Top Trumps Software Development Project

By: Chris Castaldo

Table of Contents

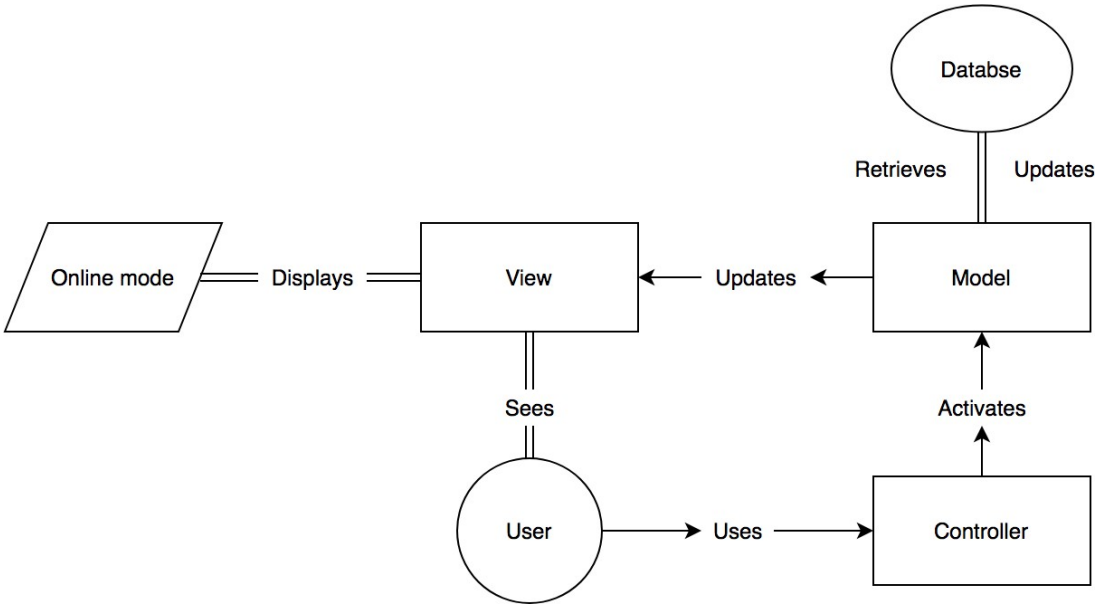
Introduction:

The following document is a written technical report outlining the development process of the Top Trumps Software Project. Included within is a summary of the project's requirements and design, a description of the iterative development process, proof of testing, and the programs shortcomings and deficiencies. The technologies used during the development lifecycle include:

- Microsoft Teams for communication
- GitHub for code warehousing
- Trello for task management
- Draw.io for design

Methodology:

This project was developed using the Scrum methodology. The development lifecycle spanned from 15/01/2020 – 17/02/2020. In this 33-day time span I completed two sprint iterations. Both sprints lasted two weeks. The first sprint focused on the planning and implementation of the programs back-end functionality. The second sprint focused on front-end interface. The software was build using an MVC (Model-View-Controller) structure. Below is an outline of each sprint and the software architecture.



Sprint 1: Database and Command Line

Ideal Days	Target Number User Stories	Actual Number User Stories
15	**?**	**?**

Sprint 2: Online Mode

Ideal Days	Target Number User Stories	Actual Number User Stories
15	**?**	**?**

Development

The development phase started with identifying and defining the user stories. This section will break down those user stories for both of the sprints. They will be categorized by the priority of their responsibilities and the role they play in ensuring the functionality of the program.

Sprint 1: The Database and Command Line Epic

Must Haves: These are stories that include methods that are integral to the functionality of the program.

- **Card Object and Card Deck**: A method(s) that establishes attribute and name instances of each card, uses a scanner to assign the instances, and then places them into a 40-card deck.
- **Player Object and Player List:** A method(s) that establishes the attribute and name instances of each player and then places them into an array list.
- **Card Distribution:** A method that distributes cards to each player in the array list

- **Determine Turn**: A method that determines whose turn it is
- **Play Top Card**: A method that plays the top card in each players hand
- **Choose Attribute**: A method that allows the player to choose the attribute they want to play
- **Compare Method:** A method that compares the value of the chosen attribute of each card played and distributes them into the winning players hand
- **Database Storage and Retrieval:** A method(s) that stores and retrieves the results of the game from a database
- **AI Choice**: A method that tells the AI bots to choose their highest attribute during their turn

Should Haves: These are stories that include methods that aren't necessarily integral to the functionality of the game but do increase the playability and functionality.

