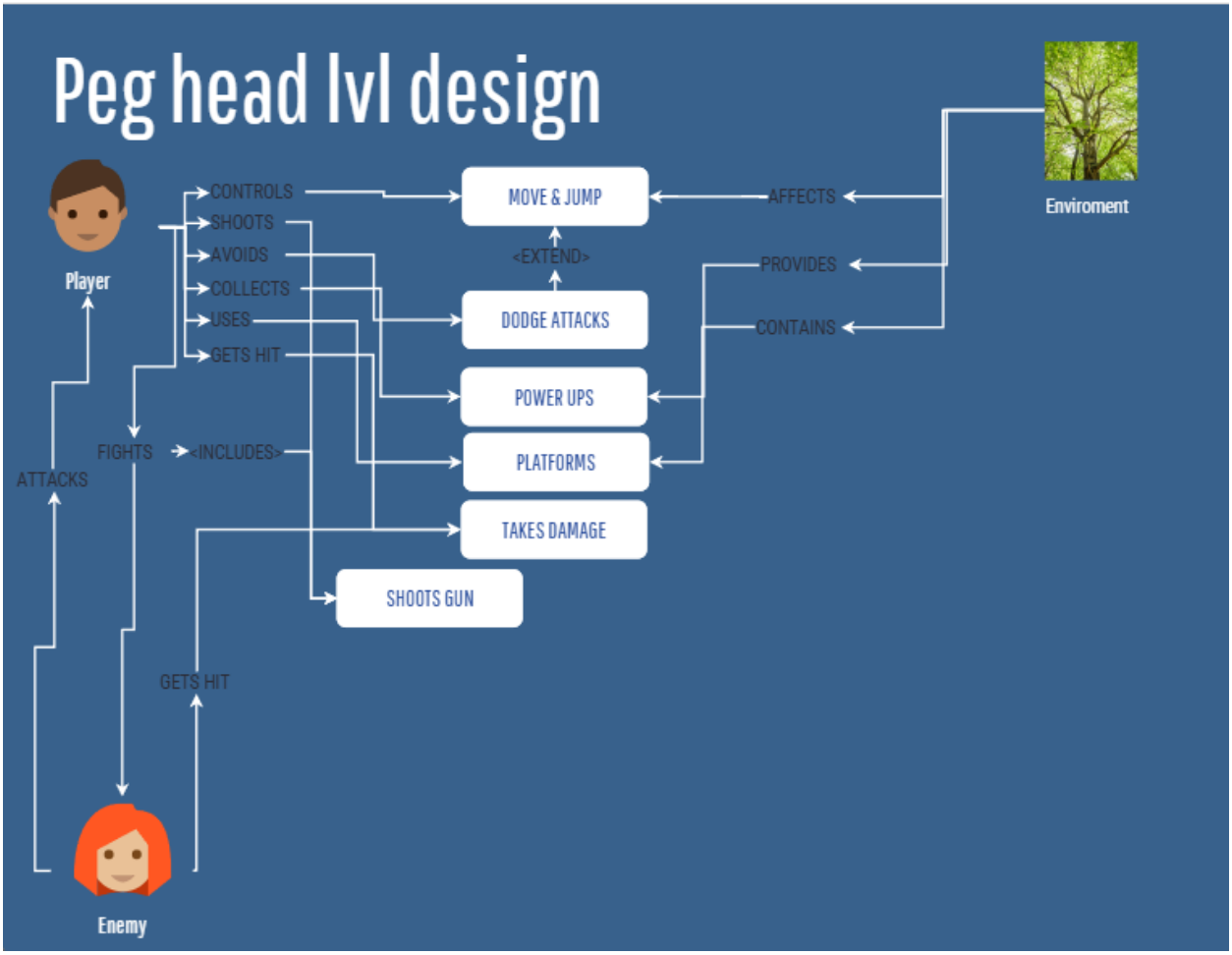
Name____Miles Handlen_____ Mark _____/50

1. Brief introduction _/3

This feature introduces an ever-changing backdrop that challenges the player's adaptability. As the player progresses through the level, environmental elements—such as platforms, obstacles, or hazards—shift and evolve in real time.

