Name: Haddie Meyer Mark_____/50

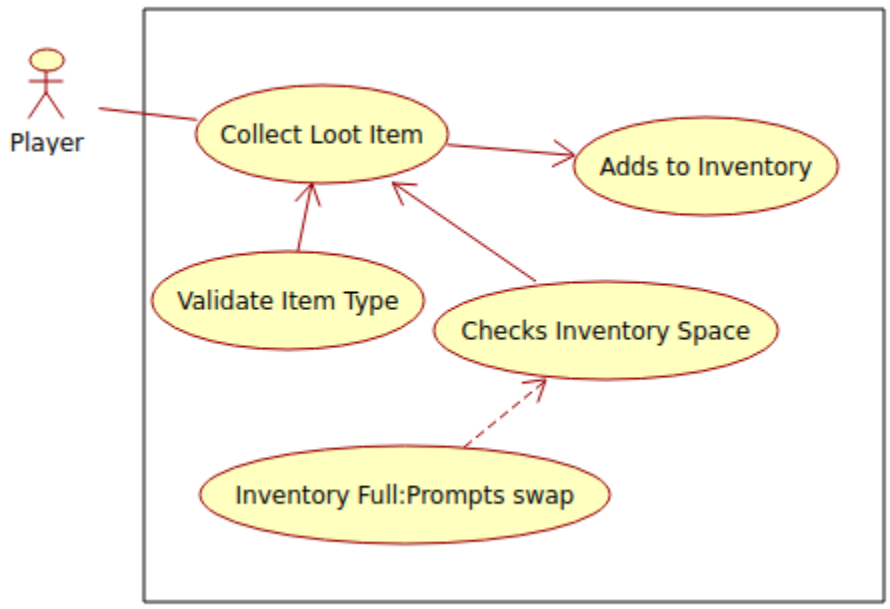
1. Brief introduction __/3

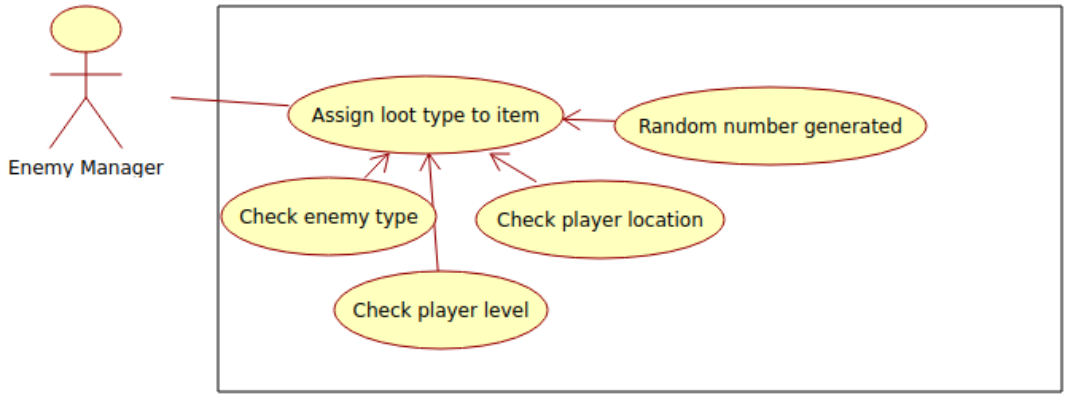
My feature for our video game, Return of the Exiled: Where is my Sandwich, is going to be the enemy drop and loot system. This will include how these items are designed, generated, and collected by the player. Due to the game's progressive nature, I will also need to create a system for ensuring that the items are useful and tied to the player's progress. Loot will include anything from healing items to weapons.

My system will ensure each enemy drops the appropriate item based on the loot table. Then the item will be spawned into the world at the enemy's prior position, and will be available for the player to interact with.

2. Use case diagram with scenario __14

Use Case Diagrams





Scenarios

Scenario 1 (1st Use Case Diagram):

Name: Collect Loot Item
Summary: A loot item must be chosen and deposited in the world for the player to pick up. The player must have enough space in their inventory to hold the item.
Actors: Player
Preconditions: An enemy has been defeated, and the loot item has already been spawned in the world.
Basic sequence:
Step 1: The player chooses to interact with the loot.
Step 2: The loot item must be validated to ensure it can be safely interacted with.
Step 3: The inventory space of the player must be checked for available space.
Step 4: The item is added to the player's inventory.
Exceptions:
Step 3: The player's inventory is full. An error message appears, prompting the player to swap an existing item.
Post conditions: The player now has the correct item stored in their inventory.
Priority: 1

Scenario 2 (2nd Use Case Diagram):

Name: Assign loot type to item
Summary: A correct type must be chosen for the loot item before it can be spawned in the world. The enemy manager must take into account all factors, including the enemy that prompted the drop, the player's level, and a random chance.
Actors: Enemy Manager
Preconditions: An enemy has been defeated.