- **Deck Shuffle**: A method that shuffles the deck ensuring each game unique
- **Choose Number of Opponents**: A method that allows you to choose how many AI players you want to face.

Could Haves: These are stories that include methods that would add non-integral features.

- **AI Difficulty:** A method that allows the player to choose the difficulty level of the AI players
- **Point Mode**: A second game mode where players can wager points every round

Story Cards Not Implemented or Imprecisely Implemented

- AI Difficulty
- Point Mode
- Compare Method

Story Cards (Sprint 1):

#1 Card Object, Scanner. and Deck (Must Have)		
As a Developer I Want to: • Create a card object that stores all five attributes of each card • Assigns values to each attribute using a scanner • Create a deck that can store the card objects **As a Player I Want to:** • Be able to see the top card in my hand **Acceptance Criteria:** • Card Objects are successfully created and added to the deck • An instance in the deck array can be called form the command line and the associated card will be displayed		
Priority: High (Completed)	**Expected Effort:** 3 hours	**Real Effort:** 3 hours

#2 Player Object and Player List (Must Have)		
As a Developer I Want to: • Create a player object that stores all the information of the user player and AI • Add the player objects to an array list **As a Player I Want to:** • Choose the number of AI players to participate in the game **Acceptance Criteria:** • Player objects can be called from their array index • The number of active players can be chosen from the command line		
Priority: High (Completed)	**Expected Effort:** 1 hours	**Real Effort:** 1 hours

#3 Card Distribution (Must Have)		
As a Developer I Want to: • Distribute the cards of the deck instance to the active players **Acceptance Criteria:** • Cards are evenly distributed amongst the players		
Priority: High (Completed)	**Expected Effort:** 2 hours	**Real Effort:** 2 hours

#4 Determine Turn (Must Have)		
As a Developer I Want to: • Create a method to randomly assign the first round to a player **Acceptance Criteria:** • First turn is randomly determined		
Priority: High (Completed)	**Expected Effort:** 1 hours	**Real Effort:** 1 hours

#5 Get Top Card (Must Have)		
As a Developer I Want to: • Create a method that choose the top card in each player's hand **Acceptance Criteria:** • The top card in each player's hand array is accurately drawn by the method		
Priority: High (Completed)	**Expected Effort:** 1 hours	**Real Effort:** 1 hours

#6 Compare Attributes (Must Have)		
As a Developer I Want to: • Create a method(s) that compares the players or computers chosen attribute to that of the other players and chooses the one with the highest value **Acceptance Criteria:** • Values are accurately compared and the rightfully winner is consistently returned		
Priority: High (Completed)	**Real Effort:** 5 hours	**Real Effort:** 5 hours

#8 Deck Shuffle (Should Have)		
As a Developer I Want to: • Shuffle the array instances of the deck **Acceptance Criteria:** • Each game results in different cards being distributed		
Priority: High (Completed)	**Expected Effort:** 1 hours	**Real Effort:** 1 hours

#9 Round Win Conditions		
As a Developer I Want to: • Compare attributes from top cards to find the winner of a particular round **Acceptance Criteria:** • If our testing can show that the player with the highest attribute of the type selected is returned as the winner of that round		
Priority: High (Completed)	**Expected Effort:** 2 hours	**Real Effort:** 4 hours

#10 Card Assignation (both for draws and clear wins)		
As a Developer I Want to: • Assign the top cards of each hand to the winning player • If draw, assign cards to communal pile to await the winner of the next round **Acceptance Criteria:** • If our testing can show that the player with the highest attribute of the type selected is returned as the winner of that round		
Priority: High (Completed)	**Expected Effort:** 2 hours	**Real Effort:** 4 hours

#11 Game Win Conditions		
As a Developer I Want to: • Tell when a player has been eliminated from the game • Tell when a player has won the game **Acceptance Criteria:** • When our testing shows that players are eliminated at the appropriate times and that the correct player is announced winner at the end of the game		
Priority: High (Completed)	**Expected Effort:** 2 hours	**Real Effort:** 2 hours

#12 AI Choice of Attribute		
As a Developer I Want to: • Get the AI to pick attributes competitively from their top card **Acceptance Criteria:** • When our testing shows that the AI is selecting the highest attribute from their top card		
Priority: High (Completed)	**Expected Effort:** 2 hours	**Real Effort:** 2 hours

