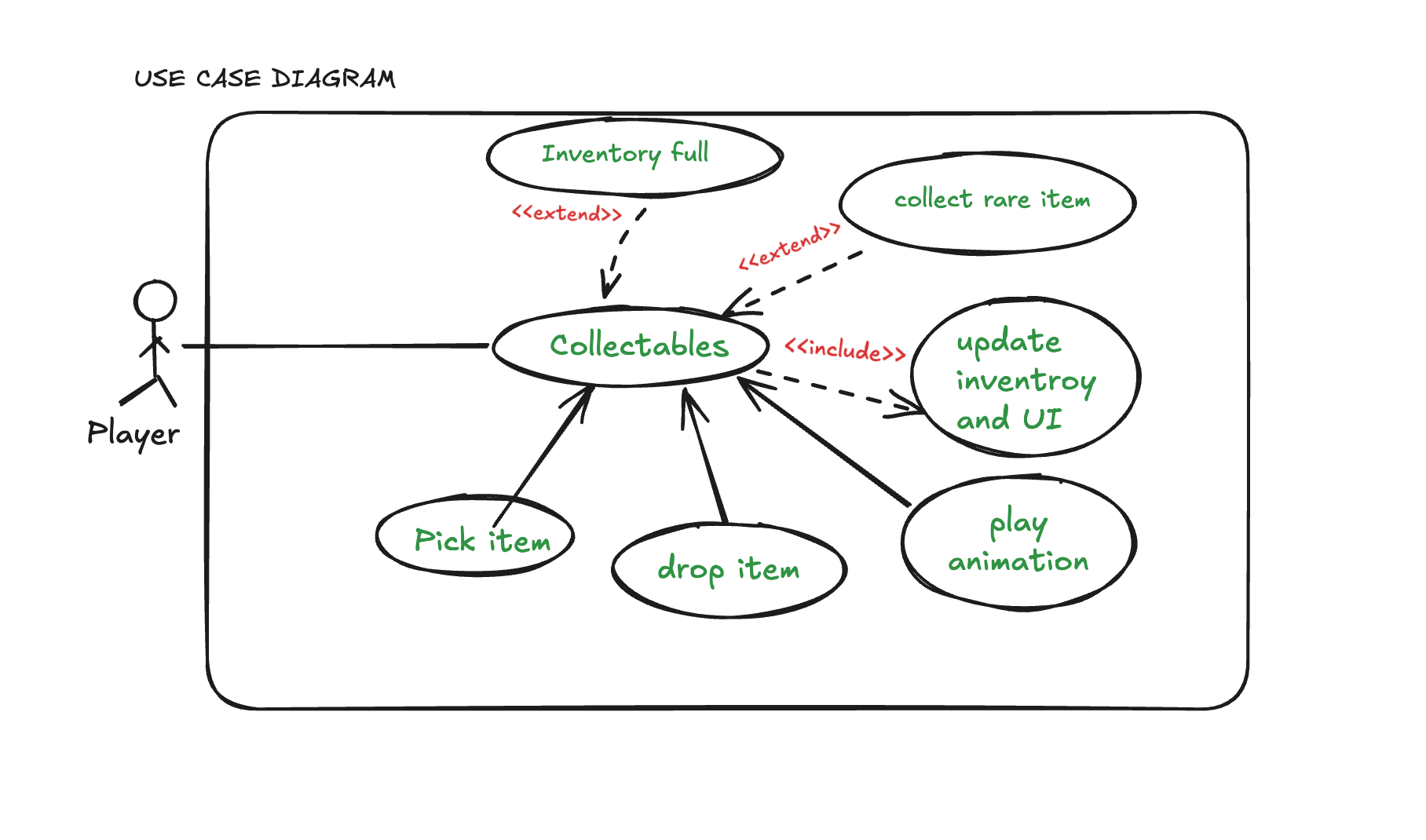
Name\_\_\_\_\_\_**Sikha Chaudhary**\_\_\_\_\_\_\_\_ Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

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

This document outlines the game development process for the collectible system in a combat-based game. In this game, I will be working on a feature called collectables. Players engage in one-on-one matches, and before each fight, they have a limited time to collect various items. These collectibles provide advantages such as health recovery, power-ups, and tactical equipment, influencing the outcome of battles. Some of the things that can be collected are : Blades, Crossbows, healing potions, Adrenaline shots and so on.

