A PROJECT REPORT

On

OFFLINE QUIZZ APP

Submitted in partial fulfillment of the requirement of
University of Mumbai for

Python Programming Mini Project
In
Computer Engineering (III SEM)

Submitted By

Onkar Vagare
Abhijith Nair
Aryan Jamdar
Rohan Shedge

Department of Computer Engineering
PILLAI COLLEGE OF ENGINEERING
(AUTONOMOUS)
New Panvel – 410 206
UNIVERSITY OF MUMBAI
Academic Year 2025 – 26

DEPARTMENT OF COMPUTER ENGINEERING
Pillai College of Engineering (Autonomous)
New Panvel – 410 206

CERTIFICATE

This is to certify that the requirements for the report entitled ‘Offline Quiz Application’ have been successfully completed by the following students:

Name	**Roll No.**
Onkar Vagare	A367
Abhijith Nair	A371
Aryan Jamdar	A372
Rohan Shedge	A378

In partial fulfillment of Python Programming mini Project in the Department of Computer Engineering, Pillai College of Engineering (Autonomous), New Panvel – 410 206 during the Academic Year 2025 – 2026.

Subject Incharge

DEPARTMENT OF COMPUTER ENGINEERING
Pillai College of Engineering (Autonomous)
New Panvel – 410 206

PROJECT APPROVAL FOR

This project entitled “Offline Quiz App” by onkar vagare, abhijit nair, aryan jamdar, and rohan shedge are approved for the degree of Bachelor of Engineering in Computer Engineering.

Examiners:

1. _________________

Date:

DEPARTMENT OF COMPUTER ENGINEERING
Pillai College of Engineering (Autonomous)
New Panvel – 410 206

DECLARATION

We declare that this written submission for the Python Programming Mini Project entitled "offline quizz app" represents our ideas in our own words and where others' ideas or words have been included. We have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any ideas / data / fact / source in our submission. We understand that any violation of the above will cause disciplinary action by the institute and also evoke penal action from the sources which have not been properly cited or from whom prior permission have not been taken when needed.

Project Group Members:

Onkar vagare & Sign:

____

Abhijith Nair & Sign:

____

Aryan jamdar & Sign:

____

Rohan Shedge & Sign:

____

Table of Contents

Abstract

The Quiz Application is a Python-based project designed to provide an interactive platform for users to test their knowledge across various topics. The system uses a database to store quiz questions, multiple-choice options, and correct answers, ensuring efficient data management and retrieval.

This application allows users to select quiz categories, answer a set of randomized questions, and view their scores at the end of the quiz. It demonstrates the implementation of Python database connectivity, including the use of functions such as connect(), cursor(), execute(), and fetchall() to perform database operations.

The main goal of this project is to create a user-friendly and educational tool that promotes learning through self-assessment. The Quiz App showcases the practical use of Python programming, database handling, and GUI development (using Tkinter or similar libraries) to build a complete and interactive application.

List Of Figures

List Of Tables

Chapter 1
INTRODUCTION

1.1 Background

In today’s digital learning environment, technology plays a vital role in enhancing education and knowledge assessment. Traditional quizzes and examinations conducted on paper are gradually being replaced by interactive digital platforms that provide instant feedback and automated evaluation. A Quiz Application simplifies the process of conducting assessments by enabling users to participate in quizzes, view results, and track their progress in real time.
Developed using Python, the Quiz App aims to provide a user-friendly, efficient, and flexible platform for learning and self-evaluation. It reduces manual work, eliminates human error in checking answers, and offers an engaging experience for both learners and educators.

1.2 Need for a Quiz Application

Manual quiz management systems are time-consuming and prone to errors in question handling, scoring, and evaluation. Moreover, they lack instant feedback and performance tracking features. The need for an automated quiz system arises from the growing demand for smart educational tools that can provide personalized learning experiences.
A Quiz App fulfills this requirement by automating question display, result calculation, and user interaction. It allows educators to create quizzes easily and enables users to test their knowledge anytime, anywhere.

1.3 Overview of the Project

The Quiz App is developed using Python with a simple and interactive interface. It allows users to answer a set of questions, view their score instantly, and analyze their performance. The system reads questions from a database or file, evaluates the answers, and displays the result in real time.
The system architecture consists of:

Frontend Layer: Provides the user interface for quiz interaction.Backend Layer: Handles the logic of question retrieval, answer validation, and score calculation.Database Layer: Stores quiz questions, options, and correct answers for dynamic access.

Chapter 2 :Objective

2.1 Problem Statement

Traditional methods of conducting quizzes or assessments involve manual preparation, distribution, and evaluation, which are both time-consuming and prone to human errors. Managing large sets of questions, calculating scores, and providing timely feedback can be tedious tasks for educators and learners alike.
The primary goal of this project is to design and develop an intelligent Quiz Application that automates the quiz process using Python. The system will allow users to take quizzes, receive instant feedback, and analyze their performance in real-time, thereby improving learning efficiency and user engagement.