Story Cards (Sprint 2):

Online Mode:

#1 View the Game on a Web Browser (online mode)		
As a user I want to: View the starting position of the game, with a landing screen, a statistics page and the ability to navigate between the three **Acceptance Criteria:** User loads the web page and GUI interface with option to play the game is displayed User can navigate between the three primary pages		
Priority: High (Completed)	**Expected Effort:** 10 hours	**Real Effort:** 30 hours

#2 Display Game Statistics (online mode)		
As a user I want to: View statistics of past games played **Acceptance Criteria:** User loads the web page and GUI interface with option to play the game is displayed		
Priority: High (Not completed)	**Expected Effort:** 10 hours	**Real Effort:** 20 hours

#3 View Top Card of User Player (online mode)		
As a user I want to: View my top card and be able to select an attribute for the round **Acceptance Criteria:** • User's turn is indicated by the system • Top card of active player is displayed • Attributes of top card are displayed		
Priority: High (Completed)	**Expected Effort:** 10 hours	**Real Effort:** 20 hours

#4 View Round Winner (online mode)		
As a user I want to: See the top cards for all the players at the end of the round So I can see the winner of the round **Acceptance Criteria:** • All players top cards is displayed with selected attribute at the end of each round • Winner of the round is identified		
Priority: High (Partially Completed)	**Expected Effort:** 5 hours	**Real Effort:** 10 hours

#5 View Game Winner (online mode)		
As a user I want to: View the game winner at the end of a game **Acceptance Criteria:** • Round information is displayed • Count of Deck for the winner of game is displayed Game winner's name is displayed		
Priority: High (Not Completed)	**Expected Effort:** 2 hours	**Real Effort:** 5 hours

#6 View Statistics (online mode)		
As a user I want to: View the statistics of the last game played and overall games played statistics **Acceptance Criteria:** Statistics of Current Game is displayed for user to view: • Number of draws in the game • Winner of the game • Number of rounds played in the game Number of rounds each player won		
Priority: High (Not Completed)	**Expected Effort:** 5 hours	**Real Effort:** 10 hours

#7 Display Game Over (Online Mode)		
As the API I want to: Allow user to end the game So a new instance can be started **Acceptance Criteria:** Successful close of game screen		
Priority: High (Not Completed)	**Expected Effort:** 2 hours	**Real Effort:** 5 hours

Command Line Mode:

#8 View Statistics (Command Line Mode)		
As a user I want to: View the statistics of the last game played and overall games played statistics **Acceptance Criteria:** Statistics of Current Game is displayed for user to view: • Number of draws in the game • Winner of the game • Number of rounds played in the game • Number of rounds each player won • Average number of draws in all games • Highest number of rounds across all games • Total number of human wins • Total number of AI wins		
Priority: High (Completed)	**Expected Effort:** 5 hours	**Real Effort:** 10 hours

#9 Expand the View Class (Command Line Mode)		
As a user I want to: View game progress through the command line **Acceptance Criteria:** Game is understandable and coherent through command line mode through effective implementation of a view		
Priority: High (Completed)	**Expected Effort:** 5 hours	**Real Effort:** 5 hours

#10 Print Test Log		
As a user I want to: Output critical game data for analysis **Acceptance Criteria:** When I can successfully print a game log that tests every dimension of our game to ensure it works effectively		
Priority: High (Completed)	**Expected Effort:** 5 hours	**Real Effort:** 5 hours

