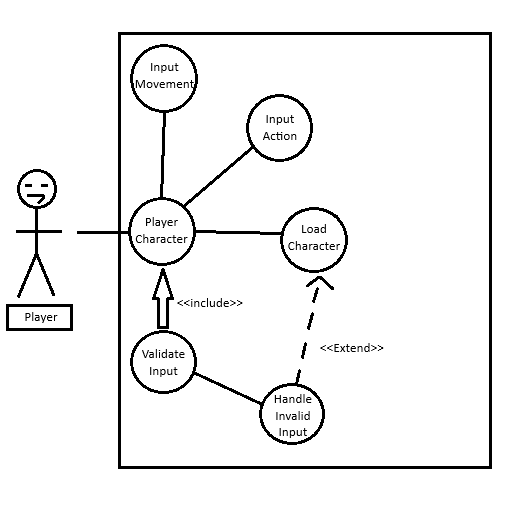
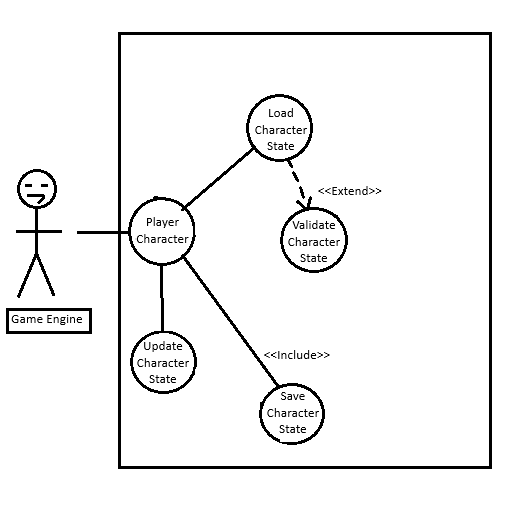
Name: *Gabriel Bybee* Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

**1. Brief introduction \_\_/3**

The feature I will be developing will enable the player to control the in-game character through keyboard and mouse input. The feature should use the inputs given in real-time to complete core gameplay functions such as movement, interactions, and fighting systems. This will ensure that gameplay is responsive and will mesh with animation and other game systems.

**2. Use case diagram with scenario**





**Scenarios**

Name: Control Player Character

Summary: The player uses input device(s) to control character movement and actions in the overall game environment.

Actors: Player

Preconditions: Game is running

Player Character is loaded and active within scene

Input system is set up and loaded correctly

Basic sequence:

Step 1: The player presses the movement key.

Step 2: System receives input and calls a validate input function.

Step 3: If Input is valid, send the input to the player controller.

Step 4: Character moves in the specified direction.

Step 5: The player presses the action key.

Step 6: System validates input, and executes animation / behavior.

Step 7: Character performs the action.

—-------------------------------------------------------------------------------------------------------------------------

Name: Load and Manage Player Character

Summary: The game engine loads the player character from saved data, validates said data, and either updates or saves this data during the gameplay.

Actors: Game Engine

Preconditions: Game is active.

Save File exists / default template is available for character.

The character system is set up and runs correctly.

Basic sequence:

Step 1: Game Engine starts a Load Character process

Step 2: System validates character data.

Step 3: If Input is valid, character is loaded into the game.

Step 4: During gameplay, the character state is updated via a save button

Step 5: On save, validate character data again.

Step 6: If character data is valid, state is saved from a save character state function.

