# **Quiz Application with CSV Backend**

## **Introduction**

The "Quiz Application with CSV Backend" is a Python-based project designed for CBSE Class 12th students. This project allows users to attempt quizzes, store questions and scores in CSV files, and generate performance reports with visualizations. It is ideal for understanding file handling, data visualization, and Python programming concepts.

## **Objective**

To develop a user-friendly quiz application with the following functionalities:

* Manage questions and answers using CSV files.
* Conduct quizzes with dynamic feedback on performance.
* Analyze performance through visualizations like bar charts, pie charts, and line graphs.

## **Features**

### **1. Admin Mode**

* Add, edit, and delete quiz questions.
* View all stored questions.

### **2. Quiz Mode**

* Interactive quiz for users.
* Tracks correct and incorrect answers.
* Saves scores and user details.

### **3. Report Generation**

* View top scores and detailed performance analysis.
* Generate:
  + Bar chart for overall scores.
  + Pie chart for correct vs. incorrect answers.
  + Line graph tracking user progress over time.

## **Technologies Used**

* **Programming Language:** Python 3.x
* **Data Storage:** CSV Files
* **Libraries:**
  + csv for file handling.
  + pandas for data manipulation.
  + matplotlib for data visualization.

## **System Requirements**

* Python 3.x
* Libraries: Install using pip
* pip install pandas matplotlib

## **File Structure**

### **1. questions.csv**

* Stores quiz questions and answers.
* Format:  
   Question,Option1,Option2,Option3,Option4,CorrectAnswer
* Example:  
   What is the capital of India?,New Delhi,Mumbai,Kolkata,Chennai,1
* What is 2+2?,3,4,5,6,2

### **2. scores.csv**

* Stores user scores and timestamps.
* Format:  
   Name,Score,Date
* Example:  
   Jiya Rawat,2,2025-01-08 14:30:00
* Alice,1,2025-01-08 15:00:00

## **Code Implementation**

### **Modules and Functions**

#### **1. Admin Mode**

* add\_question(): Add questions to questions.csv.
* load\_questions(): Load all questions from the file.
* visualize\_scores(): Display a bar chart of scores.

#### **2. Quiz Mode**

* start\_quiz(): Run the quiz for the user.
* save\_score(): Save the user's score and timestamp to scores.csv.
* quiz\_performance\_pie(): Display a pie chart of quiz performance.

#### **3. Report Generation**

* track\_user\_progress(name): Generate a line graph of user’s progress over time.