Basic sequence:

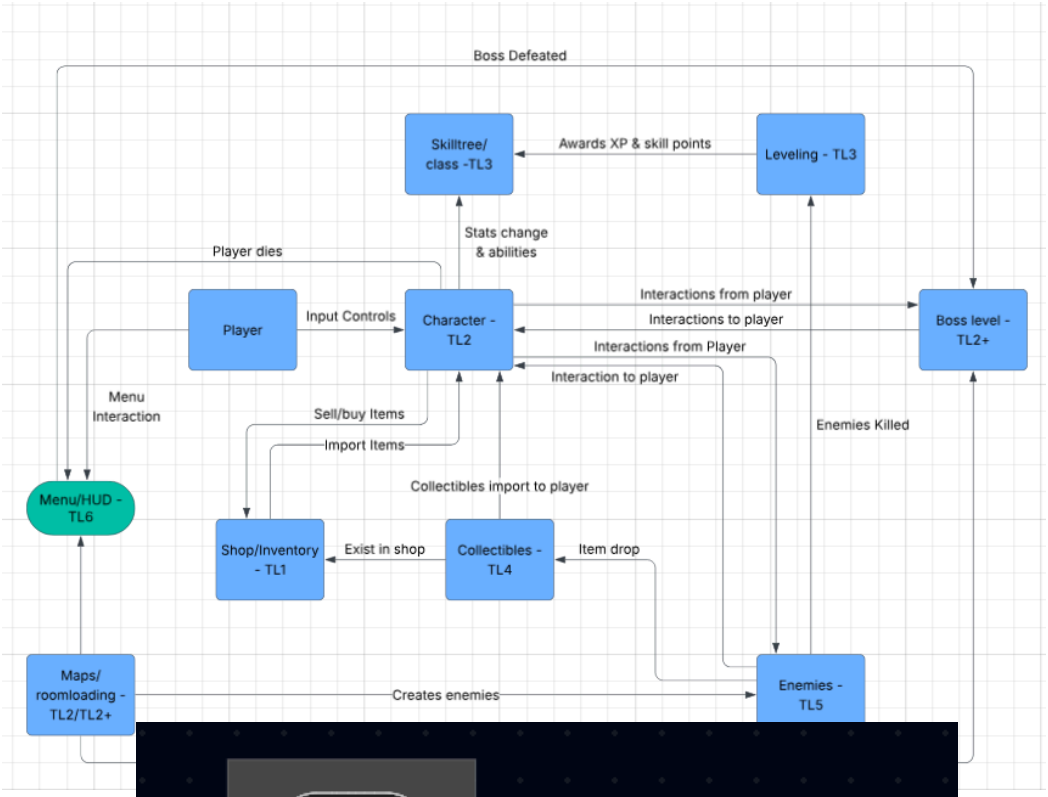
Step 1: An enemy is defeated, and the enemy manager must generate loot.
Step 2: The enemy type is checked against the table of available loot.
Step 3: The player's level is checked against the table of available loot.
Step 4: The player's location is checked against the remaining available loot.
Step 5: A random number is generated to choose from the remaining loot.

