INVESTIGATORY PROJECT

TestApp

2019-20

SUBMITTED BY

VIGNESH BALAJI S
XII A

UNDER THE GUIDANCE OF

Smt. R. SriKeerthy
PGT(Computer Science)
Department of Computer Science
Kendriya Vidyalaya HVF, Avadi
Chennai-54

CERTIFICATE

This is to certify that ***VIGNESH BALAJI S*** of class ***XII A*** has successfully prepared the report on the project titled "**TestApp**" for the academic year **2019-2020**. This report is the result of his efforts and endeavours. The report is found worthy of acceptance as final project report for the subject ***Computer Science*** of class XII.

Internal examiner **External examiner** **Principal**

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to ***Smt. R. SriKeerthy*** for guiding me immensely through the course of the project. She always evinced keep interest in my work. Her constructive advice and constant motivation have been responsible for the successful completion of this project.

My sincere thanks goes to our principal, ***Mr. S. Arumugam*** for his co-ordination in extending every possible support for the completion of this project.

I would also like to thank my parents and friends for their constant support, timely help and patience to sit through my doubts and listen. I would like to thank all those who helped me directly or indirectly towards the completion of this project.

CONTENTS

HARDWARE/SOFTWARE SPECIFICATIONS

HARDWARE USED:

- ♦ Laptop
  - CORE i5
  - 15.6 inch screen
  - RAM: 8GB
  - SSD: 512GB
- ♦ Mouse
- ♦ Pendrive
- ♦ Printer

SOFTWARE USED:

- ♦ Windows 10 (64 bit)
- ♦ Sublime Text (64 bit)
- ♦ Python 3.7 (64bit)
- ♦ Photoshop CC 2019(64 bit)
- ♦ MS Paint
- ♦ MS Notepad
- ♦ MS Calculator
- ♦ Picsart (android)

INTRODUCTION

Millions of trees are uprooted every year for the sake of manufacturing papers.
Schools are the major consumer of papers.
If every school conduct their tests through a software, millions of trees can be saved.
This project **testApp** is mainly based on this notion.

testApp is an online application where teachers can create and publish a test assignment with an unique testId.
Teachers can share this testId to students and hence students will be able to access the created test by entering the testId.
Students can write their test sitting at home through their computers.
Once they complete the test, their response will be evaluated automatically and the data of their marks will be saved in the database, and hence teachers can view their student's performance whenever they want by simply entering the testId created by them.

This program is developed in such a way where only multiple choice questions can be asked, this makes the evaluation process quick and accurate.
So much of time can be saved as the evaluation is an automatic process.
Students can come to know about their results so quickly so that they can focus on the test analysis.

In today's world, most of the competitive exams are of MCQ type, so it'd be a great practice for the students.

WORKING DESCRIPTION

When a teacher creates a test, he/she has to give an unique id for the test and after filling out all the details for creating a test like marks,time,no. of qs,etc.
A test window will be opened , in that window, the teacher can type or paste the question and options in the respective text boxes, and he/she has to select the correct option, which is very important for the evaluation process.
Once the teacher fills all the details, he/she can submit it.

After submitting, a new worksheet will be created in the google spreadsheet which is the primary database for my application. All the data of the created test will be stored in the worksheet which is also named after its testID.

When a student wants to write the exam, he/she has to enter his/her name and if they have'nt written the test already , a test window will be opened, where the student can choose the right answer by just clicking the option button, after giving all of his/her response, one can click the submit button.

After submitting, in a new worksheet which is exclusively created for that particular test, all the evaluated marks of the students will be entered automatically corresponding to their names.

When a teacher wishes to view the performance of the students, he/she can enter the testID and they can download a bar graph showing marks of all the students corresponding to their respective names.

If a teacher wishes to remove/delete an existing test, she can click the manage database option to enter into the spreadsheet, and can delete the worksheet of the test which he/she wished to be removed from the database.

Once deleted, no student will be able to attempt the test.

LIBRARIES USED

♦ pygame
    -For visual effects and homepage UI.

♦ gspread
    -For controlling google spreadsheets.

♦ oauth2client.service_account
    -For connecting python with google spreadsheets.

♦ tkinter
    -For UI.

♦ datetime
    -For taking record of date and time.

♦ pygetwindow
    -To switch between tabs opened in the windows OS.

♦ matplotlib.pyplot
    -To visually represent the result data.

♦ webbrowser
    -To open database in browser.

