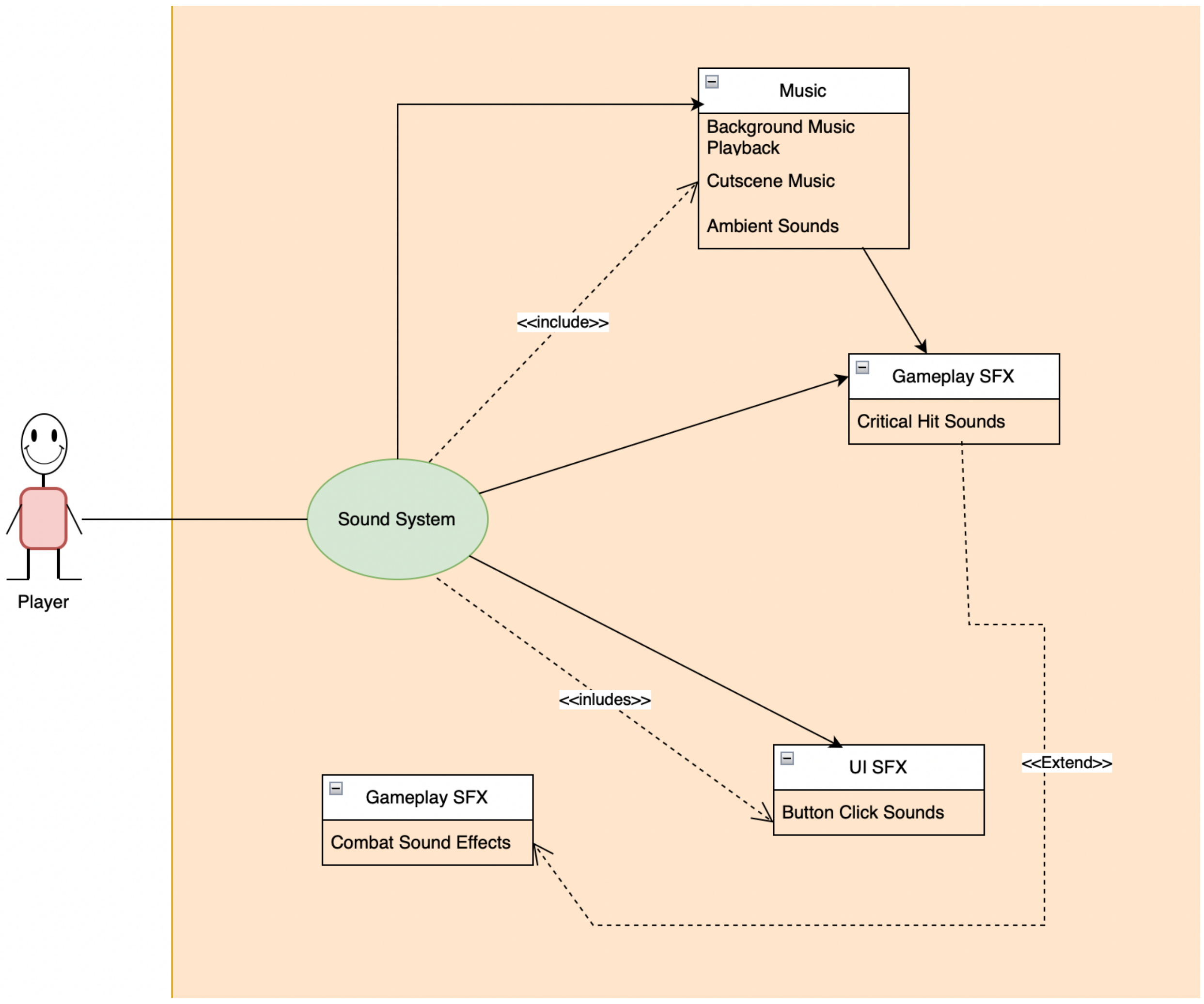
Name : Alija Acharya                    Mark ______________________/50

[**Instructions**: Remove everything that is not a heading below and fill in with your own diagrams, etc.]

## 1. Brief introduction __/3

I am responsible for the audio system of the game. My role is to design and deploy sound effects, music, and ambient sounds to increase immersion and to provide feedback to players. Sound messages will signal important game events such as enemy attacks, character status updates, or changing maps.