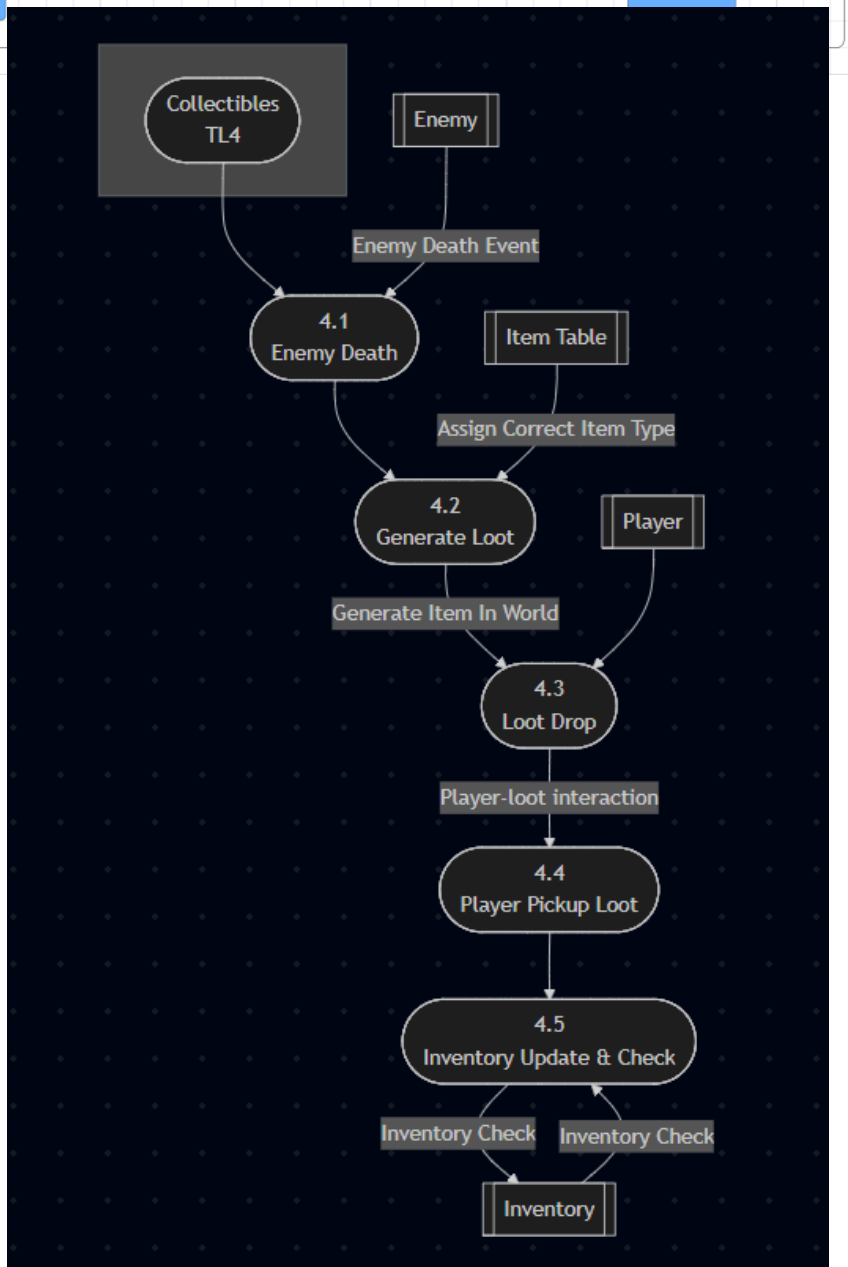
Post conditions: The correct loot type is assigned to the item.
Priority: 2

