CM50109 Coursework 2

“Get Rich or Die Trying” Documentation

By the Grey Team: Cheuk Hei Chan (Ray), Kimberley Chong, Alex Gooding, David Hurst, Jessica Lok and Megan Slattery.

1.0 Abstract

This document details Grey Team's creation of 'Get Rich or Die Trying', a rogue-like game. It presents the various stages of analysis, specification, design and implementation undertaken during the software development life cycle. It also presents the group's experience of teamwork and the various communication tools used to ensure effective collaboration.

This project followed Scrum's guidelines. This document is therefore split into sprints. For ease, each sprint ran from Monday to Sunday. Within each sprint-section of this document, there are 3 sections. The first is titled 'Overview'. This contains a review of what was achieved during the sprint. It also includes a record of the team's sprint review. This is a more personal insight into what the team felt their strengths and weaknesses had been during the sprint. The second section is titled 'Process'. This records the various activities undertaken during the sprint. It features: the backlog; exception handling forms; and, records of the meetings and pair programming sessions held during the sprint. The third section is titled 'Products'. This contains any materials created while engaging in various agile practices and techniques. Examples include: user stories, use cases, CRC cards and testing records. Each product is given an individual version name. The first set of user stories, for example, is labelled User Stories 1. Any re-versions are labelled User Stories 2, 3 etc. Whenever a specific product (e.g. User Stories 1) is referred to within the document, its title and page number are referenced.

2.0 Week 5 30/10–05/11/17

2.1 Overview

In accordance with Scrum, we used our first sprint to focus upon the analysis stage of the coursework. To ensure the team had a thorough comprehension of the customer's requirements, we completed user stories and a customer interview analysis. A key challenge this week was also engaging with the human resources aspect of this project. Therefore, a portion of time was dedicated to introductions and distribution of roles.

Sprint Review 1 05/11 Attendance: Full. Length: 0.25 hrs.

Overall, the team was pleased with progress made. Especially with the clear choice of roles and efficient communication, enabled by social media. Yet, the team felt that team organisation had taken up a significant portion of time. It was hoped that in the next sprint we would be able to achieve more analysis and specification of the actual product. It was also noted that the user stories had not been entirely completed and formatted by the sprint's end. The team acknowledged that in future sprints they should finish all tasks within a strictly delineated period of time, as required by Scrum.

2.2 Process

Backlog 1

Task	Member to complete	Date set	Deadline date
Fill in Experience Form 1 with experiences and preferences.	Everyone.	30/05/17	31/05/17
Interview customer.	Everyone.	30/05/17	31/05/17
Write initial user stories on sticky notes.	Everyone.	30/05/17	02/11/17
Read Cockburn's guide to use cases.	Everyone.	30/05/17	02/11/17

2.2.1 Meeting Records

Meeting 1 30/10 Attendance: Full. Length: 1.45 hrs.

1	Introductions	Each team member introduced themselves. We discussed our various academic backgrounds and experiences with the different aspects of Coursework 2. R had experience with Java, K had previously created an online game. We decided to record preferences and strengths etc. in a table on Google Docs. M volunteered to write minutes for meetings, these were put on Google Docs.
2	Process	Discussed our knowledge of Scrum. This ensured that we all had a comprehensive understanding of its structure and key terms including backlog, sprint, Scrum Master etc. Decided to use GitHub to enable version control whilst coding. Also, chose to upload written documentation to Google Docs so that all products could be continuously integrated. This would ensure all team members could be aware of any contributions and developments.
3	Roles	Discussed who would take the roles of Scrum Master and Product Master. Decided that they would be chosen tomorrow, giving the team time to reflect.
4	Customer Interview Preparation	We began to prepare for the meeting with the customer. Team members noted the importance of gaining a comprehensive analysis and suggested many questions for the customer. These were transcribed into Google Docs. The team looked through the questions and marked those which were most important. Decided that D would lead the customer interview, others would ask questions when appropriate. We questioned whether we should record the customer, but decided M would note responses in same Google Docs document. This was titled Customer Interview 1 (4).

5	Tasks	We discussed how we should progress and looked to Scrum and Coursework 1 for guidance. We decided to write user stories during this sprint and confirmed that everyone knew the functionality of a user story. We also decided that everyone should look at Cockburn's writing on use cases so that the team could be prepared for the next stage of analysis.[1]

Meeting 2 01/11 Attendance: Full. Length: 1.75 hrs.

1	Roles	With reference to the document containing preferences and experience (4), roles were chosen. The team decided that the role of Scrum Master would be appointed on a rotating basis. As M was experienced with documentation and was already writing minutes, she would initially be product master.
2	Preparation for Customer Interview	We ensured that everyone was satisfied with the questions that the customer would be asked.
3	Customer Interview	The team asked the customer pre-prepared interview questions. The answers were transcribed into Customer Interview 1 (4), which was uploaded to Google Docs. The team confirmed a follow-up meeting for the next week.

Meeting 3 02/11 Attendance: Full. Length: 1.5 hrs.

1	Customer Interview Review	The team felt satisfied with the previous customer meeting. They were struck by the relative freedom they would have in the game creation. They noted the importance of creating a game which allowed everyone to use their strengths and past experience. K noted that the customer had recommended the use of Unity, if we were to use a game engine. After research, she however, found that Unity did not support their implementation language: Java. K said that she would look up different options. The team concluded that for next week's interview, the team should note questions in a specific Google Doc as they thought of them (Customer Interview 2, 12).
2	User stories	Decided to begin the meeting with writing user stories which documented the most basic requirements of the game. We came up with these stories as a team, while various scribes wrote them on sticky notes. Next, the team created user stories detailing more advanced features. These were collected in User Stories 1 (5).
3	Features	These user stories encouraged the team to analyse some features. K and D discussed mini-maps: whether we wanted one which displayed the entire map or just shows a certain portion. Questioned the distinction between passages and rooms: should they be distinct or should the game consist of a series of rooms? The team thought about the conditions for winning: does the game have a time limit; how many lives should a user get; could you win by collecting a certain amount gold; will the game have different levels? The discussion was concluded with the idea that these questions would be returned to when preparing future customer interviews and implementation stages.

[1] Cockburn, A., 2000. *Writing Effective Use Cases*. London: Addison Wesley.

4	Documentation	The team was reminded of the importance of commenting when coding. This would ensure that all pieces of code could be understood and modified by any team member. The team spoke about documentation more generally. They decided that, as well as creating an ordered system of files in Google drive for our documents, we should have a physical file within which each version of user stories, crc cards etc, would be stored. The team noted the importance of having physical tokens to aid process visualisation.
5	Roles	D volunteered to be Scrum Master. This was agreed.

2.3 Products

As detailed in Backlog 1 (2), the products created during this sprint were: resources for team working; user stories; and the recorded customer interview. These products were all crucial for the analysis stage of product development.

Experience form 1 31/10

This form was created to facilitate initial organisation of roles. It is also intended that in later sprints this will be referred to in order to monitor if team members are being given opportunity to develop upon previous experience. We believe individual development is a key benefit of teamwork.

Initial (username)	Experience and strengths	Prefered roles and tasks
A (ag2303)	Logical design, file management, coding.	System design and implementation.
K (klzc20)	Game development, GitHub documentation, C# and Python.	System documentation and implementation.
D (dh731)	Coding, GitHub, recording logs and version control, logical design.	Testing, coding.
R (chc217)	Java, C++, TDD.	Coding, testing.
J (jl2332)	Logical design, TDD, coding, team organisation.	Scrum master, coding, pair-programming.
M (mas250)	Report writing, team organisation, coding.	Product Master, coding, organisation.

Customer Interview 1 01/11

The team posed pre-prepared questions to the customer in a 10 minute meeting. The questions mainly centered around basic functionality, and hence reflect the sprint's objective of analysing the game's most fundamental requirements. The questions, and the customer's transcribed answers, are preserved in the following table:

1. **What is your main priority for the game? What are three key functional requirements?**
This question is too broad. Think about what you can do and how you can do it incrementally. It is important to think about priorities. For example, real time vs multiple players vs quality of graphical experience– which is most significant to the team? Also consider the team's own technical experience. Could choose to explore areas which you are competent in already. The process of creating the game is as important as product.

2. **Which platform should the game be on? Examples include: OS, mobile and web based.**
This also depends upon the experience which the group already has and what is going to give you the best chance of producing a functioning system. Also when making the choice you might not want to rely upon the skills of a single team member.

3. **Can you define a 'rogue-like' game?**
Essentially, it is the idea of having a character which wanders around a level and has various things to do on that level. For example, the character could interact with a simple AI mechanism which could be trying to kill it.

4. **Can we use a game engine?**
This depends on how much it would do for you. Do not use one for the sake of it. If it is helpful then it could be quite interesting. Have a look at Unity, for example.

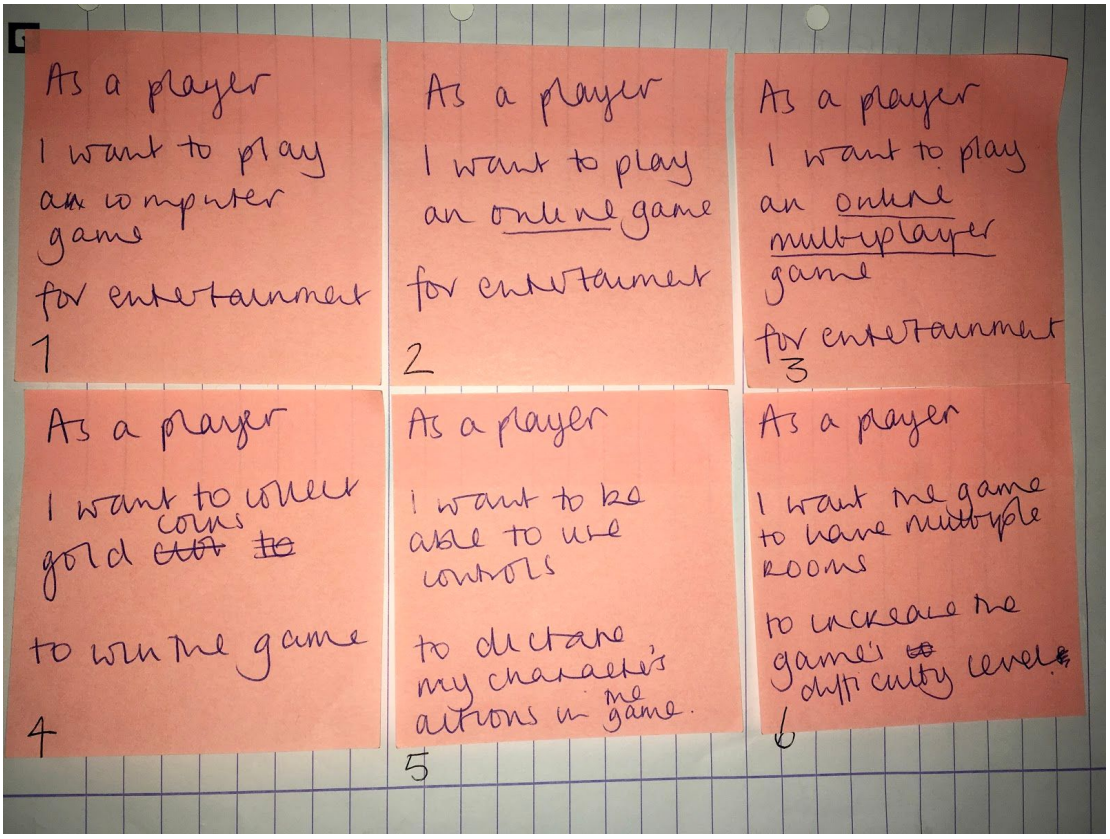
5. **Who is the game for? For example, do you have a specific age of player in mind?**
No, the game does not have to be for a particular audience.

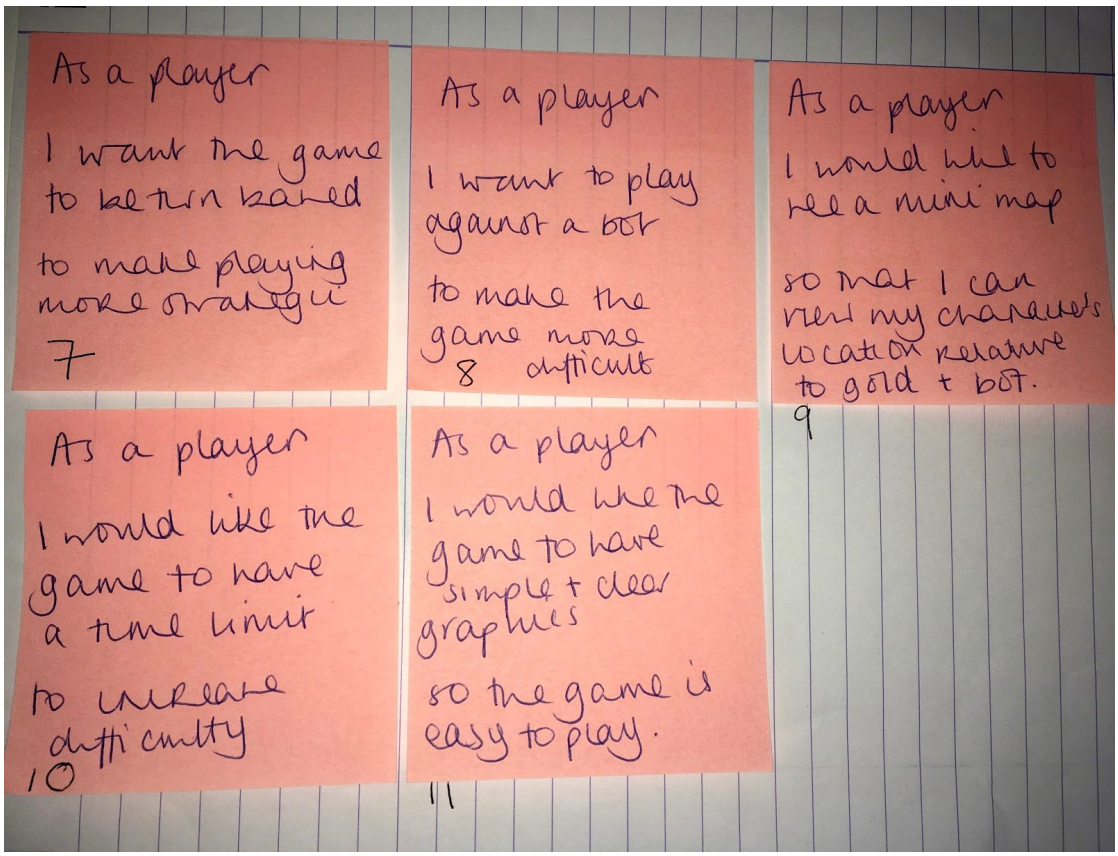
6. **Is procedural generation required?**
Yes, for two reasons. Firstly, a game which loads the same engine each time is not exciting for the player. Secondly, it would be tedious for the team to author a multiple maps itself. Making some hand drawn maps at the beginning of the process is fine. However, later it would be wise to build a program to generate new maps each time.

7. **Could you define 'turn based game'? Could a turn be a single motion or room completion?**
Meaning of 'turn' depends upon the game which you are creating. When playing against a bot in singleplayer mode, a turn could be a single movement. The bot would respond quickly and therefore rate of progress would not be necessarily slow. Also if you are playing against a bot, you need the game to be turn-based. It would otherwise be too easy to die. However, if you are providing an online multiplayer game, turn should be interpreted differently. The game should be in real time, and you should not have to wait for all other players to move before any one player can take a turn.

User Stories 1 02/11

These user stories were created with reference to the customer interview, held the day before. We created user stories on a just enough basis.


As a player
I want to play a computer game
for entertainment
1
As a player
I want to play an online game
for entertainment
2
As a player
I want to play an online multiplayer game
for entertainment
3
As a player
I want to collect gold coins
to win the game
4
As a player
I want to be able to use controls
to dictate my character's actions in the game.
5
As a player
I want the game to have multiple rooms
to increase the game's difficulty levels
6


As a player
I want the game to be turn based
to make playing more strategic
7
As a player
I want to play against a bot
to make the game more difficult
8
As a player
I would like to see a mini map
so that I can view my character's location relative to gold + bot.
9
As a player
I would like the game to have a time limit
to increase difficulty
10
As a player
I would like the game to have simple + clear graphics
so the game is easy to play.
11

3.0 Week 6 06–12/11/17

3.1 Overview

During this sprint we continued to analyse requirements and began to specify which features we would include in our initial product. To do this, we employed a variety of agile techniques and processes. These were: the creation of CRC cards, more user stories and a use case; the use of TDD; and a customer interview. In this sprint we began to think more seriously about documentation. We created a template within which progress could be formally stored. While this was maintained by the Product Master, it was uploaded to Google Docs. This ensured that all team members could reference and add any developments.

Sprint Review 2 12/11 Attendance: Full

The team was satisfied with the progress made in terms of analysis and specification. They noted that the CRC cards and TDD were especially useful for specification. They gave a clear picture of what the system would include. The second batch of user stories, however, were less useful at this stage. While it gave a clearer idea of future possibilities, it was currently more important for the team to develop on a just enough basis. The team also voiced regret at the human resources error that week, and noted the importance of all members attending future customer meetings. Looking forward, the team agreed that it was necessary to begin implementing the simplest possible system. Analysis and specification in Scrum should be incremental and doing anymore at this stage would be unnecessary. They decided that their focus for the next sprint would be lightweight design and implementation.

3.2 Process

Backlog 2

Task	Member to complete	Date set	Deadline date
Make User Stories 2.	Everyone.	06/11/17	06/11/17
Create CRC cards 1 and 2.	Everyone. Further developed by A.	06/11/17	08/11/17
Form framework for team submission.	M.	06/11/17	10/11/17
Interview Customer.	M, R, J.	08/11/17	10/11/17
Create Use Cases 1 and 2.	M, R, J. Use Case 2 completed in detail by J.	10/11/17	10/11/17
Create template for testing records.	D.	10/11/17	10/12/17

Fill in Testing Records 1.	D.	10/11/17	12/11/17

Exception Handling 1 12/11

Discussed in: Sprint Review 2 (7).

Exception: Postponement of customer interview and partial absence of team during rescheduled interview.

Handled: The team should reprioritise their understanding of important tasks in the unit. Each member should understand that interviews are essential to this process. This is due to the analysis and advice which are gained from interviews, and also the fact that the customer is central to any agile software development. Beyond this re-evaluation, the team also decided that J would be Scrum Master. J is organised and therefore would ensure that no meetings would be missed from now. To make amends with the customer, those team members absent from the meeting would email the customer to apologise for wasted time.

3.2.1 Meeting Records

Meeting 4 06/11 Attendance: Partial (M ill). Length: 1 hrs.

1 GitHub — To prepare for later programming, each team member created a GitHub account. K and D shared their experience with GitHub with the rest of the team.

2 User Stories — Decided that the previous user stories had not given the team enough options for a product backlog. The first ten minutes of the meeting were spent creating user stories for more advanced features. These were collected, to be formalised by M in User Stories 2 (9).

3 CRC cards — The team began to think about how these user stories would be implemented. After referring to the Scrum framework, the team decided that making CRC cards would structure this discussion. The CRC cards detailed and clearly labelled the most fundamental objects of the system. Their creation helped the team recognise the significance of using an object oriented language for their project. As the CRC cards were not completed during this meeting, A volunteered to finish them later. They were collected in CRC 1 and 2 (10).

4 Customer Interview — After discussing their progress, the team thought that they did not have any questions to ask the customer. Due to this, and a team member's absence, the team decided to postpone meeting with customer.

5 Tasks — The team questioned how they were to progress. They noted that while a lot of material to document the software development process had been recorded, this had not been formally stored. They decided that M, as the Product Master, should formalise the notes and create a template within which future products could be stored. As the team was keen to move on from analysis to the specification stage of development, they decided that it was important to begin TDD. D agreed to begin testing, and provide a document in which all tests could be recorded (Testing Records 1, 15). While these tasks were recorded in Google Docs, the team questioned whether Trello would be preferable.

Meeting 5 10/11 Attendance: Partial (D, K, A, absent). Length: 1 hrs.

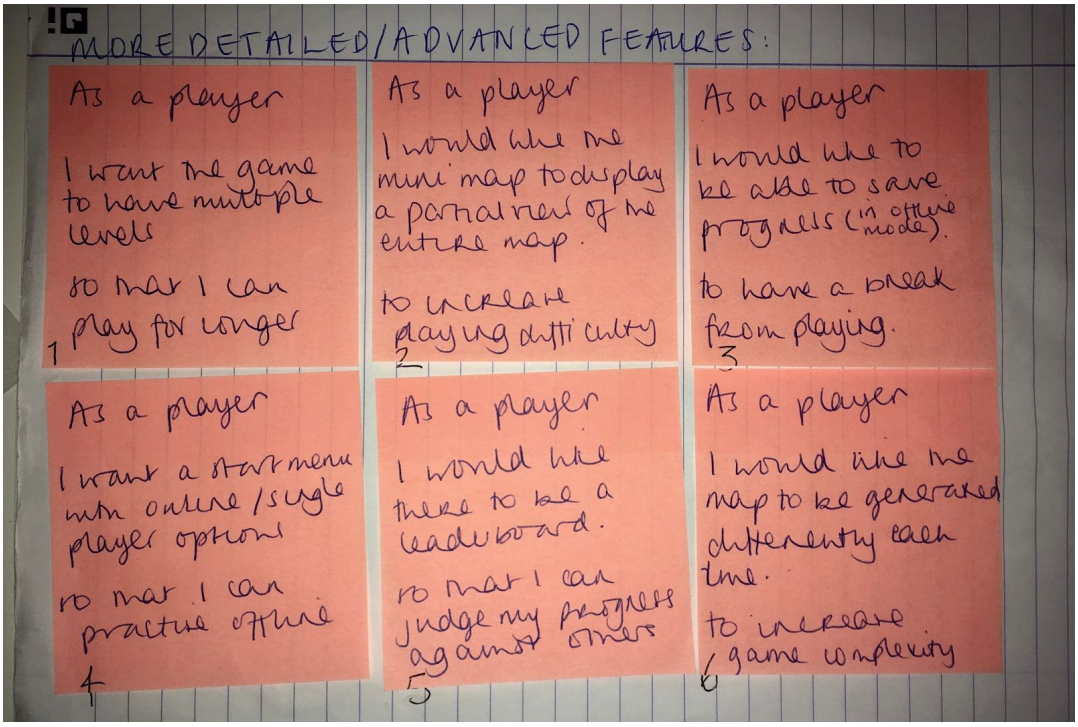
1	Preparation for Customer Interview	Team members drafted questions for the customer with reference to the minutes from Meetings 2 and 3, and the User Stories 2.
2	Use cases	With reference to Scrum, the team decided that writing use cases would aid them their transition from specification to design. They began by creating a use case diagram for playing an online multiplayer game.
2	Customer Interview	The team interviewed the customer with pre-prepared questions. The meeting was finished with the team asking questions to the customer in their capacity as technical consultant. Answers were transcribed into Customer Interview 2 (12). M alerted the absent team members to this document by online messenger.
3	Roles	The team reviewed the performance from previous weeks, and decided that J would be Scrum Master.
4	Documentation	M showed the other team members the template for documentation, which had been filled in with products from the previous week. She explained where future products should be inserted and then uploaded the template to Google Docs so that other team members could access and alter it.
5	Use cases	With reference to the previous customer interview, the team began to transfer their use case diagram to a written use case. To ensure that this task was completed by the end of the sprint, J agreed to finish the use-case after the meeting. To keep all team members aware of progress, she would upload the document to team's shared Google Drive and alert the other team members to this via online messenger.