### **Complete Code**

**import csv**

**import os**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**from datetime import datetime**

**# Load questions from the CSV file**

**def load\_questions():**

**questions = []**

**if os.path.exists("questions.csv"):**

**with open("questions.csv", "r") as file:**

**reader = csv.reader(file)**

**for row in reader:**

**questions.append(row)**

**return questions**

**# Add a new question to the CSV file**

**def add\_question():**

**question = input("Enter the question: ")**

**options = [input(f"Enter option {i+1}: ") for i in range(4)]**

**correct\_answer = input("Enter the correct answer (1/2/3/4): ")**

**with open("questions.csv", "a", newline="") as file:**

**writer = csv.writer(file)**

**writer.writerow([question, \*options, correct\_answer])**

**print("Question added successfully!")**

**# Start the quiz**

**def start\_quiz():**

**questions = load\_questions()**

**if not questions:**

**print("No questions available!")**

**return**

**score = 0**

**total\_questions = len(questions)**

**print("Starting the quiz...\n")**

**for i, question in enumerate(questions):**

**print(f"Q{i+1}: {question[0]}")**

**for j, option in enumerate(question[1:5]):**

**print(f"{j+1}. {option}")**

**answer = input("Your answer (1/2/3/4): ")**

**if answer == question[5]:**

**score += 1**

**print("Correct!\n")**

**else:**

**print(f"Wrong! The correct answer is {question[5]}\n")**

**print(f"Your total score: {score}/{total\_questions}")**

**save\_score(score)**

**# Display Pie Chart**

**quiz\_performance\_pie(score, total\_questions)**

**# Save user score to the