INSTITUTE OF COMPUTER SCIENCE AND DIGITAL INNOVATION

ACADEMIC SESSION: MAY - SEPTEMBER 2025

BIC1114/BBA1134: INTRODUCTION TO PROGRAMMING/PROGRAMMING FOR FINTECH

ASSIGNMENT **DEADLINE: 5TH AUGUST 2025 5.00 PM**

Student Name: Hussain Mohammad habib khan **Student ID:** 1002577628

Student Name: Abdallah Mohammed A Alsahli **Student ID:** 1002578829

Student Name: Salah EL-Ghobbas **Student ID:** 1002476835

Subject Name: introduction to programming
Subject Code: BIC1144

Lecturer/Tutor: **ASST. PROF. DR. SAMAR GHAZAL**

Assignment Number/Title: Smart Online Quiz System (SOQS)

Due Date: **5TH AUGUT 2025**

Assignment Weightage: **30%**

All work must be submitted by the due date.

Student's Statement:

I hereby declare that the work submitted is my own. I confirm that I have read and understood the University regulations with regard to plagiarism, and that Plagiarism, Collusion and Cheating in this work will be penalized.

Note:

1) You are expected to retain copies of your assignment report. Student Signature ……

2) If you require an acknowledgement receipt of this assignment,

please prepare a duplicate copy of this form. Date 11/8

Table of Contents

1.0 Introduction

Learning the Python language is by far the most useful thing a new programmer could do, not only has it had a great impact but it's also very simple and easy to learn, therefore we designed a simple program that we will be talking about.

1.1 Overview

the simple quiz system is a program just like its name allows you to take a specifically built and designed quiz, get feedback, observe scores of others and edit input questions. The goal of this program is to practice everything we know in Python up until now and to work like a team in developing the structure.

2.0 System design

1. main.py:

Runs the main program, displays menus to the user, and handles interactions between other modules (e.g., starting the quiz, viewing the leaderboard, admin mode).

2. quiz.py:

Contains quiz logic: loading questions, displaying questions randomly, calculating scores, and providing feedback.

3. admin.py:

Manages the admin functionalities and allows adding, editing, and deleting questions. Requires admin password to access.

4. utils.py:

Contains helper functions such as input validation, formatting time, and randomizing questions.

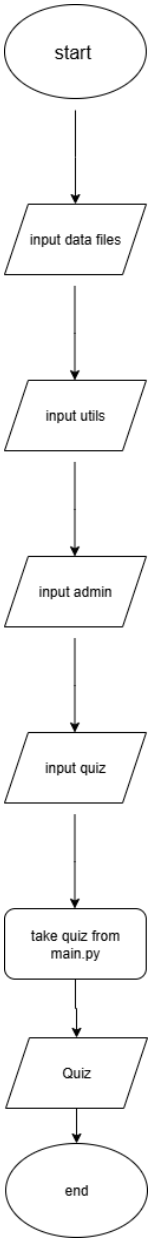
5. leaderboard.py:

Manages storing and displaying user results on the leaderboard.json, Includes functions for saving results (save_result) and viewing the leaderboard (view_leaderboard).

2.1 File structure:

```
main.py              # Main driver script to run the application
quiz.py              # Quiz-related logic: loading, displaying questions, and calculating scores
admin.py             # Admin functionalities: add, edit, delete questions
utils.py             # Utility functions: input validation, time formatting, etc.
data/
    questions.json     # JSON file storing quiz questions
    leaderboard.json   # JSON file storing leaderboard data
```

2.2 Quiz structure flowchart:



2.3 Samples:

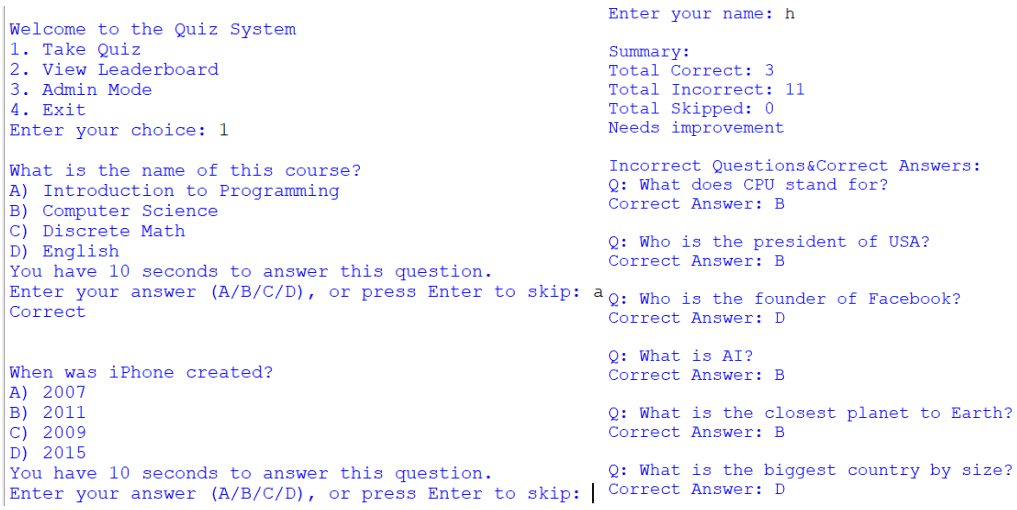


Figure 1 and 2

3.0 Individual contributions

Member 1: Hussain Khan:

- Data files
- Utils.py
- Quiz.py
- Leader

Member 2: Abdullah alsahli:

- Admin.py
- Readme
- Leaderboard.py

Member 3: Salah EL-Ghobbas

- Main.py
- Report
- Testing

4.0 **Testing and results**

The system was first tested individually each one tested their parts: after writing each function or module a test was performed, and after finishing each file the whole program is tested consistently by member 3.

An example test case is the first question file, the file had to be tested to see if it was working or not , so in utils load_questions does the testing perfectly , if the question file wasn't working utils wouldn't be working as well.

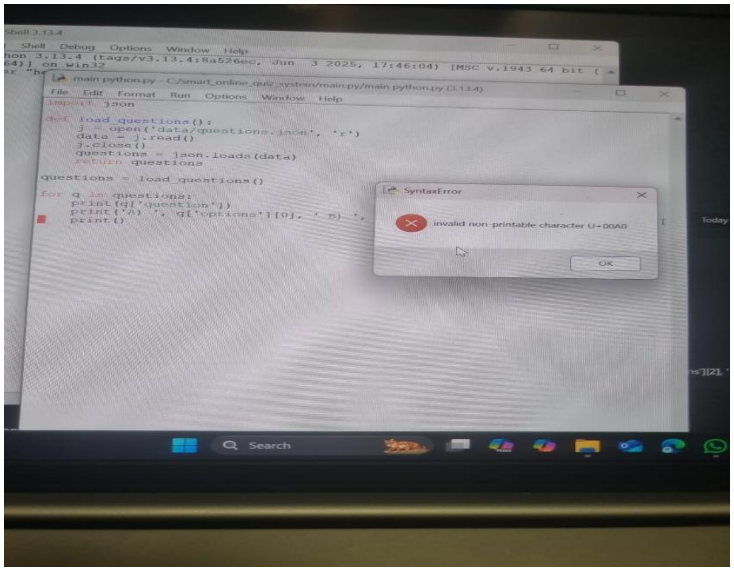


Figure 3: an example of a test outcome

5.0 Challenges faced

Many challenges was crossed by when designing the system, for example It was a bit challenging to guarantee the admin entered the correct data and avoid mistakes during loading or saving the questions file , and there was slight overlapping logic that needed manual adjustments, but the biggest one was we couldn't figure out how to merge all the codes in a single file without having errors in different places.

6.0 Conclusion

Building a full system is definitely not as easy as it seems, but the outcome of the build is interesting and it's a great practice for all the members, which everyone learned many useful stuff, like verifying admin input first can prevent mistakes in the questions database, or Using modular programming to make codes more organized by separating logic into different files but with proper planning and finally the simpleness of finding teaching sources that can help you with the process

And after evaluating the codes and our effort we came out with ideas that we suggest using in future programs, for example Adding an admin panel with buttons and forms instead of typing commands or Expand the different type of questions asked allowing constant new questions each time.

In the end, for a first project we can say that it was a great experience for all of us, hope that would be the same for our quiz takers.

7.0 Appendix

Source code:

Utils.py:

```
import json
def load_questions():
    f=open("data/questions.json", "r")
    questions=json.load(f)
    f.close()
    return questions
```

leaderboard.py:

```
import json
from datetime import datetime

def save_result(name,score):
 f=open("data/leaderboard.json","r")
 leaderboard=json.load(f)
 f.close()
 now=datetime.now()
 entry={"name":name,"score":score,"time":str(now)}
 leaderboard=leaderboard+[entry]

 for i in range(len(leaderboard)):
  for j in range(i+1,len(leaderboard)):
   if leaderboard[j]["score"]>leaderboard[i]["score"]:
    temp=leaderboard[i]
    leaderboard[i]=leaderboard[j]
```

```
    leaderboard[j]=temp

 leaderboard=leaderboard[:10]
 f=open("data/leaderboard.json","w")
 json.dump(leaderboard,f)
 f.close()

def view_leaderboard():
 f=open("data/leaderboard.json","r")
 leaderboard=json.load(f)
 f.close()
 print("Leaderboard:")
 print()
 for i in range(len(leaderboard)):
  entry=leaderboard[i]
  print(i+1,entry["name"],"| Score:",entry["score"],"| Time:",entry["time"])
print()
```