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



### **Use Case Diagrams**

### **Scenarios**

**Use case scenario for Collectables(Collect an item before the game starts).**

**Name:** Collectables

**Summary:** Players gather collectible items that offer strategic advantages before a fight begins.

**Actors:** Player

**Preconditions:** The match has been initialized, and the pre-fight collection phase is active.

**Basic sequence:**

**Step 1:** The game enters the pre-fight collection phase.

**Step 2:**The player moves around the environment to find collectibles.

**Step 3:** The player picks up an item.

**Step 4:** The game updates the player's inventory with the collected item.

**Step 5 :** The phase ends when time expires or the player chooses to proceed to battle.

**Exceptions:**

**Step 1:**If inventory is full, the player cannot pick up additional items..

**Step 2:**If time expires, the player is automatically moved to battle mode.

**Post conditions:** Picking up an item includes an animation and sound effect for better player feedback and if the player collects a rare item, a special event may trigger.

**Priority:** 1\*

**ID:** C01

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

**Use case scenario for Pick an item.**

**Name:** Pick an item

**Summary:** The player collects an item from the environment.

**Actors:** Player

**Preconditions:** The player is within range of a collectible item.

**Basic sequence:**

**Step 1:** The player moves towards the item.

**Step 2:**The player interacts with the item to pick it up.

**Step 3:** The game verifies if the inventory has space.

**Step 4:** If space is available, the item is added to inventory.

**Exceptions:**

**Step 1:**If inventory is full, the player cannot pick up additional items..

**Step 2:**If time expires, the player is automatically moved to battle mode.

**Post conditions:** Picking up an item includes an animation and sound effect for better player feedback and if the player collects a rare item, a special event may trigger.

**Priority:** 1\*

**ID:** C02

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

**Use case scenario for Drop an item.**

**Name:** Drop an Item  
**Summary:** The player drops an item from their inventory to free space.  
**Actors:** Player  
**Preconditions:** The player has at least one item in inventory.

**Basic Sequence:**

1. The player opens the inventory.
2. The player selects an item to drop.
3. The game removes the item from inventory.
4. The item is placed back into the game environment.
5. UI updates to reflect the inventory change.

**Exceptions:**

* **Step 1:** If the inventory is empty, the player cannot drop an item.

**Post Conditions:** The item is no longer in inventory and appears back in the game world.  
**Priority:** 1\*  
**ID:** C03

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

**Use case scenario for Update Inventory and UI.**

**Name:** Update inventory

**Summary:** The game updates the player's inventory and reflects those changes in UI after picking up an item.

**Actors:** Game System

**Preconditions:** The player has collected an item or dropped an item.

**Basic sequence:**

**Step 1:** The game tracks the collected/dropped items by the player.

**Step 2:** The game system executes the logic.

**Step 3:** The UI updates to reflect the new item.

**Exceptions:**

**Step 1:**If inventory is full, the item is not added.

**Post conditions:** inventory correctly reflects collected items.

**Priority:** 1\*

**ID:** C04

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

**Use case scenario for Play Animation**

**Name:** Play Animation

**Summary:** An animation is played when a player picks up an item.  
**Actors:** Game System.

**Preconditions:** The player has collected an item or dropped an item.

**Basic sequence:**

**Step 1:** The game triggers the pickup animation.

**Step 2:** A sound effect plays to confirm collection.

**Exceptions:**

**Step 1:**Only plays when an item is picked or dropped.

**Post conditions:** The player sees and hears feedback confirming the pickup.

**Priority:** 3\*

**ID:** C05

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

**Use case scenario for Collect Rare Item**

**Name:** Collect Rare Item  
**Summary:** A rare item is collected, triggering a special event.  
**Actors**: Player, Game System  
**Preconditions:** The player has collected a rare item.

**Basic Sequence:**

1. The game identifies the item as rare.
2. A special animation or effect plays.
3. The rare item's power-up is applied.

**Exceptions:**

**Step 1:**There will be few rate items to collect.

**Post Conditions:** The player gains a unique advantage.  
**Priority:** 3\*  
**ID:** C06

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

**Use case scenario for Inventory Full**

**Name:** Inventory Full  
**Summary:** The player attempts to collect an item but has no available inventory space.  
**Actors:** Player, Game System  
**Preconditions:** The player's inventory is at capacity.

**Basic Sequence:**

1. The player attempts to collect an item.
2. The game checks inventory capacity.
3. The game prevents the item from being picked up.
4. A message is displayed informing the player.

