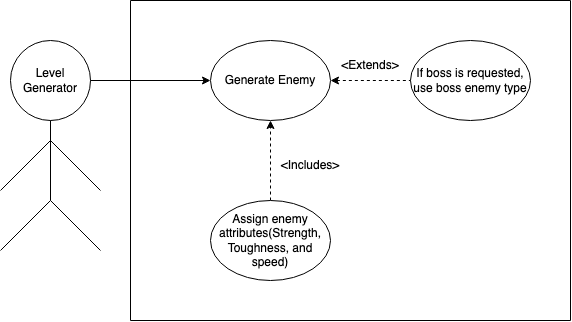
**1. Brief introduction \_\_/3**

My feature for our game, Rogue Realm, will be to design and create all of the enemies in the game! Our goal is to have a variety of enemy types with different behaviours, stats, and rarity. This will go hand-in-hand with Jacob’s random placement of entities throughout each level as they are generated.

Additionally, I will be creating a “bestiary”, or journal, of all of the monster's descriptions, sprites, and tips on how to deal with them. This will serve as a handy way for players to read about the different creatures that will appear in-game and allow them to form strategies accordingly!

**2. Use case diagram with scenario \_\_14**

Example:



Scenario: The level generator has just made a request for an enemy to spawn at a given spawn-point in the level

Name: Spawn Enemy

Summary: The Level Generator will ask for an enemy to place and an enemy will be selected from a library and be assigned particular attributes based on difficulty level.

Actors: Level Generator

Preconditions: Level has been generated and spawn-points initialised.

Basic sequence:

Step 1: Receive request for a new enemy.

Step 3: Generate a new enemy based on current level difficulty.

Step 4: Send new enemy object back to the level generator for placement.

Exceptions:

Step 1: A boss enemy has been requested.

Step 2: Continue process like normal but with boss enemy qualities.

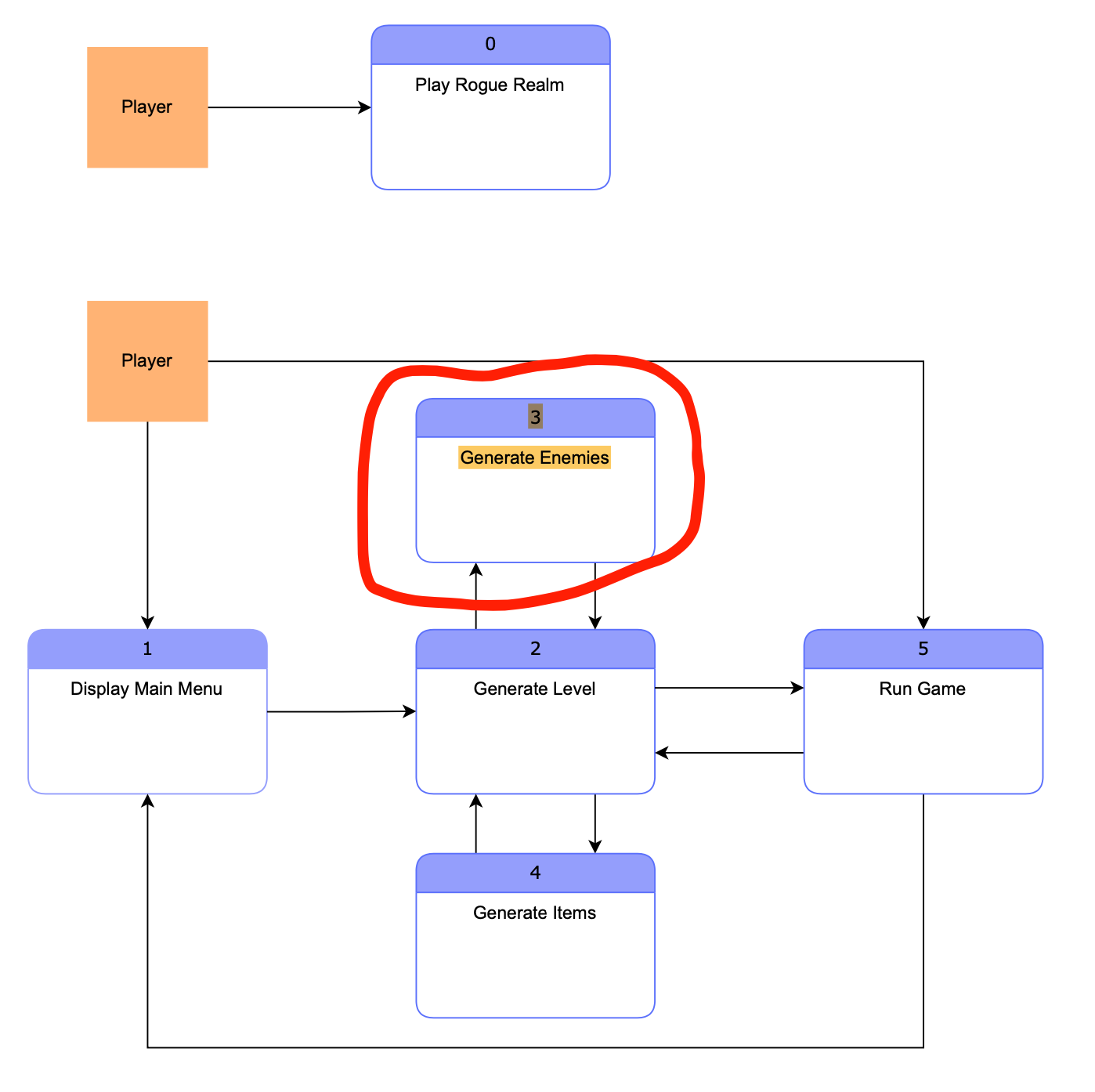
Post conditions: Enemy is spawned onto map.

