**APPENDIX A: SPRINT CYCLE 1 - board**

|  |  |
| --- | --- |
| 1. **Summary data** | |
| Team number | 7 |
| Sprint technical lead(s) | Ethan, Sarah |
| Sprint start date | 24/3/21 |
| Sprint end date | 07/4/21 |

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

|  |
| --- |
| 1. **User stories / task cards** |
| * The game is played by moving around a playing board that represents the ground floor of Archers Avenue |

|  |
| --- |
| 1. **Requirements analysis** |
| F2 - There must be a visual representation of the board, using identifiers (such as “bathroom”) and a grid format, allowing users to be able to see the entire board. This should be completed above any other requirement so that other elements are considered with the board in mind.  F10 - The board size must be sufficient to contain all 9 rooms described in the game  F31 - The board shall be graphically customisable to enhance the user’s experience  F32 - the customisable map must not violate any structural rules of the map  F20 - The board graphics shall contain an “X” - the murder envelope shall be located on this and only this area of the board  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  NF10 - The board should have an adequate number of squares to mean that it does not take long for players to get between rooms but also does not give the player the choice of multiple rooms from one dice roll  NF6 - Images can be implemented into the board to increase the visual aspect of the game. This is not necessary for the game to run but would assist the game's visuals.  NF9 - The board shall have 9 rooms  NF8 - The software will have to read data from an external file to start the game  D1 - The board should be aesthetically pleasing and accurately represent the house of Dr. Phlox  D2 - The board should not obviously offend any cultural groups |

|  |
| --- |
| 1. **Design** |
| A 2d board composed of individual ‘tiles’ that the player will be able to traverse upon. Each Tile shall be an object of the Tile class, containing the following information: the x and y coordinate of the corresponding tile, the room that the tile is located in, whether the tile is traversable or not. The tiles shall be stored in a 2d array which makes up the entire board.  The graphics for the board shall be overlaid onto the tile structure. The tiles will line up with the grid design of the board.  The following image shows the design of the board, provided by the user. This will be the design used in the final version of the game:    The following image shows a rough design of the 2d array that will make up the board. The tiles will have different values depending on the content of the tile in the game. Each symbol in the image represents a tile in the game and corresponds to a tile in the graphics:    Another key component of this sprint was the design of the GUI. The following images show some examples of the designs for the different tiles. These designs were created in photoshop. These images were 25x25 pixels to fit the size of a tile. Each Image has a different permutation to account for the different edges.    The next image shows a UML design of the board class, including its methods and attributes:    More details on the design and specification can be found in the sequence diagrams and the use case diagrams    This image shows the Sequence diagram when the game is started |