CSV file**

**def save\_score(score):**

**name = input("Enter your name: ")**

**with open("scores.csv", "a", newline="") as file:**

**writer = csv.writer(file)**

**writer.writerow([name, score, datetime.now().strftime("%Y-%m-%d %H:%M:%S")])**

**print("Score saved successfully!")**

**# Visualize scores with a bar chart**

**def visualize\_scores():**

**if not os.path.exists("scores.csv"):**

**print("No scores available for visualization!")**

**return**

**df = pd.read\_csv("scores.csv", names=["Name", "Score", "Date"])**

**grouped = df.groupby("Name")["Score"].sum()**

**grouped.plot(kind="bar", color="skyblue", figsize=(8, 6))**

**plt.title("Total Scores by Participants")**

**plt.xlabel("Participants")**

**plt.ylabel("Total Score")**

**plt.xticks(rotation=45)**

**plt.tight\_layout()**

**plt.show()**

**# Generate a pie chart for quiz performance**

**def quiz\_performance\_pie(correct, total):**

**incorrect = total - correct**

**labels = ['Correct', 'Incorrect']**

**sizes = [correct, incorrect]**

**colors = ['green', 'red']**

**plt.figure(figsize=(6, 6))**

**plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)**

**plt.title("Quiz Performance Breakdown")**

**plt.axis('equal')**

**plt.show()**

**# Track a user's progress with a line graph**

**def track\_user\_progress(name):**

**if not os.path.exists("scores.csv"):**

**print("No scores available!")