| Module Code | IY453 |
|---|---|
| Group | Group B |
| Module Title | Software Design and Implementation |
| Assessment Type | Coursework Stage 3 |
| Module Tutor Name | Jonathan Shore |
| ID Number | P420588 |
| Date of Submission | 16^th^March 2025 |

*I confirm that this assignment is my own work. *

*Where I have referred to academic sources, I have provided in-text citations and included the sources in the final reference list.*

**Contents**



# **1 **Introduction

In this project, I am developing a text-based adventure game inspired by One Punch Man using C++. The purpose of the game is to create an interactive experience where the player makes choices that impact the storyline, player stats and overall progression. The game will include scenarios such as combat, puzzles and item collection, with a scoring system and lives to track progress.

# **2 **Analysis and Design

## 2.1 ****Outline program specification.

The game will allow the player to interact with different scenarios by making binary choices that affects the outcome. The player will be able to solve puzzles, engage in combat and collect items that influences their stats.

## 2.2 ****Identify inputs, outputs, and processes.

Inputs: player’ s name – player’ s choices in scenarios – player’ s answers to puzzles.

Processes: managing the different types of scenarios – checking answers and updating the stats – handling combat system.

Outputs: scenario descriptions and choices – feedback on answers and choices – updated stats.

## **2.3 **Algorithms

```
RECORD Player
    name: String
    health: Integer
    attackPower: Integer
    defensePower: Integer
    score: Integer
    inventory: array of items
ENDRECORD

RECORD Enemy
    name: String
    health: Integer
    attackPower: Integer
    defensePower: Integer
ENDRECORD

RECORD Item
    name: String
    type: String
    effect: Integer
ENDRECORD
```

# **1 **Testing

## **1.1 **Test Table

| Test No | Item to Test | Test Description (with Example Data) | |
|---|---|---|---|
| 1 | cin >> name; | User enters "Alice" when prompted for name. | Program accepts |
| 2 | cin >> name; | User enters an empty name. | Program prompt |
| 1 | cin >> choice; | User enters "1" when prompted for a choice in run_scenario. | Program accepts |
| 2 | cin >> choice; | User enters "2" when prompted for a choice in run_scenario. | Program accepts |
| 3 | cin >> choice; | User enters an invalid choice "3" when prompted for a choice in run_scenario. | Program prompt |
| 4 | cin >> choice; | User enters "abc" (non-numeric characters) when prompted for a choice in run_scenario. | Program prompt |
| 5 | cin >> choice; | User enters "*" (non-numeric characters) when prompted for a choice in run_scenario. | Program prompt |
| 1 | cin >> answer; | User enters "apple" when prompted for a riddle answer. | Program accepts |
| 2 | cin >> answer; | User enters "orange" when prompted for a riddle answer. | Program rejects i |
| 3 | cin >> answer; | User enters "AnagramAnswer" (case-sensitive) for an anagram puzzle. | Program convert |
| 4 | cin >> answer; | User enters "42" when prompted for a math problem answer (correct answer). | Program accepts |
| 5 | cin >> answer; | User enters "50" when prompted for a math problem answer (incorrect answer). | Program detects |
| 6 | cin >> answer; | User enters a non-integer input like "ten" when prompted for a math problem. | Program prompt |

## **1.2 **Annotation Screenshots

## **1.3 **Technical Elements

| Name | Type | Purpose |
|---|---|---|
| Player | Class | Defines a blueprint for player objects, encapsulating attributes like health, attack power and inver |
| | Library | Provides input and output operations, such as cin and cout, to interact with the user. |

| Name | Type | Purpose |
|---|---|---|
| | Library | Allows the use of string data type and string manipulation functions like std::string. |
| | Library | Provides dynamic array functionality to store and manage collections of objects like Enemy, Scena |
| | Library | Provides functions for performing general tasks such as generating random numbers (rand()), me |
| | Library | Enables file input and output operations, used for reading from and writing to files. |
| | Library | Provides functionality for manipulating strings as streams, useful for string parsing and conversio |
| | Library | Contains algorithms like std::sort and std::find, used for manipulating collections. |
| Player | Class | Defines the blueprint for the Player object with attributes and methods to manage player stats an |
| player_name | Attribute | Stores the name of the player. |
| player_health | Attribute | Stores the player's health |
| player_attackPower | Attribute | Stores the player's attack power. |
| vector inventory | Attribute | Stores the player's inventory as a dynamic list of items (strings). |
| string getName() | Method | Returns the player's name. |
| void setHealth(int health) | Method | Sets the player's health to a specified value. |
| void addItem(string item) | Method | Adds an item to the player's inventory and outputs a message. |
| void displayStats() | Method | Displays the player's current stats (name, health, attack power, defense power, score, lives). |
| Enemy | Class | Defines a blueprint for enemy objects, encapsulating attributes like name, health, attack power, a |
| enemy_name | Attribute | Stores the name of the enemy. |
| int getAttackPower() | Method | Returns the attack power of the enemy. |
| void setHealth(int) | Method | Sets the health of the enemy. |
| | | |
| | | |

## **2 **References

## **3 **Appendix

plaintext

```
#include <iostream>
#include <string>
#include <vector>
#include <cstdlib>
#include <fstream>
#include <sstream>
#include <algorithm>

using namespace std;

class Player {

private:
    string player_name;
    int player_health;
    int player_attackPower;
    int player_defensePower;
    int player_score;
    int player_lives;
```

```
vector<string> inventory;
```