|  |
| --- |
| 1. **Test plan and evidence of testing** |
| Junit Tests: Visit the main program code for more information as the testing code links to this. Do this for all future sprints also.  import unittest  import json  from src.board import Board  from pathlib import Path  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  def test\_parse\_map\_data(self):  board = Board()  result, data = board.setup\_board()  self.assertEqual(result, True)  #def test\_place\_weapons\_in\_rooms(self):  # board = Board()  # result = board.place\_weapons\_in\_rooms(w,r,s,t)  def test\_get\_surrounding(self):  y = 10  x = 10  data = self.get\_json\_data()  board = Board()  tile\_map = data["map"]["tiles"]  result = board.get\_surrounding(x, y, tile\_map) != False  self.assertEqual(result, True)  def test\_get\_surrounding\_negative(self):  y = -2  x = -3  data = self.get\_json\_data()  board = Board()  tile\_map = data["map"]["tiles"]  result = board.get\_surrounding(x, y, tile\_map) == False  self.assertEqual(result, False)  #def test\_place\_weapons\_in\_rooms(self):  # board = Board()  # result = board.place\_weapons\_in\_rooms(w,r,s,t)  def test\_get\_surrounding(self):  y = 10  x = 10  data = self.get\_json\_data()  board = Board()  tile\_map = data["map"]["tiles"]  result = board.get\_surrounding(x, y, tile\_map) != False  self.assertEqual(result, True)  def test\_generate\_objects\_from\_tiles(self):  data = self.get\_json\_data()  board = Board()  r1, r2, r3, r4, r5 = board.generate\_objects\_from\_tiles(data)  result = r1 != False  self.assertEqual(result, True)  def test\_correct\_count\_object\_ref(self):  data = self.get\_json\_data()  board = Board()  result = board.correct\_count\_object\_ref(data)  self.assertEqual(result, True)  def test\_check\_valid\_doors(self):  data = self.get\_json\_data()  board = Board()  result = board.check\_valid\_doors(data)  self.assertEqual(result, True)  def test\_find\_instance(self):  y = 10  x = 10  data = self.get\_json\_data()  board = Board()  tile\_map = data["map"]["tiles"]  result1 = board.find\_instance(tile\_map[y][x], tile\_map, True)  self.assertEqual(result1 != False, True)  self.assertEqual(len(result1), 2)  result2 = board.find\_instance(tile\_map[y][x], tile\_map, False)  self.assertEqual(result2 != False, True)  #def test\_seperate\_board\_and\_players(self):  # data = self.get\_json\_data()  # board = Board()  def test\_find\_instance(self):  y = 10  x = 10  data = self.get\_json\_data()  board = Board()  tile\_map = data["map"]["tiles"]  result1 = board.find\_instance(tile\_map[y][x], tile\_map, True)  self.assertEqual(result1 != False, True)  self.assertEqual(len(result1), 2)  result2 = board.find\_instance(tile\_map[y][x], tile\_map, False)  self.assertEqual(result2 != False, True)  #def test\_seperate\_board\_and\_players(self):  # data = self.get\_json\_data()  # board = Board()  if \_\_name\_\_ == '\_\_main\_\_':  unittest.main() |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **System Testing** | | | | | | | |
| **ID** | **Req** | **Description** | **Inputs** | **Expected** | **Actual** | **Pass/Fail** | **Action** |
| 1 | F2 | Test to determine if the system consists of a 2d board that accurately represents the house of Dr. Phlox. | n/a - just compile | The board graphic overlay will accurately represent the house of Dr.Phlox and will accurately fit the grid of tiles | The board consists of a 2d array that accurately represents the house of Dr. Phlox | Pass | n/a |
| 2 | F2 | Test to see that the board has identifiers to allow the user to understand the layout of the board | visual inspection | The rooms have identifiers to help the users understand the layout of the board | The board does not have identifiers on the room GUI | Fail | Need to add image labels to the rooms to help user experience |
| 3 | F2 | test to see if the board is in a grid format | visual inspection | The board is presented in an obvious grid format | The board is presented in an obvious grid format | Pass | n/a |
| 4 | F10 | Determine that the visual scale of the rooms and board is comparable to the floor design of Dr.Phlox. | visual inspection | The board is large enough to fit all 9 rooms prescribed in the system spec, all to appropriate scales | The visual scale of the board is comparable to the floor design of Dr.Phlox | Pass | n/a |
| 5 | F31 | Test if the board design is customisable in some way | change text file containing board layout | That the user can change the layout of the board as they wish | That the user can change the layout of the board by editing the text file | Pass | n/a |
| 6 | F32 | Test to see if the customised map does not violate any structural rules of the game | change the text file data to invalid layouts (e.g. having doors on the border of the map | An error appears when the user attempts to edit the map to an invalid structural layout | An error appears when the user attempts to edit the map to an invalid structural layout | Pass | n/a |
| 7 | F20 | Test to see if the board graphics contains an “X” | visual inspection | The board graphics contain an “X” | The board graphics contain an “X” | Fail | create a graphical tile to represent the X |
