An Online Multiplayer Version of Balderdash
ELEN4010: Group 5

Alexandros Kastanos (99004322), Sanvir Dessai (811790), Evan Rubin (598561), David Beder (811071)

School of Electrical & Information Engineering, University of the Witwatersrand, Private Bag 3, 2050, Johannesburg, South Africa

Abstract: The development of an online multiplayer version of Balderdash is undertaken using agile project management and continuous integration techniques based on the scrum methodology. The tools and processes employed to achieve this, as well as the functional objectives, are described throughout this report. This report is intended to be read in conjunction with the other documents in the artefacts directory of this project. The final Minimum Viable Product (MVP) release of the website application is hosted on the Heroku Cloud Service at `https://balderdash-group-5.herokuapp.com`.

Key words: Balderdash, Scrum, Github, Javascript, Travis Continuous Integration, Heroku

1. Introduction

The creation of this multiplayer game is inspired by the classic board game, Balderdash. Agile project management and continuous integration techniques are utilised in the four week development process. The management structures and product cycle iterations are described throughout the report to follow. Reference [1] provides the game rules for a traditional Balderdash game. Throughout the development process of this game, Github was used for version control while Trello was used for developer coordination and project management. Furthermore the project incorporated unit, continuous integration, and acceptance testing using Mocha and Chai, Travis CI, and CodeceptJS with Selenium.

2. Git Workflow

A "fork and branch" git workflow which is similar to that described in reference [2] has been adopted. Each developer forks the shared *witseie-elen4010-2017/Group-5-Lab* repository and clones that forked version onto their local device. This allows the developer to commit and branch as they please while using their forked repository for version control. Hence greater freedom is given to the developers while they work on their forked versions but as soon as any attempt is made to merge code into the shared repository, all the coding conventions, testing standards and other checks are imposed. More about these tests and conventions are expanded upon in a later section of this report. When a significant amount of work is done or a git issue is resolved a pull request is made from the forked repository to the shared *witseie-elen4010-2017/Group-5-Lab* repository.

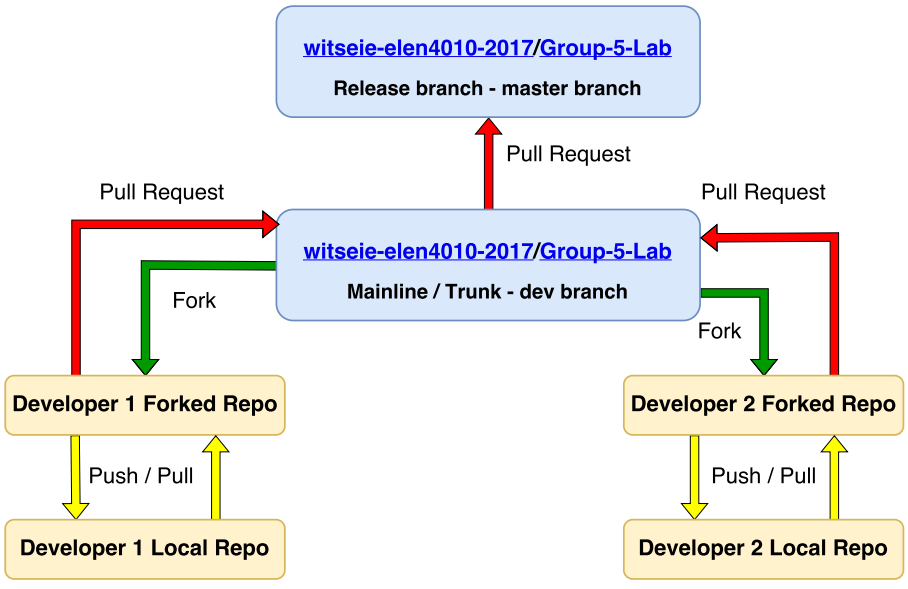

Figure 1 : An illustration of the Git-flow pattern adopted over the course of the game development.

As is clear from the diagram, the *dev* branch on the shared repository acts as the trunk or mainline. This allows for trunk-based development to be achieved with greater flexibility than a git branching workflow which is based upon each developer working in a dedicated branch on the shared repository.

3. Prioritising a Minimum Viable Product

The process of prioritising a Minimum Viable Product (MVP) for this project was initiated by identifying epic user stories. These stories were extracted from the functional requirements of the Balderdash game. Once these had been established, each epic story was sliced into thinner user stories. Trello, a collaboration tool, was then used to allow all members of the team to access and modify the stories via a board. Additionally Trello provides an easy to use interface for the story map which results from this process. The user map is contained in a single Trello board, epic stories are represented by lists, and user story slices are represented by cards within the appropriate list. A screenshot of part of the user story map is presented in figure 4 in the appendix. For greater insight on the user story map see the screenshots in the artefacts directory or navigate to the URL given in reference [3]. The Trello board provides additional information about the acceptance criteria of the user story in the description of each card.