3.3 Products

The products created in this sprint reflect our motion towards more advanced analysis and specification. The user stories helped the team analyse more complex features, while the use cases, CRC cards and preliminary testing helped the team to specify which features would be included in their product. The customer interview consolidated our analysis of advanced features.

User Stories 2 06/11

These user stories contain more advanced features than those in User Stories 1 (5). They will be stored for future sprints and used if there is enough time to develop the product beyond its basic functions.

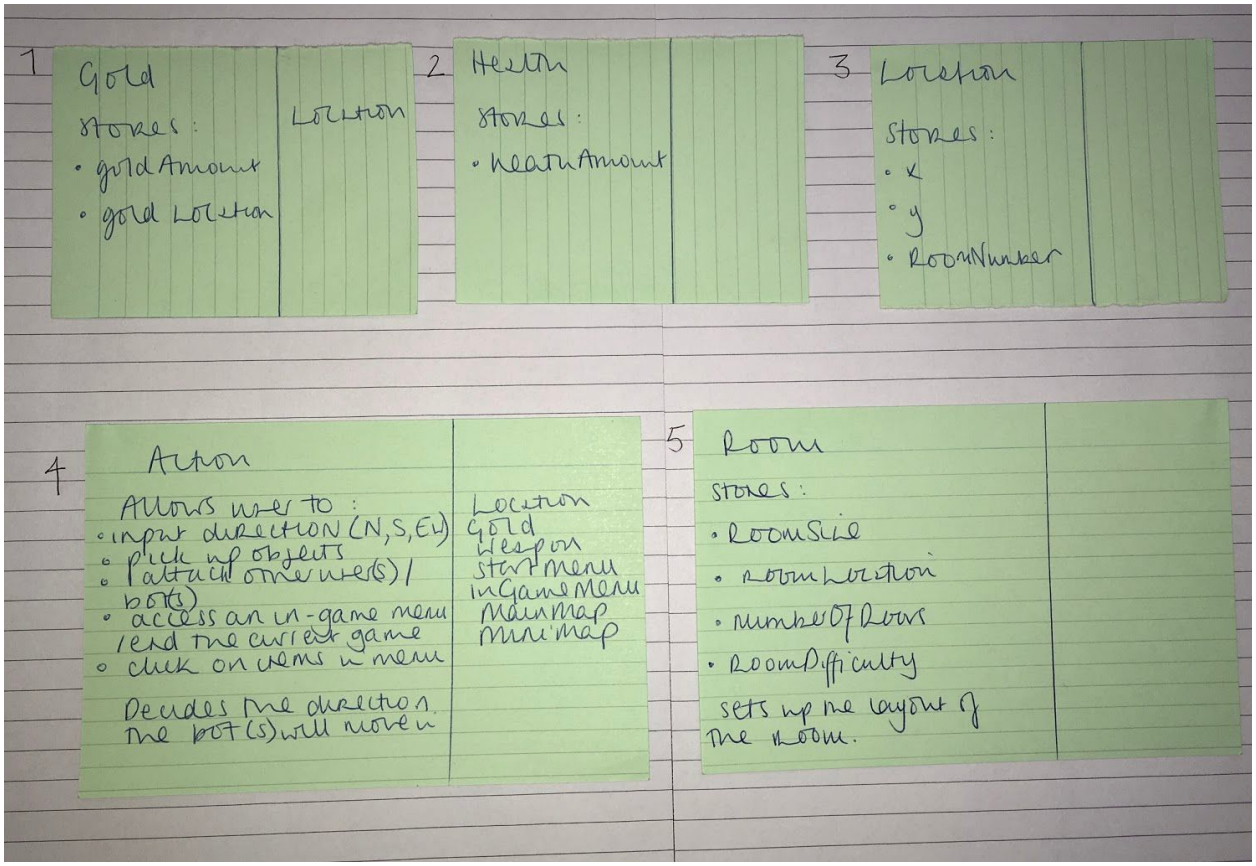


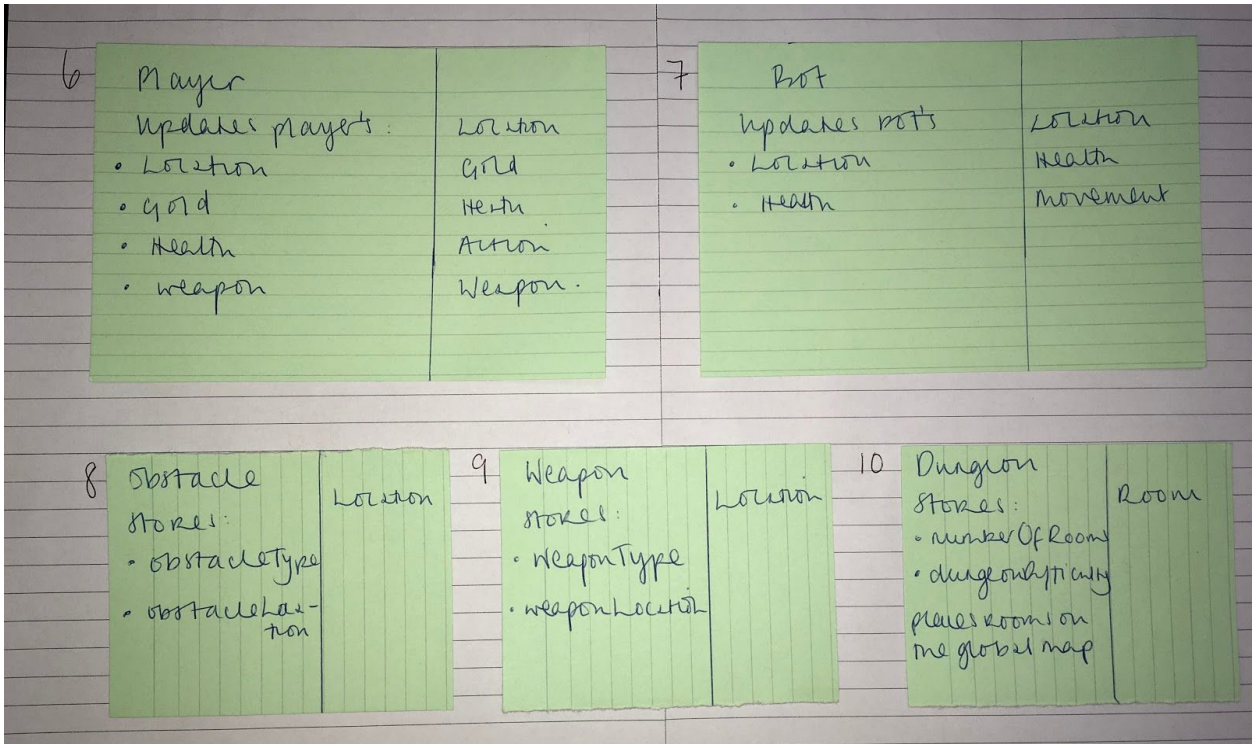
CRC Cards 1 06/11

These are the CRC cards in their simplest form. They outline object titles in no particular order.

CRC Cards 2 06/11

This consists of a more detailed version CRC Cards 1 (10). It presents both what the objects should do and which other classes they interact with.


1
Gold
Stores:
• goldAmount
• gold Location
Location
2
Health
Stores:
• HealthAmount
3
Location
Stores:
• x
• y
• RoomNumber
4
Action
Allows user to:
• input direction (N,S,E,W)
• pick up objects
• attack other user(s) / bot(s)
• access an in-game menu / end the current game
• click on items in menu
Decides the direction the bot(s) will move in
Location
Gold
Weapon
startMenu
inGameMenu
MainMap
MiniMap
5
Room
Stores:
• RoomSize
• RoomLocation
• NumberOfRows
• RoomDifficulty
sets up the layout of the room.


6
Player
Updates players:
• Location
• gold
• Health
• weapon
Location
Gold
Health
Action
Weapon.
7
Bot
Updates bots
• Location
• Health
Location
Health
Movement
8
Obstacle
Stores:
• obstacleType
• obstacleLocation
Location
9
Weapon
Stores:
• WeaponType
• weaponLocation
Location
10
Dungeon
Stores:
• numberOfRooms
• dungeonDifficulty
places rooms on the global map
Room

Customer Interview 2 08/11

This week's customer interview represented the sprint's aims of: analysing more advanced features, and specifying a basic game. The interview began with the team showing the customer the User Stories 1 (5) and 2 (9). The questions followed on from the user stories. They evaluated both the customer's preferences regarding various advanced features, and which features he thought most necessary to include in a basic game. As they had began the documentation process, the team also asked for feedback on their current style. Below are transcribed questions and answers:

1. **What does the specification mean by a 'passages'?**

It is not specifically defined. You could define passages as a series of conjoined rooms: you can go from room to room instantaneously. However, others have passages from which you can go into distinct rooms. Decision depends on what game you choose to make. Also depends upon how you wish to present game view to the players. Either bird's eye view or 1st person. With the former either definition of room is possible to implement, while with the latter it might be more simple to create a series of consecutive rooms.

2. **How should we represent the minimap?**

This depends upon how much you want the player to know. You could design a map which gives a partial view of the game. For example, imagine a microscope view that moves around— you could have a parameter, which you would vary, to get different views. But you could equally present a complete map of the game.

3. **Does the user need to be able to save progress?**

It is not possible to save progress when playing online with concurrent human players. Here, you should use a leaderboard for players to keep track of progress. For offline singleplayer practise, a save mechanism is a good idea.

4. **How many levels would you prefer the game to have?**

The amount of levels depends upon the game which you decide to create. There should be an increase in difficulty between levels. Difficulty can be measured in various ways: the player has to collect more gold before they can pass the level; the player has less time in which to collect the same/more gold; the player has to avoid more bots.

5. **Do you have a prefered documentation style?**

There is not a specific style which you should use. Documentation should be a process which occurs naturally. Minutes from meetings, the state of the backlog, CRC cards, use cases: these will all feature in your documentation and they feature in the development process anyway. All products should be versioned and cross referenced. You should avoid a documentation frenzy in the final week by keeping a template within which the products for each sprint are inserted as completed.

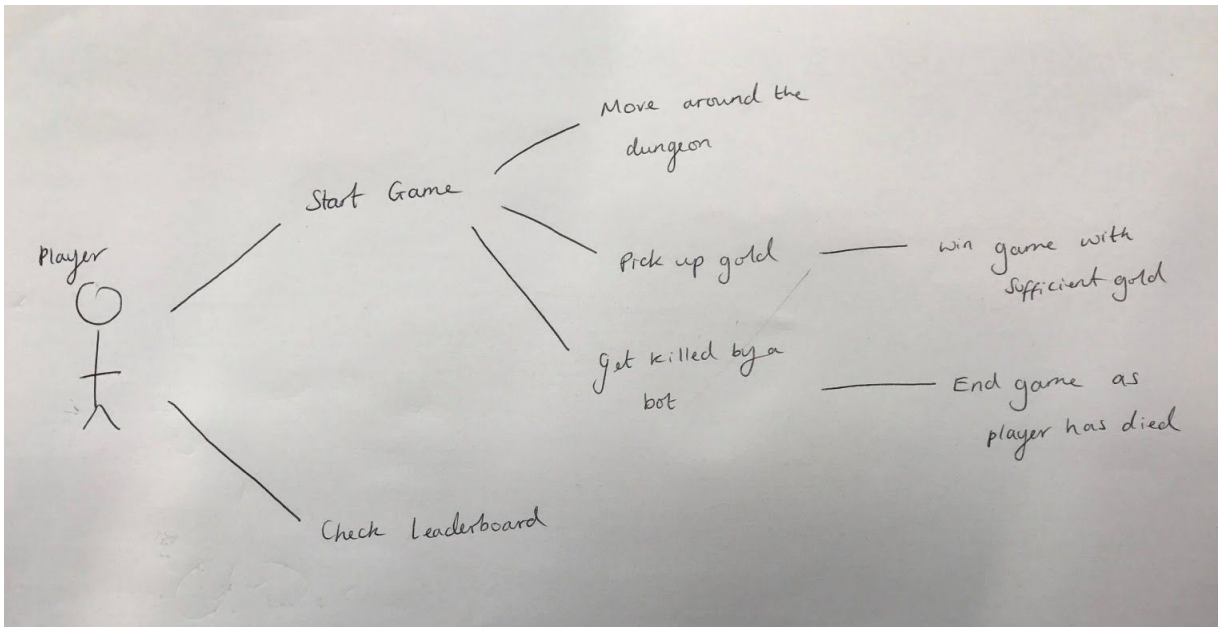
6. **As an advisor, do you recommend that we use Trello for the backlog?**

Trello can be used, but it is not particularly good for output. Could, however, use screenshots to show implementations.

7. How do you recommend we form use cases? Will this example be useful? (Show Use Case 1)
In my opinion, stick and ellipse diagrams are not particularly useful. Look to Cockburn for some good examples. You should write textual use cases. It is most important to write a use case for a player. It is not necessary to write use cases for the programmers and customers, but could write a use case for a zombie/bot.

Use Case 1 10/11

The team drew a use case diagram, as inserted below. This aided the team in mapping out the various objects included in the system. However, the limitations of this method were discussed in Customer Interview 2 (12). The team decided that this diagram was not useful for detail and alternative cases. Also it was not as clear to interpret as a written, and cross-referenced, use case.



Use Case 2 10/11

This use case was written with reference to Cockburn (2000).

2.1 Use Case: Play an online multiplayer game.

2.2 Author: Team Grey

2.3 Date: 14/11/2017

2.4 Purpose: Play an online multiplayer game for entertainment.

2.5 Overview: A player would access the game through a link. A start menu will allow the player to input a nickname for a leaderboard. Once the player clicks on the Start button the game will begin.

Alternative: The player wants to wait for friends to join then he/she will join a lobby until their friend's nicknames have appeared. Then they can start the game.

The player then uses keyboard controls to walk around a set of rooms picking up items and interacting with bots along the way. Once a gold limit is reached the player can then win the game.

Alternative 1: The player loses health from bots attacking and loses the game.

Alternative 2: The player leaves the game because it takes too long to find gold.

The game then updates the leaderboard with the player's score and nickname and the player has the option to play again or quit.

2.6 Cross References: User Stories 1.1; 1.2; 1.3; 1.4; 1.5; 1.6; 1.8; 2.4; 2.5 (6; 10).

2.7 Actors: Player/s

2.8 Pre-Conditions:

2.Pre-1: The game must have a web link for players to access the game.

2.Pre-2: The game must have a user interface for the player to use.

2.Pre-3: The game must allow more than one player to play.

2.Pre-4: The game must have a main menu with "Start" and "Quit" buttons.

2.Pre-5: The game must have have an in-game menu.

2.Pre-6: The game must have a character controlled by user input.

2.Pre-7: The game must have a set of rooms in a dungeon the character can move in.

2.Pre-8: The game must have gold that can be picked up.

2.Pre-9: The game must allow characters to interact with bots.

2.Pre-10: The game must allow the players to have a health bar.

2.Pre-11: The game must have a quit button.

2.9 Postcondition: The leaderboard is updated with player scores and nicknames.

2.10 Flow of events:

Actor's actions:	Systems actions:
1. Player inputs their chosen nickname.	
	2. System stores the player's nickname.
3. Player begins the game.	
4. Player chooses where to move and what to interact with.	
	5. System reads the input moves of the player. For each input move the bot locations are updated.
	6. System updates and checks how much gold the player has collected.
	7. System updates and checks how much health the player has so the game can end when the player dies.
8. Players wins the game by collecting enough gold.	
	9. Leaderboard is updated when the player wins.
10. Player quits the game.	

2.11 Alternative flow of events:

- Step 3: Player waits for their friends to input nicknames and join the game. They may then begin the game.
- Step 8: Player is attacked by a bot, after all health is lost the game is lost. Go to step 1.
- Step 8: The player gets bored of the game and quits. Go to step 10.

2.12 Exceptional flow of events:

- Power failure during gameplay. The leaderboard is updated with most recent score at Step 9.

Testing Records 1 12/11

This is the first version of testing records. It includes the tests for a game's most basic features. It was created with reference to User Stories 1 (5), User Stories 2 (9), CRC Cards 1 (10) and CRC Cards 2 (10). While these tests are used in later pair programming sessions, they also provide the basis for specification during this sprint. The testing records here, and throughout the document, are structured as follows:

1. A test is given a reference in numerical form (e.g. 1.1). The first number refers to the current testing records version and the second refers to its position within that version.
2. This test is outlined and linked to relevant user stories, use case and CRC cards.
3. The test outline is followed by an executable test. This gives details of how the test will be run and what features it will check. The executable has a reference consisting of three separate numbers (e.g. 1.1.1). The first two reference the executable's corresponding test id, and the final refers to its version in this document.
4. Whenever the implementation of a test is the specific focus of a programming session, it will be re-versioned (e.g. 1.1.1 will become 1.1.2).
5. Testing records will not include tests for visual and auditory enhancements. As such features are a mark of personal preference, we have classed these as 'ad hoc' tests.

Test 1.1 11/11

Test Priority:	High.
Test Title:	Player movement.
Description:	Test the player navigation using WASD keyboard input.
CRC Card:	2.4.Action; 2.6.Player (11).
Use Case:	2.Pre-6 (13).
User Story:	1.5 (6)

Executable 1.1.1 11/11 Corresponding Test: 1.1 (above)

	Test Steps	Test Data	Expected Result	P/F
1	Press W to move character north.	Input "W" using keyboard.	Character moves north.	F.
2	Press D to move character east.	Input "D" using keyboard.	Character moves east.	F.
3	Press S to move character south.	Input "S" using keyboard.	Character moves south.	F.
4	Press A to move west.	Input "A" using keyboard.	Character moves west.	F.

Test 1.2 11/11

Test Priority:	High.
Test Title:	Player movement.
Description:	Test the player navigation using the keyboard arrow keys.
CRC Card:	2.4.Action; 2.6.Player (11).
Use Case:	2.Pre-6 (10).
User Story:	1.5 (6)

Executable 1.2.1 11/11 Corresponding Test: 1.2 (above)

	Test Steps	Test Data	Expected Result	P/F
1	Press UP to move character north.	Input UP using arrow keys.	Character moves north.	F.
2	Press LEFT to move character west.	Input LEFT using arrow keys.	Character moves west.	F.
3	Press DOWN to move character south.	Input DOWN using arrow keys.	Character moves south.	F.
4	Press RIGHT to move east.	Input RIGHT using arrow keys.	Character moves east.	F.

Test 1.3 11/11

Test Priority:	High.
Test Title:	Gold test.
Description:	Test the player can collect gold from the floor by moving the avatar over the gold location.
CRC Card:	2.1.Gold; 2.5.Room (11)
Use Case:	2.Pre-8 (14).
User Story:	1.4 (6)

Executable 1.3.1 11/11 Corresponding Test: 1.3 (above)

	Test Steps	Test Data	Expected Result	P/F

1	Begin the test.	Load up the executable/source code and press start game.	Gold is visible on the floor.	F.
2	Move character towards the gold.	Using arrow keys, move avatar towards the gold pieces on the dungeon floor.	Gold is removed from the user interface and can no longer be picked up.	F.

Test 1.4 12/11

Test Priority:	High.
Test Title:	Procedural generation test.
Description:	Verify that a new dungeon is created every time the game is run.
CRC Card:	2.5.Room; 2.10.Dungeon (11).
Use Case:	2.Pre-7 (14).
User Story:	2.6 (10)

Executable 1.4.1 12/11 Corresponding Test: 1.4 (above)

	Test Steps	**Test Data**	**Expected Result**	**P/F**
1	Start a new game.	Press enter when game is loaded.	A dungeon is generated.	F.
2	Start a new game.	Press enter when game is loaded.	A dungeon with a specific number of rooms is generated.	F.
3	Exit the game.	Press Escape to exit the dungeon.	The game is exited.	F.
4	Try to start a new game again.	Press enter when game is loaded	A new, visually different dungeon is generated around a player.	F.

Test 1.5 11/11

Test Priority:	High
Test Title:	Room spawning test.

Description:	Verify that a room spawns with 4 walls and at least 1 door
CRC Card:	2.5.Room (11).
Use Case:	2.Pre-7 (14).
User Story:	1.6 (5).

Executable 1.5.1 11/11 Corresponding Test: 1.5 (above)

	Test Steps	Test Data	Expected Result	P/F
1	Begin the test.	Run the source code.	A collection of rooms is generated.	F
2	Begin the test.	Run the source code.	Rooms have 4 walls.	F
3	Begin the test.	Run the source code.	Rooms have at least 1 door.	F
4	Begin the test.	Run the source code.	Rooms are interconnected.	F

4.0 Week 7 13–19/11/17

4.1 Overview

During this sprint, the team built upon their previous specification by focussing upon their system's design and implementation. The key task for the sprint was creating the necessary Java classes for a basic system. To direct this activity, the team began by ordering their CRC cards. They also engaged in some system design. They ensured that the design was lightweight so that it could be adapted in later sprints. The agile practises of pair-programming and TDD were also instrumental. These provided the basis of the team's techniques for coding the classes. In Customer Interview 3 (30), the team asked questions which were more focussed on practical elements of the project.