| 8 | NF1 | Ensure that the code for the board was written in python | visual inspection | The code for the board was written in python | The code was programmed in python - in visual studio code | Pass | n/a |
| 9 | D1 | Test if the graphics for the board are aesthetically pleasing for the players and accurately represent the game | visual inspection | The board graphics are aesthetically pleasing and accurately represent the game | The board graphics are aesthetically pleasing and accurately represent the game | pass | n/a |
| 10 | D2 | Test if the graphics for the board are suitable and that it does not unintentionally provoke or insult any cultural groups | visual inspection | The graphics for the board do not offend and cultural groups | The board graphics do not obviously offend any cultural groups | pass | n/a |
| 11 | F32 | Check that there are a correct amount of each type of tile on the board | compare output to the specified amount of each tiles, specified on the txt file | That there are correct amount of each type of tile located on the map | That there are correct amount of each type of tile located on the map | Pass | n/a |
| 12 | NF8 | Test that the board reads from the text file and updates according to that layout | Compare the GUI output to the txt file of the board | That the board shall display the correct layout, given by the text file | That the board shall display the correct layout, given by the text file | Pass | n/a |
| 13 | NF9 | Test to see that all the required rooms are included in any given customised map | visual inspection | Checks to see that all the required rooms are included in the customised map | All the required rooms are included in the customised map | Pass | n/a |
| 14 | F32 | Test to see that the custom map cannot be smaller than 25x25 | Edit the text file to contain a board less than 25x25 | The map should be no smaller than 25x25 tiles | An error occurs when the user attempts to create a board of less than 25x25 | Pass | n/a |
| 15 | F32 | Test that no doors are located on the edge of the custom map | Edit text file to include doors on the edge of the map | That an error occurs when the user attempts to place a door on the edge of the map | That an error occurs when the user attempts to place a door on the edge of the map | Pass | n/a |
| 16 | F32 | Check that every door is located to only a single room in the map to reduce ambiguity | visual inspection | That every door is linked to only a single room in the board | That every door is linked to only a single room in the board | Pass | n/a |
| 17 | NF10 | Check that board has an adequate number of squares so that it does not take long for players to get between rooms but also does not give the player the choice of multiple rooms from one dice roll | visual inspection and runthrough of the map to ensure that traversing the board doesn't take too long | The board has an adequate number of squares so that it does not take long for players to get between rooms but also does not give the player the choice of multiple rooms from one dice roll | The board has an adequate number of squares so that it does not take long for players to get between rooms but also does not give the player the choice of multiple rooms from one dice roll | Pass | a/a |
| 18 | NF6 | Check that Images are implemented into the board to increase the visual aspect of the game. | visual inspection | Images are implemented into the board to increase the visual aspect of the game. | The images for the board appear on the GUI | Pass | n/a |
| 19 | NF9 | Check that the board has 9 rooms |  | The board has all 9 rooms | All 9 rooms are included on the preset map | Pass | n/a |
| 20 | NF7 | Test that the software implements an appropriate GUI for the user that represents that of a board game | visual inspection | The GUI for the game is appropriate and represents a board game to some degree | The GUI for the game is appropriate and represents a board game to some degree | Pass | n/a |

|  |
| --- |
| 1. **Summary of sprint** |
| What went well:   * We have written several tests within the board object to validate if the map file is correctly formatted * It was fairly simple to parse in the json data and turn it into a representation of a board from the way it was structured, was therefore good planning * the following image shows the finished prototype of the board   What went wrong:   * We didn’t think through enough how to actually store the representation of the board, so we ended up using json to store the data of the map as it is quite user friendly without having to implement a whole map editor to then store the object * The creating the json schema was new and finding the documentation for objects within an array and validating they are in the right format was slightly harder than thought * We didn’t think through the map parsing and saw just the basics of what was needed   Next sprint:   * More programming will happen due to more members setting up the environment on their computers * Need to split tasks up in a way where they can be concurrently programmed * Next sprint will include less requirements as this sprint contained a lot and took a while to finish, will be better if we take smaller steps to keep momentum |