Name\_\_Asim Sapkota Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

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

This document details the design and implementation of combat mechanics for the game I am developing. The core focus of the project revolves around creating an immersive and dynamic combat experience, integrating core movement mechanics, hit detection systems to deliver engaging player interactions. I will be focused on activities like core Movement (jump, crouch, dash, kick, roll & kick, etc), hit detection & collision logic and combos & special move mechanics which will make the game more interesting and involving.

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

Use case diagram A diagram of a combat mechanism

AI-generated content may be incorrect.

### **Use Case Diagrams**

### **Scenarios**

**Scenario for Combat mechanics**

**Name:** Gameplay and combat mechanics

**Summary:** The player engages in combat using a combination of core movement and attack/defend mechanics. The player can jump, crouch, and punch during combat while defending with block or roll.

**Actors:** Players.

**Preconditions:** The game is on, and the player is within the combat area with an enemy.

**Basic sequence:**

**Step 1:** The player enters combat mode and can perform movement actions (e.g., Jump, Crouch, Dash).

**Step 2:** The player selects either an attack or a defense move (e.g., Punch, Kick, Block, Roll).

**Step 3:** If the player attacks, the attack move is executed, and the hit detection logic checks whether the attack lands on the enemy. If the attack hits, the damage is calculated, and the enemy’s health is reduced.

**Step 4:** If the player defends (e.g., Block or Roll), the defense is executed. The damage is either mitigated or completely avoided based on the type of defense move.

**Step 5 :** The player continues to fight until one of the combatants (player or enemy) loses all health, or the battle objectives(getting the pineapple) are achieved.

**Exceptions:**

**Step 1:** If the player is not in combat mode, combat actions are not available, and the player must enter combat mode/ final level.

**Step 2:** If the player is knocked down or incapacitated, they temporarily cannot perform any attack or defense action.

**Post conditions:** The player's actions (movement, attack, or defense) are executed according to the sequence.

**Priority:** 1\*

**ID:** AS01

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

**Scenario for Attack**

**Name:** Perform Attack  
**Summary:** The player performs an attack action (such as Punch, Kick, Jump attack) to damage the enemy.  
**Actors:** Player  
**Preconditions:**

The enemy is within range of the player’s attack.

**Basic sequence:**

**Step 1:** The player selects an attack move (e.g., Punch, Kick).

**Step 2:** The player presses the corresponding attack button.

**Step 3:** The attack move is executed, and the player’s character performs the animation for the selected move.

**Step 4:** The attack hits the enemy if the player is within the required range and no defense actions (like a block or roll) are in place from the enemy.

**Step 5:** The damage is calculated based on the collision and player stats.

**Step 6:** The enemy takes damage, and the health bar is updated.

**Step 7:** The attack animation completes, and the player can continue attacking or defending.

**Exceptions:**

**Step 1:** If the player is out of range, the attack does not land.

**Step 2:** If the wrong button is pressed, no attack is performed, and the player may perform a different move (e.g., a movement action).

**Step 3:** If the attack is blocked by the enemy or a defense action is in place, the attack does minimal or no damage.

**Step 4:** If the player’s energy is too low, the attack cannot be executed.

**Post conditions:**

The enemy’s health is reduced by the calculated damage.

The attack animation finishes, and the player has the option to continue the next action.

**Priority: 1  
ID:** AS01

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

**Scenario for Defend**

**Name:** Perform Defend  
**Summary:** The player performs a defensive action (such as Block or Roll) to reduce or avoid incoming enemy attacks.  
**Actors:** Player  
**Preconditions:**

The enemy is in the process of attacking.

**Basic sequence:**

**Step 1:** The player selects a defense action (e.g., Block or Roll).

**Step 2:** The player presses the corresponding button for defense.

**Step 3:** The defense action is executed, and the player’s character performs the animation for the selected move.

**Step 4:** If the player blocks, incoming damage is reduced based on the defense value of the block. If the player rolls, the attack is evaded entirely if timed correctly.

**Step 5:** The player’s health is updated, with damage reduced or avoided if the defense was successful.

**Step 6:** The defense action finishes, and the player can resume attacking or moving.

**Exceptions:**

**Step 1:** If the player does not select a defense action, no defense is performed, and the player takes full damage from the incoming attack.

