

# Cover Page

**Candidate Number:** 291105

**Game Title:** *Adventure World (Quest for the Queen)*

# **Problem Statement and Overview of the Game**

## **Problem Statement:**

This project aims to design and implement a text-based adventure game called Adventure World (Quest for the Queen). In order to save the Queen from the dragon, the player must battle opponents, gather supplies, and make his way through the castle till he reaches the dragon's lair, fights the dragon, and saves the queen.

## **Game Overview:**

The game begins with the player as a knight outside the magical castle. The player must:

- Navigate through 11 interconnected rooms.
- Pick up and use items like swords, shields, and health drinks.
- Fight soldiers and the dragon in combat scenarios.
- Solve easy puzzles to unlock hidden areas.

## **Win Condition:**

The game is won when the player defeats the dragon in its lair and rescues the Queen from captivity.

**End Goal:** Defeat the dragon and save the queen.

## **Number of Locations:**

The game features 11 locations/ rooms, including:

1. Garden
2. Outside
3. Entrance Hall
4. Dining Room
5. Armory
6. Library
7. Dungeon
8. Tower Room
9. Hidden Chamber
10. Queen's Quarters
11. Dragon's Lair

## **Instructions:**

### **How to Launch/ Play the Game?**

1. *Unzip the project folder.*
2. *Open the project in PyCharm (or your preferred IDE).*
3. *Run the game.py file.*
4. *Go to any given direction you want*
5. *Fight soldiers to get rewards (optional)*
6. *Wander through the rooms and Pick items to help you in your fight with the dragon*
7. *It's mandatory to pick the key to be able to go to Queen's quarter*
8. *Fight the dragon with fight and attack and heal commands*
9. *Fight soldiers with fight soldiers and attack and heal commands*