Sprint Review 3 18/11 Attendance: Full

This sprint was successful. The team completed the construction of basic java classes within the sprint period. A highlight of the sprint was pair-programming. Members voiced its benefits. These included: the free-flow of ideas; clean code; and pooled knowledge. The team also noted how TDD had initially presented some difficulties. A proportion of the team had not used this practise before and therefore found it difficult to conceptualise how testing records could be used as the primary means for implementing their system. On reflection, the team were pleased with how they had drawn upon other team member's experience, organisation and effective communication in order to overcome this obstacle. The members

who were initially unfamiliar with the practical implementation of the TDD now expressed its benefits. Overall, the team were relieved to have began the implementation stage of the software design lifestyle. However, they were also aware that they had not yet created a working, basic game. They suggested that this should be a focus for the next sprint.

4.2 Process

Backlog 3

Task	Member to complete	Date set	Deadline date
Document previous and current sprints in framework for team submission.	M.	13/11/17	19/11/17
Create CRC cards 3.	A, J, M.	13/11/17	13/11/17
Create the user interface design.	K and M.	13/11/17	13/11/17
Create User Stories 3.	Everyone.	13/11/17	13/11/17
Create a template for pair-programming records.	M.	13/11/17	15/11/17
Interview customer.	Everyone.	15/11/17	15/11/17
Design dungeon.	A, J.	18/11/17	18/11/17
Create Location class for storing coordinate positions.	A, J.	15/11/17	19/11/17
Create room/ dungeon class from testing records.	A, J.	15/11/17	19/11/17
Create player/actions class from testing records.	R, K.	15/11/17	19/11/17

4.2.1 Meeting Records

Meeting 6 13/11 Attendance: Full. Length: 3 hrs.

1 Tasks The team discussed which tasks they would like to undertake during this sprint. They referenced Sprint Review 2 (7), in which they had expressed the need to begin design and

implementation. The team concluded that the overall aim for this sprint would be to create the basic classes needed for their game.

2 CRC cards — In order to move from specification to implementation, the team engaged in some preliminary design. A and J had re-versioned the CRC cards, collected these as CRC Cards 3 (27), and placed them in an order which represented the system's functionality. They showed these to the group.

3 System design — K had created a model of the system's user interface (System Design 1, 29). This was enhanced by the team's suggestions. However, the team were careful to keep the model as basic as possible. If lightweight, then it could be easily adjusted with customer demands or system faults.

4 Specification — When deciding upon which classes to focus their initial efforts, the team referred to User Stories 1 and 2 (5; 9). They selected some stories with which to start, and collected these in User Stories 3 (29). The team then referred to Testing Records 1 (15) as this detailed the most essential features of their game. The team decided to begin by coding the room and player classes. The team noted that once these had been created and combined on GitHub, they would have an offline, singleplayer game. This would fulfil the agile mantra of progress being incremental. They would then develop this game in further iterations.

5 Pair Programming — The team decided that in order to achieve the cleanest code, they would pair-program these classes. A and J volunteered to program the rooms and dungeons. K and R volunteered to program the player and action classes. M would create a template so that the pairs could record their progress during each pair programming session. This would be useful for documentation and, if uploaded to Google Docs, the other team members could be made aware of what happened during any coding session.

6 Testing — The team noted the importance of using TDD. When beginning coding, the team members should look at Testing Records 1 (15). To aid their work, team members should also use a test harness. J-unit would be ideal. Some team members were unfamiliar with the practical application of TDD. While they had observed how testing records were useful for specification, they were unclear as to how testing would provide the basis for all programming. To solve this issue, the team decided that they would make the testing records template more comprehensive. A team member most familiar with testing would outline the tests sufficiently before each pair-programming session, upload them to Google Docs, and alert the other team members to progress by online messenger. Team members would be able process each test prior to programming and have time to ask any questions. D volunteered to improve the testing record template and write the tests. R then explained to the group how to use J-unit as they coded.

7 Manuals — As the team had began implementation, M thought that it was also time to consider the manuals. She thought that the user manual could be started first as it would necessarily present at least the simplest version of the game. It could therefore be framed from the previously created user stories. The technical and maintenance manuals would perhaps have to be created later, after the finer details of the game had been set. The team voiced overall uncertainty as to when and how they should make these manuals. They noted this in their document of customer questions, in preparation for the next interview (Customer Interview

3, 30).

Meeting 7 15/11 Attendance: Partial (D absent). Length: 2 hrs.

1	Preparation for Customer interview	The team prepared for the customer interview by gathering the process materials created during the previous sprint. They then documented the questions which arose from these materials. As they were beginning to focus upon the implementation stage of the software lifecycle, the team noted that—besides using the session to engage in further product analysis—they should seek advice from the customer in the capacity of technical consultant. Questions were noted accordingly.
2	Customer Interview	The team attended the customer interview. The product master recorded the answers in Google Docs as Customer Interview 3 (30).
3	Customer interview Review	The team reflected upon the customer's advice regarding the user manual. While the team did not wish to enforce unrealistic deadlines which would undermine the flexibility of Scrum, they agreed that also would be difficult to write manuals in full before the game's completion. Therefore, they made a preliminary decision to finish the game a week before the deadline. This would give them a chance to create the manuals and enhance the game if they had time.
4	Communication	As a team member was absent, the other members alerted them to the minutes and customer's answers which had been uploaded to Google Docs. They also encouraged the absent member to voice any queries about the uploaded material.

Pair Programming 1 16/11 Pair: J and A. Length: 7 hrs.

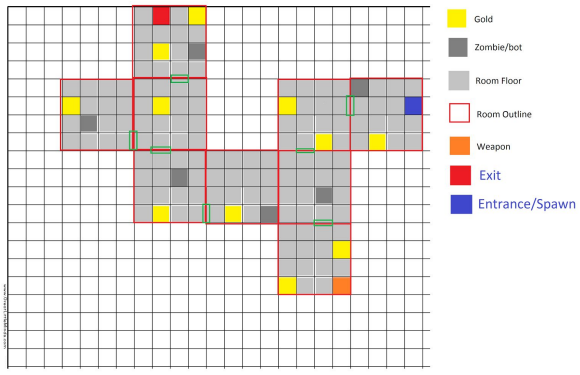
Aim:	To create the Room.java class so a room is generated with a random number of doors and in random locations. To do this Location.java had to be created to store x, y coordinates and the room number.
CRC cards:	3.3.Room (27).
Use Case:	2.Pre-7 (14).
User Story:	1.6 (6); 2.6 (10).
Achieved:	A Room class was created with comments. To do this Location.java had to be created to store x, y coordinates and the room number. This class contains an equals() method to compare whether two locations are the same and a distanceFrom() method to return the distance from one location to another. When a room is created the bottom left corner coordinate was stored as the room location. This creates a good reference point for wall and door locations. The method stores the possible four door locations (N, E, S, W) and chooses a random amount of doors of 0-4. The door locations are then stored.

Visualisation:	No visualisation created.
Difficulties:	No specific difficulties.
Next Actions:	We would like to improve the number of doors. A high number of rooms with 3 or 4 doors may make the map too square.
Tests:	**Executable 1.5.2** 18/11 Corresponding Test: 1.5 (19).

	Test Steps	Test Data	Expected Result	P/F
1	Begin the test.	Run the source code.	A collection of rooms is generated.	P.
2	Begin the test.	Run the source code.	Rooms have 4 walls.	P.
3	Begin the test.	Run the source code.	Rooms have at least 1 door.	P.
4	Begin the test.	Run the source code.	Rooms are interconnected.	F.

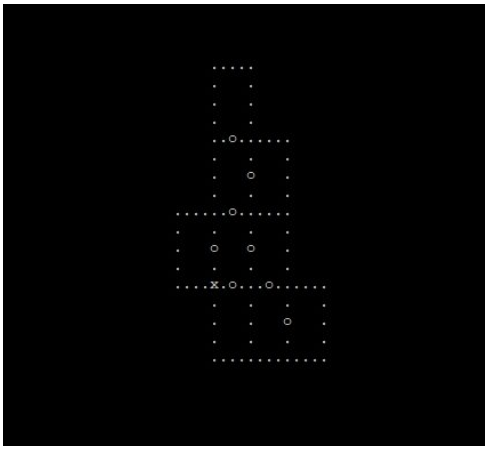
Pair Programming 2 18/11 Pair: J and A. Length: 3 hrs.

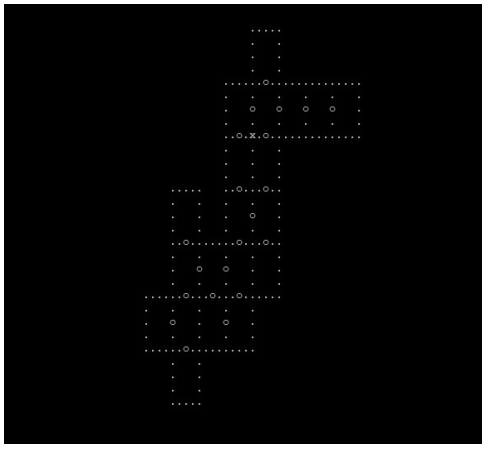
Aim:	Create a Dungeon.java class which generates the map.
CRC cards:	3.4.Dungeon, 3.3 Room (27). A door.java class was also added. This is not referenced until CRC 4.12.Door (49).
Use Case:	2.Pre-7 (14).
User Story:	1.6 (6); 2.6 (10)
Achieved:	We created a hand drawn map. This provided an outline of how we would like our game's dungeon to appear. It was then necessary to create a door class to set the location and cardinal point of the doors. The cardinal helps identify where the next room can be generated. We then generated a dungeon class which generates new rooms using recursion. The first room was placed at coordinate (50, 50) with room size 4. The method generateRooms() can be used to place rooms on the map where there are doors. This will ensure that each room can be reached. However, this meant the number of rooms generated, depended on the number of doors in each room.
Visualisation:	**System Design 2** 18/11 This is a manually created room design. It provided a visual point of reference when creating the dungeon class.



Game Visualization 1 18/11

These are example dungeons, procedurally generated by Dungeon.java. They are, therefore, only two random examples of possible dungeons. We created an array of dots and circles to visualise the dungeon. The dots represent walls, the circles represent doors and the cross represents the bottom left corner of the initial room. These differ from System Design 2 (above) slightly. Here: doors are centered on the walls and defined on the coordinate; the map is a larger size; and the map is lacking entrance and exit points.





Difficulties:	Creating a Door.java class to store the cardinal point of the door (N, E, S, W) was

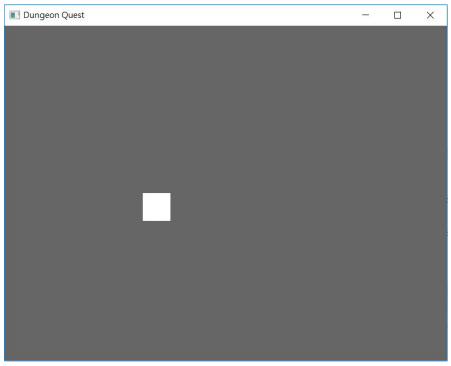
	time-consuming and delayed our start on the Dungeon.java class. Moreover, at times the dungeon would create dead-ends too soon. This resulted in some dungeons being too small. Occasionally the dungeon would be too big where rooms had generated with 3 or 4 doors each. We tried to get around this by limiting the number of rooms but this either generated rooms with doors leading to nowhere as the recursion stopped too soon or the method failed to create a dungeon due to the number of rooms not being reached.
Next Actions:	Find a suitable probability distribution to vary the number of doors so we can limit the number of rooms generated. Create visualisation using simple graphics. Also merge with player and action classes, which will be created by another pair.
Tests:	**Executable 1.4.2** 18/11 Corresponding Test: 1.4 (18).

	Test Steps	Test Data	Expected Result	P/F
1	Try to start a new game	Press enter when game is loaded	A dungeon is generated.	P.
2	Start a new game.	Press enter when game is loaded.	A dungeon with a specific number of rooms is generated.	F.
2	Exit the game	Press Escape to exit the dungeon.	The game is exited. .	P.
3	Try to start a new game again.	Press enter when game is loaded	A new, visually different dungeon is generated around the player.	F.

Pair Programming 3 18/11 Pair: R and K. Length: 4 hrs.

Aim:	Create a character which moves in response to user input.
CRC cards:	This initially required us to refer to 3.9.Player; 3.5Action (27). However, we decided to create and use a class not referenced until 4.4.Map (48).
Use Case:	2.Pre-6 (14).
User Story:	1.5 (6).
Achieved:	We created a project on Eclipse with LWJGL. This consisted of a window visualisation with a player moving in response to user input. We decided not to use 'WASD' to control the player's movements. This test (1.1.2, 26), therefore, failed and will no longer be used.
Visualisation:	**Game Visualisation 2** 18/11

The white square represents the player which can move within the window, in response to the user's input.



Difficulties:

The documentation of LWJGL was scarce and mostly outdated.

Next Actions:

It would be useful for us to both research the relevant technology prior to the next pair programming session. This would mean the sessions would be used primarily for coding, rather than research as well.

Tests:

Executable 1.1.2 18/11 Corresponding Test: 1.1 (16).

	Test Steps	**Test Data**	**Expected Result**	**P/F**
1	Press W to move character north.	Input "W" using keyboard.	Character moves north.	F.
2	Press D to move character east.	Input "D" using keyboard.	Character moves east.	F.
3	Press S to move character south.	Input "S" using keyboard.	Character moves south.	F.
4	Press A to move west.	Input "A" using keyboard.	Character moves west.	F.

Executable 1.2.2 18/11 Corresponding Test: 1.2 (16).

	Test Steps	**Test Data**	**Expected Result**	**P/F**
1	Press UP to move character north.	Input UP using arrow keys.	Character moves north.	P.
2	Press LEFT to move character east.	Input LEFT using arrow keys.	Character moves west.	P.

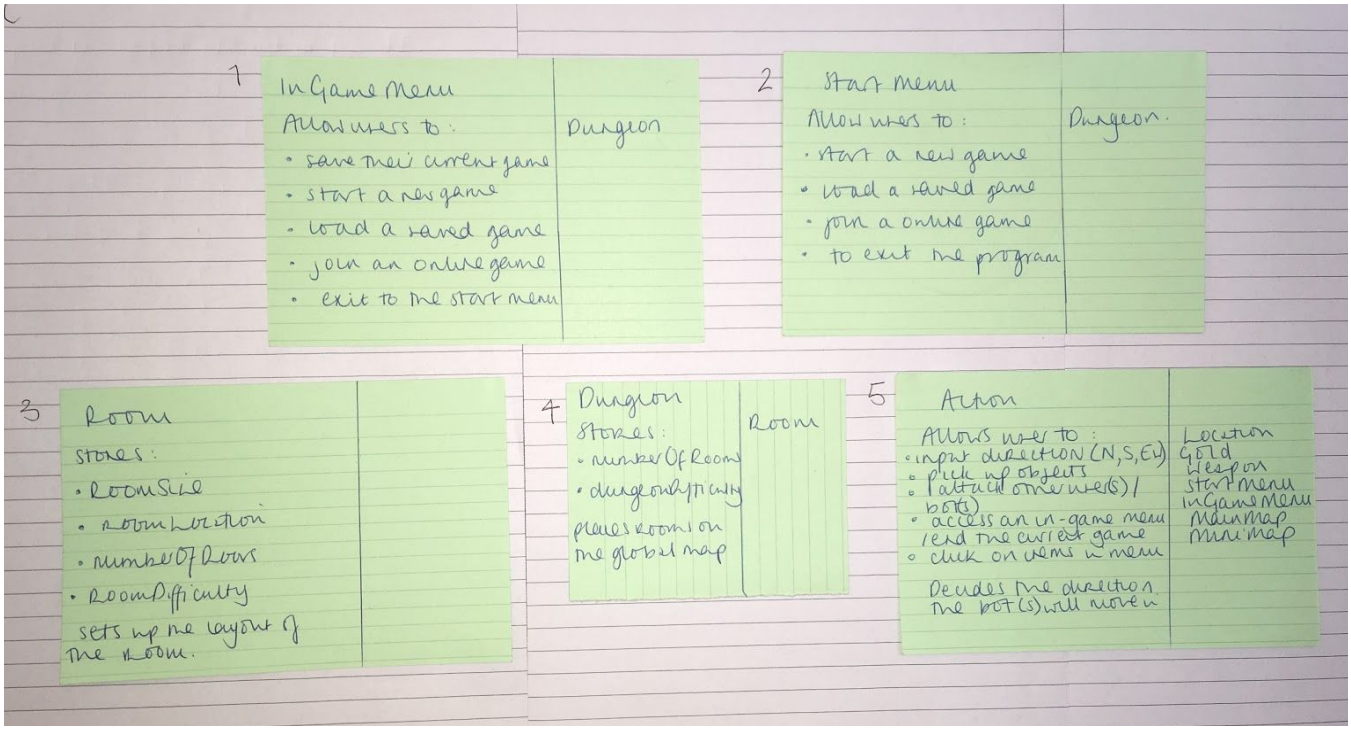
3	Press DOWN to move character south.	Input DOWN using arrow keys.	Character moves south.	P.
4	Press RIGHT to move west.	Input RIGHT using arrow keys.	Character moves east.	P.

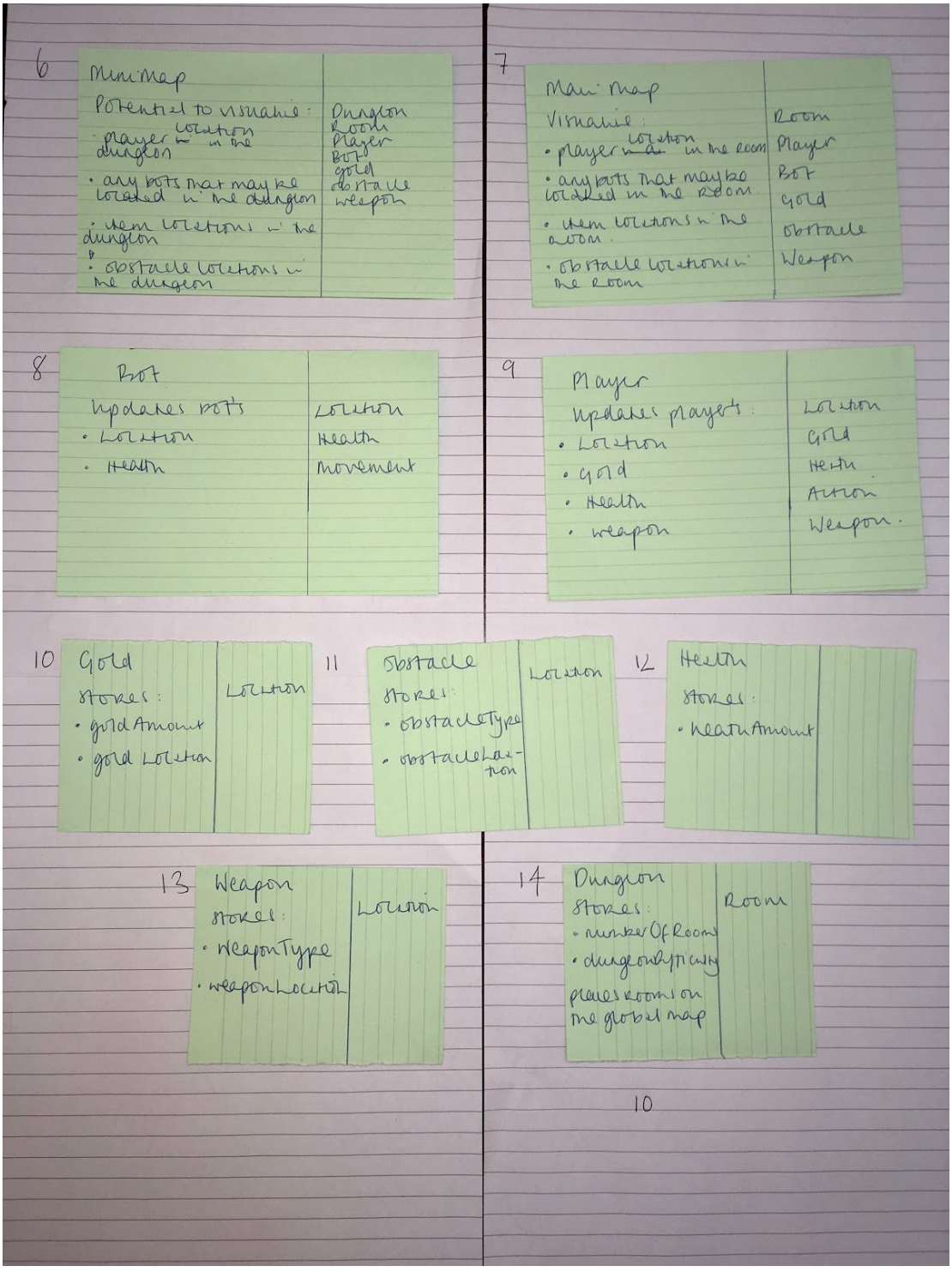
4.3 Products

The products created during this sprint include: further CRC cards and user stories; sketches of system's architecture; testing records; and visualisations of the code. Creating User Stories 3 (29) was particularly useful for specifying which features we would implement during this sprint. The other products represent the team's motion towards design and system implementation.

CRC cards 3 13/11

This version differs to CRC Cards 2 (10), as user interface classes (e.g. StartMenu.java) have been added and all classes have been ordered in terms of user experience.





System Design 1 13/11

This lightweight design outlines the team's thoughts about the user's relationship with the game.

Identifying target audience and requirements:

- Target audience: General (All ages).

