PROJECT REPORT

Python Quiz Game

Submitted by:-
Student Name: Animesh Singh
Roll Number: 25BCE11355
Department of Computer Science
Submitted to: Dr. D. Saravanan
Department of Computer Science
University Name: VIT Bhopal University
Date: November 23, 2025

[TABLE OF CONTENT]

[CLI Quiz Game]

1. Introduction

This project is a text-based quiz game made with Python. It works like a TV trivia show. The computer asks you questions, and you answer by typing numbers. It keeps track of your score while you play. The game runs fast and helps you learn new things.

2. Problem Statement

We need simple tools for learning and fun. Building applications with fancy graphics (GUIs) is often too complicated and slow just for a simple quiz. This project solves that problem by creating a text-based tool. It runs easily in the command line and doesn't require you to install any extra software.

3. Functional Requirements

The system fulfills the following functional requirements:

1. **Question Display:** The system must display a question along with four numbered options.
2. **Input Handling:** The system must accept an integer input (1-4) representing the user's choice, or '0' to quit.
3. **Validation Logic:** The system must compare the user's input against the stored correct answer index.
4. **Scoring System:** The system must add points (10 points) for every correct answer and maintain a cumulative score.
5. **Game Termination:** The game must end if the user selects the wrong answer, chooses to quit, or completes all questions.

4. Non-functional Requirements

- **Usability:** The interface should be text-based and easy to navigate using a standard keyboard.
- **Performance:** Response to user input should be instantaneous.
- **Portability:** The script should run on any system with a standard Python 3.8 interpreter installed.
    - **Maintainability:** The code structure should allow for easy addition of new questions.

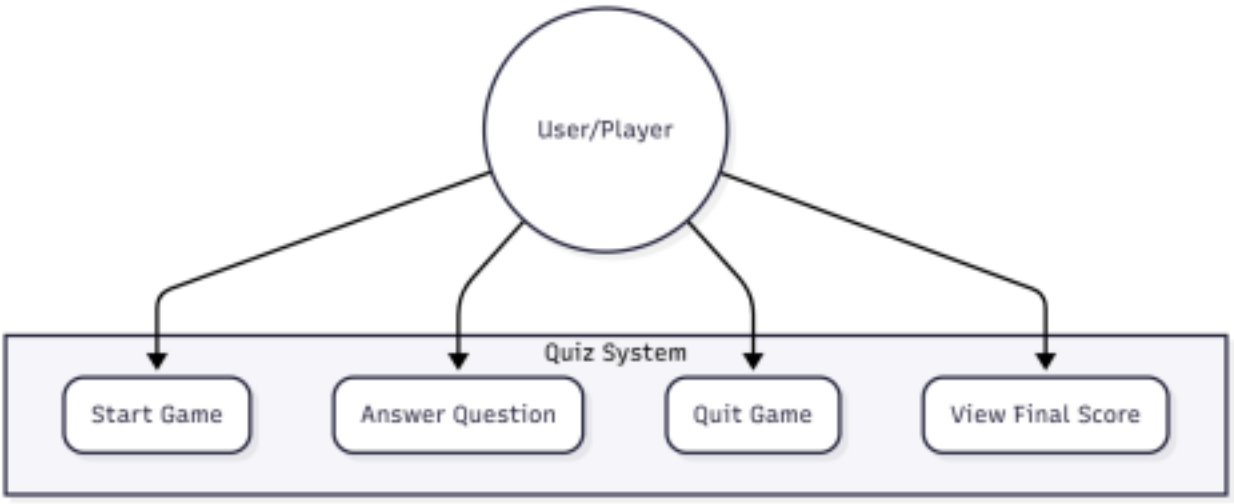
5. System Architecture

The application follows a procedural architecture typical of simple scripts. It utilizes a monolithic design where data (questions) and logic (game loop) reside in a single module.

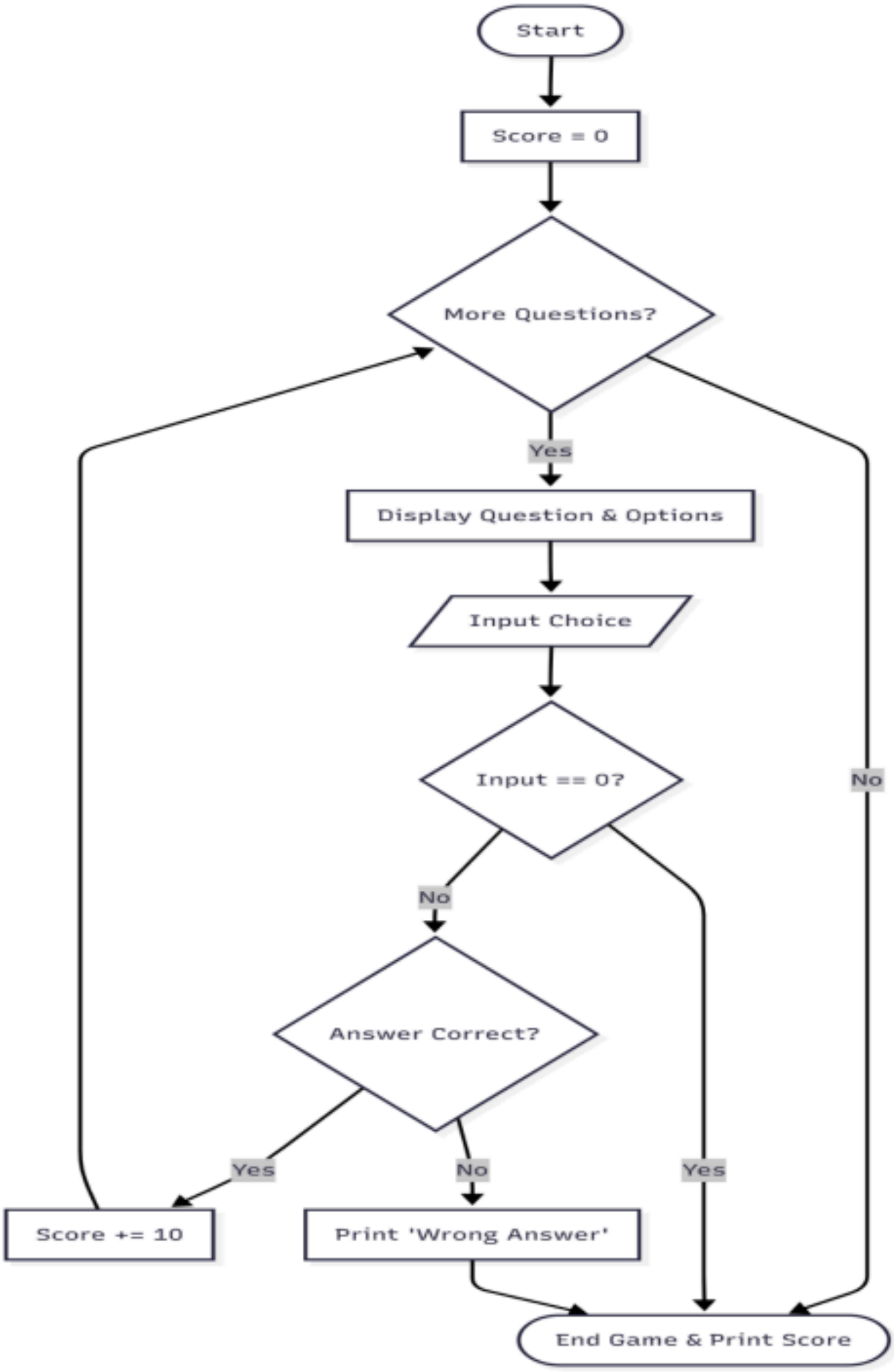
- **Data Layer:** A List of Lists structure stores the questions, options, and the index of the correct answer.
- **Logic Layer:** A for loop iterates through the data, handles I/O, and updates the state (score).
- **Presentation Layer:** Standard Output (print) and Standard Input (input) function as the UI.

6. Design Diagrams

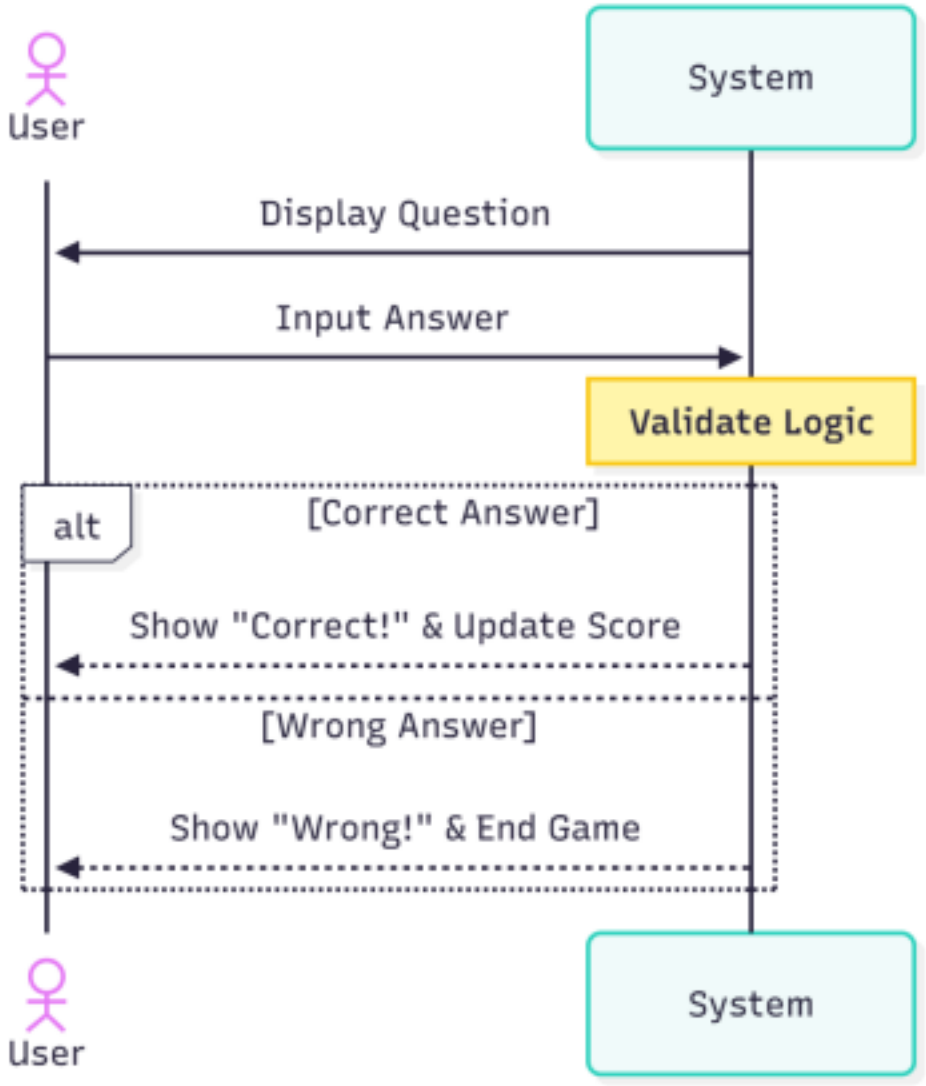
Use Case Diagram



Workflow Diagram (Flowchart)


Start
Score = 0
More Questions?
Yes
No
Display Question & Options
Input Choice
Input == 0?
No
Yes
Answer Correct?
Yes
No
Score += 10
Print 'Wrong Answer'
End Game & Print Score

Sequence Diagram



Data Structure Schema

Since this project uses in-memory lists rather than a database, an Entity-Relationship diagram is replaced by a Data Structure Schema.

List Structure: [Question, Opt1, Opt2, Opt3, Opt4, Correct_Index]

Example Data:

- Row 1: ["Capital?", "Mumbai", "Delhi", ..., 2]
- Row 2: ["5+5?", "10", "11", ..., 1]

7. Design Decisions & Rationale

- **Choice of Language:** Python was chosen for its clean syntax and ease of handling string manipulations and lists.
- **Data Structure:** A List of Lists was selected to store questions. This eliminates the need for external file parsing (CSV/JSON) for a small dataset, keeping the project self-contained.
- **Index-based Answers:** The correct answer is stored as an integer index (1-4) rather than

the string text. This simplifies the validation logic by allowing direct numerical comparison.

8. Implementation Details

The core logic utilizes a for loop to iterate through the list of questions. The input() function captures user choices, which are type-cast to integers.

```
questions = [
   ["Which language was used to create fb?", "Python", "French", "JavaScript", "Php", 4],
   ["What is the capital of India?", "Mumbai", "Delhi", "Chennai", "Kolkata", 2], ["What is 5
   + 5?", "10", "11", "25", "55", 1],
   ["Which planet is known as the Red Planet?", "Earth", "Mars", "Venus", "Jupiter", 2],
["Which language was used to create fb?", "Python", "French", "JavaScript", "Php", 4] ]

score = 0

# Loop through the questions
for i in range(0, len(questions)):
   q = questions[i]
   print()
   print("Question", i+1)
   print(q[0])
   print("1.", q[1])
   print("2.", q[2])
   print("3.", q[3])
   print("4.", q[4])

   # Asking for input
   ans = int(input("Enter choice (1-4) or 0 to quit: "))

   if ans == 0:
      print("Quitting game")
      break
   if ans == q[5]:
      print("Correct Answer!")
      score = score + 10
   else:
      print("Wrong Answer!")
      break

print("Final Score:", score)
```

9. Screenshots / Results

Below is a simulation of the program output in the console.

Scenario 1: Correct Answer

Question 1
Which language was used to create fb?
1. Python
2. French
3. JavaScript
4. Php
Enter choice (1-4) or 0 to quit: 4
Correct Answer!

Scenario 2: Wrong Answer & Game Over

Question 2
What is the capital of India?
1. Mumbai
2. Delhi
3. Chennai
4. Kolkata
Enter choice (1-4) or 0 to quit: 3
Wrong Answer!
Final Score: 10

10. Testing Approach

The application was tested using White Box testing methods.

- **Positive Testing:** Entered correct answers for all questions to ensure the score reaches the maximum (50).
- **Negative Testing:** Intentionally entered wrong answers to verify the "break" loop functionality works.
- **Boundary Testing:** Entered '0' to test the immediate quit functionality.

11. Challenges Faced

● **Input Handling:** Initially, string inputs caused crashes. Converting inputs to int requires the user to be careful not to enter letters (Future improvement: Try-Except blocks). ● **Data Consistency:** Ensuring the correct answer index (last element of the list) actually matched the correct option position.

12. Learnings & Key Takeaways

- Learned how to iterate over multi-dimensional lists in Python.
- Understood the importance of Type Casting (int(input())).
- Gained experience in designing control flow using break statements.

13. Future Enhancements

1. **Error Handling:** Add try-except blocks to prevent crashes if non-integer data is entered. 2.

Randomization: Use the random module to shuffle questions so the order isn't predictable.

3. **File I/O:** Load questions from a JSON or text file instead of hardcoding them in the script.
4. **Timer:** Implement a countdown timer for each question to increase difficulty.

14. References

- Python 3.10 Documentation - https://docs.python.org/3/
- Class Lectures on Python Data Structures.
- Wikipedia - "Who Wants to Be a Millionaire?" game format.