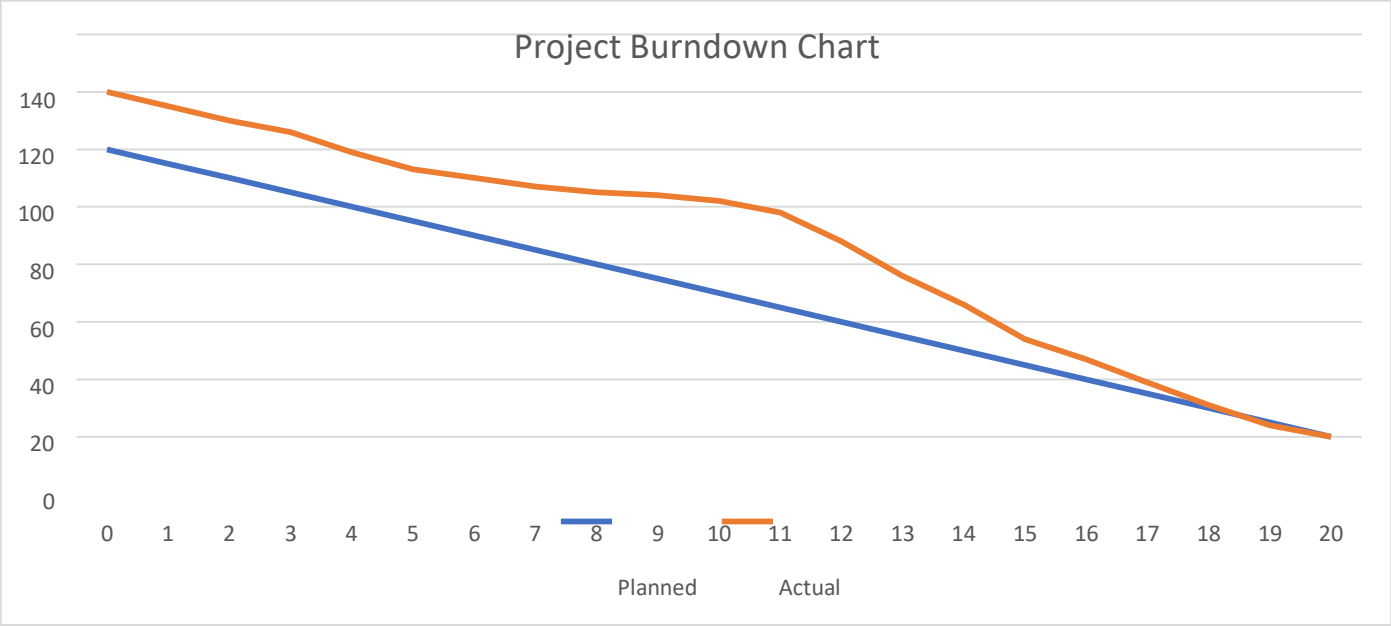
#11 Connect Java to Database (Command Line Mode)		
As a user I want to: Access the SQL database to view statistics from the command line mode **Acceptance Criteria:** When I can successfully view the correct statistics in the SQL database		
Priority: High (Completed)	**Expected Effort:** 2 hours	**Real Effort:** 5 hours

Sprint Planning and Review Reports (Combined)

Period	15 – 29 / Jan-20
Report Type	Sprint 1 Review
Planned Activity for Sprint 1	Complete implementation of Command Line Mode of the game Connect Database to the program
Planning Feedback	Prior to this sprint everyone had reviewed the coursework draft and the baseline template package to understand the task. I then developed high-level user stories from the human user's perspective in order to develop the classes to be built for the game.
	Initial User Story – 12
	Total Completed – 12
	Uncompleted Story – 0
Expected Challenges	When the card is dealt and you win, how do you get the cards from other people and move it into another person's deck?
	How do you add used/won cards to the bottom of the deck and shuttle the rest down one space?
	Potential issues with 3 active players.
Actual Challenges	I initially had issues passing values of the statistic variables into the database
	How do you add used/won cards to the bottom of the deck and shuffle the rest
	I had a big chunk of code of over 400 lines which I needed to now break further into smaller classes for easier interaction within the M-V-C model
	Issues with the randomization method for shuffling cards
Resolutions	• Randomisation of shuffle cards was a simple fix using collections library I found a shuffle method which rearranges the order of an array list with only one line of code • When cards were moved between player hands it was simple to remove then add to the other players' hand. Communal pile acting as player hand also made this a simple process for filling and emptying • Arraylists automatically shift all remaining card indexes down by one • A slight change in the way the loop runs fixed the 3 player issue I began to refactor the code into a MVC class structure which streamlined the Model (although more refactoring would be required in Sprint 2)