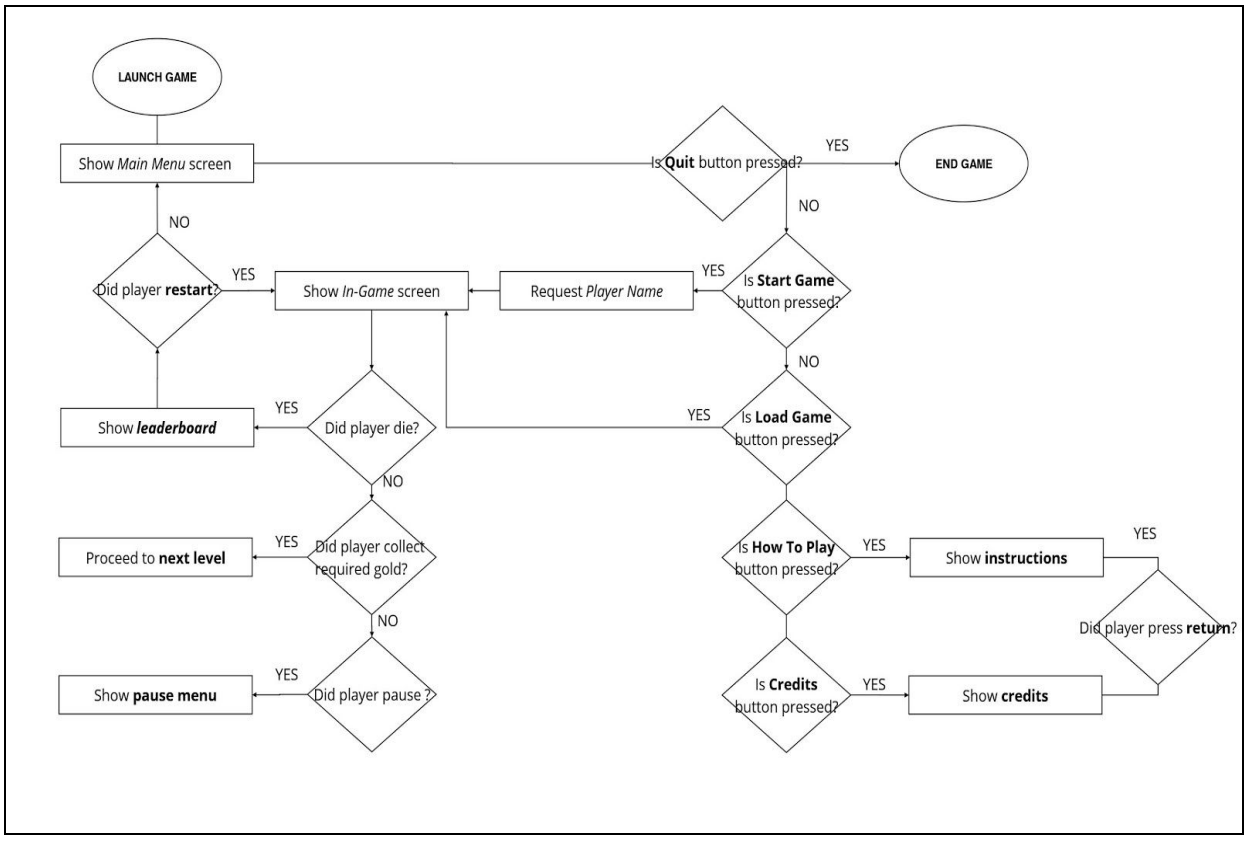
User Interface Requirements:

- Can be understood by a human user.
- Is able to track users' scores.
- Can visualise the whole dungeon.
- Can visualise the player's location.
- Can visualise the bot's location.

User Requirements:

- Can indicate direction of movement.
- Can direct player to pick up gold (or other items).
- Can understand the requirements of a current room.
- Can exit the game.

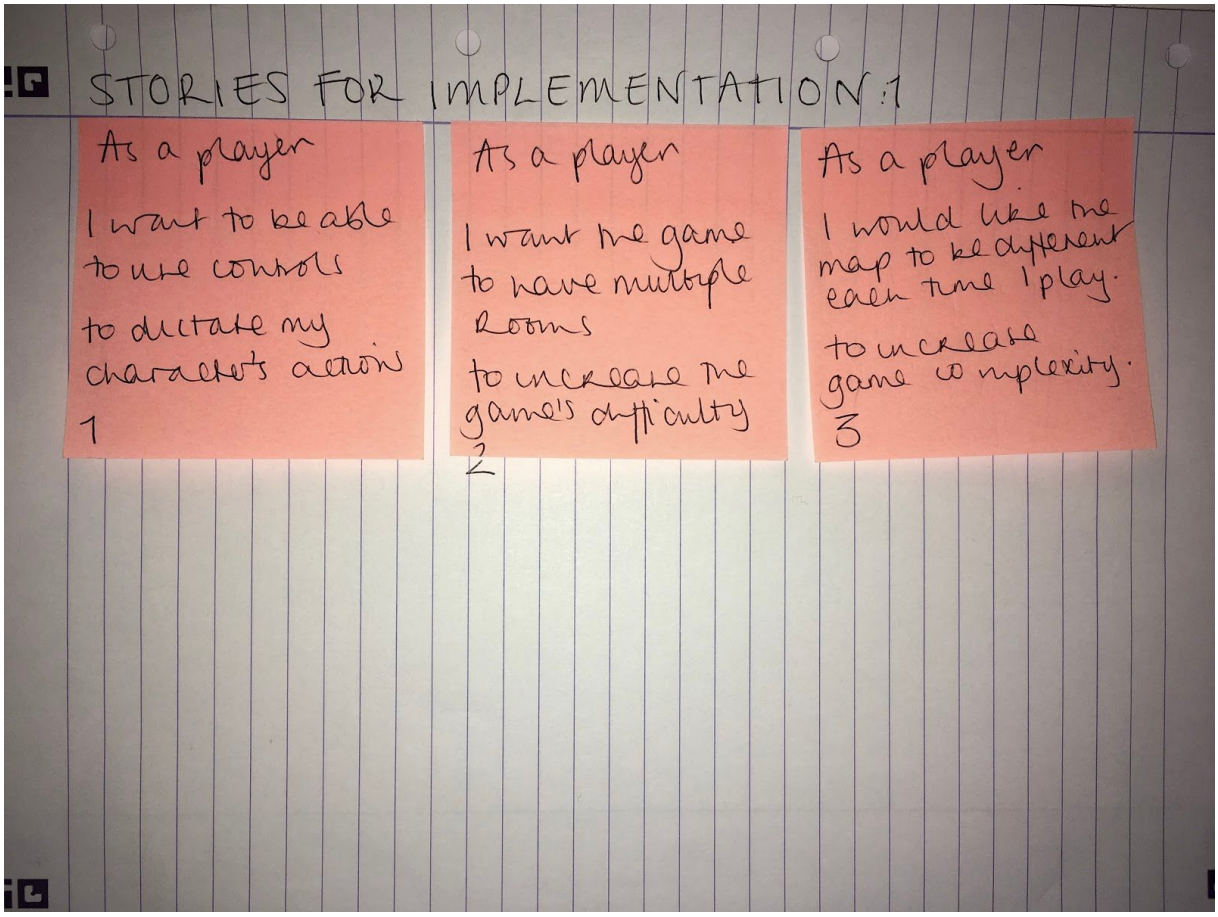
User Flow Diagram:



User Stories 3 13/11

This version of user stories was created to help the team select which features to implement during this sprint. It was formed by team members selecting relevant stories from User Stories 1 and 2 (5; 9). The

chosen stories detail a game with a room and a player, which the team believed to be the minimum requirements of any game. This technique aided the team in coding on a just enough basis.



Customer Interview 3 15/11

As the team were beginning to consider implementation of the simplest possible game, they sought advice from the customer in their capacity of technical advisor. Moreover, to ensure that the transition from design to implementation was handled correctly, the team also sought advice as to how roles should be delegated.

1. **Does the game need to have an intelligent bot?**

No, the game does not need to have an intelligent bot. Remember this is an agile process, and therefore progress should be made incrementally. The first level of development would be a bot which moves randomly, independent of the player's motion. A more advanced bot's motions would correspond to the players. As a side note, this is partly why it is important that the game is turn based. The bot would capture the player too quickly otherwise.

2. **When do you recommend the manuals should be written; is there a prefered style?**

The user manual should be a scenario based document. It would derive naturally from user stories. Therefore, once the game is at implementation stage or even nearing completion, the relevant user stories should be chosen and expanded upon. You should also populate the manual with screenshots.

3. How would you like testing to be documented? Is this satisfactory? (Show testing records 1).
Yes, fine. It is a good idea to have a template to store tests in as they are completed.

4. When would you like a prototype by?
I would like to see incremental process on a weekly basis. The prototype, however, does not need to be finished until the deadline. How quickly we progress depends upon our own timetabling. For next week it would be good to see some working code. To do this, just begin testing anywhere. Start, and bear in mind how to instantiate MVC.

5. How important is it for all team members to be involved with programming?
Ideally, everyone would be involved with each part of the project on a rotating basis. However, it depends what is possible. It is not necessary for everyone to code. There are other important tasks. Everyone, however, should be involved with testing, and should have a clear idea of how the project is developing in each aspect.

5.0 Week 8 20–26/11/17

5.1 Overview

The aim for this sprint was to create a just working game. This would be achieved by merging the existing classes and adding basic features including collectable gold, a bot and simple graphics. Therefore, the practices of pair-programming and TDD were essential this week. Again, re-versioning user stories proved a useful way in which the team could specify which features they would like to install next, and updating testing records gave the programmers clear goals. It was also necessary to re-version the CRC cards as the actual system had deviated from the previous version fairly significantly. Another aim for this sprint was to take the user stories which had been implemented, and convert these into a draft user manual. As usual, the form for team submission was updated alongside the sprint.

Sprint Review 4 26/11 Attendance: Full
In some ways, this sprint was the team's most successful. The team were pleased to have a just functioning game. As recorded in Meeting 6, the team had previously not expected to have a working game until the penultimate sprint (Week 10). This progress was useful, as it allowed the team to begin working on their user manual. However, the increase in productivity had caused a slight lapse in organisation. During the previous sprint, the system design had deviated from the original CRC cards. While the team had noted this and re-versioned the CRC cards accordingly (CRC Cards 4, 47), there was an interval of 5 days in which one pair was coding with reference to an expired architectural model and

tests. This: made it difficult for this pair to understand the current system and how their work would fit within it; meant that a set of tests had to be re-done; and, meant that the program did not exactly follow an MVC model by the sprint's end. This could have been avoided by improved communication. The team agreed to message other pairs more frequently when programming during future sprints. For the following sprint they decided to: add new features, refine their game, and begin work upon the installation guide.

5.2 Process

Backlog 4

Task	Member to complete	Date set	Deadline date
Document and format current sprint.	M.	20/11/17	26/11/17
Creation of User Stories 4.	Everyone.	20/11/17	20/11/17
Begin user manual.	M.	20/11/17	26/11/17
Creation of Items class.	A, J.	20/11/17	22/11/17
Creation of Gold class to extend Items class.	A, J.	20/11/17	22/11/17
Merging existing classes into a single system.	R, K.	20/11/17	26/11/17
Create Testing Records 2.	D.	22/11/17	23/11/17
Creation of game menu.	R, K.	22/11/17	23/11/17
Creation of Bot class.	A, J.	22/11/17	26/11/17
Visualisation of the game.	R, K.	22/11/17	26/11/17
Create CRC cards 4	R and M.	24/11/17	24/11/17

Exception Handling 2 23/11

Discussed in: Pair Programming 7 (38).

Exception: It was difficult for us (A and J) to construct a bot class which worked with K and R's code. This is because K and R's code was all contained in one class, which did not correspond with any of the cards in CRC Cards 3 (27).

Handled: The Bot class was made to store the location of the bot and allow location updates. The bots were constructed in the Room class to be accessed through the dungeon class. This made it possible to only have the option of one or no bots for each room. It also allowed K and R to integrate the bots as they wished so we did not have to worry about changing their code. We also called a meeting for K and R to explain their code.

Exception Handling 3 26/11

Discussed in: Meeting 9 (39) ; Sprint Review 4 (31).

Exception: A pair deviated from the original CRC cards without communicating this to other team members. This made it difficult for other pairs to understand the existing code, and how their code would fit within the design.

Handled: After experiencing difficulties, A and J held an emergency meeting to gain a clearer understanding of the code. A pair who understood the updated code gave a thorough explanation of it and agreed to update the CRC cards accordingly. The testing records were checked and updated to align with the system's current architecture. Beyond this, the team agreed to be more communicative in the future to avoid wasted work or confusion.

5.2.1 Meeting Records

Meeting 7 20/11 Attendance: Full. Length: 1 hrs.

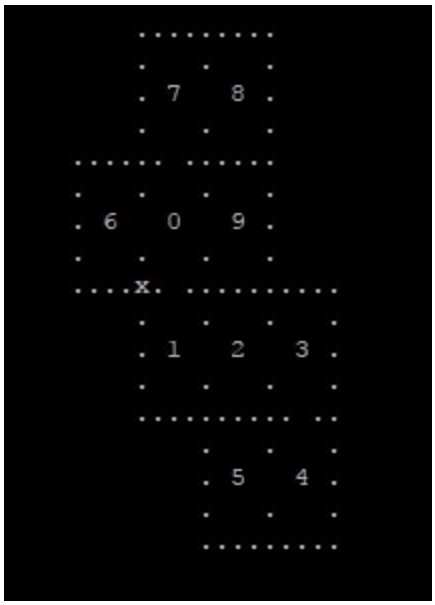
1 Code explanation — The pairs began the meeting by outlining the progress which they had made. Each showed their game visualisations on a shared computer. A and J showed how their map could be procedurally generated, K and R showed their representation of a player responding to user input.

2 Tasks — The team discussed the progress they would like to make that week. They agreed that it would be feasible to create a just functioning game within the sprint. The team noted that even when merged, the system would not have a winning condition, and therefore would not be a game. Therefore, the team decided to firstly install a feature from which a winning condition could be derived. The team now decided to create some user stories to both choose this feature and other more advanced features.

3 User stories — For this sprint's user stories, the team referred to User Stories 1 and 2 (5; 9). From these, they selected stories which they would implement during this sprint. The features included the bot, menus and improved graphics. These were collected into User Stories 4 (42) and D would create tests accordingly. These would be collected in Testing Records 2 (43). The team noted that once these features had been created and merged, they would have a just functioning game.

4 User manual — M observed that as User Stories 3 and 4 provided the basis for a just functioning game, they would also provide the basis for the user manual. She volunteered to write a basic user manual.

Pair Programming 4 21/11 Pair: J and A. Length: 4 hrs.

Aim:	To improve the dungeon generation algorithm to ensure it runs successfully every time and with a specific number of rooms. Also made sure that doors did not lead to nowhere.
CRC cards:	3.3.Room; 3.4.Dungeon (27).
Use Case:	2.Pre-7 (14).
User Story:	3.2; 3.3 (30).
Achieved:	The dungeon generating algorithm finishes every time with all doors leading to another room. However, it does not generate a specific amount of rooms. Instead it uses a distribution on the number of doors generated in a room. This limits the total number of rooms in the dungeon, since eventually rooms will generate no doors leading from them. Also stored the boundaries of the dungeon, excluding doors and door locations of the dungeon in ArrayLists in the Dungeon class.
Visualisation:	**Game Visualization 3** 21/11 The initial room is labelled with a 0. The order the other rooms were generated in is visualised from 1-9. Doorways are left blank. 
Difficulties:	Limiting the amount of rooms generated by the algorithm was difficult. In the end we used a gaussian distribution with mean 1.5 and standard deviation 1, limited to integers between 0 and 4. If we assume more rooms defines a more difficult game, this distribution could be adapted relative to dungeon difficulty.
Next Actions:	Further refine the algorithm which limits the amount of rooms. The test for this was not yet made to pass.

Tests Used:	**Executable 1.4.3** 21/11 Corresponding Test: 1.4 (18).

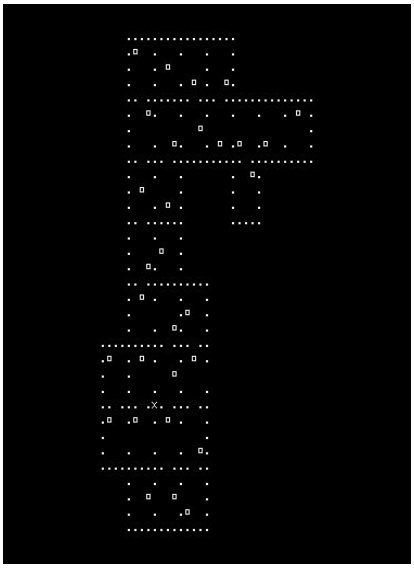
	Test Steps	Test Data	Expected Result	P/F
1	Start a new game.	Press enter when game is loaded.	A dungeon is generated.	P.
2	Start a new game.	Press enter when game is loaded.	A dungeon with a specific number of rooms is generated.	F.
3	Exit the game.	Press Escape to exit the dungeon.	The game is exited.	P.
4	Try to start a new game again.	Press enter when game is loaded	A new, visually different dungeon is generated around the player.	F.

Meeting 8 22/11 Attendance: Full. Length: 1 hrs.

1	Preparation for customer interview	Questions were prepared for the pending customer interview. To ensure that their documentation was fulfilling the requirements, the team also decided to show the customer the most up-to-date submission form.
2	Customer Interview	The team attended the customer interview. M recorded the answers in Customer Interview 4 (43). This was uploaded to Google Docs.

Pair Programming 5 22/11 Pair: J and A. Length: 3 hrs.

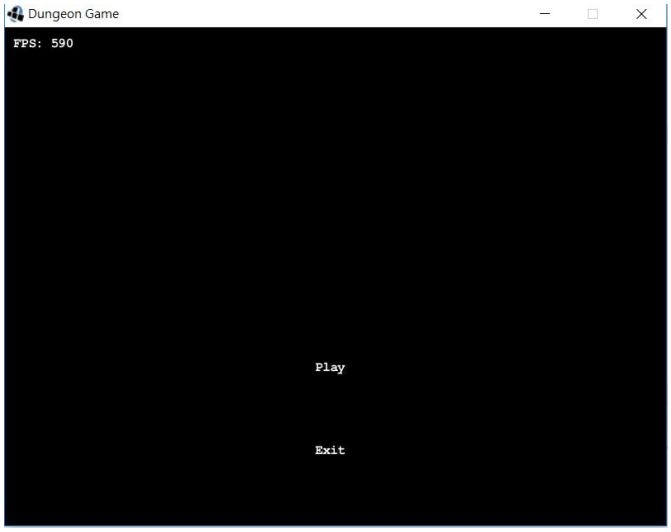
Aim:	To create an item class that a gold class can extend for use in the dungeon class. Store a certain amount of gold in each room.
CRC cards:	3.3.Room; 3.4.Dungeon; 3.10.Gold (27). Also created an item class which is not referenced until 4.10.Item (49).
Use Case:	2.Pre-8 (14).
User Story:	4.1 (42).
Achieved:	The item class stores a location of an item. This could be used as the superclass for any item class created in the future, such as weapon or health. The gold class was created that extends item. This stores the amount of gold at a particular location in the dungeon. Methods were added to the room class to randomly generate gold locations to a specific bound relative to roomDifficulty.

Visualisation:	**Game Visualization 4** 22/11 This is a visualisation of random gold locations in a dungeon. Each gold location is represented by a square. 
Difficulties:	Had slight difficulty in utilising inheritance initially since we had little experience with it. However, with some research and trial and error, it was possible to integrate.
Next Actions:	Visualise the gold on the dungeon map.
Tests Used:	**Executable 1.3.2** 22/11 Corresponding Test: 1.3 (17).

	Test Steps	Test Data	Expected Result	P/F
1	Begin the test.	Load up the executable/source code and press start game.	Gold is visible on the floor.	P.
2	Move character towards the gold.	Using arrow keys, move avatar towards the gold pieces on the dungeon floor.	Gold is removed from the user interface and can no longer be picked up.	F.

Pair Programming 6 23/11 Pair: R and K. Length: 3 hrs.

Aim:	Create a main menu for the game with functioning “Start Game” and “Quit” buttons.
CRC cards:	3.2.StartMenu (27).

<table>
<tr><td>Use Case:</td><td>2.Pre-4 (14).</td></tr>
<tr><td>User Story:</td><td>4.5 (42).</td></tr>
<tr><td>Achieved:</td><td>Created a main menu with working buttons and a player visualised as a quad, moving based upon user input. Slick was used to aid implementation.</td></tr>
<tr><td>Visualisation:</td><td>Game Visualization 5 23/11
The image below captures the main menu which currently contains ‘Play’ and ‘Exit’ options.
 
</td></tr>
<tr><td>Difficulties:</td><td>Difficulties with using LWJGL due to the lack of documentation.</td></tr>
<tr><td>Next improvements:</td><td>We will start to use Slick2D rather than LWJGL, as the latter has an open source library.</td></tr>
<tr><td>Tests:</td><td>Executable 2.1.2 23/11 Corresponding Test: 2.1 (44).
<table>
<tr><th></th><th>Test Steps</th><th>Test Data</th><th>Expected Result</th><th>P/F</th></tr>
<tr><td>1</td><td>Try to start a new singleplayer game.</td><td>Select “start singleplayer” on menu</td><td>A singleplayer instance of the game is started.</td><td>P.</td></tr>
<tr><td>2</td><td>Try to continue a previously saved game.</td><td>Select “continue” on menu</td><td>A previously played instance is started.</td><td>F.</td></tr>
<tr><td>3</td><td>Try to join an online game</td><td>Select “multiplayer” on menu.</td><td>An online version of the game is started.</td><td>F.</td></tr>
</table></td></tr>
</table>

4	Try to exit the program	Select “exit” on menu.	The application shuts down.	P.

Pair Programming 7 23/11 Pair: J and A. Length: 5 hrs.

Aim:	To add a bot class that stores the location of the bot and updates its location based on a direction input. Also, to add methods which set how to move a bot randomly, within the confines of its designated room.
CRC cards:	3.4.Room; 3.8.Bot (28).
Use Case:	2.Pre-9 (14).
User Story:	4.2 (42).
Achieved:	The bot location is stored and moves randomly within its room without exiting through any doors.
Visualisation:	No visualisation.
Difficulties:	Integrating the bot into the game required some thought before programming. To make the game challenging having a bot in every room, with it confined to its own room, seemed like the best way to proceed. However, this choice was not obvious. Initially we thought about bots being able to chase players around the rooms and even allowing the bots to travel around the whole dungeon. Another chief difficulty was understanding the structure of the code more generally. It had deviated from CRC Cards 3 (27) and there was a Map.java class of which we did not properly understand the function. See Exception Handling 2 (32).
Next Actions:	Visualise the bots in the dungeon. Call a meeting in which the current design of the code can be explained and new CRC cards can be created.
Tests:	**Executable 2.3.2** 23/11 Corresponding Test: 2.3 (45).

	Test Steps	**Test Data**	**Expected Result**	**P/F**
1	Move player to room with zombie.	Using WASD or arrow keys.	Bot moves one step as the player moves one step.	P.
2	Move player into the bot.	Walk player into the bot.	The bot removes a player life, or if there is only one life the player dies and is shown the “you died” screen.	F.

