

**Presented by**

Joe Msann

Michel Achkouti

Wednesday, May 1, 2024

Table of Contents

[Section 1: General Description 2](#_Toc164774031)

[**Executive Summary** 2](#_Toc164774032)

[**System Description:** 2](#_Toc164774033)

[Section 2: Requirements 5](#_Toc164774042)

[**Non-Functional Requirements** 5](#_Toc164774043)

[**Functional Requirements** 7](#_Toc164774054)

[Section 3: Functional Design 9](#_Toc164774064)

[**Use Case Descriptions** 9](#_Toc164774065)

[**Use Case Diagrams** 16](#_Toc164774066)

[**Activity Diagrams** 20](#_Toc164774074)

[Section 4: Structural Design 24](#_Toc164774079)

[**Class Diagram** 24](#_Toc164774080)

[**Class Responsibility Collaboration Cards** 25](#_Toc164774081)

[Section 5: Behavioral Design 29](#_Toc164774082)

[**Interaction Diagrams** 29](#_Toc164774083)

[**State-Chart Diagrams** 35](#_Toc164774091)

[**CRUDE Matrix** 39](#_Toc164774092)

# **Section 1: General Description**

## **Executive Summary**

Minecraft is an open-world game where a player can freely explore the world and interact with it. World generation relies on complex algorithms. The game’s world consists of 3 dimensions, or layers. What is outside these dimensions is the Void, i.e., nothingness, similar to outer space.

Layer 0 is the starting dimension called the “Overworld” where the player spawns. There exists a layer -1 called the “Nether” below the Overworld and a layer 1 above called the “End”. Both can be accessed using their respective portals (player-activated gateways).

All dimensions employ the coordinate system, with Y being the vertical axis, and X and Z being the horizontal axes. World generation is almost infinite.

Each dimension has a specific set of biomes (ecosystems), generated structures, and mobs (entities).

Biomes house unique varieties of flora and fauna and dictate certain mob behaviors as well as weather and structure generation. Structures generate according to the biome they are located in as well as their coordinates. Mobs can sometimes generate within structures.

## **System Description:**

## **The Dimensions:**

The Overworld is the starting dimension where the player first spawns. It is split into the surface and the underground. The surface consists of forests, jungles, plains, deserts, oceans, and mountains, while the underground is a vast and complex system of caves and rare structures.

The Nether can be accessed by building a portal to it from the Overworld. It is a hell-like landscape, where hostile mobs are predominant, and oceans of lava stretch vastly. Structures containing essential items for the game’s progression generate in the Nether. One unit on the coordinate system in the Nether is equivalent to 8 units in the Overworld.

The End can only be accessed through a pre-existing End Portal, found in a specific structure in the Overworld called the Stronghold. To get there, the player must use a unique item called the Eye of Ender that path-finds towards the portal. These Eyes are also used to activate the portal. Once the player arrives at the End, he must defeat a boss entity called the “Ender Dragon”. By defeating it, the player will beat the game[[1]](#footnote-1). The End consists of one main island where the Ender Dragon resides, and an infinite generation of smaller islands distant from the main island containing End Cities.

## **The Player**

Minecraft can be played single-player in a private world or multi-player on a server with other players. The player has a hidden inventory to store his items and a hotbar to carry items visibly, and can interact freely with the world.

There are 3 main modes the player can be in in a world: Creative, Survival, and Spectator.

In Creative mode, the player has access to all the blocks and items the game has to offer, as well as a means to manually spawn almost all mobs. The player can also fly and use special commands, and is immune to all types of damage.

In Survival mode, the player spawns with nothing in his inventory. He has to use his bare hands to “mine” (break) blocks around him and collect them. The player has access to a 2x2 grid that he can use to “craft” (build) new items from already acquired items. The main goal a player seeks at the start of the game is to craft a “Crafting Table” which upgrades the 2x2 grid to a 3x3 grid, allowing him to craft much more items.

The player in this mode has a specific number of health points, and can die if his health is depleted. However, the player can respawn. The player has a hunger or stamina bar which depletes over time and when sprinting. To replenish it, the player must eat. A player gains EXP (Experience points) after killing mobs or mining ores.

In Spectator mode, the player can move across all axes, even phasing through solid blocks like a ghost. However, the player is not able to interact with the World.

In terms of movement, the player can walk, sprint, jump, crouch, swim, and fast-swim. In creative, the player can also fly. In spectator, the player can also phase through solids.

## **The Inventory**

The inventory is comprised of 4 slots exclusively for armor the player is wearing, 27 slots for storage, 9 slots for the hotbar, and 1 slot for the off-hand (extra slot used mainly for tools).

There exist storage blocks such as the chest that offer 27 to 54 slots of extra storage in a fixed place.

## **Items**

Items are abstract; some items can be realized into blocks, while others cannot. There are 6 categories for items: Static Blocks, Dynamic Blocks, Interactive Blocks, Utility items, Materials, and Creative-Exclusive. As the name suggests, Creative-Exclusive items are items only available in Creative mode.

Block items are items that can be placed or broken and collected by the player. Block items of the same type can be stacked together in a single slot in the inventory. Static Blocks exist in a fixed state, while Dynamic Blocks can change states. Interactive Blocks have additional functions useful to the player, such as storage or tool management.

Utility items are non-placeable, non-block items, and are purely functional. Almost all Utility items are crafted using Materials. There are weapons such as the sword and the bow, mining tools such as the axe and the pickaxe, and armor items such as the chest plate, helmet, and the shield. Utility items have specific durability’s and can break if the durability is exhausted. Utility items can be enhanced using “Enchantments”.

Materials are obtained via various means, most commonly by mining ore blocks using a respective type of pickaxe. They can be used to craft and enhance tools and armor, as well as many more complex functions.

## **Blocks**

Blocks can be either found naturally in the World or are crafted by the player.

In Creative mode, blocks can be broken instantly, and do not drop their respective item.

In Survival mode, each block has a specific hardness level, and a specific tool that breaks it efficiently and quickly. For example, a wood block can be broken with a shovel, but it is more efficient and faster to break it with an axe. When a block is broken, it drops its respective item which can be picked up by the player.

## **Enchanting and Experience**

EXP can be used as fuel to enchant (enhance) tools and armor using an Interactive Block called the “Enchanting Table”.

An enchantment is an upgrade applied to a tool. It can improve mining speed, item drop rate (for variable drop probability items such as ores and mob loot), durability (for armor as well) or provide new attributes to the tool. For example, a sword can receive the Fire Aspect enchantment and apply burning damage to its victims.

Some enchantments have levels; the higher the level, the more potent the enchantment, but the more EXP it costs to apply.

## **Structures**

Some structures generate randomly throughout the world, while others generate only in specific biomes. Structures can be either one building, like the Desert Temple, or a series of buildings, like villages. Almost all structures contain a collection of items the player can loot, as well as specific mob spawning patterns.

Each structure has its own level of rarity and item quantity and spawn probability.

## **Mobs**

Mobs are non-player entities that roam the world. All mobs have a health bar and can be killed. Most mobs drop items when they are killed. Golems are a type of mob that can be built using a specific totem pattern of blocks. Some mobs can carry items, including weapons.

Hostile mobs attack the player on sight. A significant number of hostile mobs can only spawn at night or in the dark (Example: Zombies and Skeletons).

Passive (or Peaceful) mobs stay passive even when provoked. (Example: Cows and Sheep)

Neutral mobs are passive unless threatened or attacked. (Example: Polar Bears and Wolves)

Mobs have specific algorithms that allow them to path-find across the world, attack the player, or escape from the player or other mobs.

Some mobs can be tamed by the player (ex: the Wolf). Most tamed mobs follow the player and can sit in place when commanded. Some tamed mobs can be ridden (ex: the Horse or the Llama).

Some mobs can be bred together using specific items to produce offspring.

# **Section 2: Requirements**

## **Non-Functional Requirements**

### Cultural Requirements

* 1. Languages
     1. Should support all languages.
     2. Should support entertainment languages. (Ex: Pirate Speak, Shakespearean)
     3. Should support text-to-speech for visually impaired users.
  2. Regions
     1. Should be available in all regions.
  3. Should include a color-blind mode.

### Portability

* 1. Should be available for Windows, Mac, Linux, PS3, PS4 series, PS5 series, Xbox 360, Xbox One series, Xbox series X/S, Android, iOS.
  2. Should allow cross-play across different platforms.
  3. Should provide the source code for users to modify their game to their liking.

### Usability

* 1. The user interface should consist of the World and the item grids. (hot-bar and inventory)
  2. The player should be able to move using the W-A-S-D keys or arrow keys.
  3. The player should be able to look using the mouse, touchpad, or controller joystick.
  4. The player should be able to view different interfaces relating to specific game details using assigned shortcuts. (Ex: view debug screen)
  5. The player should be able to set his own bindings for keyboard keys and mouse as well as controller input.

### Compatibility

* 1. The player should be able to use a keyboard and mouse setup, or a controller setup.

### Operational Requirements

* 1. Should be coded in one language for all versions.
  2. Offline games should be backed up to game servers.
  3. Players should be able to change an offline world to an online one.
  4. Should allow users to host their own private servers.
  5. Should provide cloud-based servers.
  6. Offline games should not require servers or an internet connection.
  7. Players connected to the same LAN should be able to join any of their worlds.

### Security Requirements

* 1. The game should implement security measures to protect against cheating and hacking.
  2. The player should have a unique gamertag.
  3. Passwords should meet certain strength criteria.

### Multiplayer Server Requirements

* 1. Should be moderated regularly.
  2. Should provide the option to be set to public or private.
  3. Should be able to whitelist and blacklist players.
  4. Should be able to censor chats.

### Performance Requirements

* 1. The game should automatically optimize its graphics settings to accommodate to the operating system’s computing capacity.
  2. The player should be able to manually customize the following graphics settings:
     1. Render distance[[2]](#footnote-2)
     2. Simulation distance[[3]](#footnote-3)
     3. Full screen toggle
     4. Entity shadows
     5. World shaders
     6. Lighting style
     7. Field of vision

### Reliability Requirements

* 1. The game should be able to handle as much load as the user creates.
  2. The game should always provide a backup file for each world in case of a crash to avoid corruption.
  3. In case of a crash, the game should generate a crash report and send it to the developer team.

### Availability

* 1. The game should have 100% uptime.
  2. The game should have consistent patch updates and bug fixes.
  3. The user should be able to play any current or previous version of the game.
  4. The user should have access to upcoming version snapshots[[4]](#footnote-4).

## **Functional Requirements**

### Worlds:

* 1. The game should generate infinite, procedurally generated worlds.
  2. The game should generate the various biomes with their respective structures. (ex: jungles with jungle temples, snow mountains with igloos…)
  3. The game should generate all the terrain types including plains, mountains, caves and oceans.
  4. Players should be able to configure world generation preferences. (Ex: Single biome, flat world, amplified terrain)
  5. Players should be able to access specific worlds using seeds[[5]](#footnote-5).
  6. Players should be able to save their worlds and load them from previous saved states.

### Block mechanics:

* 1. Players should be able to modify the world by adding, removing and interacting with blocks.
  2. Every type of block should have its own distinct properties. (ex: you can jump on slime blocks, slide on ice blocks, get hurt by cactus…)
  3. Players should be able to collect and place blocks as well as use them in crafting.

### Crafting System:

* 1. The game should have a crafting system manipulated using the 2x2 crafting grid found in the player’s inventory and a 3x3 crafting grid when using a crafting table.
  2. The crafting system should allow players to create tools, food and new blocks using collected resources.
  3. Most of the craftable items should only be exclusively acquired using the crafting table.