Period	1 –13 / Feb-2020
Planned Activity for Sprint 2	Complete implementation of Online Mode of the game Compilation of Technical Report Testing of the Game
Planning Feedback	• No one in the team had knowledge of using JavaScript, APIs or HTML so a lot of prior reading had to be done before I could commence the conversion to the online mode of the game. • I knew this would greatly increase the workload compared to the previous sprint as I would have to simultaneously learn and implement the code.
	Online User Story – 20
	Total Completed – 17
	Uncompleted Story – 3 (Connection of Javascript)
	Missed Story – 2
	Defunct Story - 1
Expected Challenges	Learning HTML, JavaScript and APIs to the level required for effective implementation in two weeks
	Time Management: due to the increased workload, I expected a huge challenge in terms of individual members' bandwith (considering other university commitments)
	Developing a comprehensive Test Log
	Existing bugs and absent features yet to be fixed/written in Command Line Mode
Actual Challenges	The rapid learning of the key languages required for implementation of the online mode impacted the project much as I had expected.
	Time spent on the front end (learning and implementing) detracted from other key parts of the project, including the database and its interactions with the model.
	Issues with version control on GitHub were created by rapid individual development from team members own codebase without regular commits/merges (this was caused by inexperience – I began to learn version control halfway through the project, from other modules).
	Resolving bugs and implementing absent features from the first sprint did indeed turn out to be a challenge
Resolutions	Issues regarding learning new programing languages and the time burden created could only be dealt with by increasing hours spent on the project considerably compared to the first sprint.
	In order to resolve bugs and implement absent features from the first sprint, I diverted the attention of a key team member for the duration of the second sprint
	Although I had little experience in creating Test Logs of programs of this size and complexity, it turned out for the most part to be easier than expected.

Screenshots of Command Line:



Testing

Screenshots of Command Line:

When the program starts in command line the user is initially prompted to enter an integer between 1 and 3 to choose how they will proceed:

```
Bens-MacBook-Pro:jartest benscott$ java -jar TopTrumps.jar -c
--------------------
--- Top Trumps   ---
--------------------

Please choose whether you would like to view persistent game data = 0, play a new game = 1, or quit = 2
```

If the user chooses to play a new game they will then be given an option to select the number of players they wish to play against.

```
--------------------
--- Top Trumps   ---
--------------------

Please choose whether you would like to view persistent game data = 0, play a new game = 1, or quit = 2
1
Please enter an integer between 2 and 5 to select total number of players (including human)
```

The game will then begin, with the first round starting. A random player will be chosen to begin the game. The user will be presented with the player with the first choice, and their top card.

The next output shows which card attribute the player has selected.

Once the player has made their selection the card value of the chosen category is displayed for all players along with their top cards name. These values are compared and if a winner is found it outputs their player number and the value of the card with which they won:

```
Please choose whether you would like to view persistent game data = 0, play a new game = 1, or quit = 2
1
Please enter an integer between 2 and 5 to select total number of players (including human)
5
ROUND 1
Player 2's top card is:

Civilization: British Empire
Geographic Size: 9
Duration: 7
Population: 9
Antiquity: 4
Cool Factor: 4

Player 2's choice
Player 2 has chosen Geographic size

Player 1 has Mexican Empire with size 6
Player 2 has British Empire with size 9
Player 3 has Polish Empire with size 6
Player 4 has Dutch Empire with size 3
Player 5 has Umayyad Dynasty with size 8
Player 2 has the highest card for this round, with a value of 9
Press ENTER to continue
```

To proceed to the next round the user is then prompted to press enter. The player who wins the round will again make the choice of attribute in the subsequent round, and will continue until another player wins - even if there are draws:

```
Player 2 has the highest card for this round, with a value of 9
Press ENTER to continue

ROUND 2
Player 2's top card is:

Civilization: Zulu Kingdom
Geographic Size: 3
Duration: 3
Population: 2
Antiquity: 3
Cool Factor: 8

Player 2's choice
Player 2 has chosen Geographic size

Player 1 has Macedonian Empire with size 6
Player 2 has Zulu Kingdom with size 3
Player 3 has Mongol Empire with size 9
Player 4 has Ethiopian Empire with size 5
Player 5 has First-French Empire with size 4
Player 3 has the highest card for this round, with a value of 9
Press ENTER to continue
```

When the user has their choice, they will be prompted to enter the card attribute they wish to select from keyboard input, by entering an integer between 1 and 5:

```
ROUND 7
Player 1's choice
Your top card is:

Civilization: Aztec Empire
Geographic Size: 4
Duration: 4
Population: 4
Antiquity: 4
Cool Factor: 9

Please select an attribute. 1 for Geographic, 2 for Duration, 3 for Population, 4 for Antiquity, 5 for Cool Fac
tor.
```