In order to achieve an iterative development pattern, these user stories were prioritised in terms of the MoSCoW method. This method involves categorising user stories or features by one of the following labels: 'must have', 'should have', 'could have', or 'won't but would like'. All user stories which fall into the 'must have' category make up MVP. The team designated these stories by considering those stories which are vital to achieving a playable product which resembles the original Balderdash game. Table 1 below shows the labels applied on the Trello board to indicate which user stories are categorised by which MoSCoW category. Furthermore, a green label is used to indicate completed user stories.

Table 1 : MoSCoW categories and the corresponding Trello labels

MoSCoW category	**Trello Label Colour**
Must have	Red
Should have	Orange
Could have	Yellow
Won't but would like	Pink

4. Game Functionality and Code Technicality

4.1 Game Functionality

The final product functionality closely resembles the Balderdash board game. The game flow is depended on two main use cases: Creating a new game as a host and joining a game that is already hosted. The flow diagram representing the setting up of a game is given below in figure 2. On the main menu page, there is a link the a site with the traditional balderdash game rules. This makes it easy for new players to find out how the game works. In addition to this, the game provides helpful guides and explanations on the various help pages in the game. This makes the game a more user friendly experience. The user manual in the artefacts directory provides a guide for non-technical users, administration actions and developer perspective.

In figure 3 below, the flow diagram representing the game play logic is given.

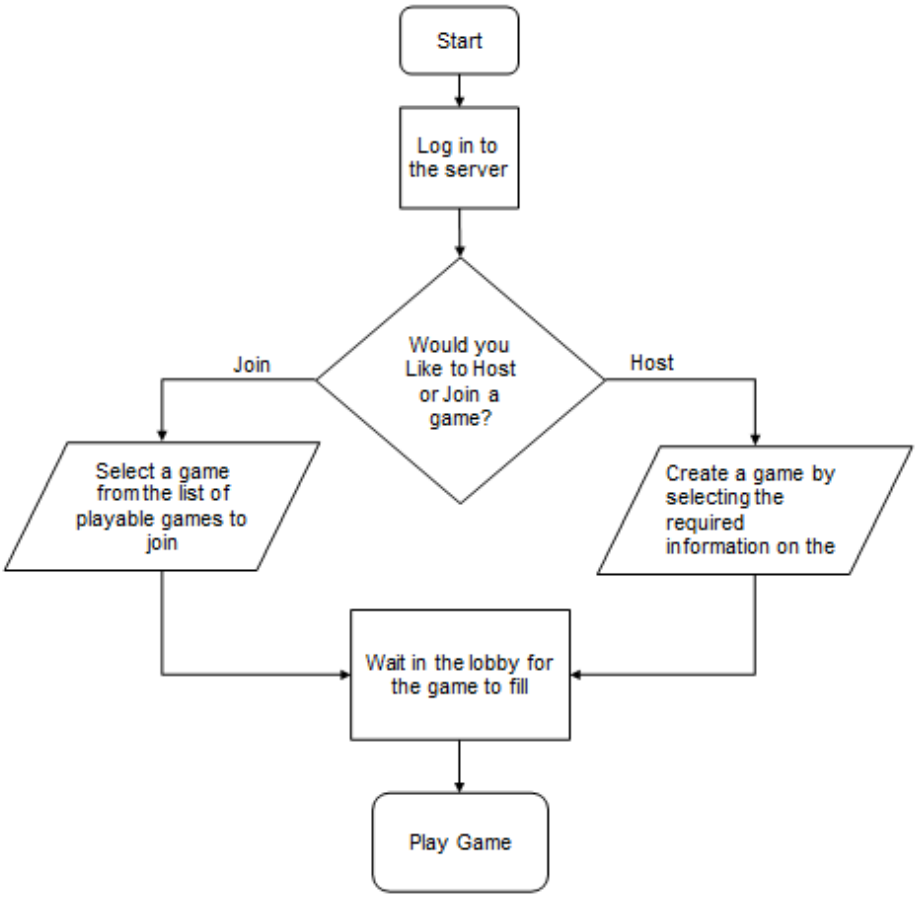

Figure 2 : Flow diagram showing the flow of game functionality

The above two flow diagrams graphically describe the game functionality. Additionally it is important to note that the dasher is given the dictionary definition of the word but must still input the definition to be used into a text box. This allows the dasher to reword the definition or even provide a bluff. This adds an extra dimension to the game and allows very formal dictionary definitions to be reworded in a more casual style.

