

MSc IT+ Masters Team Project

Group 19

Ioannis Athanasiadis (2057536)
Christopher Bellingham (2320989)
Joseph Doogan (2342934)
Pavlos Evangelidis (2349102)
Torquil MacLeod (2349654)

Contents

1	Introduction	4
1.1	A Sample Subheading	4
2	Development Process	5
2.1	Requirements Analysis	5
2.2	User Stories	5
2.3	Design	5
2.4	Command Line Mode	6
2.5	Online Mode	7
2.6	Sprints	7
2.6.1	Sprint #1	7
2.6.2	Sprint #2	7
2.6.3	Sprint #3	8
3	Technical	9
3.1	Assumptions	9
3.2	Functionality	10
3.3	Online Mode	12
3.4	Testing	13
3.5	Deficiencies	16
	Appendix A Minutes of Meeting	17
A.1	Project Kick-Off	17
A.2	Sprint Planning Meeting	18
A.3	System Design Meeting	22
A.4	Sprint Review/Planning Meeting	25
A.5	Sprint Review/Planning Meeting	28
A.6	Sprint Retrospective	30
	Appendix B Customer Specification	32
B.1	Purpose	32
B.2	Customer Specification	32
B.2.1	Context	32
B.2.2	Aim	32
B.2.3	Functionality	33
	Appendix C Final User Stories	38
C.1	Sprint #1: Command Line Mode	38
C.2	Sprint #2: Online Mode	45
C.3	Sprint #3: Online Mode (Additional)	52
	Appendix D Screenshots	56

List of Figures

1	Main Menu.	4
2	Full System UML.	6
3	Burndown Chart Sprint #1	7
4	Burndown Chart Sprint #2	7
5	Burndown Chart Sprint #3	8
6	User Stories in Trello.	20
7	Initial Project Plan	21
8	Proposed Architecture	23
9	Proposed UML	23
10	Proposed Logical Flow	24
11	Project Plan prior to Sprint #2.	27
12	Project Plan prior to Sprint #3.	29
13	Project Plan at Sprint #3 completion.	31

Todo list

<input type="checkbox"/> This section must be replaced with a proper introduction.	4
<input type="checkbox"/> A todo example	4
<input type="checkbox"/> Should we include an updated coverage matrix in an appendix?	5
<input type="checkbox"/> Some discussion needed.	5
<input type="checkbox"/> Some discussion needed.	6
<input type="checkbox"/> Some discussion needed.	7
Figure: Command line stats screenshot required.	11
Figure: Command line game screenshot required.	11
Figure: Command line winner screenshot required.	11
Figure: Online menu screenshot.	12
Figure: Online gameplay screenshot.	12
Figure: Online DB screenshot.	12
Figure: Include UML with Joe's updates.	23
<input type="checkbox"/> Check validity of each story.	38
<input type="checkbox"/> Check validity of each test.	38
<input type="checkbox"/> Check duration of each story.	38
<input type="checkbox"/> Allocate actual duration to each story.	38

1 Introduction

This section must be replaced with a proper introduction.

Give each sentence its own line in the source text. This makes it easier to review changes in GitHub.

An empty line in the source text can be used to force a new paragraph in the rendered document.

1.1 A Sample Subheading

A todo
exam-
ple

This is bold text. *This is italic text.* This is an example of text tagged with a todo item. Images can be inserted as below, put them into the *img* folder. If you want a caption (which will automatically be numbered and referenced in the list of figures), include the *captionof* tag as below.

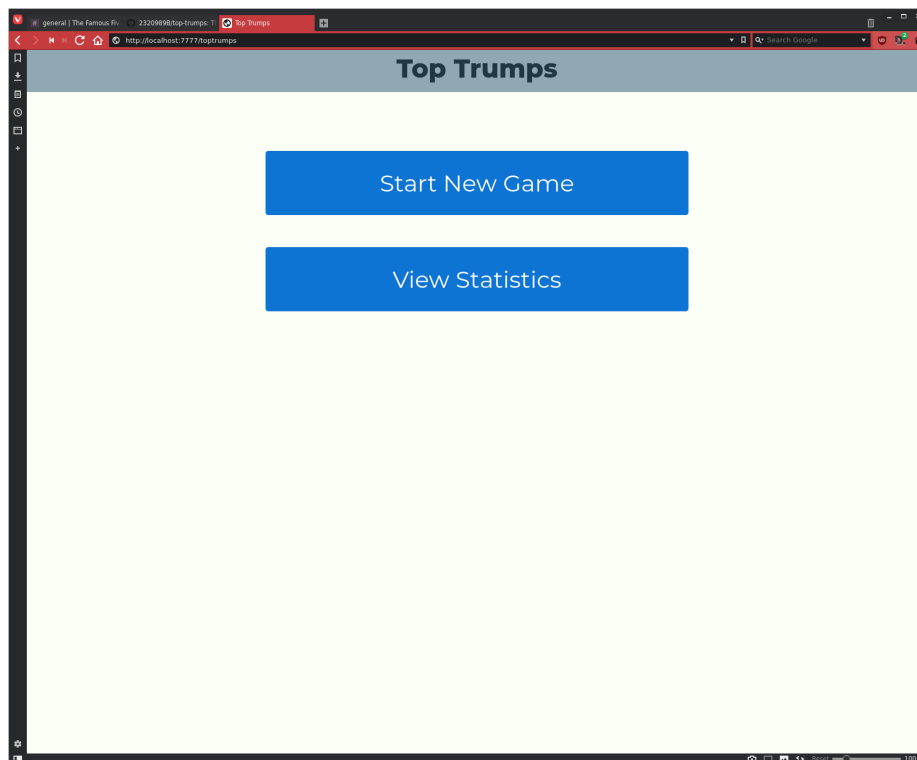


Figure 1: Main Menu.

2 Development Process

2.1 Requirements Analysis

Prior to generating user stories, the team took a disciplined approach to mapping out requirements. Functional and non-functional requirements were extracted from the provided specification document (ITSD2018-TaskDocument), and a “customer specification” was produced, which formed the basis for generation of user stories. Each requirement is given a unique requirement ID, allowing individual requirements to be discussed without confusion, and to allow cross-referencing with user stories.

A copy of the customer specification is provided in Appendix B.

2.2 User Stories

An initial set of user stories were brainstormed by each team member and posted on a Trello board. Reference is made to Appendix A.2 for minutes of the initial planning meeting, where user stories were finalised. To ensure full coverage of requirements, a coverage matrix was produced to give the team confidence the main aspects were captured.

The final set of user stories can be found in Appendix C.

Should we include an updated coverage matrix in an appendix?

2.3 Design

A dedicated design meeting was held to agree on the overall technical approach. Reference is made to Appendix A.3 for further details.

Some discussion needed.

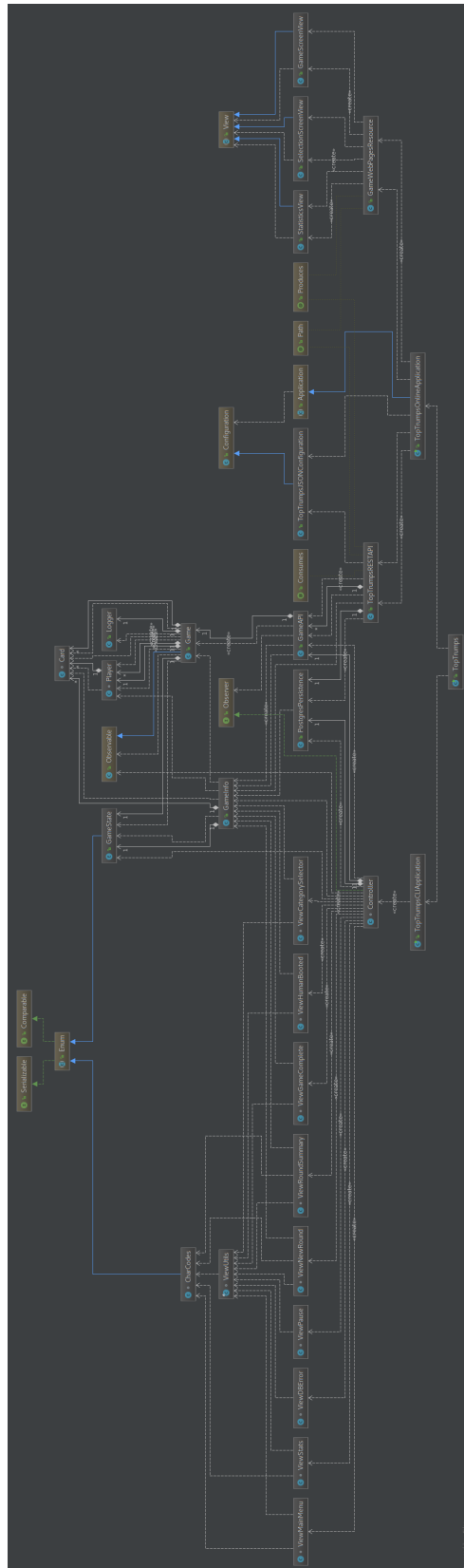


Figure 2: Full System UML.