The selected attribute is then used in the card comparison for this round:

```
Please select an attribute. 1 for Geographic, 2 for Duration, 3 for Population, 4 for Antiquity, 5 for Cool Fac
tor.
4
Player 1 has Russian Empire with antiquity 3
Player 2 has Spanish Empire with antiquity 4
Player 3 has Abbasid Caliphate with antiquity 5
Player 4 has United States with antiquity 3
Player 5 has Japanese Empire with antiquity 2
Player 3 has the highest card for this round, with a value of 5
Player 4 has been eliminated!
Press ENTER to continue
```

If a player is eliminated from the game a message is presented to the user to notify the Eliminated players will then no longer be included in any future rounds:

```
Player 4 has been eliminated!
Press ENTER to continue

ROUND 9
Player 3's top card is:

Civilization: Macedonian Empire
Geographic Size: 6
Duration: 7
Population: 4
Antiquity: 5
Cool Factor: 7

Player 3's choice
Player 3 has chosen Duration

Player 1 has Soviet Union with duration 3
Player 2 has Mexican Empire with duration 1
Player 3 has Macedonian Empire with duration 7
Player 5 has Japanese Empire with duration 3
Player 3 has the highest card for this round, with a value of 7
Player 5 has been eliminated!
Press ENTER to continue
```

If two players cards have the same highest attribute this will result in a draw. The user will be alerted to this and the same player will choose in the next round.

```
Player 2 has chosen Duration

Player 1 has Portuguese Empire with duration 7
Player 2 has Polish Empire with duration 7
Player 3 has Mongol Empire with duration 4
There is a draw and the cards will be added to the communal pile
Press ENTER to continue
```

```
Player 2 has chosen Duration

Player 1 has Portuguese Empire with duration 7
Player 2 has Polish Empire with duration 7
Player 3 has Mongol Empire with duration 4
There is a draw and the cards will be added to the communal pile
Press ENTER to continue

ROUND 12
Player 2's top card is:

Civilization: Dutch Empire
Geographic Size: 3
Duration: 6
Population: 4
Antiquity: 4
Cool Factor: 5

Player 2's choice
```

The game will continue until only one player remains.

When this happens, the winner will be announced.

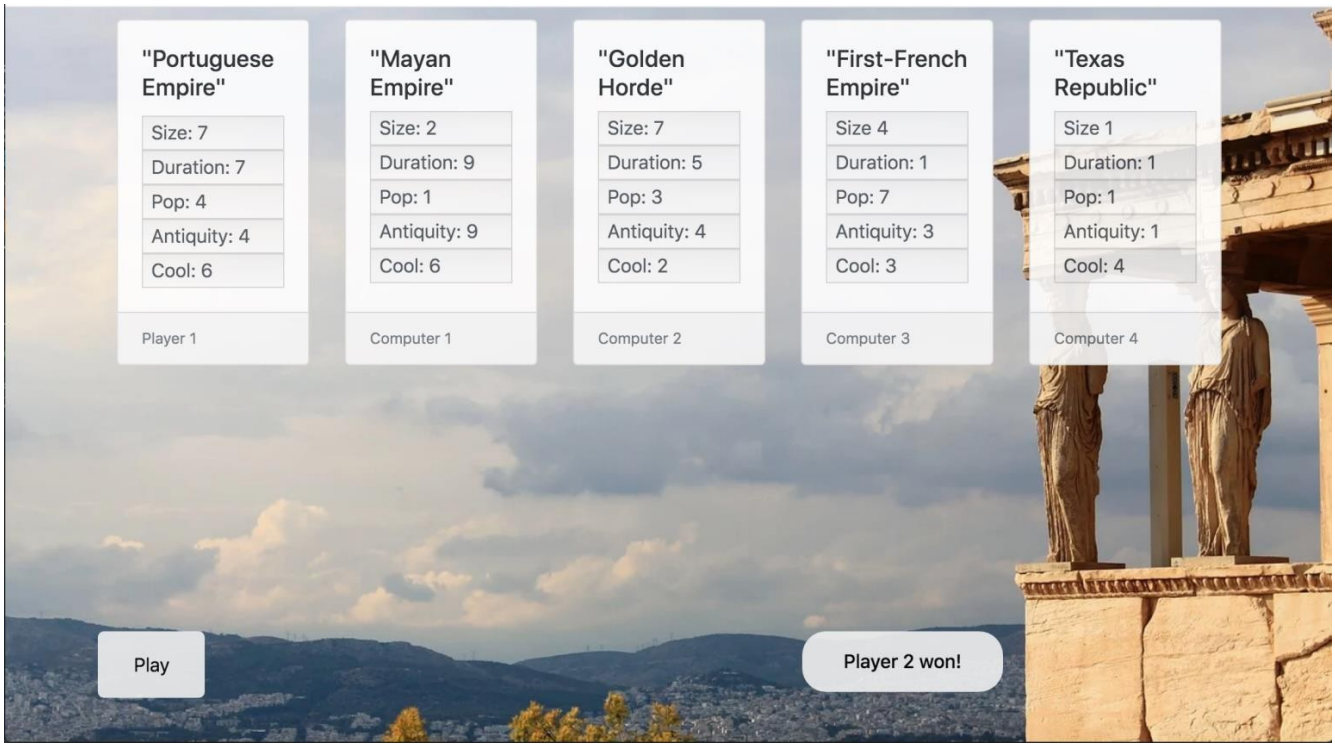
The game will then end, and the player will be prompted to select whether to start a new game, quit, or view the persistent game data:

```
Player 2 has Abbasid Caliphate with duration 8
Player 3 has Ethiopian Empire with duration 1
Player 2 has the highest card for this round, with a value of 8
Player 3 has been eliminated!
Press ENTER to continue

Player 2 has won the game

Please choose whether you would like to view persistent game data = 0, play a new game = 1, or quit = 2
```

