Name: Gabriel Arizmendi Mark ______________________/50

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

1. Brief introduction __/3

My feature for the Justy Fishin' game will be implementing the fish and a portion of their mechanics.

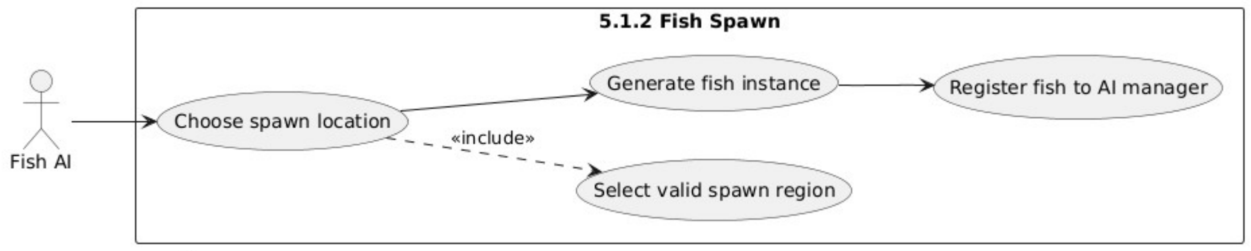
When the boat launches from shore and goes into sea, I need to make sure the player has plenty of fish to hopefully catch. The fish that spawn when the player goes to sea will be randomly generated from a set of fish that will be different every time the player goes out.

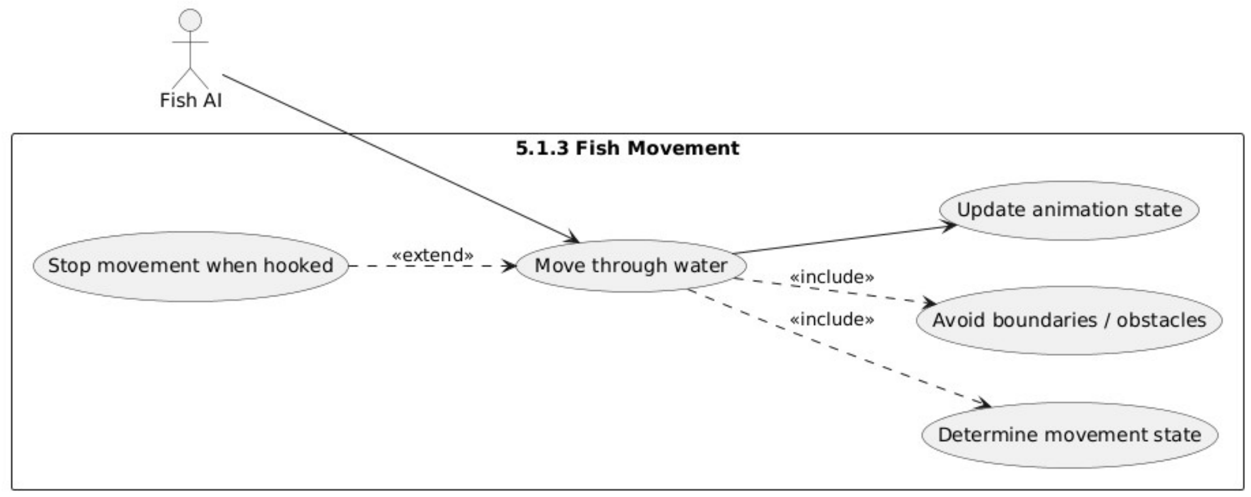
The fish will also need to fight against the player so that the game is not too easy. Each fish will fight differently, with some fighting harder than others. The fish that fight harder may weigh more or they may be a rarer fish that is worth more than a basic fish.

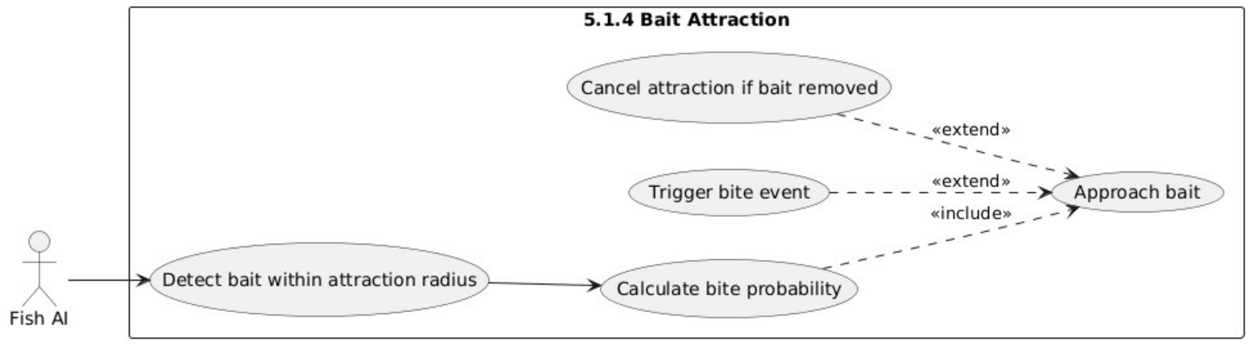
I will also need to make sure that when the player uses bait, that the fish are more likely to be attracted to the user's rod. How attracted the fish are to the lure will most likely depend on the rarity of the bait used.