2.4 Command Line Mode

Some discussion needed.

2.5 Online Mode

Some discussion needed.

2.6 Sprints

2.6.1 Sprint #1

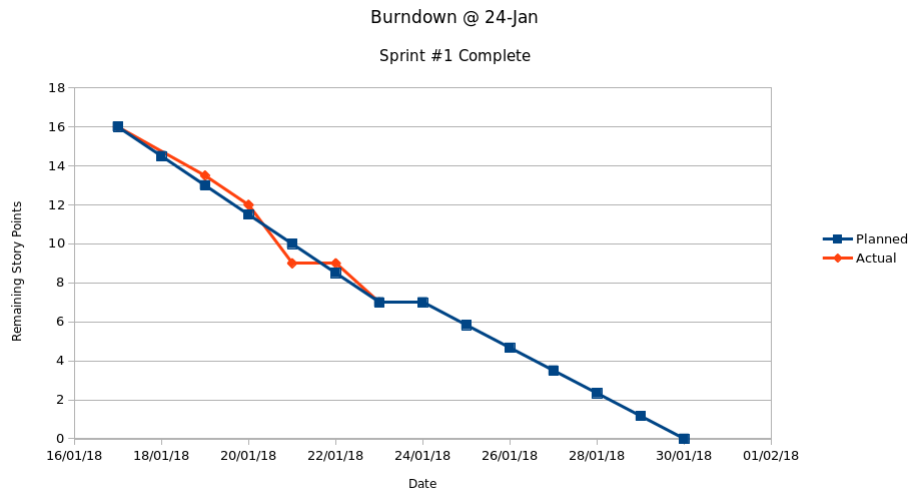


Figure 3: Burndown Chart Sprint #1

2.6.2 Sprint #2

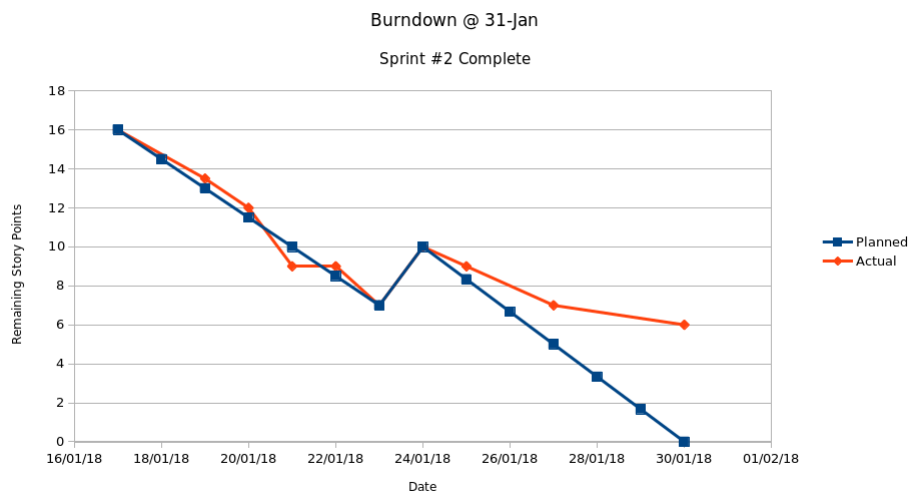


Figure 4: Burndown Chart Sprint #2

2.6.3 Sprint #3

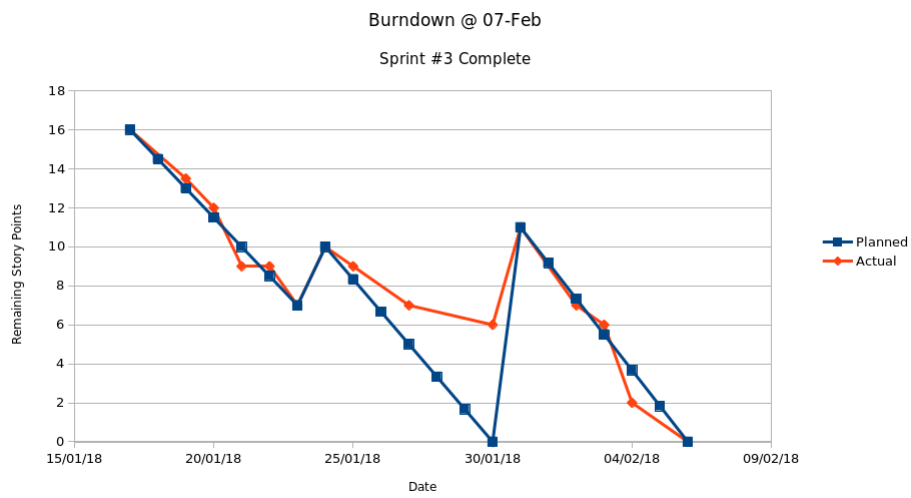


Figure 5: Burndown Chart Sprint #3

3 Technical

3.1 Assumptions

The following section contains an outline of the assumptions that have been made during the development of the product, assumptions that the prospective user should keep in mind.

When players have only one card in their individual decks and the outcome of the current round is a draw, then they lose the game as their cards go to the communal deck of cards. This means that, if the game has only two players left and the above scenario happens, then the winner ends up having less than the total of 40 cards in hand.

Another assumption is that in the case of a draw, the next stage of the game is considered to be a new round and not an extension of the draw round. Last winner player will still choose the category, but the game is between all of the players that are still in the game i.e. the ones that still have cards in their decks, and not only those that had the same category values.

It should also be noted that an AI player will always select the category that holds the best value on his top most card, and the selection will not be random. That will enhance the realistic feel of the game.

It can also be assumed that after every round the content of the communal deck is being collected by the winner player. This means that for the program, it doesn't matter if the result of a round was a draw or not. If the pile is empty and there is a winner, all players submit their cards in the empty pile and the winner player gets the cards from the round back. If the pile is non empty (i.e. the previous round resulted to a draw), then the winner still gets the current cards and the ones that were previously submitted because of the draw.

Another point that should not surprise a potential gamer, is that, if a player has only one card in hand and wins the current round, then his winning card will still be on top for the next round to come. That happens because the winner collects the cards from the other players and puts them at the back of his deck. Since he, had only one card to play with, this card remains on top by default. Again, that will only happen in the case of one card in hand and a winning result for this card holder.

The online mode assumes that once the player has pressed the button for selecting a category, he cannot change his mind and chose another one. That is, his option has been saved and the only available next step is to show the outcome of the round.

In command line, after the player has selected the category he wants to play with, his choice is only saved after pressing "enter", so he has the chance to change the category before selecting to proceed to the next step.

Another point is, that, as soon as the human player has been eliminated, he has no interest in watching the rest of the game, and therefore, in both modes, there is an option to fast- forward to the end, for which is notified.

Finally, the number of AI players in the online mode is determined by adjusting the JSON file.

3.2 Functionality

The functionality laid out by the program specification was largely the same across both modes – command line and online – with some additional individual functionality required by each.

Game Logic The logic of the Top Trumps game was implemented as specified. A shuffled deck of cards is distributed in turn to each player until all have been given out. One player will be human controlled with up to 4 AI players. Each card has 5 categories each with a positive integer value. The first active player is selected and chooses a category. All the card values are then revealed along with the winner (the holder of the card with the highest value). The winner takes all the cards in play and is the active player for the next turn. In the event of a draw, all the cards are added to a communal pile which will be won by the winner of the next round in addition to the cards in play.

Persistent Game Data Both modes were required to communicate with a database in order to store persistent data about gameplay. The user is provided with the total number of games played, the number of human and AI wins, the average number of draws per game and the longest game played (in number of rounds). This was achieved using PostgreSQL. The database schema (appendix X - schema file) allowed the required information to be saved and provided to the user.

Command Line Mode The command line opens with a menu asking if the user wants to see past statistics from the database or play a new game. Each round of a new game the user is provided with the round number, the active player and the human player's top card. The game logic is then followed and the user is informed who the round winner is before the next round commences. The number of communal cards is given in the event of a draw. If the user is eliminated the game runs to completion automatically. Once the game is completed the user is sent back to the menu and the database is updated with the game statistics.

The command line displays the game information in plaintext and allows the user to enter simple text commands via a numbered interface.

The command line also offers a test log function, whereby the following is logged:

- The contents of the deck when read
- The contents of the deck when shuffled
- The contents of all player decks when dealt
- The contents of the communal pile when altered
- The contents of the players card in play for each round
- The category selected
- The contents of each player's deck at round end
- The winner of the game

This output is saved to a text file.

The command line functionality requirements were met fully as shown below:



Command line stats screenshot required.