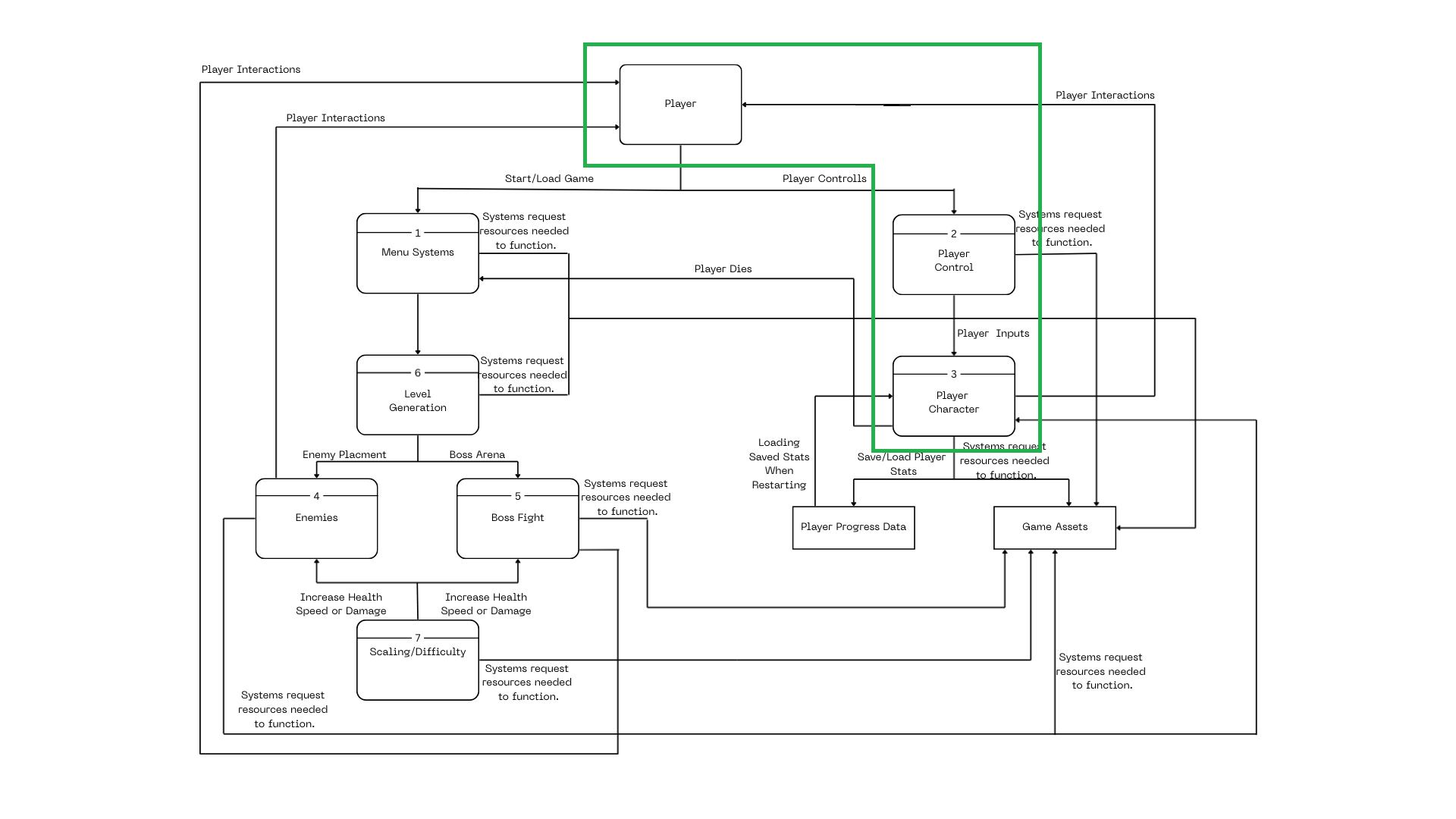
Exceptions:

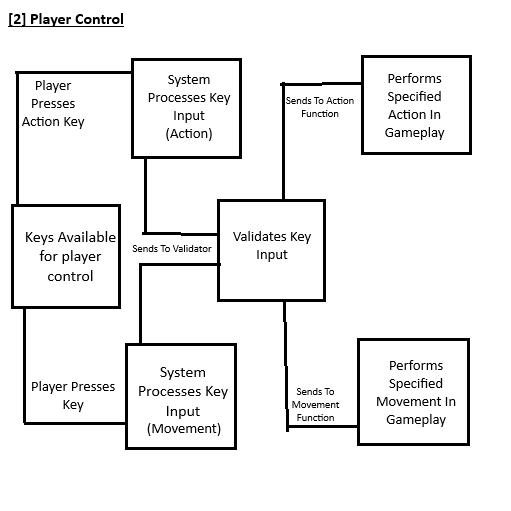
Character data is invalid / corrupted.

