are few deficiencies including the UI design of Game-Screen page and implement of the Javascript function. There is a small design problem in the UI of the game interface, that is, the position of the clickable button will be changed a little after each function is completed, which may not be convenient for users to operate. While there are many modules to implement javascript functionality, there may be multiple ways to implement a module. One is the possibility of code redundancy, and the other is multiple requests to the back end Java class (REST-API) section for return values.

Screenshots

Command-line application

```
Do you want to see past results of play a game?
1. Print Game Statistics
2. Play game
3. Exit game
Enter the number for your selection: 2

|
How many AI do you want to play with?
4

Game Start
Round 1
Round 1: Players have drawn their cards
You drew 'Hornet' :
> Size: 2
> Speed: 5
> Range: 3
> Firepower: 4
> Cargo: 1

There are '7 cards in your deck
Round 1: Player Player_AI_3 won this round
The winner card was 'm50':
> Size: 1
> Speed: 10 <--
> Range: 2
> Firepower: 2
> Cargo: 0

if you want to end game, please input "Exit", the winner will be the player who has the most cards
if you want to continue, please press enter
```

Fig 13. Screenshots 1.

```

Round 8
Round 8: Players have drawn their cards
You drew 'Vanguard' :
> Size: 3
> Speed: 4
> Range: 5
> Firepower: 5
> Cargo: 2

Round 8: Player Player_AI_3 won this round
The winner card was 'Idris':
> Size: 8
> Speed: 2
> Range: 7
> Firepower: 10 <--
> Cargo: 6

if you want to end game, please input "Exit", the winner will be the player who has the most cards
if you want to continue, please press enter

You have lost!

Round 9
Round 9: Players have drawn their cards
This round was a Draw, common pile now has 3 cards
the winning card is
> Size: 2
> Speed: 5 <--
> Range: 4
> Firepower: 3
> Cargo: 2

Game Over

The overall winner was Player_AI_3
Scores:
You: 0
Player_AI_1: 2
Player_AI_2: 1
Player_AI_3: 10
Player_AI_4: 0

Successful to write to database.
Do you want to see past results of play a game?
1. Print Game Statistics
2. Play game
3. Exit game
Enter the number for your selection: 1

Game Statistics:
Number of Games: 1
Total games users won: 0
Total games computers won: 1
Average draws per game: 4
Largest Number of rounds in a game: 17

```

Fig 14. Screenshots 2.

Conclusion and Recommendation

In conclusion, this project overall can achieve each user story. It has an MVC structure, and it has benefits to maintenance in the future. In addition, the expansibility of this project makes it easy to append the function which might receive from constructive ideas or feedbacks of users. There are two modes which were completed in two Sprints. The most logical edition is command line mode which has the Model to build the program. However, there might be other patterns to reduce coupling. As for online mode, users can play games graphically. It can bring a more friendly experience to the users.

Appendix

| | gameid [PK] integer | numberofround integer | gamewinner integer | numberofdraw integer |
|---|------------------------|--------------------------|-----------------------|-------------------------|
| 1 | 1 | 17 | 0 | 4 |
| 2 | 2 | 33 | 0 | 3 |
| 3 | 3 | 88 | 0 | 18 |
| 4 | 4 | 180 | 0 | 30 |
| 5 | 5 | 180 | 0 | 30 |
| 6 | 6 | 67 | 0 | 8 |
| 7 | 7 | 72 | 0 | 10 |
| | | | | |

Appendix Figure 1

GitHub link: <https://github.com/GuodongYang95/IT-Team-Project.git>