Testing Plan

Our testing approach was a black-box testing structure focusing on only the interface provided
Our testing approach for milestone 3 is to first understand and testing expected behaviour in the specification using black-box integration tests. After that, we will implement classes and functionality as we understand to eventually pass tests.
Then we will write unit tests for our implemented methods to verify their robustness and reliability.

Integration Testing:

A lot of our testing was integration testing due to the complex nature of the game. We required the player to interact with a lot of entities for the relevant behavior to trigger, per specifications.
For static entities, we checked that their behavior, apart from correct initial positioning, was correct by making the player entity interact with them. For walls, doors, portals and boulders, we made the player walk into them.
We also tested combat and enemy behavior via the same method, forcefully making the player interact with them to test the corresponding behavior.
Similarly, for collectibles and buildables, we created integration tests but only used the basic features of player movement and interaction to test the features so that we could better isolate where issues may lie.
We then did some more complex integration testing when testing the goals feature as this feature required all many features to work cohesively for the relevant goal to be triggered. This
was a good opportunity to not only test goals but also create through integration tests that combined multiple parts of our system written by different people.
We still did create very basic tests for goals, such as just checking if the correct output was given if the player walked onto an exit.
Due to the complex nature of the project, it was difficult to utilize white box testing as we needed outputs outside individual classes for the correct behavior to occur.

Unit Testing:

To unit test the Controller class, we tested that the game was initializing correctly and that the relevant entities were in the correct position.
We then tested very basic functions, such as moving the player and checking that the player had the correct final position by using the testing utility functions.
We also tested individual moving entities behavior as unit tests in the Controller class by

isolating them out by again using the testing utility functions provided.

System Testing:

We created system tests in the goals tests, creating games that ran start to finish, utilizing as many features of the game in one test to check that the entire system worked correctly and cohesively.

Usability Testing:

We utilized usability testing by linking our backend project with the frontend implementation
provided and tested if behavior occurred correctly as per the specifications given.