4.2 Code Technicality

The code base revolved around two main file types: HyperText Markup Language (HTML) files and JavaScript (js) files. The HTML files act as the base for the user interface (UI), the views and front-end of the application, while the js files act as the controllers and models running the back-end of the application. The js files are furthermore used to add dynamic functionality to the HTML pages.

The application was designed as a client-server application. This was done to allow multiple users to access the same server for different objects simultaneously. The interaction between the client and the server was done through sockets. This infers that the client will send a HTTP request to the server through a socket, the server will process this request and respond to the client with the relevant response through the same socket and shortly following this interaction, close the connection.

AJAX is implemented in order to allow for the display to update without refreshing. This is particularly vital when it comes to the gameplay. Initially mouse movements were used to trigger these updates but this resulted in extreme lags and delays in the responsiveness of the game. As a result a change was made such that the pages update on mouse clicks. Although this is less initiative it greatly improved the functionality of the game. Additionally it was found that the game latency is less severe when playing on an Apple Macintosh computer.

For the admin page, a MD5 hash was used as given in reference [4]. This was done to ensure the security of the admin password. This is deemed a safe enough for the purposes of this project.

5. Code Dependencies and Javascript Modules

A dependency is any external programs which a piece of software requires to run. In this section the relevant dependencies and modules used will be listed, discusses and rationalised.

The dependencies of the Balderdash game are as follows:

- Javascript - used to code functionality into the games HTML pages.
- HTML - used to create a User Interface for a player to interact with the game,
- CSS - used the BOOTSTRAP library for styling the HTML pages.
- JQuery - used for various gameplay/user interface interactions.
- Node - used npm and express to link the Balderdash game created to the express server.
- socket.io - used to perform all communications between the javascript pages and the server.
- Heroku - used to host the Balderdash game on its servers.

The agile process, version control, continuous integration and testing require the following technologies:

- Git: Git/Github is used to collaborate as a group on the project, to allow for a version control of software released and create a form of peer software review.
- Trello: Trello is used to plan the process of creating the Balderdash game. Created two Trello boards, one for User Stories and one for Sprints where both these communications follow a scrum agile programming methodology. The User Stories board is a more business side approach to the problem and is used to illustrate to the developers what the customers is expecting from the software. Moreover, the Sprints board is a more technical document. It is used for the developers to communicate along as well as see what is to be and has been completed for each sprint within the production of the software.
- travis.ci: Travis Continious Integration is used to perform all unit and integration tests on the software every time a pull request was performed. It also was used to automatically deploy releases of the Balderdash game to the Heroku server.
- Mocha and Chai: These two dependencies are used in conjunction with one anther to perform all unit tests on the code. Mocha is the testing framework while Chai is an assertion library which allows for Behavior-Driven Development and Test Driven Development.
- CodeceptJS: This is used to run the automated acceptance tests.
- Selenium: A web browser automation tool used by CodeceptJS.
- WebDriver: Another web browser automation tool used by CodeceptJS.

The Javascript modules used in the game are as follows:

- LobbyGame.js: A module used to create a model of a game instance including all game specific details.
- playableGame.js: A module used to create a model of a playable game instance's functionality within the Balderdash game with functionality. However, owing to passing object issues most of this functionality was rather implemented in app.js as well.
- player.js - used to create a model of a player within the Balderdash game.

6. Testing

The project testing comprises three main types of tests. These are unit tests, continuous integration tests and acceptance tests. The proportion of project code that is tested by unit tests is fairly low as it proved difficult to test the client side code as much of it is embedded in HTML files. Code coverage can be obtained from the coveralls analysis in the latest Travis build or by running *npm run cover*.

6.1 Unit Tests using Mocha and Chai

As previously stated, the project code makes use of various modules. The main application also operates using several socket calls and returns. Unit tests were written to ensure the proper operation of all the modules and their behaviours. The majority of the socket functions are also tested. There are approximately 50 unit tests in total. These tests make up the bulk of the testing code and are run often on the developers local machines, when programming new features. The Mocha and Chai frameworks are used as the developers had prior knowledge of their use and they integrate easily with Node.js.

6.2 Continuous Integration Tests using Travis CI

The Github project repository is linked up to Travis CI. Every instance of a pull request from a developers forked repository to the shared project repository's 'dev' branch on Github initiates a Travis CI build. Travis then starts up a Linux virtual machine, installs all the project dependencies and executes the unit test code for each module as well as the server. Pull requests that do not pass the Travis build are either closed without merging or given additional commits in order to fix the broken segments of the code. This is done to ensure that any added code that is to be merged into the main project does not break the functionality of the existing code-base. Additionally, a coverage analysis takes place, which gives the developers an indication of the quantity of the project testing itself.