Source Code

functions.py

```
import pygame
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
from tkinter.ttk import Progressbar
import datetime
import pygetwindow as win
import time
import csv
from dbTest import *
from tkinter import filedialog
import matplotlib.pyplot as p
import numpy as np
import webbrowser
# from testApp import *
pygame.init()
gameDisplay = pygame.display.set_mode((1366, 768),pygame.FULLSCREEN)
pygame.display.set_caption('testApp')
clock = pygame.time.Clock()
black = (0, 0, 0)
white = (255, 255, 255)
purple = (255, 10, 246)
skyblue = (0, 215, 255)
lightPurple = (245, 150, 240)
green = (0, 200, 30)
green1 = (0, 200, 100)
homeButtonL = (200, 247, 12)
homeButtonR = (200, 247, 12)
red = (235,8,0)
yellow = (200, 247, 12)
exitColor, infoColor ,backColor,availableTestsColor,viewResultsColor = yellow, yellow, yellow,yellow,yellow
blue = (2, 59, 181)
backIcon = pygame.transform.scale(pygame.image.load('backIcon.png'), (35,35))
exitIcon = pygame.transform.scale(pygame.image.load('exitIcon.png'), (35, 35))
infoIcon = pygame.transform.scale(pygame.image.load('infoIcon.jpg'), (35, 35))
vigbalityBG = pygame.transform.scale(pygame.image.load('vigbalityBG.jpg'), (1366, 768))
testappBG = pygame.transform.scale(pygame.image.load('testappBG.jpg'), (1366, 768))
kvhvfBgImg = pygame.transform.scale(pygame.image.load('kvhvfBgImg.jpg'), (1366, 768))
def minimizeAll():
  taskList = win.getAllTitles()
  for task in taskList:
     hwnd = win.getWindowsWithTitle(task)[0]
     hwnd.minimize()
def navigateTo(window):
  # appWindow =
  taskList = win.getAllTitles()
  for task in taskList:
    if window in task:
      appWindow = win.getWindowsWithTitle(task)[0]
      break
  appWindow.maximize()
def preDataForCreatingTest():
  minimizeAll()
  mainroot=Tk()
  mainroot.title('Text Editor For TestApp')
  mainroot.state('zoomed')
  mainroot.configure(background='#0c0c0c')
  mainroot.overrideredirect(True)
  inputPanel = Frame(mainroot,bg = '#3a3b33')
  inputPanel.place(height = 370,width = 370,x =500,y = 185)

  nameOfTestLabel = Label(inputPanel,text = 'Name of the test:',font = ('Helvetica', '15'),bg
='#a6ff00',border=0,relief ='raised')
  nameOfTestLabel.place(x=30,y=30,height=30,width = 180)
  nameOfTest= ttk.Entry(inputPanel,font = ('Helvetica', '12'))
  nameOfTest.place(x=210,y=30,height=30,width = 130)
  subjectNameLabel = Label(inputPanel,text = 'Subject:',font = ('Helvetica', '15'),bg ='#a6ff00',border=0,relief
='raised')
  subjectNameLabel.place(x=30,y=70,height=30,width = 160)
  subjectName= ttk.Entry(inputPanel,font = ('Helvetica', '12'))
  subjectName.place(x=190,y=70,height=30,width = 150)
  markForEachQuesLabel = Label(inputPanel,text = 'Marks awarded for each correct answer:',font = ('Helvetica',
'11'),bg                                          ='#a6ff00',border=0,relief ='raised')
  markForEachQuesLabel.place(x=30,y=110,height=30,width = 270)
  markForEachQues= ttk.Entry(inputPanel,font = ('Helvetica', '12'))
  markForEachQues.place(x=300,y=110,height=30,width = 40)
  noOfQuesLabel = Label(inputPanel,text = 'Number of questions:',font = ('Helvetica', '12'),bg
='#a6ff00',border=0,relief ='raised')
```

```
  noOfQuesLabel.place(x=30,y=150,height=30,width = 160)
  noOfQues = ttk.Combobox(inputPanel,values =(5,10,15,20,25,30),font = ('Helvetica', '15'))
  noOfQues.place(x=190,y=150,height=30,width = 150)
  timeAllowedLabel = Label(inputPanel,text = 'Time allowed(HH:MM):',font = ('Helvetica', '12'),bg
='#a6ff00',border=0,relief ='raised')
  timeAllowedLabel.place(x=30,y=190,height=30,width = 160)
  timeAllowedHH = ttk.Combobox(inputPanel,values =(0,1,2,3,4),font = ('Helvetica', '15'))
  timeAllowedHH.place(x=190,y=190,height=30,width = 70)
  timeAllowedMM = ttk.Combobox(inputPanel,values = tuple(range(0,60,5)),font = ('Helvetica', '15'))
  timeAllowedMM.place(x=265,y=190,height=30,width = 75)
  testId=[]
  def checkData():
    nonlocal mainroot
    nonlocal testId
    allOk= 0
    def entryData():
      nonlocal mainroot
      progress=Progressbar(mainroot,orient=HORIZONTAL,length=100,mode='determinate')
      progress.place(x= 500,y = 415 ,width = 370,height = 40)
      for i in range(1,30):
       progress['value']=i
       mainroot.update_idletasks()
       time.sleep(0.05)
      progress['value']=35
      mainroot.update_idletasks()
      time.sleep(1)
      progress['value']=40
      mainroot.update_idletasks()
      time.sleep(1)
      progress['value']=50
      mainroot.update_idletasks()
      time.sleep(1)
      try:
        testDetails=[]
        testDetails.append(nameOfTest.get())
        testDetails.append(subjectName.get())
        testDetails.append(markForEachQues.get())
        testDetails.append(noOfQues.get())
        testDetails.append(timeAllowedHH.get())
        testDetails.append(timeAllowedMM.get())
        testDetails.append(datetime.datetime.now().strftime("%d:%m:%y"))
        rowStr = int(noOfQues.get())
        rangeValue = int(noOfQues.get())+1
        global dataFile
        worksheet = dataFile.add_worksheet(title=testDetails[0], rows=str(rowStr+1), cols="7")
        cell_list = worksheet.range('A{0}:G{1}'.format(rangeValue,rangeValue))
        ind=0
        for cell in cell_list:
         cell.value =testDetails[ind]
         ind+=1
        worksheet.update_cells(cell_list)
        global availableTests
        availableTestsWorksheet=availableTests.worksheet("mainSheet")
        rangeValue=vacantRow(availableTestsWorksheet)
        cell_list = availableTestsWorksheet.range('A{0}:G{1}'.format(rangeValue,rangeValue))
        ind=0
        for cell in cell_list:
         cell.value =testDetails[ind]
         ind+=1
        availableTestsWorksheet.update_cells(cell_list)
        global resultData
        resultData.add_worksheet(title=testDetails[0], rows="100", cols="2")
        for i in range(50,101):
          progress['value']=i
          mainroot.update_idletasks()
          time.sleep(0.05)
        mainroot.destroy()
    except:
        messagebox.showinfo('Oops',"Something went wrong!\nPlease check your internet connectivity and try again.")
        progress.destroy()
        pass
    try:
      dummy =int(markForEachQues.get())+int(noOfQues.get())+int(timeAllowedHH.get())+int(timeAllowedMM.get())

      if int(noOfQues.get())  not in [5,10,15,20,25,30]:
        messagebox.showinfo('Warning!',"1)Number of questions should be a multiple of 5\n2)Should be less than or
equal to 30")
      else:
        allOk+=1

      dummyTestName= str(nameOfTest.get())
      if dummyTestName.isalpha():
        allOk+=1
      else:
        messagebox.showinfo('Warning!',"1)Test name can contain no numbers.\n2)Test name can contain no white
spaces.'")

      try:
        dummyWorksheetCheck = dataFile.worksheet(str(nameOfTest.get()))
        messagebox.showinfo('Warning!',"1)Test Name already exists.\n2)Try deleting old tests from database.\n3)Try a
different name.")
```

```
      except:
        allOk+=1

      if int(timeAllowedHH.get())  not in (0,1,2,3,4):
        messagebox.showinfo('Warning!',"1)Maximum time allowed is '4 hrs 55 mins'")
      else:
        allOk+=1
      if allOk==4:
        testId.append(nameOfTest.get())
        testId.append(int(noOfQues.get()))
        entryData()
    except:
      messagebox.showinfo('Warning!',"1)Please enter numeric values wherever required.")


  def goBack():
    navigateTo('testApp')
    mainroot.destroy()
  backButton=Button(mainroot, text="BACK",command= goBack,font = ('Helvetica', '20'),bg
='#a6ff00',fg='black',border=0)
  backButton.place(height = 40, width = 80,x = 20,y = 20)
  doneButton=Button(inputPanel, text="DONE",command= checkData, height=2, width=13,font = ('Helvetica', '30'),padx =
200,pady = 200,border=0,relief ='raised',bg ='#a6ff00')
  doneButton.place(height = 70, width = 180,x = 95,y = 290)
  mainroot.mainloop()
  return testId
def createTest(testId):
  optionText ='please type/paste option {} here'
  questionText ='Please type/paste the question here...'

preText=[1,questionText,optionText.format('A'),optionText.format('B'),optionText.format('C'),optionText.format('D'),'A
']

  root=Tk()
  root.title('CREATE TEST')
  root.state('zoomed')
  root.configure(background='#0c0c0c')
  root.overrideredirect(True)



  dataDict={}
  def nextButtonFunc():
      nonlocal testId
      dataDict.update({currentQuesNum.get():[currentQuesNum.get(), questionBox.get("1.0",'end-1c'),
optionA.get("1.0",'end-1c'), optionB.get("1.0",'end-1c'), optionC.get("1.0",'end-1c'), optionD.get("1.0",'end-1c'),
var.get()]})
      globals()['b%s' % currentQuesNum.get()].configure(bg = '#15ff0d')
      if currentQuesNum.get() ==testId[1]:
        if len(dataDict)==testId[1]:
          messagebox.showinfo('Info',"Looks like you've reached the end.\nClick SUBMIT to continue. ")
        else:
          for i in range(1,testId[1]+1):
           if i not in dataDict.keys():
            globals()['b%s' %i].invoke()
            break
      else:
        for i in range(currentQuesNum.get(),testId[1]+1):
          if i not in dataDict.keys():
            globals()['b%s' %i].invoke()
            break




  def submitButtonFunc():
      nonlocal root

      progress=Progressbar(root,orient=HORIZONTAL,length=100,mode='determinate')
      progress.place(x= 10,y = 430 ,width = 1050,height = 40)
      for i in range(1,30):
       progress['value']=i
       root.update_idletasks()
       time.sleep(0.05)
      progress['value']=35
      root.update_idletasks()
      time.sleep(1)
      progress['value']=40
      root.update_idletasks()
      time.sleep(1)
      progress['value']=50
      root.update_idletasks()
      time.sleep(1)
      try:
        global dataFile
        nonlocal testId
        strr=str(testId[1])
        worksheet = dataFile.worksheet(testId[0])
        cell_list = worksheet.range('A1:G{}'.format(strr))
        row=1
        col=0
```

```
        for cell in cell_list:
          cell.value =dataDict[row][col]
          if col ==6:
            row+=1
            col =0
          else:
            col+=1
          worksheet.update_cells(cell_list)

        for i in range(50,101):
          progress['value']=i
          root.update_idletasks()
          time.sleep(0.05)
        messagebox.showinfo('Info',' Succesfully Saved!\nClick OK to continue with TestApp.')
        root.destroy()
        navigateTo('testApp')
      except:
        messagebox.showinfo('Oops',"Submission failed\nPlease check your internet connectivity and try again.")
        progress.destroy()
        pass


  sNo =1
  ques = Label(root , text = 'Q' +str(sNo) +')', font = ('Helvetica', '20'),bg ='#a6ff00')#str(preText[0])
  ques.place(x =10 , y = 30,height = 400, width = 60)
  questionBox=Text(root,bg ='lavender',font = ('Helvetica', '18'))
  questionBox.insert(INSERT, preText[1])
  questionBox.place(x = 70,y = 30, height = 400, width = 990)

  var = StringVar()
  var.set(preText[6])
  optA = Radiobutton(root , text = 'A', font = ('Helvetica', '25'),variable = var,value = 'A',bg = '#a6ff00'
,indicatoron = 0,selectcolor = 'green')
  optA.place(x =10 , y = 470,height = 100, width = 60)
  optionA =Text(root,bg ='lavender',font = ('Helvetica', '18'))
  optionA.insert(INSERT,preText[2])
  optionA.place(x=70,y = 470,height =100,width = 450)

  optB = Radiobutton(root , text = 'B', font = ('Helvetica', '25'),variable = var,value = 'B',bg =
'#a6ff00',indicatoron = 0,selectcolor = 'green')
  optB.place(x =10 , y = 590,height = 100, width = 60)
  optionB = Text(root,bg ='lavender',font = ('Helvetica', '18'))
  optionB.insert(INSERT,preText[3])
  optionB.place(x=70,y = 590,height =100,width = 450)
  optC = Radiobutton(root , text = 'C', font = ('Helvetica', '25'),variable = var,value = 'C',bg =
'#a6ff00',indicatoron = 0,selectcolor = 'green')
  optC.place(x =550 , y = 470,height = 100, width = 60)
  optionC = Text(root,bg ='lavender',font = ('Helvetica', '18'))
  optionC.insert(INSERT,preText[4])
  optionC.place(x=610,y = 470,height =100,width = 450)

  optD =Radiobutton(root , text = 'D', font = ('Helvetica', '25'),variable = var,value = 'D',bg =
'#a6ff00',indicatoron = 0,selectcolor = 'green')
  optD.place(x =550 , y = 590,height = 100, width = 60)
  optionD = Text(root,bg ='lavender',font = ('Helvetica', '18'))
  optionD.insert(INSERT,preText[5])
  optionD.place(x=610,y = 590,height =100,width = 450)

  rightPanel = Canvas(root, bg="#3a3b33", height=655, width=270)
  rightPanel.place(x =1080,y  =30)


  rightPanel.create_line(0, 100, 270, 100, fill="black")
  rightPanel.create_line(0, 102, 270, 102, fill="black")
  rightPanel.create_line(0, 440, 270, 440, fill="black")
  rightPanel.create_line(0, 442, 270, 442, fill="black")


  quickAcc= Label(root , text = 'Quick Access:', font = ('Helvetica', '10'),bg ='#a6ff00')
  quickAcc.place(x =1080 , y = 110,height = 20, width = 275)
  userOpt= Label(root , text = 'Options:', font = ('Helvetica', '10'),bg ='#a6ff00')
  userOpt.place(x =1080 , y = 450,height = 20, width = 275)

  nextButton=Button(root,padx = 200,pady = 200,border=0,font = ('Helvetica', '25'),bg ='#a6ff00', height=2, width=13,
text="SAVE &\n  NEXT",command= nextButtonFunc )
  nextButton.place(height = 80, width = 200,x = 1115,y = 480)

  submitButton=Button(root,padx = 200,pady = 200,border=0,font = ('Helvetica', '30'),bg ='#a6ff00', height=2,
width=13, text="SUBMIT",command= submitButtonFunc )
  submitButton.place(height = 80, width = 200,x = 1115,y = 585)


  quickAccessPanel = Frame(root,bg='#3a3b33')
  quickAccessPanel.place(height = 270,width = 270,x =1082,y = 135)
  quesQueue = []
  quesNum = IntVar()
  num = 1
  def quickAccButList():
    sNo = currentQuesNum.get()
    ques.configure(text = 'Q' +str(sNo) +')')
```

```
    try:
      globals()['b%s' % currentQuesNum.get()].configure(bg = 'red')
      questionBox.delete('1.0', END)
      optionA.delete('1.0', END)
      optionB.delete('1.0', END)
      optionC.delete('1.0', END)
      optionD.delete('1.0', END)
      questionBox.insert(INSERT, dataDict[sNo][1])
      optionA.insert(INSERT,dataDict[sNo][2])
      optionB.insert(INSERT,dataDict[sNo][3])
      optionC.insert(INSERT,dataDict[sNo][4])
      optionD.insert(INSERT,dataDict[sNo][5])
      var.set(dataDict[sNo][6])
    except:
      globals()['b%s' % currentQuesNum.get()].configure(bg = 'red')
      questionBox.delete('1.0', END)
      optionA.delete('1.0', END)
      optionB.delete('1.0', END)
      optionC.delete('1.0', END)
      optionD.delete('1.0', END)
      questionBox.insert(INSERT, preText[1])
      optionA.insert(INSERT,preText[2])
      optionB.insert(INSERT,preText[3])
      optionC.insert(INSERT,preText[4])
      optionD.insert(INSERT,preText[5])
      var.set(preText[6])




  num = 1
  currentQuesNum = IntVar()
  currentQuesNum.set(1)
  noOfQuesValue = testId[1]
  for r in range(int(noOfQuesValue/5)):
   for c in range(5):
      if num<10:
        globals()['b%s' % num] = Radiobutton(quickAccessPanel , text = '0{}'.format(str(num)), font = ('Helvetica',
'15'),bg = 'red',indicatoron = 0,selectcolor = '#a6ff00',command = lambda:quickAccButList(),variable=currentQuesNum,
value = num)
        globals()['b%s' % num].grid(row=r,column=c,padx=11,pady = 5)
      else:
        globals()['b%s' % num] = Radiobutton(quickAccessPanel , text = '{}'.format(str(num)), font = ('Helvetica',
'15'),bg = 'red',indicatoron = 0,selectcolor = '#a6ff00',command =
lambda:quickAccButList(),variable=currentQuesNum,value = num)
        globals()['b%s' % num].grid(row=r,column=c,padx=11,pady = 5)
      num+=1

  rightPanel.create_line(0, 379, 270, 379, fill="black")
  rightPanel.create_line(0, 381, 270, 381, fill="black")
  rightPanel.create_polygon(4, 385, 4,415,34,415,34,385,fill="#15ff0d")
  Label(rightPanel , text = 'Saved', font = ('Helvetica', '10'),bg ='#3a3b33',fg='white').place(x =50 , y = 389)

  rightPanel.create_polygon(130, 385, 130,415,160,415,160,385,fill="red")
  Label(rightPanel , text = 'Not Saved', font = ('Helvetica', '10'),bg ='#3a3b33',fg='white').place(x =170 , y = 389)
  root.mainloop()

  navigateTo('testApp')
#-------------------------------------------------------------------------------------------------------------------
def getStudentName():
  minimizeAll()
  mainroot=Tk()
  mainroot.title('User Input')
  mainroot.state('zoomed')
  mainroot.configure(background='#0c0c0c')
  mainroot.overrideredirect(True)
  inputPanel = Frame(mainroot,bg = '#3a3b33')
  inputPanel.place(height = 370,width = 370,x =500,y = 185)
  studentNameValue=''
  testNameValue=''

  studentNameLabel = Label(inputPanel,text = 'Student Name:',font = ('Helvetica', '18'),bg ='#a6ff00',border=0)
  studentNameLabel.place(x=20,y=70,height=40,width = 170)
  studentName= ttk.Entry(inputPanel,font = ('Helvetica', '12'))
  studentName.place(x=190,y=70,height=40,width = 160)
  testNameLabel = Label(inputPanel,text = 'Test Name:',font = ('Helvetica', '18'),bg ='#a6ff00',border=0)
  testNameLabel.place(x=20,y=120,height=40,width = 170)
  testName= ttk.Entry(inputPanel,font = ('Helvetica', '12'))
  testName.place(x=190,y=120,height=40,width = 160)
  listReturn=[]
  def takeData():
    nonlocal studentNameValue
    nonlocal testName
    nonlocal listReturn
    givenValue=str(studentName.get())
    testNameValue=str(testName.get())

    try:
      try:
        worksheet = resultData.worksheet(testNameValue)
        studentsWrittenList = worksheet.col_values(1)
```

```
        if givenValue in studentsWrittenList:
          messagebox.showinfo('Warning!','1)Looks like you have already taken this test.\n2)If not, enter a different
name.' )
        elif not givenValue.isalpha():
          messagebox.showinfo('Warning!','1)Name can contain no numbers.\n2)Name can contain no white spaces.')
        else:

          listReturn.append(testNameValue)
          listReturn.append(givenValue)

          mainroot.destroy()


      except:
        messagebox.showinfo('Warning!','1)Please enter the testName correctly.\n2)Test not found.\n3)Contact your
teacher' )

    except:
      messagebox.showinfo('Oops',"Something went wrong!\nPlease check your internet connectivity and try again.")

  def goBack():
    mainroot.destroy()
  backButton=Button(mainroot, text="BACK",command= goBack,font = ('Helvetica', '20'),bg
='#a6ff00',fg='black',border=0)
  backButton.place(height = 40, width = 80,x = 20,y = 20)
  doneButton=Button(inputPanel, text="DONE",command= takeData, height=2, width=13,font = ('Helvetica', '30'),padx =
200,pady = 200,border=0,bg ='#a6ff00')
  doneButton.place(height = 70, width = 180,x = 95,y = 290)


  mainroot.mainloop()
  return listReturn
def writeTest(getStudentName):

  if getStudentName !=[]:
    testName,studentName=getStudentName[0],getStudentName[1]
    errMsg='Server issue.\nTry again.'

    preText=[1,errMsg,errMsg,errMsg,errMsg,errMsg]
    root=Tk()
    root.title('WRITE TEST')
    root.state('zoomed')
    root.configure(background='#0c0c0c')
    root.overrideredirect(True)

    dataDict={}
    worksheet= dataFile.worksheet(testName)
    testDataRows = worksheet.get_all_values()
    answerKey=[]