Objectives of the Project
To develop an interactive and user-friendly interface for conducting quizzes.
To store and manage quiz questions, options, and answers efficiently using a database or structured files.
To automatically evaluate user responses and generate scores instantly.
To provide feedback and display correct answers after quiz completion.
To design a flexible system that allows easy addition or modification of quiz questions.
To enhance the learning experience by offering an engaging, automated, and accurate testing platform.

Scope of the Project
The Quiz App focuses on delivering a digital, efficient, and easy-to-use solution for conducting quizzes across various domains such as education, training, and self-assessment.
It supports multiple-choice questions (MCQs) and provides instant score calculation and performance feedback. The system can be extended to support user login, timed quizzes, and question randomization.
The project emphasizes simplicity, accuracy, and real-time evaluation, making it ideal for schools, colleges, online learning platforms, and individual learners seeking self-improvement.

2.2 System Requirements:

Software Requirements:

components	specification
Operating System	Windows 10 / 11
Programming Language	Python 3.x
Framework	Tkinter / Flask (depending on UI type)
Libraries	Tkinter / Flask, Pandas, Random, JSON / SQLite
Database	SQLite or CSV file

(Table no : 2.1)

Hardware Requirements:

components	specification
Processor	Intel i3 or above
RAM	Minimum 4 GB
Storage	250 GB or more
Display	1366×768 resolution

(Table no : 2.2)

Chapter 3 : Methodology

3.1 Database Schema and ER Diagram

Data Collection

The data for this project consists of a set of quiz questions, options, and correct answers, which can be stored in a database (SQLite) or a structured file (CSV/JSON). Each record contains a question, multiple choices, and the correct answer key.
This dataset serves as the foundation for quiz generation and evaluation. The questions can cover various topics such as general knowledge, science, mathematics, or programming, depending on the application's scope and target users.

Data Preprocessing

To ensure accurate question handling and error-free execution, the data undergoes preprocessing before being used in the application. The preprocessing steps include:

1. Data Validation: Checking for missing or duplicate questions to maintain dataset integrity.
2. Formatting: Ensuring questions, options, and answers follow a consistent format for smooth retrieval.
3. Encoding: Converting data from CSV or JSON into Python-readable objects (e.g., lists or dictionaries).
4. Shuffling: Randomizing the order of questions and options to make each quiz attempt unique.
5. Storage: Saving the processed data in an optimized format for faster access during runtime.

Quiz Logic and Evaluation

The preprocessed data is utilized by the backend logic to conduct and evaluate quizzes. The main processes include:
• Question Display: Questions are retrieved one at a time and displayed to the user via the interface.
• Answer Input: The user selects one of the provided options.
• Answer Validation: The system checks the user's response against the correct answer stored in the dataset.
• Score Calculation: The total score is updated dynamically as the quiz progresses.
• Result Display: At the end of the quiz, the application presents the final score and performance feedback.

ER-DIAGRAM:

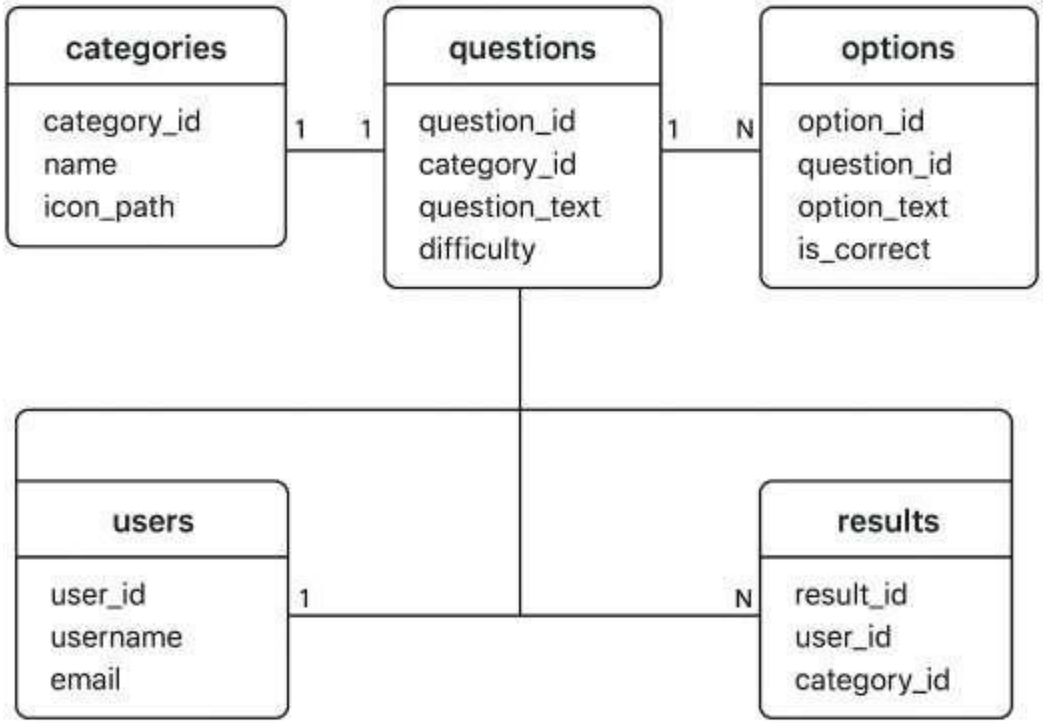


Figure no(3.1)