Travis CI is configured to automatically deploy a new release of the project if the release passes the test build and is successfully merged into the main project master branch. This ensures that the deployed website application is always in an operational state.

6.3 Automated Acceptance Tests using Codeceptjs and Selenium

Acceptance testing is done to present non-technical feedback to clients. These tests are slow and less numerous than unit test and integration tests. If the time frame for this project was extended the acceptance tests could be extended to play out an entire game of balderdash. This would provide good insight into the gameplay functionality. Currently the acceptance tests test the process of logging in, creating a game, joining a game and having a game start automatically when a lobby becomes full. The framework used is CodeceptJS which uses Selenium and Webdriver.io. Additionally, manual acceptance testing was carried out by the developers by giving the game to non-engineering students to play and test. This provided useful feedback on areas of the game which require improvement.

7. Scrum and Agile Process

A Scrum based methodology was adopted as the basis for the development process over the four weeks. This allowed for an agile process with four one week iterations to be planned. Each iterations, or sprint, has a tagged release which contains screenshots of the user story map, sprint backlog and a sprint retrospective. Each retrospective documents the sprint velocity, who was assigned to which sprint task as well as the expected time, expected velocity and actual velocity of each task. These documents can be found in the artefacts directory.

In order to assist with the agile process, a sprints board is set up in addition to the user stories board. The non-technical user stories are further decomposed into developer stories which are represented by cards in the sprints board. This board allows cards to be placed in one of five lists. These lists are: backlog, to do, in progress, to verify, and done. This allows developers to track the activities of other team members thus maintaining a level of co-ordination and organisation. Over and above the MoSCoW labels as described above, the agile process introduces new labels for Trello cards. This assists the team identify the time frame in which certain tasks need to be completed. These are listed in table 2 below. Additionally, developer story cards which correspond with a particular Github issue can be linked to that issue thus linking the activity on Github to the activity on the Trello boards.

Table 2 : Sprint labels used on Trello

Sprint Code	**Trello Label Colour**
Sprint 1	Light blue
Sprint 2	Dark Blue
Sprint 3	Purple
Sprint 4	Black
Removed from sprint (de-scoped)	Pink

7.1 Sprint Planning

As mentioned above, the team initially planned for four sprints, each being a week long. This schedule was maintained until midway through the second sprint when the team met and agreed that due the external pressures, it would be beneficial to extend the sprint into a two week period. This resulted in sprints three and four being reduced to four and three days respectively. This change in schedule was necessary in order to complete a sufficient number of user stories in the second sprint such that a potential user could give useful

feedback on the incremental change in the game since the first sprint.

Table 3 : The product backlog of the game user stories with key scrum parameters

Story	**Priority / Reward**	**Estimate Effort**	**Actual Effort**
As a user, I would like to login to the system so that I can identify myself in a game	3	1	1
As a user, I would like to change my profile picture	1	8	0
As a user, I would like to set and change my password	1	8	0
As a player, I would like to see my total wins	2	3	0
As a player, I would like to see my total losses	2	3	0
As a player, I would like to see my total gameplay time	2	3	0
As a player, I would like to start the game I am hosting	3	7	16
As a player, I would like to see who my opponents are	3	3	3.75
As a player, I want to select the number of players	2	1	2
As a player, the number of points needed to win the game	2	2	1
As a player, I need to be able to cancel a game before it starts	1	1	0
As a player, I would like to choose a game to join	3	4	6
As a player, I would like to see other game players	2	1	0.5
As a player, I would like to leave a game before it starts	2	2	0
As a player I would like to see the accumulative score of each player in the game	2	5	6
As a player, I would like to see the word selected for the round	3	4	3.75
As a player, I would like to know whose turn it is to be the dasher	3	4	1.5
As a player, I would like to see my opponents	3	2	2
As a player, I would like to know all the players current scores	3	2	1.5
As a player, I would like a graphical user interface	2	32	0
As a player, I would like to write my definition of the word of the round	3	5	3
As a player, I would like to see who has submitted their definition	2	3	0
As the dasher, I must be able to see the real definition of the world	3	7	2
As the dasher, I must be able to write my definition of the word of the round	3	4	2
As the dasher I would like to select the word for the round	1	9	0
As the dasher I need to select if the definitions provided are correct.	3	6	6
As a general player, I would like to see who won the round	3	1	1
As a general player, I would like to know whose turn it is to be the dasher next	2	2	0
As a general player, I would like to see the word selected by the dasher	3	2	1
As a player I would like to see if I won the game	3	4	1
As a general player, I would like to vote which definition is correct	3	7	16
As a general player, I would like to see all the definitions of the word given	3	5	4
As a general player, I would like to see the points awarded at the end of the round	2	2	2

The sprint timeline and basic sprint objectives are outlined in table 4.