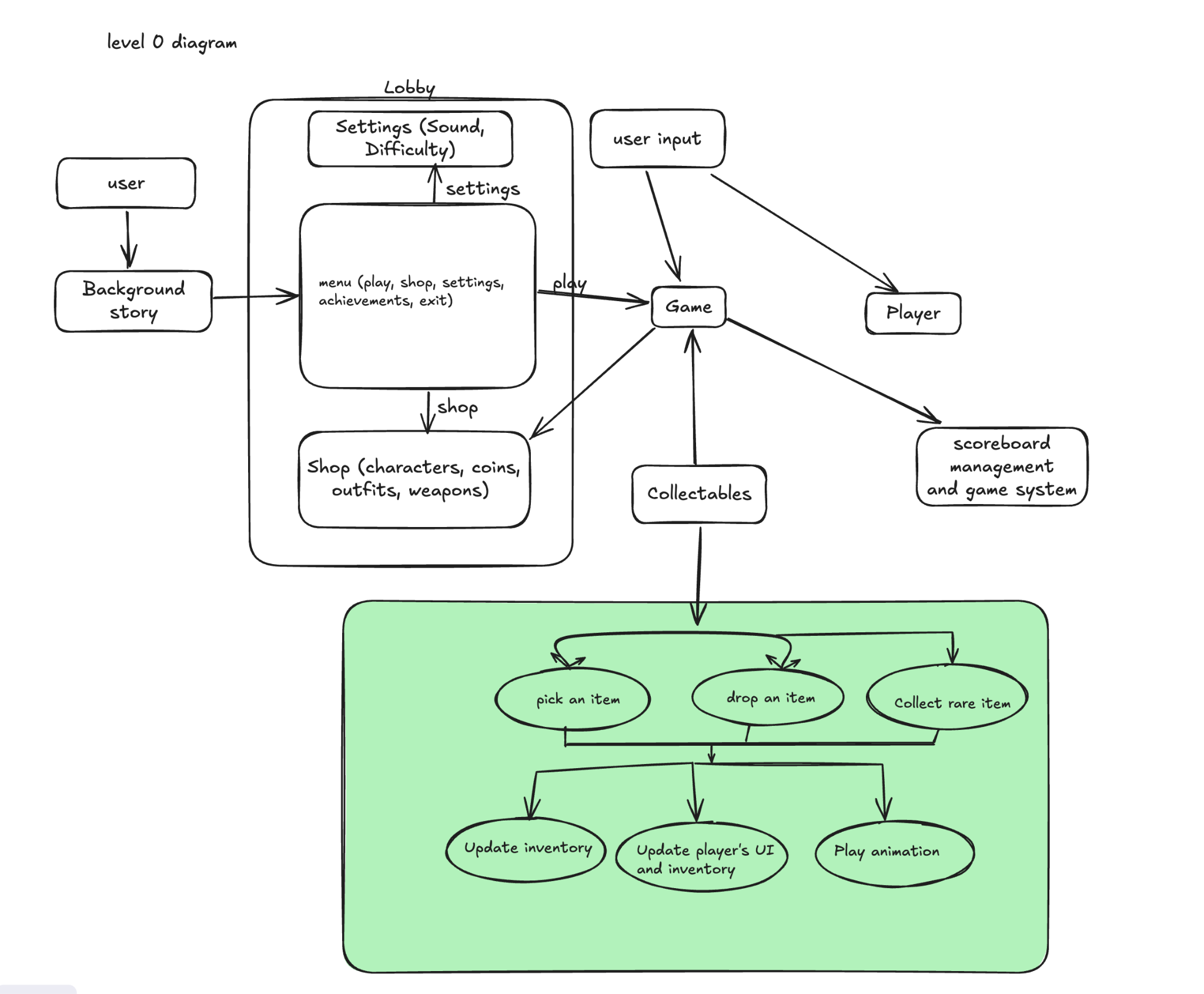
**Post Conditions:** The item is not added to inventory.  
**Priority:** 2\*  
**ID:** C07

\*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**

**Level 0 diagram**



### **Process Descriptions**

## 

**1. Start: Pre-fight Phase**

This is the phase where players can collect items before the battle.

* Decision 1: Is the pre-fight phase active?
  + **Yes :**  Proceed to Step 2 (Detect player movement).
  + **No :**  End process.

**2. Detect Player Movement**

The game detects whether the player is near any collectible items.

* Decision 2: Is the player near a collectible?
  + **Yes :** Proceed to Step 3 (Item Interaction).
  + **No :** Return to Step 2 (Continue detecting movement).