3. Data Flow diagram(s) from Level 0 to process description for your feature _____14

Data Flow Diagrams

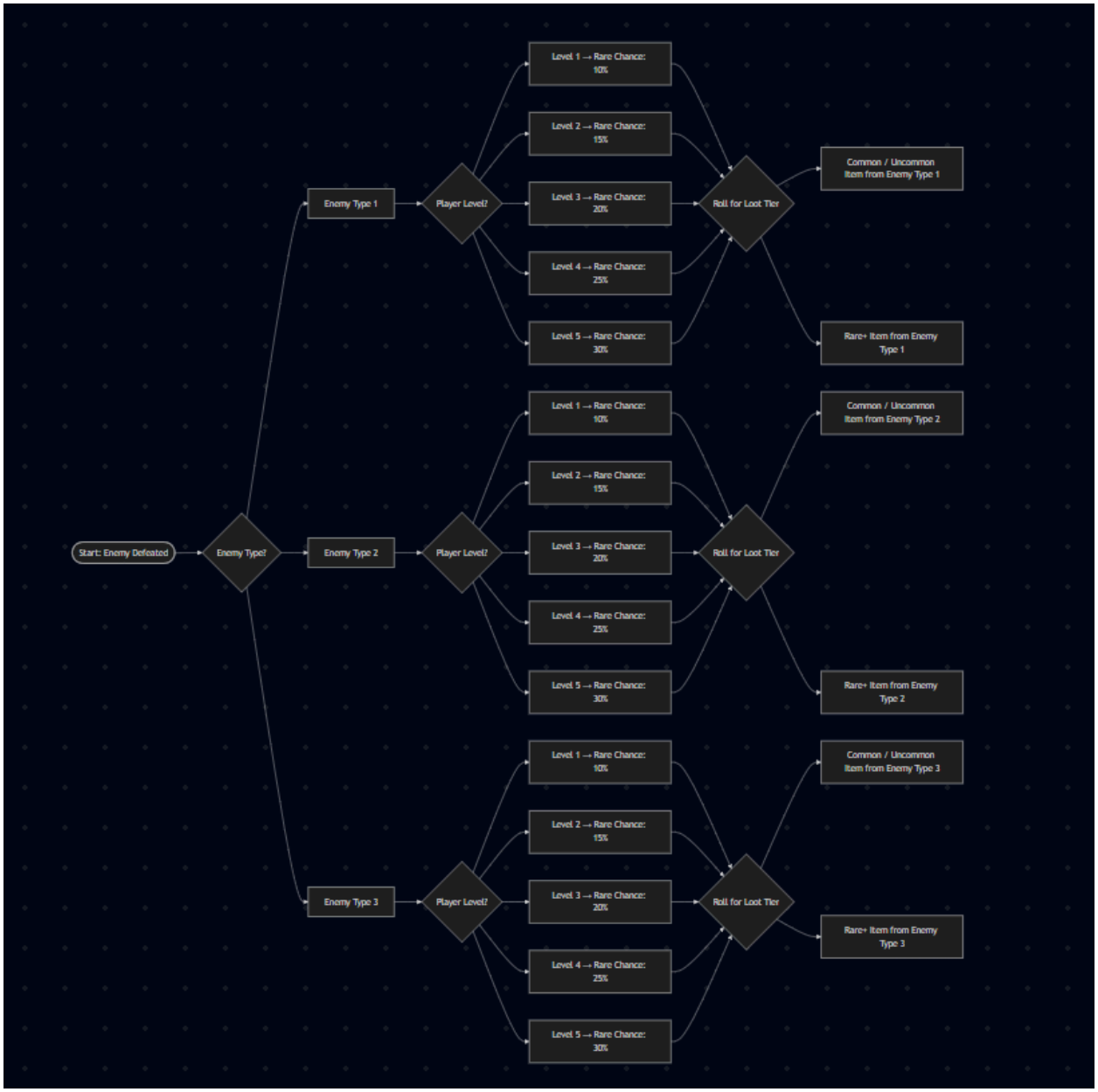
In the data flow diagrams below, I will be covering my enemy loot feature.


Boss Defeated
Skilltree/ class -TL3
Awards XP & skill points
Leveling - TL3
Stats change & abilities
Player dies
Player
Input Controls
Character - TL2
Interactions from player
Interactions to player
Boss level - TL2+
Interactions from Player
Interaction to player
Menu Interaction
Sell/buy Items
Import Items
Enemies Killed
Collectibles import to player
Menu/HUD - TL6
Shop/Inventory - TL1
Exist in shop
Collectibles - TL4
Item drop
Maps/ roomloading - TL2/TL2+
Creates enemies
Enemies - TL5


Collectibles TL4
Enemy
Enemy Death Event
4.1 Enemy Death
Item Table
Assign Correct Item Type
4.2 Generate Loot
Player
Generate Item In World
4.3 Loot Drop
Player-loot interaction
4.4 Player Pickup Loot
4.5 Inventory Update & Check
Inventory Check
Inventory Check
Inventory

Process Descriptions

I will be using a decision tree to show the Item Assignment part of my feature.



4. Acceptance Tests _____9

This feature will need to be tested on its ability to choose the correct item and its ability to generate the item in the world. It will also need to be tested on its ability to check the player's inventory space.

Item Assigner:

Run the feature 1000 times per enemy type and player level combination, across 3 enemy types and 5 player levels, for a total of 15000 executions. Send the output of each run to a file. It should have the following:

- Each enemy type has 5 possible items it can drop for each player level, from common to rare.
- Common types should appear more often than rare types. Rarest items should only appear, at most, 100 times every 1000 runs.
- No single item per combination should appear more than 400 times.
- No duplicate drops in 5 consecutive runs per combination.

Inventory Checker:

Run the feature 500 times per item type, testing against players with varying inventory storage vacancy. At least 100 of the tests should include full inventories. Send the output of each run to a file. It should have the following:

- Pickup should succeed only if there is at least 1 empty slot.
- No inventory should ever exceed its item limit.
- If pickup is unsuccessful, the feature should log an error message and display it to the player.
    - The item must remain in the world after inventory failure.