3.2 Requirement for Implementation

The implementation of the Quiz Web Application requires a combination of software tools, libraries, and system components to ensure efficient performance and smooth functionality. The project is developed using the Python programming language with the Flask web framework for backend logic and user interaction. The SQLite database is used to store quiz questions, answer options, and user results in a lightweight and portable manner. The HTML, CSS, and JavaScript technologies are employed in the frontend for designing interactive user interfaces, while images and icons are managed through the static directory.

The system is designed to run on any standard operating system such as Windows, Linux, or macOS with Python 3.x installed. Additional dependencies like Flask and other required libraries are managed through a requirements.txt file for easy setup. A modern web browser (Google Chrome, Edge, or Firefox) is required for accessing and interacting with the application. Overall, the environment provides all essential tools for building, testing, and deploying the quiz system effectively.

3.2.1 Tools and Techniques

Tool / Library	Purpose
Python 3.x	Core programming language used for logic and backend development
Tkinter / Flask	User interface framework for GUI or web-based interaction
Pandas / JSON	Data handling and question management
SQLite	Database for storing quiz questions and results (optional)
Random Module	Used for question shuffling and randomization

Table no(3.1)

3.2.2 Block Diagram

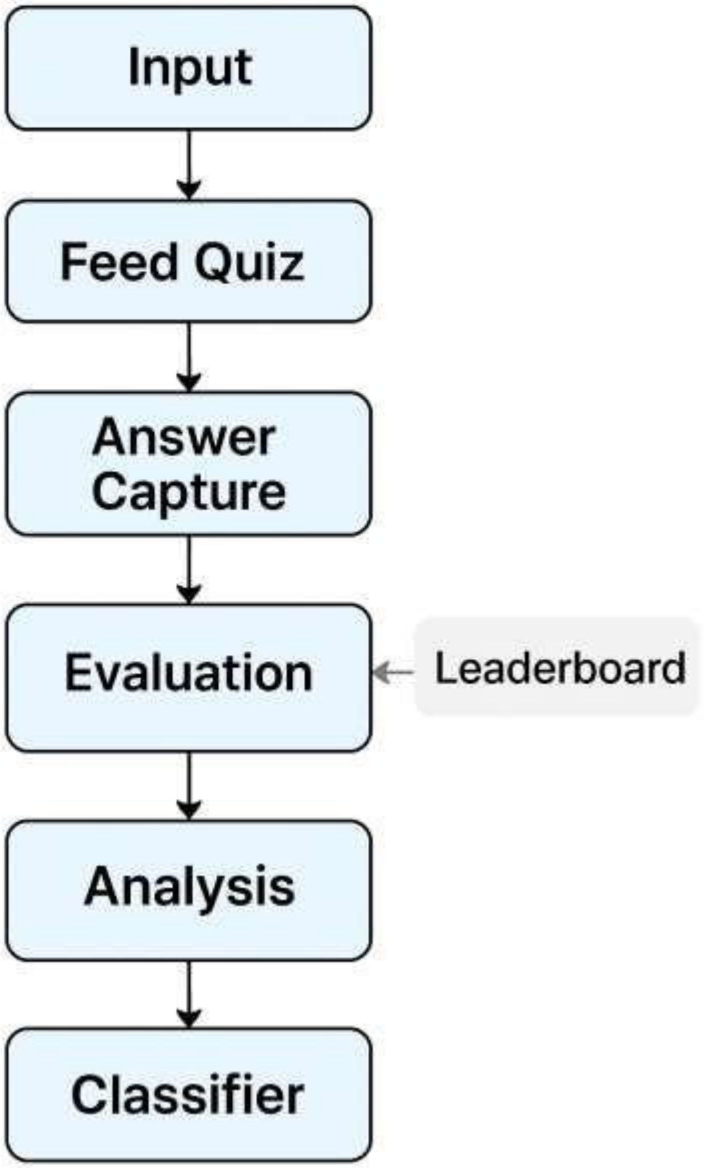


Figure no(3.2)

Chapter 4 : Implementation

4.1 System Overview

The Quiz Application has been implemented using Python, with either Tkinter (for GUI) or Flask (for web-based version) as the main interface framework. The system integrates multiple components such as question handling, user interaction, score calculation, and result display. The application follows a modular architecture, ensuring that each part—data management, quiz logic, and interface—functions independently yet cohesively.The system is based on a client–server model in the web version, or an event-driven model in the desktop version. The user interacts with the frontend (GUI/web interface), which communicates with the backend Python logic. The backend retrieves quiz questions, validates user answers, calculates scores, and displays the results in real time.