Table 4 : An overview of all four sprints

Sprint	Duration	Main Objective
Sprint 1	Thursday 20th of April to Thursday 27th of April	Fully linked up HTML UI with CI and AWS
Sprint 2	Thursday 27th April 2017 to Friday 12th May	Set up environment to be ready for game mechanics
Sprint 3	Saturday 13th of May to Tuesday 16th of May	MVP
Sprint 4	Wednesday 17th of May to Friday 19th of May	Added features, optimisation, report and refining MVP

7.2 Sprint Backlogs

The sprint backlogs as provided in the end of sprint documentation are summarised in tables 6 to 9 in the appendix. The artefacts directory contains the original end of sprint documentation which was compiled weekly. Additionally, reference [5] is the link the the developer stories Trello sprint board. This is a dynamic view of the sprint backlogs and how over time sprint tasks moved from being in the 'to do' list to completion. Each task moves through the 'in progress' list to the 'Verification' before reaching the 'done' list.

7.3 Product Backlog

The backlog is visually shown on Trello on the *Sprints (Developer Stories)* board. This includes the following incomplete features:

- Cancel a game before it is created.
- Include coveralls badge on Github.
- Add a database to store player profiles and high scores, as well as game play history and a word list.
- Allow players to leave a game once committing to play.
- Display a round number at the start of each round.
  Additionally all completed user stories and the sprint tasks are listen in table 3 above.

7.4 Summary of Sprint Retrospectives

An evaluation method used in the scrum methodology of agile software processes uses two key parameters. These are effort points and reward points. Effort points describe the effort required by the development team in order to complete a user story. The team found a useful measure of effort to be the time spent working to complete the user story because it fits into the scrum ideology of delivering user stories quickly. As a result one effort point is defined to be approximately 30 minutes of work. In keeping with the scrum ideology of maximising user story delivery time, reward points are defined by the number of user stories completed. The reward points assigned to each user story is strictly related to the MoSCoW method mentioned above. A 'must' story carries the most reward with three points while a 'won't' carries the least reward with zero points. The point system is integer based and thus 'should' and 'can' stories are designated two and one reward point respectively.

These two parameters allow for the calculation sprint velocity which is a key measure of sprint efficiency. Sprint velocity is a simple relationship between effort and reward as defined by equation 1.

$$velocity = \frac{reward}{effort} \tag{1}$$

The individual sprint retrospectives, in the artefacts directory, provide greater detail on issues pertinent to each sprint as well as the velocities of each user story achieved during each sprint. Table 5 below is a summary of the scrum point system for the four sprints which took place over the four weeks.

It can thus be determined that the average sprint velocity for the overall project is 0.283. The sprint velocities in sprints one and two were fairly low because of the large amount of tasks which do not solve any user stories. Such tasks are required in order to set up the collaborative version control flow described earlier in this report as well as the continuous integration pipeline. Some areas of success throughout the four sprints include the way the team handled the agile process and the maintenance of the Github and Trello structures. Furthermore, it should be noted that the team underestimated the effort required to complete many of the sprint tasks. This is a

Table 5 : Scrum velocities over the four weeks

Sprint number	Total Sprint Effort	Total Sprint Reward	Total Sprint Velocity
1	16 points	3 points	0.1875
2	80 points	9 points	0.1125
3	48 points	22 points	0.4583
4	64 points	24 points	0.375

result of the networking between client and server code which often posed unexpected difficulties. Additionally balancing this work with other responsibilities often caused a discontinuity in sprint efforts. This had a negative effect of the sprint velocity achievable by the team. This was most evident during sprint 2. Sprint 4 suffered as a result of the large amount of documentation required and hence less user stories could be achieved. A better system for completing the documentation may have mitigated the impact of this on user story completion rate of the team.

8. Development Analysis and Recommendations

From a project management point of view, during the course of the project there were processes that worked well and led to the efficient design and implementation of user stories. These processes include: using Trello to coordinate developer activities, using pull requests for code reviews, using separate development forks for different application features, and having regular group meetings to discuss project activity. There were also processes that were counter-productive to implementation of user stories. One such issue was organising group meetings that fit the schedules of all developers. Other arrangements of the group members were not taken into account during initial planning and hence the velocity assumptions could have been better thought out.