**Step 2:** If the defense button is pressed or the attack is avoided too early or too late, the player may not properly block or evade the attack.

**Step 3:** If the player is in the middle of another action (e.g., attacking), the defense action may be canceled or delayed.

**Step 4:** If the player is in the air or stunned, no defense action can be executed.

**Post conditions:**

If the player successfully blocks, incoming damage is reduced according to the defense mechanics.

If the player successfully rolls, the incoming attack is completely avoided.

The player can continue with their next action (attack or movement) once the defense animation finishes.

**Priority:** 1  
**ID:** D01

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

**Scenario for Hit Detection (Include Use Case)**

**Name:** Hit Detection  
**Summary:** The system checks if an attack has successfully hit the enemy, determining whether the attack lands within the correct hit box.  
**Actors:** Game System

**Preconditions:**

The player has performed an attack action (e.g., Punch, Kick).

The enemy is within the attack range and in combat mode.

**Basic sequence:**

**Step 1:** The player performs an attack move (e.g., Punch, Kick).

**Step 2:** The system checks the distance and position between the player and the enemy.

**Step 3:** The system checks the hit box of the player’s attack against the enemy’s hit box.

**Step 4:** If the hit box of the attack intersects with the enemy’s hit box, the attack is considered a successful hit.

**Step 5:** If the attack successfully hits, the damage calculation step is triggered.

**Step 6:** If the attack misses (i.e., no intersection between the hit boxes), the attack is considered a miss, and no damage is inflicted.

**Exceptions:**

**Step 1:** If the enemy is out of range of the attack, the attack does not hit, and a message is displayed indicating the attack missed.

**Step 2:** If the enemy is blocking or rolling, the hit detection may register as no damage or partial damage.

**Post conditions:**

The system identifies whether the attack hit the enemy or missed.

If a hit occurs, the system proceeds to the damage calculation step.

If the attack misses, the player’s turn ends, and the enemy may retaliate.

**Priority:** 1  
**ID:** AS01

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

**Scenario for Damage Calculation (Include Use Case)**

**Name:** Damage Calculation  
**Summary:** The system calculates the damage inflicted on the enemy when an attack lands successfully, factoring in the type of attack, player stats, and any special conditions (e.g., critical hit).  
**Actors:** Player, Enemy (Antagonist)  
**Preconditions:**

The player’s attack has successfully hit the enemy (as verified by hit detection).

The enemy has health remaining and is still in combat mode.

**Basic sequence:**

**Step 1:** The hit detection confirms the attack has landed.

**Step 2:** The system retrieves the attack type (e.g., Punch, Kick, Special Move) and the player’s stats (e.g., Attack Power, Critical Hit chance).

**Step 3:** The system checks if the attack was a critical hit based on the player’s stats and luck chance. If a critical hit occurs, the damage is increased.

**Step 4:** The system calculates the base damage for the attack using the attack type and player stats.

**Step 5:** The system factors in any additional modifiers such as critical hits, combos, or buffs.

**Step 6:** The calculated damage is subtracted from the enemy’s health.

**Step 7:** The damage value is displayed in the form of a hit number (e.g., “Enemy hit for 15 damage!”).

**Exceptions:**

**Step 1:** If the player’s attack power is too low, the damage calculation may result in minimal or no damage.

**Step 2:** If the enemy has a defense modifier, the damage will be reduced based on the defense value.

**Step 3:** If the enemy’s health is reduced to 0 or below, the enemy is defeated, and the battle ends.

**Post conditions:**

The enemy’s health is updated based on the calculated damage.

The attack damage is displayed, and the player can proceed with the next action (attack, defend, or movement).

If the enemy is defeated, the battle ends, and the player progresses to the next phase.

**Priority:** 1  
**ID:** DC01

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

**Scenario for Critical Hit (Extend Use Case)**

**Name:** Critical Hit  
**Summary:** The player’s attack has the chance to land a critical hit, dealing additional damage based on certain conditions such as player stats or luck.  
**Actors:** Player  
**Preconditions:**

The player has performed an attack move that is capable of landing a critical hit (e.g., Punch, Kick, Special Move).

The player has a critical hit chance based on their stats (e.g., Luck, Critical Hit Rate).