2. Use case diagram with scenario __14

Use Case Diagrams






5.1.4 Bait Attraction
Cancel attraction if bait removed
«extend»
Trigger bite event
«extend»
Approach bait
«include»
Fish AI
Detect bait within attraction radius
Calculate bite probability

Scenarios

5.1.2 Fish Spawn

Summary: Fish AI spawns a new fish by selecting a valid water region, choosing a spawn point, creating the fish, and registering it so it can be managed by AI.

Main Steps:

1. Fish AI requests a spawn action (timer/event triggers spawning).
2. System selects a valid spawn region within the map's water areas.
3. System chooses a spawn location inside the selected region.

4. System generates a fish instance (type/rarity/size initialized).
5. System registers the fish to the AI manager so it can be tracked and updated.

Exception Case:

- If no valid spawn region is available or the fish cap is reached, the system cancels the spawn and logs/returns a “spawn failed” result with no fish created.

5.1.3 Fish Movement

Summary: Fish AI updates a fish’s movement by deciding its movement state, moving it through water, preventing it from leaving the water area, and updating animation—unless the fish becomes hooked.

Main Steps:

1. Fish AI begins the movement update for a fish on each game tick.
2. System determines movement state (idle, roam, approach bait, etc.).
3. System moves through water using the fish’s speed and direction.
4. System avoids boundaries/obstacles to keep the fish in valid water space.
5. System updates animation state to match current movement.

Exception Case:

- If the fish becomes hooked, the system stops normal roaming movement and switches to the hooked/struggle behavior (movement control is handed off to the fishing/hook mechanic).

5.1.4 Bait Attraction

Summary: Fish AI detects a lure in range, calculates the chance to bite, and if conditions are met the fish approaches and may trigger a bite event; attraction can be canceled if the lure disappears.

Main Steps:

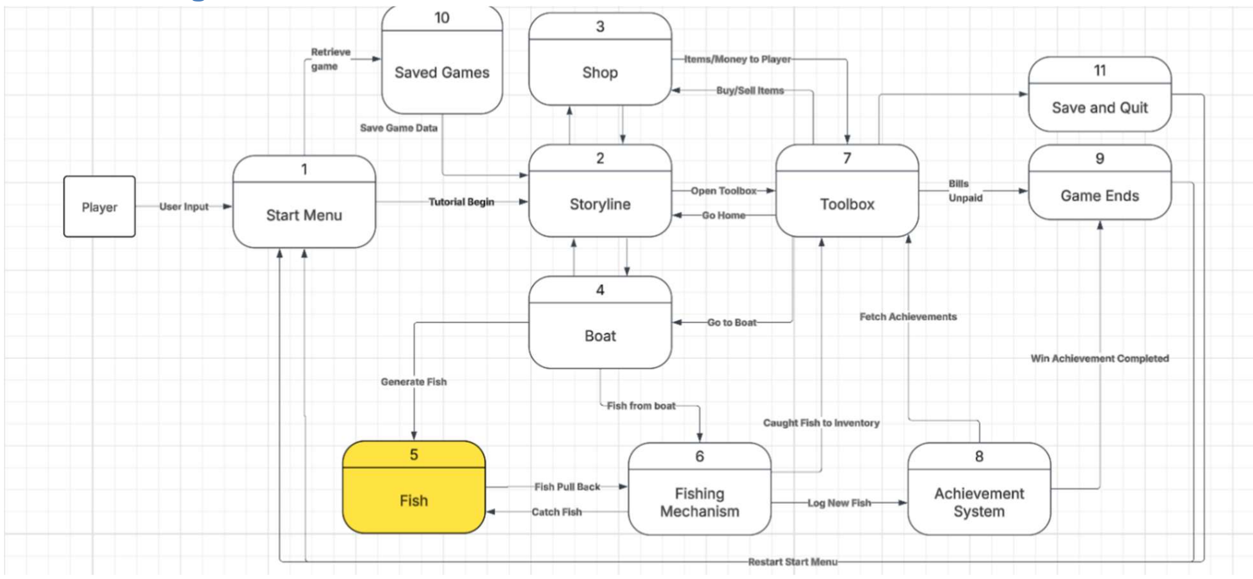
1. Fish AI checks whether the lure/bait is near a fish.
2. System detects bait within attraction radius.
3. System calculates bite probability using distance and fish/bait factors.
4. System includes “Approach bait” (fish begins moving toward the lure).
5. If the fish reaches bite range and chance succeeds, the system extends to “Trigger bite event” (bite occurs and hook logic is notified).