admin.py:

```
import json

ADMIN_PASSWORD="bic1144"
QUESTIONS_FILE="data/questions.json"

def save(questions):
 with open(QUESTIONS_FILE,"w")as f:json.dump(questions,f)

def load():
 with open(QUESTIONS_FILE,"r")as f:return json.load(f)
```

```
def add_question(questions):
 q=input("Question: ")
 opts=[]

 print("Enter the options:")
 opts.append(input("A: "))
 opts.append(input("B: "))
 opts.append(input("C: "))
 opts.append(input("D: "))
 ans=input("the Correct answer (A/B/C/D): ").upper()

 if ans not in["A","B","C","D"]:
  print("Invalid.");return
 questions.append({"question":q,"options":opts,"correct_answer":ans})
 save(questions)
 print("Added.")

def search_edit_delete(questions):
 k=input("enter the Keyword: ").lower()
 found=False
 for i in range(len(questions)):
  q=questions[i]
  if k in q["question"].lower():
   print(q["question"])
   print("A:",q["options"][0])
   print("B:",q["options"][1])
   print("C:",q["options"][2])
   print("D:",q["options"][3])
   print("Correct:",q["correct_answer"])
```

```
   act=input("[E] to edit, [D]to delete, [Enter] to Cancel: ").upper()

   if act=="E":
    new_q=input("New Question (leave to skip): ")

    if new_q:q["question"]=new_q

    for n in range(4):
     v=input(f"New {['option A','option B','option C','option D'][n]} (leave to skip): ")
     if v:q["options"][n]=v

    a=input("New Answer (A/B/C/D, leave to skip): ").upper()
    if a in["A","B","C","D"]:q["correct_answer"]=a
    print("Updated.")

   elif act=="D":
    if input("Confirm delete (Y): ").upper()=="Y":
     questions.pop(i)
     print("Deleted.")

   save(questions)
   found=True
   break

 if not found:print("Not found.")

def admin_menu():
 if input("Admin password: ")!=ADMIN_PASSWORD:
  print("Wrong password.");return
 questions=load()
```

```
 while True:
  print("\n1) Add\n2) Search/Edit/Delete\n3) Exit")
  c=input("Choice: ")
  if c=="1":add_question(questions)
  elif c=="2":search_edit_delete(questions)
  elif c=="3":break
  else:print("Wrong input.")
```

quiz.py:

```
import time
import random
from utils import load_questions


def calculate_score(score,total,incorrect,skipped):
    print()
    print("Summary:")
    print("Total Correct:",score)
    print("Total Incorrect:",len(incorrect))
    print("Total Skipped:",skipped)
    percentage=(score/total)*100
    if percentage>=90:
        print("Excellent")
    elif percentage>=70:
        print("Very good")
    elif percentage>=50:
        print("Fair")
    else:
        print("Needs improvement")
    if incorrect:
```

```
        print()
        print("Incorrect Questions&Correct Answers:")
        for q in incorrect:
            print("Q:",q["question"])
            print("Correct Answer:", q["correct_answer"])
            print()

def take_quiz():
    score=0
    questions=load_questions()
    random.shuffle(questions)
    time_limit=10
    incorrect_questions=[]
    skipped=0

    for q in questions:
        print()
        print(q["question"])
        print("A)", q["options"][0])
        print("B)", q["options"][1])
        print("C)", q["options"][2])
        print("D)", q["options"][3])
        print(f"You have {time_limit} seconds to answer this question.")

        start_time=time.time()
        answer=input("Enter your answer (A/B/C/D), or press Enter to skip: ").upper()
        end_time=time.time()
        taken_time=end_time-start_time

        if answer=="":
```

```
            print("Question skipped")
            skipped+=1
            print()
            continue
        if taken_time > time_limit:
            print("Time is up, Your answer was not counted.")
            skipped+=1
        elif answer == q["correct_answer"]:
            print("Correct")
            score+= 1
        else:
            print("Wrong answer")
            incorrect_questions=incorrect_questions+[q]
        print()

    return score,incorrect_questions
```

main.py

```
from quiz import take_quiz, calculate_score
from leaderboard import save_result, view_leaderboard
from admin import admin_menu

while True:
    print("Welcome to the Quiz System")
    print("1. Take Quiz")
    print("2. View Leaderboard")
    print("3. Admin Mode")
    print("4. Exit")
    choice=input("Enter your choice: ")
```

```
    if choice=="1":
        score,incorrect_questions=take_quiz()
        name=input("Enter your name: ")
        save_result(name,score)
        calculate_score(score,len(incorrect_questions),incorrect_questions,0)

    elif choice=="2":
        view_leaderboard()

    elif choice=="3":
        admin_menu()

    elif choice=="4":
        print("Goodbye")
        break
    else:
        print("Invalid choice. Try again.")
```

data file sample:

question.json:

[

{"question": "What is the name of this course?", "options": ["Introduction to Programming", "Computer Science", "Discrete Math", "English"], "correct_answer": "A"}

]

Leaderboard.json:

[
{"name": "h", "score": 3, "time": "2025-08-12 05:44:31.590862"
]