### Game Modes:

* 1. The game should have a creative mode for the player to access any item in the game, be able to fly, and teleport to any biome/structure without risk of dying.
  2. The game should have a survival mode where the player should have a health bar, hunger bar, and should combat against mobs to survive. Players should gather resources, craft items, build shelters and explore the structures.
  3. The game should have a spectator mode where players can spectate other players and move through objects without being able to interact with the world.

### Time & Weather Cycle:

* 1. The game should have a dynamic day/night cycle that affects gameplay, with some entities only being able to spawn at night and hurt by the sunlight.
  2. The game should have a dynamic weather cycle that affects gameplay, with thunderstorms summoning lightning strikes that can kill the player, snow appearing on mountains and in snow biomes, and rain that can keep night entities alive during the day.

### Multiplayer:

* 1. The game should be able to support multiplayer gameplay, by allowing players to connect to servers, host their own server, or join a world if connected to the same LAN.
  2. Players should be able to log in or register using a Microsoft account.
  3. Each player should be able to add friends using the gamertag or the email address of the other player.
  4. The player should be able to access new player skins, their friends list and open online servers when logged in to their account.
  5. The game should provide official servers where events are hosted on a regular basis.

### Entities:

* 1. Non-player characters (NPCs) should exhibit basic AI for pathfinding, interaction and environmental navigation.
  2. Hostile and neutral entities should especially exhibit basic AI for combat and interaction with the player.

### Inventory System:

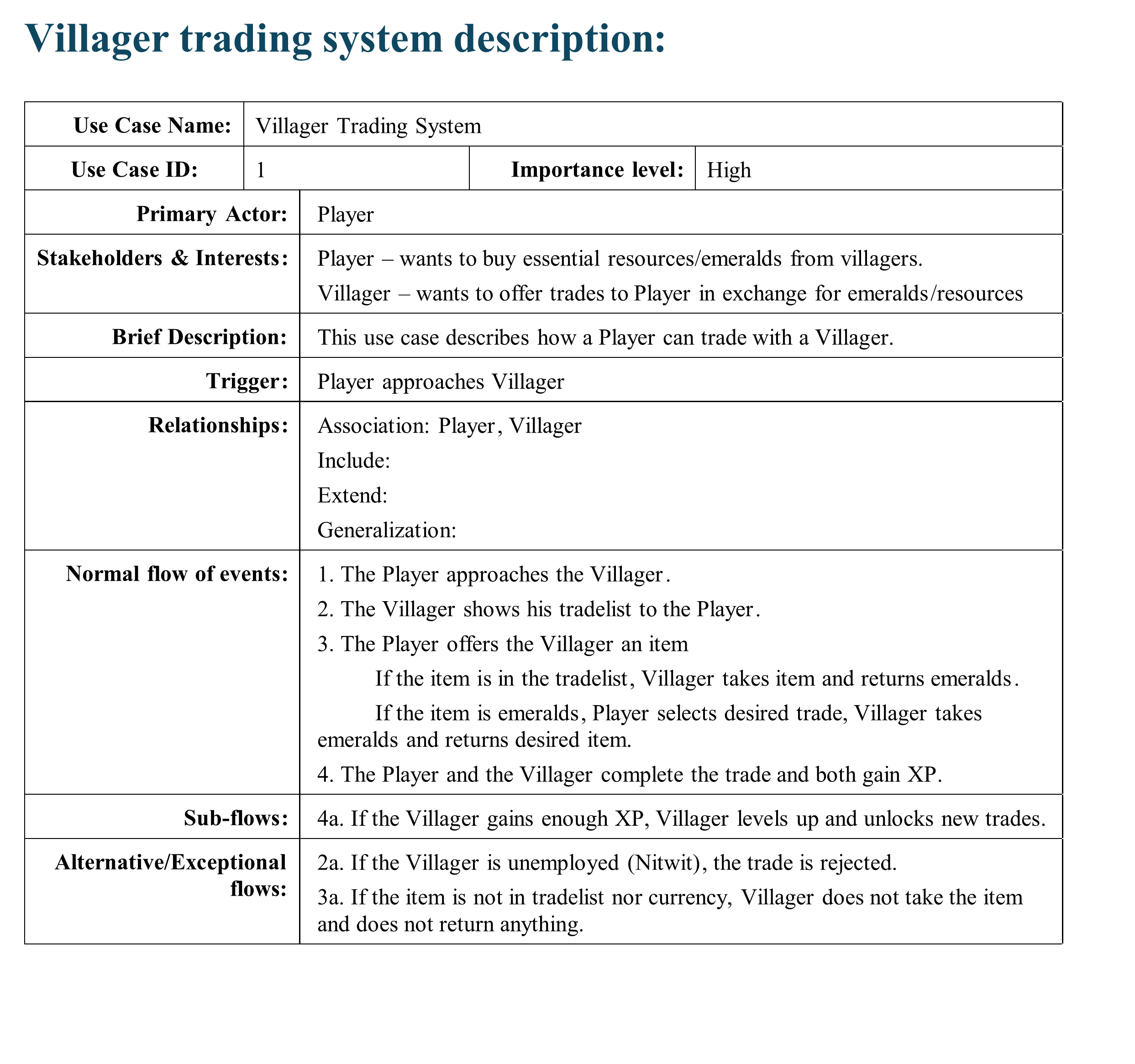
* 1. Players should have an 3x9 inventory grid to store and manage acquired resources.
  2. The inventory system should contain a 2x2 crafting grid that allows the crafting of some items.
  3. The inventory should allow the player to equip armor, and place an item in his left hand for dual wielding (cannot use the item when held by the left hand of the player).

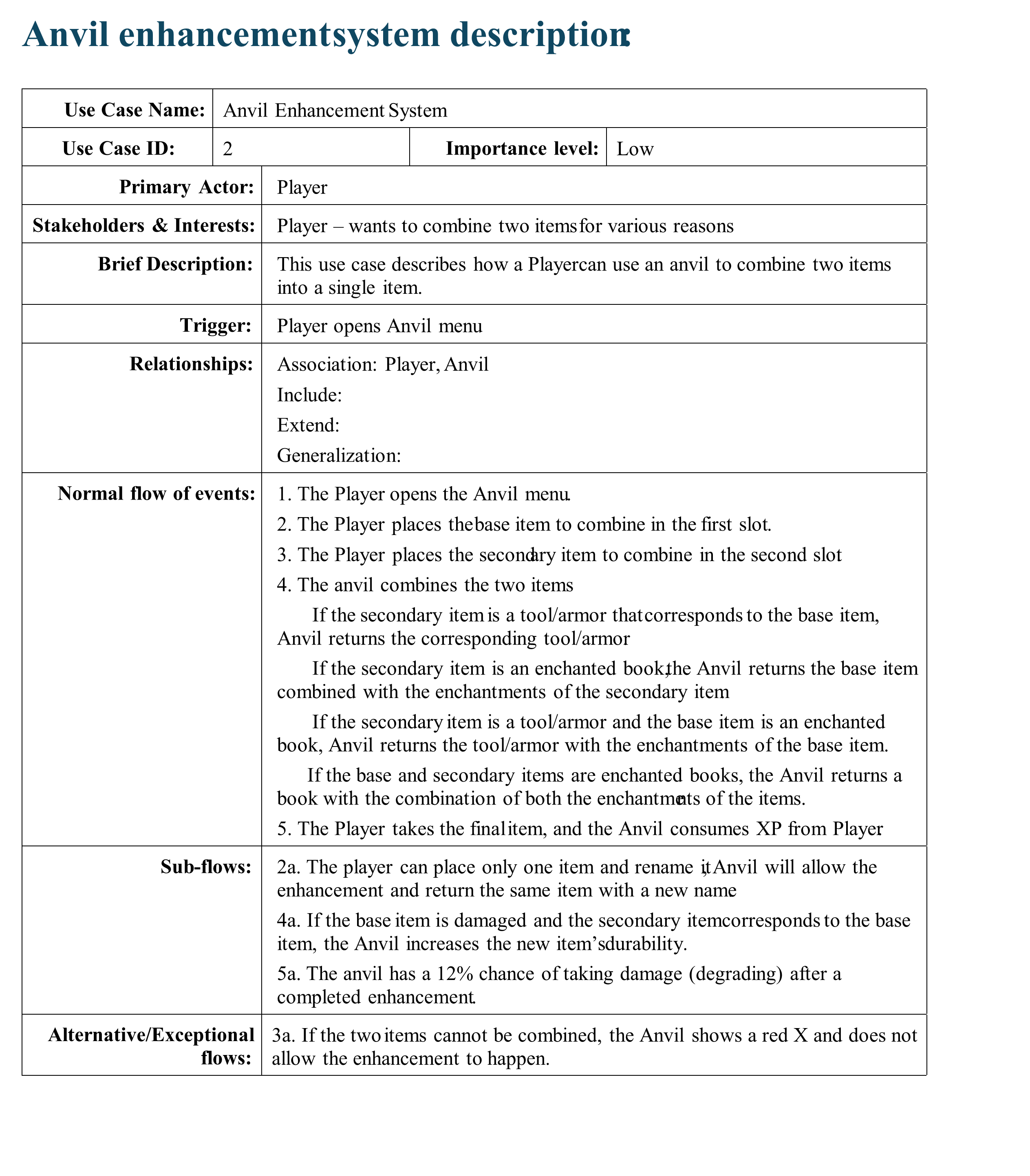
### Mods support:

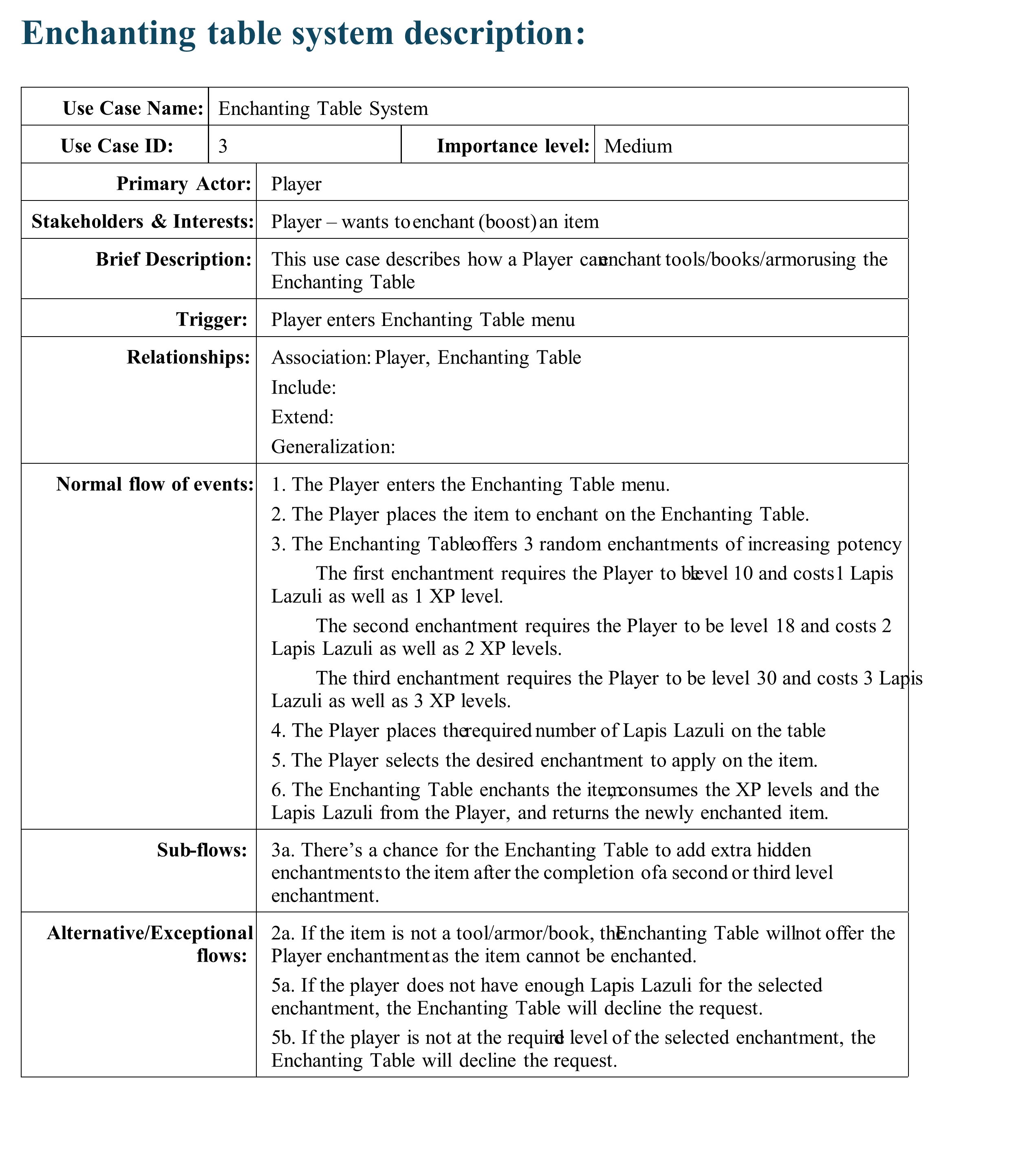
* 1. The source code of the game should be provided to players to send feedback to developers and ease the modification procedure.
  2. The player should be able to introduce modifications that add new content to the game. (ex: add a new mob, new biome, new portal…)
  3. The game should support modifications that alter the graphics of the base game.
  4. The game should allow players to download modifications that alter gameplay or change the behavior of the game.

