**Flashcards Project Portfolio**

Table of Contents

[**1. Analysis** 3](#_Toc160201901)

[*1.1. Problem Overview:* 3](#_Toc160201902)

[*1.2. Existing Solutions:* 3](#_Toc160201903)

[1.2.1 Solution 1: 3](#_Toc160201904)

[*1.3. Market Research:* 4](#_Toc160201905)

[*1.4. Objectives:* 4](#_Toc160201906)

[**2. Design** 5](#_Toc160201907)

[*IPSO Diagram:* 5](#_Toc160201908)

[**3. Technical Solution** 6](#_Toc160201909)

[*3.1. Completeness of Solution:* 6](#_Toc160201910)

[*3.2. Technical Skills:* 6](#_Toc160201911)

[**4. Testing** 7](#_Toc160201912)

[**5. Evaluation** 12](#_Toc160201913)

[What I did: 12](#_Toc160201914)

[What I could have done: 12](#_Toc160201915)

[What was out of my scope: 12](#_Toc160201916)

# **1. Analysis**

## *1.1. Problem Overview:*

My project is going to be about a flash card quiz. It is going to provide questions to the user to test their knowledge.

## *1.2. Existing Solutions:*

### 1.2.1 Solution 1:

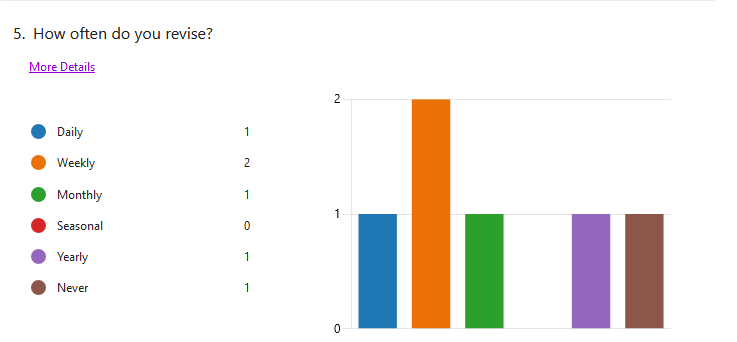
Seneca Learning:

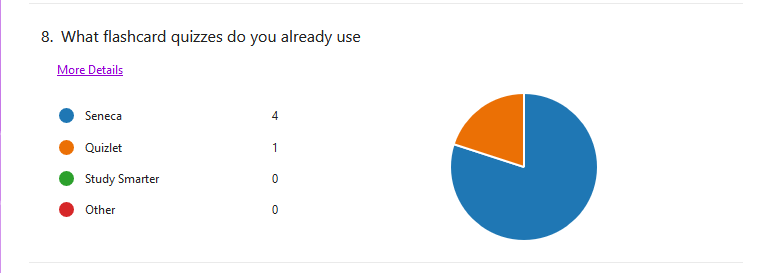


* Seneca gives the user information in forms of flashcards.
* It then asks question in either multiple choice, fill in the gaps, or sliders.
* When done, it gives you a percentage of answers correct.
* It has a user interface that makes it look nice and easy to use.

## *1.3. Market Research:*

<https://forms.office.com/e/F3hSYKQLdm>





## *1.4. Objectives:*

1. The basic parts of the project are:
   1. It must be able to read/modify .txt files
   2. Link cards out of keywords and definitions (in .txt file)
   3. Make multiple choice questions about the cards.
      1. Give the definition and ask for keyword
      2. Give the Keyword asks for definition
      3. Questions will be in a random order
      4. If done correctly multiple times it will remove it from the .txt and add it to a different completed .txt file
   4. A menu to allow re/starting or exiting of the quiz
   5. A score system to track users incorrect answers
   6. A timer to see how long it takes, in the form h:m:s
2. The project would be nice with:
   1. An interactive GUI
   2. Multiple .txt files for different subjects and being able to choose between them
   3. being able to create/delete .txt files

# **2. Design**

## *IPSO Diagram:*

|  |  |
| --- | --- |
| **IPSO** | **Information** |
| Input | Text File User input |
| processes | handling text files Proccessing User interface |
| Storage | Keywords Definitions |
| Output | Feedback (in/correct) Time taken (h:m:s) |

## *User interface design*;

I plan to make a graphical user interface for my code. It will look a bit like this:  
**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

# **3. Technical Solution**

## *3.1. Completeness of Solution:*

I Completed all of the main objectives I set and 2 of the 3 secondary ones.

* The code can modify and read .txt files
* It makes a 2d array out of the keywords and definitions in the file
* It can make the questions.
  + Getting a random keyword
  + Getting its definition and 2 other random ones
  + Displaying them
* When they are done multiple (2) times they are not asked unless the quiz is restarted
* When finished it shows time and the number they got incorrect

it also has a GUI and can use multiple text files

## *3.2. Technical Skills:*

I made an interactive Graphical User Interface (GUI) that utilises custom Tkinter (CTk).   
this consist of multiple windows Menu, File picker, Question widow and End.

Menu code:  
A computer screen shot of text

Description automatically generated

This is the menu code and it makes the menu which is the actual CTk app that the whole code runs off.

