Name: Dristanta Silwal Mark _____/50

1. Brief introduction __/3

My feature for "Legends of Warriors" involves implementing AI and Enemy Behavior. The following shows the detailed outline that my features include:

a. Enemy Attack Pattern: As without the enemy, there's no fun playing the game. So, creating varieties combat styles, special powers come under this task.

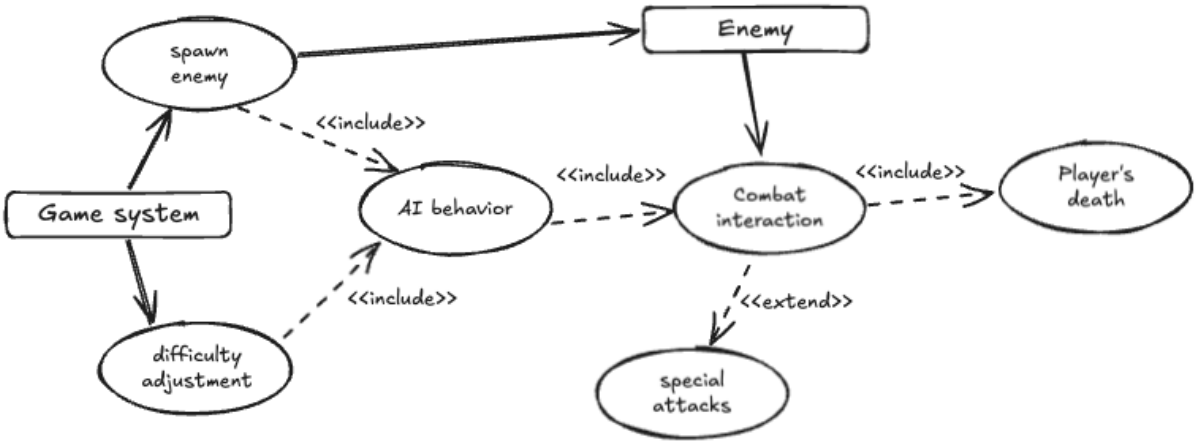
b. Change of Level Difficulty: Making sure that the parameters like health, attack power, damage are progressive based on game's need.

c. Bot movement and their attacking pattern: Designing movement, attack of bots for making sure that they're steering the game.

d. Random Enemy Selection: Other that the story mode, enemies will be chosen based on my game's logic for better organization of the game.

2. Use case diagram with scenario __/14

Use Case Diagrams



Scenarios

1. Scenario 1: Enemy Spawning & Behavior

   Name: Spawn and Configure Enemy

   Summary: The system spawns enemy, assign its attributes (health, outfits, weapon) and adjusts them based on game difficulty level dynamically.

   Actors: Enemy, Bots

   Preconditions: A spawn event is triggered for enemy and proximity timer for bots (for their random generation in the game scene).

   Basic sequence:

   Step 1: Player triggers spawnEnemy() in EnemySpawner.

   Step 2: Enemy Spawner creates EnemyAI instance (for bots, its timer based)

   Step 3: Adds enemy's attributes (health, outfits, weapon).

   Step 4: Calls adjustEnemyStats() in DifficultyManager.

   Step 5: EnemyAI attack() player

   Step 6: DifficultyManager adjusts enemy's attributes dynamically.

   Exceptions:

   Step 1: If max number of bots is reached, no new bots will be introduced.

   Step 2: If adjustEnemyStats() fails, default stats are applied.

   Post conditions: A fully configured enemy along with bots are spawned and engages in combat with player.

   Priority: 1 (Must have)

   ID: ESB1

2. Scenario 2: Enemy Defeat and Cleanup

   Name: Remove Defeated Enemy & bots

   Summary: Once bots are defeated (health == 0), it is removed from the game, and active bots count is updated. If an enemy is dead, there'll be end of that round.

   Actors: Enemy, Bots, Player

   Preconditions: Health of enemy & bots reaches 0.

   Basic sequence:

   Step 1: Player attacks EnemyAI instances.

   Step 2: Enemy & bots calls takeDamage() and reduces their health.

   Step 3: Once health is 0, notifyDefeat() in EnemySpawner. If enemy is defeat, it's the end of that round. If bot, then, updateBotsCount().

   Step 4: EnemySpawner removes the defeated bots.

   Exceptions:

Step 1: if notifyDefeat() fails, the enemy remains in the game temporarily without movement.