# **Section 3: Functional Design**

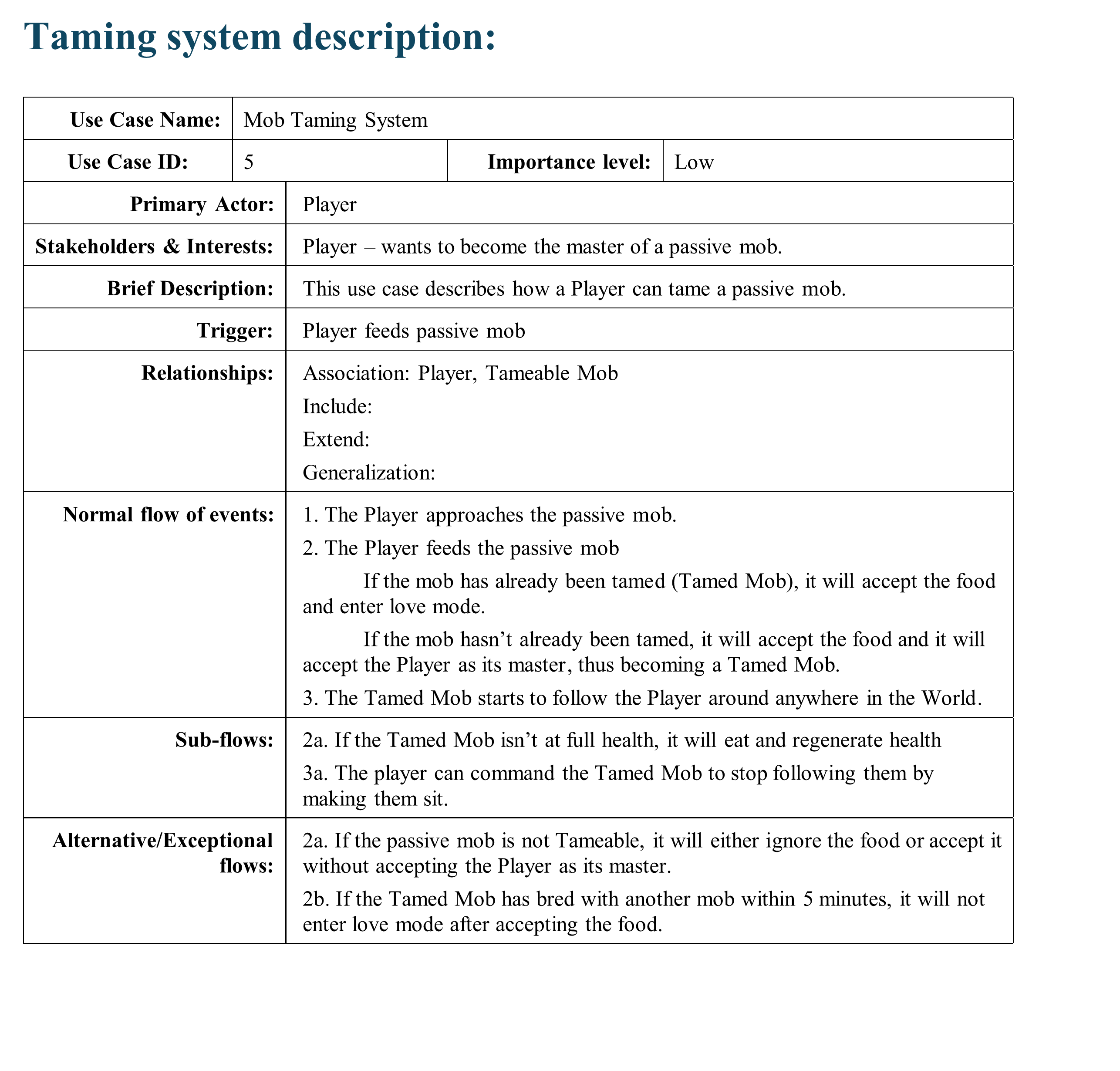
## **Use Case Descriptions**

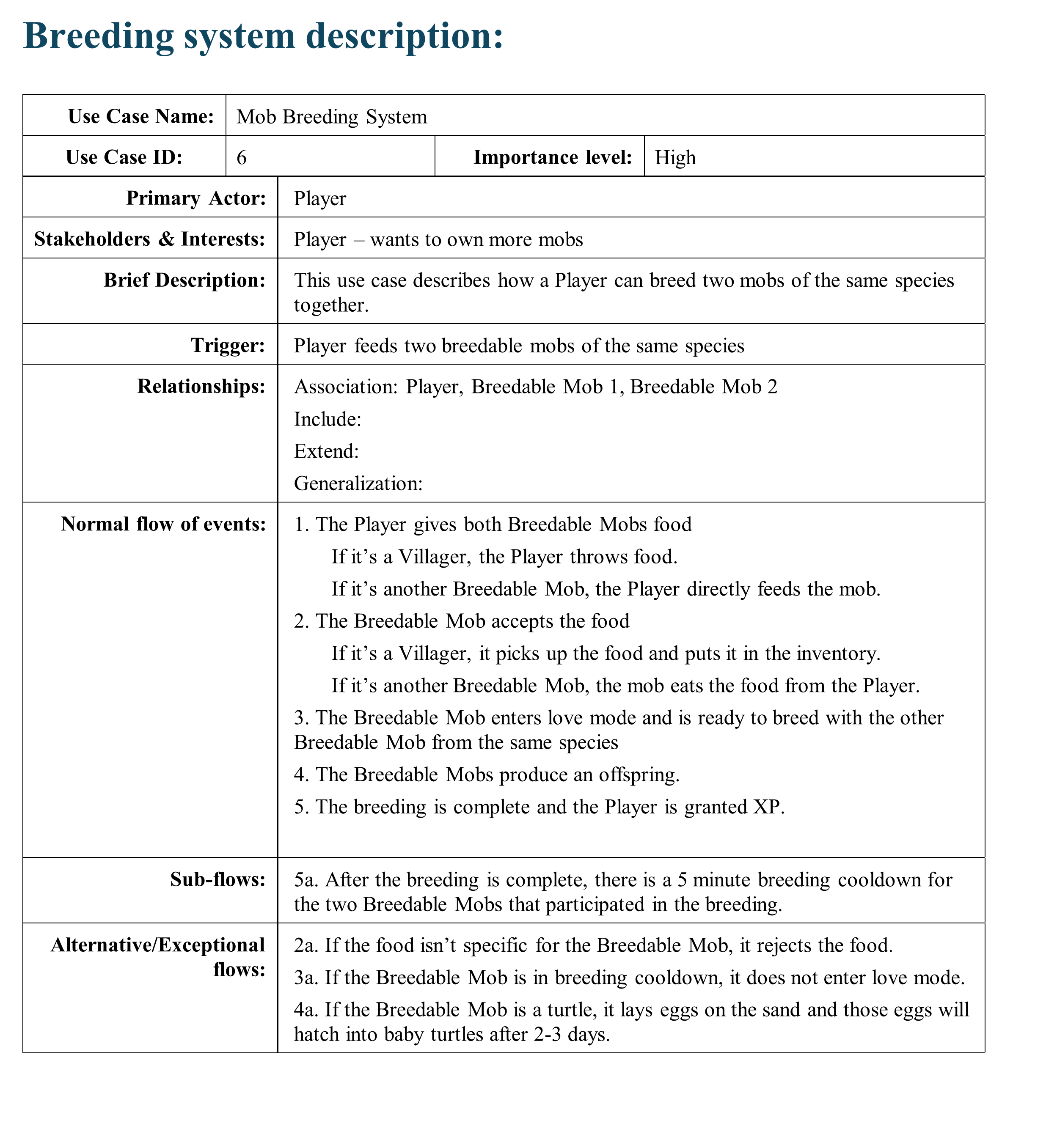


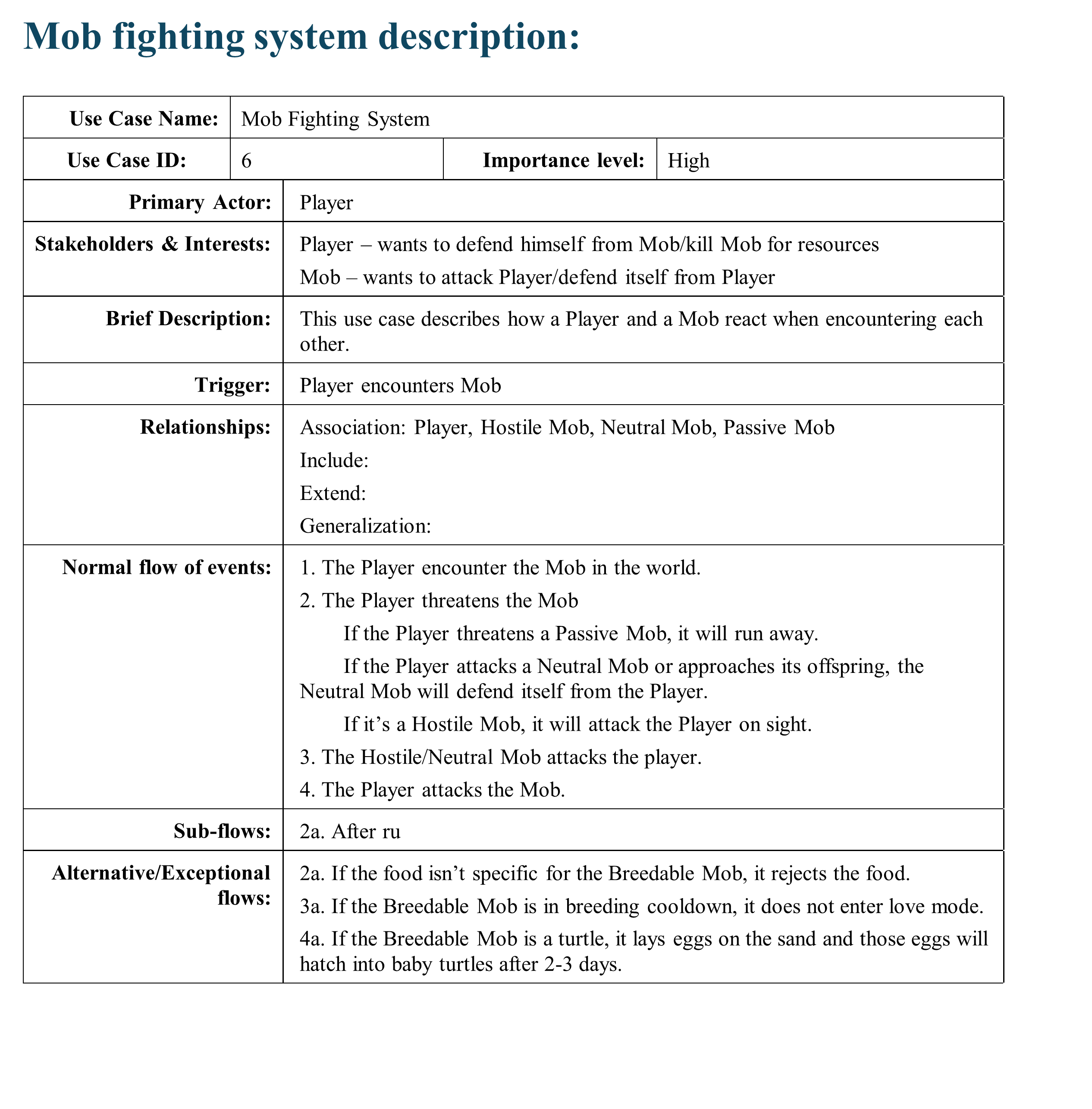






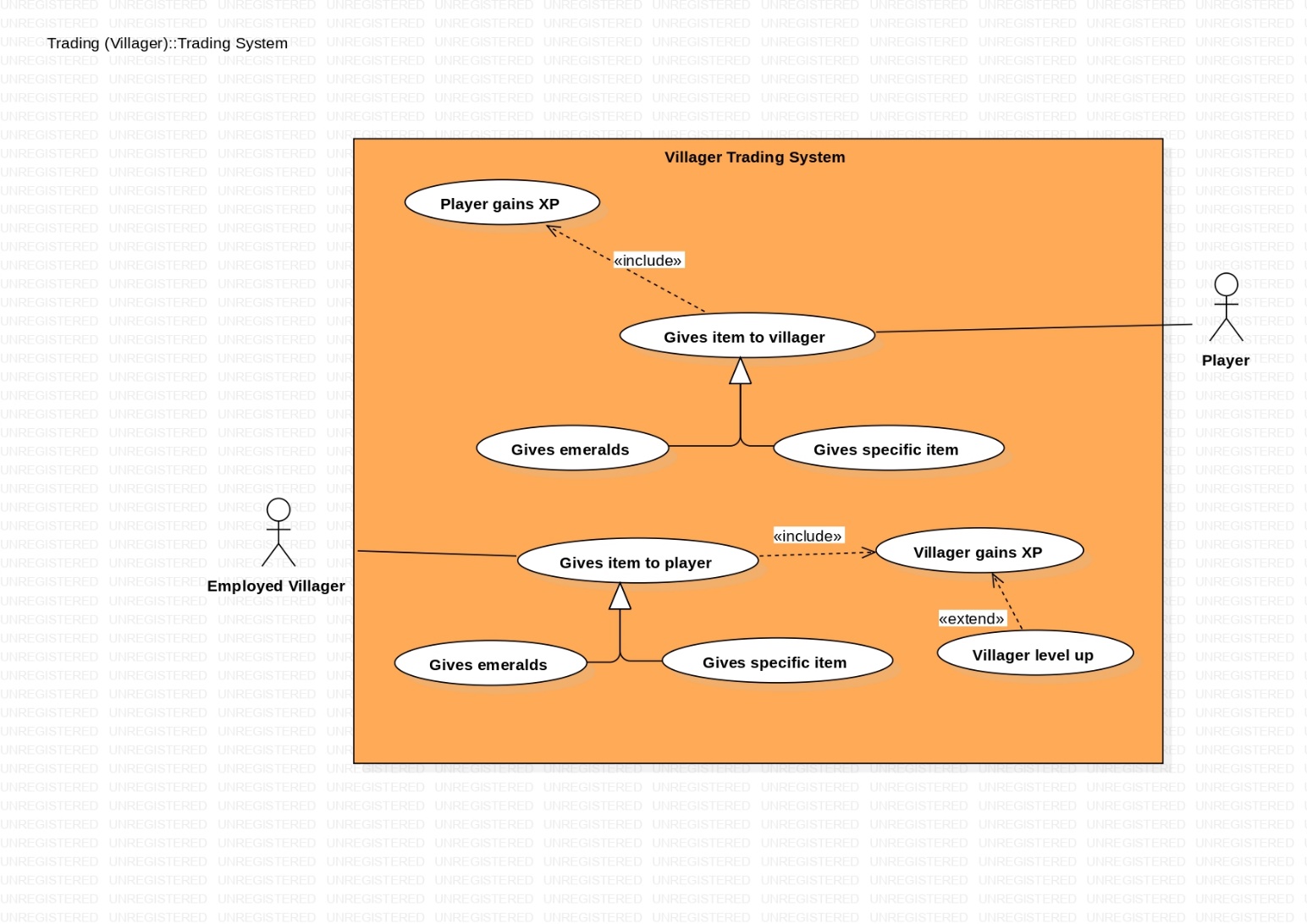




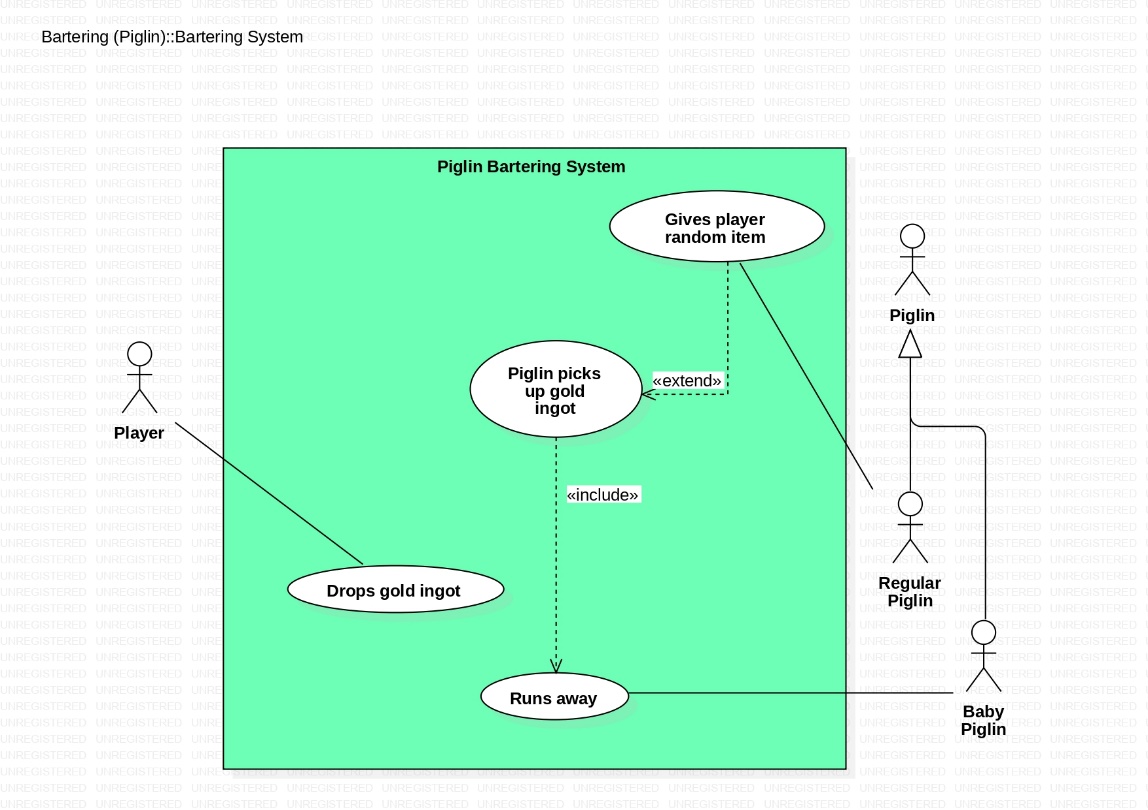


## **Use Case Diagrams**

### Villager Trading System



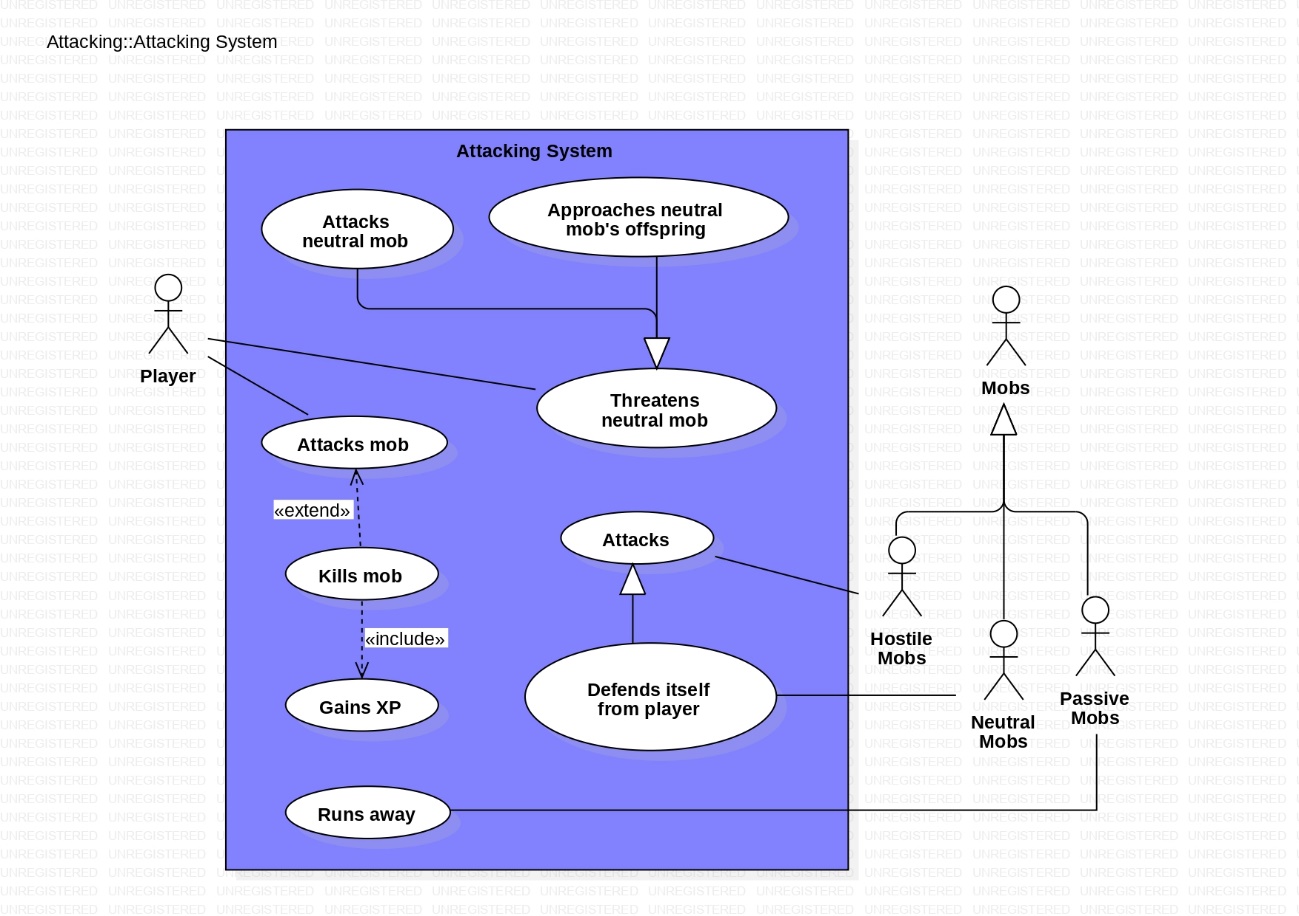
### Piglin Bartering System



### Anvil Enhancement System

### Enchanting Table System

### Attacking System

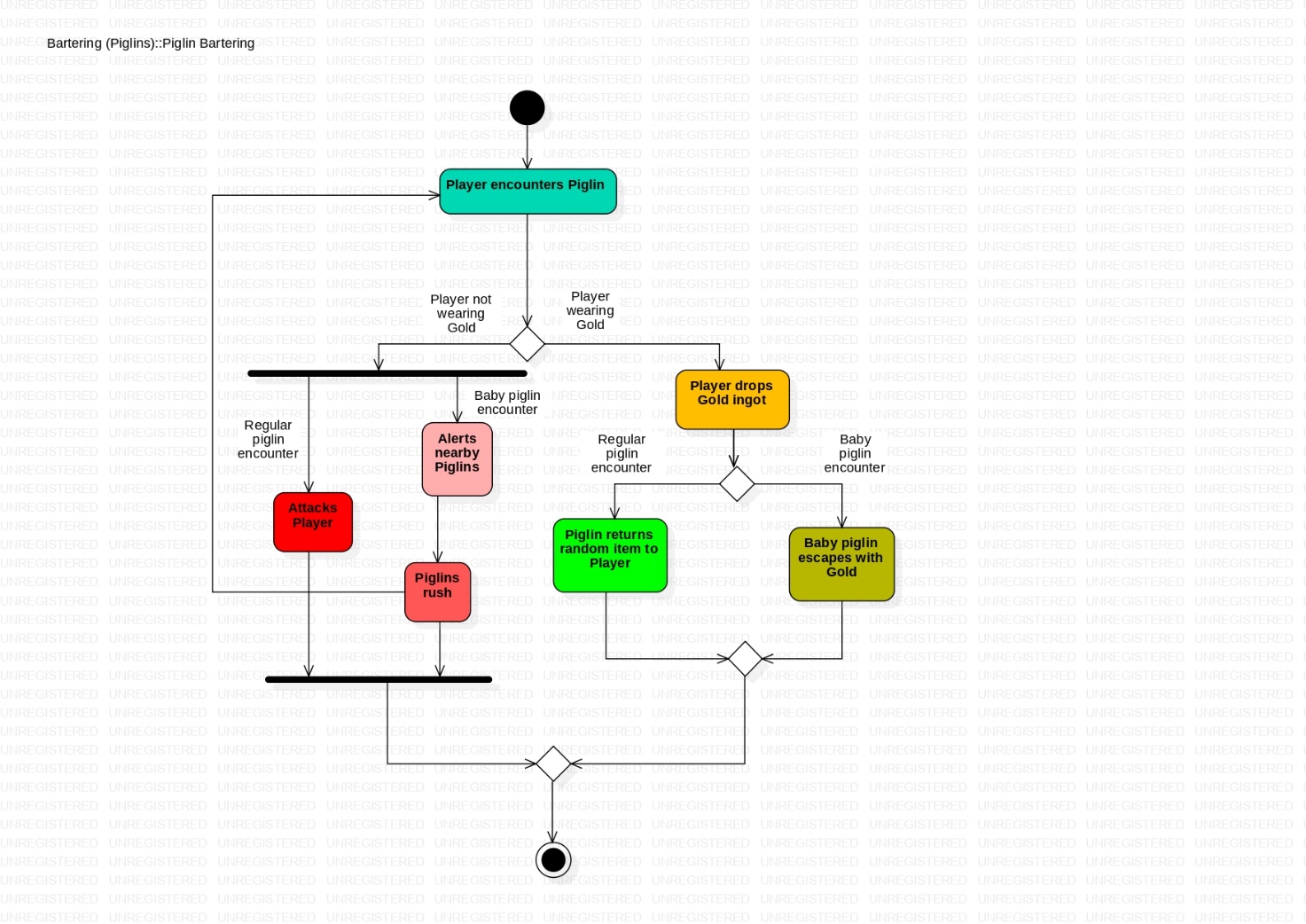


### Taming System

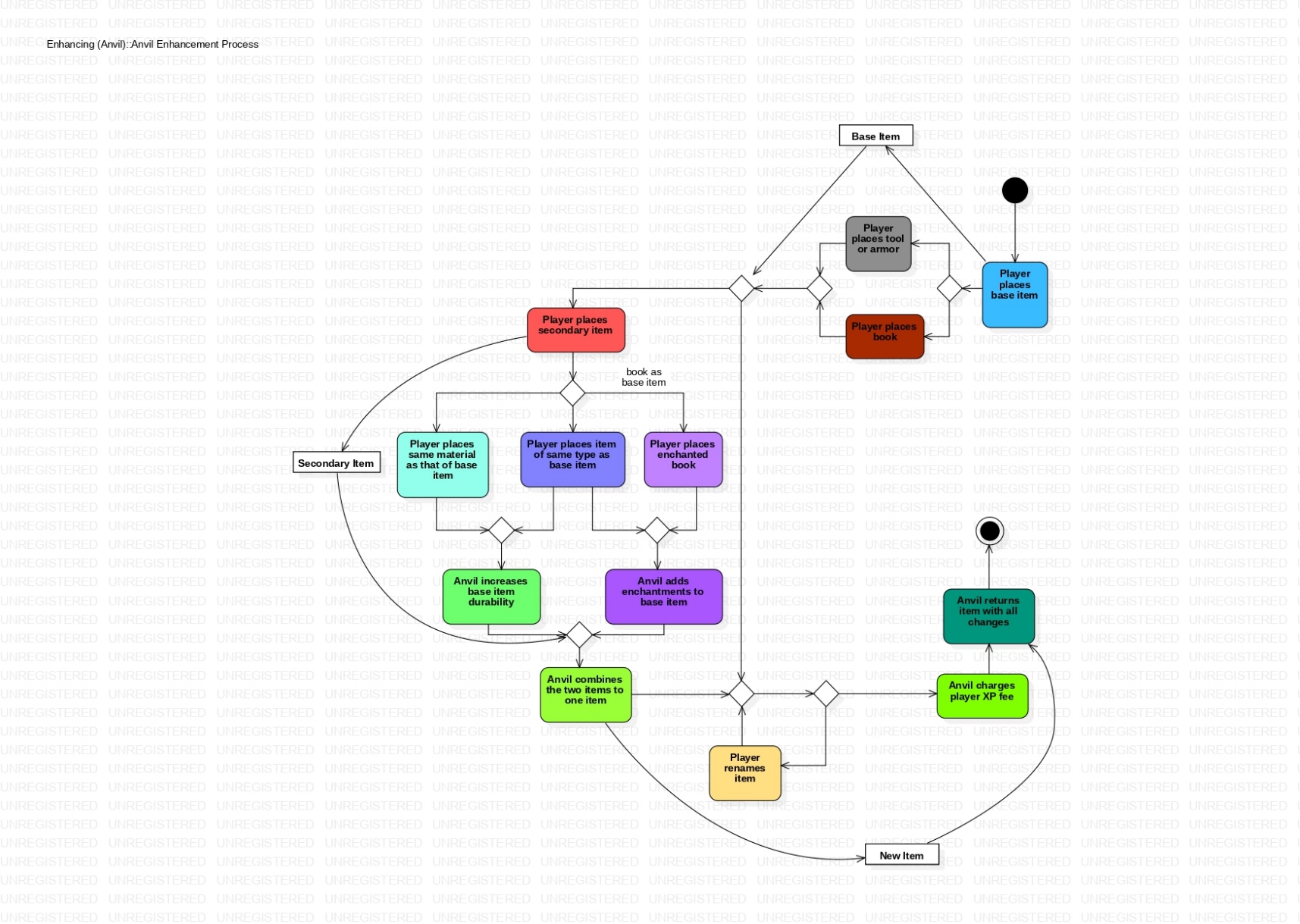
### Breeding System

## **Activity Diagrams**

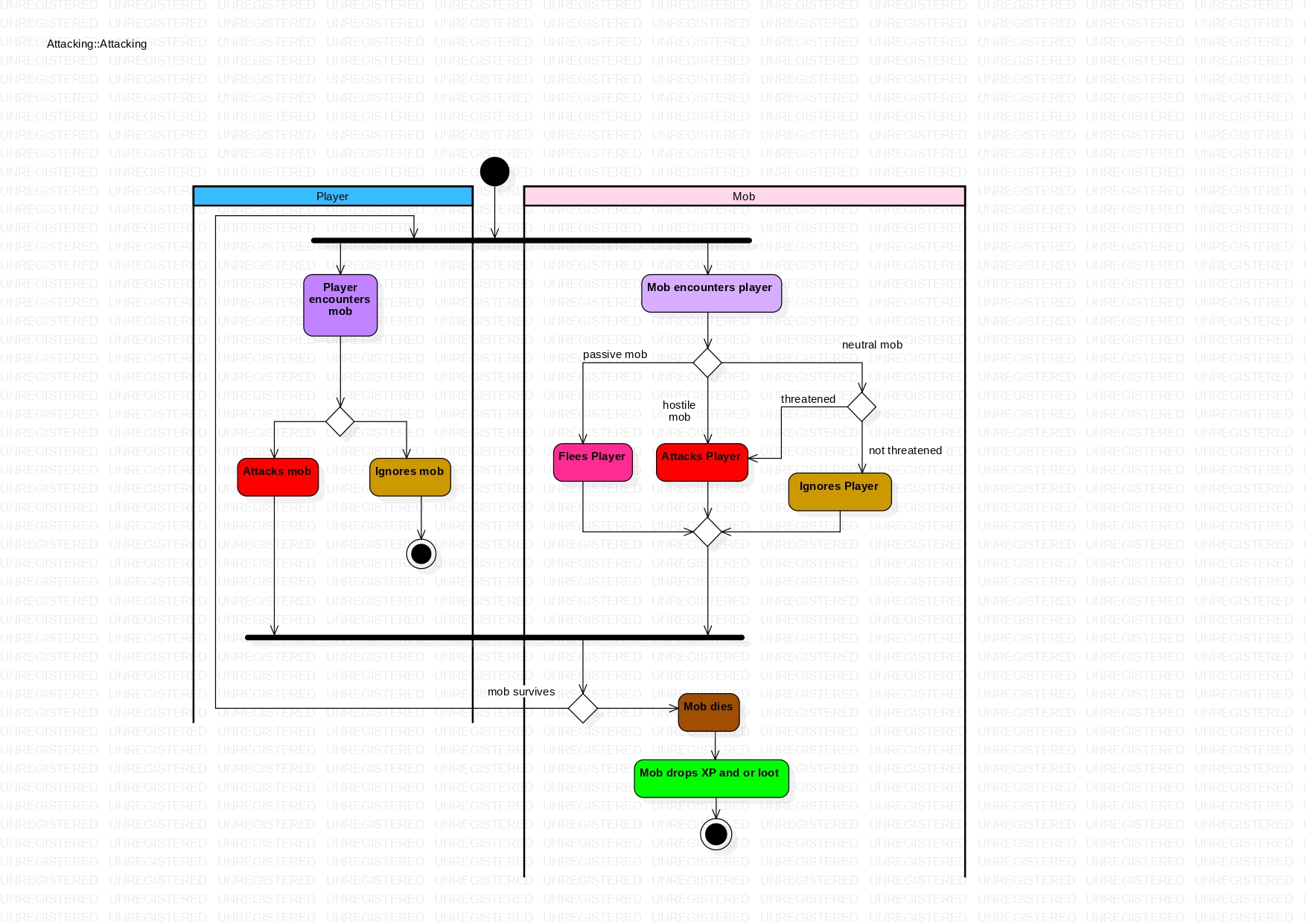
### Piglin Bartering

****

### Anvil Enhancement

****

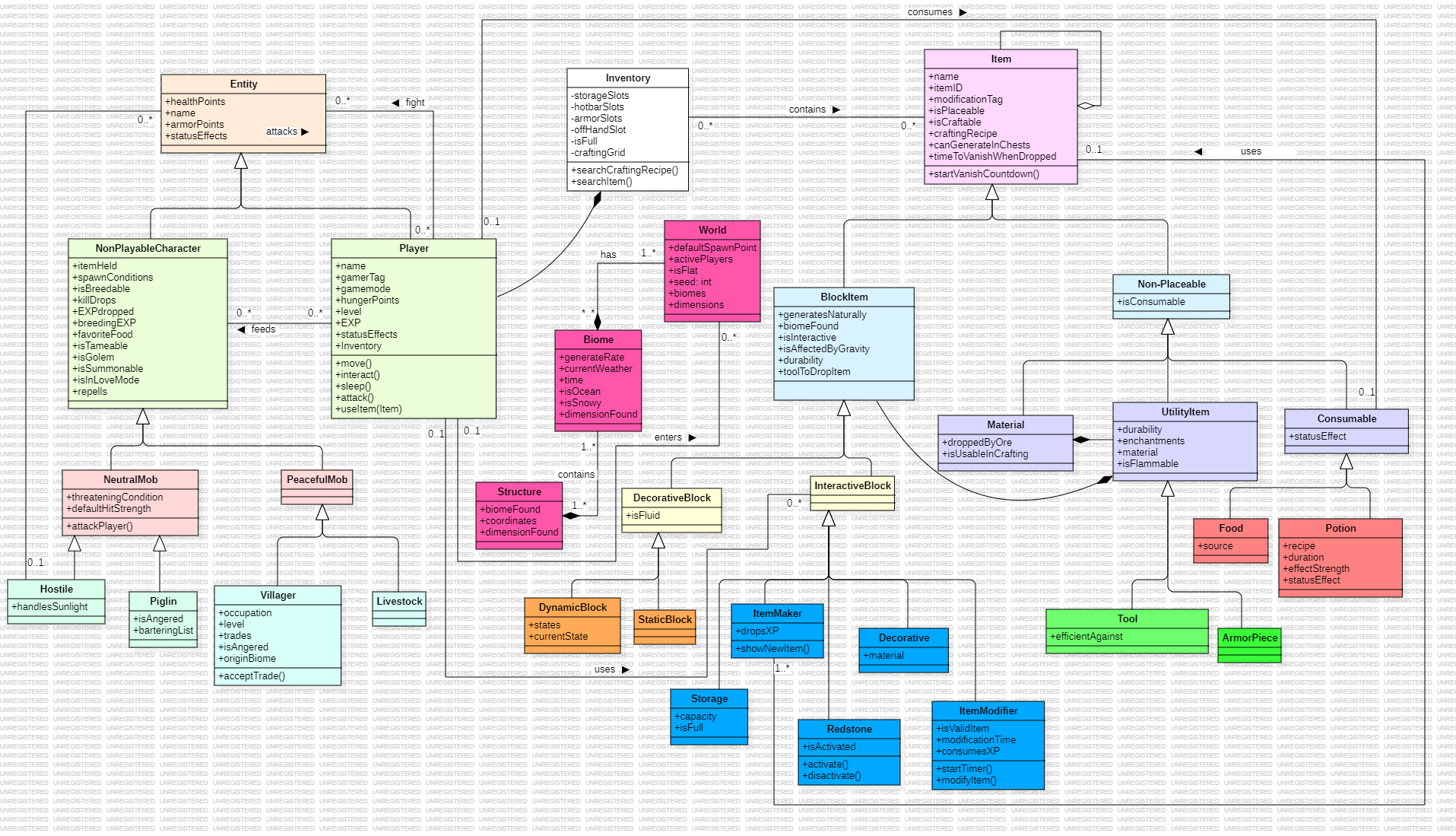
### Taming



### Breeding

# **Section 4: Structural Design**

## **Class Diagram**

****

## **Class Responsibility Collaboration Cards**

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name:** Player | **ID: 1** | | **Type:** Concrete, Domain |
| **Description:** The user manifested in a world. In creative mode, the player is an all-powerful being. In survival mode, the player has to fend for his own life against the dangers of this world | | | **Associated Use Cases: 7** |
| **Responsibilities**  Mine blocks  Collect items  Craft new items  Enhance / Enchant items  Attack mobs  Trade with villagers  Barter with Piglins  Consume food or potions  Tame and breed animals | | **Collaborators**  BlockItem, Tool  Item, Inventory  Item, ItemMaker  ItemModifier Tool, NonPlayableCharacter  Villager  Piglin  Consumable  Consumable, NeutralMob, PeacefulMob | |
| **Attributes:**   * healthPoints (double) * name (String) * armorPoints (double) * statusEffects(StatusEffect[]) * Inventory (Inventory) | | * gamertag (String) * gameMode (GameMode) * hungerPoints (int) * level (int) * EXP (int) | |