3	Bot does not leave its room boundaries.	Move player around and see if bot leaves the room it is assigned to.	The bot stays in the walls, it does not leave the room it is assigned to.	P.
4	Bots movements are random.	Move player around and see if bot moves randomly every time.	Bot moves randomly, no pattern is shown.	P.

Meeting 9 24/11 Attendance: Full. Length: 1 hrs.

1	Need to re-version CRC cards:	The meeting began with J and A explaining that they were confused about the current structure of the game. The current version of the CRC cards was not being adhered to, as there was now a Map.java class. Team members suggested that this should not necessarily be a problem, as agile encourages flexibility. However, in this case, the re-structure had not been communicated. It was necessary to regroup and re-version the CRC cards. This would help all future tests and pieces of code to have a coherent structure. As he was most familiar with the current state of the game, R agreed to re-version the CRC cards later that day.
2	Code explanation:	K and R verbally explained the game's current state to the group. Everyone was now satisfied with their understanding of the current system design.

Group Programming 1 24/11 Group: A, J, K, R Length: 4 hrs.

Aim:	To: merge the existing classes; format the code, so that it is written in 4 tab size style; and, begin improving graphics.
CRC cards:	4.2.Main; 4.3.Map; 4.4.Map; 4.5.Map; 4.6.Room; 4.7.Dungeon; 4.8.Location; 4.9.Bot; 4.10.Item; 4.11.Gold; 4.12.Door (48).
Use Case:	2.Pre-2; 2.Pre-6 ; 2.Pre-7 (14).
User Story:	3.1; 3.2; 3.3 (30); 4.3 (42).
Achieved:	The classes were formatted, more sprites for rendering graphics were found, and user input code was improved. Merging the code (placing the player onto the map) was started.
Visualisation:	No visualisation.
Difficulties:	There were difficulties when trying to merge the existing classes as the two pairs had different coding styles.
Next Actions:	Continue merging the different classes into one program. Also a bot needs to be created.

Tests:

Executable 1.4.4 24/11 Corresponding Test: 1.4 (18).

	Test Steps	Test Data	Expected Result	P/F
1	Start a new game.	Press enter when game is loaded.	A dungeon is generated.	P.
2	Start a new game.	Press enter when game is loaded.	A dungeon with a specific number of rooms is generated.	F.
3	Exit the game.	Press Escape to exit the dungeon.	The game is exited.	P.
4	Try to start a new game again.	Press enter when game is loaded	A new, visually different dungeon is generated around a player.	F.

Pair Programming 8 26/11

Pair: R and K. Length: 4 hrs.

Aim:	Merge the existing classes and install graphics. This should result in the visualisation of a player, controlled by user input, walking on a map.
CRC cards:	4.2.Main; 4.3.Map; 4.4.Map; 4.5.Map; 4.6.Room; 4.7.Dungeon; 4.8.Location; 4.9.Bot; 4.10.Item; 4.11.Gold; 4.12.Door (47).
Use Case:	2.Pre-2; 2.Pre-6 ; 2.Pre-7 (14).
User Story:	3.1; 3.2; 3.3 (30); 4.3 (42).
Achieved:	A visualization of a map and a player. The player can be controlled by user input and can move through rooms and passages. The program, however, does not detect collision and the player ignores gold and walls.
Visualisation:	**Game Visualisation 6** 26/11 The image below presents the merged classes with our current graphics.[2] It contains a player within a map. The red blocks are walls, the blue blocks are doors, the smaller red and black blocks represent bots, the white and gold blocks represent gold and the player can be found in the top left hand room.

[2] The sources for graphics used are as follows:
Bot. http://www.icons-land.com. [Accessed 26/11/17].
Gold. http://www.777icons.com. [Accessed 26/11/17].
Wall; Door; Player: https://opengameart.org/content/dawnlike-16x16-universal-rogue-like-tileset-v181. [Accessed 26/11/17].

Difficulties:

Despite previous in-person demonstrations and comments, it was difficult to understand A and J's code. Also, sometimes the dungeon would spawn outside the screen bounds. This means that the user cannot see certain rooms and therefore cannot control the player's movements.

Next Actions:

Improve the graphics. Also, find a solution for the dungeon spawning off-screen. We could make a camera class. The camera will track the player. Therefore, it would not matter if the dungeon spawned offscreen, as the user could still see the character and control it.

Tests:

Executable 1.4.5 26/11 Corresponding Test: 1.4 (18).

	Test Steps	Test Data	Expected Result	P/F
1	Start a new game.	Press enter when game is loaded.	A dungeon is generated.	P.
2	Start a new game.	Press enter when game is loaded.	A dungeon with a specific number of rooms is generated.	F.
3	Exit the game.	Press Escape to exit the dungeon.	The game is exited.	P.
4	Try to start a new game again.	Press enter when game is loaded	A new, visually different dungeon is	P.

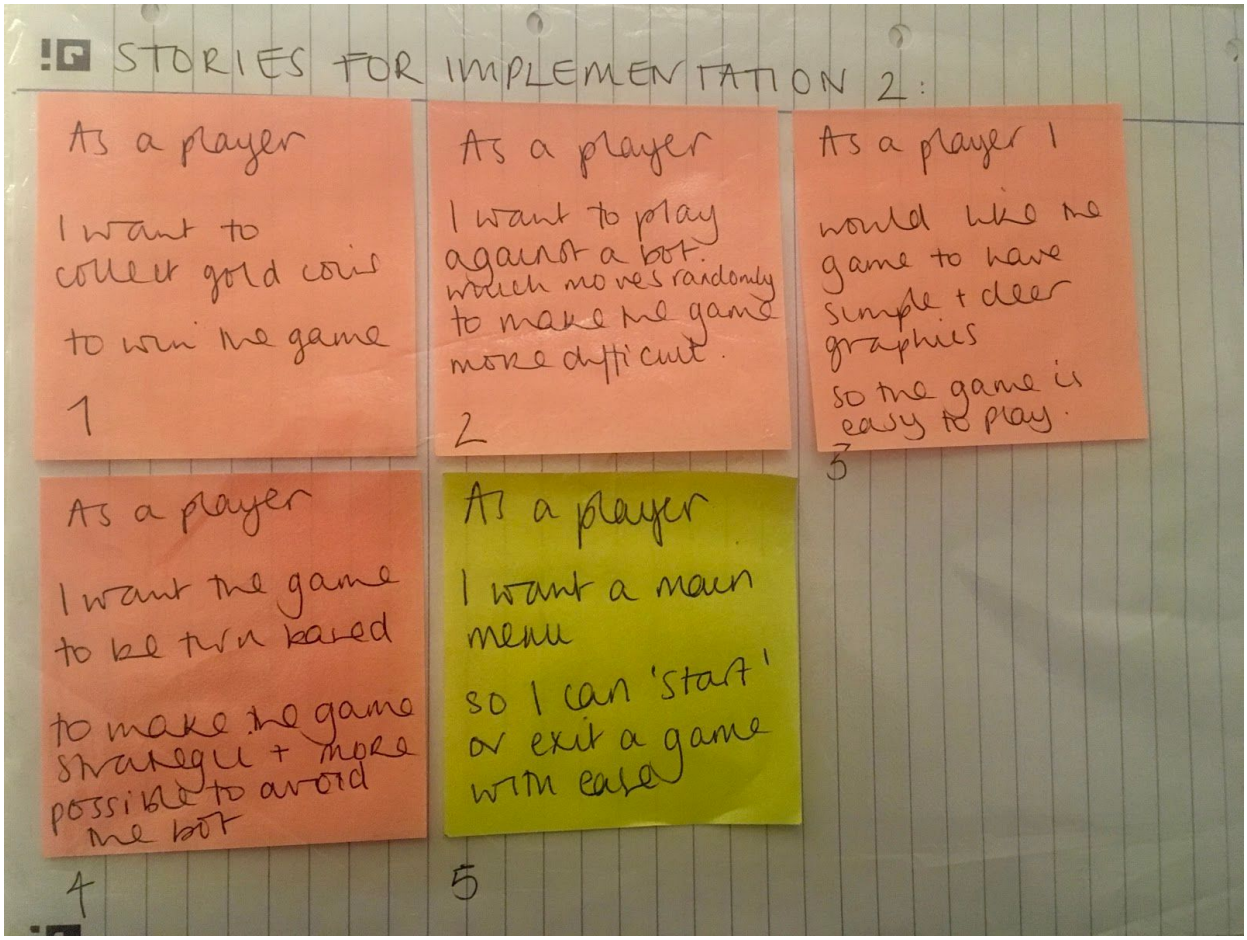
			generated around a player.	

5.3 Products

Products created during this sprint include: further versions of user stories and CRC cards; testing records; and, the customer interview. The most significant item produced during this sprint is the code for a just functioning game which is stored on GitHub. Visualisations of this code at each stage of production can be found in these records. Also produced during this sprint was the first draft of the user manual. It is, however, omitted from these records as it is simply a formatted version of User Stories 3 and 4.

User Stories 4 20/11

This version of user stories functions similarly to User Stories 3 (29). It was created to direct which features the team would implement during this sprint. It consists of select stories from User Stories 1 and 2 (5; 9). These stories were selected as they detail features which—when combined with those from User Stories 3 (30)—form a basic game.



Customer Interview 4 22/11

The team were confident that the features which they had chosen to implement for this sprint had been discussed in a previous customer meeting. They, therefore, used this customer interview to check that their project as a whole was fulfilling the specifications basic requirements. The question matter ranges from documentation to design, and is detailed below:

1. What does the specification mean by: 'File manipulation/database operations in Java'?
Database operations would be used if you decided to implement a scoreboard, for example. There could be stages to this implementation. The first level would be a local file. The second would be a connection to MySQL. The third could be where the game is connected to a server somewhere.

2. For testing, it is necessary to test every method and in a separate file?
You should test for as many components as you see necessary. The tests should be derived from user stories and CRC analysis. In documentation the tests should be linked to the corresponding stories. It would also be a good idea to link tests to particular use cases.

3. Is a Java build tool recommended for this project?
Decide upon whatever works for you. There is no requirement to use one.

4. How detailed should the maintenance guide be? Should it include implementation difficulties?
The maintenance guide should be based upon your final piece of code. Therefore, with regard to implementation difficulties, it depends upon whether they have been resolved by your project's end. If they have been resolved then they should not be detailed. If not, then they could be noted. In any case, it would be useful to provide a record of any difficulties. It will be easy to recall them when you come to write the maintenance guide at the project's end. Remember, a maintenance guide is about providing enough information for someone to navigate your code. It should explain to a programmer what the different functionalities are. If you wanted to add a new icon, this is where and how you could do it etc.

5. How do you recommend that we cross-reference products in our submission form? (Show document).
I do not have an exact method in mind. The most important thing is that I can find my way around the document easily. Version names and page numbers seem fine. You should also number each section in this document, and perhaps move the review up to the opening of each sprint. This way the reader can obtain a high-level overview of the sprint, before going into more detail.

Testing Records 2 22/11

These tests derive from the analysis and specification performed, in Meeting 7 (33), while User Stories 4 (42) were created. They detailed features including a bot and menu, and basic tests for a merged game. They also include a test for an 'in-game' menu. While this feature was not discussed in a meeting, D created this in case it was needed at a later date.

Test 2.1 20/11

Test Priority:	Medium.
Test Title:	Start menu test.
Description:	Verify that a menu is shown to the user upon start-up. Check that its features (including a “start game” and “exit game” buttons) function correctly.
CRC Card:	3.2.StartMenu (27).
Use Case:	2.Pre-4 (14).
User Story:	4.5 (47).

Executable 2.1.1 20/11 Corresponding Test: 2.1 (above)

	Test Steps	Test Data	Expected Result	P/F
1	Try to start a new singleplayer game.	Select “start singleplayer” on menu.	A singleplayer instance of the game is started.	F.
2	Try to continue a previously saved game.	Select “continue” on menu.	A previously played instance is started.	F.
3	Try to join an online game.	Select “multiplayer” on menu.	An online version of the game is started.	F.
4	Try to exit the program.	Select “exit” on menu.	The application shuts down.	F.

Test 2.2 20/11

Test Priority:	Medium.
Test Title:	In-game Menu Test.
Description:	Testing that a user can access a menu whilst in an active game, and that the menu functions correctly.
CRC Card:	N/A. A CRC card has not been made for this feature.
Use Case:	2.Pre-5 (14).
User Story:	N/A. A user story was not made for this feature.

Executable 2.2.1 20/11 Corresponding Test: 2.2 (above)

	Test Steps	Test Data	Expected Result	P/F
1	Load the application.	Select “start game”.	Dungeon is loaded onto screen.	P.
2	Try to save the current game.	Select “save-game” on in game menu.	The instance of the game is saved.	F.
3	Try to exit to the start menu.	Select “exit” to start on menu.	The game returns to the initial start-up menu.	F.
4	Try to exit the program.	Select “exit” on menu.	The application shuts down.	F.

Test 2.3 21/11

Test Priority:	Medium.
Test Title:	Bot Test
Description:	Test to see how bot interacts.
CRC Card:	3.8.Bot (28).
Use Case:	2.Pre-9 (14).
User Story:	4.2 (42).

Executable 2.3.1 21/11 Corresponding Test: 2.3 (above)

	Test Steps	Test Data	Expected Result	P/F
1	Move player to room with zombie.	Using WASD or arrow keys.	Bot moves one step as the player moves one step.	F.
2	Move player into the bot.	Walk player into the bot.	The bot removes a player life, or if there is only one life the player dies and is shown the “you died” screen.	F.
3	Bot does not leave its room boundaries.	Move player around and see if bot leaves the room it is assigned to.	The bot stays in the walls, it does not leave the room it is assigned to.	F.

4	Bots movements are random.	Move player around and see if bot moves randomly every time.	Bot moves randomly, no pattern is shown.	F.

Test 2.4 22/11

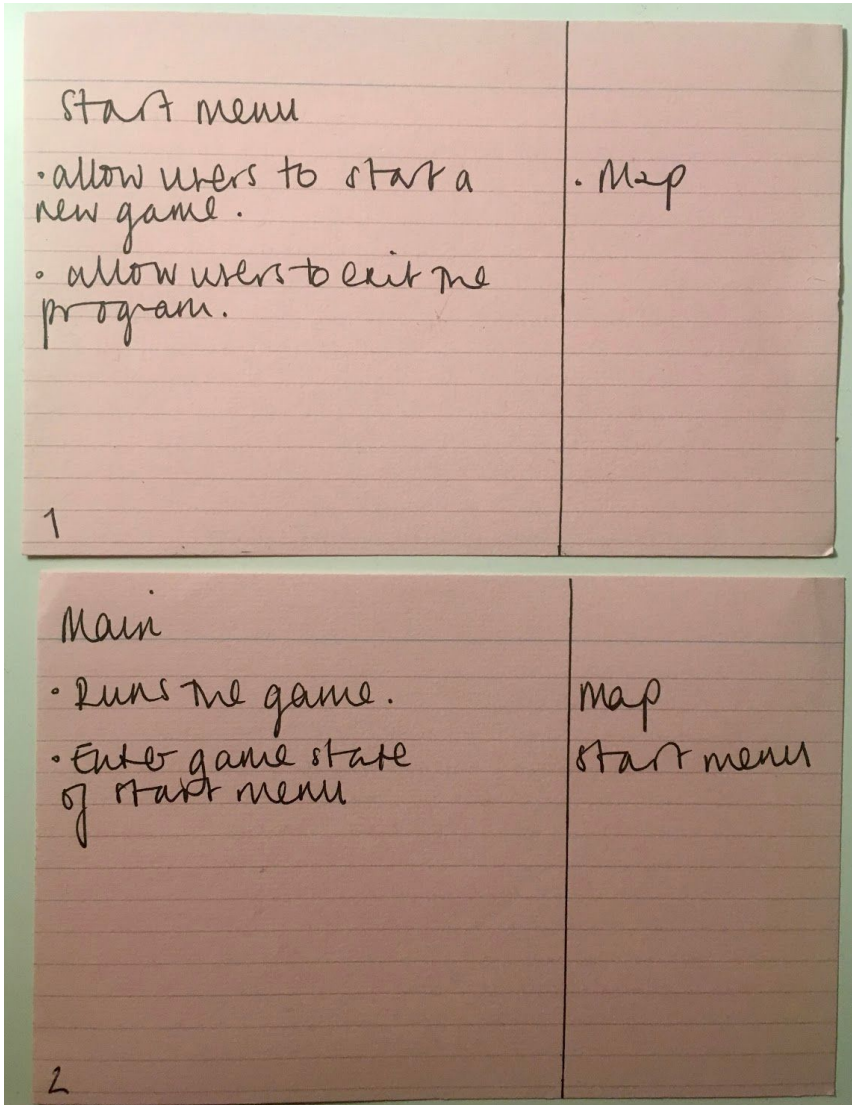
Test Priority:	High.
Test Title:	Map Test.
Description:	Testing that the map functions correctly: the correct items are spawned in the correct quantities and the player cannot enter incorrect boundaries.
CRC Card:	3.3.Room; 3.4.Dungeon (27).
Use Case:	2.Pre-2; 2.Pre-7; 2.Pre-8; 2.Pre-9 (14).
User Story:	N/A.

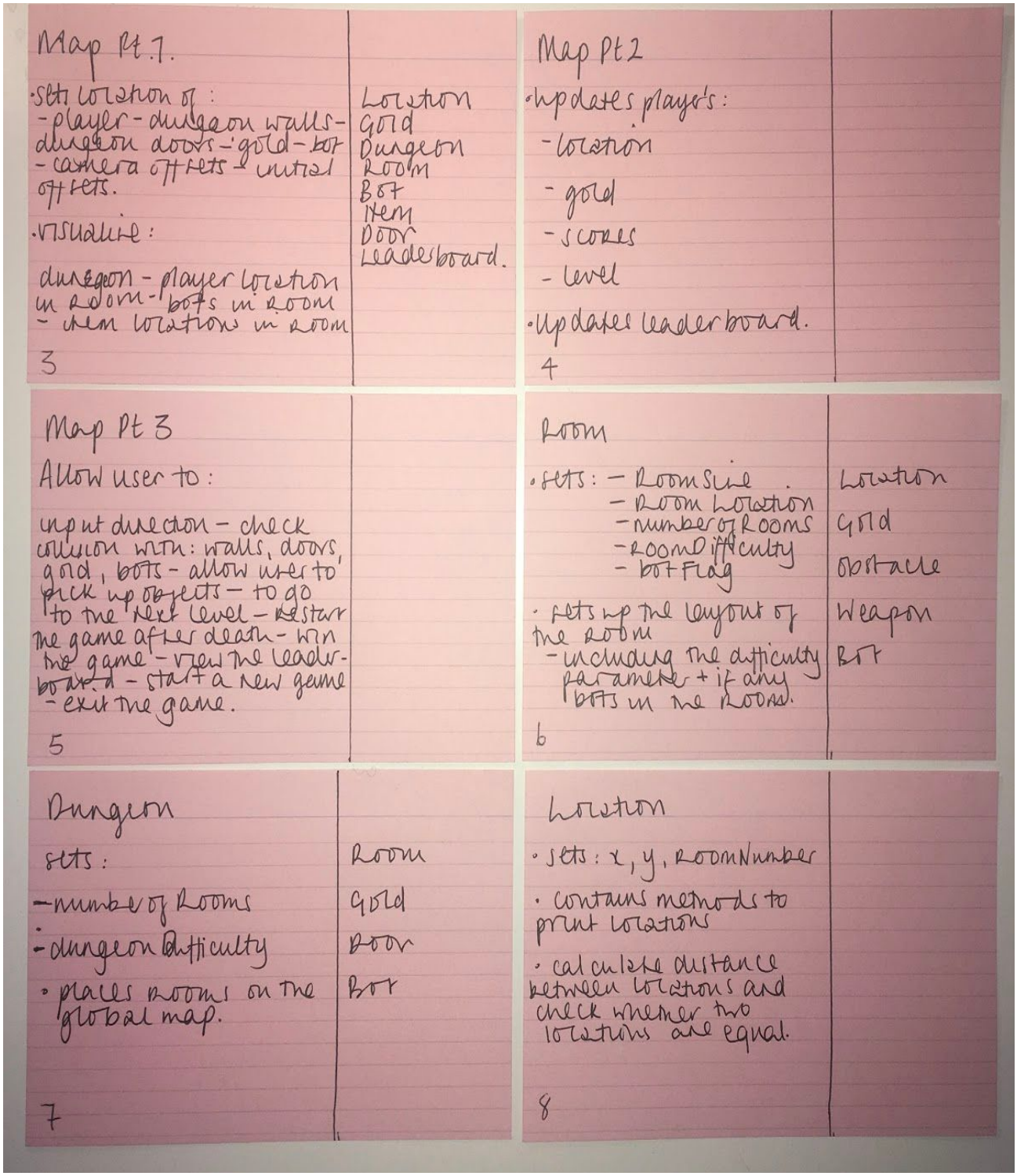
Executable 2.4.1 22/11 Corresponding Test: 2.4 (above)

	Test Steps	**Test Data**	**Expected Result**	**P/F**
1	Gold is spawned around the map, in correct quantity.	Gold is visible to the player and is spawned throughout the map.	There is at least one piece of gold in each room.	F.
2	Monsters are spawned throughout the map.	The monsters are visible to the player and are spawned throughout the map.	The amount of monsters present in a single room is equal to 0 or 1. No more than one monster per room based on difficulty.	F.
3	Rooms are spawned throughout the map, with doors that can be entered.	The doors are visible to the player and can be bypassed.	No more than four doors per single room, and entering a door permits movement to the next room.	F.
4	Walls are spawned throughout the map.	The walls are visibly different than the doors to the player.	The walls cannot be bypassed by the player; walking up to the wall causes the player to stop.	F.