Command line game screenshot required.



Command line winner screenshot required.

3.3 Online Mode

The online mode hooks into the same game logic as the command line mode through API calls, however calls for different and more complex user interface functionality. The requirements for this interface were:

- A 'menu' page, similar to the command line, where the user selects either 'New Game' or 'View Statistics'
- The user is shown their own top card and allowed to select the category via buttons when it is their turn
- The user is shown who's turn it currently is
- The interface should clearly show every player's cards after a category is selected and highlight the winner
- The user should be informed when the round resulted in a draw
- The user should be able to see how many cards are left in each player's deck and the communal deck
- The winner should be indicated at the end of the game

These requirements were fully satisfied by the implementation as shown below:



Online menu screenshot.



Online gameplay screenshot.



Online DB screenshot.

3.4 Testing

The tests have been derived from the user stories of section 2.

The user persona Bob, wants to play a Top Trumps game in the command line.

User Story	S0010
Type of Test	Initiate the game from command line
Used Variable	- "java -jar ITSD2018Project-1.0.jar -c" - "java -jar ITSD2018Project-1.0.jar -c -t"
Result	[Passed] The program returns the main menu to the player, (figure 1 Appendix). As shown above, both game options were used as variables to test the initialisation of the game.
Anti-Variables	- "java -jar ITSD2018Project-1.0.jar -command" - "java -jar ITSD2018Project-1.0.jar -c -o" - "java -jar ITSD2018Project-1.0.jar -n"...
Results	[Exception] The program never initiates the game, if the flag is anything else excepts "-c", "-c -t" or "-o". The program throws an exception in case where the user tries to operate both modes, command line mode and on-line mode. In all other cases the program shows the greeting message and terminates.

The user persona Bob can now select one of the three options of the game.

User Story	S0010
Type of Test	Select an integer number from the provided options of the main menu.
Used Variables	- integer number, where n=[1,3].
Result	[Passed] The player enter the integer n=1, which initiates a new game. The program returned with a new game.
Anti-Variables	- String - Character - Double or Integer, where n!= [1,3]
Results	[Exception] The program throws an exception when the player does not enter the specified integer between n = [1,2,3]. In case the player enters a different integer, the program throws a message indicating the exact integer variable, where in all other situations the program prints out a message demanding an integer.

After the proper response of Bob to the game commands, he expects from the game to load the cards, shuffle them, and divide to all players as equally as possible.

User Story	S0020
Type of Test	Examination of the test log file for the correct loading of deck, correct shuffling and dividing
Used Variables	No variables for this test.
Result	[Passed] The program does the deck loading, shuffling, creating players' deck and choose randomly the active player correctly.
Anti-Variables	<i>No anti-variables for this test</i>
Results	<i>[No Exception] There is no results for this test.</i>

Bob wants to have a detailed round and understand who is active player, which category is selected and who won the round.

User Story	S0030 & S0040
Type of Test	Game round quality test.
Used Variables	- Press Enter button to show winner
Result	[Passed] The program shows the round number, who is active player, all the cards that participate in this specific round, the selected category inside square brackets and the round winner.
Anti-Variables	<i>No anti-variables for this quality test</i>
Results	<i>[No Exception] there is no exception for this test, only through the test log file.</i>

Bob, who is an expert user of command line, wants to have a record of the entire game in details in a test log file. He is able to do so, with the -t flag when he initiate the game

User Story	S0050
Type of Test	Run the game with the "-t" flag.
Used Variables	- "java -jar ITSD2018Project-1.0.jar -c -t"
Result	[Passed] The program recognise the -t flag and checks if there is no file as "toptrumps.txt" log file, it creates one, or it rewrites in the existing one.
Anti-Variables	- "java -jar ITSD2018Project-1.0.jar -c -*", where * is anything else than "-t".
Results	<i>[No Exception] The program recognise the -c flag but it does not produce or write in the "toptrumps.txt" file.</i>

Bob is very cautious about the validity of the game. As a result he wants to examine who won the round and if the AI player selected the highest attribute.

User Story	S0130
Type of Test	Print players top cards. Quality test
Used Variables	<i>No variables used to this test</i>
Result	[Passed] The program prints out the top cards from all available players. Moreover, the program shows the selected category.
Anti-Variables	<i>No anti-variables used for this test.</i>
Results	<i>[No Exception] There is no exception for this test.</i>

Bob is the active player in the game and wants to select a category. The program provides him with the option to enter a integer number that matches a specific category.

User Story	S0030
Type of Test	Human Player is active player and selects a category
Used Variables	Integer number n=[1,5]
Result	[Passed] The program receives the input integer variable and makes the desired category active. Then, it returns to the system out the results of the round.
Anti-Variables	String, Characters, Double or Integer variables except n=[1,5].
Results	[Exception] The program throws an exception when the human player did not enter one of the five options. The program catches NULL values, strings, characters, doubles and integer number except the n=[1,5]

Bob continues his game and he managed to eliminate some of the players, but he wants to know who has been eliminated.

User Story	S00–
Type of Test	Show who players have been eliminated while the human player is in the game. Quality test.
Used Variables	<i>No variables used for this test.</i>
Result	[Passed] The program prints out a list with all the active players of the game in each round, right above the players hand.
Anti-Variables	<i>No anti-variables used for this test.</i>
Results	<i>[No exception] There is no exception for this test.</i>

3.5 Deficiencies

The program has been tested thoroughly, during different steps of development, after every sprint in a larger scale, and as a final complete product multiple times. No missing functionality has been observed, and all tests were passed.

Minor points can be mentioned, but again they are not part of functionality and they do not affect the program in any way. For example, in command line mode, cards are represented graphically as rectangles. If the user tries to reduce the terminal window size below a certain point, the rectangles overlap and do not re-shape automatically to fit the window. It is assumed though, that player would not want to run the game in a very small terminal, so that should not be an issue. Perhaps this would be something to adjust if the period available for the project completion was a little longer.

A Minutes of Meeting

A.1 Project Kick-Off

Date	Fri 12-Jan 2018 (Week 1)
Venue	Boyd Orr
MoM Author	Christopher Bellingham
Participants	Ioannis Athanasiadis [IA] Christopher Bellingham [CB] Joseph Doogan [JD] Pavlos Evangelidis [PE] Torquil MacLeod [TM]
Apologies	-

Item	Detail	Resp.	Due
1	Team reviewed and approved Team Organisation Document.	INFO	INFO
2	Publish latest Team Organisation Document to Slack to allow each team member to submit to Moodle.	CB	Fri 12-Jan
3	Create Customer Specification, capturing all requirements. Post-meeting note, see Appendix B.	CB	Sun 14-Jan
4	Populate Trello with User Stories based on Customer Spec prior to Sprint Planning meeting on Wed-17.	ALL	Wed 17-Jan
5	Consider architecture and high-level Object Orientated design prior to Design meeting on Wed-17.	ALL	Wed 17-Jan

A.2 Sprint Planning Meeting

Date	Wed 17-Jan 2018 (Week 2)
Venue	Slack
MoM Author	Christopher Bellingham
Participants	Ioannis Athanasiadis [IA] Christopher Bellingham [CB] Joseph Doogan [JD] Pavlos Evangelidis [PE] Torquil MacLeod [TM]
Apologies	-

Item	Detail	Resp.	Due
1	User stories submitted to Trello (Figure 6) were reviewed. Duplicate stories were eliminated, remaining stories were allocated durations.	INFO	INFO
2	Unique Story IDs were granted to each User Story, in the form SXXXX. This will aid communication as development progresses.	INFO	INFO
3	Team agreed to define each story as a Must Have, since all generated stories are strictly limited to the requirements of the specification. The team reserves the right to de-prioritise individual stories should the workload be considered too high.	INFO	INFO
4	The sum of all story points is 16. Broadly, Team feels development can be split into two main phases - command line mode, and online mode. Resultant stories can be found in Appendix C.1 and Appendix C.2 respectively.	INFO	INFO
5	Command line mode covers stories S0010, S0020, S0030, S0040, S0050, S0130, S0180. This equates to a total of 9 story points.	INFO	INFO
6	Online mode covers stories S0190, S0200, S0210, S0220, S0230, S0240, S0250. This equates to a total of 7 story points.	INFO	INFO