    for row in testDataRows:
      answerKey.append(row.pop(6))
    testDetailRow = worksheet.row_values(len(testDataRows))
    for quesNum in range(1,int(testDetailRow[3])+1):
      dataDict.update({quesNum:testDataRows[quesNum-1]})

    timeRemList = []
    timeRemList.append(int(testDetailRow[4]))
    if int(testDetailRow[5])>0:
      timeRemList.append(int(testDetailRow[5])-1)
    else:
      timeRemList.append(int(testDetailRow[5]))
    timeRemList.append(5)
    studentResponse = {}
    noOfQuesValue = int(testDetailRow[3])
    def nextButtonFunc():

        studentResponse.update({currentQuesNum.get():var.get()})
        globals()['b%s' % currentQuesNum.get()].configure(bg = '#15ff0d')
        var.set(None)
        if timeUp:
          submitButtonFunc()
        else:
          if currentQuesNum.get() ==noOfQuesValue:
            if len(studentResponse)==noOfQuesValue:
              messagebox.showinfo('Info',"Looks like you've reached the end.\nClick SUBMIT to continue. ")
            else:
              for i in range(1,(noOfQuesValue+1)):
               if i not in studentResponse.keys():
                globals()['b%s' %i].invoke()
                break
          else:
            for i in range(currentQuesNum.get(),(noOfQuesValue+1)):
              if i not in studentResponse.keys():
                globals()['b%s' %i].invoke()
                break




    def submitButtonFunc():
```

```
      nonlocal root
      progress=Progressbar(root,orient=HORIZONTAL,length=100,mode='determinate')
      progress.place(x= 10,y = 430 ,width = 1050,height = 40)
      for i in range(1,30):
       progress['value']=i
       root.update_idletasks()
       time.sleep(0.05)
      progress['value']=35
      root.update_idletasks()
      time.sleep(1)
      progress['value']=40
      root.update_idletasks()
      time.sleep(1)
      progress['value']=50
      root.update_idletasks()
      time.sleep(1)
      nonlocal timeUp
      nonlocal timeRefresh
      nonlocal answerKey
      nonlocal noOfQuesValue
      # global resultData
      nonlocal testName
      nonlocal studentName
      marks=0
      for i in range(1,noOfQuesValue+1):
        if i in studentResponse.keys():
          if studentResponse[i]==answerKey[i-1]:
            marks+=1
      print(marks)
      try:
        worksheetResulData = resultData.worksheet(testName)
        values_list = worksheetResulData.col_values(1)
        vacantRowNo=len(values_list)+1
        worksheetResulData.update_cell(vacantRowNo,1,studentName )
        worksheetResulData.update_cell(vacantRowNo,2,marks)
        for i in range(50,101):
            progress['value']=i
            root.update_idletasks()
            time.sleep(0.05)
        if timeUp:
          msg='Succesfully Saved!\nYour time is up!\nClick OK to continue with TestApp.'
        else:
          msg =' Succesfully Saved!\nClick OK to continue with TestApp.'
        messagebox.showinfo('Info',msg)
        timeRefresh=False
        timeRemLabel.destroy()
        root.destroy()
        navigateTo('testApp')
      except:
        messagebox.showinfo('Oops',"Submission failed\nPlease check your internet connectivity and try again.")
        progress.destroy()



    sNo =1
    ques = Label(root , text = 'Q' +str(sNo) +')', font = ('Helvetica', '20'),bg ='#a6ff00')#str(preText[0])
    ques.place(x =10 , y = 30,height = 400, width = 60)
    questionBox=Label(root,bg ='lavender',font = ('Helvetica', '18'),text = dataDict[sNo][1],anchor=W, justify=LEFT)

    questionBox.place(x = 70,y = 30, height = 400, width = 990)

    var = StringVar()
    # var.set(preText[6])
    optA = Radiobutton(root , text = 'A', font = ('Helvetica', '25'),variable = var,value = 'A',bg = '#a6ff00'
,indicatoron = 0,selectcolor = 'green')
    optA.place(x =10 , y = 470,height = 100, width = 60)
    optionA = Label(root,bg ='lavender',font = ('Helvetica', '18'),text = dataDict[sNo][2])

    optionA.place(x=70,y = 470,height =100,width = 450)

    optB = Radiobutton(root , text = 'B', font = ('Helvetica', '25'),variable = var,value = 'B',bg =
'#a6ff00',indicatoron = 0,selectcolor = 'green')
    optB.place(x =10 , y = 590,height = 100, width = 60)
    optionB = Label(root,bg ='lavender',font = ('Helvetica', '18'),text = dataDict[sNo][3])

    optionB.place(x=70,y = 590,height =100,width = 450)
    optC = Radiobutton(root , text = 'C', font = ('Helvetica', '25'),variable = var,value = 'C',bg =
'#a6ff00',indicatoron = 0,selectcolor = 'green')
    optC.place(x =550 , y = 470,height = 100, width = 60)
    optionC = Label(root,bg ='lavender',font = ('Helvetica', '18'),text = dataDict[sNo][4])

    optionC.place(x=610,y = 470,height =100,width = 450)

    optD =Radiobutton(root , text = 'D', font = ('Helvetica', '25'),variable = var,value = 'D',bg =
'#a6ff00',indicatoron = 0,selectcolor = 'green')
    optD.place(x =550 , y = 590,height = 100, width = 60)
    optionD = Label(root,bg ='lavender',font = ('Helvetica', '18'),text = dataDict[sNo][5])

    optionD.place(x=610,y = 590,height =100,width = 450)

    rightPanel = Canvas(root, bg="#3a3b33", height=655, width=270)
```

```
    rightPanel.place(x =1080,y  =30)

    rightPanel.create_line(0, 20, 270, 20, fill="black")
    rightPanel.create_line(0, 22, 270, 22, fill="black")
    rightPanel.create_line(0, 100, 270, 100, fill="black")
    rightPanel.create_line(0, 102, 270, 102, fill="black")
    rightPanel.create_line(0, 440, 270, 440, fill="black")
    rightPanel.create_line(0, 442, 270, 442, fill="black")
    timeRem= Label(root , text = 'Time remaining:', font = ('Helvetica', '10'),bg ='#a6ff00')
    timeRem.place(x =1080 , y = 30,height = 20, width = 275)

    quickAcc= Label(root , text = 'Quick Access:', font = ('Helvetica', '10'),bg ='#a6ff00')
    quickAcc.place(x =1080 , y = 110,height = 20, width = 275)
    userOpt= Label(root , text = 'Options:', font = ('Helvetica', '10'),bg ='#a6ff00')
    userOpt.place(x =1080 , y = 450,height = 20, width = 275)

    nextButton=Button(root,padx = 200,pady = 200,border=0,font = ('Helvetica', '25'),bg ='#a6ff00', height=2,
width=13, text="SAVE &\n  NEXT",command= nextButtonFunc )
    nextButton.place(height = 90, width = 200,x = 1115,y = 480)

    submitButton=Button(root,padx = 200,pady = 200,border=0,font = ('Helvetica', '30'),bg ='#a6ff00', height=2,
width=13, text="SUBMIT",command= submitButtonFunc )
    submitButton.place(height = 90, width = 200,x = 1115,y = 585)



    timeText = str(timeRemList[0])+':'+str(timeRemList[1])+':'+str(timeRemList[2])
    timeRemLabel = Button(root,padx = 200,pady = 200,borderwidth= 10,font = ('Helvetica', '20'),bg ='#3a3b33',fg =
'white', height=2, width=13, text=timeText,)
    timeRemLabel.place(height = 48, width = 220,x = 1107,y = 57)


    timeUp= False
    timeRefresh = True
    def timeRemClock():
       nonlocal timeRefresh
       nonlocal timeRemList
       nonlocal timeText
       nonlocal timeUp
       timeRemList[2]-=1
       timeText = str(timeRemList[0])+':'+str(timeRemList[1])+':'+str(timeRemList[2])
       timeRemLabel.config(text=timeText)

       if timeRemList == [0,0,0]:
          timeUp = True
          timeRefresh=False
       if timeRemList[2]==0 and timeRemList[1]!=0:
          timeRemList[2]=59
          timeRemList[1]-=1

       if timeRemList[1]==0 and timeRemList[2]==0 :
          timeRemList[2]=59
          timeRemList[0]-=1
          timeRemList[1]=59

       if timeRefresh:
          timeRemLabel.after(1000, timeRemClock)


    timeRemClock()






    quickAccessPanel = Frame(root,bg='#3a3b33')
    quickAccessPanel.place(height = 270,width = 268,x =1082,y = 135)
    quesQueue = []
    quesNum = IntVar()
    num = 1
    def quickAccButList():
      sNo = currentQuesNum.get()
      ques.configure(text = 'Q' +str(sNo) +')')
      if timeUp:
        submitButtonFunc()
      else:
        try:
          globals()['b%s' % currentQuesNum.get()].configure(bg = 'red')

          questionBox.configure(text = dataDict[sNo][1])
          optionA.configure(text =dataDict[sNo][2])
          optionB.configure(text =dataDict[sNo][3])
          optionC.configure(text =dataDict[sNo][4])
          optionD.configure(text =dataDict[sNo][5])
          try:
            var.set(studentResponse[sNo])
          except:
            pass
        except:
          globals()['b%s' % currentQuesNum.get()].configure(bg = 'red')
```

```
          questionBox.configure(text = preText[1])
          optionA.configure(text =preText[2])
          optionB.configure(text =preText[3])
          optionC.configure(text =preText[4])
          optionD.configure(text =preText[5])
        # var.set(preText[6])




    currentQuesNum = IntVar()
    currentQuesNum.set(1)
    for r in range(int(noOfQuesValue/5)):
     for c in range(5):
        if num<10:
          globals()['b%s' % num] = Radiobutton(quickAccessPanel , text = '0{}'.format(str(num)), font = ('Helvetica',
'15'),bg = 'red',indicatoron = 0,selectcolor = '#a6ff00',command = lambda:quickAccButList(),variable=currentQuesNum,
value = num)
          globals()['b%s' % num].grid(row=r,column=c,padx=11,pady = 5)
        else:
          globals()['b%s' % num] = Radiobutton(quickAccessPanel , text = '{}'.format(str(num)), font = ('Helvetica',
'15'),bg = 'red',indicatoron = 0,selectcolor = '#a6ff00',command =
lambda:quickAccButList(),variable=currentQuesNum,value = num)
          globals()['b%s' % num].grid(row=r,column=c,padx=11,pady = 5)
        num+=1

    rightPanel.create_line(0, 379, 270, 379, fill="black")
    rightPanel.create_line(0, 381, 270, 381, fill="black")
    rightPanel.create_polygon(4, 385, 4,415,34,415,34,385,fill="#15ff0d")
    Label(rightPanel , text = 'Saved', font = ('Helvetica', '10'),bg ='#3a3b33',fg='white').place(x =50 , y = 389)

    rightPanel.create_polygon(130, 385, 130,415,160,415,160,385,fill="red")
    Label(rightPanel , text = 'Not Saved', font = ('Helvetica', '10'),bg ='#3a3b33',fg='white').place(x =170 , y =
389)
    root.mainloop()

    navigateTo('testApp')
  else:
    navigateTo('testApp')
#_____________________________________________________________________________________________________________-
def blitText(text,fontSize,pos,color,fontname_with_ext):
    Font = pygame.font.Font(fontname_with_ext, fontSize)
    gameDisplay.blit(Font.render(text, True, color),pos)

def showText(text, sleep, color, fontSize, center):
    Font = pygame.font.Font('AllerDisplay.ttf', fontSize)
    textSurf = Font.render(text, True, color)
    textRect = textSurf.get_rect()
    textRect.center = center  # (663,495)
    gameDisplay.blit(textSurf, textRect)
    pygame.display.update()
    time.sleep(sleep)
def showGrids(scale):
    for i in range (1,1366,int(scale)):
        pygame.draw.line(gameDisplay, white, (i, 0), (i, 768), 1)
        pygame.draw.line(gameDisplay, white, (0,i), (1366,i), 1)
def popUp(message):
    ok = False
    okBcolor =(5,5,5)
    while not ok:
        pygame.draw.rect(gameDisplay,(5,5,5),[360, 200, 650, 400])
        pygame.draw.rect(gameDisplay,(161,255,0),[365, 205, 640, 390],2)
        pygame.draw.rect(gameDisplay,okBcolor,[585, 545, 200, 40])
        pygame.draw.rect(gameDisplay,(161,255,0),[585, 545, 200, 40],2)
        blitText(message.center(90,' '),20,(400,250),(161,255,0),'AllerDisplay.ttf')
        blitText('OK',45,(658,546),(161,255,0),'AllerDisplay.ttf')
        pygame.display.update()
        for event in pygame.event.get():
          if event.type == pygame.MOUSEMOTION:  # for button color change
              xm, ym = event.pos
              if (585 < xm < 785) and (545 < ym < 590):
                  okBcolor = (227, 255, 115)
              else:
                  okBcolor = (5,5,5)
          elif event.type == pygame.MOUSEBUTTONUP:  # for button clicking
              pos = pygame.mouse.get_pos()
              xClick, yClick = pos
              if (585 < xClick < 785) and (545< yClick < 590):
                  ok = True
def confirm(message):
    ok = False
    yesBcolor =(0, 33, 74)
    noBcolor =(0, 33, 74)
    response = 'NO'
    while not ok:
        pygame.draw.rect(gameDisplay,(5,5,5),[360, 200, 650, 400])
        pygame.draw.rect(gameDisplay,(161, 255, 0),[365, 205, 640, 390],2)
        pygame.draw.rect(gameDisplay,yesBcolor,[460, 545, 200, 40])
        pygame.draw.rect(gameDisplay,noBcolor,[700, 545, 200, 40])
        pygame.draw.rect(gameDisplay,(161,255,0),[460, 545, 200, 40],2)
```

```
        pygame.draw.rect(gameDisplay,(161,255,0),[700, 545, 200, 40],2)

        blitText(message.center(90,' '),20,(400,250),(161,255,0),'AllerDisplay.ttf')
        blitText('     YES',45,(460,546),(161,255,0),'AllerDisplay.ttf')
        blitText('     NO',45,(700,546),(161,255,0),'AllerDisplay.ttf')
        pygame.display.update()
        for event in pygame.event.get():
          if event.type == pygame.MOUSEMOTION:  # for button color change
              xm, ym = event.pos
              if (460 < xm < 660) and (545 < ym < 585):
                  yesBcolor = (227, 255, 115)
              elif (700 < xm < 900) and (545 < ym < 585):
                  noBcolor = (227, 255, 115)
              else:
                  yesBcolor,noBcolor = (5,5,5),(5,5,5)
          elif event.type == pygame.MOUSEBUTTONUP:  # for button clicking
              pos = pygame.mouse.get_pos()
              xClick, yClick = pos
              if (460 < xClick < 660) and (545 < yClick < 585):
                  response = 'YES'
                  ok = True
              if (700 < xClick < 900) and (545 < yClick < 585):
                  ok = True
    if response == 'YES':
        return True
    else:
        return False
def passChecker():
  minimizeAll()
  mainroot=Tk()
  mainroot.title('User Input')
  mainroot.state('zoomed')
  mainroot.configure(background='#0c0c0c')
  mainroot.overrideredirect(True)
  inputPanel = Frame(mainroot,bg = '#3a3b33')
  inputPanel.place(height = 370,width = 370,x =500,y = 185)
  passLabel = Label(inputPanel,text = 'Enter the Password:',font = ('Helvetica', '15'),bg ='#a6ff00')
  passLabel.place(x=10,y=70,height=40,width = 190)
  passwordEntry= ttk.Entry(inputPanel,font = ('Helvetica', '12'),show='*')
  passwordEntry.place(x=200,y=70,height=40,width = 160)
  correctPassword=False
  def takeData():
    nonlocal passwordEntry
    nonlocal correctPassword
    passwordEntryValue=str(passwordEntry.get())

    try:
      if passwordEntryValue == 'key':
        correctPassword=True
        mainroot.destroy()
      else:
        messagebox.showinfo('Oops!','Incorrect pasword!')
        mainroot.destroy()
    except:
      messagebox.showinfo('Warning!','SOMETHING WENT WRONG, TRY AGAIN' )

  def goBack():
    mainroot.destroy()
  backButton=Button(mainroot, text="BACK",command= goBack,font = ('Helvetica', '20'),bg
='#a6ff00',fg='black',border=0)
  backButton.place(height = 40, width = 80,x = 20,y = 20)

  doneButton=Button(inputPanel, text="LOGIN",command= takeData, height=2, width=13,font = ('Helvetica', '30'),padx =
200,pady = 200,bg ='#a6ff00',fg='black',border=0)
  doneButton.place(height = 70, width = 180,x = 90,y = 290)


  mainroot.mainloop()

  navigateTo('testApp')
  return correctPassword
def getTestNameForResult():
  minimizeAll()
  mainroot=Tk()
  mainroot.title('User Input')
  mainroot.state('zoomed')
  mainroot.configure(background='#0c0c0c')
  mainroot.overrideredirect(True)
  inputPanel = Frame(mainroot,bg = '#3a3b33')
  inputPanel.place(height = 370,width = 370,x =500,y = 185)
  testNameValue=''

  testNameLabel = Label(inputPanel,text = 'Test Name:',font = ('Helvetica', '18'),bg ='#a6ff00')
  testNameLabel.place(x=20,y=70,height=40,width = 170)
  testName= ttk.Entry(inputPanel,font = ('Helvetica', '12'))
  testName.place(x=190,y=70,height=40,width = 160)

  def takeData():
    nonlocal testName
    listReturn=[]
    testNameValue=str(testName.get())
```

```
    try:
      pathName=filedialog.askdirectory()

      worksheet = resultData.worksheet(testNameValue)
      datasheet = dataFile.worksheet(testNameValue)


      dat = datasheet.get_all_values()[-1]
      resultList=worksheet.get_all_values()
      totMarks=int(dat[2])*int(dat[3])

      listReturn.append(resultList)
      listReturn.append(totMarks)
      l = listReturn
      x = []
      y = []
      for i in l[0]:
        x.append(i[0])
        y.append(int(i[1]))
      dicRan={}
      lent=len(y)
      for i in range(lent):
        dicRan.update({y[i]:x[i]})
      y.sort()
      yFin= y
      xFin=[]
      for i in yFin:
        xFin.append(dicRan[i])
      p.bar(xFin,yFin)

      p.xticks(rotation='vertical')
      p.yticks(np.arange(0,l[1]+1))
      p.savefig(fname='{0}/{1}.png'.format(pathName,testNameValue))
      p.close('all')
      messagebox.showinfo('DONE!','Results are successfully saved in the chosen directory.')
      mainroot.destroy()


    except:
      messagebox.showinfo('Warning!','1)Please enter the testName correctly.\n2)Test not found.\n3)Contact your
teacher' )
      mainroot.destroy()
  def goBack():

    mainroot.destroy()

  backButton=Button(mainroot, text="BACK",command= goBack,font = ('Helvetica', '20'),bg
='#a6ff00',fg='black',border=0)
  backButton.place(height = 40, width = 80,x = 20,y = 20)

  doneButton=Button(inputPanel, text="DONE",command= takeData, height=2, width=13,font = ('Helvetica', '30'),padx =
200,pady = 200,bg ='#a6ff00')
  doneButton.place(height = 70, width = 180,x = 90,y = 290)


  mainroot.mainloop()
  # teacherPage()
def manageDb():
  msgDb='''
  ------------------------------------BE CAREFULL-----------------------------------------
  1)sign in with the id: "testappforkvhvf3@gmail.com"
  2)Make sure you delete the worksheet of your test from all the three spreadsheets.
  3)Open availableTest spreadsheet to view the list of tests in the database.
  4)Don't modify the name of any spreadsheets, the application may stop working.
  '''
  minimizeAll()
  mainroot=Tk()
  mainroot.title('User Input')
  mainroot.state('zoomed')
  mainroot.configure(background='#0c0c0c')
  testNameLabel = Label(mainroot,text = msgDb,font = ('Helvetica', '18'),bg ='white',anchor=W, justify=LEFT)
  testNameLabel.place(height = 370,width = 950,x =200,y = 185)
  def openDb():
    try:

webbrowser.open_new('https://accounts.google.com/signin/v2/identifier?service=wise&passive=1209600&continue=https%3A%2
F%2Fdocs.google.com%2Fspreadsheets%2Fu%2F0%2F&followup=https%3A%2F%2Fdocs.google.com%2Fspreadsheets%2Fu%2F0%2F&ltmpl=s
heets&flowName=GlifWebSignIn&flowEntry=ServiceLogin')
      mainroot.destroy()
    except:
      messagebox.showinfo('oops!','Something went wrong try again')
      mainroot.destroy()
  def goBack():
    mainroot.destroy()
  backButton=Button(mainroot, text="BACK",command= goBack,font = ('Helvetica', '20'),bg
='#a6ff00',fg='black',border=0)
  backButton.place(height = 40, width = 80,x = 20,y = 20)
  doneButton=Button(mainroot, text="OK",command= openDb, height=2, width=13,font = ('Helvetica', '30'),padx = 200,pady
= 200,bg ='#a6ff00')
  doneButton.place(height = 70, width = 180,x = 560,y = 580)
```

```
  mainroot.mainloop()
```