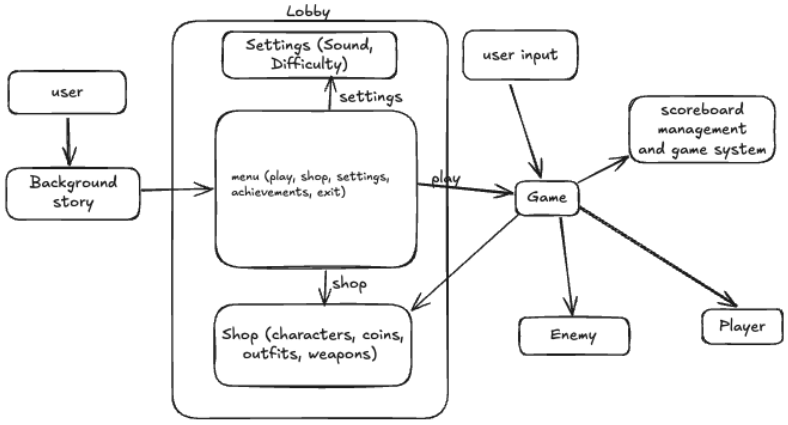
Post conditions: The defeated bots are removed, and their active count is updated.

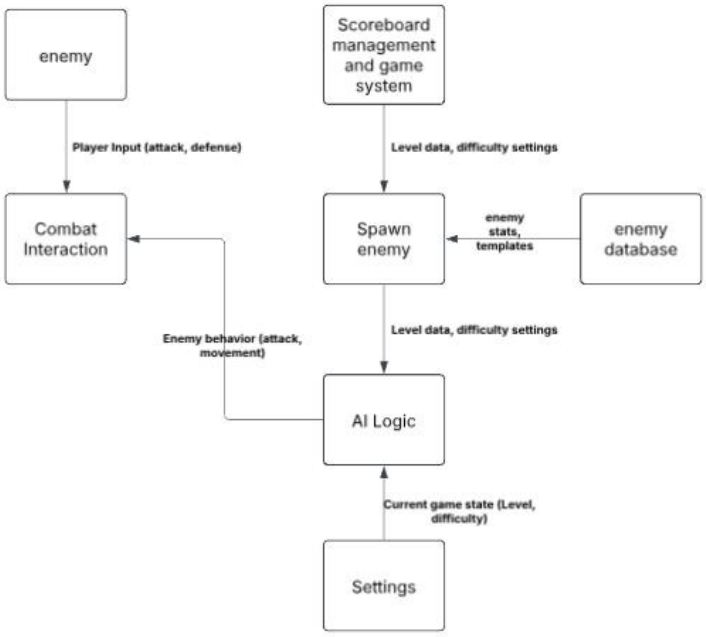
Priority: 2 (Essential)

ID: EDC1

3. Data Flow diagram(s) from Level 0 to process description for your feature _____/14

Data Flow Diagrams





Process Descriptions

Process 1: Enemy & Bots spawning

Input: Spawn request

Output: Enemy Instances for enemy and bots with randomized attributes

Handles the creation of new enemy, bots and their attributes.

Process 2: Difficulty Adjustment

Input: EnemyAI instances, Difficulty level

Output: Adjusted stats

Dynamically change enemy, bots attributes on game.

Process 3: Enemy & Bots Behavior execution

Input: EnemyAI instance, Player position

Output: Enemy, Bots movement and attack actions

Executes movement, attacks, and evasion strategies during combat.

Process 4: Enemy & Bots Defeat & Cleanup

Input: Health status

Output: Removal of bots

Removes bots from the game and update active bots count.

a. Enemy Spawning tests

- **Ensure enemy is spawned in particular location for each scene.**
- **Make sure all bots are spawned within the designated area.**
- **Test that no more than the maxBots are active at any time in the game scene.**

b. Difficulty scaling tests

- **Verify enemy stats and bots' stats are correctly adjusted on each difficulty level and game rounds.**
- **Make sure that health, speed, attack, damage scale proportionally.**

c. Behavior Tests

- **Checks enemies movement and engagement based on their type (melee & boss).**
- **Ensure special abilite on enemy are triggered with proper timing and placement as planned.**

Example for Difficulty Scaling

Difficulty Level	Bots Type	Health	Speed	Attack Power	Notes
Easy	Melee	100	1	10	Default
Medium	Melee	150	1.2	15	Moderate increase
Hard	Melee	200	1.5	20	For challenging distractions
Hard	Boss	500	1.3	30	Focusing on long game time