From a technical point of view, there was a major issue encountered with hosting the website application. Initially it was decided to use the Amazon Web Service (AWS). However, after significant administrative problems it was decided to use the Heroku Cloud Service (HCS) instead. The time devoted towards the AWS set up could have been better used on other project activities such as improving test coverage. HCS was easier to use and set up than AWS. The HCS website hosting service is also free and does not require any credit card details. Thus it is recommended that for future projects HCS should be considered as oppose to AWS. Furthermore, future work on an online multiplayer version of balderdash should aim to incorporate a database of game words or an API to make adding new words easier as opposed to the hard-coded Javascript array implemented in this project.

9. Conclusion

An online multiplayer version of the classic board game, Balderdash is presented. The game is based on the original Balderdash however, some differences have been introduced. This programming project makes use of agile, scrum based project management processes. This is done through the use of Github and Trello as discussed throughout this report. Furthermore, the project incorporated unit testing, integration testing, and acceptance testing. Although there remain some gameplay bugs, the current release is sufficient for a MVP version of the game.

REFERENCES

[1] D. D. Limited. "Absolute Balderdash Game Rules.", 2016. URL `https://www.drumondpark.com/rules/absolutebalderdash`. Last accessed 17 May 2017.

[2] S. Lowe. "Using the Fork-and-Branch Git Workflow. Scott's Weblog.", January 2015. URL `http://blog.scottlowe.org/2015/01/27/using-fork-branch-git-workflow/`. Last accessed 17 May 2017.

[3] Group-5. "Trello user story map.", 2017. URL `https://trello.com/b/wOdUCnOI/user-stories`. Last accessed 19 May 2017.

[4] C. Coyier. "JavaScript MD5.", 2011. URL `https://css-tricks.com/snippets/javascript/javascript-md5/`. Last accessed 19 May 2017.

[5] Group-5. "Trello Sprints / Developer Stories.", 2017. URL `https://trello.com/b/ANziKPdR/sprints-developer-stories`. Last accessed 19 May 2017.

