**APPENDIX A: SPRINT CYCLE 6**

|  |  |
| --- | --- |
| 1. **Summary data** | |
| Team number | 7 |
| Sprint technical lead(s) | Sarah, Ethan |
| Sprint start date | 23/5/21 |
| Sprint end date | 30/5/21 |

|  |  |
| --- | --- |
| 1. **Individual key contributions** | |
| **Team member** | **Key contribution(s)** |
| Ethan | Programmer - game mechanics and logic |
| Sarah, Ethan | Programmer - GUI |
| Umar | Sprint documentation |
| Will | JavaDoc testing |
| Sam | Management and JavaDoc testing |
| Adam | Management and JavaDoc testing |

|  |
| --- |
| 1. **User stories / task cards** |
| * The object of the game is to solve, by means of elimination and deduction, the problem of the murder of Dr Phlox, the owner of the house * The winner is the first person to identify in one accusation the murderer, the weapon used and the room in which the crime was committed * The electronic version should be for desktop machines, and ideally should be playable on both Mac and PCs. If this is difficult, then PC development should be preferred. * Secret passages enable players to move from certain rooms to those indicated on the board. This can be done on a player’s turn without throwing the dice, and constitutes the move. |

|  |
| --- |
| 1. **Requirements analysis** |
| F8 - The first person to correctly assume the weapon, room and character shall win the game  F12 - Upon winning the game, the gameplay shall be terminated and the system shall give appropriate visual and/or auditory feedback to convey that a player has won  F13 - A single game should be able to be played over multiple devices  F14 - Players shall be able to choose the in game character they wish to play before the game starts  F30 - If not all player pieces are being played by a player or ai they should be moved into a room before the game starts.  NF1 - The software shall be written in python to ensure maximum portability across platforms  NF4 - The software design shall accommodate future updates or maintenance through well designed code  NF2 - The software should have a mean time between faults of no less than 3 complete run throughs  NF3 - The software shall be secure and unable for unauthorised persons to access the game while in play  NF5 - The user interface shall be intuitive for persons of ages 8+  NF11 - There should be a menu screen offering the players options on how they would like to start the game  NF12 - There should be an option for one of the players to leave the game  NF13 - On the event of a player leaving the game, the player who has left the game should be replaced by an AI character |

|  |
| --- |
| Design |
| This aspect of the game does not require much design in terms of class diagrams, sequence diagrams etc. It mostly consists of adding the final touches to the game and working on any aspects that still need work. The following images show some examples of the plan for the menu screen. It shows the idea for how the players will select their characters for the game. Each character has an option to be played by a human or AI. By clicking on the image of the character, the character will be disabled. We found this design to be very user friendly and therefore hit the non-functional and domain requirements for user accessibility.      This sprint alspo includes the implementation of the user feedback when winning the game. After correctly accusing, that player will win the game as shown in the next diagrams. |