**Basic sequence:**

**Step 1:** The player performs an attack move in the critical body parts of the enemy.

**Step 2:** The system checks the player's critical hit chance using player stats.

**Step 3:** If the system determines a critical hit based on player stats, the damage value is multiplied by a critical hit factor (e.g., 2x or 3x).

**Step 4**: The attack damage is updated to reflect the critical hit value.

**Step 5:** The enemy takes the increased damage based on the critical hit value.

**Step 6:** The critical hit is displayed as part of the attack animation or as a message (“Critical Hit!”).

**Exceptions:**

**Step 1:** If the player’s critical hit chance is too low, no critical hit occurs, and the damage calculation proceeds normally.

**Step 1:** If the player’s critical hit chance is temporarily disabled (e.g., a debuff), the critical hit does not occur.

**Step 3:** If the player’s attack power is too low, the critical hit may still not result in a significant damage boost.

**Post conditions:**

The enemy takes the increased damage due to the critical hit.

The attack is logged with the critical hit factor, and the player can proceed with the next action.

If the enemy’s health reaches 0, the enemy is defeated, and the battle progresses.

**Priority:** 2  
**ID:** CH01

\*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**

**Data Flow Diagrams**

A diagram of a game

AI-generated content may be incorrect.

### **Process Descriptions**

Process Description for Gameplay and Combat Mechanics

**1. Player Input Handling**

**Trigger:** The player provides input via keyboard, controller, or touch interface.

**Process:**

The system captures player input (e.g., movement, attack, defense).

The input is processed to determine the action (jump, crouch, attack, block, etc.).

If the input corresponds to a valid combat action, it is passed to the respective combat mechanics module.

**Output:** The player's action is executed, triggering animations and system calculations.

**2. Movement System**

**Trigger:** The player presses a movement key (jump, crouch, dash, roll).

**Process:**

The system checks if movement is allowed (e.g., player is not stunned).

The movement is executed based on the player's direction and input.

If applicable, special movement animations or physics (e.g., momentum for dashes) are applied.

**Output:** The player moves in the game environment.

**3. Attack System**

**Trigger:** The player presses an attack button.

**Process:**

The system verifies that the player is in combat mode.

The attack animation is triggered.

The hit detection module is activated to check if the attack lands.

If successful, the damage calculation module processes the inflicted damage.

If the attack is part of a combo sequence, the combo module is triggered.

**Output:** The enemy receives damage if the attack lands, and the player's action is recorded.

**4. Defense System**

**Trigger:** The player presses a block or roll button.

**Process:**

The system checks if the player is in combat mode.

The defense animation is triggered.

If blocking, the system calculates damage reduction.

If rolling, the system determines if the evasion is successful.

**Output:** The player's defense action is executed, and damage is mitigated or avoided.

**5. Hit Detection System**

**Trigger:** An attack is performed.

**Process:**

The system retrieves the attack's hitbox coordinates.

The system retrieves the enemy’s hitbox coordinates.

A check is performed to determine if the hitboxes intersect.

If an intersection occurs, the attack is registered as a hit.

If no intersection occurs, the attack is considered a miss.

**Output:** The attack either lands and proceeds to damage calculation or misses.

**6. Damage Calculation System**

**Trigger:** An attack successfully lands.

**Process:**

The system retrieves attack type and player stats.

The system checks if any modifiers apply (e.g., critical hit, buffs).

The base damage is calculated based on attack strength and enemy defense.

The final damage is subtracted from the enemy’s health.

**Output:** The enemy’s health is updated, and the result is displayed on-screen.

**7. Critical Hit System**

**Trigger:** A critical hit chance is determined during damage calculation.

**Process:**

The system retrieves the player’s critical hit rate.

A probability check determines if a critical hit occurs.

If successful, the damage multiplier is applied.

**Output:** The attack deals extra damage, and a visual effect or text appears.

**8. Health and Battle State Management**

**Trigger:** Player or enemy receives damage.

**Process:**

The system updates the health bar based on damage received.

If health reaches zero, the character is defeated.

If the enemy is defeated, the battle concludes, triggering the win/loss state.

**Output:** The battle progresses or ends based on health status.

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

**Acceptance Test for Combat Mechanics**

**Test Case: Entering Combat/final Mode**

**Test ID:**

**Description:** Ensure the player can enter final level when an enemy/witch is present.

**Preconditions:** The player is in the game world, and a witch is nearby.

**Test Steps:**

Move the player near an enemy.