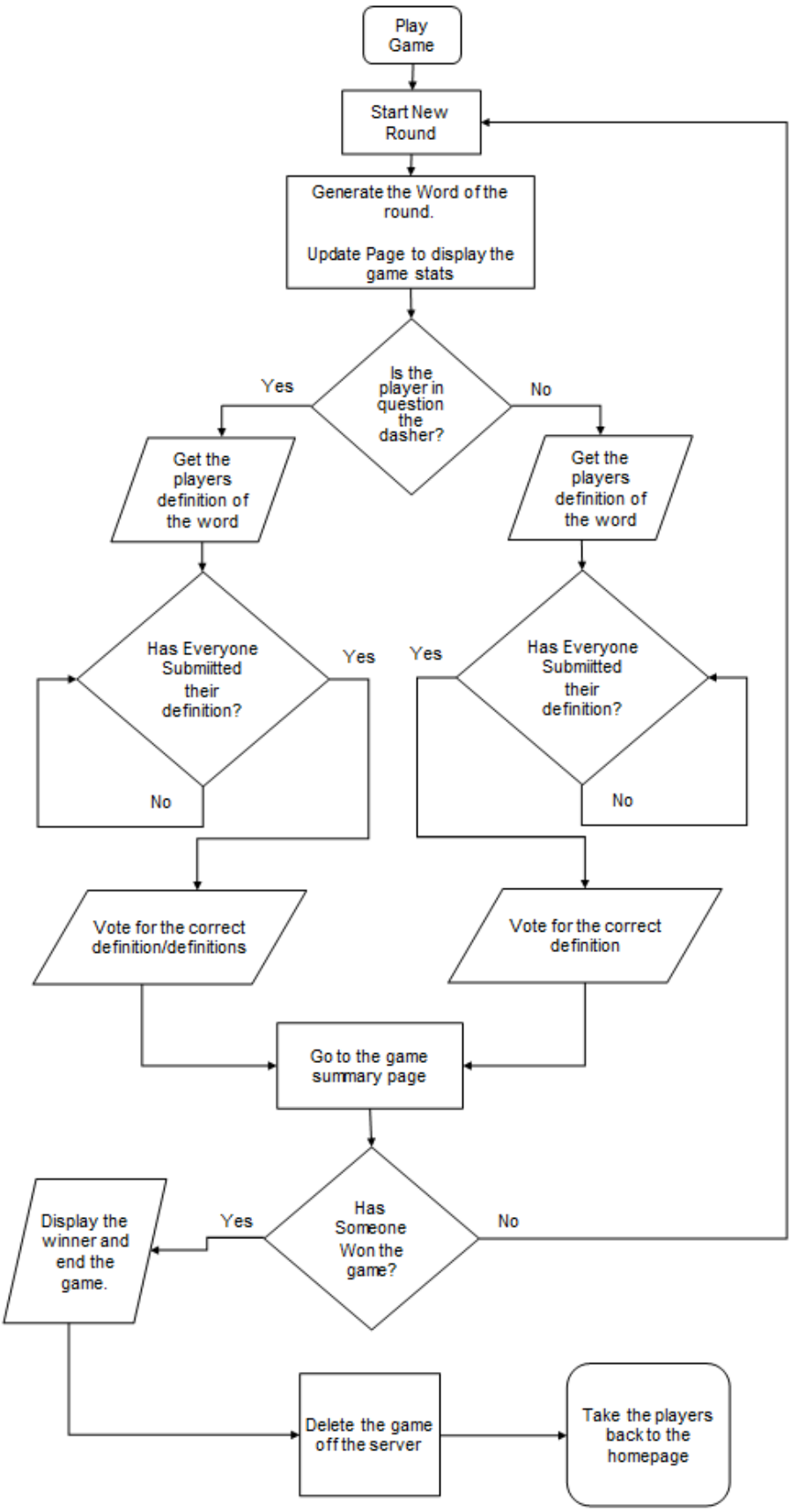

Figure 3 : Flow diagram showing the flow of game play functionality

APPENDIX

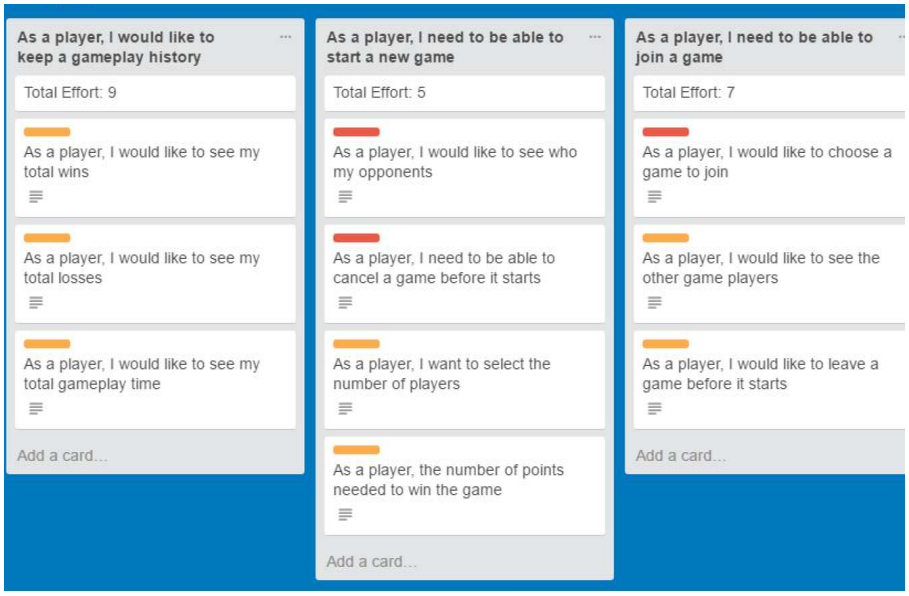

Figure 4 : A segment of the Trello user story map

Table 6 : The sprint backlog for Sprint 1

Story	**Sprint Backlog Tasks**	**Task Allocations**
As a user, I would like to login to the system so that I can identify myself in a game	Create sign in page	Rubin 598561
	Create locally hosted server for testing	Rubin 598561
	Add Javascript testing for server	Rubin 598561
	Create a host game page	Kastanos 99004322
	Create a main menu page	Dessai 811790
	Create join game page	Beder 811071
Tasks which are not tied to one particular user story	Discus Trello boards layout and set-up	All members
	Research online multiplayer games	All members
	Set up git repository and decide on file structure	All members
	Create link to game rules	Beder 811071
	Research and get familiar with Travis CI	All members
	Complete set-up for Travis CI	Kastanos 99004322
	Define the coding style conventions	All members
	Define a code review guide	All members
	Add artefacts directory	Kastanos 99004322
	Set up linting for Javascript	Dessai 811790
	Set up linting for HTML	Kastanos 99004322
Uncompleted tasks	Create AWS server to access	