**3. Item Interaction**

The player attempts to collect an item.

* Decision 3: Does the player interact with a collectible item?
  + **Yes :**  Proceed to Step 4 (Check Inventory Space).
  + **No :**  Return to Step 2 (Continue detecting movement).

**4. Check Inventory Space**

The game checks if the player's inventory has space for the item.

* Decision 4: Is there space in the inventory?
  + **Yes :**  Proceed to Step 5 (Add Item to Inventory).
  + **No :**  Exception: Item is not collected. Proceed to Step 6 (Display Inventory Full Message).

**5. Add Item to Inventory and Update UI**

The item is added to the player’s inventory and updates the UI.

**Action :** Add item and Update the UI.

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

**Test Case 1: Collecting Items Successfully**

**Steps:**

1. Start the pre-fight phase.
2. Move player near a collectible.
3. Player collects the item.
4. Check if the item is added to inventory.
5. Verify UI updates.

**Expected Result:**

* Item added to inventory.
* UI reflects the new item.

**Test Case 2: Inventory Full**

**Steps:**

1. Start the pre-fight phase with a full inventory.
2. Move player near a collectible.
3. Player tries to collect it.
4. Verify the item isn't collected.
5. Ensure the "inventory full" message shows.

**Expected Result:**

* Item not added.
* "Inventory full" message displayed.

**Test Case 3: Time Expiry**

**Steps:**

1. Start the pre-fight phase.
2. Wait for time to expire without collecting items.
3. Verify transition to battle mode.

**Expected Result:**

* Game automatically transitions to battle mode after time expires.

**Example for Collecting item feature :**

| Output | Player Action | Notes | Expected Result |
| --- | --- | --- | --- |
| Pre-fight Phase Active | Start the pre-fight phase. | The game transitions into the pre-fight phase where the player can begin collecting items. | Game enters the pre-fight collection phase. |
| Item in Range | Move player near a collectible. | The player moves towards a collectible item that is within range to be picked up. | Collectible items should be visible and within range of the player for collection. |
| Collect Item | Player collects the item. | The player interacts with the collectible (e.g., by pressing a button or clicking on the item). | Items are successfully added to the player's inventory. |
| Inventory Updated | Check if the item is added to inventory. | After collecting, the game should check if there is space in the inventory, then add the item if space is available. | Items appear in the player’s inventory list. |
| UI Reflects Changes | Verify UI updates to show the collected item | The game's user interface should reflect the new item in the inventory panel, showing the correct item icon, name, or effect. | UI updates with the collected item displayed in the inventory panel. |
| Confirm Collection | Verify inventory and UI after collection. | Ensure the UI reflects all collected items. If there is a UI for the inventory, it should show the correct number and type of items collected. | The inventory UI shows the newly collected item, confirming the collection process is successful. |