4.2 Project Working with Snapshots

The Quiz Web Application is designed to provide an interactive and user-friendly platform for conducting quizzes on various topics. The project follows a systematic flow starting from user interaction to result evaluation. The working of the system can be divided into several stages as described below:

1. **Home Page Display:**
   When the user launches the application, the home page (index.html) is displayed. It provides a simple interface to start the quiz and may include options to select a category such as Science, Technology, or History.

2. **Quiz Initialization:**
   Once the user starts the quiz, the system retrieves a set of questions from the database (quiz_database.db). The backend, developed using Flask, fetches the data through functions defined in quiz_logic.py and displays it dynamically on the quiz.html page.

3. **Question and Option Display:**

Each question is presented along with multiple answer options. The user can select one option for each question. The questions and options are shuffled to ensure that every quiz attempt is unique and unpredictable.

4. **Answer Submission and Evaluation:**
   After answering all the questions, the user submits the quiz. The submitted data is captured through an HTML form and processed by Flask's request.form method in main.py. The answers are then evaluated using the evaluate_quiz() function from quiz_logic.py, which compares user responses with the correct answers stored in the database.

5. **Score Calculation:**
   The system calculates the total score based on the number of correct answers. The scoring logic ensures fairness and accuracy. Optionally, percentage and performance levels can also be displayed.

6. **Result Display:**
   The final score and performance summary are displayed on the result.html page. This page provides feedback to the user about their quiz attempt, including the number of correct answers and total marks obtained.

7. **Data Storage (Optional):**
   The user's quiz attempt, including the score and category, can be saved into the results table for future analysis or leaderboards.

8. **End of Quiz:**
   After viewing the results, the user can choose to return to the home page and attempt another quiz, allowing continuous practice and engagement.

System Snapshots:

- **Home Page:** Displays the system name and a text box for entering reviews.

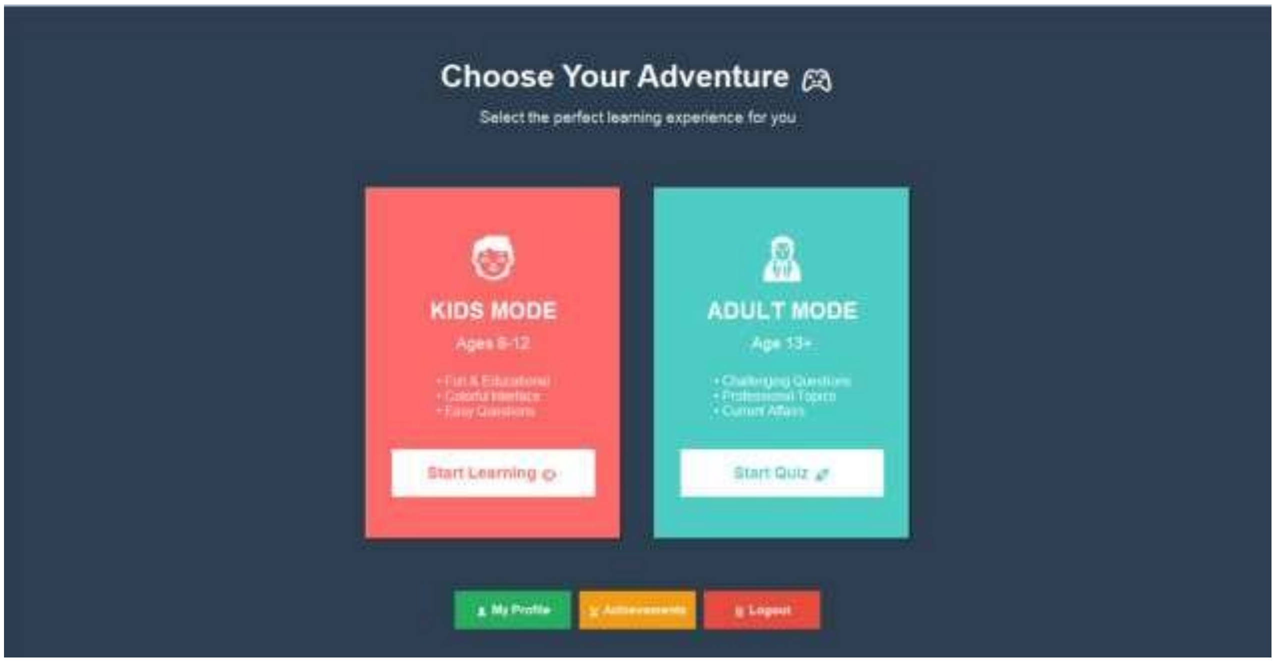
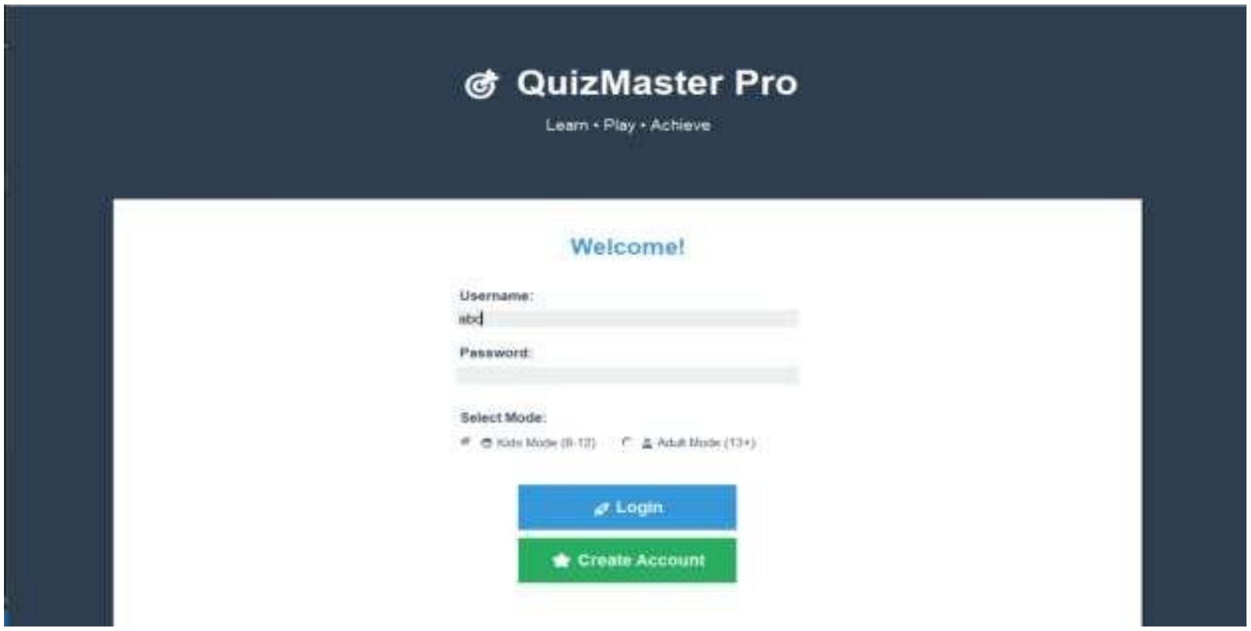


