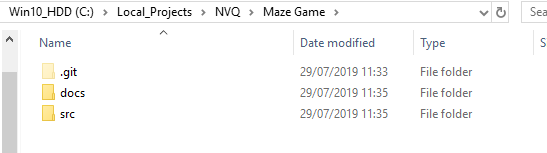
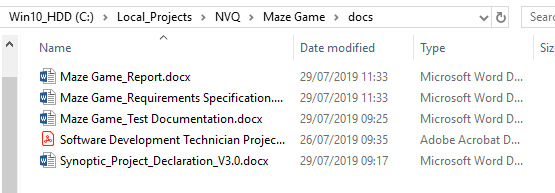
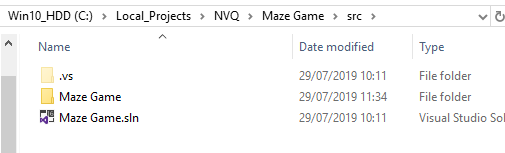
**Final Project – Olde Worlde Phunne Maze Game**

**Day 1:**

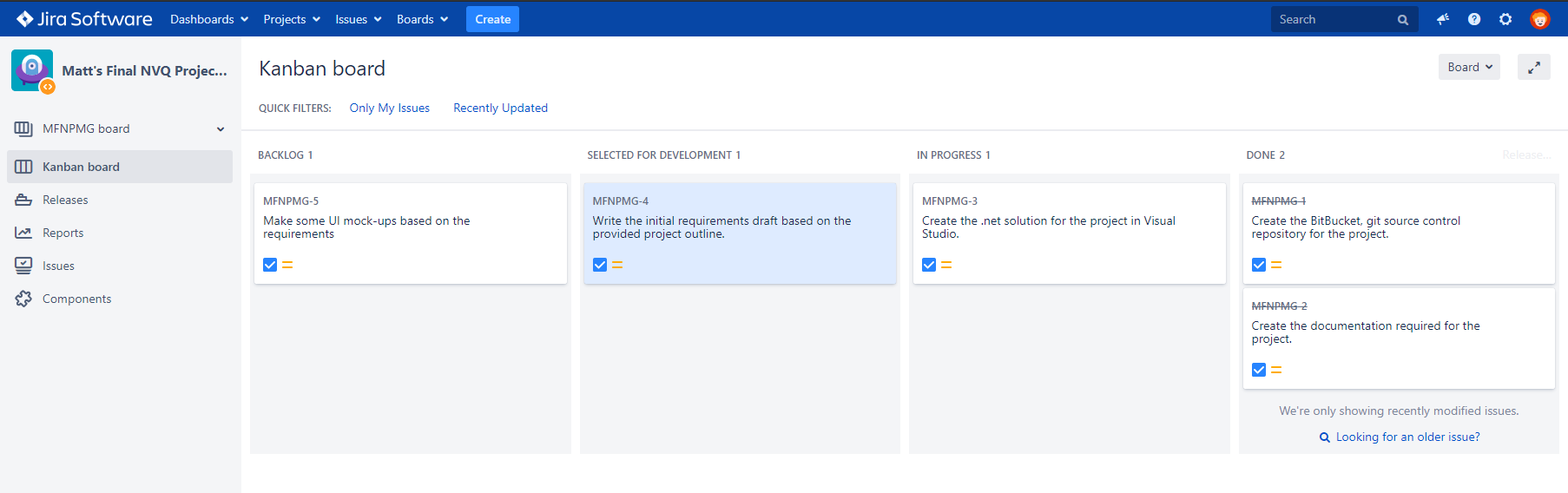
The first thing I did on day 1 was to read the project outline documentation. Following this I put together the word documents I needed, including the requirements specification and the test plan. I then created a git repository in my company’s source control server for this project, cloned this repository 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.