**

**return**

**df = pd.read\_csv("scores.csv", names=["Name", "Score", "Date"])**

**user\_data = df[df["Name"] == name]**

**if user\_data.empty:**

**print(f"No data available for user: {name}")**

**return**

**user\_data["Date"] = pd.to\_datetime(user\_data["Date"])**

**user\_data = user\_data.sort\_values(by="Date")**

**plt.figure(figsize=(8, 6))**

**plt.plot(user\_data["Date"], user\_data["Score"], marker="o", linestyle="-", color="blue")**

**plt.title(f"Score Progress for {name}")**

**plt.xlabel("Date")**

**plt.ylabel("Score")**

**plt.xticks(rotation=45)**

**plt.tight\_layout()**

**plt.show()**

**# Admin menu**

**def admin\_menu():**

**while True:**

**print("\nAdmin Menu:")**

**print("1. Add Question")**

**print("2. View Questions")**

**print("3. Visualize Scores (Bar Chart)")**

**print("4. Exit")**

**choice = input("Enter your choice: ")**

**if choice == "1":**

**add\_question()**

**elif choice == "2":**

**questions = load\_questions()**

**for question in questions:**

**print(question)**

**elif choice == "3":**

**visualize\_scores()**

**elif choice == "4":**

**break**

**else:**

**print("Invalid choice! Try again.")**

**# Main function**

**def main():**

**while True:**

**print("\nMain Menu:")**

**print("1. Start Quiz")**

**print("2. Admin Menu")**

**print("3. Track User Progress")**

**print("4. Exit")**

**choice = input("Enter your choice: ")**

**if choice == "1":**

**start\_quiz()**

**elif choice == "2":**

**admin\_menu()**

**elif choice == "3":**

**name = input("Enter your name to track progress: ")**

**track\_user\_progress(name)**

**elif choice == "4":**

**print("Exiting the application. Goodbye!")**

**break**

**else:**

**print("Invalid choice! Try again.")**

**if \_\_name\_\_ == "\_\_main\_\_":**

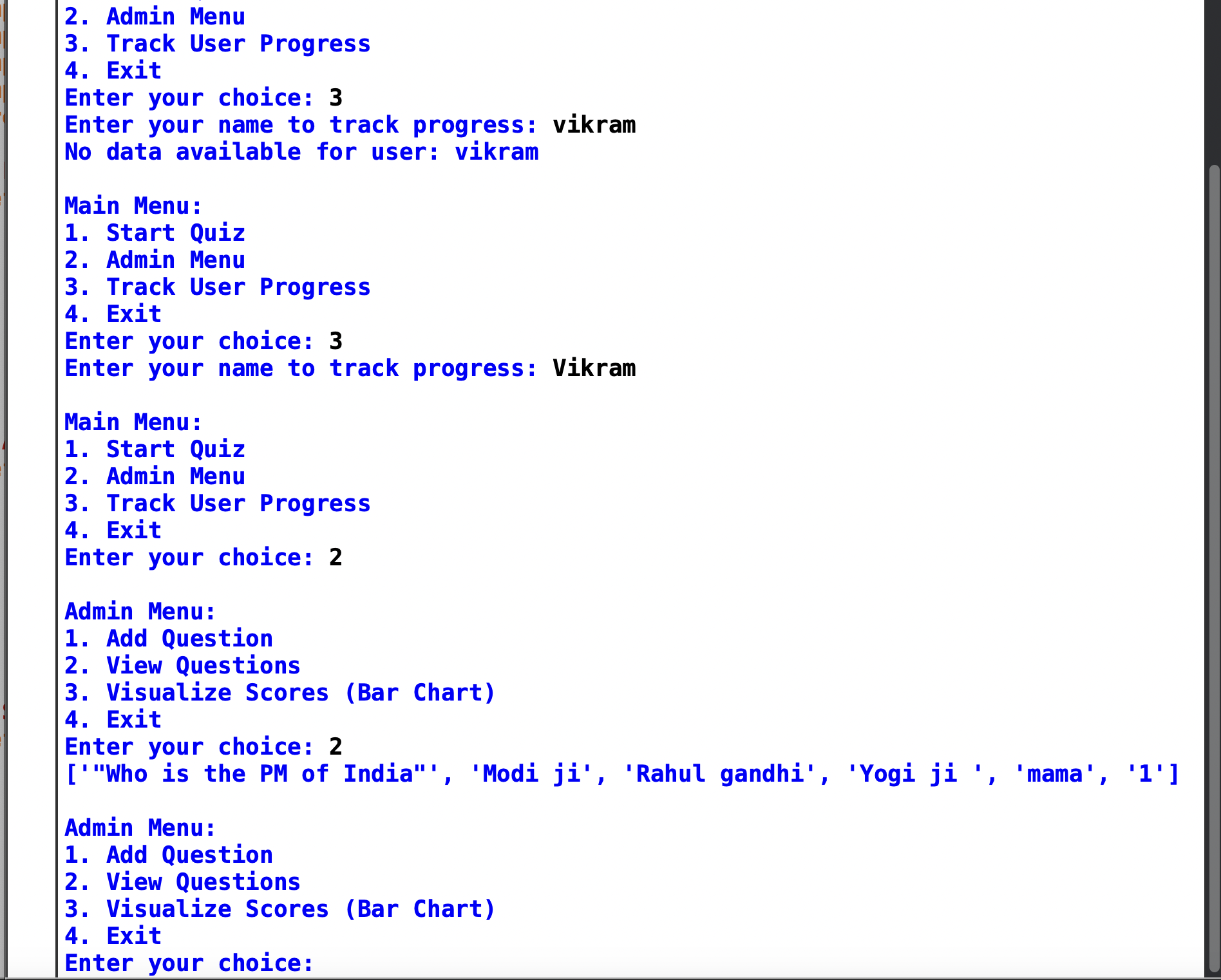
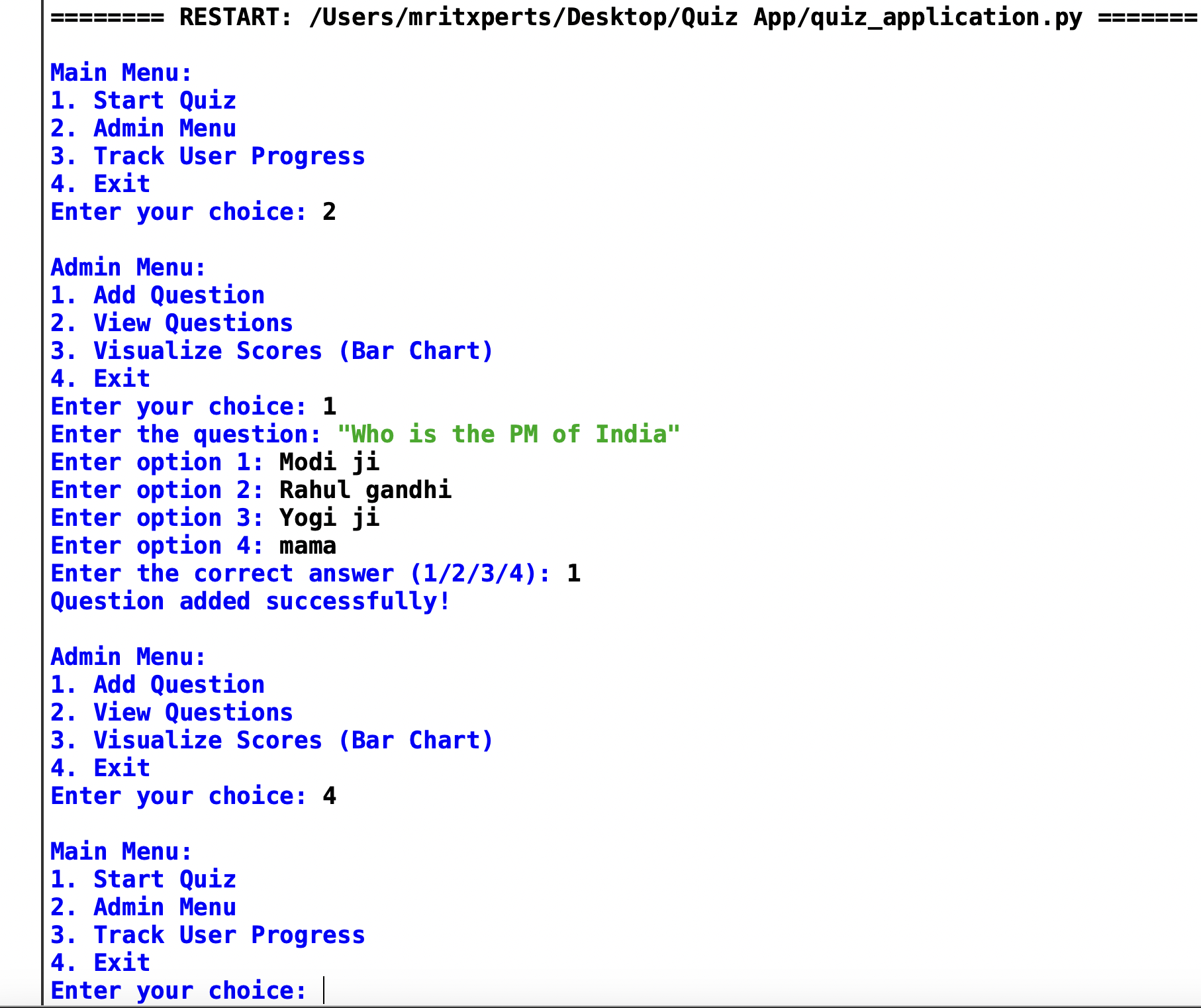
**main()**