| **Relationships:**  **Generalization:** Entity  **Composition:** Inventory  **Other Associations:** World, Entity, Item, Consumable, InteractiveBlock | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Class Name:** Item | | **ID: 2** | | | | **Type:** Abstract, Domain |
| **Description:** The building block of the Minecraft world | | | | | | **Associated Use Cases: 7** |
| **Responsibilities**  Used for world generation  Used for crafting other items  Used as currency for trading  Despawns when left for 5 minutes | | | | **Collaborators**  World  Player, ItemMaker  Player, Villager | | |
| **Attributes:**   * name (String) * itemID (int) * modificationTag (int) * isBlock (Boolean) | | | | * craftingRecipe (Item[][]) * canGenerateInChests (Boolean) * despawnTimer (seconds) * isCraftable (Boolean) | | |
| **Relationships:**  **Generalization:**  **Composition:** Item (Aggregation)  **Other Associations:** ItemMaker, Inventory | | | | | | |
| **Class Name:** NonPlayableCharacter | **ID:** 3 | | | | | **Type:**  Abstract, Domain |
| **Description:** Non-sentient entities that spawn all over the World and are controlled by an algorithm. | | | | | | **Associated Use Cases:**  5 |
| **Responsibilities**  Move around the World  Interact with the Player  Interact with other NonPlayableCharacters  Roam structures and/or biomes | | | | | **Collaborators**  World  Player  NonPlayableCharacter  Structure, Biome | |
| **Attributes:**   * itemHeld (Item) * spawnConditions * isGolem (boolean) * isSummonable (boolean) * EXPDropped (int) * killDrops (Item[]) * repells (NonPlayableCharacter[]) | | | * isBreedable (Boolean) * breedingEXP (int) * favoriteFood (Consumable) * isTameable (Boolean) * isInLoveMode (Boolean) | | | |
| **Relationships:**  **Generalization:** Entity  **Aggregation:** Biome  **Other associations:** Item, NonPlayableCharacter | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name:** Villager | **ID:** 4 | | **Type:** Concrete, Domain |
| **Description:** An entity that spawns and works in villages and interacts with the Player | | | **Associated Use Cases:** 1 |