|  |
| --- |
| 1. **Test plan and evidence of testing** |
| *import json*  *import unittest*  *import random*  *from pathlib import Path*  *from src.room import Room*  *from src.player import Player*  *from src.weapon import Weapon*  *from src.board import Board*  *from src.solution import Solution*  *class MyTestCase(unittest.TestCase):*  *def get\_json\_data(self):*  *data = []*  *config\_dir = str(Path.home()) + "/Clue"*  *with open(config\_dir + '/clue.json', encoding='UTF-8') as file:*  *data = json.loads(file.read())*    *return data*  *#Old tests, broken due to not using generate anymore*    *def test\_set\_and\_get\_solution(self):*  *board = Board()*  *room = random.choice(list(board.rooms.items()))*  *player\_card = random.choice(list(board.player\_cards.items()))*  *weapon = random.choice(list(board.weapons.items()))*  *room = {room[0]: room[1]}*  *player\_card = {player\_card[0]: player\_card[1]}*  *weapon = {weapon[0]: weapon[1]}*  *solution = Solution(room, player\_card, weapon)*  *r, p, w = solution.get\_solution()*  *self.assertEqual(r, room)*  *self.assertEqual(p, player\_card)*  *self.assertEqual(w, weapon)*  *def test\_check\_solution(self):*  *board = Board()*  *room = random.choice(list(board.rooms.items()))*  *player\_card = random.choice(list(board.player\_cards.items()))*  *weapon = random.choice(list(board.weapons.items()))*  *room = {room[0]: room[1]}*  *player\_card = {player\_card[0]: player\_card[1]}*  *weapon = {weapon[0]: weapon[1]}*  *solution = Solution(room, player\_card, weapon)*  *b = solution.check\_solution(list(room.values())[0], list(player\_card.values())[0], list(weapon.values())[0])*  *self.assertEqual(b, True)*  *b = solution.check\_solution('why', list(player\_card.values())[0], list(weapon.values())[0])*  *self.assertEqual(b, False)*  *if \_\_name\_\_ == '\_\_main\_\_':*  *unittest.main()* |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **System Testing** | | | | | | | |
| **ID** | **Req** | **Description** | **Inputs** | **Expected** | **Actual** | **Pass/Fail** | **Action** |
| 1 | NF11 | Test that the menu screen appears on game start |  | The menu GUI appears on the screen on game start | The menu screen GUI is not implemented | Fail | implement a menu screen GUI that appears on game start |
| 2 | NF11 | Test that The players can choose their characters on the menu screen |  | The players can choose what characters they would like to play on the menu screen | All 6 players are always in play | Fail | Give the users an option to choose specific players to play |
| 3 | NF11 | Test that is an option on the menu screen to select how many characters will play |  | There is an option to select how many players want to play | All 6 players are always in play | Fail | Add an option for less players to be active, move inactive players into a room on game start |
| 4 | NF11 | Test that the once the play button is pressed, the game board is loaded |  | Once players are selected, the game is loaded | There is no menu screen GUI implemented | Fail | Implement the menu GUI and a menu screen |
| 5 | F8 | Test that once a successful accusation is made, that player wins the game |  | Once a player makes a successful accusation, they win the game | Once the successful accusation is made, that player wins the game | Pass | n/a |
| 6 | F12 | Test that upon winning the game, the gameplay is terminated |  | The gameplay is terminated when someone wins or all players lose | Once a player has won the game, gameplay is terminated | Pass | n/a |
| 7 | F12 | Test that upon winning the game, an appropriate screen or notification appears |  | Once someone wins, there is an appropriate message showing as such | No notification appears upon winning or losing the game | Fail | add some user feedback when the game is won |
| 8 | F13 | Test that a single game can be played over multiple devices |  | A single game can be played with multiple devices | The game must be played over 1 windows PC | Fail | If time allows it, run game on a server so that multiple devices can access it |
| 9 | F30 | Test that if not all players are being played, the inactive characters are moved into a room before the game starts |  | if not all players are being played, the inactive characters are moved into a room before the game starts | All 6 players are always in play | Fail | Add capability for the users to choose how many players they want to play |
| 10 | NF12 | Test that there is an option for a player to leave the game at any point |  | there is an option for a player to leave the game at any point | there is no option for a player to leave the game at any point | Fail | Add a button for a player to leave the game on their turn |
| 11 | NF13 | Test that if a player leaves the game, they are replaced by an AI |  | if a player leaves the game, they are replaced by an AI | AN AI does not replace any players that have left | Fail | Add and AI and then functionality so that the AI replaces any players that leave |
| 12 | NF2 | Test that the full game should have a mean time between faults of no less than 3 complete runs |  | the full game should have a mean time between faults of no less than 3 complete runs | the full game should have a mean time between faults of no less than 3 complete runs | Pass | n/a |
| 13 | NF1 | Test that the entire game is programmed in python |  | The entire program is programmed in python | The entire program is programmed in python | Pass | n/a |
| 14 | NF4 | test that the game is coded so as to accommodate for future updates and maintenance |  | The game is coded as to accommodate future updates and maintenance | The game is coded as to accommodate future updates and maintenance | Pass | n/a |
| 15 |  | Test that the correct amount of players are loaded into the game from the menu screen selection |  | The correct number of players are loaded into the game from the menu screen | The menu screen has not been implemented and all 6 players are always in play | Fail | Add a menu GUI to give the users an option to choose specific players to play |
| 16 |  | Test that the correct players are loaded into the game from the menu screen selection |  | The correct specific players are loaded into the game | All players are loaded into the game | Fail | Add a menu GUI to give the users an option to choose specific players to play |

|  |
| --- |
| 1. **Summary of sprint** |
| *You should consider and discuss:*  *cli - command line interface*   * *Did you achieve your objectives for this sprint?*   + *We completed the objectives regarding the end game, what happens then a player wins or loses the game. The main game functionality is therefore complete, allowing for a fully working run through of the game.*   + *not all objectives, solution was created and was implemented quickly, but as our main GUI programmer was absent due to personal issues. This meant that we were unable to implement the menu screen GUI and other tasks linked to it* * *Is there a working prototype?*   + Yes for the cli, the gui has not been implemented to the newest version of the game logic. The full game will therefore be run using the command line interface (cli) * *What went well, and what did not go well? If things didn't go well, what have you learned and what will you do differently for the next sprint?*   + Solution works perfectly. The game can be played in its entirety without issue. The solution class was easy to implement, allowing us to focus on streamlining the program and further removing bugs.   + Due to the lack of time, and other factors, we could not implement GUI in a reasonable amount of time.   + This is the last sprint, however, we learnt to have contingencies in case any risks occur (seen in risk assessment) we also learnt to have more of a buffer time, as this sprint ended exactly on the deadline, meaning we didn't have a lot of time for additional functionality or graphics. |