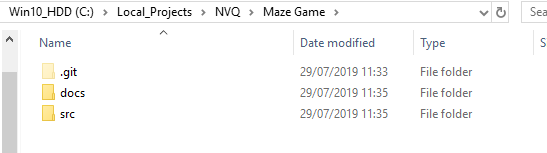
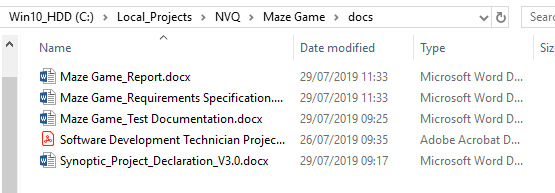
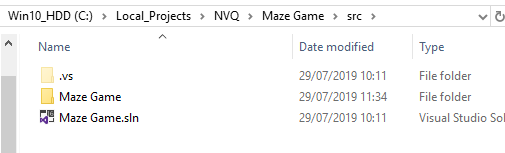
**Final Project – Olde Worlde Phunne Maze Game Report**

**Project Setup:**

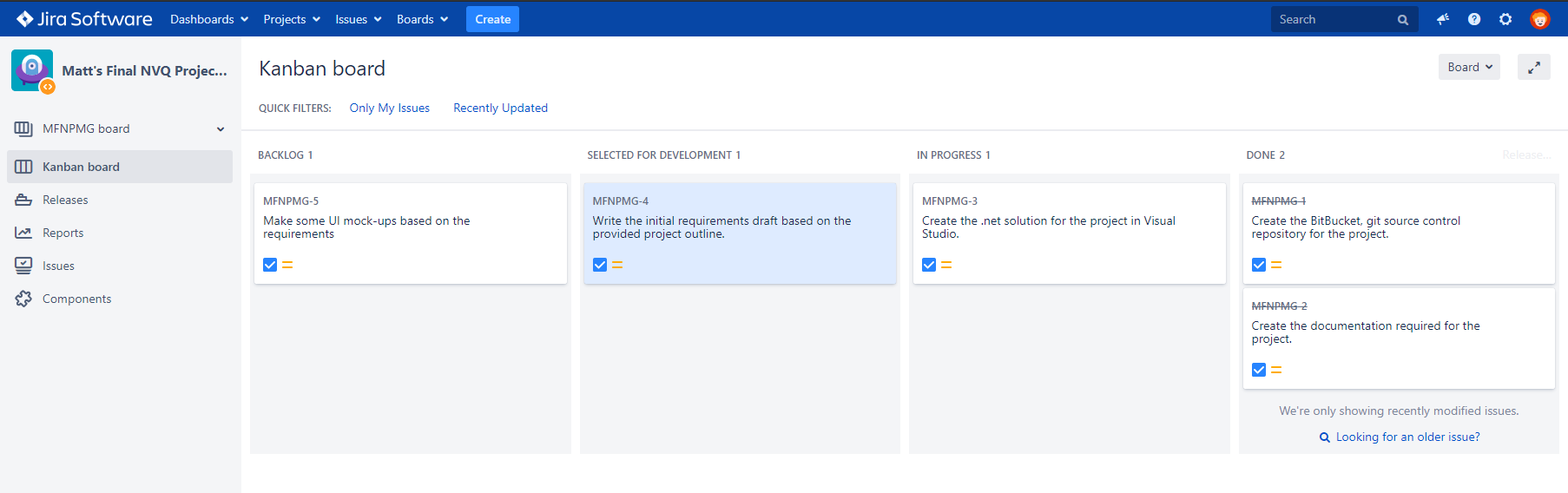
The first thing I did on day 1 was to read the project outline documentation. Following this I put together the word documents I needed, including the requirements specification and the test plan. I then created a git repository in my company’s source control server for this project, cloned this repository onto my local machine and moved my Visual Studio solution and documentation files into the project directory, committing these files in an initial .git commit and pushing them to the remote repository. The files for this project can be seen below:







Following creation of the project files, I then created a Kanban board on my Company’s JIRA server instance in which to register tasks and track their progress.



With the project file structure and JIRA task board set up, the first task to undertake was to put together a requirements specification for the system, based on the project outline document. The requirements specification I put together was based on the requirements specification form I helped develop for my team and is what we currently use for producing specifications for new systems.