## **How to Run the Project**

1. Create the following empty CSV files in the same directory as the script:
   * questions.csv
   * scores.csv
2. Save the provided Python code as quiz\_application.py.

* Run the script:  
   python quiz\_application.py

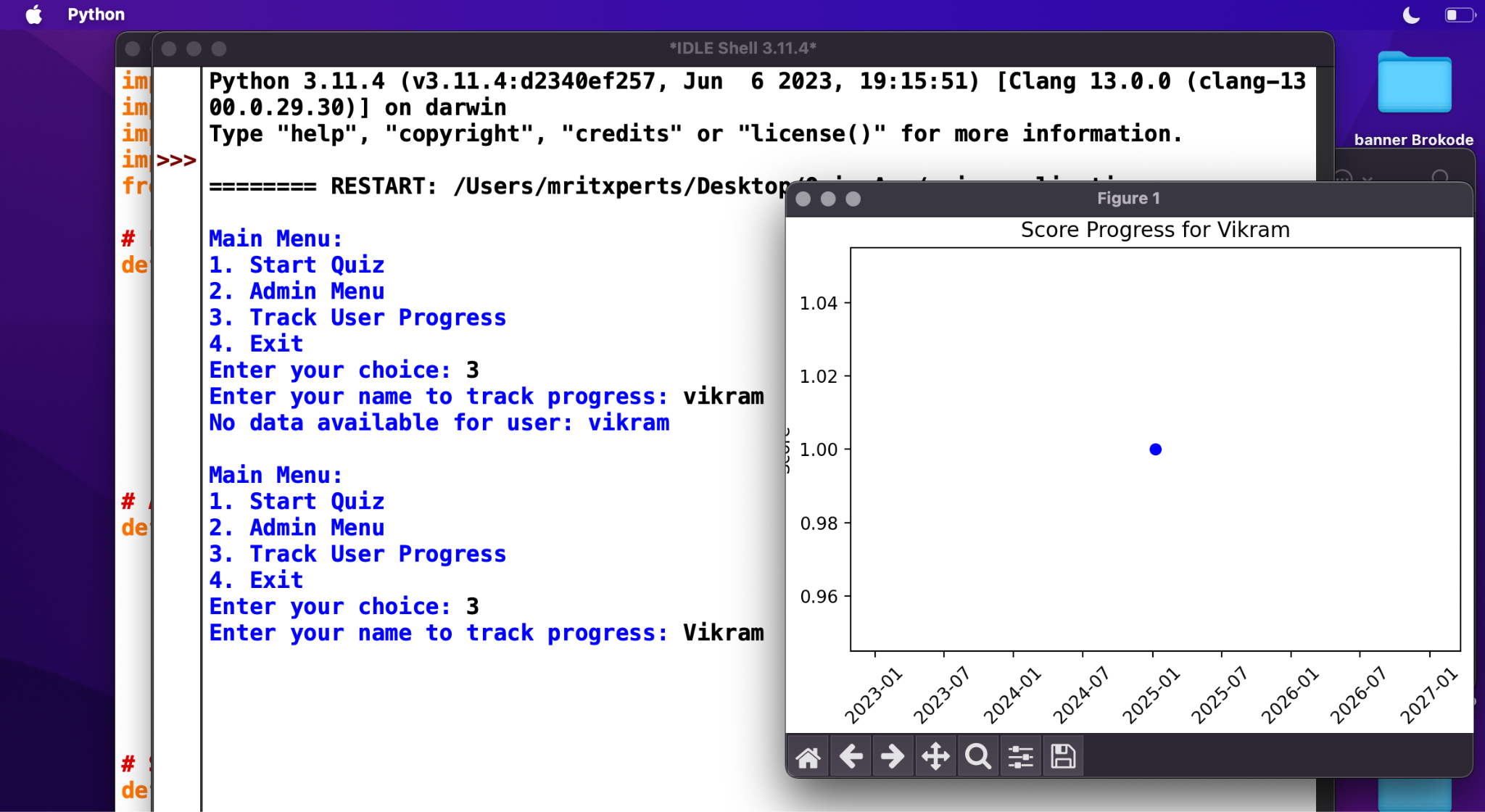
1. Follow the menu options:
   * Start a quiz.
   * Add or view questions (Admin Mode).
   * Generate reports.



## **Visualization Examples**

### **1. Bar Chart**

* Displays total scores by participants.



### **2. Pie Chart**

* Shows the percentage of correct and incorrect answers.

### 

### **3. Line Graph**

* Tracks individual performance over time.

## **Learning Outcomes**

* Understanding of Python’s file handling with CSV.
* Implementation of data visualization using Matplotlib.
* Practical experience with modular programming.

## **Conclusion**

This project demonstrates the practical use of Python for developing applications with data storage and visualization capabilities. By working on this project, students will gain hands-on experience in handling real-world problems using programming.

## **Acknowledgment**

This project, "Quiz Application with CSV Backend," was created by **Jiya Rawat** with the guidance and support of **Vikram Singh**, who provided valuable insights and mentorship throughout the development process.

## **Future Enhancements**

* Add a graphical user interface (GUI) using Tkinter or PyQt.
* Enable user authentication for personalized experiences.
* Store data in a database for scalability.