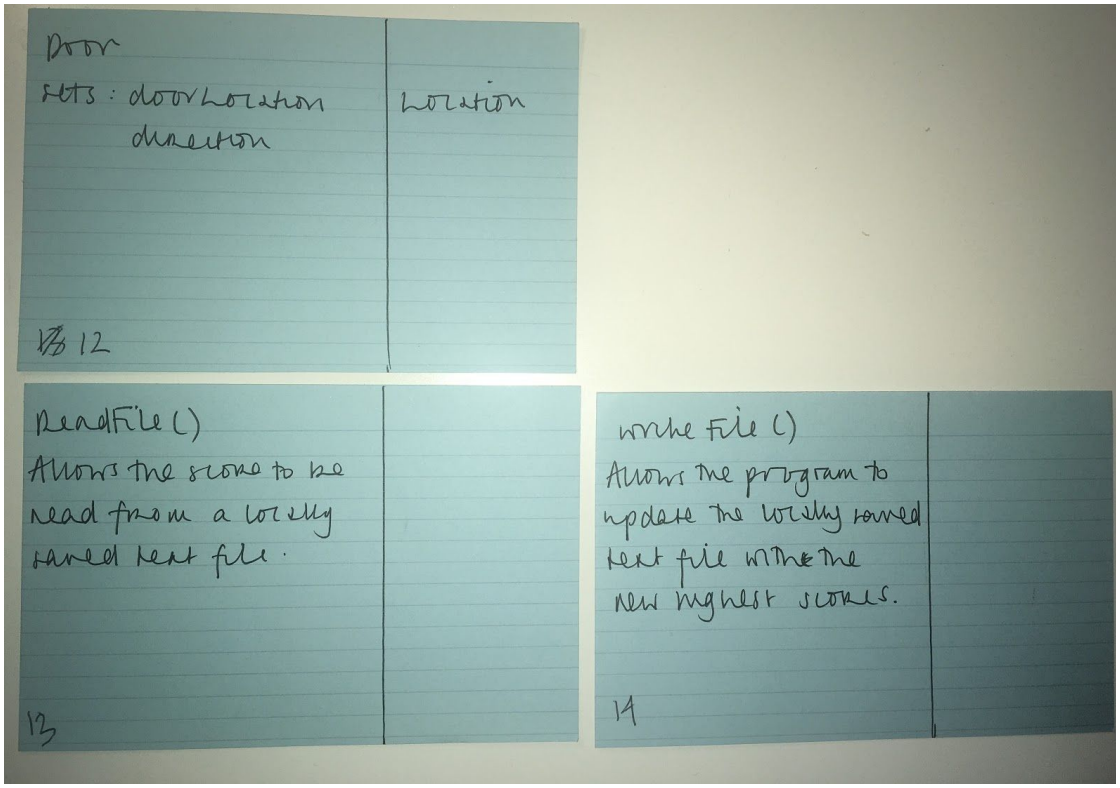
CRC cards 4 24/11

This version of CRC cards was made necessary by developments made during pair programming sessions. As documented in Exception Handling 3 (33), they were made slightly later than would have been optimum.




Map Pt 1.
•Sets location of :
- player - dungeon walls - dungeon doors - gold - bot - camera offsets - control offsets.
•Visualize :
dungeon - player location in room - bots in room - item locations in room
3
Location
Gold
Dungeon
Room
Bot
Item
Door
Leaderboard.
Map Pt 2
•Updates player's :
- location
- gold
- scores
- level
•Update leaderboard.
4
Map Pt 3
Allow user to :
input direction - check collision with : walls, doors, gold, bots - allow user to pick up objects - to go to the next level - restart the game after death - win the game - view the leaderboard - start a new game - exit the game.
5
Room
• sets : - Room Size
- Room location
- number of Rooms
- Room difficulty
- bot Flag
• sets up the layout of the room
- including the difficulty parameter + if any bots in the Rooms.
6
Location
Gold
Obstacle
Weapon
Bot
Dungeon
sets :
- number of Rooms
- dungeon difficulty
• places rooms on the global map.
7
Room
Gold
Door
Bot
Location
• sets : x, y, RoomNumber
• contains methods to print locations
• calculate distance between locations and check whether two locations are equal.
8


Bot
• sets : botlocation
• updates : bot's location
bot's health
location
Health
9
Item
• sets : itemlocation
location
10
Gold
sets :
• goldAmount
• goldlocation
location
Item
11


Door
sets : doorLocation
direction
location
12
ReadFile ()
Allows the score to be read from a locally saved text file.
13
writeFile ()
Allows the program to update the locally saved text file with the new highest scores.
14

6.0 Week 9 27/11–03/12/17

6.1 Overview

During this sprint, the team focussed on improving the code. While this included debugging and refactoring, it also involved adding some extra features. The sprint, therefore, included analysis and specification. The team created user stories, specified which should be implemented, and wrote tests accordingly. The use cases were also updated. Moreover, the team developed the user manual and started writing an installation guide for their game. As usual, processes and products were documented.

Sprint Review 5 03/12 Attendance: Full

This sprint had been productive. The team were pleased to have improved the game's graphics and added a theme song. This made the playing experience more attractive. The team had also implemented some basic features including a "level-up" feature. The team agreed that a focus for the following week would be adding some more advanced features. The team also discussed the failure to implement a camera class. The team reflected that they should have researched whether creating a camera was the most efficient way of resolving the issue of the dungeon spawning off-screen before attempting to install it. This would have prevented time being wasted. The team would evaluate at the beginning of the next sprint whether they should persist with the camera class, or find an alternative solution. Finally, the team were pleased that documentation for this sprint was completed and uploaded on Google Docs. This, combined with a frequently used online messenger group, ensured that all team members were aware of any progress.

6.2 Process

Backlog 5

Task	Member to complete	Date set	Deadline date
Document current sprint.	M.	27/11/17	03/12/17
Write Testing Records 3.	D and M.	27/11/17	28/11/17
Implement the player score.	D and M.	27/11/17	28/11/17
Create a camera class.	R and K.	27/11/17	N/A.
Creation of User Stories 5.	Everyone.	27/11/17	27/11/17
Creation of Use Case 3.	J and M.	27/11/17	27/11/17
Improve the user manual.	M.	27/11/17	03/12/17

Begin installation guide.	D.	29/11/17	03/11/17
Add collision logic to game.	R and K.	27/11/17	29/11/17
Improve the game's graphics.	R and K.	29/11/17	31/11/17
Limit the number of rooms in the dungeon without stray doors being present.	A and J.	29/11/17	03/12/17
Improve the amount of gold generated in large rooms.	A and J.	29/11/17	03/12/17
Produce a theme tune.	A.	29/11/17	03/12/17

Exception Handling 4 27/11

Discussed in: Sprint Review 5 (50).

Exception: We neglected to update our use cases. As a result, our tests and programming did not have the support which a current use case provides.

Handled: This was noted in a team meeting, and Use Case 3 (62) was created that day. In the future, tests and user stories should be compared to this use case. As the project develops, the use case should also be re-versioned.

Exception Handling 5 03/12

Discussed in: Sprint Review 5 (50).

Exception: We could not get the camera to work and therefore this task was not finished by the end of the sprint.

Handled: The camera class was difficult to create and there are other features which became a higher priority. The team decided that this issue would be the chief priority in the following sprint. They also noted that this was against agile principles. Tasks should be contained to the sprint they are set. The team would be careful not to overstretch themselves in future iterations. They also noted that in the future they should choose the method which is most simple and implement that first. It was too advanced to begin with the camera class.

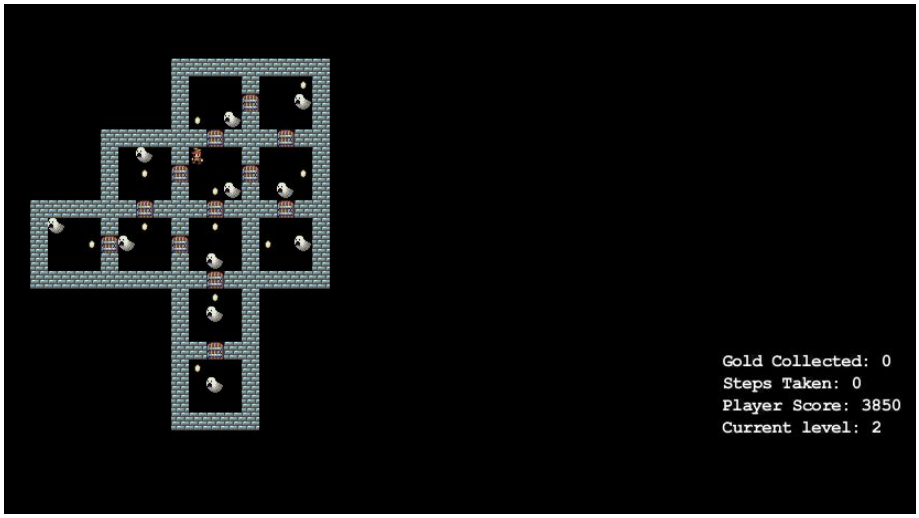
6.2.1 Meeting Records

Meeting 10 27/11 Attendance: Full. Length: 2 hrs.

1	Code explanation	This meeting began with a recap of what had been in the previous sprint. R and K explained their progress in Pair Programming 8 (40). At a shared computer, the team examined the user interface of their basic game, and read the code.
2	Basic improvements	The team agreed that a priority for this week would be making sure that the basic game did not have any obvious faults. A key issue was the fact that the dungeon spawned off-screen. This meant that user could not see certain rooms and therefore could not control their player fully. R and K had noticed this issue in Pair Programming 8 (40). They suggested making a camera feature. This would follow the player if it travelled off-screen. The team agreed. This, and other issues which had not been fixed in the following sprint, were listed. They made up part of Backlog 5 (50).
3	Advanced improvements	The team agreed that these noted issues were not major. There would also be time in this sprint to implement some more advanced features.
4	User Stories	To choose which advanced features they would implement the team decided to write some new user stories. These stories included level-up features and a camera class. They were collected in User Stories 5 (61), photographed and uploaded to Google Docs. D would write tests for these features, which M would check and enhance. Once the pairs had resolved any issues with the basic game, they could refer to User Stories 5 (61) and Testing Records 3 (63) for their next instructions.
5	Use Case	M observed that there had only been one written use case, documented as Use Case 2 (13). J said that this made programming and testing more difficult. They volunteered to re-version the use case after the meeting. This would be uploaded to Google Docs as Use Case 3 (62).
6	Manuals	M suggested that she would update the user manual further during the week. However, as all basic features had been noted, she thought that this could not be done until after more programming had been completed. M also thought that the installation guide could be completed. She offered to provide the framework, and D would fill it in.

Pair Programming 9 28/11 Pair: D and M. Length: 5 hrs.

Aim:	Implement a way of calculating player score and player level. This is the basic functionality for a leaderboard class (read and write).
CRC cards:	4.3Map; 4.4.Map; 4.5.Map; 4.13.readFile.java; 4.14.writeFile.java (48).
Use Case:	3.Pre-10 (62)
User Story:	5.1 (61).
Achieved:	Implemented a way of calculating player score and level. Player score was implemented by calculating the amount of steps it takes for players to get a piece of gold. This was the only logical way: having a time based score would not make sense in procedural generation. Every time a player gets 3 pieces of gold, the game is reset and the level increases by 1. At level 3 the

	game can be won. However, neither collision detection, nor a level-up feature, have been installed. Therefore, our tests currently fail.
Visualisation:	**Game Visualisation 6** 28/11 The bottom right of the image has a score tracker. During this pair programming session the 'Player Score' and 'Current Level' counters were added. 
Difficulties:	We could not calculate a score based upon time, due to procedural generation. We decided to create a 'step based' score system. Here the player reaches the top of the leaderboard by collecting the required amount of gold in the fewest steps. In the algorithm, a piece of gold was worth 1000 score and every step to this gold was worth -50.
Next Actions:	We should implement a leaderboard to display a collection of player scores.
Tests:	**Executable 3.1.2** 28/11 Corresponding Test: 3.1 (65).

	Test Steps	Test Data	Expected Result	P/F
1	Collect gold.	Collect 3 pieces of gold.	New map is generated, gold and step counter resets to 0, level increases by 1.	F.
2	Player score.	Collect multiple pieces of gold.	As you collect a gold coin, the score should increase (score is based on step count).	F.
3	Rooms are spawned throughout the map, with doors that can be entered.	The doors are visible to the player and can be bypassed.	No more than four doors per single room, and entering a door permits	F.

			movement to the next room.	
4	Walls are spawned throughout the map.	The walls are visibly different than the doors to the player.	The walls cannot be bypassed by the player; walking up to the wall causes the player to stop.	F.

Pair Programming 10 28/11 Pair: R and K. Length: 2 hrs.

Aim:	To add collision detection to the walls, golds, bots. Also, to install a camera class which would create the illusion of following the place around the map.
CRC cards:	4.3Map; 4.4.Map; 4.5.Map (48).
Use Cases:	3.Pre-5; 3.Pre-6; 3.Pre-7 (62).
User Stories:	3.1; 3.2; 3.3 (30); 4.1; 4.2(42) ; 5.4 (61) .
Achieved:	The player can now move within, and not outside, the map. It can also collect gold and die when it collides with the bot. The camera class was created as well.
Visualisation:	**Game Visualisation 6** 28/11 These two images detail the implementation of collision detection. These two images are screenshots of the same round of the game. They show the gold counter increase after gold has been collected by the player. 

Difficulties: It was difficult to install collision detection for the customised map. Also, the camera did not function as wanted or expected.

Next Actions: To improve the game's graphics and fix the issues with the camera class.

Tests:

Executable 3.1.3 28/11 Corresponding Test: 3.1 (64).

	Test Steps	Test Data	Expected Result	P/F
1	Collect gold.	Collect 3 pieces of gold.	New map is generated, gold and step counter resets to 0, level increases by 1.	F.
2	Player score.	Collect multiple pieces of gold.	As you collect a gold coin, the score should increase (score is based on step count).	P.
3	Rooms are spawned throughout the map, with doors that can be entered.	The doors are visible to the player and can be bypassed.	No more than four doors per single room, and entering a door permits movement to the next room.	P.
4	Walls are spawned throughout the map.	The walls are visibly different than the doors to the player.	The walls cannot be bypassed by the player; walking up	P.

			to the wall causes the player to stop.	

Executable 3.3.2 28/11 Corresponding Test: 3.3 (65).

	Test Steps	Test Data	Expected Result	P/F
1	Load a new game.	Game is loaded clicking start game on menu.	A 3x3 dungeon is displayed to the player.	F.
2	Move the player past the 3x3 boundaries.	Player sprite is moved using arrow keys into the rooms that are not displayed.	As player moves past the 3x3 dungeon, the camera adjusts and displays the new rooms.	F.
3	Move the player back within the original boundaries.	Player sprite is moved using arrow keys into the rooms that are not displayed.	The camera adjusts back to the original scene discussed in Step 2.	F.

Meeting 11 29/11 Attendance: Full. Length: 2.5 hrs.

1 Next Improvements

R and K noted they had referred to User Stories 5 (61) and noticed that the stories could be divided into two categories: extra-features and improved visuals. A said that he had too noticed this and that he was unsure which category should be a priority. A suggested that this should be a question in Customer Interview 5 (67). The team also questioned whether their game should be given a clearer theme. They had been using a generic “Dungeon Quest” idea in the earlier sprints. After looking at some possible graphics, the team decided that a haunted house theme looked exciting.

2 Preparation for Customer Interview

The team noted questions for the customer interview. These were stored in Google Docs.

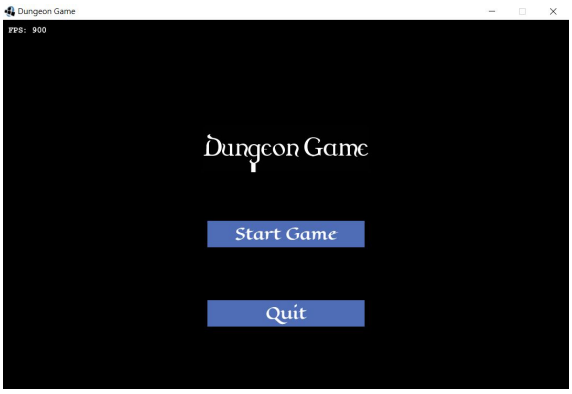
3 Customer Interview

The team asked their prepared questions. The answers were stored with the questions in Customer Interview 5 (67).

4 Interview Review

The customer had not voiced a specific preference for either visual enhancement or advanced features, but had suggested that the team should focus on whichever would improve the game-flow. R and K thought that the graphics were more important at this stage. They agreed to improve these in Pair Programming 11 (57). With reference to User Stories 5, A suggested that a background tune would accompany the improve graphics to create a better playing experience. He offered to produce one. The team agreed.

Pair Programming 11 01/12 Pair: R and K. Length: 4 hrs.

Aim:	To improve the game's graphics; to improve the game's logic flow; to deal with the issues in the camera class.
CRC cards:	4.1.StartMenu; 4.3Map; 4.4.Map; 4.5.Map (47).
Use Case:	3.Pre-3; 3.Pre-11; 3.Pre-8 (62).
User Story:	4.3 (42); 5.2; 5.3; 5.4 (61).
Achieved:	The graphics for the walls, the player, the door and game menu were changed.[3] An "ENTER" button was implemented to restart the game after the player died. Also, the level will increase if the player gets 3 gold coins.
Visualisation:	**Game Visualisation 7** 01/12 This image is of the start menu graphics.  This image is of the 'ENTER' game button.  This image is of our improved game graphics and map-regeneration after level increase .

[3] The source for the graphics is: https://opengameart.org/content/dawnlike-16x16-universal-rogue-like-tileset-v181.

Difficulties: There was not enough time to fix the camera.

Next Actions: Fix the camera.

Tests Used: **Executable 3.1.4** 01/12 Corresponding Test: 3.1 (64).

	Test Steps	Test Data	Expected Result	P/F
1	Collect gold.	Collect 3 pieces of gold.	New map is generated, gold and step counter resets to 0, level increases by 1.	P.
2	Player score.	Collect multiple pieces of gold.	As you collect a gold coin, the score should increase (score is based on step count).	P.
3	Rooms are spawned throughout the map, with doors that can be entered.	The doors are visible to the player and can be bypassed.	No more than four doors per single room, and entering a door permits movement to the next room.	P.
4	Walls are spawned throughout the map.	The walls are visibly different than the doors to the player.	The walls cannot be bypassed by the player; walking up to the wall causes the player to stop.	P,

Executable 3.4.2 01/12 Corresponding Test: 3.4 (66).

	Test Steps	Test Data	Expected Result	P/F
1	Start a new game.	Run the source code and click new game.	The game loads.	P.
2	Purposely interact with a bot.	Move the player sprite using the arrow keys until death.	The bot interacts with the player and the game is finished.	P.
3	A “You died” screen is shown.	Move the player sprite using the arrow keys until death.	Upon death, a new screen informing the player of there score is shown.	P.

Executable 3.3.3 31/11 Corresponding Test: 3.3 (65).

	Test Steps	Test Data	Expected Result	P/F
1	Load up a new game.	Game is loaded clicking start game on menu.	A 3X3 Dungeon is displayed to the player.	F.
2	Move the player past the 3x3 boundaries.	Player sprite is moved using arrow keys into the rooms that are not displayed.	As player moves past the 3x3 dungeon, the camera adjusts and displays the new rooms.	F.
3	Move the player back within the original boundaries.	Player sprite is moved using arrow keys into the rooms that are not displayed.	The camera adjusts back to the original scene discussed in Step 2.	F.

Pair Programming 12 01/12 Pair: A and J. Length: 4 hrs.

Aim:	To improve the dungeon generating algorithm so it removes doors leading nowhere. Make the gold amount is more suitable for larger rooms.
CRC cards:	4.3Map; 4.4.Map; 4.5.Map; 4.6.Room; 4.7.Dungeon; 4.8.Location (48).
Use Case:	3.Pre-5; 3.Pre-6 (62).
User Story:	3.3 (30); 4.1 (42).

Achieved:	The dungeon generating algorithm runs to a limit without storing redundant doors (i.e. doors leading off the map). Gold is more sparse in big rooms so as not to make the game too easy.
Visualisation:	No visualisation.
Difficulties:	No difficulties.
Next Actions:	Code for removing redundant doors could be more efficient and condensed.

Tests:

Executable 1.4.6 01/12 Corresponding Test: 1.4 (18).

	Test Steps	Test Data	Expected Result	P/F
1	Start a new game.	Press enter when game is loaded.	A dungeon is generated.	P.
2	Start a new game.	Press enter when game is loaded.	A dungeon with a specific number of rooms is generated.	P.
3	Exit the game.	Press Escape to exit the dungeon.	The game is exited.	P.
4	Try to start a new game again.	Press enter when game is loaded	A new, visually different dungeon is generated around a player.	P.

Executable 2.4.2 01/12 Corresponding Test: 2.4 (46).

	Test Steps	Test Data	Expected Result	P/F
1	Gold is spawned around the map, in correct quantity.	Gold is visible to the player and is spawned throughout the map.	There is at least one piece of gold in each room.	P.
2	Monsters are spawned throughout the map.	The monsters are visible to the player and are spawned throughout the map.	The amount of monsters present in a single room is equal to 0 or 1. No more than one monster per room based on difficulty.	P.
3	Rooms are spawned	The doors are visible to the	No more than four doors per single	P.

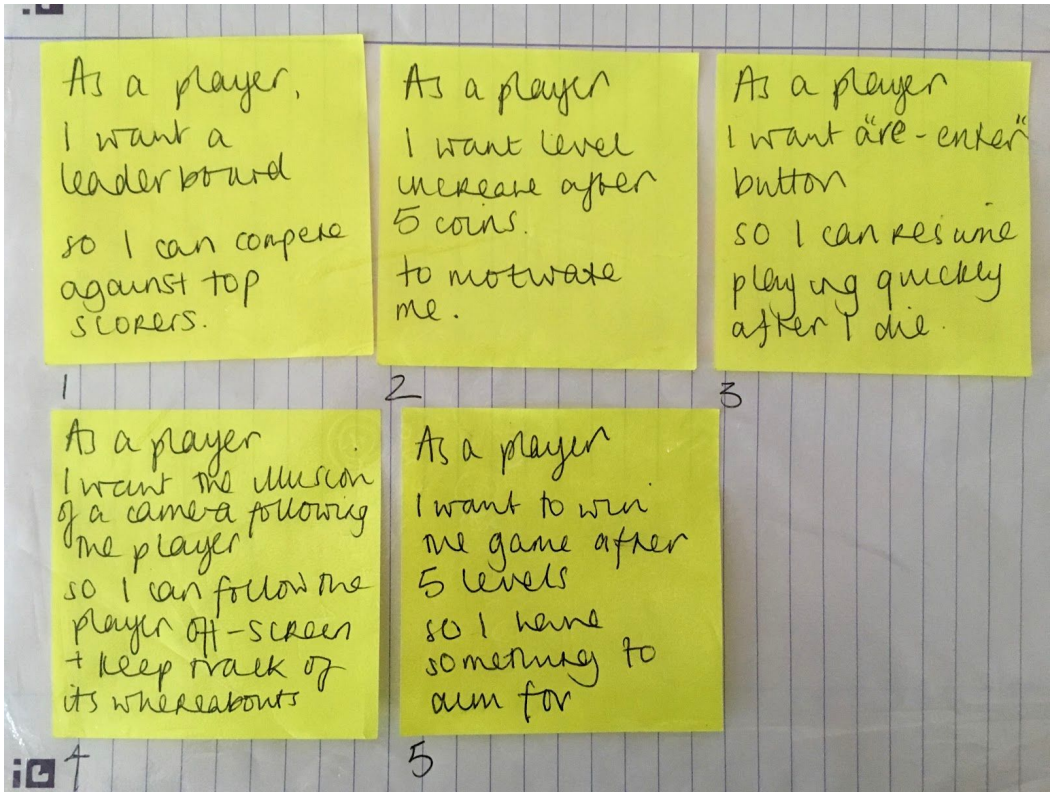
	throughout the map, with doors that can be entered.	player and can be bypassed.	room, and entering a door permits movement to the next room.	
4	Walls are spawned throughout the map.	The walls are visibly different than the doors to the player.	The walls cannot be bypassed by the player; walking up to the wall causes the player to stop.	P.