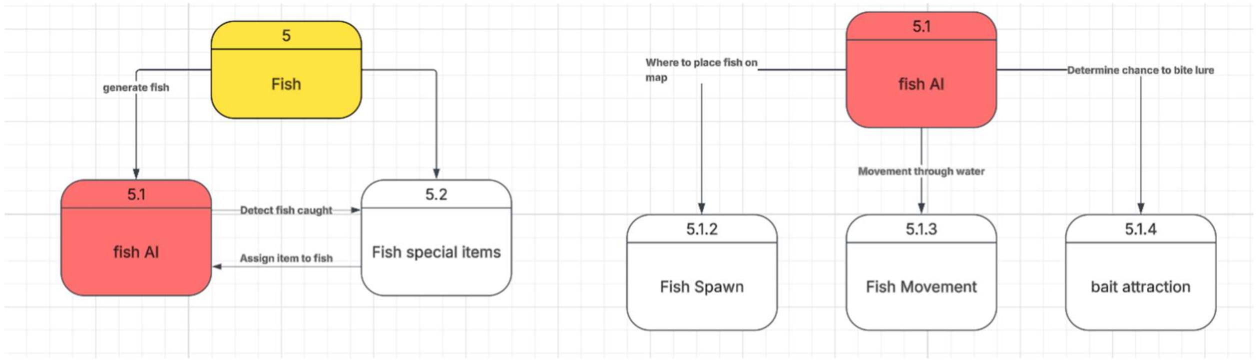
Exception Case:

- If the bait is removed or moved out of attraction radius while the fish is approaching, the system cancels attraction and the fish returns to roaming/idle without triggering a bite.

3. Data Flow diagram(s) from Level 0 to process description for your feature ______14

Data Flow Diagrams





Process Descriptions

Name: 5.1.2 – Fish Spawn

Parent process: 5.1 – Fish Ai

Summary: The fish AI system will determine the best place to spawn the fish that will fit within the game map. This process is what enables the player to catch fish and see where they are, when applicable. This

will require the inputs of the map boundaries, the water boundaries, the fish types themselves. It will output a spawned fish entity with the proper attributes, and add a fish to the fish AI system
Process Steps:
1: Validate available spawn regions within water boundaries.
2: Randomly select a spawn point within valid coordinates.
3: Randomly determine fish.
4: Initialize fish attributes (health, size, rarity, strength).
5: Spawn fish object into the scene.
6: Register fish with Fish AI system.
Exceptions:
1: Validate spawn regions
2: No valid spawn regions found
3: Do not spawn any fish, wait for valid spawn region.

Name: 5.1.4 – Bait attraction
Parent process: 5.1 – Fish Ai
Summary: The bait attraction will determine how likely the fish is to go for the player's current bait. It will then calculate the bite probability based on a couple of factors such as rarity/type, how far it is from the bait, the bait type, and a random value. The output of this process will be the fish state changing, the hook trigger, and notify the fishing system.
Steps:
1: Detect if bait is within attraction radius.
2: Compare bait type against fish preference table.
3: Calculate bite probability using:
3a: Distance modifier
3b: Rarity modifier
3c: Fish aggression level
4: Generate random value.
5: If probability threshold met, switch fish state to "Approach Bait".
6: If within bite distance, trigger bite event.
Exceptions:
1: Fish sees the bait.
2: Fish checks how hungry it is, and if it likes the bait
3: The fish is not interested so it swims away.
4: No other systems get notified of the fish.

4. Acceptance Tests ______9

5.1.2 Fish Spawn

- Normal spawn in valid water

Preconditions: Spawning enabled; at least 1 valid water region; fish count < cap.

Inputs: Spawn tick/event triggers; spawn rate > 0.

Expected Output: Exactly 1 fish instance is created **inside** a valid water region and added to AI tracking list.

- Spawn point boundary — edge of valid water

Preconditions: Known water region rectangle/polygon.

Inputs: Spawn algorithm selects a point **on the exact boundary** of the water zone.