Item	Detail	Resp.	Due
7	Development of command line and on-line mode will be tackled over the course of two 1-week long sprints. With consideration for the need for sprint planning/retrospectives, this leaves 6 days per sprint. Each team member's capacity was assumed to be one third of this duration, I.e. 2 ideal days each. Considering a team size of 5, total available capacity per sprint is 10 ideal days.	INFO	INFO
8	With consideration of team capacity, it is deemed feasible to tackle the project over the course of two 1-week sprints. Sprint 1 will tackle command line mode. Sprint 2 will tackle online mode.	INFO	INFO
9	With initial consideration to an MVC-style architecture, it was noted that each story may slice through multiple layers of this architecture. Team considers it sensible for individuals to have responsibility over different segments of the system, as such each user story will be split between multiple developers. Allocation of user stories to individuals will therefore be deferred until this afternoon's design meeting, where architecture and class structure will be finalised.	INFO	INFO
10	Project schedule was created, allowing for 6-days float/contingency prior to report submission (Figure 7).	INFO	INFO
11	It was agreed that the requirements presented in Appendix B should be cross-referenced to each user story, to ensure full coverage of requirements. A coverage matrix will be produced by the team to confirm there are no coverage gaps.	ALL	Fri 19-Jan

The screenshot shows a Trello board titled "User Stories" with a blue header. The board is organized into several lists:

- Roles:** Contains two cards: "Human Player" and "AI Player".
- Personas:** Contains two cards: "BOB: a Computer Science graduate from Aviemore, currently living in Glasgow, with a passion for gaming and puzzles. As a computing expert, he is familiar with the command line and prefers to use that instead of a browser. He has stated that he will play the game only in this mode." and "LILLY: is a lawyer from Edinburgh. She is enjoying playing video games and she spends a few hours per week on online games. Although familiar with computers and the internet, she is not a hardcore user, and therefore she is not willing to use the command line for our game. She has mentioned that she will be playing only the online mode of the game."
- User Stories (Both modes):** Contains one card: "Add a card..."
- User Stories (Command Line):** Contains three cards:
 - [S0010] Story Point: 1/2 day. Priority: Must Have. As a Human Player, I want to start the program in command line mode, so that I can either play a new game or see statistics of previous games. [C2010; C2030; C7000; C7010; C7020] BOB can start the program in CL and he is presented with a question of whether he wants to play a new game, view statistics or quit. The question asks Bob to enter "1" for a new game, "2" for viewing the stats, "3" to quit the game, and he must "press enter" to submit his choice. Acceptance tests: Try to press enter with empty string. Try with a random number other than "1", "2" or "3". Try with correct numbers.
 - [S0020] Story Points: 2. Priority: Must have. As a Human Player, I want to start a new game, so that the first active player can select a category [C1000; C1010; C2010; C2020; C3000; C4000; C4010; C4050; C4140; C4150; C4160; C7030; A1000] Bob is asked to select the number of AI opponents he would like to play against (max 4). He does this by typing a number and pressing enter. Acceptance Tests: Test numbers 1 - 4. Test numbers other than 1 - 4 inclusive. Test empty string. Test that deck is loaded. Test that deck is shuffled. Test that correct number of players are created. Test that players are dealt the correct number of cards. Test that one player is made the active player. Test that correct output is provided by standard out.
 - [S0030] Story Point: 1. Priority: Must have. As a Human Player who is selected as the active player, I want to select a category, so that the game can check the outcome of the round. [C1020; C4170; C7040; C7050] After the current round finishes, BOB can check the screen to find out who the winner was. The format is "X wins this round" with X being BOB or the AI
- User Stories (Online):** Contains three cards:
 - [S0190] Story Points: 1. Priority: Must Have. Lilly, is known for her attitude of succeeding on everything, so she always watch her performance before starting a game (access the DB before a game starts). She wants to see the number of games played overall, how many times a human won and how many the 'AI', the average draws per game, the largest number of rounds played in a game. [C5070; 5080; 5090; 5100; 5110; 5120]
 - [S0200] Story points: 1/2 Priority: Must have As a human player I want to start the game in the online mode by using an -o flag either in terminal or IDE. I want to be able to start a new game in different tabs and play different games concurrently with 1-4 opponents. After I run the command in terminal, or run in the IDE, I can open a browser and initialise a game by typing in http://localhost:7777/toptrumps Covers:[C1000; C2040; C2050; C2060; C8000] Acceptance tests: Try thought terminal with -o flag and check for errors. Run with Eclipse with argument -o provided. If no errors occur, open browser and type in the http. Check multiple tabs to check that the initial selection screen is active Check that the selection screen contains buttons for new game and statistics. Click on the Start Game button and check that the browser displays the correct number of game. Check that the human player is always present and that the AI players are present and not more than four . If number of AI players is not selected randomly, check that my input corresponds to the number of AI players on screen
 - [S0210] Story points: 1/2 Priority: Must have AS a human player I want to be able to access the DB to view statistics upon clicking the Statistics button on the selection screen. This has the same functionality as in the Command
- Binned Stories:** Contains four cards:
 - [S020] Bob, wants to read a test log file that has extensive information about the operation of the game. He will be able to do so with a -t flag in the command line. [C6000; 6010; 6020; 6030; 6040; 6050; 6070; 6080; 6090; 6100]
 - [S0150] As an 'AI' player, if there is a draw when I am the active player, I get to select the category in the next round. [C1050; C7050]
 - [S0160] As an 'AI' player, if I am the active player and lose the current round, the winning 'AI' player or 'User' becomes the active player for the next round [C1070; C7050]
 - [S0140] As an 'AI' player, after winning a round I become the active player and get to select the category in the next round. [C1070; C7050] and [S0150] As an 'AI' player, if there is a draw when I am the active player, I get to select the category in the next round. [C1050; C7050] Same tests as per human player
 - [S0100] As a human player, if I am the active player and lose the current round, the winning 'AI' player becomes the active player for the next round [C1070; C7050]
 - [S0090] As a human player, if there is a draw when I am the active player, I get to select the category in the next round. [C1050; C7050] Test: Have the contents of both cards printed on the screen. Check if there is a common values between the cards. As a player select the category that has this value so you can force a draw Check that you are announced the winner of the round and that there is a new prompt to select a category again as the winning player of the previous round.

Figure 6: User Stories in Trello.

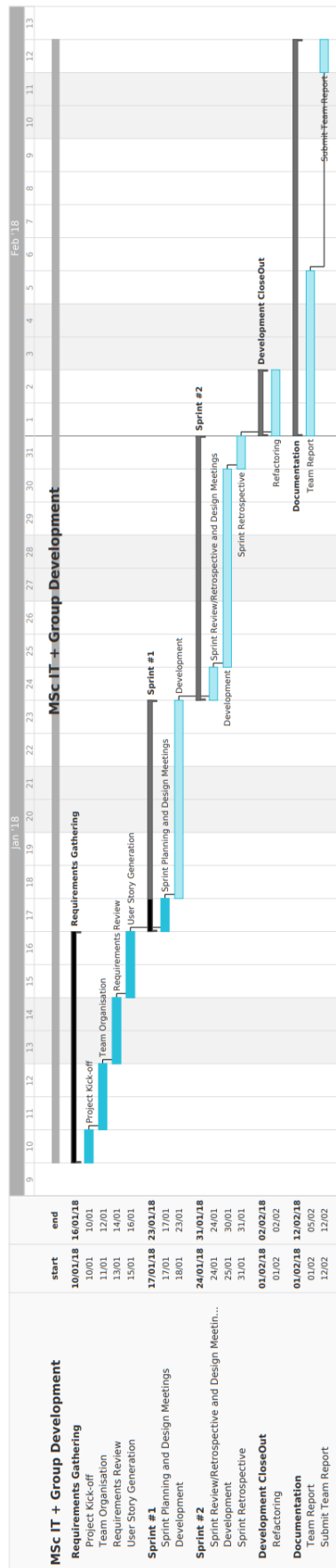


Figure 7: Initial Project Plan

A.3 System Design Meeting

Date	Wed 17-Jan 2018 (Week 2)
Venue	Slack
MoM Author	Christopher Bellingham
Participants	Ioannis Athanasiadis [IA] Christopher Bellingham [CB] Joseph Doogan [JD] Pavlos Evangelidis [PE] Torquil MacLeod [TM]
Apologies	-

Item	Detail	Resp.	Due
1	CB provided a candidate architecture diagram (Figure 8), outlining use of an MVC architecture, with a data persistence layer. CB proposed use of the Observer Pattern as a means of reducing coupling between MVC layers.	INFO	INFO
2	Use of the Observer Pattern was identified as a risk, since no team member has experience with this. CB will outline how this would work by providing an example of a simple implementation.	CB	Wed 17-Jan
3	Prior to the meeting, each team member had submitted class diagrams to enable discussion on needed class structures. Each submission was reviewed, and it was noted that a lot of commonality exists across proposed classes (typically Game, Player and Card classes).	INFO	INFO
4	CB provided detailed UML covering full system (Figure 9). It is understood that online mode can hook into key Game functionality via game's available public methods, and online components can be notified of game state changes via the Observer mechanism. The approach as depicted was agreed to be a reasonable solution, and was selected as a basis for overall design.	INFO	INFO
5	CB provided detailed sequence diagram covering logical flow for command line mode (Figure 10). This may be used as a reference during development.	INFO	INFO
6	User stories allocated to team from Appendix C.1 to allow development to commence.	INFO	INFO