2. Use case diagram with scenario _14

Use Case Diagrams



Scenarios

1. Use Case: Move & Jump

Summary: The player moves the character left or right and jumps over obstacles.

Actors: Player.

Preconditions: The game level is loaded, and the player has control of the character.

Basic sequence:

1. The player presses the left or right movement buttons.

2. The character moves accordingly.
3. The player presses the jump button.
4. The character jumps to a platform or over an obstacle.
5. If the player times the jump incorrectly, the character may fall into a trap or hazard.
6. The movement and jump actions repeat until the level is completed or the character loses all lives.

Priority: 1
ID: C01

2. Use Case: Shoot Projectiles
Summary: The player shoots at enemies using a weapon.
Actors: Player.
Preconditions: The player has a weapon.

Basic sequence:

1. The player aims using the directional controls.
2. The player presses the shoot button.
3. A projectile is fired toward the aimed direction.
4. If the projectile hits an enemy, it deals damage.
5. If the enemy's health reaches zero, the enemy is defeated.
6. The player continues shooting as needed until the enemy is defeated or the player moves on.

Priority: 1
ID: C02

3. Use Case: Dodge Attacks
Summary: The player dodges enemy attacks by moving, jumping, or dashing.
Actors: Player.
Preconditions: The enemy is attacking the player.

Basic sequence:

1. The enemy prepares an attack animation.
2. The player anticipates the attack and reacts.
3. The player presses the jump, dash, or move button at the right time.
4. If the dodge is successful, the attack misses.
5. If the dodge is mistimed, the player gets hit and takes damage.

Priority: 2
ID: C03

4. Use Case: Defeat Enemies
Summary: The player eliminates enemies using attacks or projectiles.
Actors: Player, Enemy.
Preconditions: The player has weapons

Basic sequence:

1. The player encounters an enemy.
2. The enemy starts attacking the player.
3. The player shoots or attacks the enemy.
4. If the enemy's health is reduced to zero, the enemy is defeated.
5. If the enemy is still alive, it continues attacking.
6. The cycle continues until either the enemy or the player is defeated.

Priority: 2

ID: C04

5. Use Case: Take Damage

Summary: The player loses health when hit by an enemy or hazard.
Actors: Player, Enemy, Environment.
Preconditions: The player is within range of an enemy attack or environmental hazard.

Basic sequence:

1. An enemy fires a projectile or attacks the player.
2. The player fails to dodge or block the attack.
3. The attack lands, reducing the player’s health.
4. If the player's health reaches zero, the character loses a life or the game ends.

Priority: 1
ID: C05

6. Use Case: Collect Power-ups
Summary: The player picks up power-ups that enhance abilities.
Actors: Player, Environment.
Preconditions: A power-up is available in the level.

Basic sequence:

1. **The player moves toward a visible power-up.**
2. The player touches or interacts with the power-up.
3. The power-up is added to the player’s inventory or takes effect immediately.
4. The power-up grants benefits such as extra health or increased damage.

Priority: 3
ID: C06

7. Use Case: Interact with Platforms
Summary: The player jumps onto or interacts with moving, destructible, or special platforms.
Actors: Player, Environment.
Preconditions: Platforms exist in the level and are accessible.

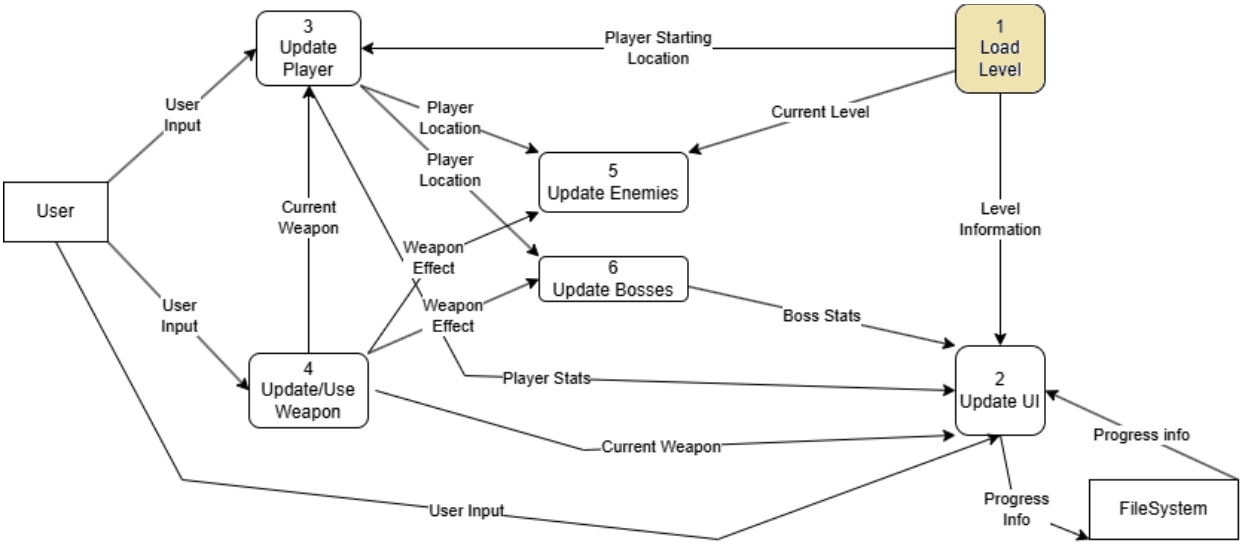
Basic sequence:

1. The player approaches a platform.
2. If it's a static platform, the player jumps onto it.
3. If it's a moving platform, the player times their jump accordingly.
4. If it's a destructible platform, the player stands on it for a short time before it breaks.
5. The player continues navigating platforms to progress through the level.