Whilst writing the requirements I made the decision to write the game using Microsoft’s C# and .Net Core framework and use Visual Studio for the IDE. I made these decisions as am familiar with this language and framework combination and we use these technologies in my team. I also made the decision that the game would take the form of a command-line based text-adventure, because the week long development timeframe does not leave much space for

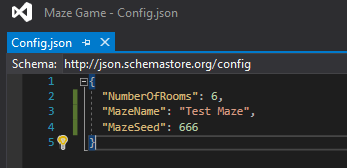
**Day 2:**

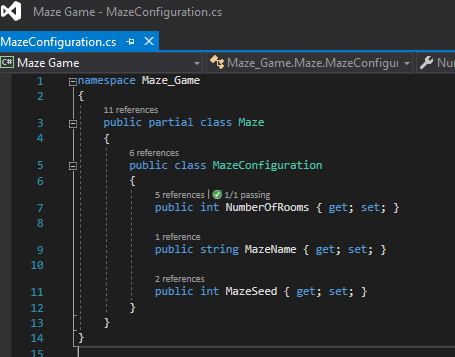
The first task I completed on day 2 was to write the entities and their relationships, as laid out in my requirements and the conceptual models. Following this I wrote the initial program flow structure, including the different stages of the overall program, from:

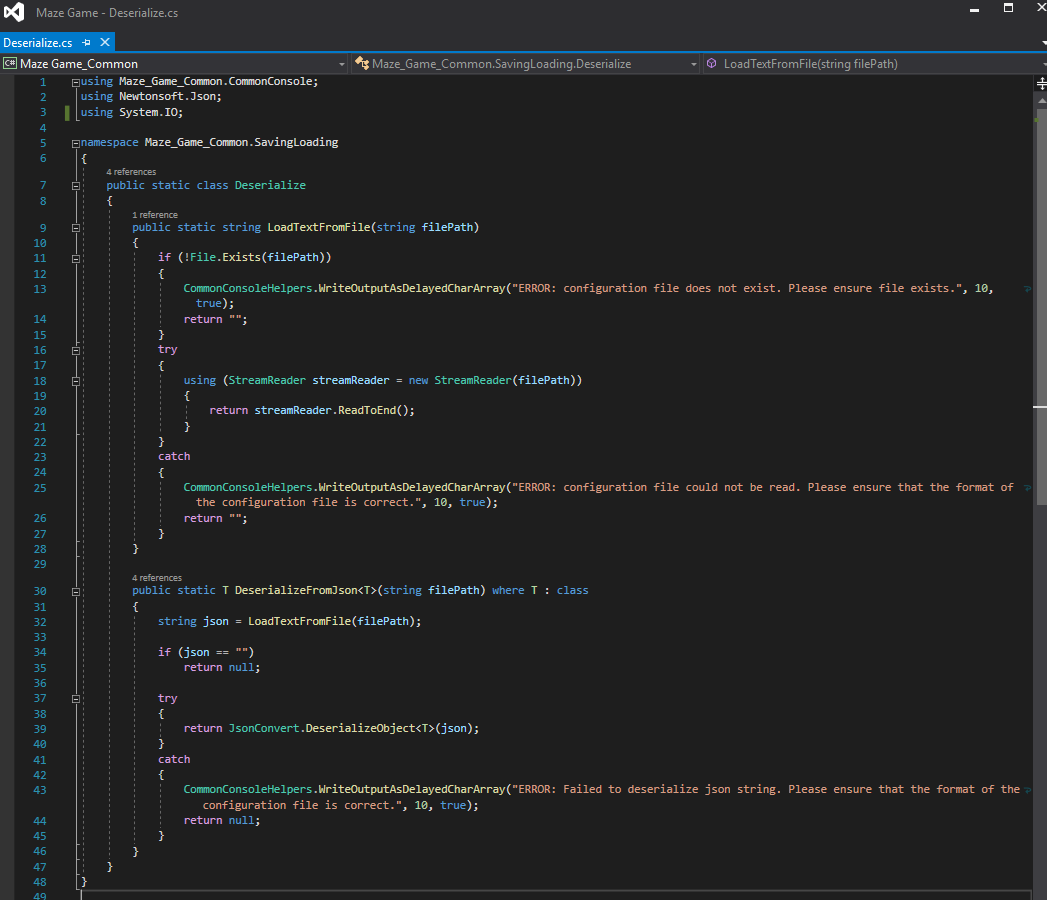
* Maze initialisation
* In-Game loop
* Resetting the in-game loop
* Presenting results.

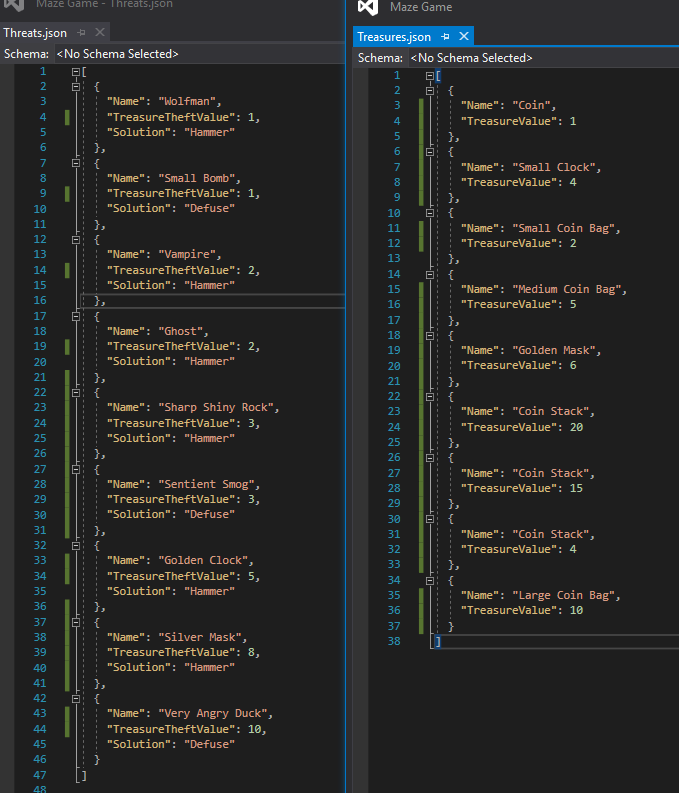
I also later added user name collection to the start of this sequence.