Question Window:

1. def question\_widow(tk=None, Qnum=1, ArrayValue=0):

2.     global incorrect

3.     global Quiz

4.     global finished

5.     hide(tk)

6.

7.     ab1 = str("Empty")

8.     ab2 = str("Empty")

9.     ab3 = str("Empty")

10.     for i in range(len(info)):

11.         if info[i][2] >= 2:

12.             info[i][0] = None

13.     global finished

14.     finished = True

15.     for i in info:

16.         if i[0] != None:

17.             finished = False

18.     trys = 0

19.     while info[ArrayValue][0] == None:

20.         ArrayValue = randint(0, len(info)-1)

21.         if trys >= 6:

22.             endScreen(correct, Quiz)  # type: ignore

23.             break

24.         trys += 1

25.

26.     value1 = randint(0, len(info)-1)

27.     value2 = randint(0, len(info)-1)

28.

29.     match randint(1, 3):

30.         case 1:

31.             ab1 = info[ArrayValue][1]

32.             while info[value1][1] == info[ArrayValue][1]:

33.                 value1 = randint(0, len(info)-1)

34.             while info[value2][1] == info[ArrayValue][1] or info[value2][1] == info[value1][1]:

35.                 value2 = randint(0, len(info)-1)

36.             ab2 = info[value1][1]

37.             ab3 = info[value2][1]

38.         case 2:

39.             ab2 = info[ArrayValue][1]

40.             while info[value1][1] == info[ArrayValue][1]:

41.                 value1 = randint(0, len(info)-1)

42.             while info[value2][1] == info[ArrayValue][1] or info[value2][1] == info[value1][1]:

43.                 value2 = randint(0, len(info)-1)

44.             ab1 = info[value1][1]

45.             ab3 = info[value2][1]

46.         case 3:

47.             ab3 = info[ArrayValue][1]

48.             while info[value1][1] == info[ArrayValue][1]:

49.                 value1 = randint(0, len(info)-1)

50.             while info[value2][1] == info[ArrayValue][1] or info[value2][1] == info[value1][1]:

51.                 value2 = randint(0, len(info)-1)

52.             ab1 = info[value1][1]

53.             ab2 = info[value2][1]

54.     if Qnum == 1:

55.         # makes the question window

56.         Quiz = CTkToplevel()

57.         Quiz.geometry("500x400+750+300")

58.         Quiz.resizable(False, False)

59.         Quiz.title("F.C.Q")

60.         start\_timer()

61.         correct = 0

62.     TimerLabel = CTkLabel(master=Quiz, text=str(

63.         get\_timer()), font=("Arial", 10))

64.     if Qnum != 1:

65.         for widgets in Quiz.winfo\_children():

66.             if widgets == TimerLabel:

67.                 continue

68.             widgets.destroy()

69.     Quiz.bind("<Escape>", lambda x: sys.exit())

70.     Title = CTkLabel(master=Quiz, text=f"Question {Qnum}",

71.                      font=("Arial", 20))

72.     Title.place(relx=0.5, rely=0.05, anchor="center")

73.

74.     TimerLabel.place(relx=0.8, rely=0.05, anchor="center")

75.     question = CTkLabel(master=Quiz, text=info[ArrayValue][0], font=(

76.         "Arial", 15)).place(relx=0.5, rely=0.15, anchor="center")

77.     Abutton1 = CTkButton(master=Quiz, text=ab1, command=lambda: Answer\_Buttons(

78.         1, Abutton1, Abutton2, Abutton3, ContBtn, ArrayValue))

79.     Abutton1.place(relx=0.5, rely=0.3, anchor="center")

80.

81.     Abutton2 = CTkButton(master=Quiz, text=ab2, command=lambda: Answer\_Buttons(

82.         2, Abutton1, Abutton2, Abutton3, ContBtn, ArrayValue))

83.     Abutton2.place(relx=0.5, rely=0.4, anchor="center")

84.

85.     Abutton3 = CTkButton(master=Quiz, text=ab3, command=lambda: Answer\_Buttons(

86.         3, Abutton1, Abutton2, Abutton3, ContBtn, ArrayValue))

87.     Abutton3.place(relx=0.5, rely=0.5, anchor="center")

88.

89.     ContBtn = CTkButton(master=Quiz, text="Continue",

90.                         fg\_color="#555555", text\_color="#000000", command=lambda: cont(Quiz, Qnum, incorrect),  state="disabled")

91.     ContBtn.place(relx=0.5, rely=0.6, anchor="center")

92.     ExitBtn = CTkButton(master=Quiz, text="Exit",

93.                         fg\_color="#550000", text\_color="#000000", command=lambda: Menu(Quiz)).place(relx=0.5, rely=0.7, anchor="center")

94.     Quiz.after(1000, update, Quiz, TimerLabel)

95.

This is the question window it gets the values for the buttons if there will be any and makes the window with the buttons. It uses a match statement to place the text on the right buttons.