Priority: 2\*

ID: C01

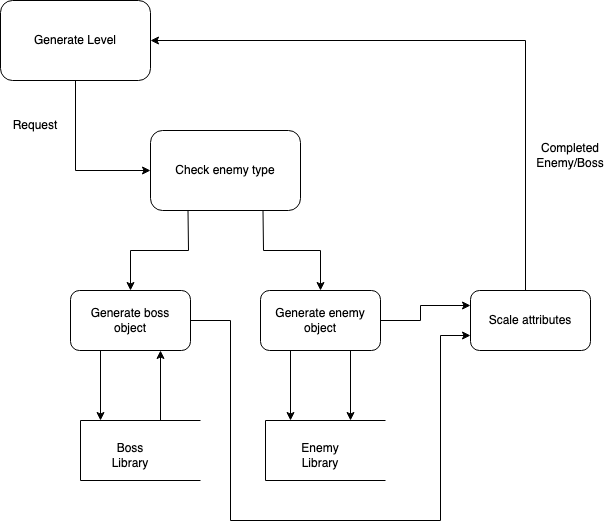
\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

**3. Data Flow diagram(s) from Level 0 to process description for your feature \_\_\_\_\_\_\_14**



Example: Above is the general data flow chart for our game on a level 0 and level 1 diagram. Below, is a level 2 diagram of the enemy spawn system and how it works

Data Flow Diagrams



Process Descriptions

1)After receiving a request from the level generator with TWO parameters(current difficulty level and whether a boss has been requested)

2)We must then move to select an enemy from a preset library of enemies or bosses that would be appropriate for the given time in-game.

3)After an enemy class object has been made from the template library, unique attributes will be assigned to the object depending on the current difficulty level reported by the level generator in order to properly scale enemy strength with player strength as the game progresses

4)The final step is to return the new enemy object to the level generator for placement in the current level..

**4. Acceptance Tests \_\_\_\_\_\_\_\_9**

[Describe the inputs and outputs of the tests you will run. Ensure you cover all the boundary cases.]

To automate this process we will create an output flow that stores info regarding the enemies such as health, damage, and frequency throughout the level generation in order to get an idea of how the enemy generator is handling requests based on current difficulty level and whether a boss has been requested.

We can test this by isolating the enemy generator and having it’s outputs stored into a text file where we can easily read the health, damage, and other relevant information about the enemies to check that the generator is creating enemies appropriately for the given scenario in each level.

**5. Timeline \_\_\_\_\_\_\_\_\_/10**