Figure no(4.1)

- **Login Page :** It can be used by registered users.



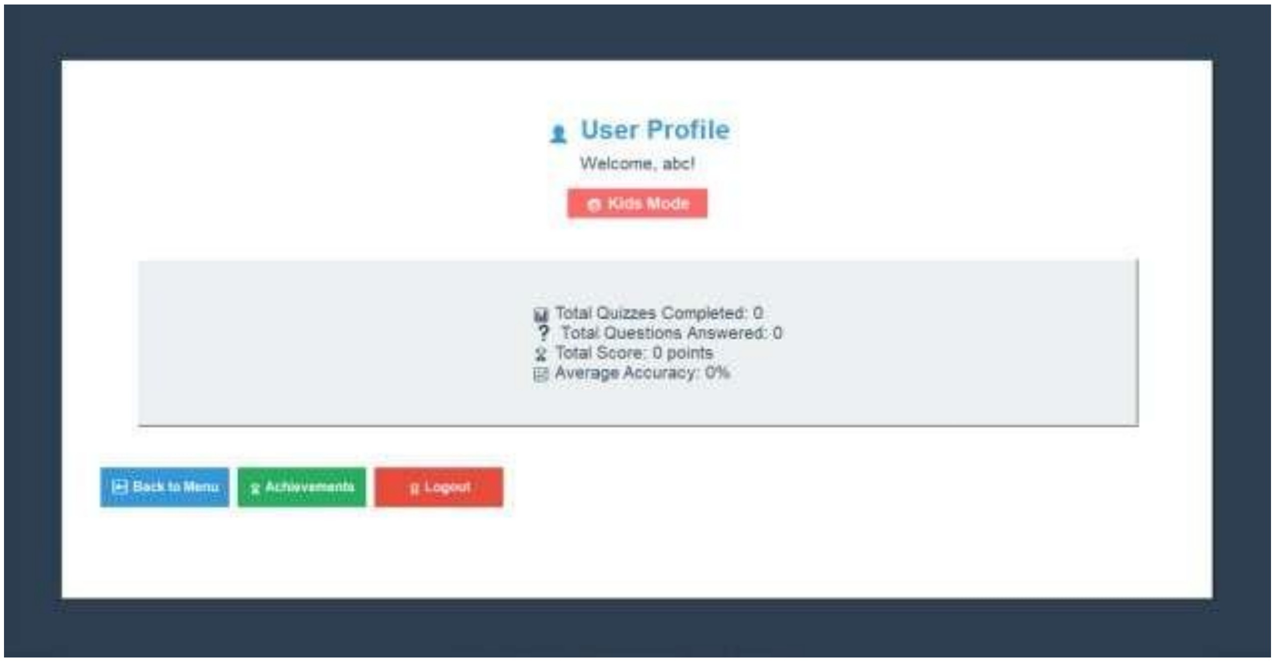
Username: abc

Password:

Select Mode: [✓ Kids Mode (8-12)] Adult Mode (13+)

Figure no(4.2)

- **User Profile Page: shows details of the user and profile**


User Profile

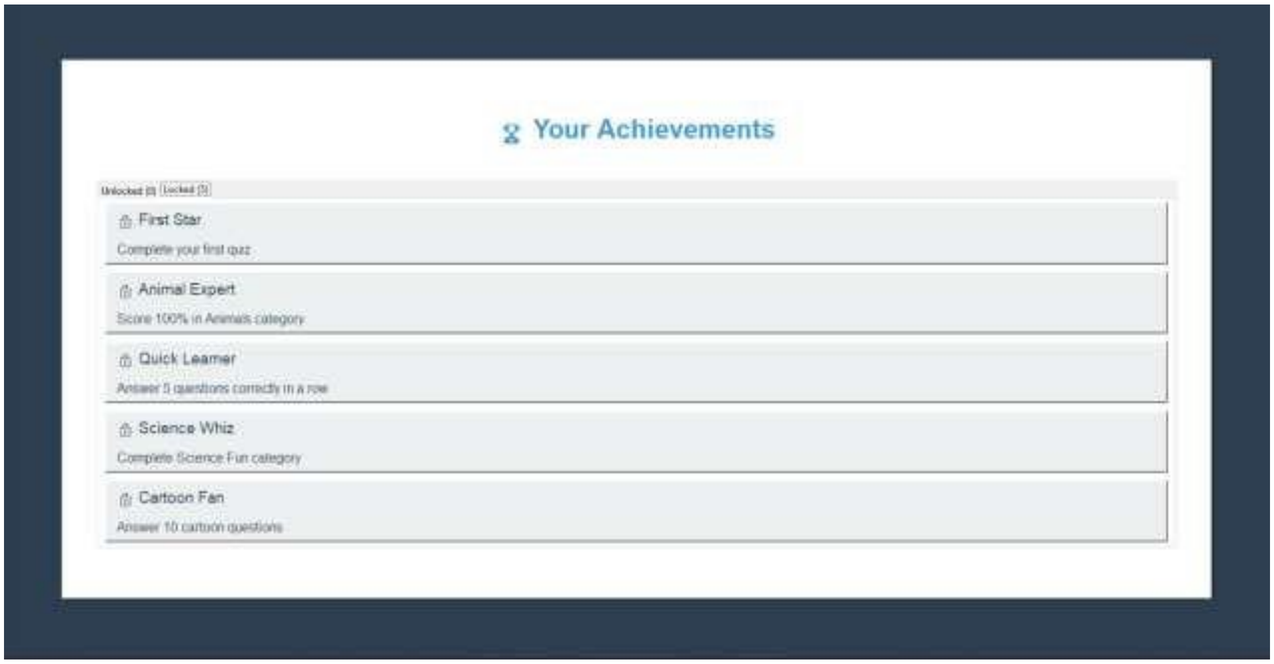
Welcome, abc!

Kids Mode

Total Quizzes Completed: 0
Total Questions Answered: 0
Total Score: 0 points
Average Accuracy: 0%

Figure no(4.3)

- **Achievement Page**


Your Achievements

First Star
Complete your first quiz

Animal Expert
Score 100% in Animals category

Quick Learner
Answer 5 questions correctly in a row

Science Whiz
Complete Science Fun category

Cartoon Fan
Answer 10 cartoon questions

Figure no(4.4)