End screen:

A computer screen shot of text

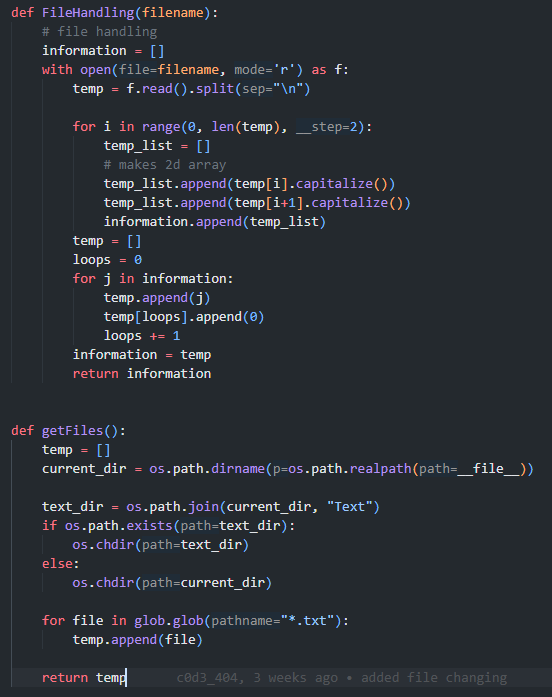
Description automatically generated

This is the end screen it displays final time and number of incorrect answers.

File handling:  
UI.py:  
A screen shot of a computer code

Description automatically generated

funcs.py:



I make a file handling system that could let the user pick between different files.

A screenshot of a computer program

Description automatically generated 

This is the answer handling it consists almost all of a match statement check if the answer is correct

# **4. Testing**

|  |  |
| --- | --- |
| Test 1: |  |
| Test 2: |  |
| Test 3: |  |
| Test 4: |  |
| Test 5: |  |
| Test 6: |  |
| Test 7: |  |
| Test 8: |  |
| Test 9: |  |
| Test 10: |  |
| Test 11: |  |
| Test 12: |  |
| Test 13: |  |

In the page below is the test table for these images/videos.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test #** | **Description** | **Expected result** | **Actual result** | **Pass/fail** | **Evidence (above)** |
| 1 | Loading of the menu | The menu to load when the code is run | The menu loads. | Pass | test1.png |
| 2 | Changing screens | The start button needs to change the screen to the question screen. The exit  button on question screen goes back to the menu and the exit on menu quits the  application | When the start button is clicked the menu closes opening the  question screen. The exit button on the screen closes the question screen and reopens the menu. The exit on the menu closes the  application | Pass | test2.mp4 |
| 3 | Icon | Custom icon to be in corner of  application | The icon is in the corner | Pass | test3.png |
| 4 | File changing | Being able to choose between all available .txt files and the last selected one to be shown | I can choose from available files  but can't see what was last picked | Fail | test4.mp4 |
| 5 | File changing test 2 | Being able to choose between all available .txt files and the last selected one to be shown | I can choose from available files  and can see what file was last picked | Pass | test5.mp4 |
| 6 | Buttons showing  definitions | The 3 buttons should show the  definitions on them one of them will be the correct one and two other will be incorrect with no duplicates | The buttons showed the correct  one but numbers for the two  incorrect ones. | Fail | Test6.png |
| 7 | Buttons showing  definitions test 2 | The 3 buttons should show the  definitions on them one of them will be the correct one and two other will be incorrect with no duplicates | The 3 buttons showed the  definitions on them one them  was the correct one and the two other were incorrect with no duplicates | Pass | Test7.png |
| 8 | Buttons showing  correct answer | The 3 buttons should go red/ green depending on which is correct | The 3 buttons go red/ green  depending on which is correct | Pass | Test8.png |
| 9 | Continue button working | The continue button shouldn’t do anything until an answer is chosen. Then it will allow the user to go to the next question. | The continue button doesn't anything. | Fail | Test9.mp4 |
| 10 | Continue button working  test 2 | The continue button shouldn’t do anything until an answer is chosen. Then it will allow the user to go to the next question. | The continue button doesn't anything until I want it to. But when I change screen I get an infinitely recursive error. | Fail | Test10.mp4 |
| 11 | Continue button working test 3 | The continue button shouldn’t do anything until an answer is chosen. Then it will allow the user to go to the next question. | The continue button doesn't anything until I want it to. When a question is answered it allows me to go to the next one | Pass | Test11.mp4 |
| 12 | Continue button working test 4 | when you answer every question twice it should continue | it works until the last one then  it only shows the last one | Fail | Test12.png |
| 13 | Continue button working test 5 | when you answer every question twice  it should continue | it works completely showing the to be designed end screen | Pass | Test13.png |

# **5. Evaluation**

## What I did:

I have made a working GUI that can change between different screens.

I made 4 screen Menu, File Picker, Questions, and end this are used throughout the code.

I added a function to be able to change the active file meaning that you don’t have to restart the app and change the file as long as the code has access to the .txt files.

## What I could have done:

I could have added the functionality to add or remove files from the application making it not needed to provide the text files before running.

More planning before the program was made as I don’t think I did enough.

Added scaling and Fullscreen to the app.

I could have made the code a bit neater.