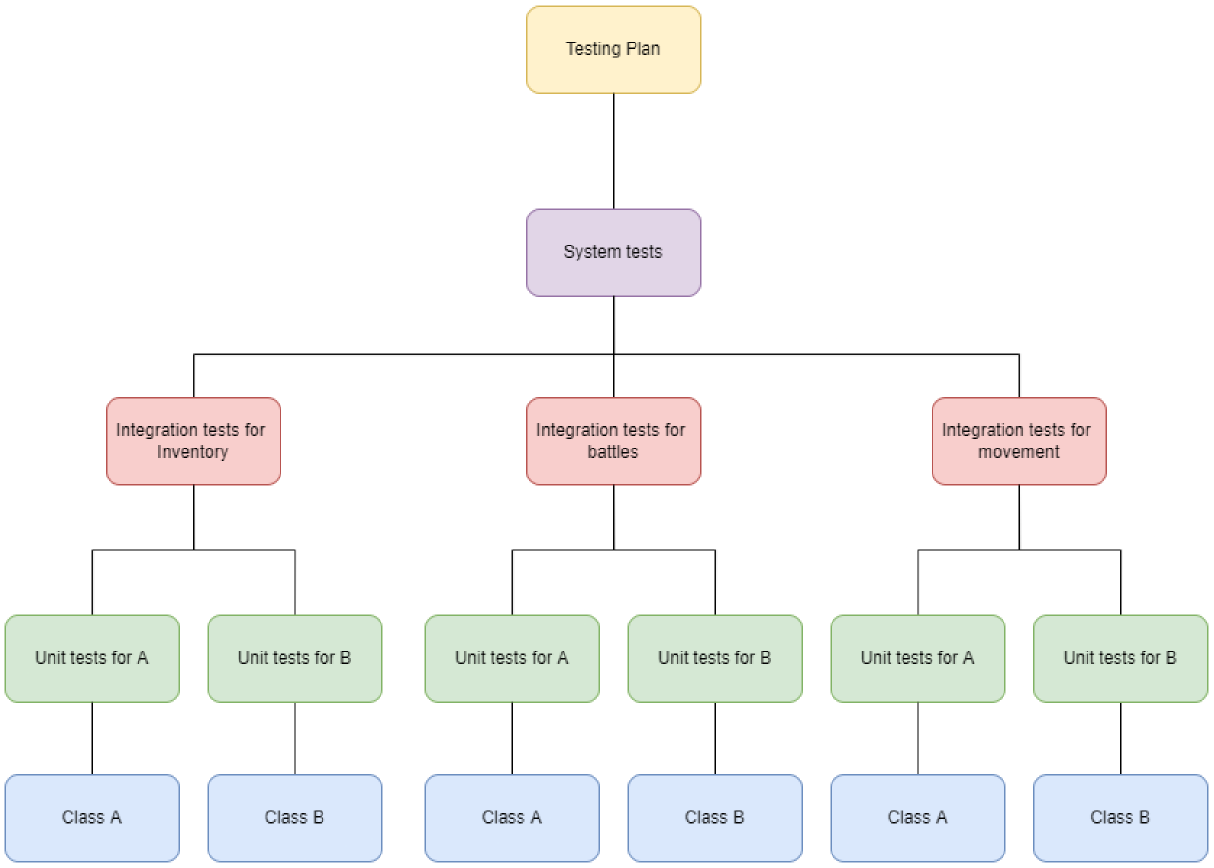
TESTING PLAN

Our teams testing plan features various testing strategies at different stages. We intend to create unit tests for each class to test specific areas of functionality and edge case and keep integration tests for major features of the project (battles, movement, goals, inventory) which cannot be tested using unit tests. We also have a few system-level tests to ensure exceptions are being thrown at the front-end correctly and to check the overall functionality of the game by playing out a few standard game scenarios. Before merging in every issue to the master branch we will make sure to conduct usability and acceptance tests throughout implementation.

Each class is made with a specific purpose so that is the functionality, which is tested in the unit tests, then the integration tests are used to test high level features of the game that may require multiple classes. An example of this is for the PlayerState classes, we have unit tests to check the transition of states as a result of different actions and then we have higher level integration tests such as testing actually drinking an invisibility potion at the player level and observing its affects on battles. The diagram below shows how a rough outline of how the testing plan works:



To complete this testing plan in conjunction with the other work we intend to write our ideas for unit tests as we make the classes (because we can be sure of the methods at that stage) and write ideas for integration tests as we complete an associated set of classes, which is why the integration level tasks are assigned to one person.