- **Quiz Category Page**

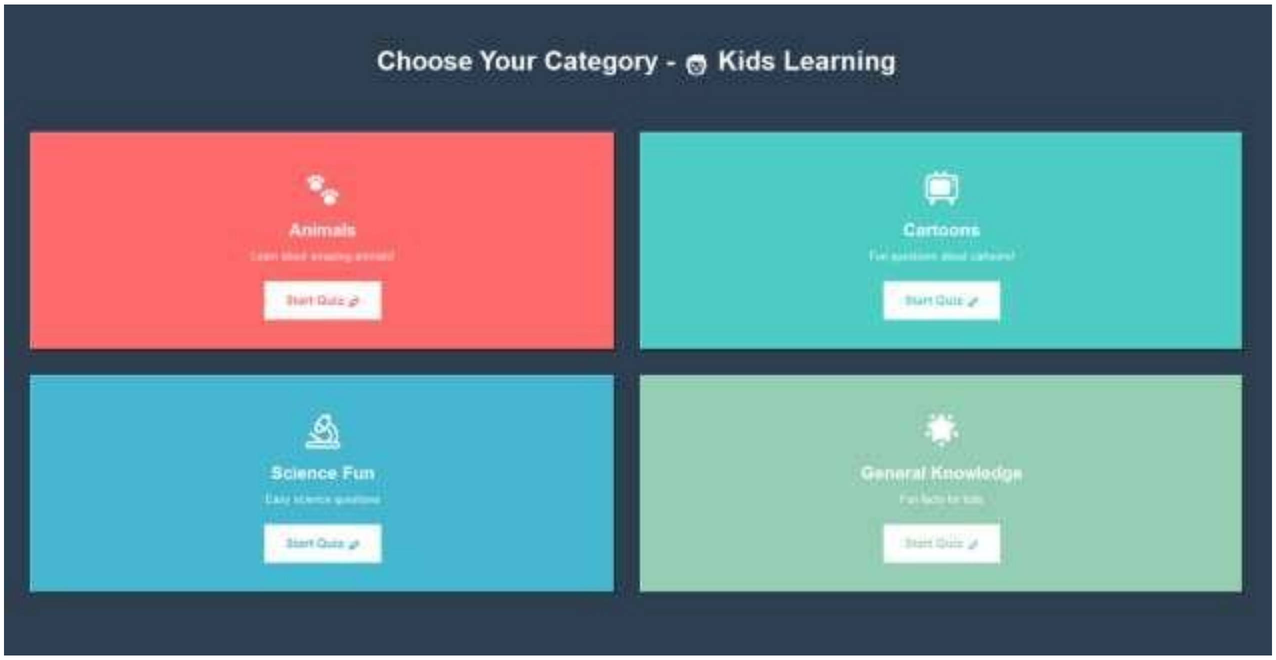


Figure no(4.5)

- **Attempting Quizz Page**

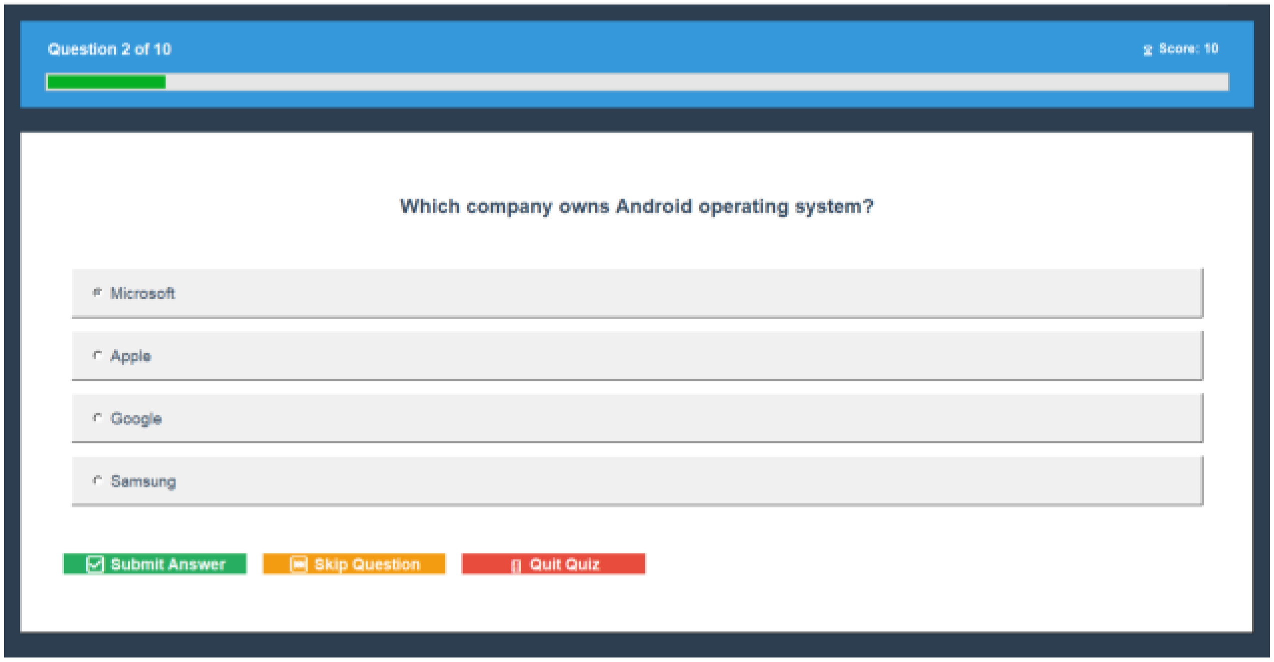


Figure no(4.6)

Chapter 5 : Applications and Future Scope

5.1 Applications

The **Quiz Application** has a wide range of applications in **education, training, entertainment, and corporate sectors**. Its core functionality—conducting automated quizzes, evaluating results, and providing instant feedback—makes it valuable for various use cases that involve learning, assessment, and engagement.