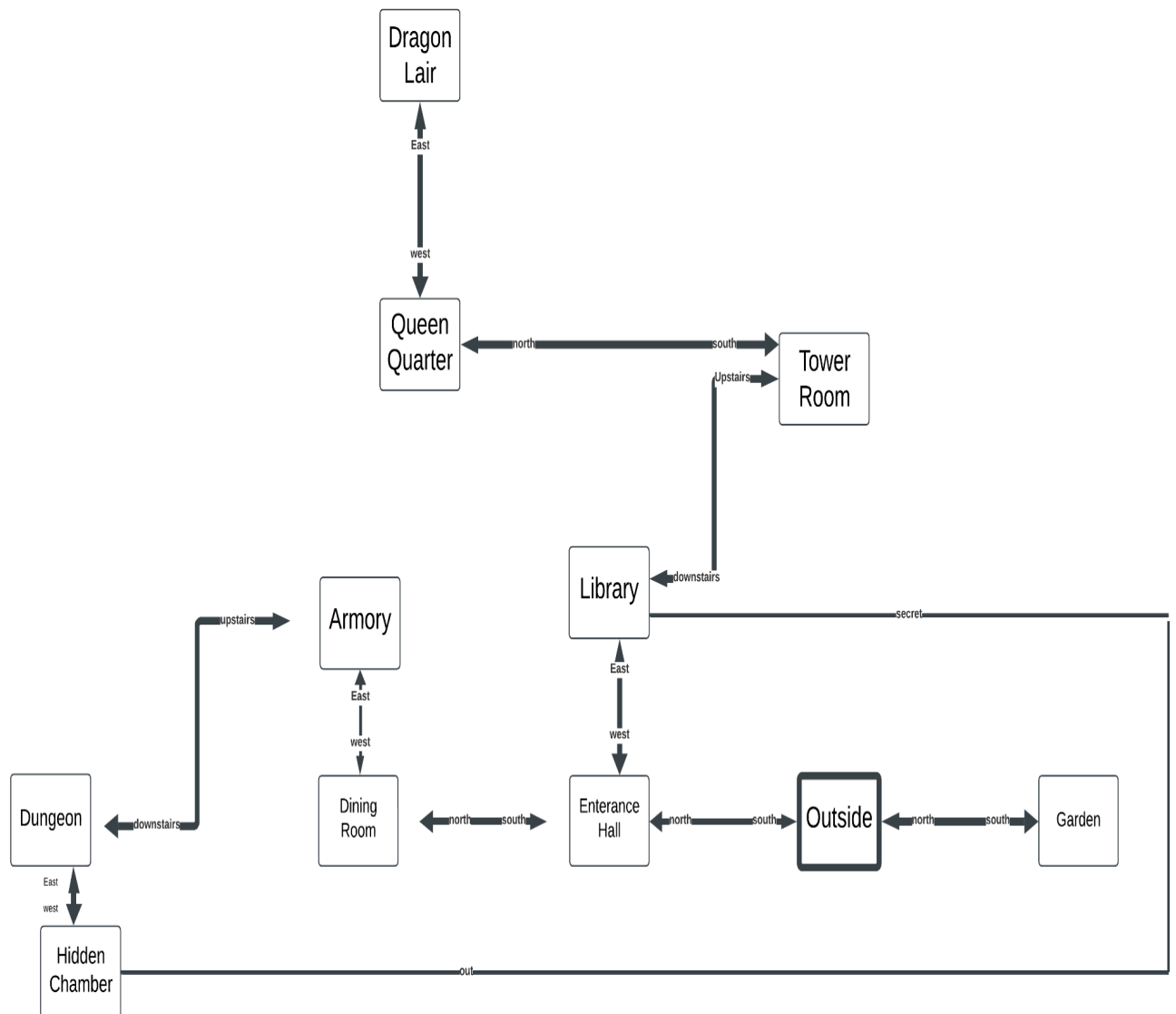
### **How to Play?**

**Use the provided commands to interact with the game world:**

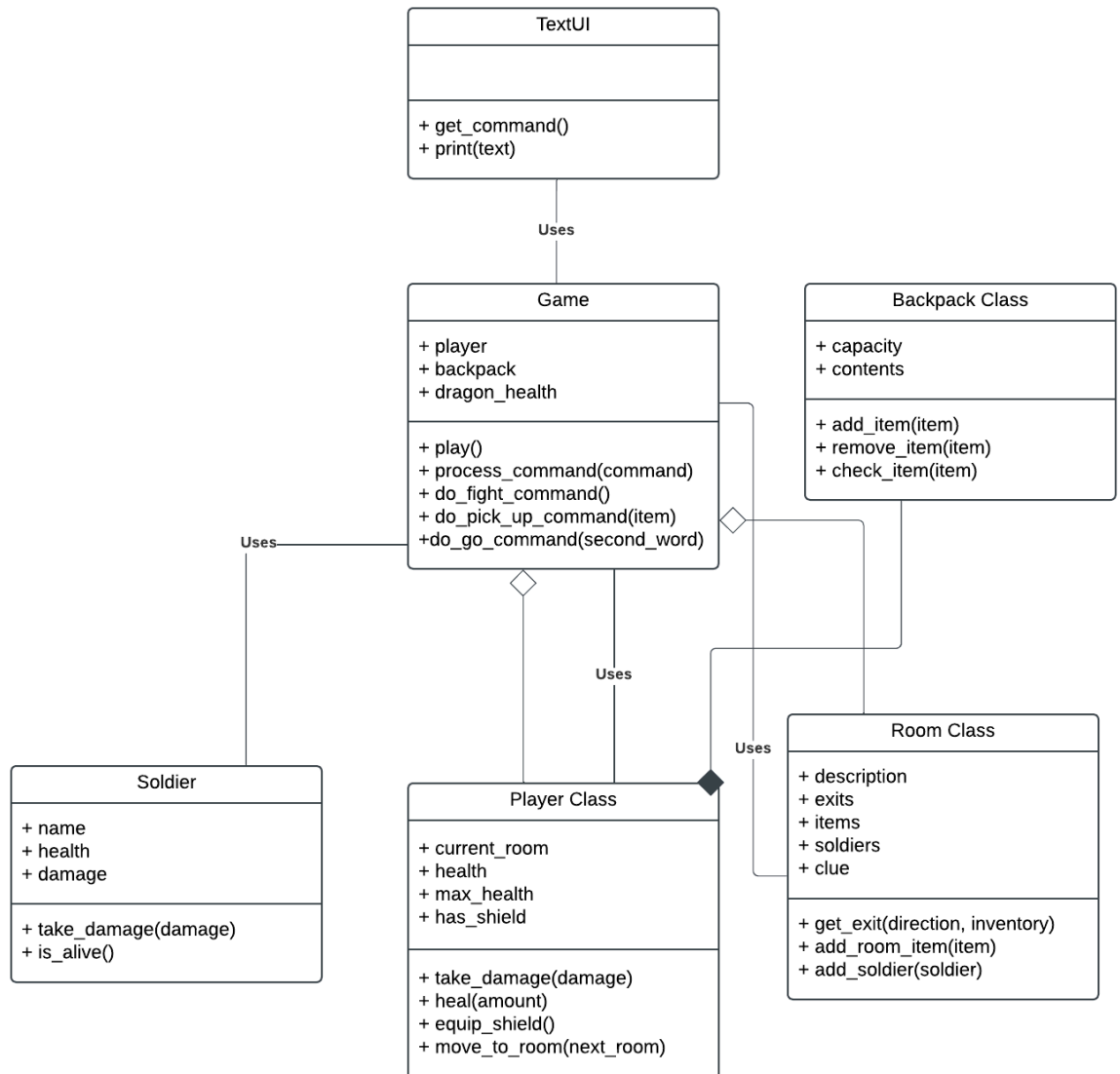
1. *Navigation: go [direction]*
2. *Pick-Up Items: pick [item]*
3. *Drop Items: drop [item]*
4. *Get Help: help*
5. *Read if there are any clues: read*
6. *Use Items: use [item]*
7. *Fight Soldiers: fight soldiers*
8. *Fight Dragon: fight*
9. *Heal/Attack soldiers or dragon during the combat : attack/heal*
10. *Solve Puzzles: solve*
11. *View Inventory: inventory*
12. *Look Around to see room contents: look*
13. *Quit the Game: quit*

## Map of the Game World:

The player begins the game outside the castle.



## 5. UML Class Diagram



## 6. Modifications to Starter Code

### Changes to Existing Classes:

- **Game Class:**
  - Added logging for player actions.
  - Created combat mechanics with choices (attack/heal).
  - Added new commands the player can use.
  - Expanded rooms to 11 locations the player can go to.
  - Added items to rooms the player can pick.
  - Added a dragon to fight at the end to rescue the queen.
  - Random damage impacts between 15 and 30 power for the dragon attacks.
  - Random rewards are given when fighting and winning against soldiers.
- **Room Class:**
  - Added clue, key item, and locked parameters.
  - Initialised values for rooms have a queen or dragon with false.
  - Added describe contents method that returns the contents of the room.
  - Created add, remove, and get room item methods to add items inside the rooms and remove them.
  - Created add, remove, and get soldier method.

### New Classes:

- **Soldier Class:**
  - Represents enemies in the game.
  - Includes attributes for health and damage.
  - Take damage method is created to receive damage.
  - Is alive method is created to indicate that soldier still has health.
- **Player Class:**
  - The player's starting point is outside the castle
  - The backpack capacity is 5 items.
  - Max health is 100

- Added take damage method that reduces damage if the player is wearing the shield.
- Added a shield attribute to enhance combat protection.
- Added heal method so the player can heal himself.

## **7. Interesting Features**

- Health bars for all game characters
- Random damage impacts between 15 and 30 power for the dragon attacks.
- Combat Mechanics:
  - Dynamic choices during battles (attack/heal) so the player can heal if he is closer to losing.
  - If the Player wins a battle against one of the soldiers, he will get a random reward (sword, Increased backpack capacity, health bag, etc)
- If the player picks up the shield, it reduces incoming damage and increases his health. It is not counted as an item inside the backpack as it's worn.
- A clue is created in the library to give the player a clue if there is an unseen secret room.
- When the player enters a room, its contents are shown and he can use the look command to see the contents to see if he wants to fight a soldier or pick something.
- If the Player picks the magic scroll it enhances the power of his sword

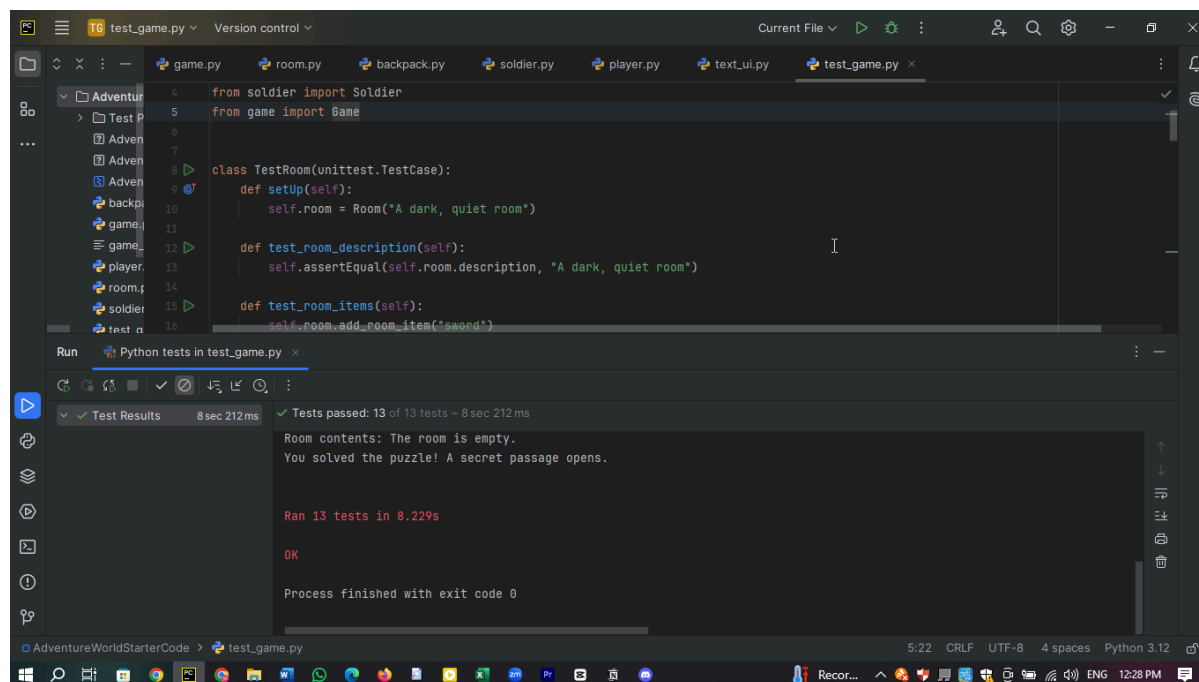
## **8. Problems Encountered**

- Command Case Sensitivity: I resolved this by normalizing all commands to lowercase.
- Dragon Combat: I created logic and adjusted it to allow players to win or lose fairly based on health and attacking strategy.

## 9. Evidence of Testing

### Testing Conducted:

- **Unit Testing:**
  - I did tests for all major classes (Room, Player, Backpack, etc).
  - I validated edge cases (e.g., inventory limits, missing items, locked rooms).
  - You can find a whole module test\_game.py in the code.



The screenshot shows a code editor with a file explorer on the left and a run console at the bottom. The file explorer shows a project structure with folders like 'Adventure' and 'Test P', and files like 'game.py', 'room.py', 'backpack.py', 'soldier.py', 'player.py', 'text\_ui.py', and 'test\_game.py'. The main editor window displays the code for 'test\_game.py', which includes imports for 'Soldier' and 'Game', and a 'TestRoom' class with methods 'setUp', 'test\_room\_description', and 'test\_room\_items'. The run console shows the output of the tests, indicating that 13 tests passed in 8.229 seconds.

```
4 from soldier import Soldier
5 from game import Game
6
7
8 class TestRoom(unittest.TestCase):
9     def setUp(self):
10         self.room = Room("A dark, quiet room")
11
12     def test_room_description(self):
13         self.assertEqual(self.room.description, "A dark, quiet room")
14
15     def test_room_items(self):
16         self.room.add_room_item('sword')
```

Run Python tests in test\_game.py

Test Results 8 sec 212 ms

Tests passed: 13 of 13 tests - 8 sec 212 ms

Room contents: The room is empty.  
You solved the puzzle! A secret passage opens.

Ran 13 tests in 8.229s

OK

Process finished with exit code 0



- **System Level Testing:**

- I tried and tested the player journey through the game to test end-to-end functionality as I used to work as a software tester.

The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer lists files: AdventureWorldStarterCode, game.py, room.py, backpack.py, soldier.py, player.py, text-ui.py, and test\_game.py. The main editor window displays the `do_go_command` method of a `Game` class. The code handles moving between rooms and printing room descriptions. The terminal shows the output of running the game, including health bars for the Knight and Dragon, and a victory message.

```
class Game:
    def do_go_command(self, second_word):
        elif next_room is None:
            self.ui.print("There is no door!")
            self.log(f"Attempted to go {second_word}, but no door exists.")
        else:
            self.log(f"Player moved from {self.player.current_room.description} to {next_room.description}.")
            self.player.current_room = next_room
            self.ui.print(self.player.current_room.get_long_description())

            # Show the contents of the room
            room_contents = self.player.current_room.describe_contents()
            self.ui.print(f"Room contents: {room_contents}")
```

Run game

```
> heal
Carrying on...
You used a health drink and restored 30 health.
The dragon breathes fire and deals 18 damage!
Knight Health: [██████████] 49/100
Dragon Health: [██████████] 40/200
What will you do? (attack / heal)
> attack
You strike the dragon!
You have defeated the dragon!
The Queen is safe! Congratulations, you win!
>
```