6.3 Products

This sprint the team mainly focussed upon refactoring and debugging: these endeavours did not produce any agile products. However, the team's decision to add some further features to the game required them to create the user stories and testing records detailed below.

User Stories 5 27/11

These user stories were created at the beginning of Week 9. They detail more advanced features than previous versions of user stories including level-increase and a leaderboard. Testing Records 4 (75) correspond with these stories.



Use Case 3 27/11

This use case differs from Use Case 2 (13). Rather than providing the system outline of an online multiplayer game, it designs an offline, singleplayer game. In this, and the features which it includes, it is more similar to the system which we have installed at this stage.

3.1 Use Case: Play an offline, singleplayer game.

3.2 Author: Team Grey

3.3 Date: 27/11/17

3.4 Purpose: Play a game to beat friends high score.

3.5 Overview: A player would download our java code and run the game through Eclipse. A start menu will allow the player to input a nickname for a leaderboard. Once the player clicks on the Start button the game will begin.

The player then uses keyboard controls to walk around a set of rooms picking up items and interacting with bots along the way. Once the player has picked up 3 gold coins the player can then progress to the next level. The player collects 9 coins to win the game.

3.1 Alternative: The player interacts with a bot and loses the game.

The game then updates the leaderboard with the player's score and nickname and the player has the option to play again or quit.

3.6 Cross References: User Stories 1.1; 1.4; 1.5; 1.6; 1.8; 2.1; 2.5 (5; 10).

3.7 Actors: Player

3.8 Pre-Conditions:

3.Pre-1: The game must have a web link to access the game code.

3.Pre-2: The game must have a user interface for the player to use.

3.Pre-3: The game must have a start menu with “Start” and “Quit” buttons.

3.Pre-4: The game must have an in-game menu.

3.Pre-5: The game must have a set of rooms in a dungeon the character can move in.

3.Pre-6: The game must have evenly distributed gold that can be picked up.

3.Pre-7: The game must allow players to interact with evenly distributed bots.

3.Pre-8: The game must have 3 levels.

3.Pre-9: The game must have winning conditions.

3.Pre-10: The game must be able to calculate the player’s score.

3.Pre-11: The game must have a “You Die” screen with an option to re-start the game.

3.Pre-12: The game must have a quit button.

3.9 Postcondition: The leaderboard is updated with player scores.

3.10 Flow of events:

Actor's actions:

1. Player inputs their chosen nickname.
2. System stores the player's nickname.
3. Player begins the game.
4. Player chooses where to move and what to interact with.
5. System reads the input moves of the player. For each input move the bot locations are updated.
6. System updates and checks how much gold the player has collected.
7. System checks whether the player has interacted with a bot so the game can end when the player stands on the same tile as a bot.
8. Players wins the game by collecting 9 gold coins.
9. Leaderboard is updated when the player wins.
10. Player quits the game.

3.11 Alternative flow of events:

- Step 1: Player wants to check the leaderboard to see what score they need to beat. The player can then go back and play the game.
- Step 8: Player is attacked by a bot and the game is lost. Go to step 1.
- Step 8: The player gets bored of the game and quits. Go to step 10.

3.12 Exceptional flow of events:

- Power failure during gameplay. The leaderboard is updated with most recent score at Step 9.

Testing Records 3 28/11

These testing records correspond with User Stories 5. They detail some advanced features which the team were committed to incorporating into their game.

Test 3.1 27/11

Test Priority:	Medium.
Test Title:	Level increment, score and collision test.
Description:	Testing: that the level increases after every 3 gold is collected; that score increases after each gold is collected; and, that the player responds to objects in the map correctly.
CRC Card:	4.3Map; 4.4.Map; 4.5.Map (48).
Use Case:	3.Pre-8; 3.Pre-10 (62).
User Story:	5.1; 5.2 (61).

Executable 3.1.1 27/11 Corresponding Test: 3.1 (above)

Step	Test Steps	Test Data	Expected Result	P/F
1	Collect gold.	Collect 3 pieces of gold.	New map is generated, gold and step counter resets to 0, level increases by 1.	F.
2	Player score.	Collect multiple pieces of gold.	As you collect a gold coin, the score should increase (score is based on step count).	F.
3	Rooms are spawned throughout the map, with doors that can be entered.	The doors are visible to the player and can be bypassed.	No more than four doors per single room, and entering a door permits movement to the next room.	F.
4	Walls are spawned throughout the map.	The walls are visibly different than the doors to the player.	The walls cannot be bypassed by the player; walking up to the wall causes the player to stop.	F.

Test 3.2 28/11

Test Priority:	Medium.
Test Title:	Leaderboard test.
Description:	Testing that the leaderboard functions correctly.

CRC Card:	4.13.ReadFile; 4.14.WriteFile (49).
Use Case:	3.Pre-10; 3.9 (62).
User Story:	5.1; 5.2 (61).

Executable 3.2.1 28/11 Corresponding Test: 3.2 (above)

	Test Steps	Test Data	Expected Result	P/F
1	Play a game.	Play the game as usual, achieve a score.	If score is higher than a position on the leaderboard, the score overwrites that position.	F.
2	Check the text file is saved when the player dies.	Play the game as usual, achieve a score.	Achieve a high score and close the game - see if the new high score appears in leaderboard.txt	F.
3	Check the text file is read when the game is opened.	Play the game as usual, achieve a score.	Change information in the leaderboard.txt, open a new game and see if this information is present on the leaderboard.	F.

Test 3.3 28/11

Test Priority:	Low
Test Title:	Camera Test
Description:	Tests that the screen follows player if they move off screen.
CRC Card:	N/A. A CRC card for this was not created.
Use Case:	3.Pre-5 (62).
User Story:	5.4 (61).

Executable 3.3.1 28/11 Corresponding Test: 3.3 (above)

	Test Steps	Test Data	Expected Result	P/F

1	Load a new game.	Game is loaded clicking start game on menu.	A 3x3 dungeon is displayed to the player.	F.
2	Move the player past the 3x3 boundaries.	Player sprite is moved using arrow keys into the rooms that are not displayed.	As player moves past the 3x3 dungeon, the camera adjusts and displays the new rooms.	F.
3	Move the player back within the original boundaries.	Player sprite is moved using arrow keys into the rooms that are not displayed.	The camera adjusts back to the original scene discussed in Step 2.	F.

Test 3.4 28/11

Test Priority:	Low.
Test Title:	'You Died' screen and re-enter button test.
Description:	Tests the death screen is present upon getting hit by a bot and that the player can restart a game using the enter key.
CRC Card:	4.5.Map (48).
Use Case:	3.Pre-11 (62).
User Story:	5.3 (61).

Executable 3.4.1 28/11 Corresponding Test: 3.4 (above)

	Test Steps	**Test Data**	**Expected Result**	**P/F**
1	Start a new game.	Run the source code and click new game.	The game loads up.	P.
2	Purposely interact with a bot.	Move the player sprite using the arrow keys until death.	The bot interacts with the player and the game is finished.	P.
3	A "You died" screen is shown.	Move the player sprite using the arrow keys until death.	Upon death, a new screen informing the player of there score is shown.	F.

4	The player can restart a game.	On death, the player can press enter to restart the game.	Pressing Enter runs the game again, counters are reset to zero.	F.

Customer Interview 5 29/11

Similarly to Customer Interview 4 (43), the team mainly focussed upon their documentation. By asking questions about presentation, and what to include in their final submission, the team were ensuring that their project would cover all requirements as thoroughly as possible. However, as the team had finished their basic game, they also asked the customer what type of improvements would be prefered as they proceeded.

1. Would you prefer future improvements to be visual or functional?
Does not necessarily have to be an "either or". The overall aim is to make the game more engaging. To achieve this you could ramp up the improvements on both aspects incrementally and together.

2. Does this pair programming document supply sufficient detail?
It is actually too detailed. You should remove the three categories which evaluate the experiences using pair-programming, TDD and the implementation language for each programming session.

3. Do we need to provide screenshots of our code when documenting TDD?
No. The code will also be submitted, so this is not necessary. You should instead submit a testing document with the number of the test, what the test does and pass or fail information etc.

4. The document is currently sectioned by overview, process and products, and then chronologically within those sections. Is this preferable?
You should choose an order which is going to be the most simple for the reader to understand. Beyond that any particular order is not important. Cross reference documents will also help you with making the document clear.

7.0 Week 10 04–10/12/17

7.1 Overview

The aim for this sprint was ensuring that the game would be completed by the following week. The team therefore focussed upon: refactoring, debugging, and ensuring any recently created features were merged with the code. Pair programming and TDD were therefore the main practices used during the sprint. Moreover, as the team were all working on existing code, communication tools were essential. Online messenger was useful for ensuring an absence of merging conflicts. During this sprint, the team also worked upon their guides: the user manual was updated with recent features; the installation guide was

completed; and, the maintenance guide was drafted. As usual, each piece of documentation was checked and formatted throughout the sprint.

Sprint Review 6 10/12 Attendance: Full

Overall, the team were pleased to have a game which could be handed in by the deadline. The issue with the dungeon spawning offscreen from the previous sprint had been resolved. The documentation was up-to-date and all of the guides had been worked on. The team concluded by noting that the main issue with their current code was that it did not follow an MVC pattern. This was because of the communication issues in Week 8 (see Exception Handling 3, 32). In the following sprint they would attempt to apply the pattern.

7.2 Process

Backlog 6

Task	Member to complete	Date set	Deadline date
Document current sprint.	M.	04/12/17	10/12/17
Create Testing Records 4.	D and M.	04/12/17	05/12/17
Implement leaderboard.	D and R.	04/12/17	04/12/17
Implement winning conditions.	J and A.	04/12/17	05/12/17
Implement camera alternative.	R and K.	04/12/17	10/12/17
Change bot movements at final stage of game.	J and A.	04/12/17	07/12/17
Start maintenance guide.	K.	04/12/17	10/12/17

Exception Handling 6 05/12

Discussed in: Pair Programming 14 (71); Sprint Review 6 (68).

Exception: Our work is not split into MVC, as this task is too time-consuming to add to this sprint.

Handled: We referred to the previous sprint in which we had taken too much work on, and not finished it (see Exception Handling 5, 51). Therefore, before starting any work on the task, we decided to leave this task for our final sprint.

7.2.1 Meeting Records

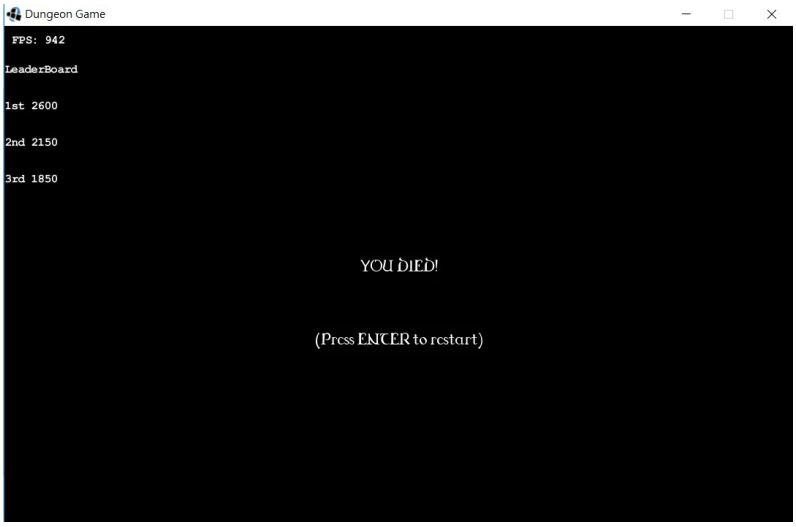
Meeting 12 04/12 Attendance: Full. Length: 1 hrs.

1	Camera Class	With reference to Sprint Review 5 (50), the team began this sprint by deciding how to handle the issue with the camera class. R and K discussed how they found it difficult and time-consuming to implement. On balance, the team decided not to make a camera class and a work-around solution was formed. Instead of following the player off screen, they would change the map so that it spawned in the center of the screen and no rooms were offscreen. R and K would implement this. The team were satisfied with this solution.
2	Theme tune	A noted that another incomplete feature, from the last sprint, was the game's theme tune. It had not yet been integrated with the game. D volunteered to do this.
3	Winning conditions	The leaderboard still needed to be created. D and R said they would make this. J and A also noted that the game did not yet have winning/finishing conditions. By implementing the gold in Week 8, they had began the task of implementing these. However, now a defined winning procedure should be put in place. The team decided that after 3 coins had been reached on the final level, a portal should appear. The player would then have to reach the portal to win. The team also considered making reaching the portal more challenging. This could be done by changing the bot settings once the portal has appeared. The team discussed 3 options: 1) the bots can move through internal walls; 2) the bots can move anywhere on the screen; 3) the bots can move through doors, like a player. The team's preference was the first option. A and J would attempt to implement this.
4	Maintenance Guide	M noted that most of the game classes had been created. It was therefore possible to begin writing the maintenance guide. K would create a first draft this week.

Pair Programming 13 04/12 Pair: R and D. Length: 4.5 hrs.

Aim:	To implement a leaderboard based on the readFile and writeFile classes created in Pair Programming 9 (52). Implement the game's theme tune, using the .ogg files created by A.
CRC cards:	4.13.ReadFile; 4.14.WriteFile (49).
Use Case:	3.Pre-10; 3.9 (62).
User Story:	5.1; 5.2 (61).
Achieved:	Basic leaderboard functionality has been implemented. It records the first, second and third highest scores, and overwrites the previous scores if a new one is set. These scores are stored in a local text file named "leaderboards.txt". Game sound has been implemented using Slick2D.
Visualisation:	**Game Visualisation 8** 04/11

This image displays the leaderboard. After the player ‘dies’, the user’s score is displayed and can be compared to the 1st, 2nd and 3rd top scores previously achieved.



Difficulties:

Using .ogg files would cause the program to crash, this was fixed by using .wav files but the reasoning for the .ogg crash was unknown. Also, the leaderboard currently only saves the high scores. The is no identification associated with them.

Next Actions:

Associate an identifier with the high scores (e.g. ABC). Implement a global leaderboard which allows players to compete with other players regardless of location. Improve the graphical user interface for leaderboard.

Tests:

Executable 3.2.2 04/12 Corresponding Test: 3.2

	Test Steps	**Test Data**	**Expected Result**	**P/F**
1	Play a game.	Play the game as usual, achieve a score.	If score is higher than a position on the leaderboard, the score overwrites that position.	P.
2	Check the text file is saved when the player dies.	Play the game as usual, achieve a score.	Achieve a high score and close the game - see if the new high score appears in leaderboard.txt	P.
3	Check the text file is read when the game is opened.	Play the game as usual, achieve a score.	Change information in the leaderboard.txt, open a new game and see if this information is	P.

			present on the leaderboard.	

Pair Programming 14 05/12 Pair: R and K. Length: 2 hrs.

Aim:	To: get a camera working; create an executable file so that players can run the game with ease; separate the MVC architecture within Map.java; add a dancing skeleton to the game's end.
CRC cards:	4.3Map; 4.4.Map; 4.5.Map; 4.6.Room (48).
Use Case:	3.Pre-1; 3.Pre-5; 3.Pre-11 (62).
User Story:	5.3; 5.4 (61).
Achieved:	A basic work around for the camera illusion: the window is now shifted so that if the player goes offscreen, the player is not followed. The map is made more central,, so that the dungeon should not spawn off the screen. Also, an executable .jar and .exe file was created.
Visualisation:	**Game Visualisation 9** 05/12 This image presents the updated 'You Died!' screen which now contains a dancing skeleton. 
Difficulties:	After reflection, splitting Map.java into MVC will be time consuming. This will be left until the final sprint (Exception Handling 6, 68).
Next Actions:	To check the coding conventions and comments in our code so far.
Tests used:	**Executable 4.2.2** 05/12 Corresponding 4.2 (76).

	Test Steps	Test Data	Expected Result	P/F

1	Load the game and select ‘Start Game’.	Run the executable file or load from source code.	A new dungeon is generated that is central to the map.	P.
2	Close the game.	Press the Esc key to close the game.	Game closes.	P.
3	Load up the game again and select ‘Start Game’.	Run the executable file or load from source code.	A new, different dungeon is generated that is also central to the map.	P.

Pair Programming 15 05/12 Pair: J and A. Length: 3 hrs.

Aim:	To add winning conditions to the game.
CRC cards:	4.3Map; 4.4.Map; 4.5.Map (48).
Use Case:	3.Pre-9 (62);
User Story:	5.5 (61).
Achieved:	Winning conditions were created. When the gold requirement is reached on the last level, a new location is generated in the middle of a room with an image occupying the tile. When the player stands on the tile a win boolean variable changes to true.
Visualisation:	**Game Visualisation 10** 05/12 This image presents the portal appearing when the 9th coin (or 3rd coin in 3rd level) has been collected. It is represented by a coin, rather than a moving portal, for the purposes of incremental implementation. 
Difficulties:	Initially we attempted to generate a new door through which the player could exit the game. However, this proved too difficult. The collision logic for the existing walls prevented this

	method from working. We decided that having the exit in the middle of the room would not actually impact the gameplay or visuals enough to justify the time necessary to create a complicated workaround.
Next Actions:	Add a win screen and a way to save your score after the win condition has been met.
Tests used:	**Executable 4.1.2** 05/12 Corresponding Test: 4.1

	Test Steps	Test Data	Expected Result	P/F
1	Load the game and select 'start game'.	Run the executable file or load from source code.	Game opens and test can run.	P.
2	Play the game as usual, collecting gold and avoiding bots.	Move player using arrow keys over gold icons until final level is reached.	When 9 gold coins have been collected, a portal appears.	P.
3	Move player through the portal that has spawned.	Move player using arrow keys over the portal icon.	A winning screen is displayed that is visually different from the losing screen. The leaderboard is displayed and the high score is updated.	F.

Meeting 13 06/12 Attendance: Full. Length: 2 hrs.

1	Preparation for Customer Interview	The team wrote questions for the pending customer interview. As the project completion date was approaching, the team decided to seek the customer's advice about whether they should implement further features. The team also wanted to confirm that their documentation was fulfilling requirements.
2	Game Demo	The team demonstrated their game to a supervisor. The team were pleased with the feedback.
3	Customer Interview	The team asked the customer their pre-prepared questions. These, and their corresponding answers, were recorded as Customer Interview 6 (77) in Google Docs.
4	Interview Review	With reference to the customer's suggestions, the team decided to finish implementation this sprint. The winning conditions would be the final addition. The team also discussed the presentation section of the project. They decided they would meet up several times before its date to refresh their understanding of the game.

Pair Programming 16 07/12 Pair: J and A. Length: 3 hrs.

Aim:	To change the bot's movement so that it is more random. Also, add different movement parameters to the bot on the final level.
CRC cards:	4.3Map; 4.4.Map; 4.5.Map; 4.6.Room; 4.9.Bot (48).
Use Case:	3.Pre-7 (62).
User Story:	4.2 (61).
Achieved:	The bot's movement settings have been changed. The bot now has a chance to not move when the player makes a move. This resolves the previous issue that if a bot was an odd number of steps from the player, they were unable to kill the player regardless of any player moves. Also, now if the winning condition has been met and the win tile has appear, the bots will be able to move through the doors, like a player. This makes it more challenging to reach the final dungeon end point.
Visualisation:	**Game Visualisation 11** 07/12 This image is of the final stage of a game. As bots are allowed to move through doors, a portal and two (rather than one) ghosts can be seen in the bottom left room. This will make it more difficult for the player to reach the portal and win the game. 
Difficulties:	Initially we wanted the bot to be able to move through the dungeon's internal walls, while still being contained by the outer wall. However, we realised this would require the code to be significantly restructured. It was not sensible to undertake this development so close to the game submission. Also, this feature might have made it too challenging for the player to reach the exit, and win the game. We contacted the team via online messenger, and they agreed with us. We decided to use option 3 from the Meeting 12 (69).
Next Actions:	To decide conclusively which of the three aforementioned methods of bot movement has the best balance of being both possible to implement and most fun for the player.
Tests:	**Executable 2.3.3** 07/12 Corresponding Test: 2.3 (45).

	Test Steps	Test Data	Result	P/F

1	Move player to room with zombie.	Using arrow keys move through the doors.	Bot moves one step as the player moves one step.	P.
2	Move player into the bot.	Walk player into the bot.	The bot removes a player life, the player dies and is shown the “you died” screen.	P.
3	Bot does not leave its room boundaries.	Move player around and see if bot leaves the room it is assigned to.	The bot stays in the walls, it does not leave the room it is assigned to (unless it is the final level).	P.
4	Bots movements are random.	Move player around and see if bot moves randomly every time.	Bot moves randomly, no pattern demonstrated.	P.

7.3 Products

The products in this sprint consist of testing records and a customer interview. This reflects the team’s ambition to finish their game and fix any issues, rather than implement various new features.

Testing Records 4 05/12

These testing records correspond with the decisions made in Meeting 12 (69). They test for clearly defined winning conditions, and a camera-work around which requires the map to be central. This is the final set of testing records, as these are the final features implemented.

