**Saint Augustine’s College, Sydney**

**Software Engineering Year 11: Programming Fundamentals**

**Cruz Leung Project “assessment.py”**

Table of Contents

[Planning 3](#_Toc150971422)

[Task Definition 3](#_Toc150971423)

[Structure Chart 4](#_Toc150971424)

[Flowchart 5](#_Toc150971426)

[Data Dictionary 6](#_Toc150971427)

[Implementation 7](#_Toc150971428)

[GitHub Repository 7](#_Toc150971429)

[Project Reflection 8](#_Toc150971436)

# Planning

## Task Definition

I have been assigned the task of developing ‘assesseent.py’ (also known as ultimate knowledge quiz), a command-line driven Python application designed for an engaging and competitive quiz experience suitable for all ages in high school or above. The goal is to create an intuitive and easy-to-use quiz game that balances fun, randomness and challenge while maintaining a smooth user experience for a productive learning experience.

The functional requirements I will need to implement include:

* A question system that includes multiple categories: Random, Sports, and Everyday
* A scoring system that tracks correct answers and rewards players accordingly.
* A hint function to assist players when they are stuck
* A leaderboard to increase competitiveness by displaying top 5 high scores and rankings.
* Randomized questions to enhance and engagement and ability to replay.
* A user-friendly command-line interface that makes playing the quiz simple and enjoyable.
* The ability to start a new quiz session once the current game concludes.

## Structure Chart

As I will be taking a functional approach to the development of ‘assessment.py’, it is appropriate to create a structure chart that will decompose the game logic into a mainline and the individual functions within, and help visualise the data/parameters that will be passed around. The following structure chart maps out the functions within my program, a simple run down is here:

A diagram of a company

AI-generated content may be incorrect.

*This diagram was generated using* [*.drawio*](https://www.drawio.com/)*. It can be viewed as a template* [*here*](https://drive.google.com/file/d/1uzQsjF8thjtgjTTYEHFJa-khEq4BfrPz/view?usp=sharing)*.*

**Main** will be the top-level mainline that starts the game.

**Welcome page** will be a function that welcomes the user and creates a banner

**Choose Category** will be the function where the user chooses what category of questions they want to play, it is also looped through after each section. Leaderboard is also displayed in this function when they are no categories remaining for the user to play

**Print questions** will be the main question function. It displays the questions and the choices depending on the user’s chosen category. It is then broken down in the following sequences:

* **Get user input** simply asks user for their answer of the question
* **Check for hint** will be used to check if user entered “hint” in the input get a hint displayed
* **Check user answer** simply checks if user has entered the correct answer
* **Apply and display consequences** will be a set of consequences that will apply to user depending on user input (e.g. score, hint, displaying retry attempts remaining)
* **Update section score to main score** simply adds the score tally to main after each section. Then loops through print questions until all categories are played

## Flowchart

This algorithm's logic can be effectively illustrated through a flowchart to visually augment comprehension. While the detailed operations of the subfunctions are simplified, this overview should adequately convey the workings of my quiz.

A diagram of a flowchart

AI-generated content may be incorrect.

*This diagram was generated using* [*.drawio*](https://www.drawio.com/)*.*

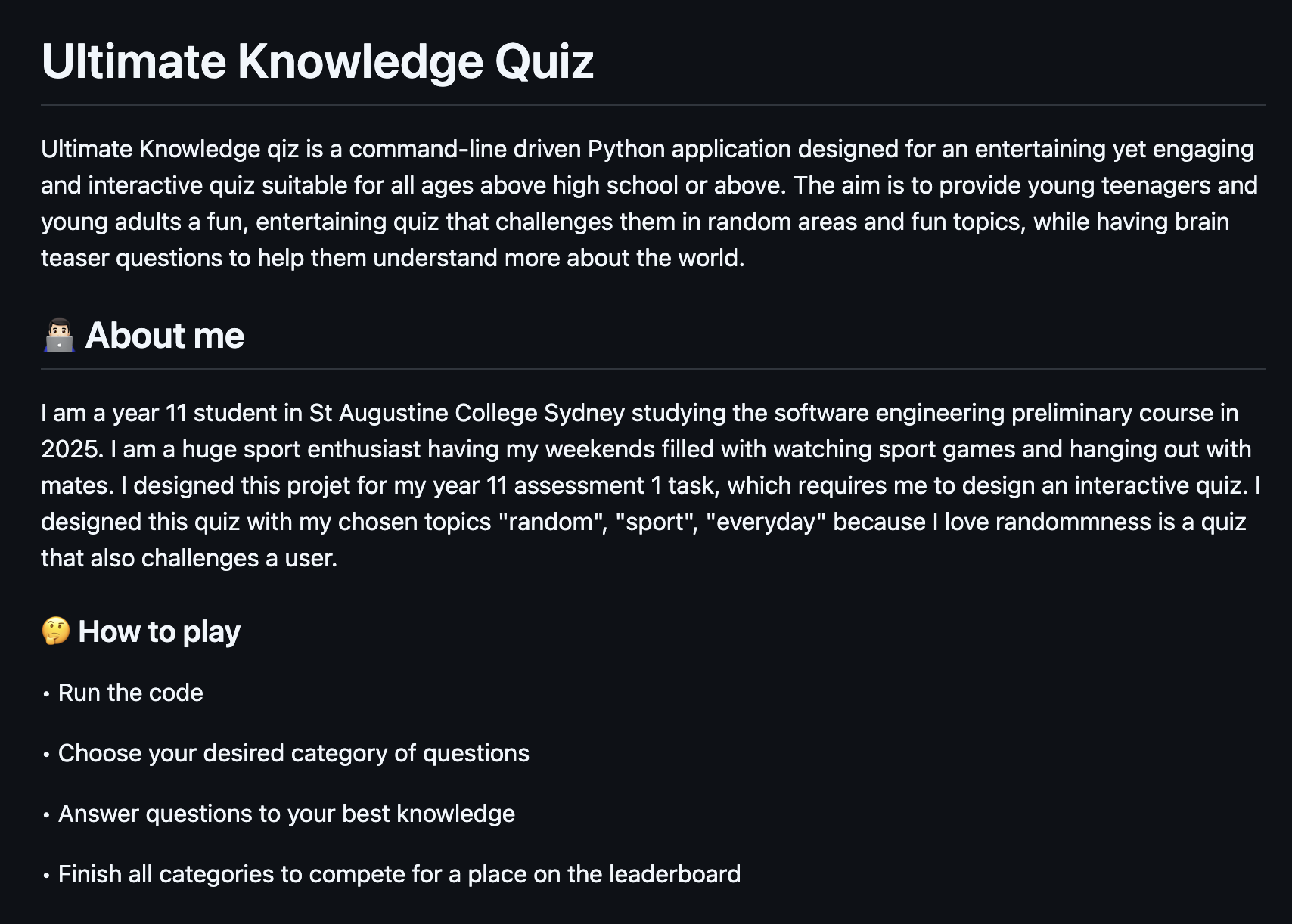
## Data Dictionary

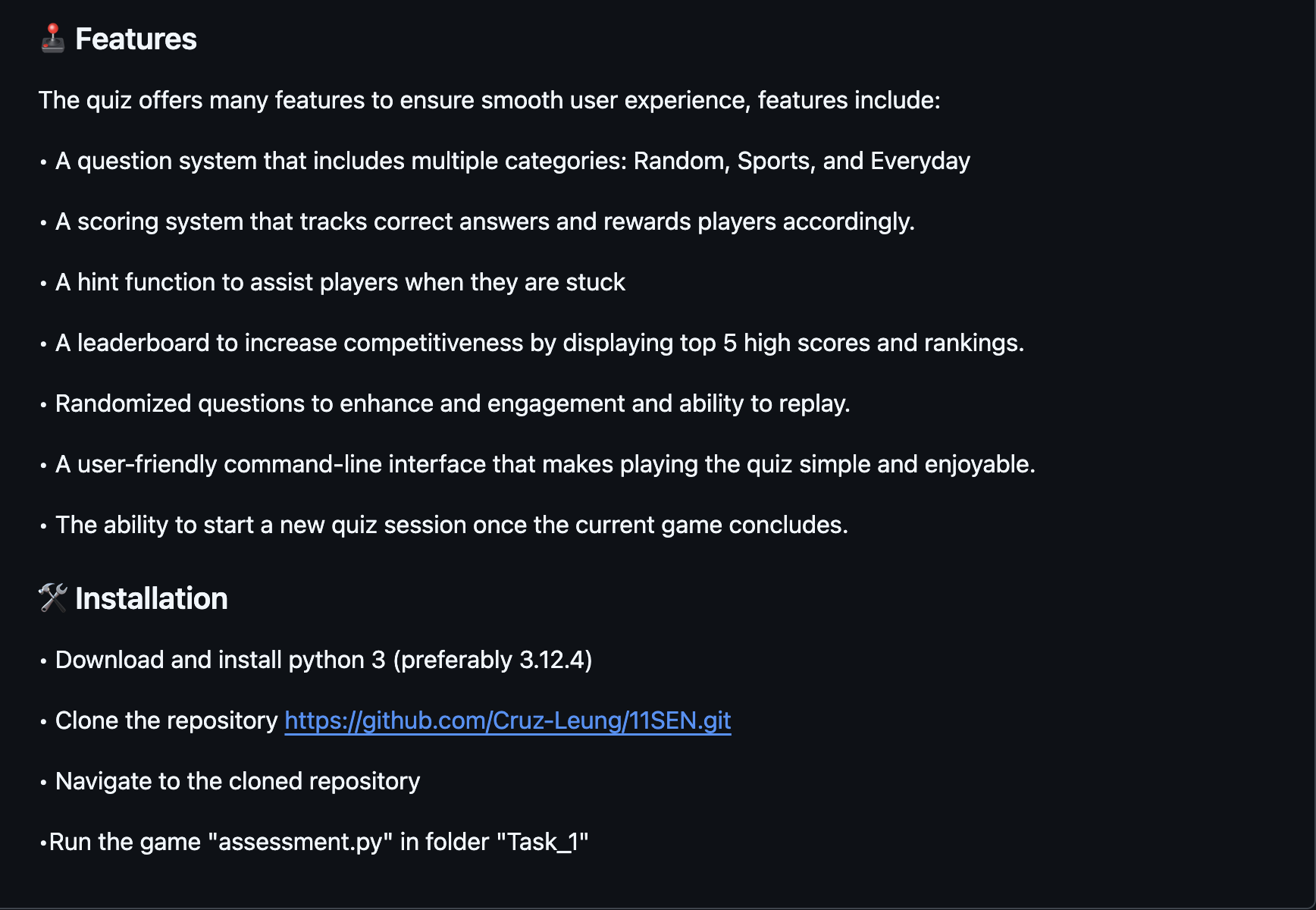
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Variable | Data Type | Format for display | Description | Example | Validation |
| score | Integer | Numeric | Accumulates the score of the player throughout the all sections of the quiz | score = 2 | Can’t be below 0 |
| section\_score | Integer | Numeric | Accumulates the score of the player in the specific section of questions. | section\_score = 1 | Can’t be below 0 and over 12 |
| name | String | Text | Name that user inputted | name = Peter | Input cannot be empty |
| categories | List of strings | Strings | Used as a tool to display all categories correctly without duplicates | categories = [“random”, “everyday”] | Cannot be topics that are not in categories |
| played\_categories | List of strings | Strings | Categories that the user has already played | played\_category = [“sport”, “everyday” ] | Cannot be topics that are not in categories |
| hint\_count | integer | Numeric | Count to keep the hint function functional and accurat. (Displays different hints depending on counts) | hint\_count = 3 | Can’t be below 0 |
| incorrect\_attempt | integer | Numeric | Keeps count of user’s incorrect attempts per question. | Incorrect\_attempt = 3 | Can’t be below 0 and over 3 |
| imported\_question | Dictionary | Dictionary of strings | Question bank | Imported\_question = ["Topic": "sport", "Question": "What is the national sport of Japan?"] | Dictionary data doesn’t change |

# Implementation

## GitHub Repository

<https://github.com/Cruz-Leung/11SEN>





*This GitHub README.md was created using* [*https://readme.so/*](https://readme.so/)

# Project Reflection

My planning phase of the algorithm started with direct coding as that is my coding preference with my past projects. However, I was met with challenges and problems as the code size increased. This assessment has helped me understand that planning with tools, such as structure chart, and flowchart should be done first to improve efficiency and have a better structured approach to my code.

At the start of my coding journey for this project, I was met with difficulties with array manipulation, loops and iterating through my question bank. I also encountered issues with my category chooser as I kept getting errors and problems with displaying the correct categories without duplicated topics and played categories. Overtime, I started to be more familiar with the language and the indexing system, particularly in utilizing multiple loops and conditions to iterate through arrays and dictionaries. By using physical paper to draw out loops and conditionals, I was able to simplify the process and understand how the loops worked to iterate through the arrays efficiently.

The GitHub repository management was a rewarding aspect of the project, particularly with the utilization of readme.io. This provided me a good base for efficient documentation throughout the project, which gave me a good, structured idea on how I wrote my code before, improving efficiency on future bug fixes and patch notes on earlier codes. However, I only began to develop my flowchart and structure chart at the end of the code, which I found quite challenging as I had to break down complex logics and loops utilized in my code, rather than having a general concept of the functions I would have to use if I started doing my charts before coding. This project has provided me more experience and developed my understand and skill of planning and documentation on charts and data structures.

My proficiency and skill in Python has advanced and developed considerably through this first project. I have mastered the structure of a basic game loop and the method of breaking down complex problems into manageable segments. With an understanding of object-oriented programming principles, I am looking forward to exploring beyond the confines of a functional approach next term.