studentPage.py

```
from studentPage import *
from teacherPage import *
def main():
  global gameDisplay
  black = (0, 0, 0)
  white = (255, 255, 255)
  purple = (255, 10, 246)
  skyblue = (0, 215, 255)
  lightPurple = (245, 150, 240)
  green = (0, 200, 30)
  green1 = (0, 200, 100)
  homeButtonL = (200, 247, 12)
  homeButtonR = (200, 247, 12)
  red = (200, 0, 0)
  yellow = (161, 255, 0)#(0, 240, 174)
  exitColor, infoColor ,backColor,availableTestsColor,viewResultsColor = yellow, yellow, yellow,yellow,yellow
  blue = (2, 59, 181)
  run = True


  while run:

      gameDisplay.fill((12, 12, 12))
      Font=pygame.font.Font('AllerDisplay.ttf', 120)
      textRectL = pygame.draw.rect(gameDisplay, homeButtonL, [135, 325, 525, 105])  # [150, 520, 516, 170])
      textRectR = pygame.draw.rect(gameDisplay, homeButtonR, [710, 325, 520, 105])  # [675, 520, 513, 170])
      textSurfL = Font.render(' STUDENT', True, black)
      textSurfR = Font.render(' TEACHER', True, black)
      pygame.draw.line(gameDisplay, white, (683, 225), (683, 544), 2)
      gameDisplay.blit(textSurfL, textRectL)
      gameDisplay.blit(textSurfR, textRectR)
      pygame.draw.circle(gameDisplay, exitColor, (50, 50), 25)

      gameDisplay.blit(exitIcon, (32, 32))
      pygame.draw.circle(gameDisplay, infoColor, (130, 50), 25)

      gameDisplay.blit(infoIcon, (114, 32))
      pygame.display.update()
      # -----------------------------------------HomeSrnEvent-----------------------------------------------------------
--------
      for event in pygame.event.get():
          if event.type == pygame.MOUSEMOTION:  # for button color change
              xm, ym = event.pos
              if (135 < xm < 135 + 525) and (325 < ym < 325 + 105):
                  homeButtonL = blue
              elif (710 < xm < 710 + 520) and (325 < ym < 325 + 105):
                  homeButtonR = blue
              elif (((xm - 50) ** 2) + ((ym - 50) ** 2) - (25 ** 2)) < 0:
                  exitColor = red
              elif (((xm - 130) ** 2) + ((ym - 50) ** 2) - (25 ** 2)) < 0:
                  infoColor = blue
              else:
                  homeButtonL, homeButtonR, exitColor, infoColor = yellow, yellow, yellow, yellow
          if event.type == pygame.MOUSEBUTTONUP:  # for button clicking
              pos = pygame.mouse.get_pos()
              xClick, yClick = pos
              if (135 < xClick < 135 + 525) and (325 < yClick < 325 + 105):
                  studentPage()
              if (710 < xClick < 710 + 520) and (325 < yClick < 325 + 105):
                  if passChecker():
                    teacherPage()
                  else:
                    pass
              if (((xClick - 50) ** 2) + ((yClick - 50) ** 2) - (25 ** 2)) < 0:
                  if confirm('Are you sure want to quit?'):
                    run = False
              if (((xClick - 130) ** 2) + ((yClick - 50) ** 2) - (25 ** 2)) < 0:
                  popUp('-With love, from Vignesh Balaji S ,Class 12 (2019-20)')#'info button'
          if event.type == pygame.KEYDOWN:  # universal quiting option escape
              if event.key == pygame.K_ESCAPE:
                  run = False
  pygame.quit()
  quit()
from functions import *
def studentPage():
```

```
  black = (0, 0, 0)
  white = (255, 255, 255)
  purple = (255, 10, 246)
  skyblue = (0, 215, 255)
  lightPurple = (245, 150, 240)
  green = (0, 200, 30)
  green1 = (0, 200, 100)
  homeButtonL = (200, 247, 12)
  homeButtonR = (200, 247, 12)
  red = (235, 8, 0)
  yellow = (166,255,0)
  exitColor, infoColor ,backColor,availableTestsColor,viewResultsColor = yellow, yellow, yellow,yellow,yellow
  blue = (2, 59, 181)
# -----------------------------loadingImages-------------------------------------------------------------------
  backIcon = pygame.transform.scale(pygame.image.load('backIcon.png'), (35,35))
  exitIcon = pygame.transform.scale(pygame.image.load('exitIcon.png'), (35, 35))
  infoIcon = pygame.transform.scale(pygame.image.load('infoIcon.jpg'), (35, 35))
  vigbalityBG = pygame.transform.scale(pygame.image.load('vigbalityBG.jpg'), (1366, 768))
  testappBG = pygame.transform.scale(pygame.image.load('testappBG.jpg'), (1366, 768))
  run = True
  while run:

      gameDisplay.fill((12,12,12))
      pygame.draw.circle(gameDisplay,availableTestsColor, (680, 350), 125)
      pygame.draw.circle(gameDisplay, black, (680, 350), 120,3)

      blitText('WRITE',50,(600,305),black,'AllerDisplay.ttf')
      blitText('TEST',50,(643,350),black,'AllerDisplay.ttf')

      pygame.draw.circle(gameDisplay, backColor, (50, 50), 25)

      gameDisplay.blit(backIcon, (32, 32))

      pygame.display.update()

      for event in pygame.event.get():
          if event.type == pygame.MOUSEMOTION:  # for button color change
              xm, ym = event.pos
              if (((xm - 50) ** 2) + ((ym - 50) ** 2) - (25 ** 2)) < 0:
                  backColor = blue
              elif (((xm - 680) ** 2) + ((ym - 350) ** 2) - (125 ** 2)) < 0:
                  availableTestsColor = blue

              else:
                  availableTestsColor, viewResultsColor , backColor = yellow, yellow, yellow
          if event.type == pygame.MOUSEBUTTONUP:  # for button clicking
              pos = pygame.mouse.get_pos()
              xClick, yClick = pos
              if (((xm - 50) ** 2) + ((ym - 50) ** 2) - (25 ** 2)) < 0:
                  run = False
              if (((xm - 680) ** 2) + ((ym - 350) ** 2) - (125 ** 2)) < 0:
                  writeTest(getStudentName())
          if event.type == pygame.KEYDOWN:  # universal quiting option escape
              if event.key == pygame.K_ESCAPE:
                  run = False
                  pygame.quit()
                  quit()
```