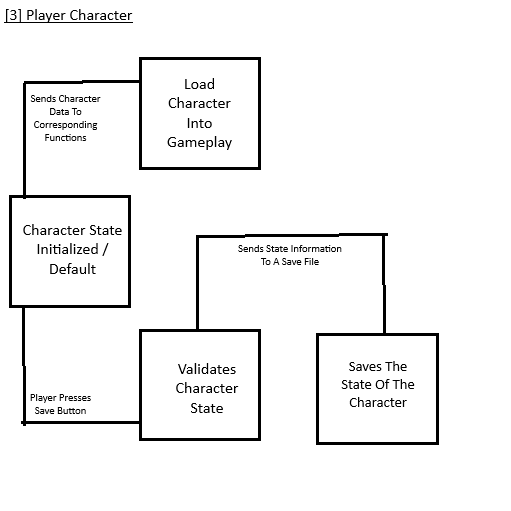
Save fails due to disk error.

**3. Data Flow Diagram(s) from Level 0 to Process Description**

Diagram 0:







Process Descriptions:

[2] The input keys are assigned and upon player interaction, are validated and then sent to the correct actions corresponding with the key(s) pressed, either Action or Movement.

[3] The State of the character is either loaded from a past session or has a default value(starting), then is loaded into the game. Upon saving via a button, the character state is validated and sent to a function / file that will save the current state.

**3. Acceptance Tests**

PLAYER CONTROLS

| **Output** | **Input(s)** | **Notes** |
| --- | --- | --- |
| Move forward | W Key Down | Character moves forwards |
| Move Backwards | S Key Down | Character moves backwards |
| Jump | Spacebar | Only if character is on ground |
| No action | Unsupported key | Input ignored |
| Attack | E Key Down | Should attack bare handed if no weapon. |
| Pause game | Escape key | Game enters pause state |
| Resume game | Escape key again | Game resumes from pause |

Boundary Cases:

Repeated input spamming -> system should drop excess commands.

Input while mid-animation -> Action either delayed or cancelled.

**5. Timeline**

| **Output** | **Input(s)** | **Notes** |
| --- | --- | --- |
| Character Loaded | Valid Save | Loads character with validated stats/ progress. |
| Default Character | Corrupted / No Save File | Loads the default player value |
| Save Successful | Valid State Of Character | Saves to file / disk; |
| Save Rejected | Invalid Character Data | Triggers validation failure |
| Retry Save | Invalid Character Data | Prompts a retry. |

Boundary Cases:

Save file missing -> Default character initiated

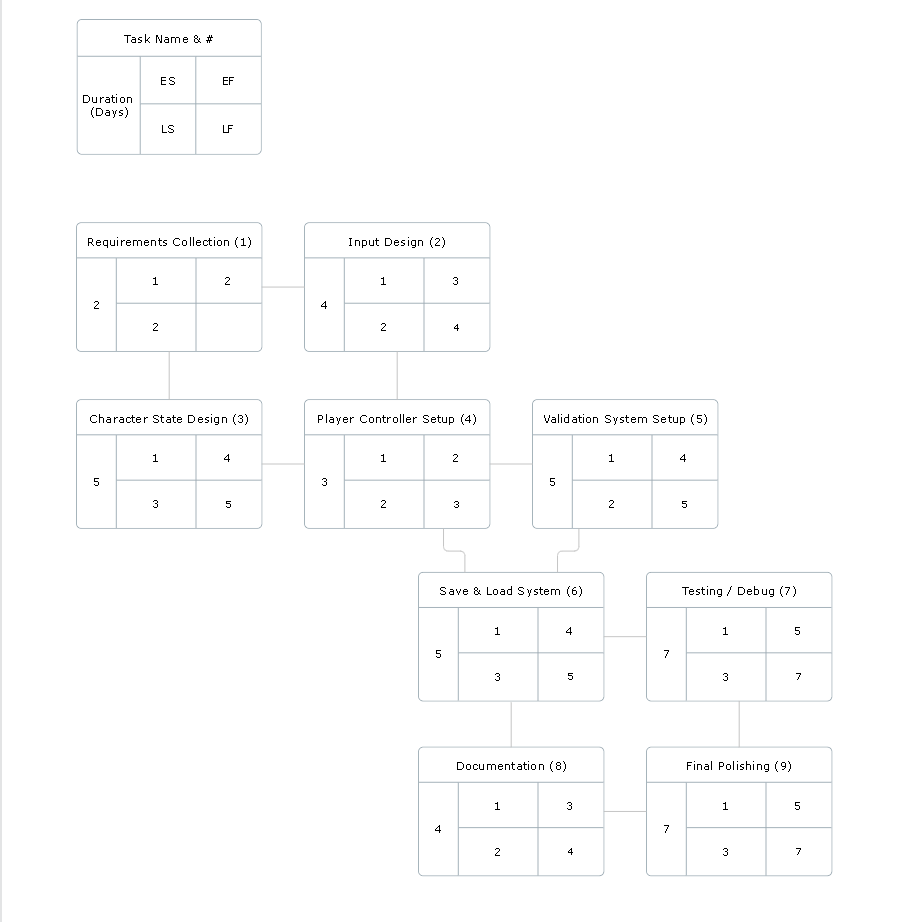
Inventory overflow -> extra items discarded

Health < 0 -> binded to minimum value

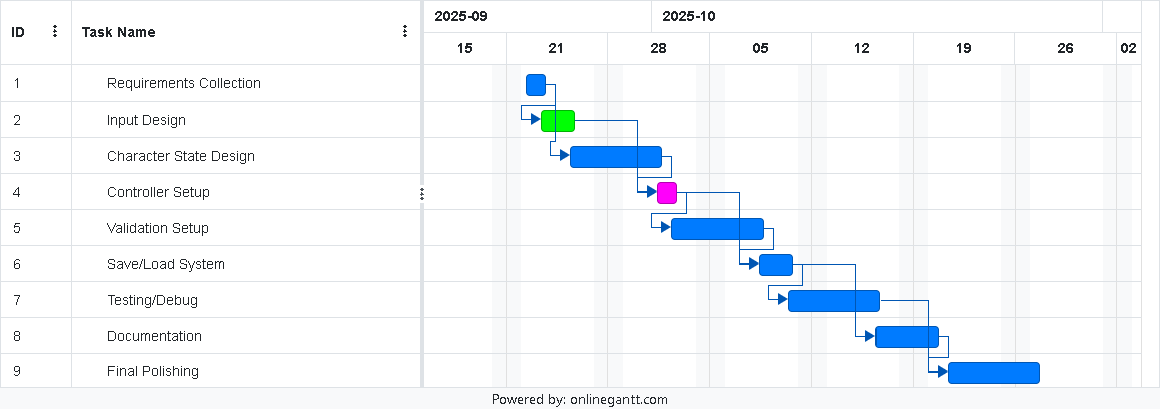
—-------------------------------------------------------------------------------------------------------------------------

| **Task** | **Duration (Days)** | **Predecessor Task(s)** |
| --- | --- | --- |
| 1. Requirements Collection | 2 | - |
| 2. Input Design | 4 | 1 |
| 3. Character State Design | 5 | 1 |
| 4. Controller Setup | 3 | 2, 3 |
| 5. Validation Setup | 5 | 4 |
| 6. Save/Load System | 5 | 4, 5 |
| 7. Testing & Debugging | 7 | 6 |
| 8. Documentation | 4 | 6 |
| 9. Final Optimization & Polish | 7 | 7, 8 |

PERT DIAGRAM



GANTT TIMELINE



(The days may be slightly off because the software wouldn’t let me work on weekends)