Candidate Architecture #1

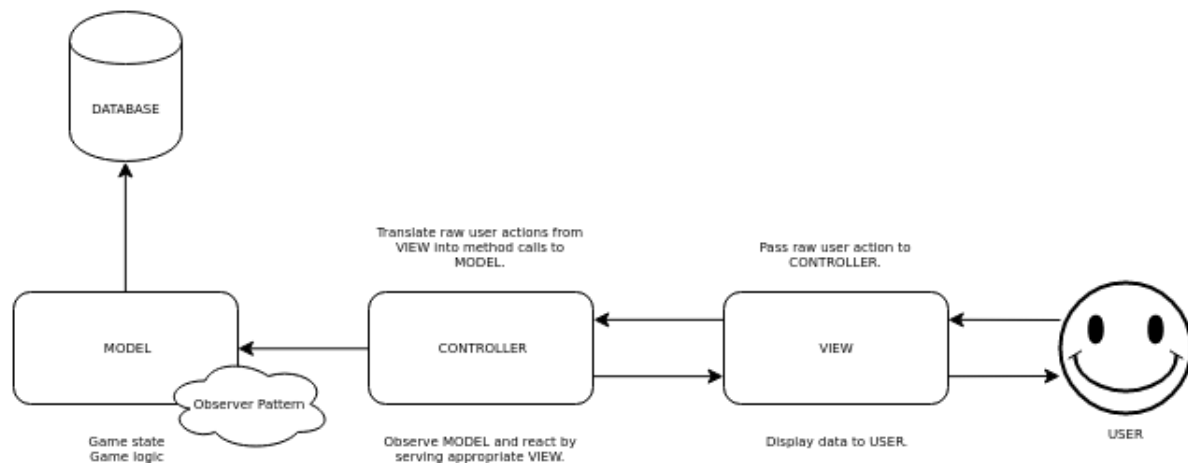


Figure 8: Proposed Architecture



Include UML with Joe's updates.

Figure 9: Proposed UML

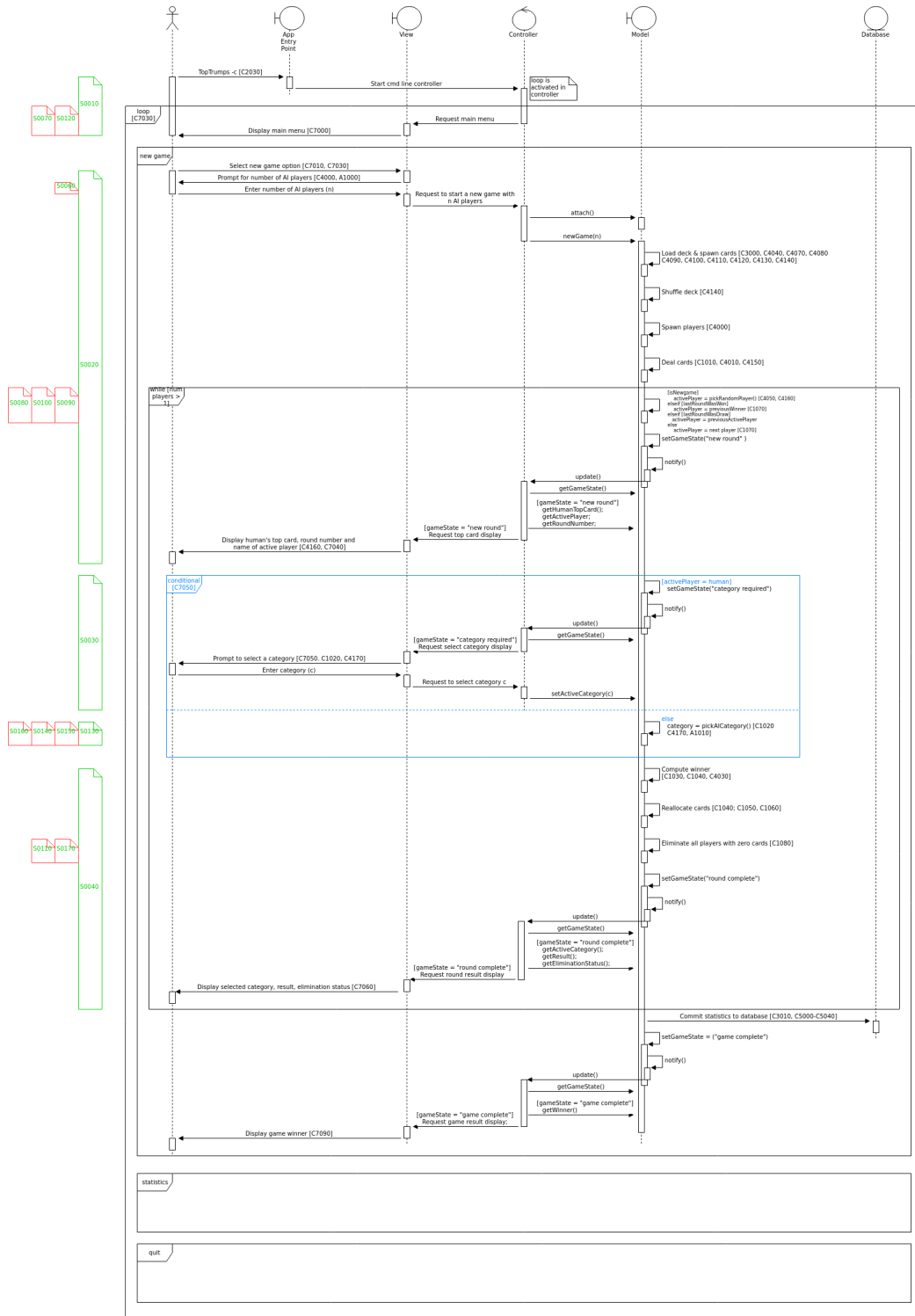


Figure 10: Proposed Logical Flow

A.4 Sprint Review/Planning Meeting

Date	Wed 24-Jan 2018 (Week 3)
Venue	Slack
MoM Author	Christopher Bellingham
Participants	Ioannis Athanasiadis [IA] Christopher Bellingham [CB] Joseph Doogan [JD] Pavlos Evangelidis [PE] Torquil MacLeod [TM]
Apologies	-

Item	Detail	Resp.	Due
1	Sprint #1 user stories were reviewed, and it was determined that all planned stories (see Appendix C.1) were completed.	INFO	INFO
2	It was noted that initial story point estimates for S0030, S0040 and S0130 were insufficient, these took longer than expected. See Appendix C.1 for actual durations. This was due to unexpected underlying complexities, and the need to build-out the initial underlying program structure in order to implement these features.	INFO	INFO
3	Team feels that the workflow, particularly use of Slack to communicate, has been effective. No changes to process are deemed necessary.	INFO	INFO
4	Discussion moved onto Sprint #2, where the stories in Appendix C.2 were reviewed. Ref item [2], to learn lessons from Sprint #1 overrun, the story points allocated to Sprint #2 stories were increased, specifically stories S0200, S0220 and S0230). This is in recognition of technology risk, as no team member has experience with Dropwizard, and only limited experience with HTML/CSS. The assumption remains that the underlying Model logic components developed in Sprint #1 can be re-used however.	INFO	INFO
5	Based on sum of all proposed Sprint #2 stories, velocity for this sprint increases from originally planned 7 to 10. Team feel able to meet the increased workload.	INFO	INFO

Item	Detail	Resp.	Due
6	Project schedule was presented, updated to show latest progress. 6 days still remain for float/contingency prior to report submission (Figure 11).	INFO	INFO
7	User stories allocated to team from Appendix C.2 to allow development to commence.	INFO	INFO

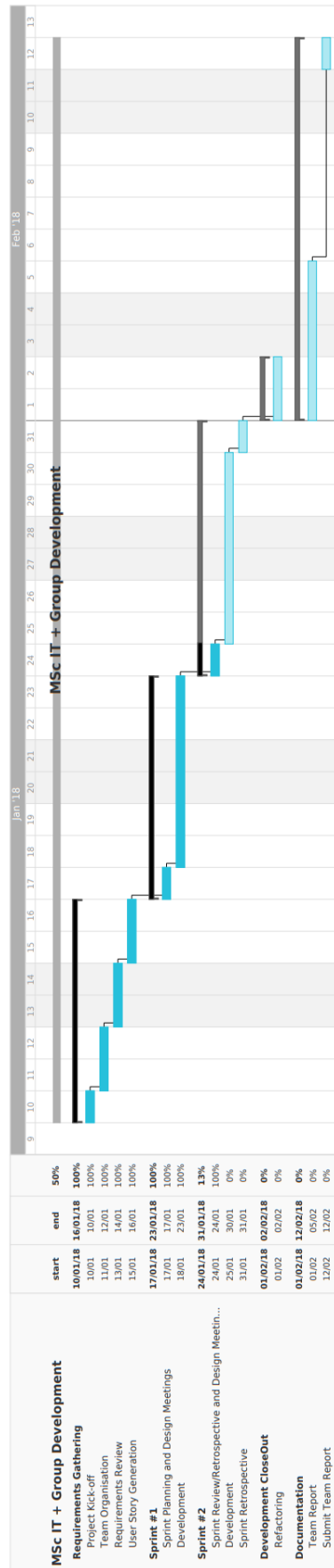


Figure 11: Project Plan prior to Sprint #2.