teacherPage.py

```
from functions import *
def teacherPage():

  black = (0, 0, 0)
  white = (255, 255, 255)
  purple = (255, 10, 246)
  skyblue = (0, 215, 255)
  lightPurple = (245, 150, 240)
  green = (0, 200, 30)
  green1 = (0, 200, 100)
  homeButtonL = (200, 247, 12)
  homeButtonR = (200, 247, 12)
  red = (235, 8, 0)
  yellow = (166,255,0)
  exitColor, infoColor ,backColor,createNewTestColor,viewResultsColor,manageDatabaseColor = yellow, yellow,
yellow,yellow,yellow,yellow
  blue = (2, 59, 181)
# -----------------------------loadingImages-------------------------------------------------------------------

  backIcon = pygame.transform.scale(pygame.image.load('backIcon.png'), (35,35))
  exitIcon = pygame.transform.scale(pygame.image.load('exitIcon.png'), (35, 35))
  infoIcon = pygame.transform.scale(pygame.image.load('infoIcon.jpg'), (35, 35))
  vigbalityBG = pygame.transform.scale(pygame.image.load('vigbalityBG.jpg'), (1366, 768))
  testappBG = pygame.transform.scale(pygame.image.load('testappBG.jpg'), (1366, 768))
```

```
run = True
while run:

    gameDisplay.fill((12,12,12))
    pygame.draw.circle(gameDisplay,createNewTestColor, (355, 350), 125)
    pygame.draw.circle(gameDisplay, black, (355, 350), 120,3)
    pygame.draw.circle(gameDisplay, viewResultsColor, (690, 350), 125)
    pygame.draw.circle(gameDisplay, black, (690, 350), 120,3)
    pygame.draw.circle(gameDisplay,manageDatabaseColor, (1025, 350), 125)
    pygame.draw.circle(gameDisplay, black, (1025, 350), 120,3)
    pygame.draw.line(gameDisplay, white, (523, 180), (523, 544), 2)
    pygame.draw.line(gameDisplay, white, (855, 180), (855, 544), 2)

    blitText('Create',50,(280,305),black,'AllerDisplay.ttf')
    blitText('New Test',50,(253,350),black,'AllerDisplay.ttf')
    blitText('Generate',50,(590,305),black,'AllerDisplay.ttf')
    blitText('Results',50,(600,350),black,'AllerDisplay.ttf')
    blitText('Manage',50,(940,305),black,'AllerDisplay.ttf')
    blitText('DataBase',50,(920,350),black,'AllerDisplay.ttf')

    pygame.draw.circle(gameDisplay, backColor, (50, 50), 25)

    gameDisplay.blit(backIcon, (32, 32))

    pygame.display.update()

    for event in pygame.event.get():
        if event.type == pygame.MOUSEMOTION:  # for button color change
            xm, ym = event.pos
            if (((xm - 50) ** 2) + ((ym - 50) ** 2) - (25 ** 2)) < 0:
                backColor = blue
            elif (((xm - 355) ** 2) + ((ym - 350) ** 2) - (125 ** 2)) < 0:
                createNewTestColor = blue
            elif (((xm - 690) ** 2) + ((ym - 350) ** 2) - (125 ** 2)) < 0:
                viewResultsColor = blue
                'viewResults button'
            elif (((xm - 1025)** 2) + ((ym - 350) ** 2) - (125 ** 2)) < 0:
                manageDatabaseColor = blue
            else:
                createNewTestColor, viewResultsColor , backColor,manageDatabaseColor = yellow, yellow, yellow,yellow
        if event.type == pygame.MOUSEBUTTONUP:  # for button clicking
            pos = pygame.mouse.get_pos()
            xClick, yClick = pos
            if (((xm - 50) ** 2) + ((ym - 50) ** 2) - (25 ** 2)) < 0:
                run = False
            if (((xm - 355) ** 2) + ((ym - 350) ** 2) - (125 ** 2)) < 0:
                try:
                  sas=preDataForCreatingTest()
                  if sas !=[]:
                    createTest(sas)
                except:
                  pass
            if (((xm - 690)** 2) + ((ym - 350) ** 2) - (125 ** 2)) < 0:
                try:
                  getTestNameForResult()
                  navigateTo('testApp')

                except:
                  pass
            if (((xm - 1025)** 2) + ((ym - 350) ** 2) - (125 ** 2)) < 0:
                manageDb()
                navigateTo('testApp')
        if event.type == pygame.KEYDOWN:  # universal quiting option escape
            if event.key == pygame.K_ESCAPE:
                run = False
                pygame.quit()
                quit()
```