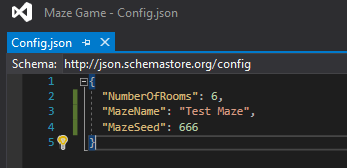


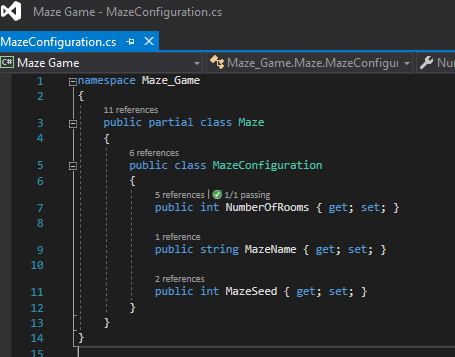
With the project file structure and JIRA task board set up, the first task to undertake was to put together a requirements specification for the system, based on the project outline document. The requirements specification I put together was based on the requirements specification form I helped develop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

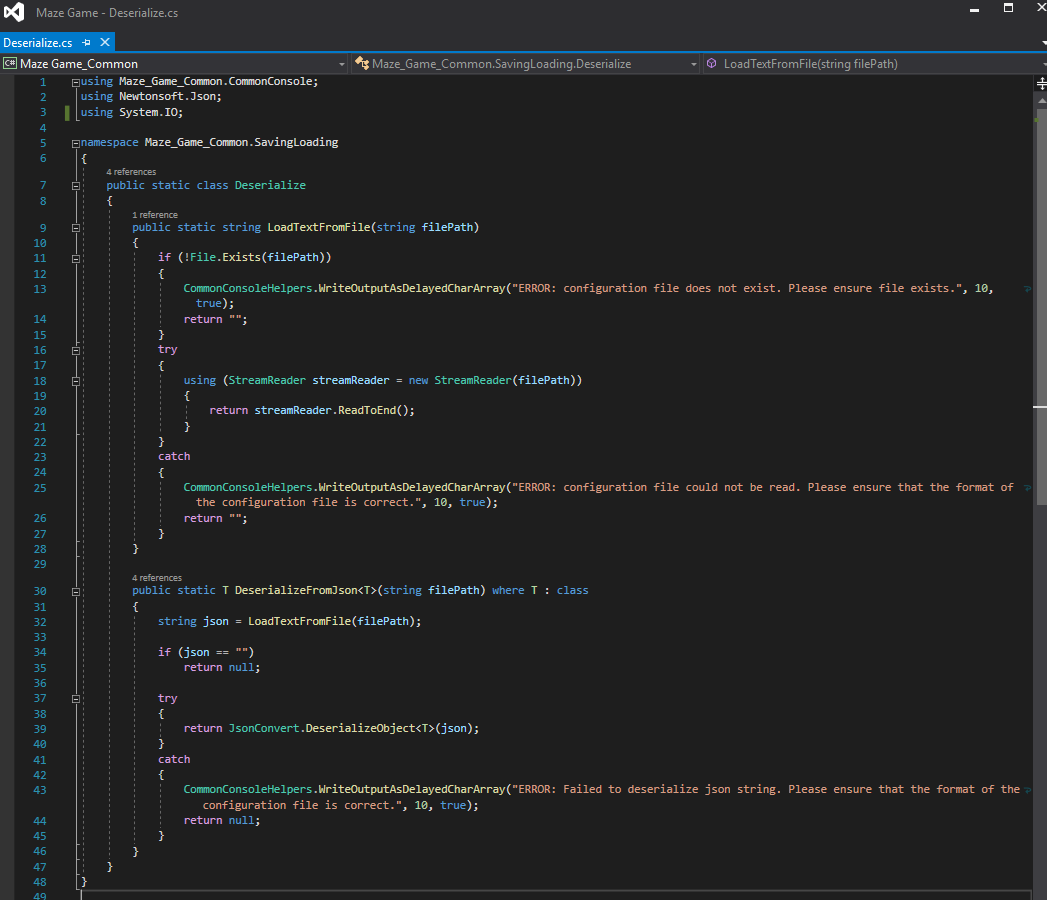
Whilst writing the requirements I made the decision to write the game using Microsoft’s C# and .Net Core framework and use Visual Studio for the IDE. I made these decisions as am familiar with this language and framework combination and we use these technologies in my team. I also made the decision that 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