Expected Output: Fish spawns successfully and is considered "in water" (no instant correction/teleport unless design says otherwise).

- Spawn point just outside valid water (boundary rejection)

Preconditions: Water boundary exists with collision/validation.

Inputs: Candidate spawn point at epsilon outside boundary (ex: x = maxX + 0.01).

Expected Output: Spawn is **rejected** and the system retries or cancels per rules; **no fish** appears outside water.

5.1.3 Fish Movement

- Normal roaming movement

- **Preconditions:** Fish exists in water; not hooked.
- **Inputs:** Δt update (frame/tick), speed > 0.
- **Expected Output:** Fish position changes smoothly; stays within water boundaries.

-Movement speed boundary — speed = 0

- **Preconditions:** Fish exists, not hooked.
- **Inputs:** speed = 0.
- **Expected Output:** Fish does not move; animation state is idle (or "swim-in-place" if that's your design).

-Very small (delta t) (near-zero frame time)

- **Preconditions:** Fish exists, not hooked.
- **Inputs:** delta t ≈ 0 (e.g., 0.0001).
- **Expected Output:** Position change is negligible; no jitter/teleport; system remains stable.

5. Timeline ________/10

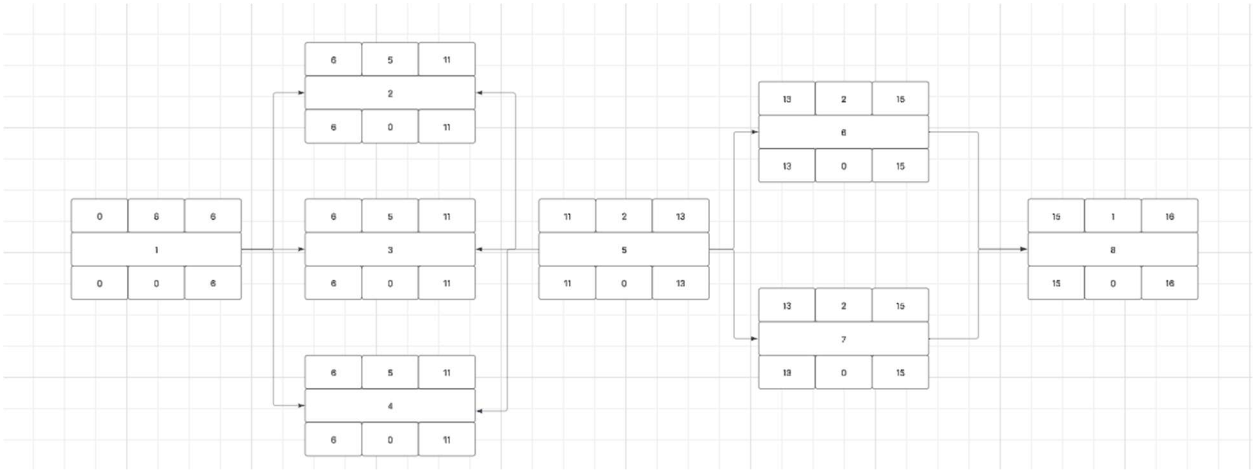
[Figure out the tasks required to complete your feature]

Example:

Work items

Task	Duration (PWks)	Predecessor Task(s)
1. Requirements Collection	6	-
2. Spawn programming	5	1
3. Movement programming	5	1
4. Attraction programming	5	1
5. test all(hopefully) interactions	2	2,3,4
6. Model/Sound Polishing	2	5
7. Final testing/fixing	3	6
8. deployment/Ins	1	6,7

Pert diagram



Gantt timeline

1	🟥	🟥	🟥	🟥	🟥	🟥										
2						🟦 1	🟦	🟦	🟦	🟦	🟦					
3						🟥 1	🟥	🟥	🟥	🟥	🟥					
4						🟦 2	🟦	🟦	🟦	🟦	🟦					
5						🟥 2	🟥	🟥	🟥	🟥	🟥					
6						🟦 3	🟦	🟦	🟦	🟦	🟦					
7						🟥 3	🟥	🟥	🟥	🟥	🟥					
8						🟦 4	🟦	🟦	🟦	🟦	🟦					
9						🟥 4	🟥	🟥	🟥	🟥	🟥					
10											🟦 5	🟦	🟦			
11											🟥 5	🟥	🟥			
12													🟦 6	🟦	🟦	
13													🟥 6	🟥	🟥	
14													🟦 7	🟦	🟦	
15													🟥 7	🟥	🟥	
16															🟦 8	🟦
															🟥 8	🟥
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16