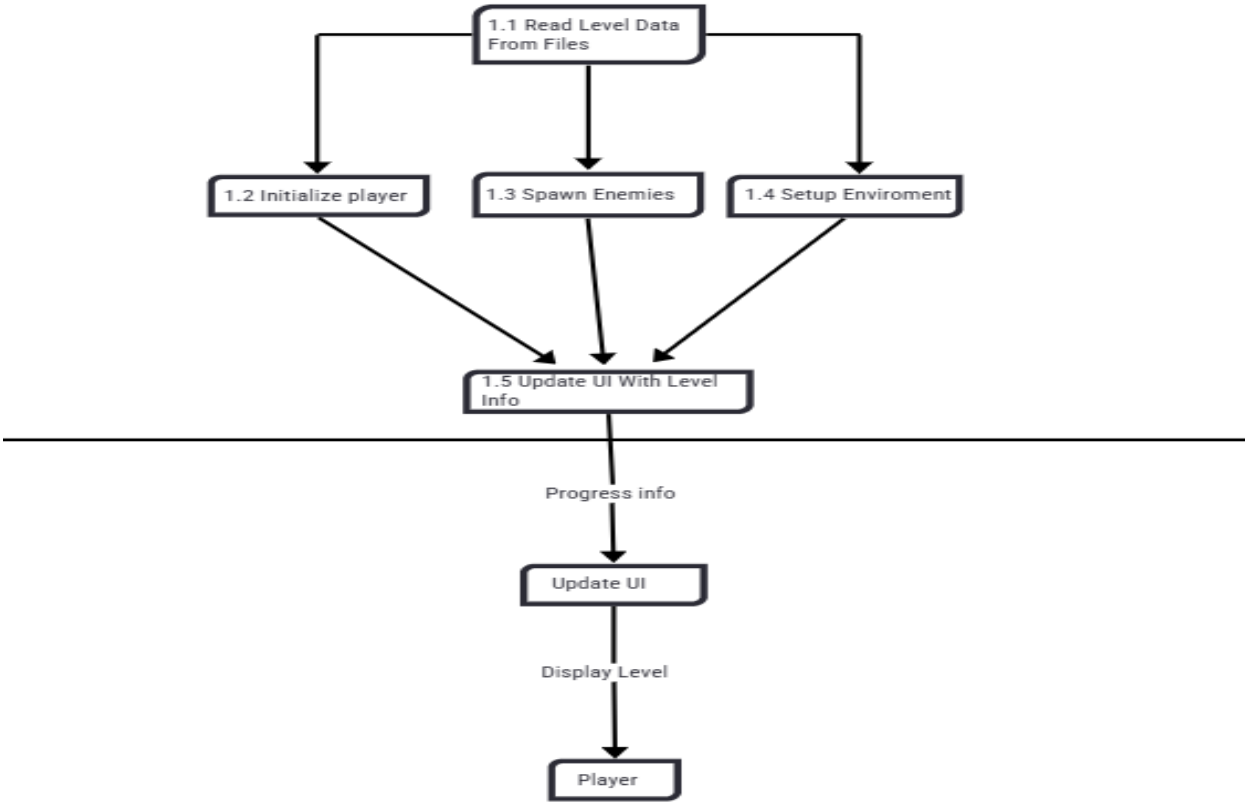
Priority: 1
ID: C07

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

Data Flow Diagrams



Load level



Process Descriptions

1.1 Read Level Data

The system retrieves level data from the File System, which contains information about the level layout, enemy positions, platform locations, and environmental interactions.

1.2 Initialize Player

The system places the player at the starting position and initializes attributes such as health, weapon selection, and abilities.

1.3 Spawn Enemies

The system spawns enemies at predefined positions based on the level data.
Each enemy type is initialized with specific attributes, including health, attack patterns, and movement behavior.

1.4 Setup Environment

The system loads all environmental objects, such as platforms, traps, and collectible items.
Interactive elements like destructible platforms, moving obstacles, and power-ups are initialized.

1.5 Update UI with Level Info

The system updates the UI to reflect the loaded level.
Displays relevant game information, such as the level name, objectives, and player status.

4. Acceptance Tests _____9

Test Case 1: Read Level Data

Description: Verify that the system correctly loads level data from the file system.
Preconditions: The game has a valid level file in the system.

Test Steps:

1. Start the game and select a level.
2. Observe if the system retrieves the correct level data.
   Expected Result: The correct level layout, enemy positions, and environmental elements are loaded.

Test Case 2: Initialize Player

Description: Ensure the player spawns at the correct starting position with default attributes.

Preconditions: The level is loaded, and player data is available.

Test Steps:

1. Start the game and enter the level.
2. Observe the player's initial position and stats.
   Expected Result: The player appears at the correct starting point with full health and default abilities.

Test Case 3: Spawn Enemies

Description: Verify that enemies spawn at the correct locations and with correct attributes.

Preconditions: Level data includes predefined enemy positions.

Test Steps:

1. Start the game and load a level with enemies.
2. Check enemy positions and movement behavior.
3. Engage in combat to verify health and attack patterns.
   Expected Result: Enemies appear in correct positions and behave according to their AI settings.

Test Case 4: Setup Environment

Description: Ensure all platforms, obstacles, and interactive objects are present and function correctly.

Preconditions: Level data contains environmental objects.

Test Steps:

1. Start the game and observe the environment.
2. Jump on platforms and interact with objects.
3. Test moving objects (e.g., elevators, traps).
   Expected Result: All objects are correctly positioned and interactable.

Test Case 5: Update UI with Level Info

Description: Confirm that UI updates correctly when a new level loads.

Preconditions: UI components should be responsive to level loading.