testApp.py

```
import pygame
import gspread
from oauth2client.service_account import ServiceAccountCredentials
from functions import *
pygame.init()
gameDisplay = pygame.display.set_mode((1366, 768),pygame.FULLSCREEN)
pygame.display.set_caption('testApp')
clock = pygame.time.Clock()
try:

    gameDisplay.blit(kvhvfBgImg, (0, 0))
    pygame.display.update()
    time.sleep(7)
    gameDisplay.blit(testappBG, (0, 0))
```

```
    pygame.display.update()
    speed = 0.01
    for i in range(1, 81):
        if i == 20:
            speed = 0.05
        if i == 60:
            speed = 0.15
        showText('Loading  ' + str(i) + '%', speed, white, 30, (670, 700))
        gameDisplay.blit(testappBG, (0, 0))
        pygame.display.update()

    scope =
["https://spreadsheets.google.com/feeds",'https://www.googleapis.com/auth/spreadsheets',"https://www.googleapis.com/au
th/drive.file","https://www.googleapis.com/auth/drive"]
    creds = ServiceAccountCredentials.from_json_keyfile_name("testapp-6d21738b9b56.json", scope)
    client = gspread.authorize(creds)
    dataFile = client.open("dataFile")
    resultData = client.open("resultData")
    for i in range(81, 101):
        speed = 0.05
        showText('Loading  ' + str(i) + '%', speed, white, 30, (670, 700))
        gameDisplay.blit(testappBG, (0, 0))
        pygame.display.update()
    main()
except:
    popUp('Please check your internet connectivity \nand try again.')
    pygame.quit()
    quit()
```