A.5 Sprint Review/Planning Meeting

Date	Wed 31-Jan 2018 (Week 4)
Venue	Slack
MoM Author	Christopher Bellingham
Participants	Ioannis Athanasiadis [IA] Christopher Bellingham [CB] Joseph Doogan [JD] Pavlos Evangelidis [PE] Torquil MacLeod [TM]
Apologies	-

Item	Detail	Resp.	Due
1	Sprint #2 user stories were reviewed, and it was determined that not all planned stories (see Appendix C.2) were completed. Stories S0220, S0230, S0240 and S0250 remain incomplete.	INFO	INFO
2	Lack of team experience with the online technologies is responsible for the overrun. The asynchronous and stateless nature of the API calls was not considered during design stage. Some additional work is needed to refactor the underlying model to suit the online technologies.	INFO	INFO
3	A third sprint must be scheduled to cover refactoring and to allow completion of stories S0220, S0230, S0240 and S0250.	INFO	INFO
4	A revised project plan is proposed, making use of the available schedule float (Figure 12).	INFO	INFO
5	Durations for user stories S0220, S0230, S0240 and S0250 adjusted to capture required refactoring. See Appendix C.3.	INFO	INFO
6	Based on sum of all proposed Sprint #3 stories, velocity for this sprint is 11. Team feel able to meet the increased workload.	INFO	INFO
7	Duration for team report is extended to account for reduced available resources. Report can be written in parallel with Sprint #3, and extended over the final weekend (Figure 12).	INFO	INFO
8	User stories allocated to team from Appendix C.3 to allow development to commence.	INFO	INFO

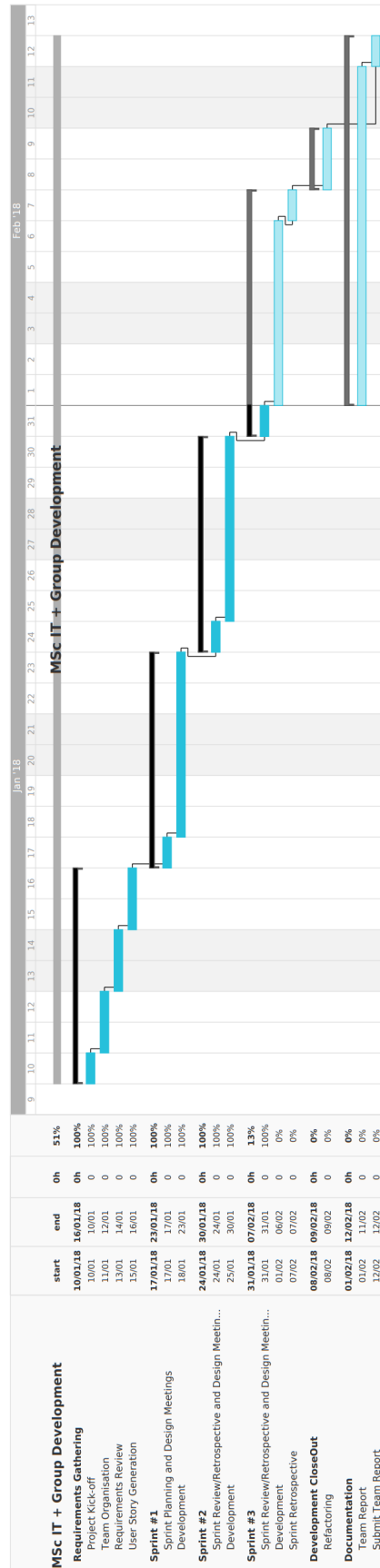


Figure 12: Project Plan prior to Sprint #3.

A.6 Sprint Retrospective

Date	Wed 07-Feb 2018 (Week 5)
Venue	Slack
MoM Author	Christopher Bellingham
Participants	Ioannis Athanasiadis [IA] Christopher Bellingham [CB] Joseph Doogan [JD] Pavlos Evangelidis [PE] Torquil MacLeod [TM]
Apologies	-

Item	Detail	Resp.	Due
1	Sprint #3 user stories were reviewed, and it was determined that all planned stories (see Appendix C.3) were completed.	INFO	INFO
2	All stories were completed in-line with their expected durations.	INFO	INFO
3	Opportunity will be taken by all team members to review all code and tidy-up prior to submission.	INFO	INFO
4	Final testing will be performed in the Boyd Orr Lab, Thurs 08-Feb, including testing the database functionality on the Yacata server. All team members will present in the Lab to support any issues.	ALL	INFO

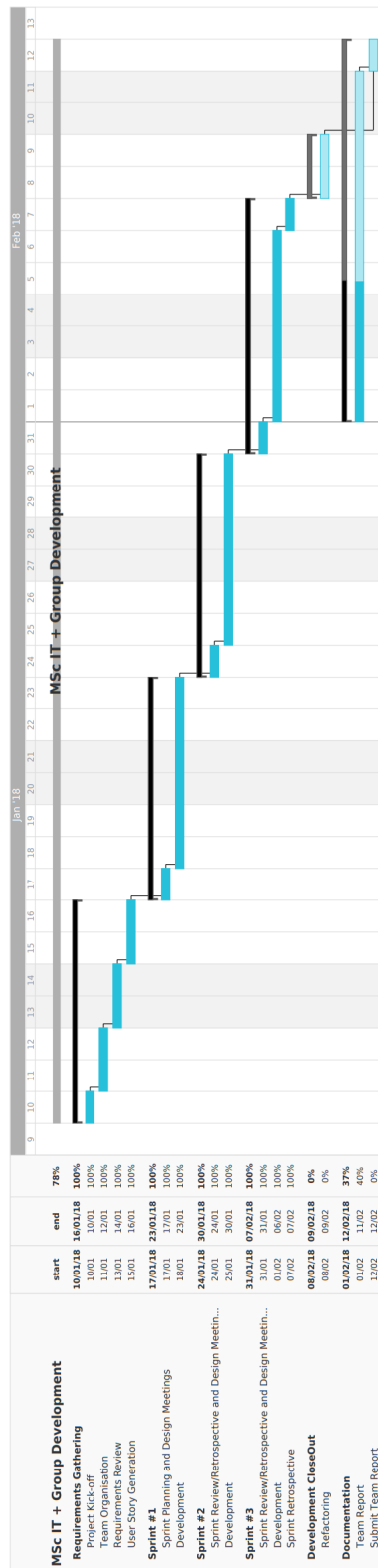


Figure 13: Project Plan at Sprint #3 completion.

B Customer Specification

B.1 Purpose

This appendix captures the foundational requirements of the system.

Section B.2 is quoted verbatim from Assignment Specification ITSD2018-TaskDocument. Requirements within are identified in **bold** and given a unique requirement ID, used for cross-referencing with User Stories.

B.2 Customer Specification

B.2.1 Context

Top Trumps is a simple card game in which decks of cards are based on a theme. For example, race cars, dinosaurs, and even TV shows like “The Simpsons”. Within a deck each card represents an entity within that topic (e.g. T-Rex for dinosaurs or Bart Simpson for the Simpsons). Within a deck each card has the same list of characteristics. For example, dinosaurs can have a height, weight, length, ferocity, and intelligence. Each card has a value for each characteristic of the deck. The objective of the game is to “trump” your opponent by selecting a category (e.g. intelligence) and having a “better” value for your card than the opponent does in their current card.

Gameplay is as follows:

- [C1000: There must be at least two players]. [C1010: The deck of cards is divided between the players]. [C1020: The first player takes their topmost card and selects a characteristic]. [C1030: The value of that characteristic is compared against the value for the same characteristic in the other players’ top card]. [C1040: The player with the best value for that characteristic wins the round and the winner takes all the cards from that round (including their own) and places them at the back of their deck]. [C1050: If there is a draw the cards from the round are placed in a new communal pile and a new characteristic is selected by the same player from the next card]. [C1060: The winner then takes the cards from the round and any in the communal pile]. [C1070: The winner of a round maintains the choice of category until they lose, then the choice moves to the next player]. [C1080: Players lose the game when they have no cards left; the player left with all the cards is the winner of the game].

