Name \_Luke Brinkerhoff\_\_\_\_\_\_ Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

## Brief introduction \_\_/3

I will be building a feature that revolves around two aspects, testing and adding functioning cheats to the game. The testing part will function as two parts, a program that will run through each individual unit test. This will allow are team to make sure that adding or removing a feature does not break other components of the game. The second core testing feature will consist of a semi intelligent bot that runs through the levels and features of game. Are game will have a god mode allowing for different mechanisms for bypassing in game restrictions I will also be championing this feature.

## Use case diagram with scenario \_\_/14

### Use Case Diagrams

A close up of a logo

Description generated with high confidence

**Scenario**

**Name:** Run Unit Test

**Summary:** The developer runs the unit test.

**Actors:** Developer.

**Preconditions:** Code has been updated.

**Basic sequence:**

**Step 1:** Upload new code.

**Step 2:** Compile new code.

**Step 3:** Run unit test.

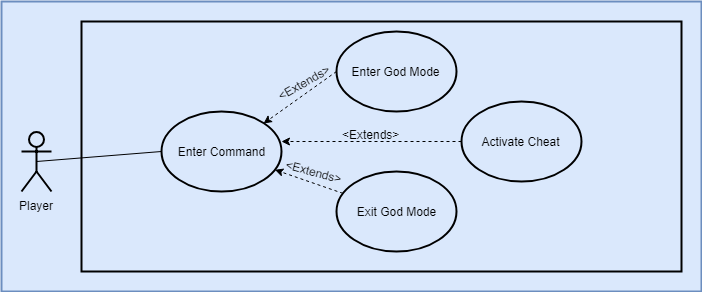
**Step 4:** Results from test are returned.

**Post conditions:** Results of the Test are displayed

**Priority:** 2\*

**ID:** C01

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.



### Scenario

**Name:** Activate Cheat

**Summary:** The player activates a cheat in game.

**Actors:** Player.

**Preconditions:** The game has be loaded and the player is in game.

**Basic sequence:**

**Step 1:** Enter the command to enter god mode.

**Step 2:** Enter cheat code.

**Step 3:** Activate cheat code.

**Exceptions:**

**Step 1:** Cheat code is entered when not in god mode: Do nothing.

**Step 2:** Exit command is given while not in god mode: Do nothing.

**Post conditions:** The player is in god mode with one or more cheats active.

**Priority:** 3\*

**ID:** C02

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

## Data Flow diagram(s) from Level 0 to process description for your feature \_\_/14

### Data Flow Diagrams

### Process Descriptions

6.1 Validate Command:

Check if command is for running bot or unit testing

6.2 Run Unit Test:

While not last test

Run test

Return result

End

6.3 Run Bot:

Start Bot

End Bot

6.4 Return Result:

Output Results from result storage.

7.1 Validate Command:

Check if command is an exit, enter or a cheat code

7.2 Enter/Exit God Mode

If exit code given exit god mode

If enter code given enter god mode

7.3 Activate Cheat:

Given cheat code apply cheat to game state

## Acceptance Tests \_\_\_\_\_\_\_\_9

**Test:** Invalid Input 1

**Input:** Invalid input

**Output:** Nothing runs

**Test:** Correct Input 1

**Input:** Bot Command

**Output:** Bot Runs results returned

**Test:** Correct Input 2

**Input:** Unit Test Command

**Output:** Unit Test runs results returned

**Test:** Invalid Input 2

**Input:** cheat given while not in god mode

**Output:** Nothings changes

**Test:** Invalid Input 1

**Input:** Exit command given while not in god mode

**Output:** Nothing changes

**Test:** Valid Input 1

**Input:** God mode command is given

**Output:** Enters god mode

**Test:** Valid Input 2

**Input:** Cheat code entered while in god mode

**Output:** Cheat activated

**Test:** Valid Input 3

**Input:** Exit code entered while in god mode

**Output:** Exit God Mode

## Timeline \_\_\_\_\_\_\_\_\_/10

[Figure out the tasks required to complete your feature]

Example:

### Work items

|  |  |  |
| --- | --- | --- |
| Task | Duration (Hours) | Predecessor Task(s) |
| 1. Environment Setup | 4 | - |
| 2. Bot Design | 5 | - |
| 3. Gather Unit Tests | 2 | - |
| 4. Program Bot | 12 | 1,2 |
| 5. Structure Unit Tests | 4 | 1, 3 |
| 6. Program Unit Test | 8 | 5 |
| 7. Program Cheats | 8 | 1 |
| 8. Bot Testing | 4 | 4 |
| 9. Unit Test Testing | 4 | 6 |
| 10. Cheat Testing | 4 | 7 |

### Pert diagram

A clock sitting in front of a keyboard

Description generated with high confidence

### Gantt timeline