| **Responsibilities**  Do specific work in the village  Trade with the Player  Reproduce with other villagers  Hide in houses at night or when raided | | **Collaborators**  Player, Item  Villager  HostileMob | |
| **Attributes:**   * occupation (String) * level (int) * isAngered (boolean) * tradeList (Item[][]) * hasDiscounts (boolean) * originBiome (Biome) | | | |
| **Relationships:**  **Generalization:** PeacefulMob  **Aggregation:** Biome  **Other associations:** Item | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name:** Potion | **ID: 5** | | **Type:** Concrete, Domain |
| **Description:** A category of items that the player can ingest. Can either give buffs or debuffs for the player | | | **Associated Use Cases: 0** |
| **Responsibilities**  Give player status effect when drunk  Apply status effect to arrow when infused with it  Give entity status effect when thrown at it | | **Collaborators**  Player  Tool  Entity | |
| **Attributes:**   * name (String) * recipe (Item[]) * duration (seconds) * effectStrength(int) * statusEffect (StatusEffect) | | | |
| **Relationships:**  **Generalization:** Non-Placeable, Item  **Composition:**  **Other Associations:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name:** Structure | **ID:** 6 | | **Type:** Concrete, Domain |
| **Description:** A combination of blocks and items that spawn all around the World in specific biomes and that can be explored by the Player | | | **Associated Use Cases:** 1 |
| **Responsibilities**  Become a home to various entities  Possess loot specific to the Structure  Open a door to other Dimensions  Adapt loot according to its origin biome | | **Collaborators**  Entity  Item  Player, Dimension  Item, Biome | |
| **Attributes:**   * biomeFound (Biome) * lootContainers (Storage[]) * coordinates * structureSpecificLoot (Item[]) * dimensionFound (Dimension) | | | |
| **Relationships:**  **Generalization:**  **Aggregation:** Item, Storage  **Other associations:** Biome, Dimension | | | |