## 2. Use case diagram with scenario __14



Scenarios

**Name:** Game Ground System

**Summary:** The player interacts with the sound system, which dynamically plays music, gameplay SFX, and UI sounds based on game context

**Actors:** Player

**Preconditions:** The sound system is initialized and running.

**Basic sequence:**

- **Step 1:** Player enters the game world.
- **Step 2:** Background music begins playing.
- **Step 3:** Gameplay events (e.g., enemy spawns, critical hit) trigger associated SFX.
- **Step 4:** UI interactions (e.g., menu selections, button clicks) trigger UI SFX.
- **Step 5:** Music and ambient sounds change as player progresses or faces boss battles.

**Exceptions:**

**Step 3:** If resources for SFX fail to load, play a fallback sound and log the error.

**Step 2:** If no audio device is detected, mute the sound and prompt the player.

**Post conditions:** The correct music and SFX are played, enhancing player immersion.

**Priority:** 1

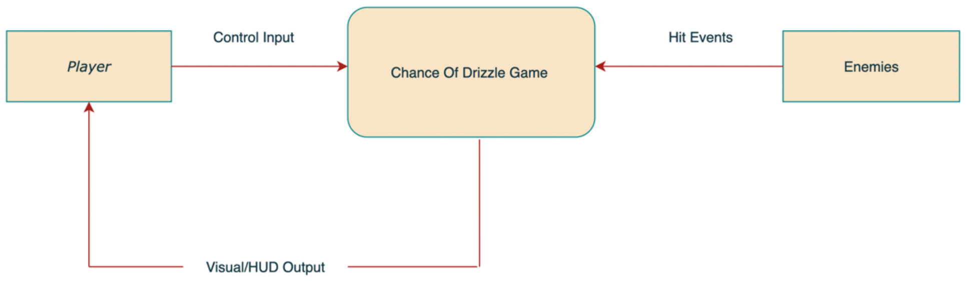
**ID:** S01

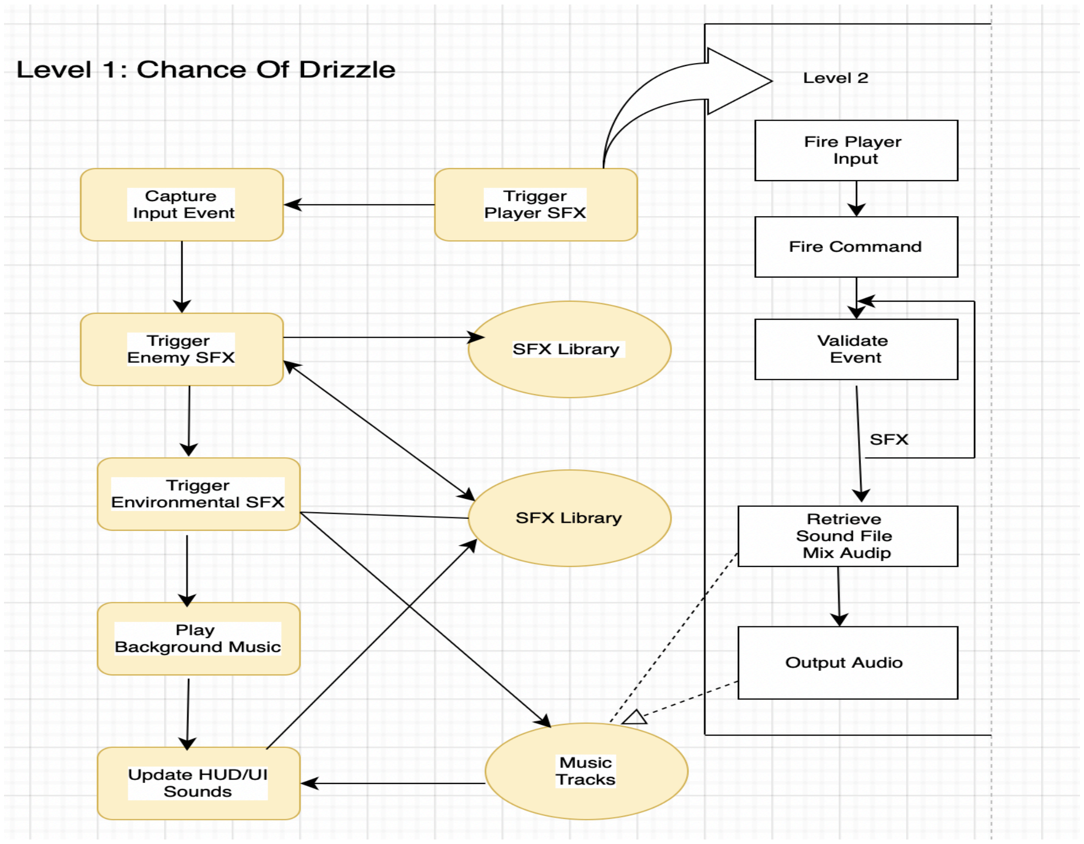
## 3. Data Flow diagram(s) from Level 0 to process description for your feature ________14

[Get the Level 0 from your team. Highlight the path to your feature]

### Data Flow Diagrams

Level 0 : Chance Of Drizzle Game





## Process Descriptions:

Manage all sound features dynamically based on game state and user interactions.

**Level 1:**

1. Capture Input Event
The system listens for player input events such as firing weapons, jumping, picking up items, or other gameplay actions. These input events trigger corresponding sound effects or music state transitions.

2. Trigger Player Sound Effects (SFX)
When a player action occurs, the sound system selects the appropriate sound effect from the SFX Library for that specific player action (e.g., shooting, footsteps, upgrades) and queues it for playback.
3. Trigger Enemy Sound Effects
Enemy-related events—such as attacks, getting hit, or deaths—are detected by the game engine

and the appropriate sound effect is selected from the SFX Library and played.
4. Trigger Environmental Sound Effects
Environmental or ambient sounds such as rain, wind, or map-specific ambiance are managed by the sound system to enhance immersion. These sounds may loop or change dynamically based on the game environment.
5. Play Background Music
Background music plays continuously, or switches tracks based on game state or events, such as entering new areas or engaging a boss. The music tracks are sourced from the Music Tracks data store.
6. Update HUD/UI Sounds
UI interactions like menu navigation, button clicks, or alerts (e.g., low health warning) prompt the sound system to play relevant UI sound effects for feedback.

**Data Stores**

SFX Library: The repository where all sound effect files (player, enemy, environmental, and UI sounds) are stored and retrieved during gameplay.
Music Tracks: A collection of background music files that the system uses to select and play depending on game states.
**Level 2 : Fire Weapon SFX**
1. Player Input: The player issues a "Fire" command via controls.
2. Validate Event: The system checks the validity of the event (weapon type, ammo, status).
3. Retrieve Sound File: Based on validation, the matching fire weapon sound effect is fetched from the SFX Library.
4. Mix into Audio Stream: The sound effect is mixed with other ongoing audio streams to ensure smooth playback.
5. Output to Player: Final mixed audio is sent to the player's audio output.

## 4. Acceptance Tests _________9

GOAL: To ensure that all sound features—background music, gameplay SFX, environmental sounds, and UI sounds—dynamically and smoothly respond to game state changes and user actions. The tests will verify proper sound playback, transitions, volume/mute handling, and the robust functioning of boundary cases.

Steps:

1. Launch the game and enter gameplay.
2. Perform player actions to trigger various sound effects.
3. Interact with UI to trigger UI sounds; adjust sound settings.
4. Switch game states (e.g., normal, combat, pause) to test music transitions and ambient sound changes.

Expected Results:

All actions produce correct, immediate sound output (no missing or delayed effects).
Background music adapts to game context (combat, exploration, pause).
UI and environmental sounds play accurately.
Volume and mute functions respond instantly.
No crashes or glitches when assets are missing, or settings are changed.

| **Input** | **Output** | **Notes** |
|---|---|---|
| Game Starts | Background music plays | Music matches gameplay context |
| Player action (shoot) | Gunfire sound plays | SFX matches action |
| Menu interaction | UI click sound plays | Immediate feedback |
| Volume adjusted/muted | Audio changes accordingly | Smooth transition |
| Error in sound asset | Fallback or silence | No crash, error logged |

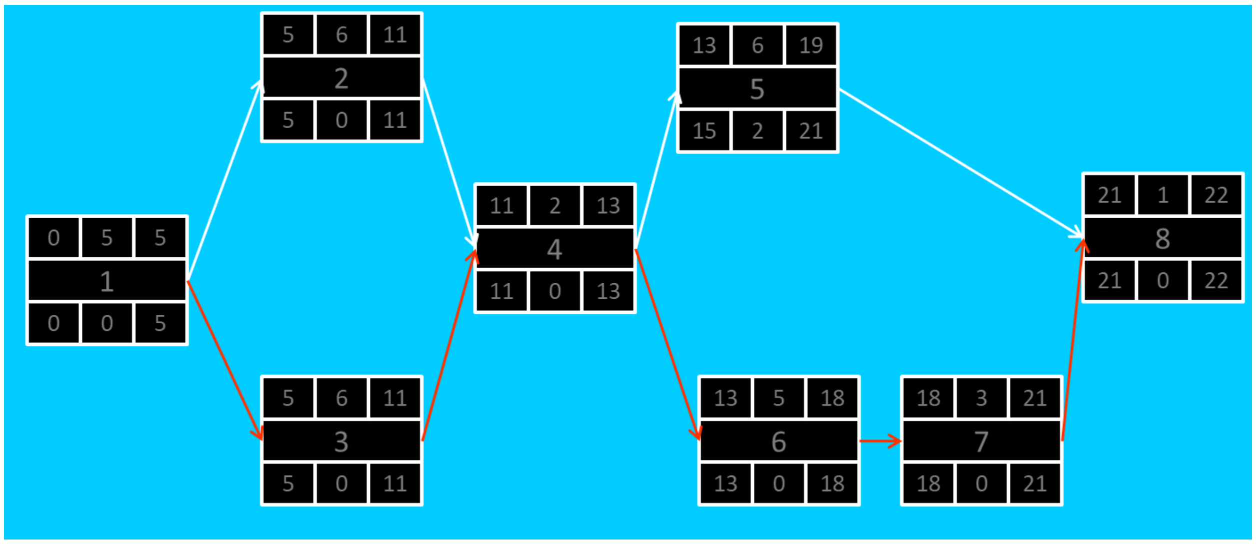
## 5. Timeline __________/10

[Figure out the tasks required to complete your feature]

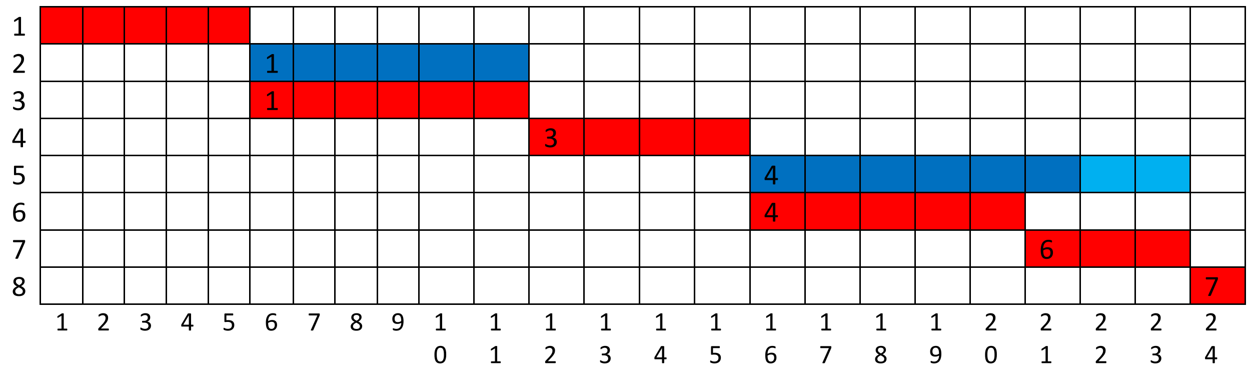
### Work items

| Task | Duration (PWks) | Predecessor Task(s) |
|---|---|---|
| 1. Requirements Collection | 5 | - |
| 2. Sound Asset Research and Selection | 4 | 1 |
| 3. Background Music Integration | 5 | 2 |
| 4. Combat and Ambient Sound Coding | 5 | 2 |
| 5. Sound System Integration & Testing | 4 | 3,4 |
| 6. User Testing & Bug Fixing | 6 | 5 |
| 7. Final Adjustments & Deployment | 4 | 6 |

## Pert diagram



## Gantt timeline


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