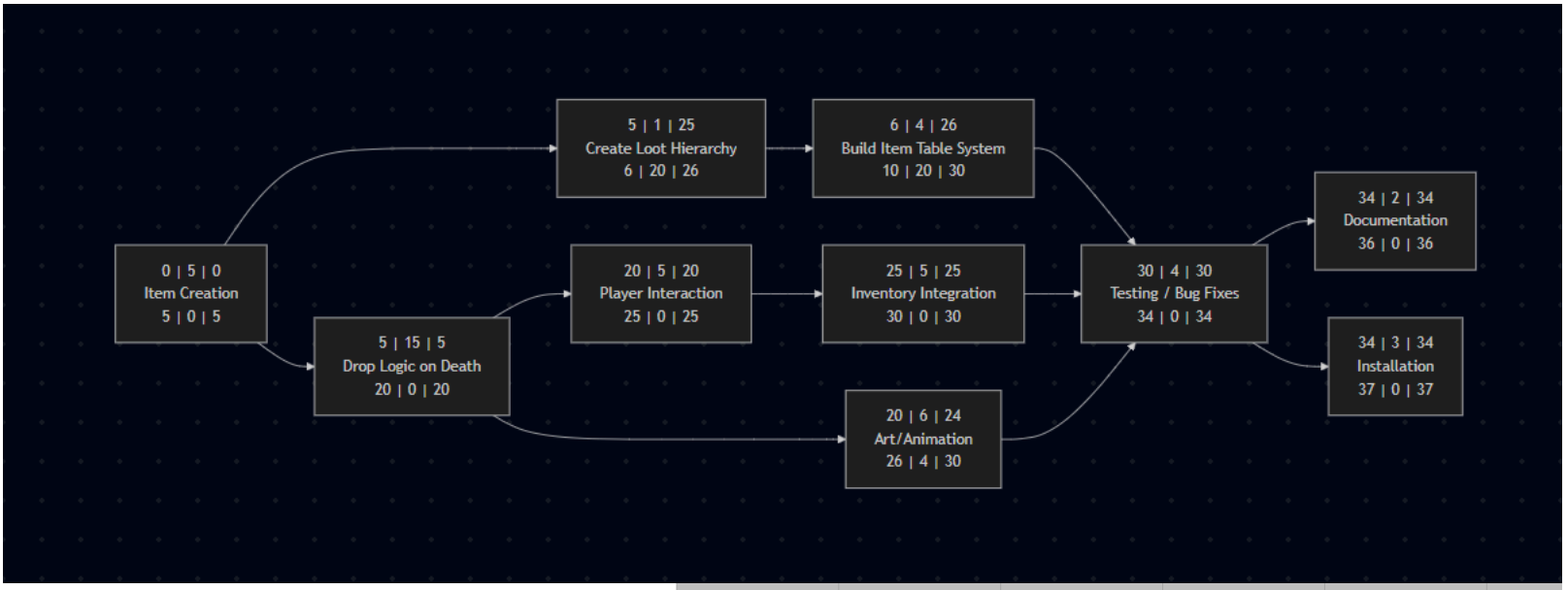
5. Timeline _____/10

Work items

Task	Duration (hours)	Predecessor Task(s)
1.) Item Creation	5	-
2.) Create Loot Hierarchy	1	1
3.) Build item table system	4	2
4.) Create and implement drop logic on enemy death	15	1
5.) Implement player	5	4

interaction		
6.) Integrate with Inventory system and implement Inventory check	5	5
7.) Art/animation	6	4
8.) Testing/bug fixes	4	6, 3, 7
9.) Documentation	2	8
10.) Installation	3	8

Pert diagram/Gantt Chart



Milestone description	Category	Assigned to	Progress	Start	Hours
Item Creation	High Risk	Haddie	0%	9/25/2025	5
Create Loot Hierarchy	Low Risk	Haddie	0%	9/30/2025	1
Build item table system	High Risk	Haddie	0%	10/1/2025	4
Create and implement drop logic on enemy death	Low Risk	Haddie	0%	9/30/2025	15
Implement Player interaction	High Risk	Haddie	0%	10/15/2025	5
Integrate with inventory system and implement inventory check	Low Risk	Haddie	0%	10/20/2025	5
Art/Animation	High Risk	Haddie	0%	10/15/2025	6
Testing/bug fixes	Low Risk	Haddie	0%	10/25/2025	4
Documentation	High Risk	Haddie	0%	10/29/2025	2
Installation	Low Risk	Haddie	0%	10/29/2025	3