**,SOFTWARE ENGINEERING G6046**

**APPENDIX A: SPRINT DOCUMENTATION TEMPLATE**

|  |  |
| --- | --- |
| 1. **Summary data** | |
| Team number | 11 |
| Sprint technical lead(s) | Anson Wong |
| Sprint start date | 26/02/2021 |
| Sprint end date | 05/03/2021 |

*The technical lead may vary from one sprint to the next. This is down to how you collectively organise your team.*

|  |  |
| --- | --- |
| 1. **Individual key contributions** | |
| **Team member** | **Key contribution(s)** |
| Anson Wong | Programmer, Planner |
| Danny Newsom | Programmer |
| Tomasz Czarnecki | Programmer |
| Abdullah Al-Hiyarat | Programmer |
| Mohammad Jallad |  |

\*almost

*This data should help you to agree your peer assessment at the end of the project. If there is a dispute over your peer assessment, the markers will refer to this section as evidence to support a final decision.*

|  |
| --- |
| 1. **User stories / task cards** |
| *Provide text descriptions of any user stories or task cards you have selected for this sprint. These should naturally emerge from the user requirements document and discussion on Canvas. If you produce task cards, they should show the relative priority of the task for this sprint.*  User Story:   * The game is for 2-6 players * There are 6 cards representing the persons in the house. There are 6 cards representing the weapons. There are 9 cards representing the rooms shown on the playing board. * The dealer shuffles the 18 remaining cards. The cards are dealt to players one at time clockwise around the table, including the dealer. * The classic game board is as shown below. It can easily be found online also. Although this is the classic, Watson Games think that the electronic version offers the possibility of board customisation by the user. * A game player agent: An agent that can take the role of 1 or more of the players. This would allow for a limited number of human players to enjoy a richer gaming experience. However, it also provides the possibility for fully autonomous play when all the players are provided by the program. This will enable you to investigate various strategies as to how to play the game to the best advantage. Such simulations could be performed at high speed. Such simulations also offer a means of testing the performance and correct operation of your game. * A means of uploading initial data: to get the simulation started you will need a means of initialising it with data on the board layout, the players and other data need to make the game function. As this data will be loaded on start-up from external files, this means that the game is easily customised, and Watson Games see this as a valuable selling point of the new electronic version. * A means of monitoring the performance of the simulation: an appropriate GUI to enable players to see and access all data that they would in the classic board game. * Players may move their pieces anywhere on the board on the squares according to the number thrown on the dice. Players must move in straight lines only i.e. forwards and cross wise, but never diagonally. * The dice used in the game must be fair with each dice have an equal probability of landing on one of its six sides.   Task Cards:   1. PlayerStats (Anson) 2. PlayerToken (Anson) 3. PlayerMovement (Anson) 4. Turn Controller (Abdullah) 5. Round Controller (Abdullah) 6. Card Classes (Tom) 7. Deal and store cards to Player Master (Tom) 8. Refactor Board Builder (Danny) |

|  |
| --- |
| 1. **Requirements analysis** |
| *For the user stories/task cards selected, set out what key functional, non-functional and domain requirements you have identified. Remember that functional and non-functional requirements can be further categorised as mandatory (“shall”) and desirable (“should”). You can use free text descriptions or tabular formats. Remember that domain requirements cannot be acted upon directly. They require domain expertise to refine them into meaningful functional and non-functional requirements. All requirements should be SMART (Specific, Measurable, Achievable, Realistic and Time-Bounded). The requirements analysis does not need to be exhaustive, but should focus on things that are important for this sprint. They should also form a basis for testing.* |

Task Cards:

1. PlayerStats (Anson)
   1. Shall:
      1. Able to assign and store a list of Cards
      2. Set what player enum
2. PlayerToken (Anson)
   1. Shall
      1. Store which grid position it is
      2. Set what character and colour it is based on the enum
      3. Set the correct PlayerMasterController
3. PlayerMovement (Anson)
   1. Shall:
      1. Know which tile the player can move to
      2. Move the player token to that selected tile and update it
4. Turn Controller (Abdullah)
   1. Shall:
      1. Change turn to next player
      2. Remove player from turn order if made the wrong accusation
5. Round Controller (Abdullah)
   1. Shall:
      1. Allow player to initiate dice roll
      2. Allow players to make suggestions
      3. Allow players to make accusation
      4. End the turn
6. Card Classes (Tom)
   1. Shall:
      1. Have subclasses for each type (Weapon, Room, Character)
7. Deal and store cards to Player Master (Tom)
   1. Shall:
      1. Pass 6 lists with random playable cards to Player Master
8. Refactor Board Builder (Danny)
   1. Shall:
      1. Remove serialized fields
      2. Pass Char Enum to player token

|  |
| --- |
| 1. **Design** |
| *Remember that you only need to do enough design to support the objectives of the sprint. For teams working with OO implementation languages (likely most of you), this would include a class diagram. You may find it useful to develop simple Application Programming Interfaces (APIs) for key classes. This will focus your attention on what each class needs to make available for other classes to use. It also supports good documentation practice and helps coders work together.*    Board Builder UML    Player Movement UML    Class Diagram (partially updated) |

|  |
| --- |
| 1. **Test plan and evidence of testing** |
| *You should consider:*   * *Unit/component level testing – typically achieved using automated test procedures such as Junit in Java. This level of testing demonstrates that individual classes are working as you intend.* * *System level testing – typically a human lead and documented test process that shows the prototype working as a whole entity.*   *Testing should show that the requirements you set out are being delivered on. They provide a means of showing that we have delivered what the user stores and task cards set out. Remember to identify a useful set of boundary test conditions.*  *Evidence of testing should demonstrate that the prototype achieved has been tested according to the test plan. If there are deficiencies, then these should be documented, as they will need further work in a subsequent sprint.*  Test Scene:   * Movement Test scene   **\*Please Refer to Testing Document** |

|  |
| --- |
| 1. **Summary of sprint** |
| *You should consider and discuss:*   * *Did you achieve your objectives for this sprint?* * *Is there a working prototype?* * *What went well, and what did not go well? If things did not go well, what have you learned and what will you do differently for the next sprint?* * *Is there any feedback from the customer?* |

Achieved:

1. Refactor Board Builder (Danny)
2. PlayerStats (Anson)
   1. It has methods to store cards for the player’s deck
3. PlayerToken (Anson)
   1. It can assign what character it is
   2. Player Master controller can take control of the currect token
4. PlayerMovement (Anson)
   1. Built in to the player master controller and player token
   2. Can move to the tile selected from the cursor

Continue working:

1. Card Classes (Tom)
2. Turn Controller(Abdullah)

We have a scene where the Player can move to a certain tile depending on the dice roll, though we can play control one of the character

We did manage to complete whole of the task cards set, but we have under estimated the workload and complexity for some of the components

We have showed the board generation and movement to Mr Raffles and he seems to be very satisfied with the board layout retaining the essence of the original game