**Example for Collecting item feature :**

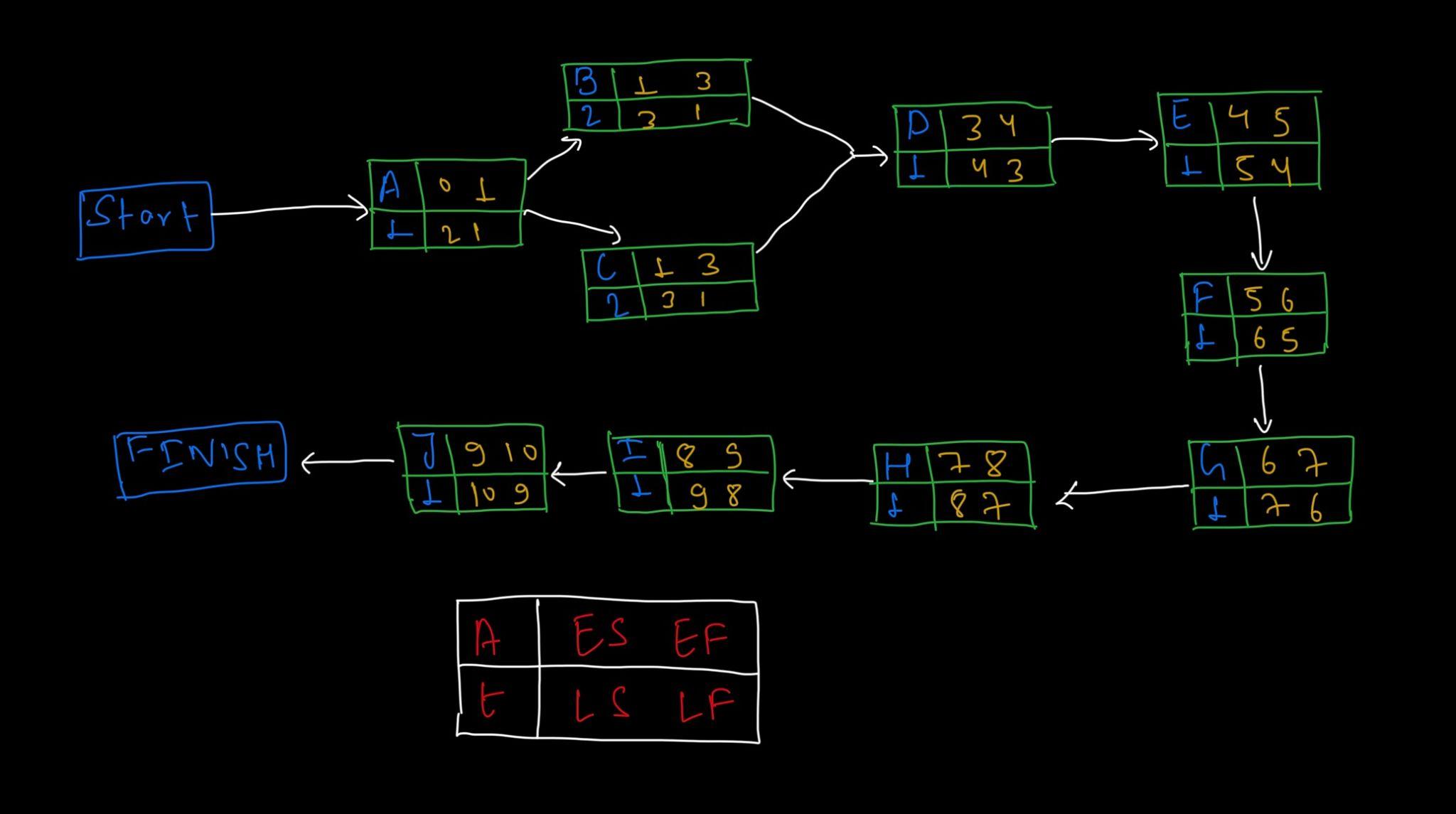
| **Output** | **Player Action** | **Notes** | **Expected Result** |
| --- | --- | --- | --- |
| **Pre-fight Phase Active** | Start the pre-fight phase. | The game begins the pre-fight phase with a countdown timer. | Game enters the pre-fight collection phase. |
| **No Collection** | Wait for time to expire without collecting items. | The player does not interact with any collectibles and simply waits. | Time runs out as no items are collected. |
| **Time Expired** | Verify time runs out without item collection. | The system verifies that the time has expired. | Timer reaches zero and no item collection happens. |
| **Automatic Transition** | Game automatically transitions to battle mode. | After time runs out, the game automatically switches to the battle phase. | Game transitions to the battle mode after time expires. |

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

### **Work items**

| Work Items / Task | Duration (PWks) | Predecessor Task(s) |
| --- | --- | --- |
| 1. Requirements Collection | 1 | - |
| 2. Collectible Item Design | 2 | 1 |
| 3. Inventory System Design | 2 | 1 |
| 4. Programming Collectibles | 1 | 2, 3 |
| 5. Animation & Sound Effects | 1 | 4 |
| 6. Testing | 1 | 5 |
| 7. Integration | 1 | 6 |
| 8. Final Polish & Debugging | 1 | 7 |
| 9. User Documentation | 1 | 8 |
| 10. Release Preparation | 1 | 9 |

### **Pert diagram**



### 

### 

### 

### 

| Work Items / Task | Duration (PWks) | Predecessor Task(s) |
| --- | --- | --- |
| 1. Requirements Collection | 1 | - |
| 2. Collectible Item Design | 2 | 1 |
| 3. Inventory System Design | 2 | 1 |
| 4. Programming Collectibles | 1 | 2, 3 |
| 5. Animation & Sound Effects | 1 | 4 |
| 6. Testing | 1 | 5 |
| 7. Integration | 1 | 6 |
| 8. Final Polish & Debugging | 1 | 7 |
| 9. User Documentation | 1 | 8 |
| 10. Release Preparation | 1 | 9 |

### 

### **Gantt timeline**

| 1 | ✴️1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 |  |  | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  | 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  | 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  | 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  | 9 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |  | 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 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |