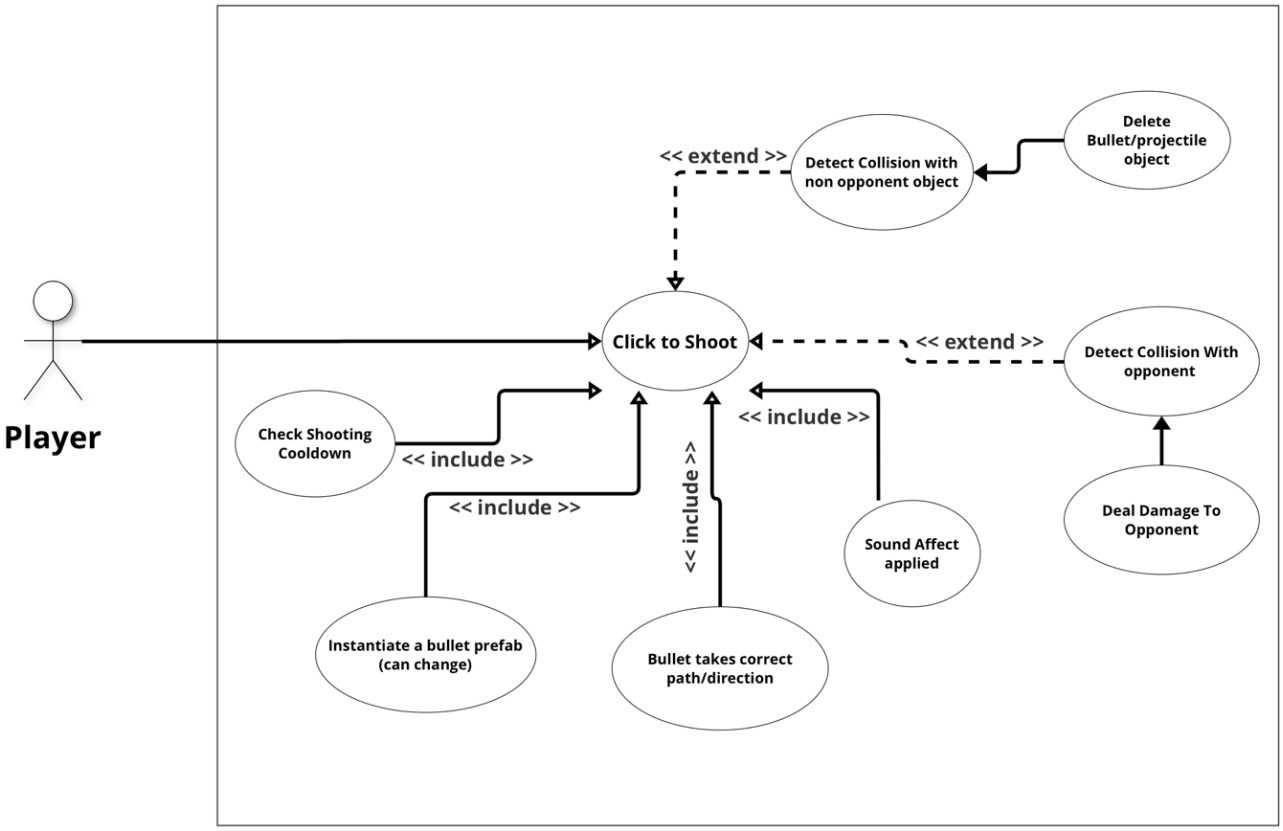
Name **Benton Wilson** Mark _____/50

## 1. Brief introduction _/3

My feature is going to be the shooting combat mechanics for the players. This will entail choosing our shooting method which is going to be a click to shoot where bullets travel in a line to where the mouse/crosshair is pointing, having to establish the cross hair. This is also going to have to interact with the health of other objects to deal damage when needed.

## 2. Use case diagram with scenario _14

### Use Case Diagrams




### Scenarios

**Name:** Shooting Mechanic

**Summary:** This will deal with how a player shoots and deals damage to items such as opponents.

**Actors:** Player

**Preconditions:** Objects needed to be used (bullets/projectiles) have been made and can be used. Health of other objects is initialized and can be manipulated.

**Basic sequence:**

**Step 1:** The player or opponent initiates a shooting mechanic. (Click)

**Step 2:** Check If cooldown on shooting.

**Step 3:** A bullet prefab is made

**Step 4:** Bullet takes correct direction (mouse)

**Step 5:** Sound affect is applied to launching of bullet.

**Exceptions:**

**Step 1:** Detect collision with some object

**Step 2:** If collides with opponent then deal damage else destroy bullet.

**Post conditions:** A bullet/projectile has been fired in the correct direction.

**Priority:** 2*

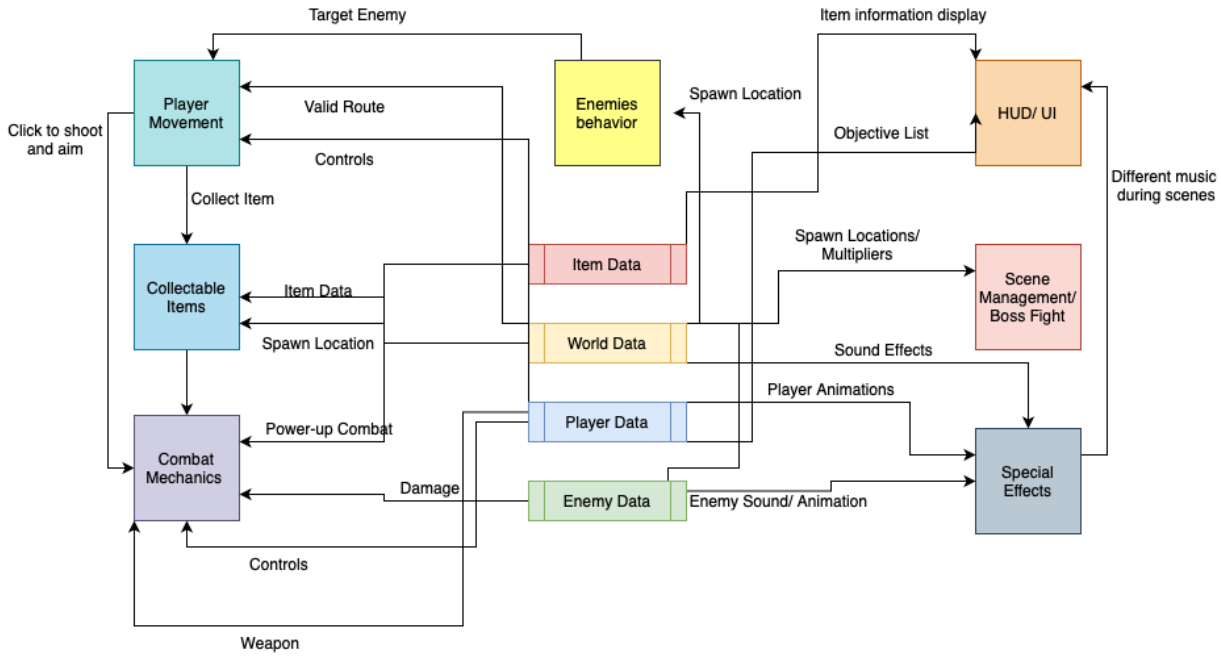
**ID:** C01

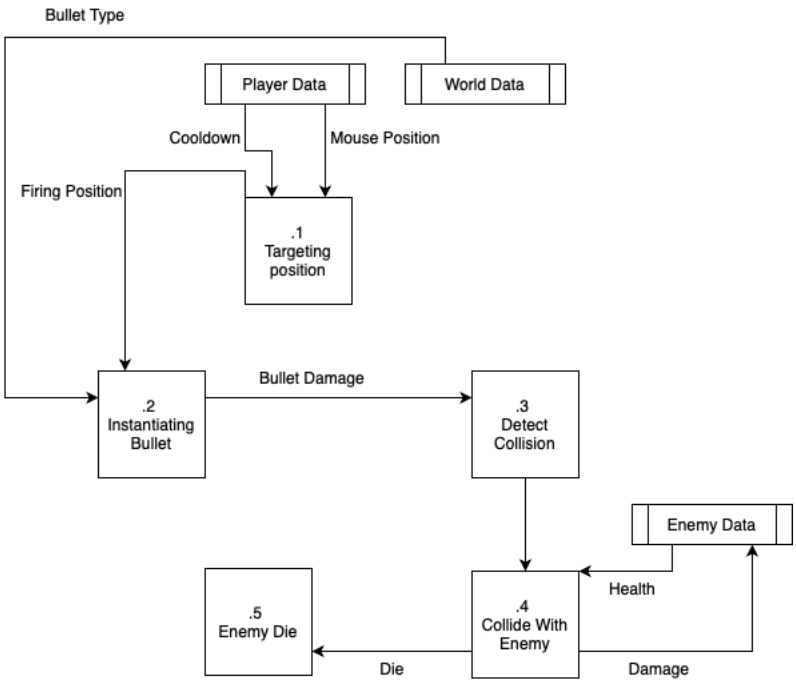
*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

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

### Data Flow Diagrams

STILL IN WORK






## Process Descriptions

Targeting Mechanics:

IF click to shoot is triggered

Get Mouse position from player data

Get Cooldown time from player data

Send firing position to Instantiate bullet

ENDIF

Instantiate Bullet:

IF receive data from targeting position

Get bullet type from world data

Get firing position from targeting position

Create a bullet object with damage field

ENDIF

Detect Collision:

IF Bullet object sent with damage

Detect if bullet object collides with enemy on screen

Move to handle enemy collision and store damage value

ENDIF

Collide with enemy:

```
IF collision detected with enemy object
        Get enemy health from enemy data
        Get bullet damage value
        IF Health <= Damage value
                Die
                Send to enemy data
        ELSE
                Damage Enemy Object
                Send to enemy data

        END
ENDIF
```

## 4. Acceptance Tests _____9

Make sure that the collision detection works between a projectile and a object on screen or some opponent object.

**Example for projectile collision**

Run feature with two objects in scene, an ordinary “obstacle object” and an “opponent” object. Both should be tagged correctly.

**Inputs:**

- A projectile object on screen
- Movement towards specific objects in game
- Some assigned damage value (Ex. 10)
- Opponent with health (tagged) (Ex. 100)
- Non-Opponent object (tagged)

**Outputs:**

- Collides with opponent:
    - Damage dealt
    - Projectile destroyed
- Collides with non-opponent:
    - Projectile destroyed

**Example for collision detection**

| Test Case | Description | Input | Output |
|---|---|---|---|
| 1 | Projectile prefab collides with an opponent tagged object | Projectile Prefab<br><br>Opponent Object | Deal damage to opponent and then should be destroyed. |

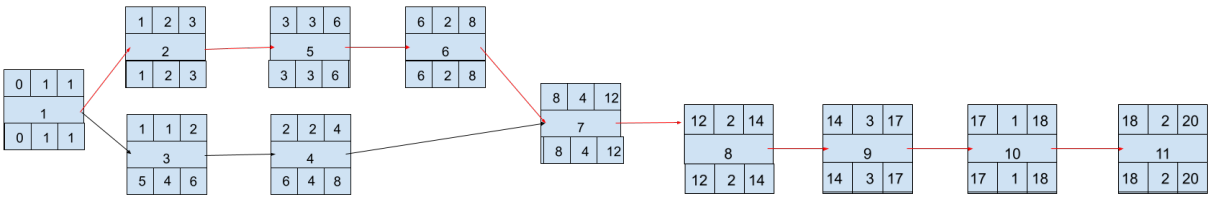
| | | | |
|---|---|---|---|
| | | Non-Opponent object | |
| 2 | Projectile prefab collides with a **non**-opponent tagged object | Projectile Prefab<br><br>Opponent Object<br><br>Non-Opponent object | Should just have the projectile be destroyed immediately. |
| 3 | Projectile misses all objects in scene | Projectile Prefab<br><br>Opponent Object<br><br>Non-Opponent object | There should be no course of action taken. (This should not happen in game as there will be an outer "boundary") |

## 5. Timeline _____/10

### Work items

| Task | Duration (PWks) | Predecessor Task(s) |
|---|---|---|
| 1 Requirement Definition | 1 | 0 |
| 2 Screen Design/ Level Design | 2 | 1 |
| 3 Create Bullet objects (sprites) | 1 | 1 |
| 4 Program bullet class | 2 | 3 |
| 5 Enemy creation | 3 | 2 |
| 6 Program Enemy health class | 2 | 5 |
| 7 Programming the shooting functions | 4 | 6,4 |
| 8 Integration of sound effects | 2 | 7 |
| 9 Testing | 3 | 8 |
| 10 Creation of the crosshair interface | 1 | 9 |
| 11 Deployment | 2 | 10 |

### Pert diagram



## Gantt timeline