Test 4.1 05/12

Test Priority:	High.
Test Title:	Winning conditions test.
Description:	Tests that the game is won after the player has collected 9 coins and moved through portal.
CRC Card:	4.3Map; 4.4.Map; 4.5.Map; 4.6.Room; 4.9.Bot (48).
Use Case:	3.Pre-9 (62).
User Story:	5.5 (61).

Executable 4.1.1 05/12 Corresponding Test: 4.1 (above)

	Test Steps	Test Data	Expected Result	P/F
1	Load the game and select ‘start game’.	Run the executable file or load from source code.	Game opens and test can run.	P.
2	Play the game as usual, collecting gold and avoiding bots.	Move player using arrow keys over gold icons until final level is reached.	When 9 gold has been collected, a portal appears.	F.
3	Move player through the portal that has spawned.	Move player using arrow keys over the portal icon.	A winning screen is displayed that is visually different from the losing screen. The leaderboard is displayed and the high score is updated.	F.

Test 4.2 05/12

Test Priority:	Low
Test Title:	Dungeon spawn (camera alternative).
Description:	Tests that the dungeon spawns centrally, and therefore does not expand beyond the screen.
CRC Card:	4.3Map; 4.4.Map; 4.5.Map; 4.6.Room (48).
Use Case:	3.Pre-1; 3.Pre-5; 3.Pre-11 (62).
User Story:	5.3; 5.4 (61).

Executable 4.2.1 06/12 Corresponding Test: 4.2 (above)

	Test Steps	Test Data	Expected Result	P/F
1	Load the game and select ‘Start Game’.	Run the executable file or load from source code.	A new dungeon is generated that is central to the map.	F.
2	Close the game.	Press the Esc key to close the game.	Game closes.	P.

3	Load up the game again and select 'Start Game'.	Run the executable file or load from source code.	A new, different dungeon is generated that is also central to the map.	F.

Customer Interview 6 06/12

As this was the team's penultimate sprint, these questions focus upon documentation and coming to the end of the software development lifecycle. The team sought assurance that their chosen theme was acceptable and that their system guides fulfilled requirements. Moreover, the team were uncertain as to how they should manage their last sprints, and sought advice from the customer in the capacity of consultant.

1. Our game has a "haunted house" theme, is this acceptable?
Yes, of course. I did not specify a certain theme. There was room to choose your own. It will be interesting to see which directions the different teams have taken their game in.

2. At what point do you recommend we stop coding and focus solely on documentation?
You will need to decide on balance what is going to work best for you. Ask yourselves: will adding "X" story, and the cost of documentation which goes with that, be possible. For example, certain things will not be feasible. Taking the game from singleplayer, to multiplayer in a week would probably not be possible.

3. What should the 'System Requirements' of the installation guide include?
If your game can be run by a variety of systems, then just give an overview of which. If you are offering a jar and an exe, then include this. Also, if there is a runtime dependency, include it.

4. We have started the maintenance guide by detailing our classes, what else should it include?
The guide should also make connections between the classes. You should think of your guide as map of your system. You want to direct a programmer around it as simply and thoroughly as possible. Ask yourselves: "if I were a developer coming to this game, what details would I need to know?". Should also direct future implementations. Guide a programmer through what could be added/changed and how this could be done.

5. What format will the group presentation be in? How should we prepare for it?
It will be a 15 minute presentation in which you will demo your game. You will be asked questions: just for further explanations about something if it looks interesting. And then there will also be a 30 minute retrospective in which you will evaluate this process.

8.0 Week 11 11–15/12/17

8.1 Overview

As this final sprint was half the length of previous sprints, the team decided to not implement any further features. The remaining time was used for ensuring that the game and its documentation were ready to be submitted. The team also had a meeting in which they discussed their experience of using agile and creating a game.

Sprint Review 7 14/12

The team were pleased with this sprint. They did not have to implement anything in a rush, as the progress had been incremental and they had been realistic with what they had wanted to achieve from the start of the process. The team were also pleased that they had managed to re-version the CRC cards and refactor the game accordingly. While their finished game did not entirely follow an MVC pattern, it was an improvement from previous sprints and the team had learnt a lot about the implementation of MVC and its relation to Java. The team also noted that by deciding to have a Product Master mainly focussed upon documentation from the start, the bulk of the large team submission was completed before this sprint. It was now a matter of: checking spelling and grammar, and ensure the cross-references were correct and given page numbers. Similarly, the guides were almost completed before this sprint. The team simply had to improve these with decorations and screenshots. Working on a just enough basis had served the team well, and overall the team enjoyed their project.

8.2 Process

Backlog 7

Task	Member to complete	Date set	Deadline date
Write CRC Cards 5.	Everyone.	11/12/17	11/12/17
Refactor code.	K and R.	11/12/17	11/12/17
Finish the large team submission.	M.	11/12/17	14/12/17
Write the opening paragraphs for the user manual.	M.	11/12/17	14/12/17
Finish the user manual.	D and K.	11/12/17	14/12/17
Finish the maintenance guide.	J and A.	11/12/17	14/12/17

Review the project and fill in the group questionnaire.	Everyone.	11/12/17	14/12/17
Check the project submission materials and confirm satisfaction with the standard.	Everyone.	14/12/17	15/12/17

Exception Handling 7 11/12

Discussed in: Meeting 14 (79); Sprint Review 7 (78).

Exception: Our game was not split into MVC architecture. The code therefore had not entirely utilised the benefits of using an object oriented programming language.

Handled: The team wrote new CRC cards, collected them in CRC Cards 5 (80). They refactored their work according to these. While the finished code was not perfectly sectioned, it had been improved upon significantly.

8.2.1 Meeting Records

Meeting 14 11/12 Attendance: Full. Length: 1.5 hrs.

1 MVC and CRC: It was noted that the game was not split into MVC. This was due to the confusion detailed in Exception Handling 3 (33). The team re-versioned their CRC Cards, this allowed them to plan the imminent refactoring. The cards were collected in CRC Cards 5 (80). R and K would amend the code later that day.

2 Finalising theme and name: The team began their final sprint by choosing a title for their game. The team thought that 'Get Rich or Die Trying' was a good title. It was humorous and had a clear message. The sentiments were closely linked with the game's objectives.

3 System guides: M noted that the guides could now be completed. Previously certain screenshots had not been included, as the game's title was needed. She also volunteered to write the opening paragraph for the user manual as aligned with their theme. K said she would decorate the manual accordingly. The team spoke about the other guides. D confirmed that the installation guide was almost finished, it simply needed screenshots with the game's title. K said that the maintenance guide was almost complete. It still need some points about further directions in which the game could be taken. J and A agreed to write these. The team agreed that when each guide was finished they would alert each other via online messenger. All members would have a chance to check the documents before they were submitted.

4 Large team submission.: M confirmed that the larger team submission was almost finished. It simply required: the documentation of this sprint to be added; proofreading; and, for each of the cross-references to be given page numbers. The team each agreed to proofread the document before submission.

This would confirm that they were all satisfied with the standard, any issues could be voiced and resolved with the Google Docs comment tool.

Pair Programming 17 11/12 Pair: K and R. Length: 4 hrs.

Aim:	To split the code into an MVC pattern.
CRC cards:	CRC Cards 5 (80).
Use Case:	Use Case 3 (62).
User Story:	N/A.
Achieved:	The code was refactored as planned in CRC Cards 5 (80).
Visualisation:	No visualisation.
Difficulties:	At this stage it was not practical to attempt to split the program entirely into an MVC pattern. We instead decided to refactor the code as planned in CRC Cards 5 (80).
Next Actions:	Ensure the game is in the correct format to be submitted.
Tests:	N/A.

Meeting 15 13/12 Attendance: Full. Length: 1.5 hrs.

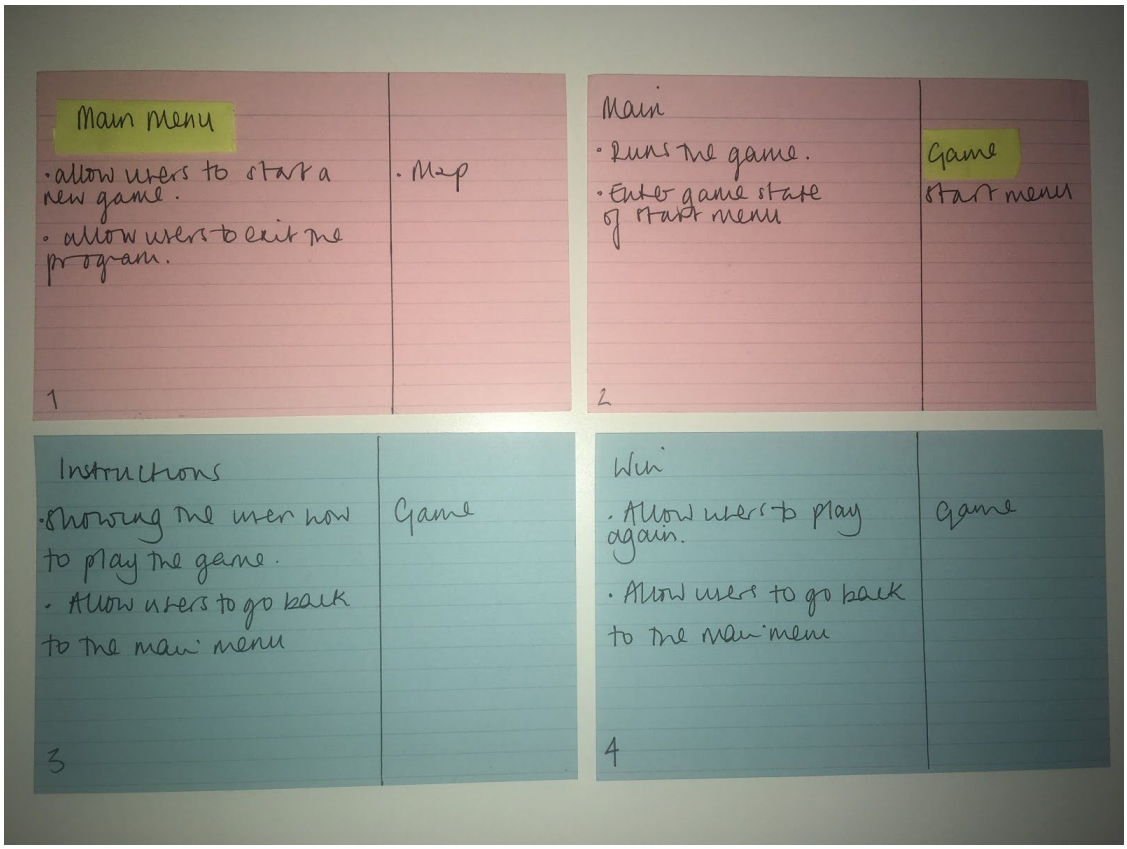
1	Customer Interview	The team noted that the customer had emailed an option to cancel the final interview. The team decided this was a sensible option as they had finished with the analysis stage of their process. They were now focussed upon their project's finishing touches.
2	Review and Questionnaire	Instead of writing questions and interviewing the customer, the team decided to review their process and fill in the questionnaire. The group were in agreement that the work for this project had been spilt fairly. Each team member had worked very hard and had been given the opportunity to excel. The team were pleased with their experience.
3	Confirmation of submission	The team decided to finish their project by the next day. This was a day before the deadline. This would give all team members a chance to give everything a final check and confirm they were satisfied with the quality. This would be the last meeting which the group would record. The next meeting would be the final submission.

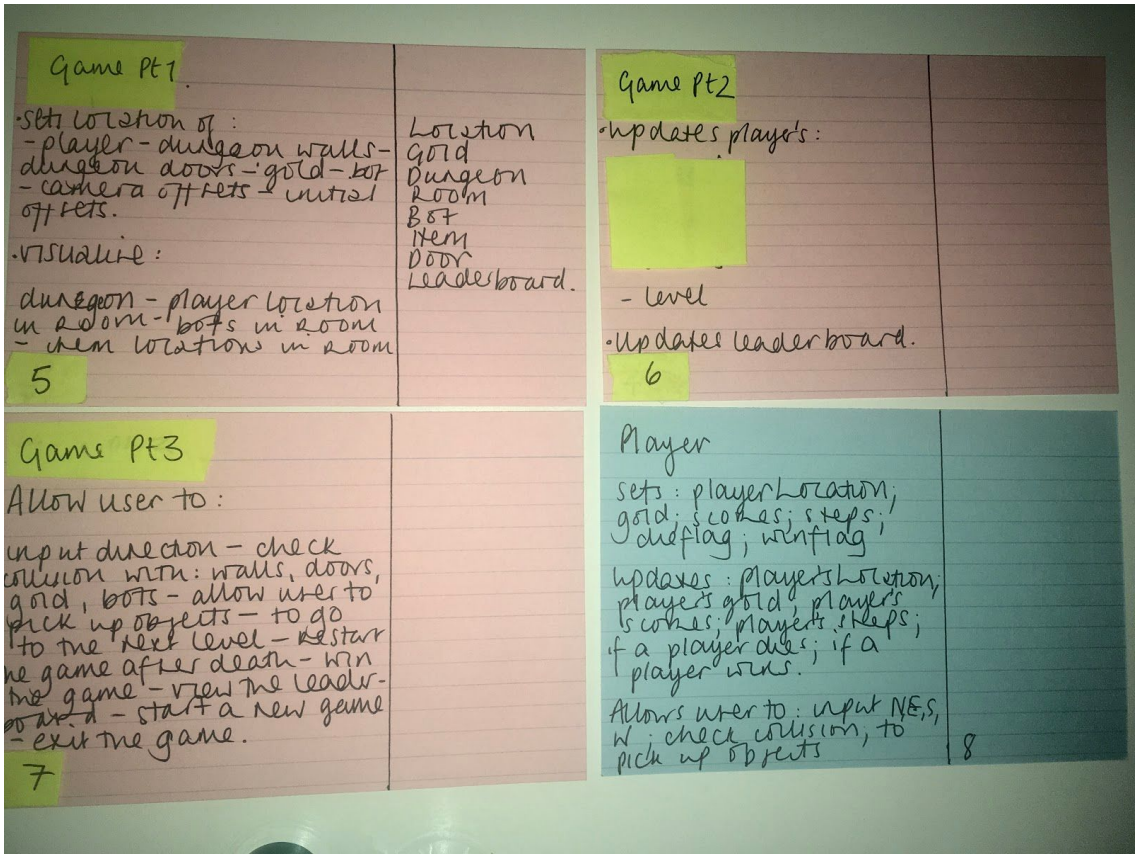
8.3 Products

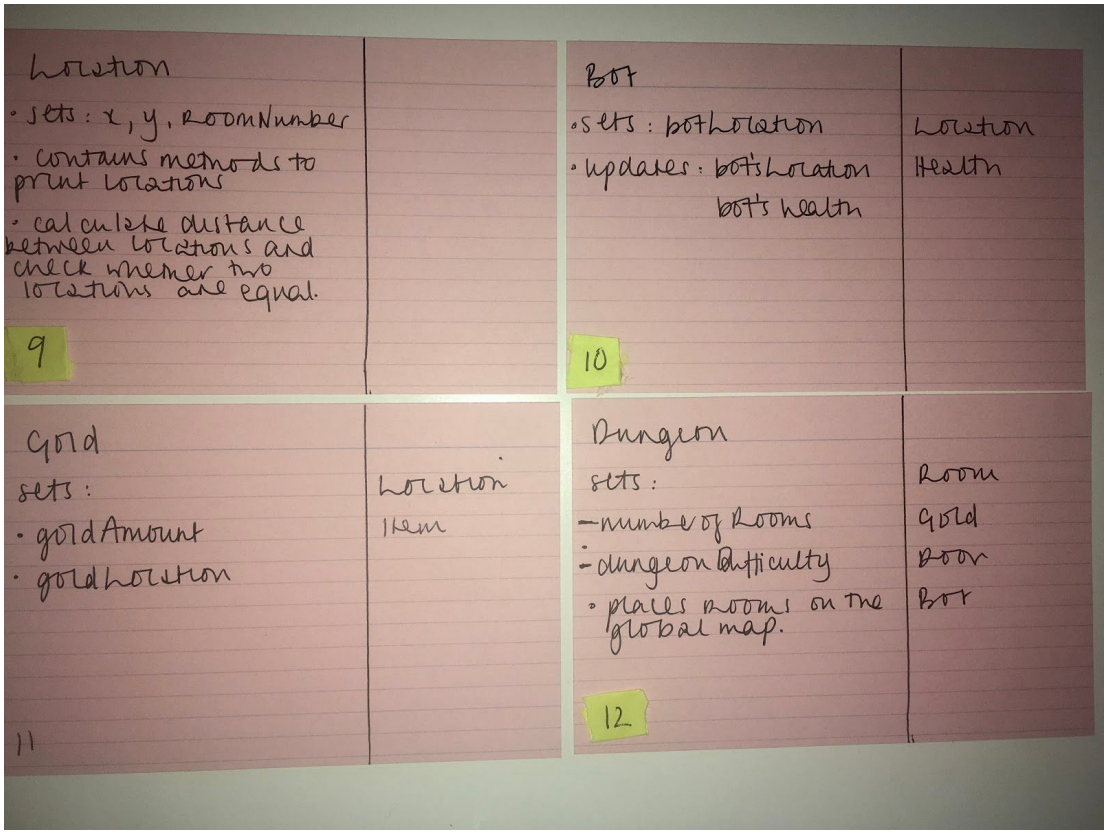
The team did not implement any further features this sprint. Therefore, the only product were the CRC cards created to aid them as they refactored their code.

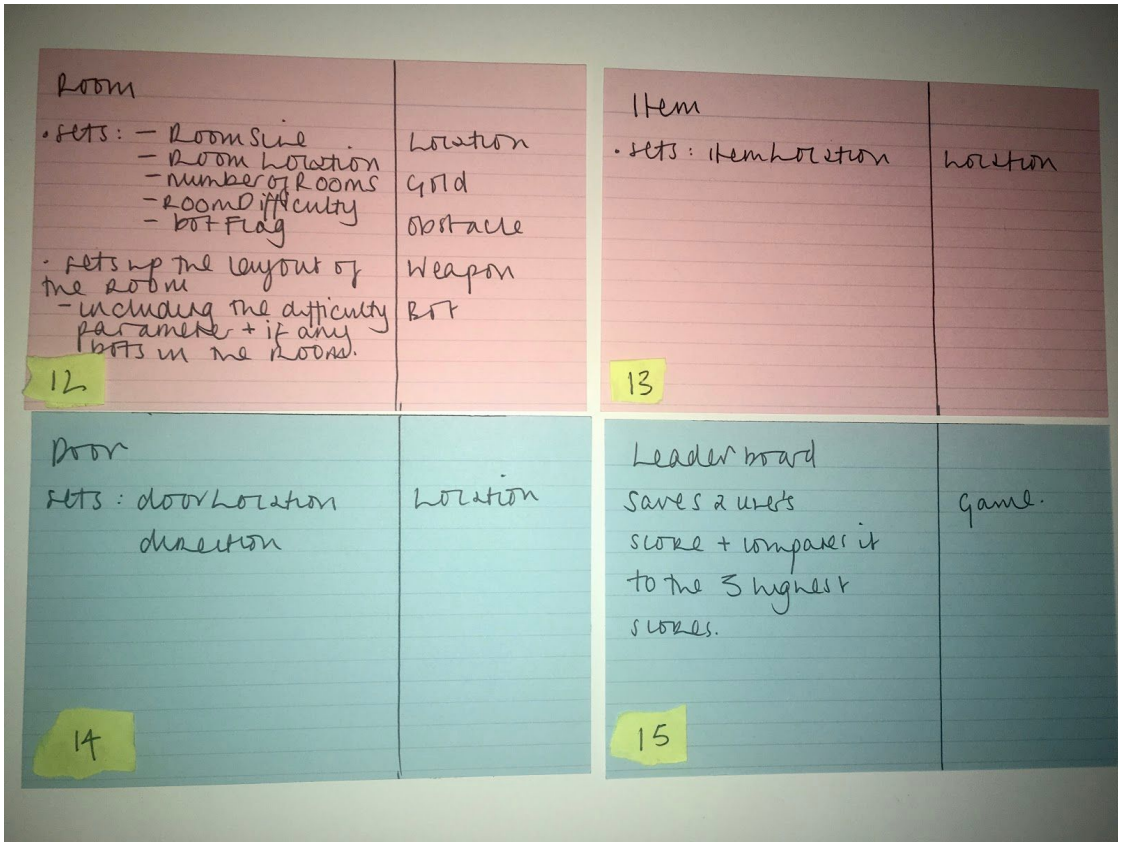
CRC Cards 5 11/12

These CRC cards were created to aid the team when refactoring their code into a MVC form.




Game Pt1
-Sets location of :
- player - dungeon walls - dungeon doors - gold - bot - camera offsets - initial offsets.
-Visualize :
dungeon - player location in room - bots in room - item locations in room
5
Location
Gold
Dungeon
Room
Bot
Item
Door
Leaderboard.
Game Pt2
-Updates player's :
- level
-Updates leaderboard.
6
Game Pt3
Allow user to :
input direction - check collision with : walls, doors, gold, bots - allow user to pick up objects - to go to the next level - restart the game after death - win the game - view the leaderboard - start a new game - exit the game.
7
Player
sets : playerLocation; gold; scores; steps; dieflag; winflag;
updates : players location; player's gold; players scores; players steps; if a player dies; if a player wins.
Allows user to : input N,E,S, W; check collision; to pick up objects
8


Location
- sets : x, y, RoomNumber
- contains methods to print locations
- calculate distance between locations and check whether two locations are equal.
9
Bot
- sets : botLocation
- updates : bot's location
bot's health
Location
Health
10
Gold
sets :
- goldAmount
- goldLocation
11
Location
Item
Dungeon
sets :
- number of Rooms
- dungeon difficulty
- places rooms on the global map.
12
Room
Gold
Door
Bot


Room
• sets: – RoomSize
– Room Location
– number of Rooms
– Room difficulty
– BOT Flag
• sets up the layout of the ROOM
– including the difficulty parameter + if any BOTS in the ROOMS.
Location
Gold
Obstacle
Weapon
BOT
12
Item
• sets: ItemLocation
location
13
Door
sets: doorLocation
direction
Location
14
Leaderboard
saves 2 users score + compares it to the 3 highest scores.
game.
15