Spawning Logic

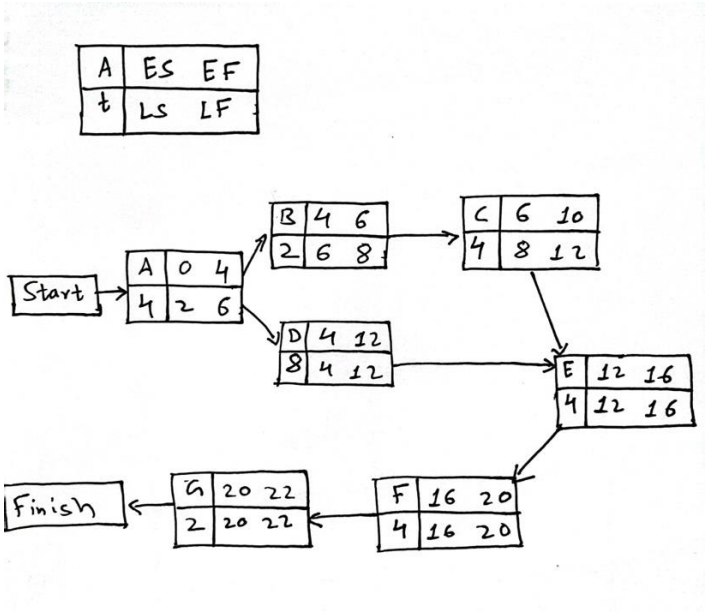
Spawn Trigger	Active bots	Action	Notes
Timer reaches 5 seconds	< maxBots	Spawn new bot	Random attributes for each spawn
Timer reaches 5 seconds	= maxBots	Don't spawn	Wait for bot to be defeated
Player enters area	< maxBots	Spawn bots near the player	Adjust difficulty based on player stats.

5. Timeline _____/10

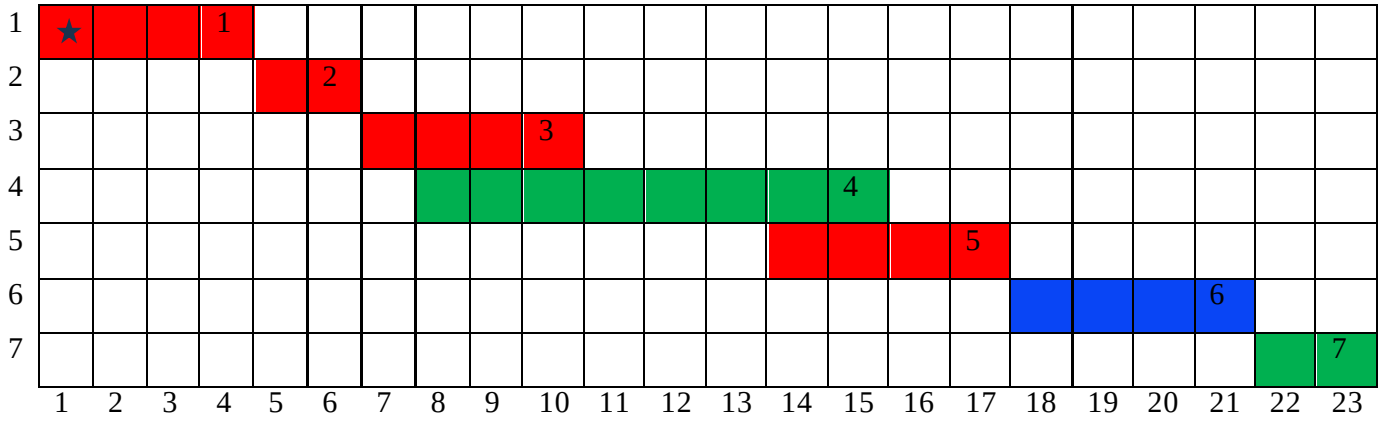
Task	Duration (hrs)	Predecessor Task(s)
1. AI state machine design	4	-
2. Enemy attribute randomization	2	1
3. Difficulty adjustment logic	4	2
4. Enemy behavior programming	8	1
5. Defeat and cleanup logic	4	3, 4
6. Testing and debugging	4	5
7. Documentation	2	6

Work items

Pert diagram



Gantt timeline



Red: Critical tasks

Blue: Testing and Iterative work

Green: Development and Documentation