* **Overview**

The objective of this project was to create an interactive and engaging quiz game using the Python programming language with a graphical user interface (GUI) implemented using Tkinter. The quiz game features multiple-choice questions, time tracking, scoring, and the ability to save scores to a file.

* **Project Components**

**1. quiz\_game.py:**

**The primary script, quiz\_game.py, serves as the main entry point for the quiz game. It is responsible for initializing the Tkinter GUI, managing the flow of questions, handling user input, and updating the display based on the user's progress.**

**Key Features:**

**Questions Shuffle: Questions are randomized to enhance the user experience.**

**Time Tracking: Each question is associated with a timer, and the game provides feedback when the time limit is reached.**

**Scoring System: The game keeps track of the user's score, providing feedback on correct and incorrect answers.**

**Results Display: At the end of the quiz, the user's score is displayed along with the total possible score.**

**2. questions.py:**

**The questions.py file contains the list of multiple-choice questions used in the quiz game. It also includes functions for loading questions and saving scores.**

**Key Components:**

**Question Data: A list of dictionaries, each representing a question with options and the correct answer.**

**load\_questions() Function: Loads the questions from the predefined list. This function can be extended to load questions from external sources like files or databases.**

**save\_score() Function: Saves the user's score along with a timestamp to a file (scores.txt). This file keeps a record of user performance.**

**3. scores.txt:**

**The scores.txt file is used to store the scores achieved by users. It is a simple text file where each line represents a record with a timestamp and the user's score.**

**Example Entry:**

**2024-01-15 18:30:45: Scored 3/5**

* **Usage Instructions**

**To use the quiz game:**

1. **Ensure that all files (quiz\_game.py, questions.py, and scores.txt) are in the same directory.**
2. **Run the quiz\_game.py script.**
3. **Answer the multiple-choice questions presented in the GUI.**
4. **Receive feedback on correct/incorrect answers and view the final score at the end.**
5. **Scores are saved in the scores.txt file for future reference.**

* **Conclusion**

**This quiz game provides an interactive and educational experience, testing users' knowledge on a variety of topics. It demonstrates the use of Python, Tkinter for GUI development, time tracking, and data storage techniques. The modular design allows for easy extension and modification, making it a versatile tool for educational purposes or as a template for more complex quiz applications.**

**The Full Code**

import tkinter as tk  
from tkinter import ttk  
import random  
from datetime import datetime  
from questions import load\_questions, save\_score  
  
class QuizGame:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Quiz Game")  
  
 self.questions = random.sample(load\_questions(), len(load\_questions()))  
 self.current\_question\_index = 0  
 self.score = 0  
 self.start\_time = None  
  
 self.question\_label = ttk.Label(root, text="")  
 self.option\_buttons = []  
 self.feedback\_label = ttk.Label(root, text="")  
 self.score\_label = ttk.Label(root, text="Score: 0")  
  
 self.setup\_ui()  
  
 def setup\_ui(self):  
 self.question\_label.grid(row=0, column=0, columnspan=2, padx=10, pady=10)  
 self.score\_label.grid(row=1, column=0, columnspan=2, padx=10, pady=10)  
  
 for i in range(4):  
 option\_button = ttk.Button(self.root, text="", command=lambda i=i: self.check\_answer(i))  
 option\_button.grid(row=i + 2, column=0, columnspan=2, padx=10, pady=5)  
 self.option\_buttons.append(option\_button)  
  
 self.feedback\_label.grid(row=6, column=0, columnspan=2, padx=10, pady=10)  
  
 self.next\_question()  
  
 def next\_question(self):  
 if self.current\_question\_index < len(self.questions):  
 question\_data = self.questions[self.current\_question\_index]  
 self.start\_time = datetime.now()  
  
 self.root.after(1000, self.update\_timer)  
 self.question\_label.config(text=question\_data["question"])  
 options = question\_data["options"]  
 random.shuffle(options)  
  
 for i in range(4):  
 self.option\_buttons[i].config(text=options[i])  
  
 else:  
 self.display\_results()  
  
 def check\_answer(self, option\_index):  
 selected\_option = self.option\_buttons[option\_index]["text"]  
 correct\_answer = self.questions[self.current\_question\_index]["answer"]  
  
 if selected\_option == correct\_answer:  
 self.score += 1  
 self.feedback\_label.config(text="Correct!", foreground="green")  
 else:  
 self.feedback\_label.config(text=f"Incorrect. Correct answer: {correct\_answer}", foreground="red")  
  
 self.current\_question\_index += 1  
 self.score\_label.config(text=f"Score: {self.score}")  
 self.clear\_options()  
 self.next\_question()  
  
 def clear\_options(self):  
 for button in self.option\_buttons:  
 button.config(state="normal")  
  
 def update\_timer(self):  
 elapsed\_time = datetime.now() - self.start\_time  
 remaining\_time = max(15 - elapsed\_time.seconds, 0)  
  
 if remaining\_time > 0:  
 self.root.after(1000, self.update\_timer)  
 else:  
 self.feedback\_label.config(text=f"Time's up! Correct answer: {self.questions[self.current\_question\_index]['answer']}", foreground="red")  
 self.current\_question\_index += 1  
 self.clear\_options()  
 self.next\_question()  
  
 def display\_results(self):  
 result\_text = f"Quiz completed!\nYour Score: {self.score}/{len(self.questions)}"  
 self.question\_label.config(text=result\_text)  
 for button in self.option\_buttons:  
 button.grid\_forget()  
  
 save\_score(self.score)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = QuizGame(root)  
 root.mainloop()

# questions.py  
from datetime import datetime # Add this line to import the datetime module  
  
questions\_data = [  
 {  
 "question": "What is the capital of France?",  
 "options": ["Paris", "Berlin", "Madrid", "Rome"],  
 "answer": "Paris"  
 },  
 {  
 "question": "Which planet is known as the Red Planet?",  
 "options": ["Earth", "Mars", "Venus", "Jupiter"],  
 "answer": "Mars"  
 },  
 {  
 "question": "What is the largest mammal?",  
 "options": ["Elephant", "Blue Whale", "Giraffe", "Hippopotamus"],  
 "answer": "Blue Whale"  
 },  
 {  
 "question": "Who wrote 'Romeo and Juliet'?",  
 "options": ["Charles Dickens", "William Shakespeare", "Jane Austen", "Mark Twain"],  
 "answer": "William Shakespeare"  
 },  
 {  
 "question": "What is the capital of Japan?",  
 "options": ["Beijing", "Tokyo", "Seoul", "Bangkok"],  
 "answer": "Tokyo"  
 },  
 # Add more questions as needed  
]  
  
def load\_questions():  
 # You can modify this function to load questions from a file or a database.  
 return questions\_data  
  
def save\_score(score):  
 with open("scores.txt", "a") as file:  
 timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  
 file.write(f"{timestamp}: Scored {score}/{len(questions\_data)}\n")