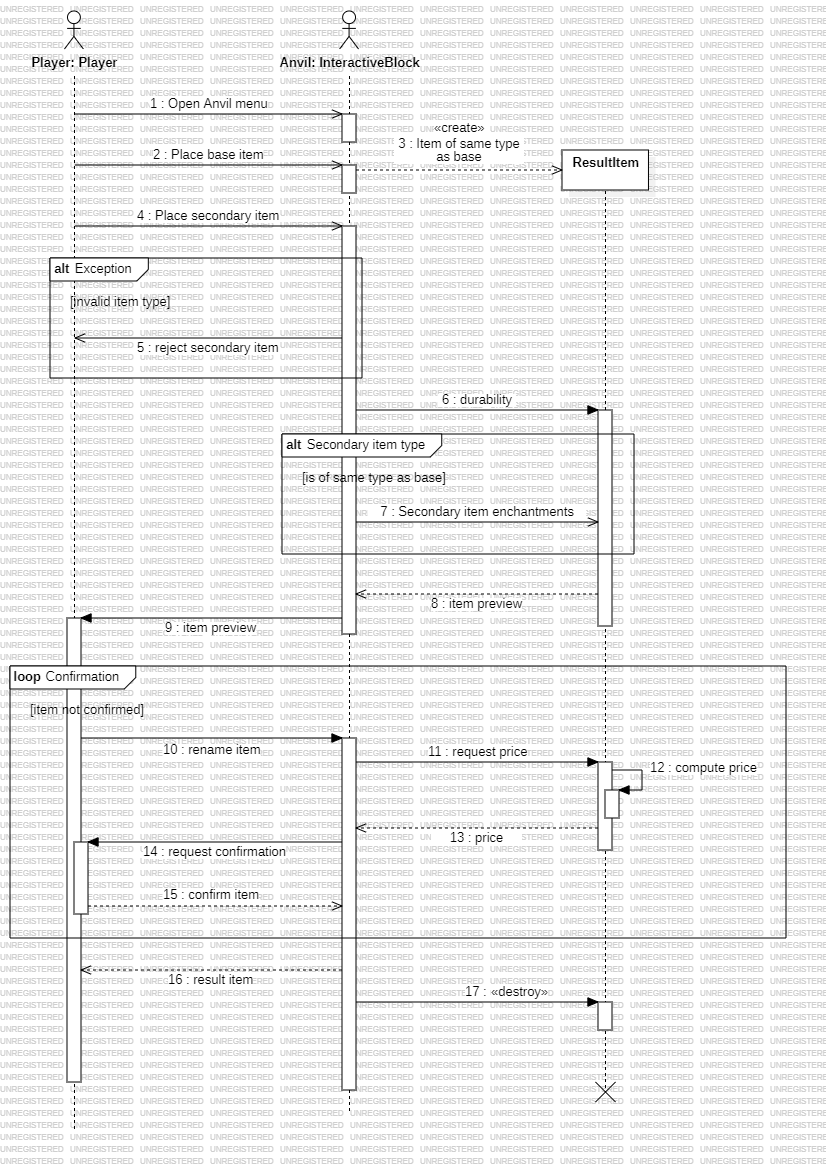
|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name:** ItemModifier | **ID:** 7 | | **Type:** Concrete, Domain |
| **Description:** An interactive block item that takes specific items from the Player and returns the altered item. | | | **Associated Use Cases:** 2 |
| **Responsibilities**  Take item(s)  Apply specific change  Return modified item | | **Collaborators**  Player, Item  Player, Item | |
| **Attributes:**   * consumesXP (boolean) * modificationTime (seconds) * isValidItem (boolean) | | | |
| **Relationships:**  **Generalization:** InteractiveBlock  **Aggregation:**  **Other associations:** Item, Player | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name:** Piglin | **ID:** 8 | | **Type:** Concrete, Domain |
| **Description:** A neutral mob that spawns in the Nether, and has a special affinity for Gold | | | **Associated Use Cases:** 1 |
| **Responsibilities**  Form gangs  Attack player not wearing gold  Barter with players for gold  Attack Hoglins (another nether mob) | | **Collaborators**  Piglin  Player, Material  Player, Material  HostileMob | |
| **Attributes:**   * isAngered (boolean) * barteringList (Item[][]) | | | |
| **Relationships:**  **Generalization:** NeutralMob  **Aggregation:** Biome  **Other associations:** Item | | | |