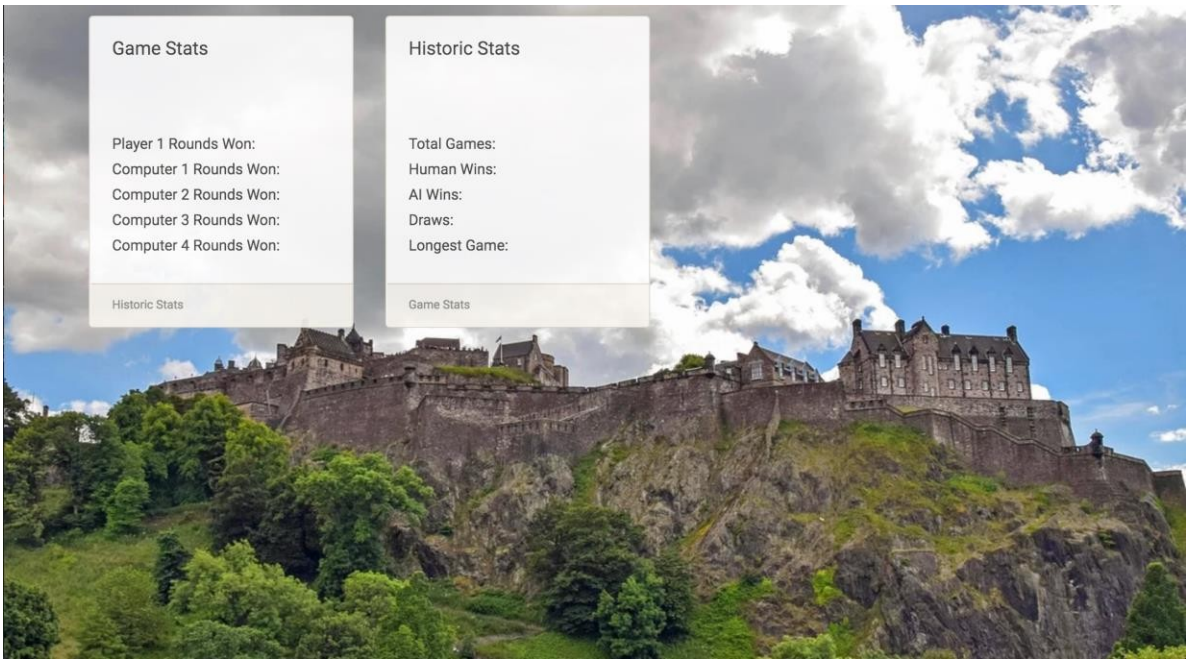
Online Mode:

Below, is the selection page (background reflects the card theme of Civilizations). Two buttons appear, one leading to the statistics page and the other to the play page:

Below is the card page (displayed as if the player number selection was disabled). Here we have the top card from each player alongside their attributes: a round has been played and player 2 has won!



Here is the stats page. Here, the last game stats and the historic stats are displayed on different cards:



TEST CASES

To test the command line application, I compared test log outputs to the command line output.