Educational Institutions
Schools, colleges, and universities can use this system to conduct online tests, practice quizzes, and assessments. It minimizes manual grading and allows teachers to easily track student performance.

E-Learning Platforms
Online learning websites and apps can integrate the Quiz Application to enhance user engagement. It can be used to test learners' understanding after lessons or tutorials, helping them reinforce knowledge through self-assessment.

Corporate Training
Organizations can use the application to conduct employee training evaluations and certification tests. Automated scoring ensures fair and consistent results across all participants.

Recruitment and Aptitude Tests
The Quiz App can be employed in recruitment processes to evaluate candidates' technical or general aptitude through objective-type quizzes, ensuring transparent and efficient screening.

Gaming and Entertainment
The application can be adapted into quiz-based games for fun and learning. It can feature topics like movies, sports, history, or current affairs to entertain users while improving their general knowledge.

5.2 Future Scope

Although the current implementation of the **Quiz Application** provides efficient performance and ease of use, several improvements can further enhance its functionality, scalability, and user experience.

Database Expansion and Categorization
Future versions can support a larger, categorized database with subject-wise quizzes and difficulty levels (Easy, Medium, Hard), allowing users to choose based on their knowledge level.

User Authentication and Profiles
Integrating login and registration features will allow multiple users to save their progress, review past performances, and compete with others.

Timed Quizzes and Leaderboards
Implementing a countdown timer and leaderboard system will make quizzes more competitive and engaging, especially for online challenges and competitions.

Web and Mobile App Integration
The application can be deployed as a **responsive web app** using Flask or as a **mobile application** using frameworks like **Flutter** or **Kivy**, expanding accessibility for all users.

Random Question Generation
Future enhancements could include dynamic question generation using algorithms that pull random questions from large datasets, ensuring variety in every quiz attempt.

Performance Analytics Dashboard
An admin dashboard can be added to visualize user performance using graphs, charts, and statistics to track learning progress and identify weak areas.

Voice and Multimedia Support
Adding voice-based questions, audio playback, or image-based quizzes can make the app more interactive and suitable for language learning or visual-based subjects.

Chapter 6 : Summary

The Quiz Application project successfully demonstrates the use of Python programming to automate the process of conducting, evaluating, and analyzing quizzes in an interactive and efficient manner. The primary goal of this project creating a digital platform for knowledge testing and self-assessment has been effectively achieved through systematic design, modular implementation, and real-time feedback features.

Developed using Python and supported by tools such as Tkinter, Flask, and SQLite, the system ensures a smooth and user-friendly experience. Its modular architecture, consisting of distinct layers for interface, logic, and data management, promotes scalability and easy maintenance. The application eliminates the need for manual grading and delivers instant results, enhancing both learning and evaluation efficiency.

The project integrates concepts of programming logic, data handling, and interface design to create a complete functional system. With features like random question generation, automatic score calculation, and performance feedback, the Quiz App provides an engaging platform for students, educators, and organizations alike.

From an educational perspective, this project bridges theoretical knowledge of programming and practical software development. It illustrates how technology can simplify assessment processes, encourage self-learning, and foster interactive education.

In conclusion, the Quiz Application serves as a practical example of how automation can enhance traditional learning systems. With future extensions such as timed quizzes, user accounts, cloud integration, and analytics dashboards, it has strong potential to evolve into a comprehensive online examination and learning management system.

References

1. Python 3.x Documentation – https://docs.python.org/3/
2. Tkinter GUI Documentation – https://docs.python.org/3/library/tkinter.html
3. Flask Framework Documentation – https://flask.palletsprojects.com/
4. SQLite Database Documentation – https://www.sqlite.org/docs.html
5. Pandas Library Documentation – https://pandas.pydata.org/docs/
6. GeeksforGeeks – “Python Quiz App using Tkinter / Flask”
7. W3Schools – “Python Programming and Web Development Tutorials”
8. TutorialsPoint – “Python Database Programming”
9. Real Python – “Building Interactive Applications with Tkinter and Flask”
10. Medium – “Developing Simple Quiz Applications with Python”

Acknowledgement

We would like to express our sincere gratitude to Prof. Sangeetha Selvan our mini project guide, for their constant guidance, valuable suggestions, and encouragement
throughout the completion of this project. Their insights and feedback have been instrumental in helping us apply the knowledge we acquired during the semester and in learning new technical concepts.

We would also like to extend our heartfelt thanks to Dr. Sharvari Govilkar, Head of the Department of Computer Engineering, for giving us the opportunity to undertake this mini project. This project provided us with hands-on experience and helped us understand the real-world applications of our academic learning.

Finally, we express our deep appreciation to Dr. Sandeep Joshi, Principal of Pillai College of Engineering, for providing us with the resources, facilities, and an encouraging academic environment to successfully complete this project.
We are grateful to all teaching and non-teaching staff members of the Department of Computer Engineering for their cooperation and support.

Project Members:
Onkar Vagare
Abhijith Nair
Aryan Jamdar
Rohan Shedge