Following this I abstracted the common commands I would use for the game, including writing the text as a crawl, instead of all text being written to the screen at once as it commonly is in Console applications. As this is not a requirement for the project, it is something I should have perhaps added this to the requirements and returned to it at a later time. However, I believed it to be important to the feel of the game and user experience. I also believed I could complete the task quickly and it was related to the static Console Helper methods I was putting together, so I proceeded to add this functionality. I also added the screen shake effect and static, common methods for looping for user confirmation and command input. Added common project for abstracted, shared code. Added deserialization class and code and tested with a json config file. Ironed out issues and added error handling.





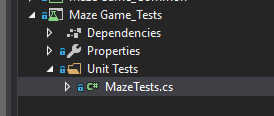


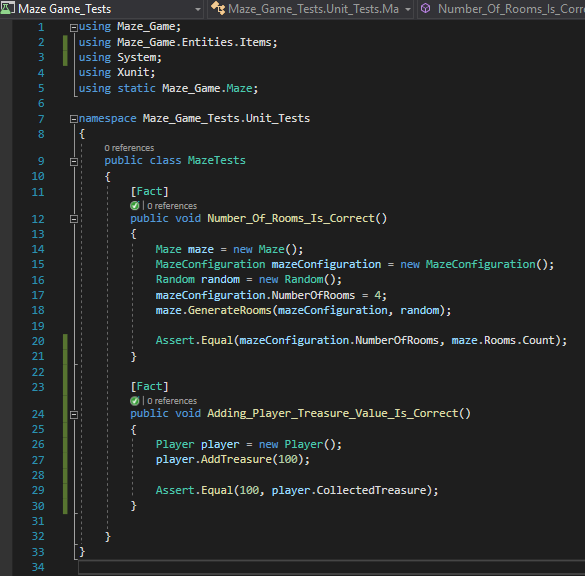


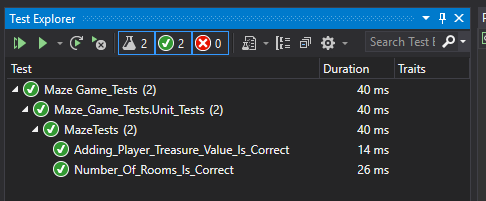
**Day 3:**

Continued writing code and bodied out most of the program’s flow with text and control statements.

Added Unit testing framework to the project. Had some errors, but fixed them. Added some unit tests to the project.







**Day 4:**

Fixed code issues and bugs. Published the project to make sure I could publish and share an executable. Share the executable with a colleague, but it failed. Investigated the failure and fixed. Added some of the missing requirements. Wrote the test plan.

Day 5:

Executed test plan. Finished this report. Made a User Guide video. Fixed some more bugs.

**Limitations, Final Thoughts and Future Improvements:**

The key limitation of the maze Game design is that it is not a visual game with graphics. This means that it is harder to convey some types of information to the player, such as where they are or what is in the room. This limitation lead me to add the feature that give the player a hint as to whether or not they are getting closer to the end of the maze, or further away from it.

Another limitation of this project is the time constraint of having a 1 week development timeframe. With a longer timeframe, more playtesting could have been carried out, as well as more testing in general; the code could have been refactored and improved – specifically the code in the “Program.cs” file, which could have been separated out into separate more discrete files to make reading and understanding the code easier.

Given more time I would also have wished to make some elements of the UI more intuitive, including making some text clearer and the layout of text more consistent throughout the project. I would also have wished to investigate the possibility of adding to the game’s design with more enemy types and a more complex action system in which the player can perform more than just the two “hit” and “defuse” actions on enemy types.

The next steps in this project would be to better establish a more comprehensive series of automated Unit tests, to ensure that they cover more areas than they currently do. This would help with testing the project whilst adding new features.

* More Unit Tests.
* Fix the problems found from the test plan results.
* Stick more rigidly to requirements.
* Provide the user with better feedback (when are they getting closer to the end)
* Improved User Guide.

As I didn’t get around to producing the User Guide until the final day of the project, and I had some issues installing OBS studio to capture video footage of the Maze Game program in-action, the user guide was a rushed production.