# **Section 5: Behavioral Design**

## **Interaction Diagrams**

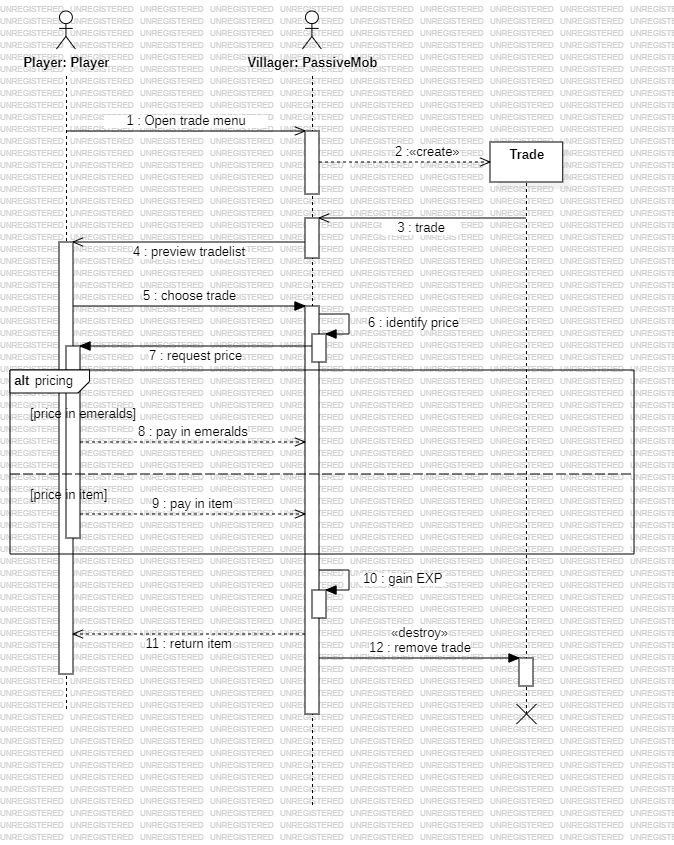
### Anvil Enhancement



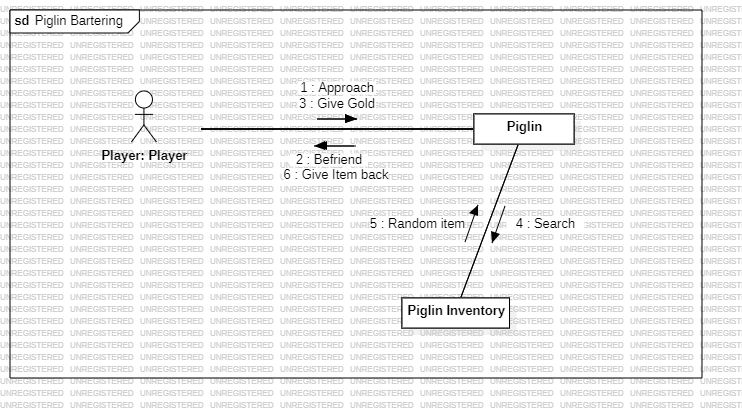
### Enchanting Table



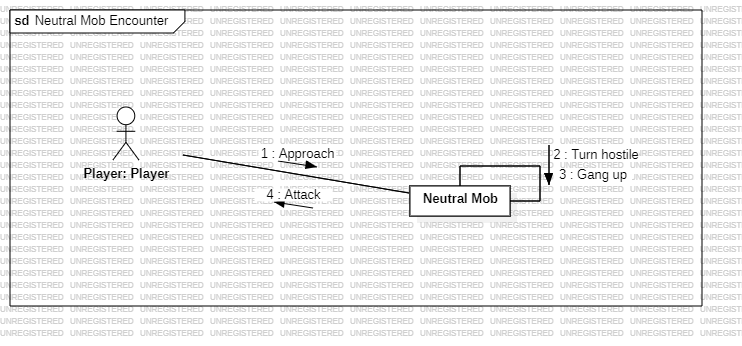
### Villager Trading



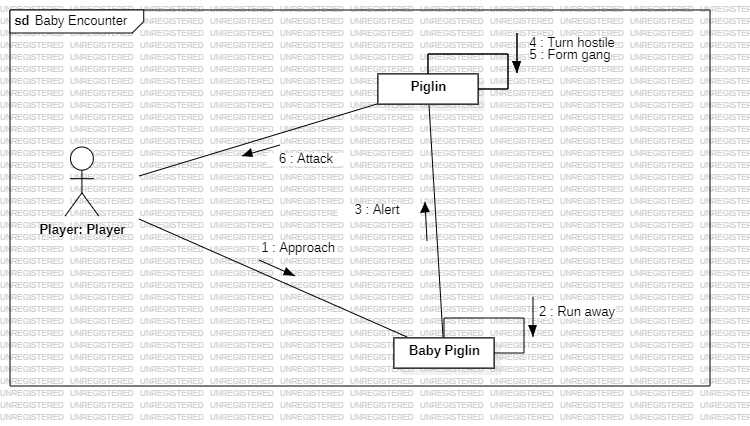
### Piglin Bartering



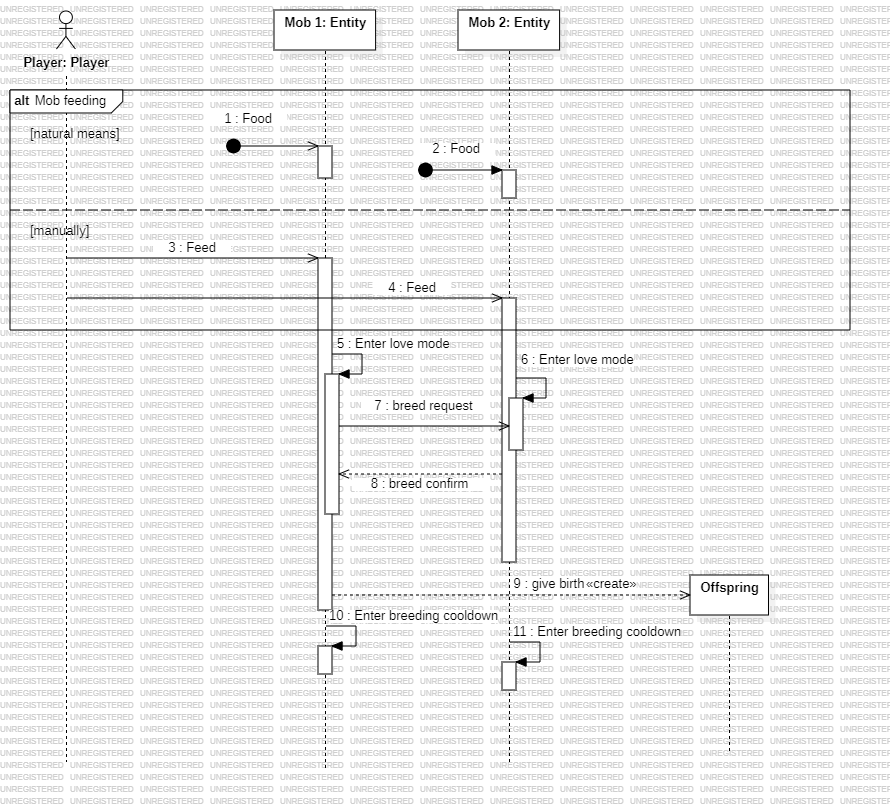
### Neutral Mob Encounter



### Piglin Hostile Encounter



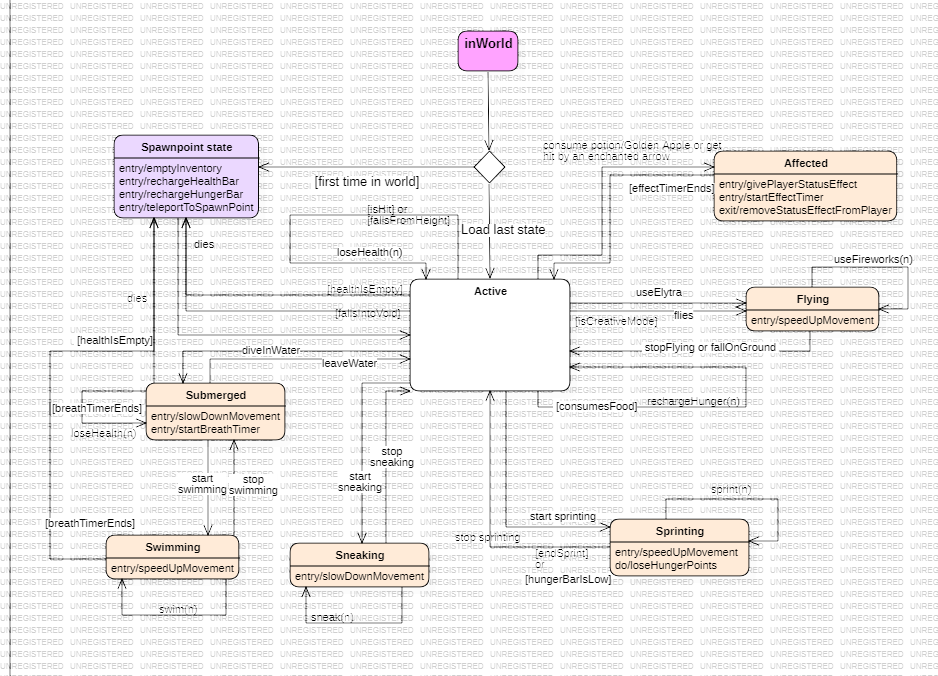
### Mob Breeding



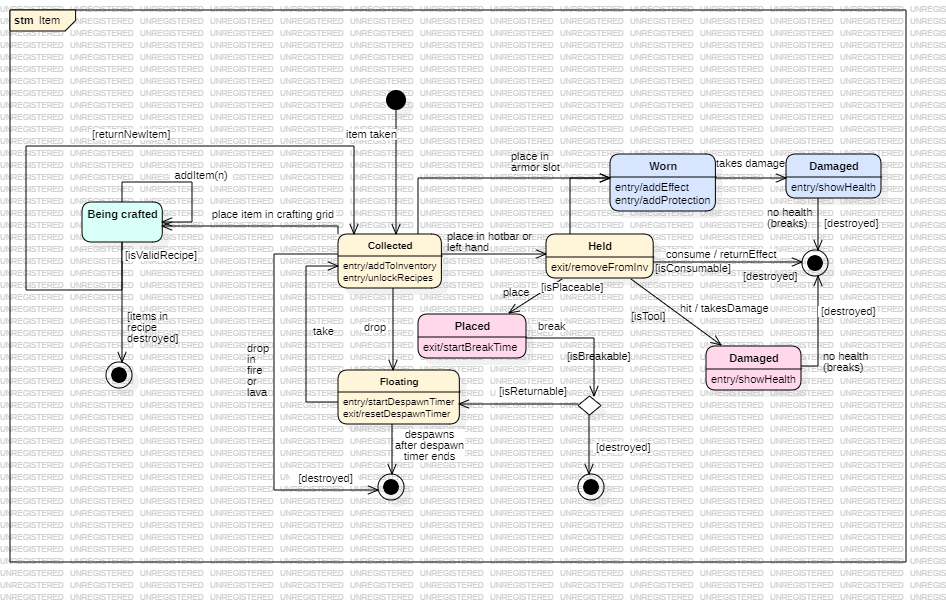
## **State-Chart Diagrams**

### Player[[6]](#footnote-6)

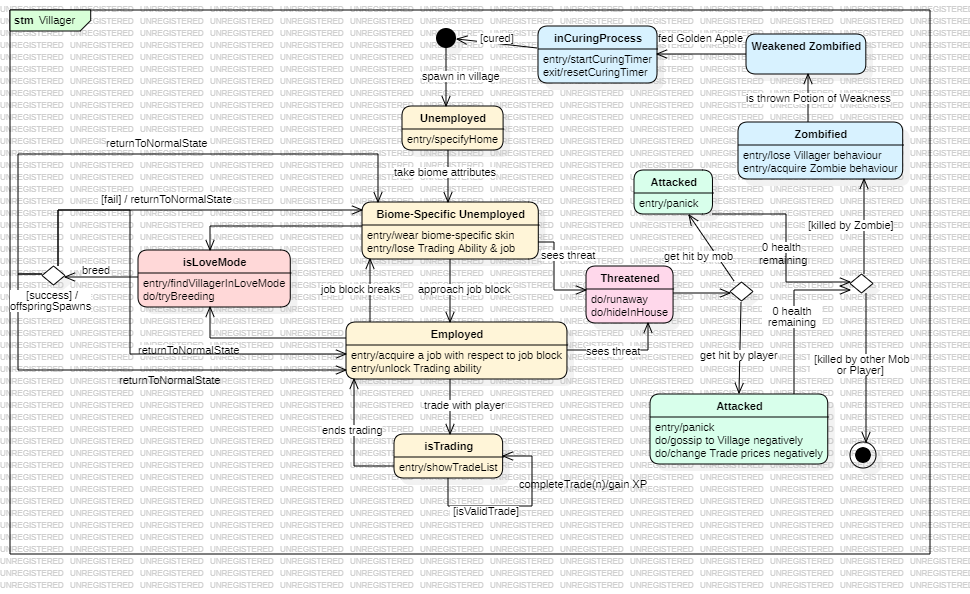




### Item

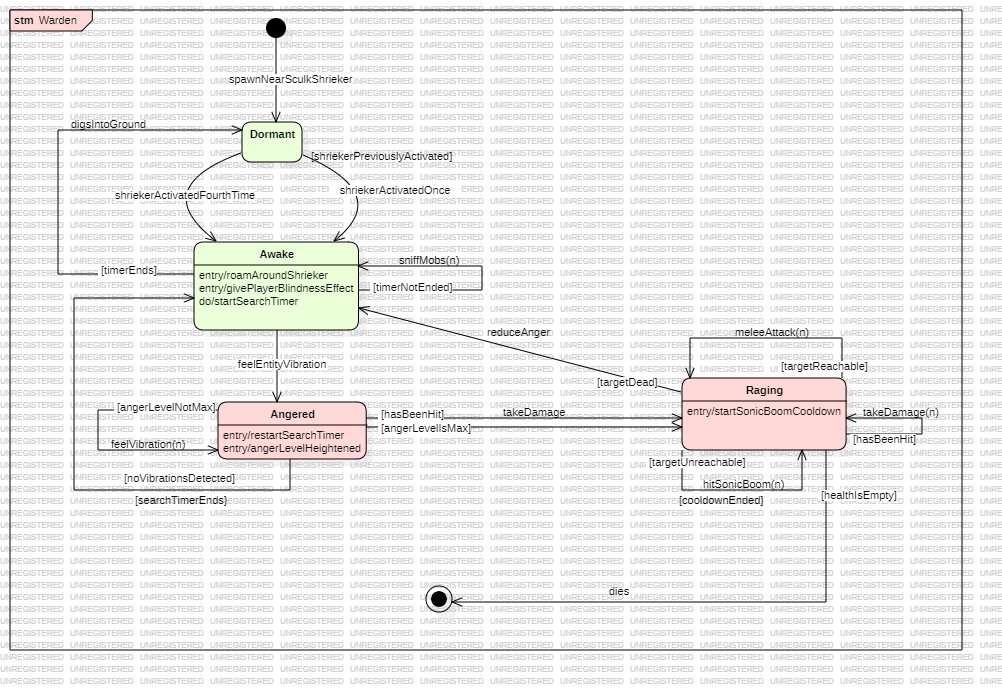


### Villager

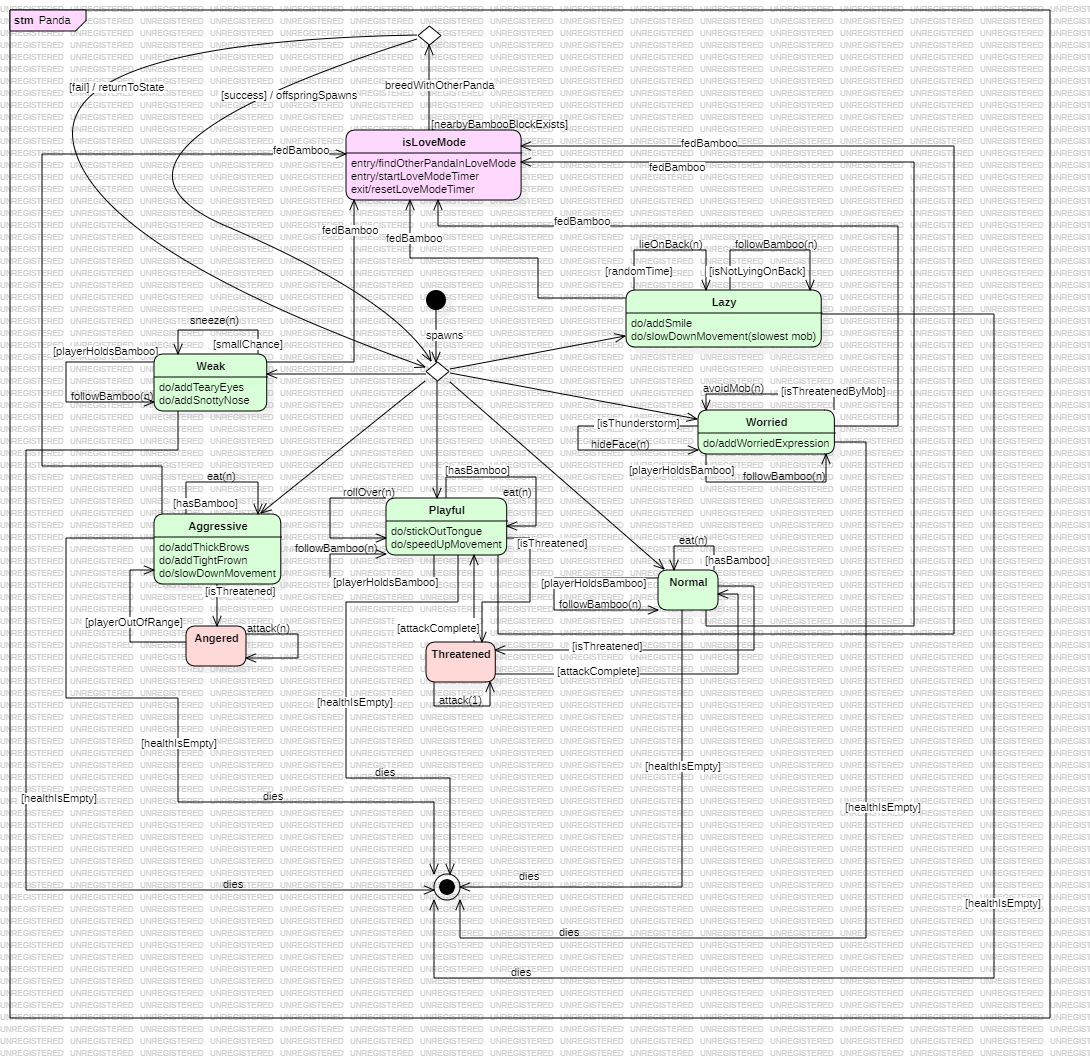


### Piglin

### Warden [[7]](#footnote-7)



### Panda[[8]](#footnote-8)



## **CRUDE Matrix**



This page has been intentionally left empty

1. Note that the game does not end here, only the main goal. The player can still access the world and can even summon the Ender Dragon a second time to defeat it again. [↑](#footnote-ref-1)
2. A chunk is a procedurally generated 16 x 16 segment of the world that extends all the way down to the lower world border up to a height of 256 blocks. Chunk render distance is the radius of visible chunks at a time. [↑](#footnote-ref-2)
3. The simulation distance is the number of chunks generated at a time in which game mechanics still apply. Simulation distance can differ from render distance. [↑](#footnote-ref-3)
4. Snapshots are a beta version of an upcoming update. They serve testing purposes for users and game developers to test new added features and bugs. [↑](#footnote-ref-4)
5. A seed is generated by the game engine when a world finishes rendering. Therefore, each unique world generation has a unique seed. Players can enter a seed and access the world associated with that seed. [↑](#footnote-ref-5)
6. The first state chart refers to the Player’s states prior to entering the world. The second refers to Player’s state when he is active inside a world. [↑](#footnote-ref-6)
7. A hostile mob encountered in the Deep Dark biome. It is blind yet highly sensitive to sounds and attacks anything when angered. The Deep Dark biome contains a specific spawn called the Sculk Shrieker that detects sounds and awakens the Warden [↑](#footnote-ref-7)
8. A passive mob encountered in the Jungle biome. [↑](#footnote-ref-8)