B.2.2 Aim

The aim is to build a computer program to allow a user to play top trumps against one or more AI opponents given a deck. [C2000: **The program should have two modes**]:

- [C2010: **Command Line Mode: The game is played only through command line input and output**]. [C2020: **In this mode, only one game can be played at a time**]. [C2030: **This mode should be selected via a -c flag when starting the program**]: `o java -jar TopTrump.jar -c`
- [C2040: **Online Mode: The game should be hosted as a web service, comprised of a REST API that provides remote access to the core game**

functionality and one or more web-pages that enable a user to play the game]. [C2050: In this mode, multiple users should be able to play the game concurrently (e.g. in different Web browser tabs)]. [C2060: This mode should be selected via a -o flag when starting the program]: o java -jar TopTrumps.jar -o

B.2.3 Functionality

In both modes, the program needs to:

- [C3000: Enable a user to play a game of top trumps with a deck that was loaded in when the program started].
- [C3010: Store the results of past games played in a database as well as visualize that information to the user on-demand].

In command line mode only:

- [C3020: Write a test log to file that contains snapshots of the programs state as it runs].

We discuss each of these requirements in more detail below:

Playing Top Trumps The program should implement the top trumps game as described in the Context section above. You can make the following assumptions to simplify the implementation. Do not add additional unnecessary functionality. This is gold-plating and will not result in a better grade. If in doubt, consult the course co-ordinator.

- [C4000: There should be one human player and up to 4 computer players (AIs)].
- [C4010: If a deck does not divide equally between the players, then some players may have less cards. For example, if there are 3 players and 40 cards, then two players receive 13 cards and one player receives 14].
- [C4020: A deck has 5 criteria and the criteria are always positive integers between 1 and 50 (inclusive)].
- [C4030: A higher number is always better for any given characteristic].
- [C4040: There are 40 cards in a deck.txt file].
- [C4050: The first player should be selected at random].
- [C4060: A draw won't continue until the point where there are only cards in the communal pile you do not need to deal with this programmatically, just assume that should this happen you are not expected to deal with it].

[C4070: The deck we will be using is stored in a text file called 'StarCitizenDeck.txt']. [C4080: The first line of the text file should contain a list of the categories in the deck, separated by a space]. [C4090: All decks should have a 'description' category to label the individual cards]. [C4100: A description within a deck can be assumed to be unique and a single word]. [C4110: The subsequent lines contain details of one card]. [C4120: You can assume the order of the categories from the first line align with 3 the values provided for the cards and that all cards have a value for all categories in a deck]. [C4130: You can assume that all categories are single words].

For example, in a dinosaur deck (numbers bear no resemblance to reality):

```
description height weight length ferocity intelligence
TRex 6 6 12 9 9
Stegosaurus 4 3 8 1 8
Brachiosaurus 12 8 16 2 6
Velociraptor 3 5 5 12 10
Carnotaurus 5 6 7 9 8
Iguanodon 2 2 3 1 9
Megalosaurus 9 9 8 6 9
Oviraptor 8 7 4 3 2
Parasaurolophus 7 7 1 3 4
Ornithomimus 10 9 8 7 5
Protoceratops 9 5 4 7 10
Riojasaurus 6 1 4 7 7
Saurolophus 7 1 10 7 8
Styracosaurus 7 3 4 1 1
Xiaosaurus 10 6 5 7 2
```

and so forth the star citizen-based deck is provided with the Template Package that can be downloaded from Moodle.

[C4140: The program must first load all card details from the deck and shuffle them (randomly order them)]. [C4150: The program should then deal the cards between the players]. [C4160: The user should then be shown the detail from their top card (note there is no need to visualise this in a complex manner, the card details can be shown in text) and the first player is randomly selected]. [C4170: If the player is an AI player it should select a category for play, if the user is the first player they should be allowed to select a category to play the round]. [C4180: The game play should then proceed as detailed in the section Context].

Persistent Game Data [C5000: Upon completion of the game, the user should automatically write the following information about the game play to a database]:

- [C5010: How many draws were there?]
- [C5020: Who won the game?]
- [C5030: How many rounds were played in the game?]

- [C5040: How many rounds did each player win?]

You should select one team members database on the yacata server, from the Database Theory and Applications course, to write to. [C5050: It is important you do not remove the username and password from the final code, as this allows us to test the software]. You should also provide details of this database (username, database name and password) in the report.

[C5060: There should also be possible, so long as a game isn't currently in progress, for the user to connect to the database and get information about previous games]. This should include the following:

- [C5070: Number of games played overall].
- [C5080: How many times the computer has won].
- [C5090: How many times the human has won].
- [C5100: The average number of draws].
- [C5110: The largest number of rounds played in a single game].
- [C5120: These values should be calculated using SQL].

Test Log In addition to the functionality described above, you should implement the following to allow for program debugging when in command line mode only. [C6000: When the program is started, if a -t flag is set on the command line, then the program should write out an extensive log of its operation to a toptrumps.log file in the same directory as the program is run], e.g.: `java -jar TopTrumps.jar -c -t`

[C6010: If a toptrumps.log file already exists, your program should overwrite that file]. [C6020: Your program should print the following information to that file, separated by a line containing dashes at the appropriate times as mentioned below]:

- [C6030: The contents of the complete deck once it has been read in and constructed].
- [C6040: The contents of the complete deck after it has been shuffled].
- [C6050: The contents of the users deck and the computers deck(s) once they have been allocated. Be sure to indicate which the users deck is and which the computers deck(s) is].
- [C6060: The contents of the communal pile when cards are added or removed from it].
- [C6070: The contents of the current cards in play (the cards from the top of the users deck and the computers deck(s))].
- [C6080: The category selected and corresponding values when a user or computer selects a category].
- [C6090: The contents of each deck after a round].
- [C6100: The winner of the game].

Command Line Mode [C7000: When started in command line mode, the program should ask the user whether they want to see the statistics of past games (see the Persistent Game Data section) or whether they want to play a game]. [C7010: The users choice should be obtained from standard in (System.in)]. [C7020: If they select to see statistics of past games, the program should print the associated statistics to standard out (System.out) and then ask the same question again]. [C7030: If they select to play a game, a game instance should be started, and the core game loop initialized]. [C7040: For each round, the round number, the name of the active player and the card drawn by the player should be printed to standard out]. [C7050: If the user is the active player, then it should ask the player to select a category, and obtain the users choice from standard in, otherwise the AI player should select a category]. [C7060: The program should then print to standard out the selected category, who won (or whether it was a draw), the winning card and whether the player has been eliminated (they have no cards left)]. [C7070: Rounds should continue to be played until a winner is determined (only one player has cards left)]. [C7080: Once the user has been eliminated, the remaining rounds should be completed automatically (without user input)]. [C7090: At the end of the game, the overall winner should be printed, and the Persistent Game statistics should be updated]. [C7100: The user should then be asked if they want to print the statistics of past games or play another game]. [C7110: An example of the command line output for the program is provided in the CLI.example.txt file on Moodle].

Online Mode Online mode is an extension to the command line mode that provides a version of the game that users can play through their Web browser. A Web application has two main components. First, a back-end Application Programming Interface (API) that provides remote access to the game functionalities (e.g. starting a new game, drawing cards or getting players/cards for display). Second, one or more webpages that use the back-end API to enable the user to play the game. In this case, each web page should be comprised of HTML elements that are displayed and Javascript functions that connect to the API.

The webpage design is down to you, but it should display the following:

- [C8000: Upon loading the web page, the user should be presented the option to view overall game statistics (as detailed previously), or play a single game].
- [C8010: During game play, the GUI should display the contents of the users top card and, when they are the active player, allow the user to select a category to play against the computer].
- [C8020: The GUI should clearly indicate whos turn it currently is, and only allow the user to select a category when it is their turn].
- [C8030: Once the category has been selected for a round and the values compared, the GUI should display the values of the category for each player and highlight who won the round].
- [C8040: If a draw should occur, the user should be notified of this].

- [C8050: The GUI should contain an indication of how many cards are in the communal pile].
- [C8060: The GUI should contain an indication of how many cards are left in the users deck and in the computers deck(s)].
- [C8070: When the round played results in the game finishing, an indication of the overall winner should be presented to the user along with an option to update the database with the statistics of the game as previously described].

C Final User Stories

Check validity of each story.

Check validity of each test.

Check duration of each story.

Allocate actual duration to each story.

C.1 Sprint #1: Command Line Mode

S0010

Must Have

Est: 1 Act: 1

As a Human Player, I want to start the program in command line mode, so that I can either play a new game or see statistics of previous games.

S0010

BOB can start the program in CL and he is presented with a question of whether he wants to play a new game, view statistics or quit. The question asks Bob to enter “1” for a new game, “2” for viewing the stats, “3” to quit the game, and he must “press enter” to submit his choice. Acceptance tests: Try to press enter with empty string. Try with a random number other than “1”, “2” or “3” Try with correct numbers.

S0020

Must Have

Est: 2 Act: 2

As a Human Player, I want to start a new game, so that the first active player can select a category.