Press the combat initiation button.

**Expected Result:** The player enters final level, and combat mechanics become available.

**Pass Criteria:** The player successfully enters combat mode, and movement/attack options appear.

**Test Case: Performing Attack**

**Test ID:**

**Description:** Verify that the player can perform an attack.

**Preconditions:** The player is in combat mode and within attack range of an enemy.

**Test Steps:**

Press the attack button (e.g., Punch, Kick).

Observe if the attack animation plays.

Check if hit detection registers the attack.

**Expected Result:** The attack is executed, and if it lands, the enemy takes damage.

**Pass Criteria:** The attack animation plays, hit detection registers the hit, and enemy health decreases if hit is successful.

**Test Case: Performing Defense (Block/Roll)**

**Test ID:**

**Description:** Ensure the player can perform a defensive action.

**Preconditions:** The player is in combat mode, and the enemy is attacking.

**Test Steps:**

Press the defense button (Block/Roll) as the enemy attacks.

Observe if the defense animation plays.

Check if damage is reduced (Block) or avoided (Roll).

**Expected Result:** The player successfully blocks (reducing damage) or rolls (evading the attack).

**Pass Criteria:** Defense animation plays, and damage is mitigated or avoided based on defense type.

**Test Case: Hit Detection**

**Test ID:**

**Description:** Validate that hit detection correctly registers hits and misses.

**Preconditions:** The player is in combat mode, and an attack is performed.

**Test Steps:**

Execute an attack while within range of the enemy.

Execute an attack while outside the range of the enemy.

Observe the hit/miss detection result.

**Expected Result:** The system correctly registers hits when the attack is within range and misses when the attack is out of range.

**Pass Criteria:** The attack only lands if within range, and misses if out of range.

**Test Case: Damage Calculation**

**Test ID:**

**Description:** Verify that damage calculation is accurate.

**Preconditions:** A successful hit has been detected.

**Test Steps:**

Perform an attack and check the enemy’s initial health.

Check the attack damage value.

Verify that the enemy’s health decreases correctly.

**Expected Result:** The enemy’s health decreases based on the damage calculation formula.

**Pass Criteria:** The enemy’s health updates correctly after each successful attack.

**Table for critical Hit activation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Output** | **Player Action** | **Notes** | **Expected Result** |
| Attack Executed | Perform an attack with a critical hit chance. | The attack is initiated and has a chance to be a critical hit. | Attack animation plays, and hit detection is triggered. |
| Critical Hit Check | System determines if an attack is critical. | The game runs a probability check to determine a critical hit. | Critical hit triggers occasionally based on probability. |
| Damage Calculation | Verify if critical hit damage is applied. | If a critical hit is triggered, the damage is multiplied. | Critical hit deals increased damage. |
| UI Feedback | Observe damage numbers on screen. | Critical hits should display a unique color or effect. | UI shows the boosted damage with a special indicator. |
| Enemy Reaction | Enemy takes damage from the critical hit. | Enemy’s health decreases based on the critical hit damage value. | Enemy's HP reduces according to the critical hit multiplier. |

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

### **Work items**

|  |  |  |  |
| --- | --- | --- | --- |
| **Work Items** | **Task** | **Duration (PWks)** | **Predecessor Task(s)** |
| 1. Core Movement System | Jump, Crouch, Dash, Roll, Kick | 2 | - |
| 2. Hit Detection & Collision | Implement and refine collision detection | 3 | 1 |
| 3. Combat Basics | Basic attack mechanics, blocking, dodging | 3 | 1 |
| 4. Tactical Item Mechanics | Implement power-ups, health recovery, equipment use | 2 | 2 |
| 5. AI Enemy Behavior | Design AI responses to attacks & movement | 1 | 3 |
| 6. Playtesting & Balancing | Adjust damage values, hitboxes, speed, etc. | 2 | 2, 4, 5 |
| 7. Final Integration | Ensure smooth integration of all mechanics | 2 | 6 |
| 8. Debugging and testing | Test the feature and debug the program in presence of any errors. | 1 | 7 |
| 9. Documentation | Document each part for easy understanding. | 1 | 8 |

### 

### **Pert diagram**

A diagram of a computer code

AI-generated content may be incorrect.

### 

### 

### 

### 

### **Gantt timeline**

| 1 | ⭐ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |