**A PROJECT REPORT**

**ON**

**“QUIZ GAME”**

# **(Final Report)**

**BY:-** ANIL KUMAR

**Domain:-** PYTHON

## DECLARATION

I, **ANIL KUMAR**, hereby declare that the presented report of the internship titled “**QUIZ GAME”** is uniquely prepared by me after the completion of one months’ work at Unicoverge Technologies Pvt. Ltd.

I also confirm that the report is only prepared for my academic requirement, not for any other purpose. It might not be used in the interest of the opposite party of the corporation.

**A close up of a name

Description automatically generated**

Signature of Student

**TABLE OF CONTENTS**

|  |
| --- |
| 1. Abstract |
| 2. Introduction |
| 2.1 Background |
| 2.2 Purpose |
| 2.3 Scope |
| 3. Quiz Game Implementation |
| 3.1 Reading Quiz Data from CSV |
| 3.2 Displaying Questions and Options |
| 3.3 Collecting User Answers |
| 3.4 Calculating User Score |
| 4. Program Structure and Functionality |
| 4.1 `load\_quiz\_data(file\_path)` Function |
| 4.2 `display\_question(question)` Function |
| 4.3 `collect\_answer()` Function |
| 4.4 `calculate\_score(quiz\_data, user\_answers)` Function |
| 4.5 `main()` Function |
| 5. Results and Output |
| 6. Conclusion |
| 7. Future Enhancements |
| 8. SUBMISSION LINK |
|  |
|  |
|  |
|  |

## ABSTRACT

The quiz game is a Python program that allows users to participate in a quiz by answering a series of questions. The program reads quiz data from a CSV file, displays each question along with options, collects the user's answers, and calculates the final score based on the correct answers provided by the user. The primary components of the quiz game are:

**1. load\_quiz\_data(file\_path):** This function reads the quiz data from the specified CSV file and returns a list of dictionaries, where each dictionary represents a question with its options and the correct answer.

**2. display\_question(question):** This function takes a question as input and displays it along with its options (a, b, c, d) for the user to select an answer.

**3. collect\_answer():** This function prompts the user to enter their answer and ensures that the input is valid (a, b, c, or d). It loops until a valid option is provided by the user.

4. calculate\_score(quiz\_data, user\_answers): This function takes the quiz data and the user's answers as inputs and calculates the user's score by comparing their answers to the correct answers in the quiz data.

**5. main():** The main function of the program that orchestrates the entire quiz game. It loads the quiz data, presents each question to the user, collects their answers, calculates the score, and finally displays the user's score at the end of the quiz.

The program provides an interactive experience for the user, allowing them to test their knowledge by answering questions. The quiz data is stored in a CSV file, making it easy to update or modify the questions and options without modifying the code.

INTRODUCTION

1. **Introduction**

Welcome to the Quiz Game Report! This section provides an overview of the quiz game project, detailing its background, purpose, and scope.

**2.1 Background**

In the digital age, education and learning have evolved beyond traditional classroom settings. Educational games and interactive quizzes have emerged as effective tools to engage learners, promote knowledge retention, and foster a fun and interactive learning experience. The motivation behind developing this quiz game lies in harnessing the power of gamification to make learning enjoyable and accessible to a wide audience.

The prevalence of technology in daily life has made learning on digital platforms increasingly popular. By creating a quiz game, we aim to capitalize on this trend and offer an innovative approach to educational assessment. By combining the allure of gaming with the pursuit of knowledge, we strive to cater to diverse learning styles and engage users of all ages in a dynamic and intellectually stimulating manner.

**2.2 Purpose**

The primary purpose of the quiz game project is to provide an interactive and entertaining platform for users to test their knowledge across various subjects. Our goals include:

**Engaging Learning Experience:** By presenting questions in a quiz format, we intend to create an enjoyable learning experience that encourages users to actively participate and explore new topics.

**Knowledge Assessment:** The quiz game will serve as a tool to assess users' understanding of different subjects, reinforcing key concepts and identifying areas for improvement.

**Versatility:** The quiz game will be designed to accommodate a wide range of topics, making it suitable for students, educators, and knowledge enthusiasts alike.

Simplicity and User-Friendliness: We aim to create an intuitive and user-friendly interface, allowing players to focus on answering questions without unnecessary complexities.

**2.3 Scope**

The scope of the quiz game project covers the implementation of a command-line-based application using Python programming language. The key features of the quiz game will include:

**Quiz Data Management:** The game will read quiz data from a CSV (Comma-Separated Values) file, allowing easy modification and expansion of question sets.

**Question Presentation:** Users will be presented with multiple-choice questions, each with four options (a, b, c, or d).

**User Interaction:** Participants will be able to select their answers by entering the corresponding option letter.

**Scoring System:** The quiz game will automatically evaluate the user's responses and calculate their final score based on the correct answers provided.

**Error Handling:** The application will incorporate robust error handling to ensure a smooth and reliable user experience.

Please join us on this exciting journey as we delve into the creation and implementation of the quiz game, aiming to promote learning through gamification.

Quiz Game Implementation

**3. Quiz Game Implementation**

In this section, we will delve into the technical aspects of implementing the quiz game. The quiz game's functionality is divided into four key components:

**3.1 Reading Quiz Data from CSV**

To efficiently manage the quiz questions and answers, we utilize a CSV (Comma-Separated Values) file format to store the quiz data. The load\_quiz\_data(file\_path) function is responsible for reading the data from the CSV file and organizing it into a list of dictionaries. Each dictionary represents a single question and contains the question text, four options (option a, option b, option c, option d), and the correct option (correct\_option) among them.

**3.2 Displaying Questions and Options**

The display\_question(question) function plays a pivotal role in presenting the quiz questions to the user. It takes a question dictionary as input and displays the question text along with the available options (labeled as 'a', 'b', 'c', and 'd') for the user to choose from. The user interface is kept simple and straightforward to enhance the user experience.

**3.3 Collecting User Answers**

The user interaction is facilitated by the collect\_answer() function. This function prompts the user to enter their answer by typing the corresponding letter ('a', 'b', 'c', or 'd') representing their chosen option. The function ensures that the input is valid and loops until the user provides a valid answer. This enables seamless interaction with the quiz game and prevents incorrect inputs.

**3.4 Calculating User Score**

The final component of the quiz game's functionality is calculating the user's score. The calculate\_score(quiz\_data, user\_answers) function is responsible for comparing the user's answers, stored in the user\_answers list, with the correct options from the quiz\_data. For each correct answer, the user's score is incremented. At the end of the quiz, the function returns the user's total score.

These four essential components work together seamlessly to create an interactive quiz game. Users can go through a series of questions, select their answers, and receive their final scores at the end of the quiz. The implementation ensures a user-friendly and enjoyable experience, encouraging users to test their knowledge and learn in a gamified manner.

# Program Structure and Functionality

**4. Program Structure and Functionality**

In this section, we will explore the program's structure and the functionality of the key functions involved in the quiz game implementation.

**4.1 load\_quiz\_data(file\_path) Function**

The load\_quiz\_data(file\_path) function serves as the foundation of the quiz game by reading the quiz data from a CSV file. It takes the file\_path parameter as input, which represents the path to the CSV file containing the quiz questions and options. The function utilizes Python's built-in csv module to handle CSV file reading efficiently.

Upon successful execution, the function returns a list of dictionaries, with each dictionary representing a single question and its related information. The structure of the dictionary is as follows:

**code**

**{**

**"question": "What is the capital of France?",**

**"optiona": "London",**

**"Option b": "Paris",**

**"Option c": "Berlin",**

**"Option d": "Madrid",**

**"Correct\_option": "b"**

**}**

**4.2 display\_question(question) Function**

The display\_question(question) function is responsible for presenting a single quiz question to the user and displaying the available options for them to choose from. It takes a single parameter, question, which is a dictionary representing the current question.

The function extracts the question text, options (option a, option b, option c, and option d), and then prints them in a user-friendly format. The user is presented with the question and the options, labeled as 'a', 'b', 'c', and 'd', and prompted to select their answer.

**4.3 collect\_answer() Function**

The collect\_answer() function facilitates user interaction by prompting the user to enter their answer. It ensures that the user's input is valid and loops until a valid option ('a', 'b', 'c', or 'd') is provided. If the user enters an invalid option, the function displays an error message and prompts them to reenter their answer.

The function returns the valid user-selected option once obtained.

**4.4 calculate\_score(quiz\_data, user\_answers) Function**

The calculate\_score(quiz\_data, user\_answers) function is responsible for evaluating the user's performance and calculating their score based on their answers. It takes two parameters: quiz\_data, which is the list of dictionaries representing all the quiz questions, and user\_answers, which is a list containing the user's selected options for each question.

The function iterates through the quiz\_data and user\_answers simultaneously and compares the user's answer to the correct\_option for each question. If the user's answer matches the correct\_option, their score is incremented. The final score is returned at the end of the function.

**4.5 main() Function**

The main() function acts as the orchestrator of the entire quiz game. It is the entry point of the program. Within the main() function, the following steps are executed:

The load\_quiz\_data(file\_path) function is called to read the quiz data from the CSV file and store it in a list.

The total number of questions in the quiz\_data is determined to set the total\_questions variable.

An empty list, user\_answers, is initialized to store the user's answers.

The display\_question(question) function is called for each question in quiz\_data. The user is prompted to provide their answer using the collect\_answer() function, and the user's answer is appended to the user\_answers list.

The calculate\_score(quiz\_data, user\_answers) function is called to compute the user's final score based on their answers.

The final score and total number of questions are displayed to the user, completing the quiz game.

The main() function controls the flow of the quiz game, allowing users to interact with the questions, providing their answers, and receiving feedback on their performance.

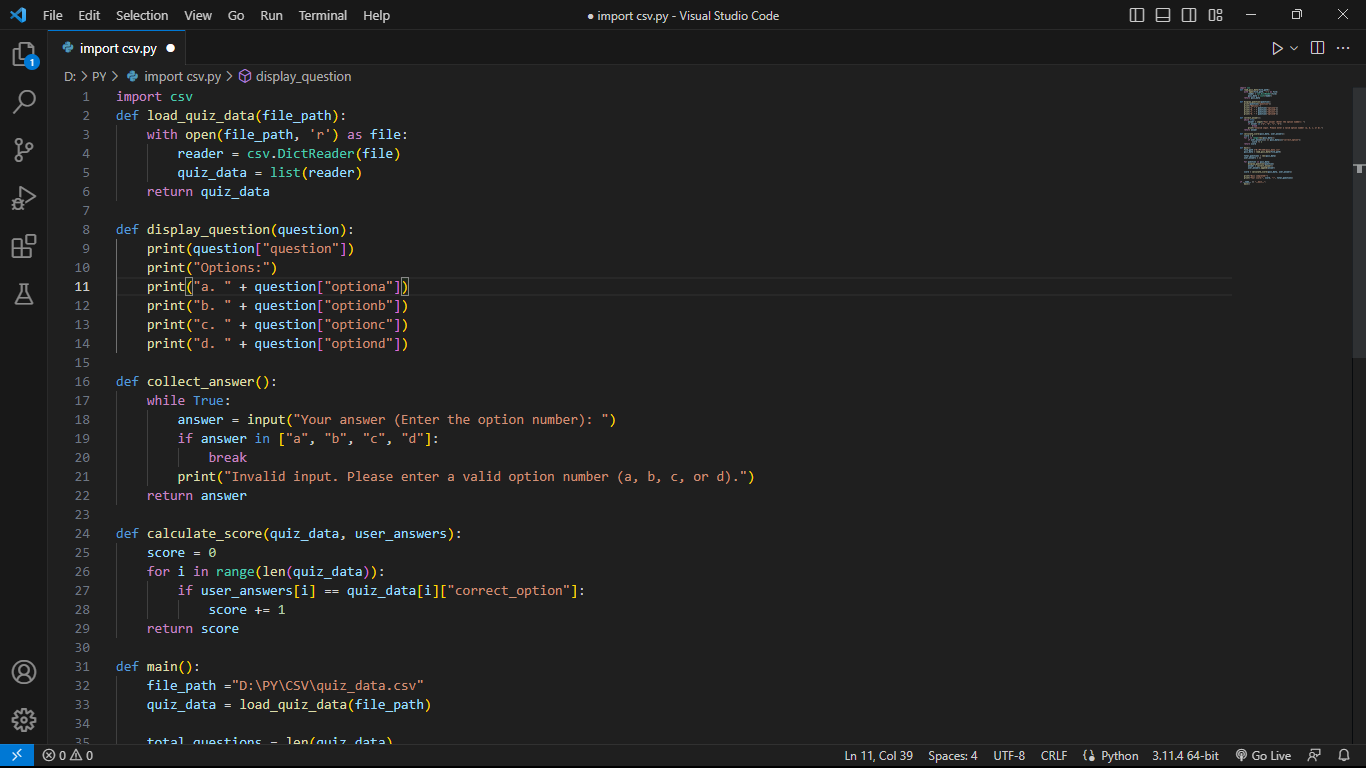
In the subsequent sections, we will explore the implementation details of each function, highlighting any challenges encountered and how they were addressed, ultimately resulting in the successful realization of the quiz game project.

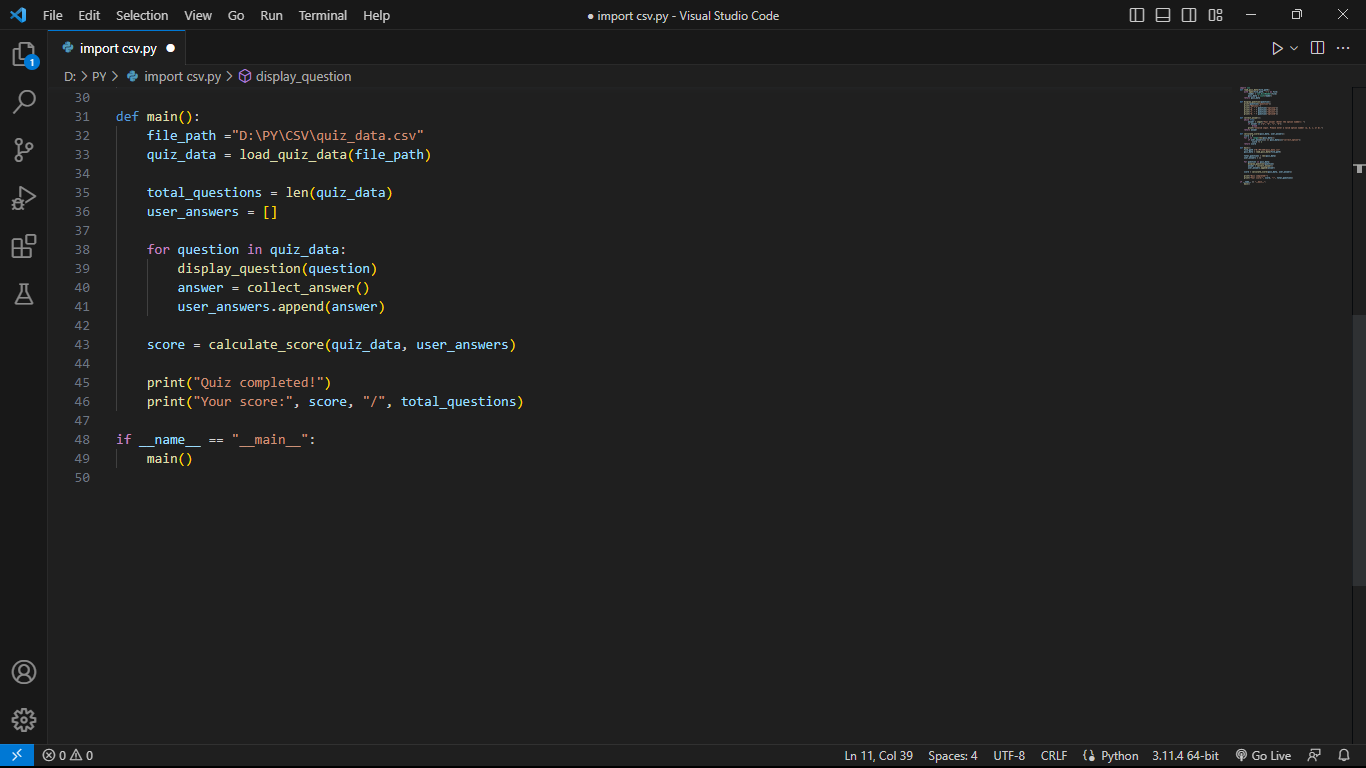
# Results and Output

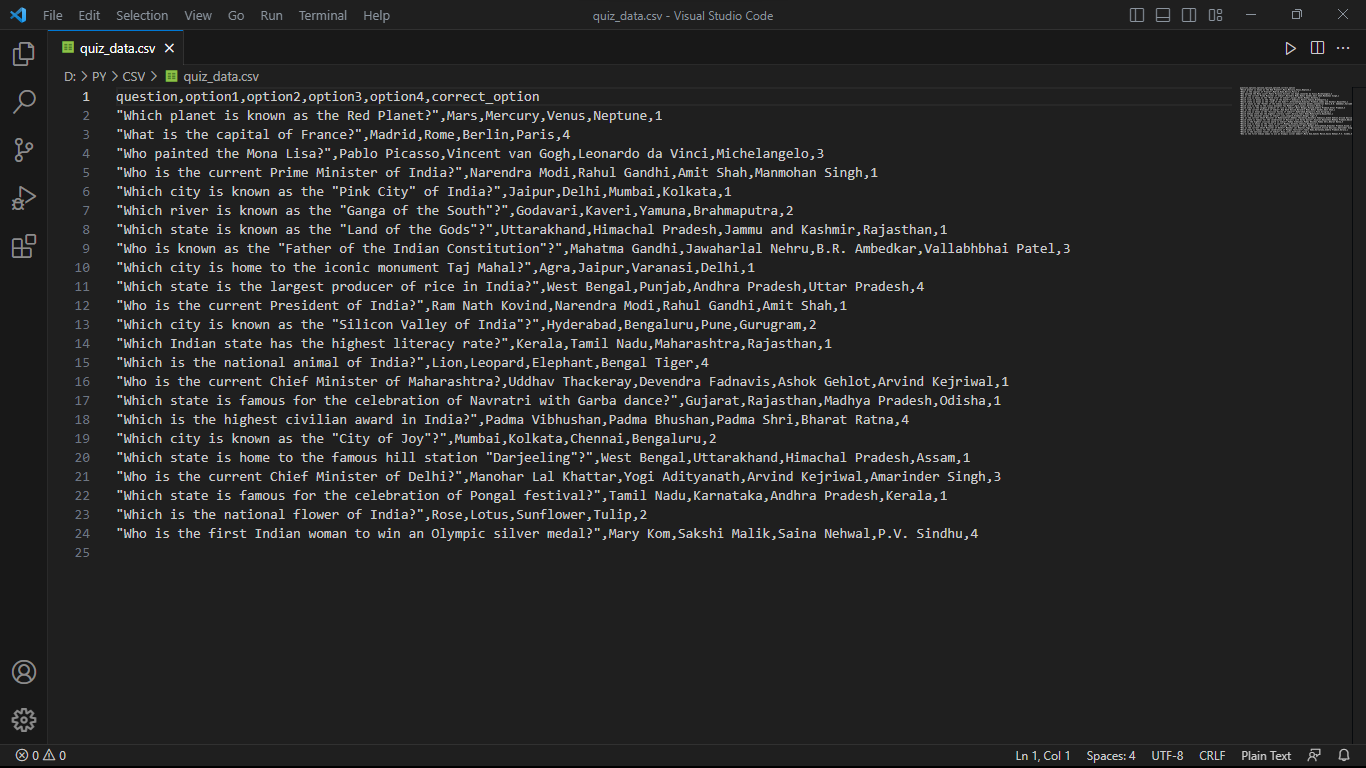
**5. Results and Output**

In the Results and Output section, we present the outcomes and user interactions from the quiz game. We include samples of the questions displayed to the users, their responses, and the final score obtained. Screenshots or textual representations of the interactions will be provided to illustrate the quiz game's functionality and user experience.

# Screenshots OF CODE





**CSV Code**

**6. Conclusion**

The Conclusion section summarizes the overall achievements of the quiz game project. It discusses how successfully the project met its objectives, highlights the strengths and weaknesses of the implementation, and evaluates the effectiveness of the quiz game as an educational and engaging tool. We will also include insights gained from user feedback and how it influenced the project's development.

**7. Future Enhancements**

In the Future Enhancements section, we explore potential improvements and additional features that can be incorporated into the quiz game. This includes ideas to enhance user engagement, expand the scope of the quiz topics, and integrate new functionalities. We will discuss the possibilities of implementing these enhancements to further enhance the quiz game's appeal and utility.

**8. SUBMISSION LINK**

• **Code Submission Link:**

https://github.com/AKtonXcoder/Quiz-game