S0020

Bob is asked to select the number of AI opponents he would like to play against (max 4). He does this by typing a number and pressing enter. Acceptance Tests: Test numbers 1-4. Test numbers other than 1-4 inclusive. Test empty string. Test that deck is loaded. Test that deck is shuffled. Test that correct number of players are created. Test that players are dealt the correct number of cards. Test that one player is made the active player. Test that correct output is provided by standard out.

S0030

Must Have

Est: 1 Act: 2

As a Human Player who is selected as the active player, I want to select a category, so that the game can check the outcome of the round.

S0030

After the current round finishes, BOB can check the screen to find out who the winner was. The format is "X wins this round" with X being BOB or the AI player. The category values for each player are presented on the screen. Acceptance Tests: Play a round and check that the correct category values are shown for each player. Check that the correct player wins each time. Check that draw takes place if there are multiple highest cards. Check that correct player becomes/remains active player for next round. Check whether there is a message at the end specifying the winner.

S0040

Must Have

Est: 1 Act: 2

As a Human Player I want to know the outcome of the round,
so that the game can progress to the next round.

S0040

BOB will know if it is his or the opponents round without
having to do anything further. Tests: Check that after each
round there is an output mentioning the winner. Check that
user can continue to progress to new rounds even if eliminated.

S0050

Must Have

Est: 1 Act: 1

As a human player I want to choose to record game log information in a file when I start a game in command line mode.

S0050

BOB can navigate to the game folder and see the game log.
(Not sure if the user has the option NOT to log the game information) Acceptance Tests: Initiate a game and set -t flag in the CL. After the game finishes check that there exists a log file in the game directory, named topTrumps.log. Write a test log file with known content. RE-run the game and check that the original file is overwritten with the correct content. Check the content of the file as per C6030- C6100 requirements.

S0130

Must Have

Est: 1 Act: 2

As a Human Player, when not the active player for a round, I want the active AI player to select a single category which has the highest or joint highest value for their topmost card.

S0130

While developing, have the system.out print all the values for the AI player, i.e. the contents of their top most card. Upon selecting a category, have a system.out for the category selected and the value. Compare with the contents of the whole card to make sure that the highest value category is always selected by the AI player.

S0180

Must Have

Est: 2 Act: 2

Bob and Lilly are really into statistics and they want to keep track of the game stats after a game ends. As experienced players they want to know how many draws were in the game, how many rounds were played, how many rounds were won by each player (with clear indication) and who won the game. They don't want to press anything for this procedure, it must be automated.

S0180

Query the Database to make sure everything has been recorded.

C.2 Sprint #2: Online Mode

S0190

Must Have

Est: 1 Act: 1

Lilly, is known for her attitude of succeeding on everything, so she always watch her performance before starting a game (access the DB before a game starts). She wants to see the number of games played overall, how many times a human won and how many the “AI”, the average draws per game, the largest number of rounds played in a game.

S0190

?

S0200

Must Have

Est: 1 2 Act: 2

As a Human Player, I want to start the game in the online mode by using an -o flag either in terminal. I want to be able to start a new game in different tabs and play different games concurrently with 1-4 opponents. After I run the command in terminal, I can open a browser and initialise a game by typing in `http://localhost:7777/toptrumps`.

S0200

Try in terminal with -o flag and check for errors. If no errors occur, open browser and type in the `http`. Check multiple tabs to check that the initial selection screen is active. Check that the selection screen contains buttons for new game and statistics. Click on the Start Game button and check that the browser displays the correct number of game. Check that the human player is always present and that the AI players are present and not more than four. If number of AI players is not selected randomly, check that my input corresponds to the number of AI players on screen.

S0210

Must Have

Est: 1 Act: 1

As a Human Player, I want to be able to access the DB to view statistics upon clicking the Statistics button on the selection screen. This has the same functionality as in the Command Line mode as described in [S0180]. The difference is the way I will access this functionality.

S0210

Check that the Statistics button is responsive. How the data will be presented in the Online mode??????? Check that all games are stores to the DB, I.e all instances as represented by different tabs.

S0220

Must Have

Est: 1 2 Act: (incomplete)

As a Human Player, I want to be able to see that the game has actually started, and be guided through the steps of what I need to do next. I want to be able to have access on the game information that allow me to navigate through it, and be presented with my options if I am the active player, or have a visual on the storyline if an AI is the active player.

S0220

After clicking on the new game button check that the screen shows: Confirmation that the game has started, by showing for example "Deck has been shuffled and dealt". The round number I am currently playing Who the active player is. Check that I have the number of cards I have in hand, either on the card, or somewhere else on screen. My topmost card with the characteristics' values. A button that allows me to proceed to the next step. That is, to select my category if I am the active player or make the AI player select his. Check that if I am not the active player I do not have the option to select my category.

S0230

Must Have

Est: 1 2 Act: (incomplete)

As a Human Player, and after I have selected to proceed with the game after selecting the button as described by S0220, I want to trigger the current round. I want to be able to select my category as an active player, or “notify” the AI player that he needs to do the same, and in the end I would like to know the outcome of the round.

S0230

If I am the active player: Check that the button makes me select a category by popping a drop down menu or similar. Check that selecting my category the game proceeds to next step If I am not the active player. Check that the game continues by having the AI player selecting category. In both cases: After selection, check that I have the option to go to the next step, I.e. show the winner After the category selection, check that all cards with their values are visible on screen and that the winner is mentioned. Check that the same winner has the privilege of selecting the new category Check that the number of available cards in hand have been recalculated.

S0240

Must Have

Est: 1 Act: (incomplete)

As a Human Player, I want to have access to the game information if a draw has occurred. That includes to be aware that the outcome was a draw and the size of the communal deck. I want to be able to re-select if I was the winner of the previous round.

S0240

Check that in the case of a draw, the screen shows that clearly. Next check that there is an indication of the number of cards in the communal pile. Check that a button is present, allowing me to order the game to proceed to the next round. Check that the round number has increased by 1 after the draw and after I have selected to go on. Check that the category selector from previous round is the current selector. In the first winning case, display the number of cards in communal pile to make sure it is reset.

S0250

Must Have

Est: 1 Act: (incomplete)

As a Human Player, I want to know who the overall winner was.

S0250

Check that when the last round is finished, the name of the winner is shown. Have a summary of winning rounds as per video example???? TO DISCUSS. Make sure there is a button to return me to the selection screen, from where I can access the DB or play new game.

C.3 Sprint #3: Online Mode (Additional)

S0220

Must Have

Est: 4 Act: 4

As a Human Player, I want to be able to see that the game has actually started, and be guided through the steps of what I need to do next. I want to be able to have access on the game information that allow me to navigate through it, and be presented with my options if I am the active player, or have a visual on the storyline if an AI is the active player.

S0220

After clicking on the new game button check that the screen shows: Confirmation that the game has started, by showing for example "Deck has been shuffled and dealt". The round number I am currently playing Who the active player is. Check that I have the number of cards I have in hand, either on the card, or somewhere else on screen. My topmost card with the characteristics' values. A button that allows me to proceed to the next step. That is, to select my category if I am the active player or make the AI player select his. Check that if I am not the active player I do not have the option to select my category.

S0230

Must Have

Est: 4 Act: 4

As a Human Player, and after I have selected to proceed with the game after selecting the button as described by S0220, I want to trigger the current round. I want to be able to select my category as an active player, or “notify” the AI player that he needs to do the same, and in the end I would like to know the outcome of the round.

S0230

If I am the active player: Check that the button makes me select a category by popping a drop down menu or similar. Check that selecting my category the game proceeds to next step If I am not the active player. Check that the game continues by having the AI player selecting category. In both cases: After selection, check that I have the option to go to the next step, I.e. show the winner After the category selection, check that all cards with their values are visible on screen and that the winner is mentioned. Check that the same winner has the privilege of selecting the new category Check that the number of available cards in hand have been recalculated.

S0240

Must Have

Est: 2 Act: 2

As a Human Player, I want to have access to the game information if a draw has occurred. That includes to be aware that the outcome was a draw and the size of the communal deck. I want to be able to re-select if I was the winner of the previous round.

S0240

Check that in the case of a draw, the screen shows that clearly. Next check that there is an indication of the number of cards in the communal pile. Check that a button is present, allowing me to order the game to proceed to the next round. Check that the round number has increased by 1 after the draw and after I have selected to go on. Check that the category selector from previous round is the current selector. In the first winning case, display the number of cards in communal pile to make sure it is reset.

S0250

Must Have

Est: 1 Act: 1

As a Human Player, I want to know who the overall winner was.

S0250

Check that when the last round is finished, the name of the winner is shown. Have a summary of winning rounds as per video example???? TO DISCUSS. Make sure there is a button to return me to the selection screen, from where I can access the DB or play new game.

D Screenshots