Table 7 : The sprint backlog for sprint 2

Story	**Sprint Backlog Tasks**	**Task Allocations**
As a player, I would like to choose a game to join	Allow joining player to join an existing game	All members
	Allow joining player to join a dummy game (hard coded)	Kastanos 99004322
As a player, I would like to see the other game players	Functionality for player to see details of a game to join	Dessai 811790
As a player, I want to select the number of players in my new game	Create a player module for game logic	Dessai 811790
As a player, I want to choose the number of points needed to win my new game	Make front and back end create game environment	Rubin 598561 & Beder 811071
Tasks which are not tied to one particular user story	Link up HTML pages using Routing	Rubin 598561 & Beder 811071
	Add coveralls (no badges)	Kastanos 99004322
	Add a log out button/link for HTML pages	Rubin 598561 & Beder 811071
Uncompleted tasks	Create AWS server to access	
	Extend game environment to a playable form	

Table 8 : The sprint backlog for sprint 3

Story	**Sprint Backlog Tasks**	**Task Allocations**
As a general player, I would like to vote which definition is correct	Create the voting system to store a vote	Rubin 598561
As a player, I would like to see who my opponents are	Display all players in the lobby game in the lobby game page	Rubin 598561 & Beder 811071
As a player, I would like to know all the players current scores	Make a GamePlay HTML Page	All members
As a player, I would like to know whose turn it is to be the dasher	Select a dasher for each round	Rubin 598561 & Beder 811071
As a player, I would like to start the game I am hosting	Make a GamePlay HTML Page	All members
	Convert a lobby game into a dummy game page when the game is full	Dessai 811790
	Convert dummy gameplay page to a functional page	Kastanos 99004322
As a player, I would like to see the word selected for the round	Display the word of the round	Kastanos 99004322
As a player, I would like to see who my opponents are (in the game lobby)	Make a GamePlay HTML Page	All members
As a player, I would like to write my definition/explanation of the word of the round	Allow players to input their definitions via the keyboard	Rubin 598561 & Kastanos 99004322
Tasks which are not tied to one particular user story	Bug fixing and page re-factoring	Dessai 811790
	Set up and implement acceptance testing framework	Kastanos 99004322
	Research AJAX and implement automatic page refreshing where needed	Dessai 811790
Uncompleted tasks	Create AWS server to access	
	Major bugs still exist in the voting system	
	Display the "dictionary correct" definition to the dasher	
	Remove a player from the lobby game if they leave the lobby page	

Table 9 : The sprint backlog for sprint 4

Story	Sprint Backlog Tasks	Task Allocations
As a player I would like to see the accumulative score of each player in the game	Define point scoring system for in-game mechanics	Kastanos 99004322 and Rubin 598561 and Beder 811071
	Allocate points to each player	Rubin 598561 and Beder 811071
	Tally up the points and compare against winning point score	Rubin 598561
As the dasher, I must be able to see the real definition of the word	Expand the hardcoded dictionary of words and definitions	Kastanos 99004322
	Display the definition of the word of the round to the dasher	Kastanos 99004322
As the dasher, I must be able to write my definition of the word of the round	Display all definitions given for voting purposes	Rubin 598561
	Provide ability for players to input their definitions	All members
As the dasher I need to select if the definitions provided are correct	Provide ability for dasher to mark definitions as correct or incorrect	Rubin 598561
As a general player, I would like to see who won the round	Test someone reaching win score ends game and takes them to home page	Beder 811071
As a general player, I would like to see the word selected by the dasher	Display the word of the round	Kastanos 99004322
As a player I would like to see if I won the game	Add end of round page with functionality to start next round	Rubin 598561 and Beder 811071
As a general player, I would like to see all the definitions of the word given	Provide ability for players to vote on a definition	Kastanos 99004322, Rubin 598561 and Beder 811071
As a general player, I would like to see the points awarded at the end of the round	Add end of round page with functionality to start next round	Rubin 598561 and Beder 811071
	Multiple rounds in gameplay with new dasher and word each time	Rubin 598561 and Beder 811071
Tasks which are not tied to one particular user story	Debug slow gameplay	Rubin 598561 and Beder 811071
	Bug Fixes and Page Re-factoring	All members
	Create Heroku server to access	Dessai 811790
	Set up Travis CI to deploy to Heroku	Dessai 811790
	Add Functionality to delete information off server	Rubin 598561
	Expand the hardcoded dictionary of words and definitions	Kastanos 99004322
	If a lobby game is empty, it should be destroyed	Dessai 811790
	Technical Documentation (Report and Artefacts)	All members
Uncompleted tasks	Complete coveralls such that the badge is visible on Github	Kastanos 99004322
Tasks which were not started	Remove a player from the lobby game if they leave the lobby page	
	Get game round number showing and incrementing	