Test Steps:

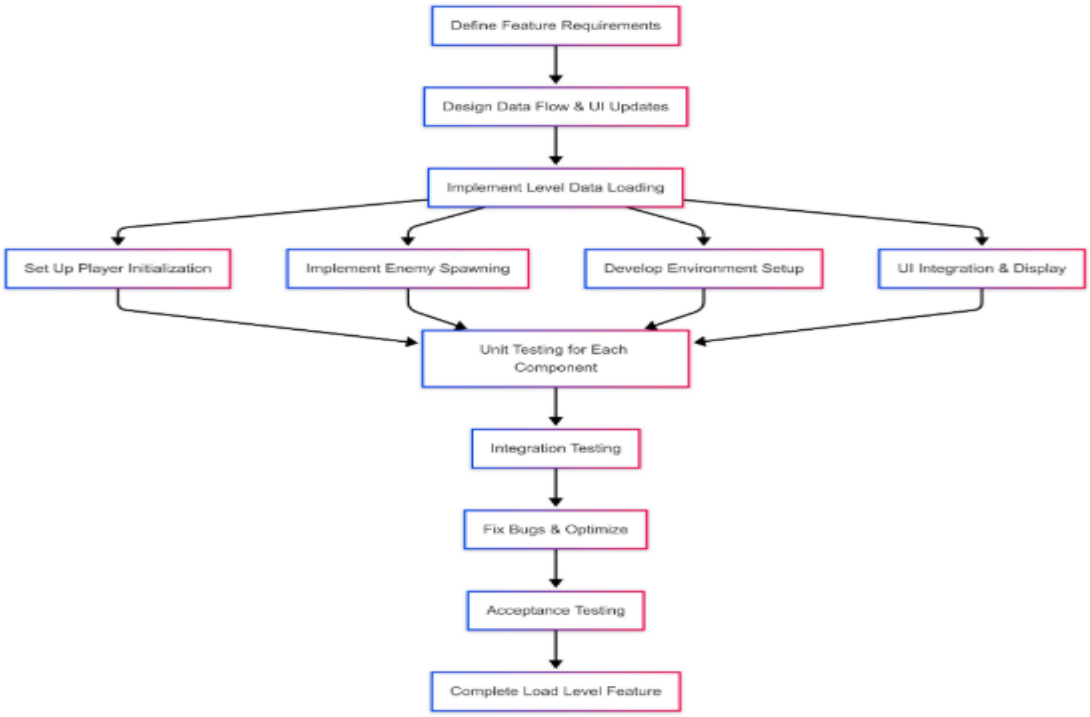
1. Load a new level.
2. Check if the UI displays level name, objectives, and player stats.
3. Verify health, score, and weapon info update correctly.
   Expected Result: The UI updates properly with level-specific information.

5. Timeline _____/10

Phase	Task	Description	Estimated Time
Planning	Define Feature Requirements	Identify what the load level system should do.	1 day
Planning	Design Data Flow & UI Updates	Create DFDs, UI mockups, and system interactions.	1-2 days
Development	Design levels	Write code and place objects within unity to create basic environments	2-3 days
Development	Set Up Player Initialization	Position player correctly, set attributes.	1-2 days
Development	Implement Enemy Spawning	Load enemy positions, AI initialization.	2-3 days
Development	Develop Environment Setup	Load platforms, traps, and interactive objects, to expand the levels	2-3 days
Development	UI Integration & Display	Ensure UI updates correctly when the level loads.	1-2 days
Phase	Task	Description	Estimated Time
Testing	Unit Testing for Each Component	Test level loading, player spawn, enemy behavior, UI.	2-3 days
Testing	Integration Testing	Check all components work together smoothly.	2-3 days
Debugging	Fix Bugs & Optimize	Resolve errors, optimize loading times.	2-4 days
Finalization	Acceptance Testing	Ensure feature meets all functional requirements.	1-2 days
Total Time	Complete "Load Level" Feature	Final estimated duration	~2-4 weeks

Work items

Pert diagram



Gantt timeline