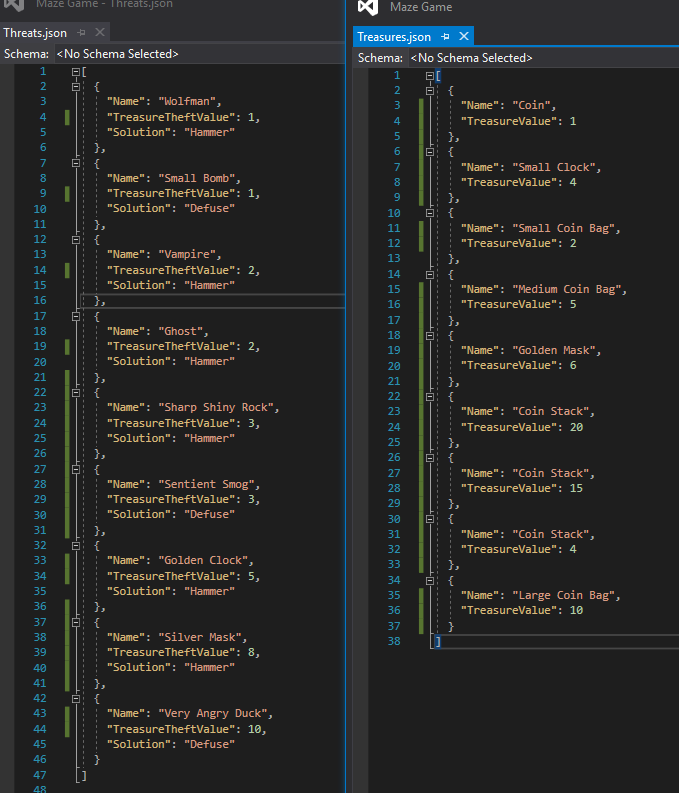
**Day 2:**

Wrote the entities with their relationships. Wrote the intial program flow structure. Abstracted the common commands I would use for the game, including writing the text as a crawl, shaking the screen and 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.





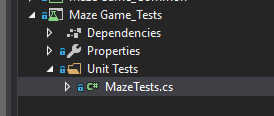


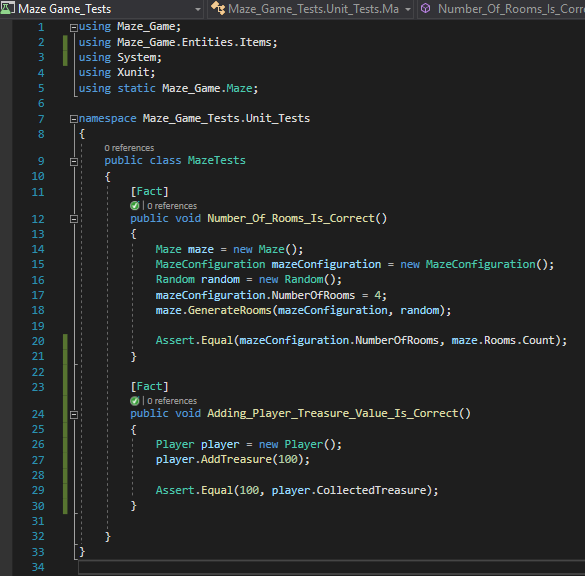


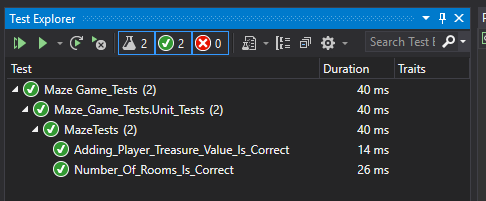
**Day 3:**

Continued writing code and

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. Fixed some more bugs.

**Limitations:**

* Not being a visual game means it is harder to convey information to the player about where they are or what is in the room.

**Final Thoughts and Future Improvements:**

* 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)