The testing done was built on the requested test log data, and to ensure the integrity of this data the command line output was compared to match the flow of the game.

- **Scenario 1** - I looked at the initial deck, as read from the text file input. In all cases this constituted a deck constructed with 40 cards. This matches what I expected and shows that the model method constructing the deck is working correctly.

- **Scenario 2 -** Next, I considered the output for the shuffled deck, once the cards had been placed into a random order. In all cases the 40 cards had been shuffled in position and in no discernible pattern between unique games.

- **Scenario 3 -** The next step of the game involves dealing the cards between the active players. The program was run to this point 3 times with each possible combination of number of players.
    - For 2 players, each player was dealt 20 cards each.
    - For 3 players, each player is dealt only 13 cards. This is incorrect as two players should have 13 cards, and one player should have 14. I had attempted to implement this using a conditional loop based on the remainder of players however this implementation does not seem to function as expected.
    - For 4 players, each player was dealt 10 cards.
    - For 5 players, each player was dealt 8 cards.

  The dealing works correctly in all cases where there are no remainders, however there is an issue with games involving 3 players where the remaining 40th card is not dealt correctly.

- **Scenario 4** - The top card for the player who will make a choice is displayed at the start of each round. When I first started testing the program, I noticed an issue where the top card would sometimes show as null. This was also reflected in the players hands when this happened. I noticed that in the test log, when it was being passed the player number integer from the controller it was drawing on the wrong array list in the model. When this was corrected all subsequent tests were correct and shows the players top card correctly and attributed the hands to the correct players.

 Scenario 5 - i also output the top cards of each player for each hand to the test log. In all tested cases the strings matched, and the correct category values were used in computing the round winner.

 Scenario 6 – I also assessed the contents of a player's hand after winning a round. In all examined cases I found no discrepancy between the expected outcome and the test log. This includes all rounds i checked which involved the communal pile.

 Scenario 7 - The communal pile also works correctly in all tested cases. We can see at the start of the following round that the correct number of cards have been added to the player who owns deck, and also that once a player has won its contents are empty in the following round with the correct number of cards distributed to the winning player.

 Scenario 8 - For each round the category selected is displayed and each player's value for their top card in that category are displayed. The command line output for this data matches the test log output, and the win conditions correctly use the same values to calculate wins and draws, and to pick the round winner.

 Scenario 9 - The winner is output at the end of the game. In all tested cases this matches between test log and command line.

- Issues Log
- Deficiencies

  While best efforts were made to ensure that the program was implemented as much as possible to the specifications of the task, I still had some identified deficiency which is outlined below. I was not able to determine how the loop will perform when the program has to run for elongated round

Deficiencies

Front end functionality:

Although I managed to produce the necessary web pages in a static form, I struggled to proceed beyond the initial round in terms of card selection and winner allocation in terms of dynamic interaction. This was a consequence of the lack of experience with front end development (no team members had any when the second sprint began) and the very short timeframe in which I attempted to learn. Since the limitations were due to knowledge, it is difficult to say exactly what it would take for us to improve/expand upon these functionalities. However, from a user story perspective, these are the functionalities left to complete:

- Round winner card allocation and display
- Game winner display
- Statistics for last game and overall game displayed on statistics page
- Number of players to be selected does not work in GUI: in order to progress beyond this screen, I have to select the number of players in the command line. However, if this functionality is removed, the game screen does itself load and the first round of the game can be played.

In order to solve these issues, I would first need a research period in which I perform a deeper dive into HTML, JavaScript and API implementation to undercover the structural issues which are likely to have limited us.

Class Structure in Command Line (primarily the limited View Class)

I created a view structure for displaying feedback to the user in the Command Line mode, however some of that user feedback is retained in the controller and model. Ideally, I would take time to restructure rhe classes, streamlining the controller and model and putting more user feedback into the view.

Refactoring of Model

Ideally, I would refactor large sections of the model into objects in order to simplify the code structure and reduce inefficiency (among the inefficient aspects of the model are the multiple array lists used for a number of methods operating on the players).

Version Control

Due to it being the team's first time using version control, it was not as optimized as it might have been. Versions were merged clumsily – often with copy and paste rather than the GitHub functionality – and the branches were created irregularly and with little discussion/agreement over best practices among the team. Individual developers were often

working from versions of their code kept only on their own laptops, and so finally combining the work was very challenging.