dbTest.py

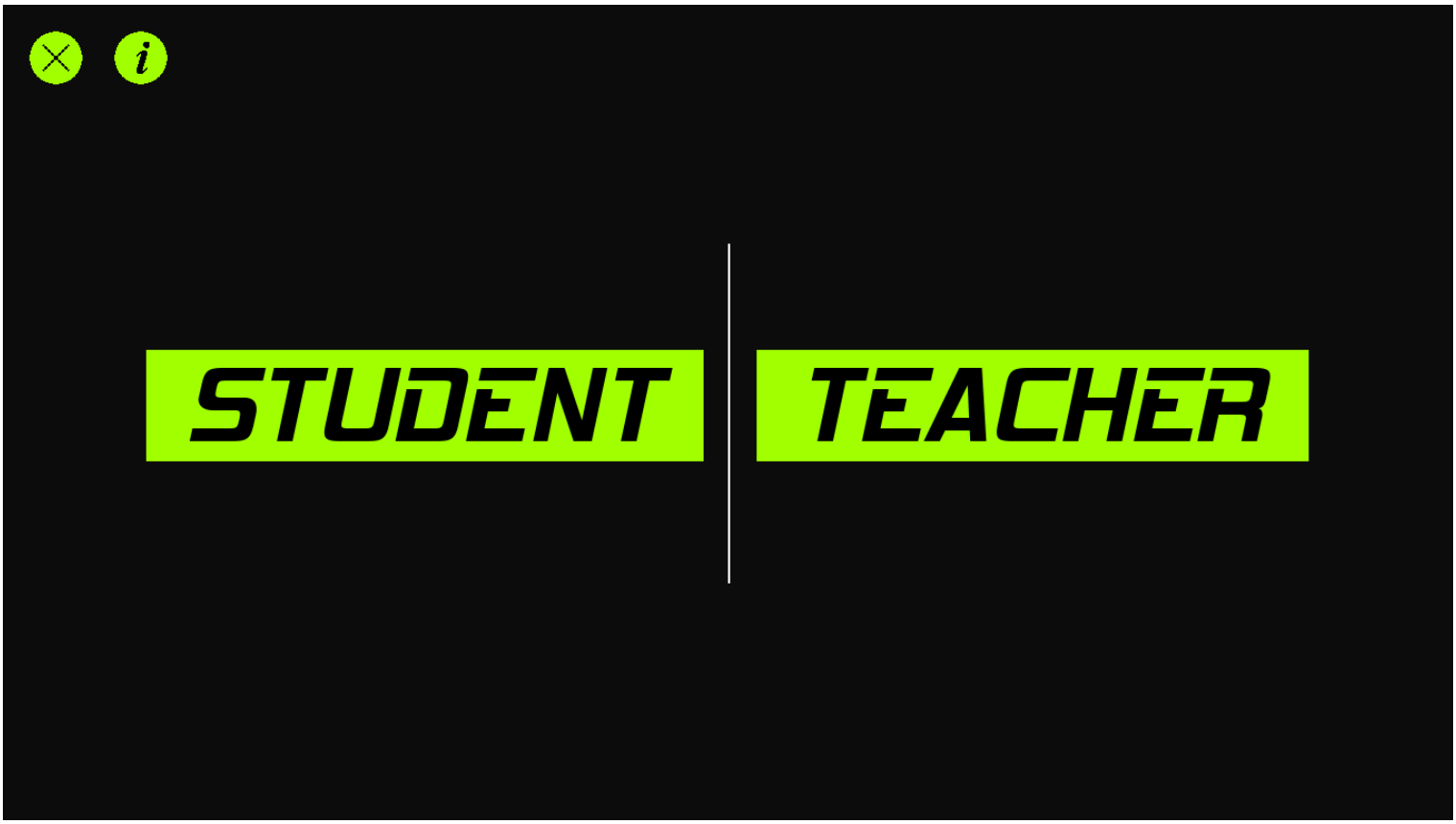
```
import gspread
from oauth2client.service_account import ServiceAccountCredentials
try:
        scope =
["https://spreadsheets.google.com/feeds",'https://www.googleapis.com/auth/spreadsheets',"https://www.googleapis.com/au
th/drive.file","https://www.googleapis.com/auth/drive"]
        creds = ServiceAccountCredentials.from_json_keyfile_name("testapp-6d21738b9b56.json", scope)
        client = gspread.authorize(creds)
        dataFile = client.open("dataFile")
        resultData = client.open("resultData")
        availableTests=client.open('availableTests')
except:
    pass
def vacantRow(worksheet):
        values_list = worksheet.col_values(1)
        return len(values_list)+1
```

----------CodeEndsHere----------

OUTPUT SCREENS

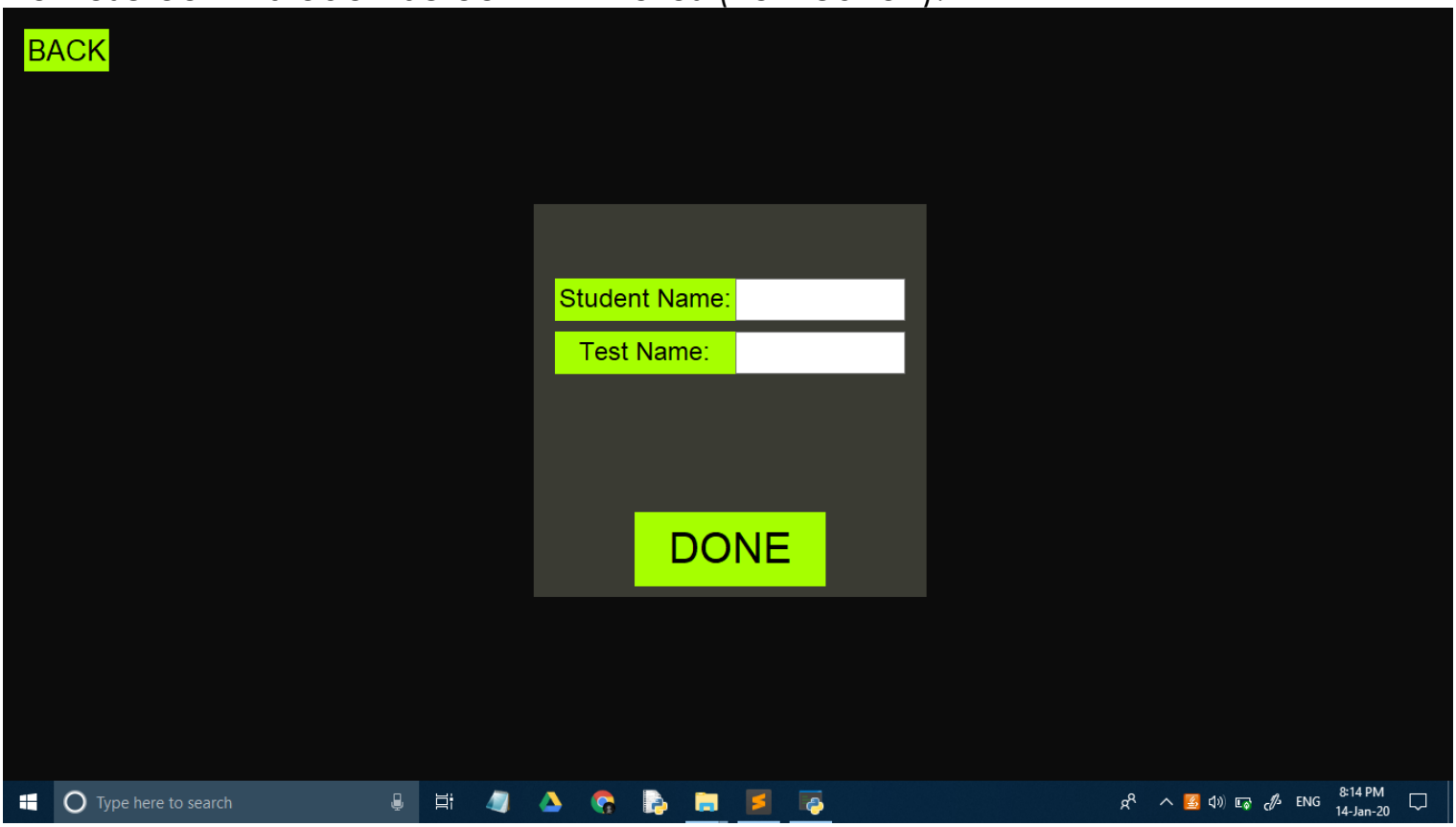
Loading Screen:

Home screen:

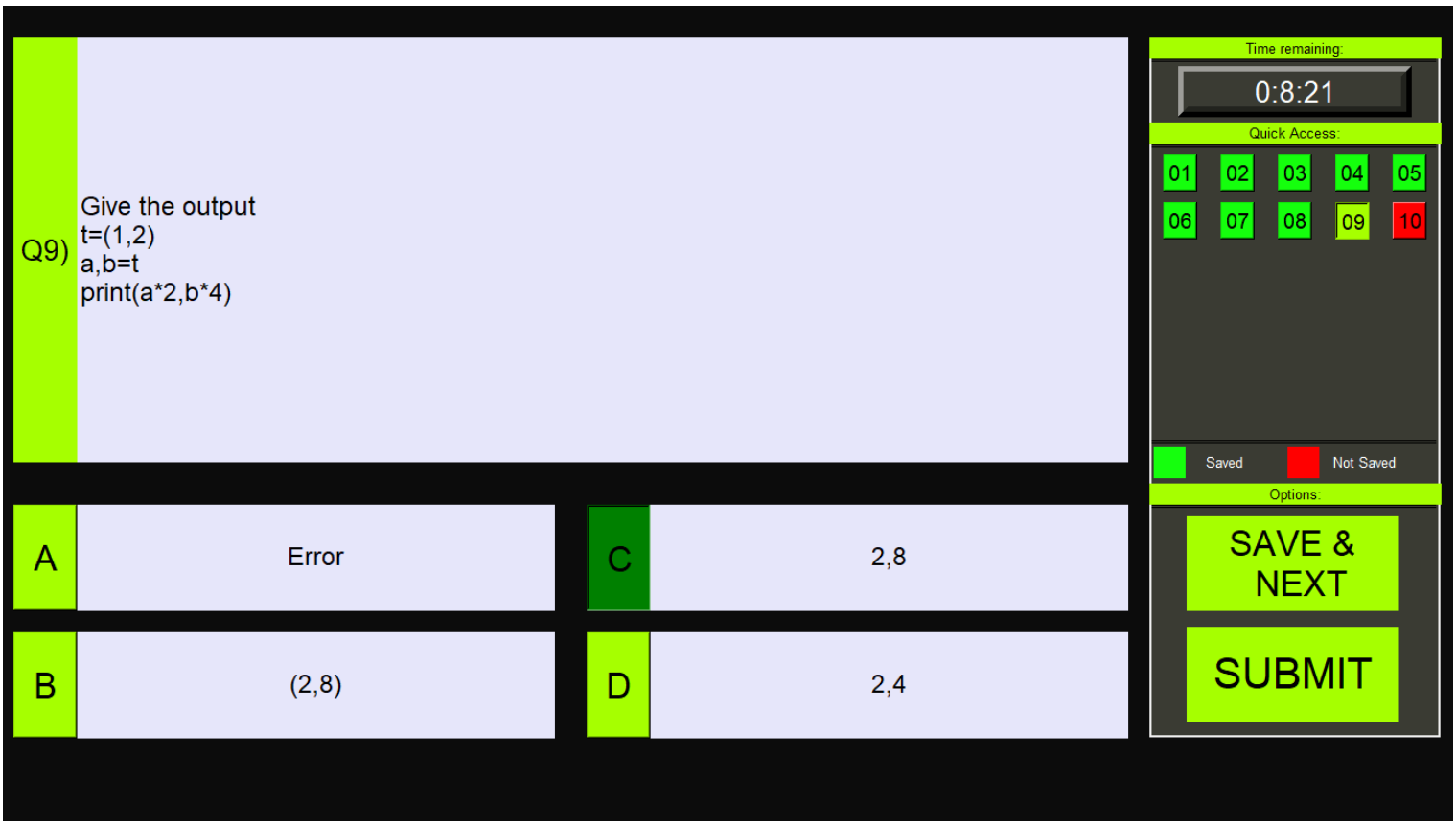


Homescreen>>StudentScreen:

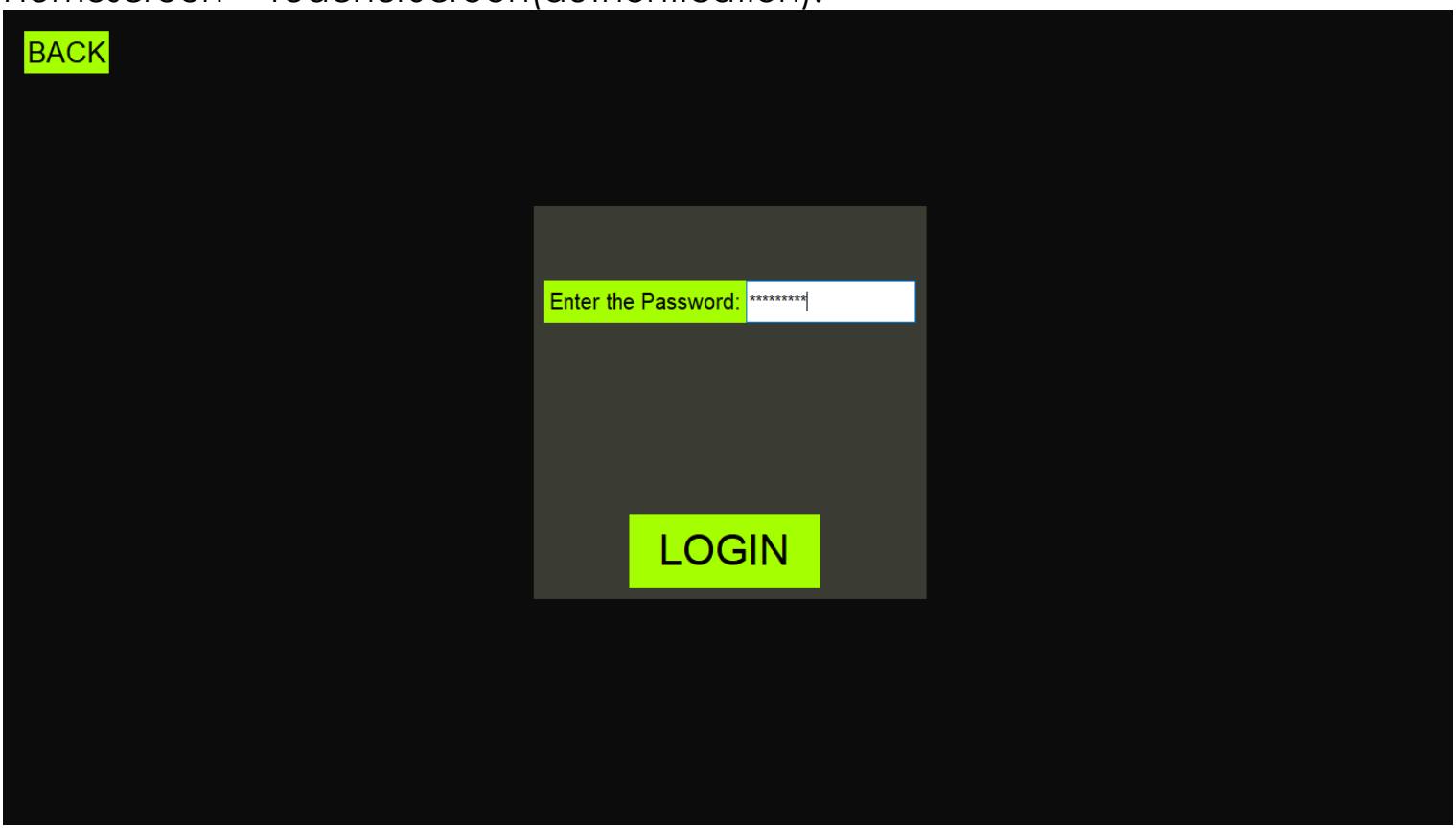
Homescreen>>StudentScreen>>WriteTest(verification):



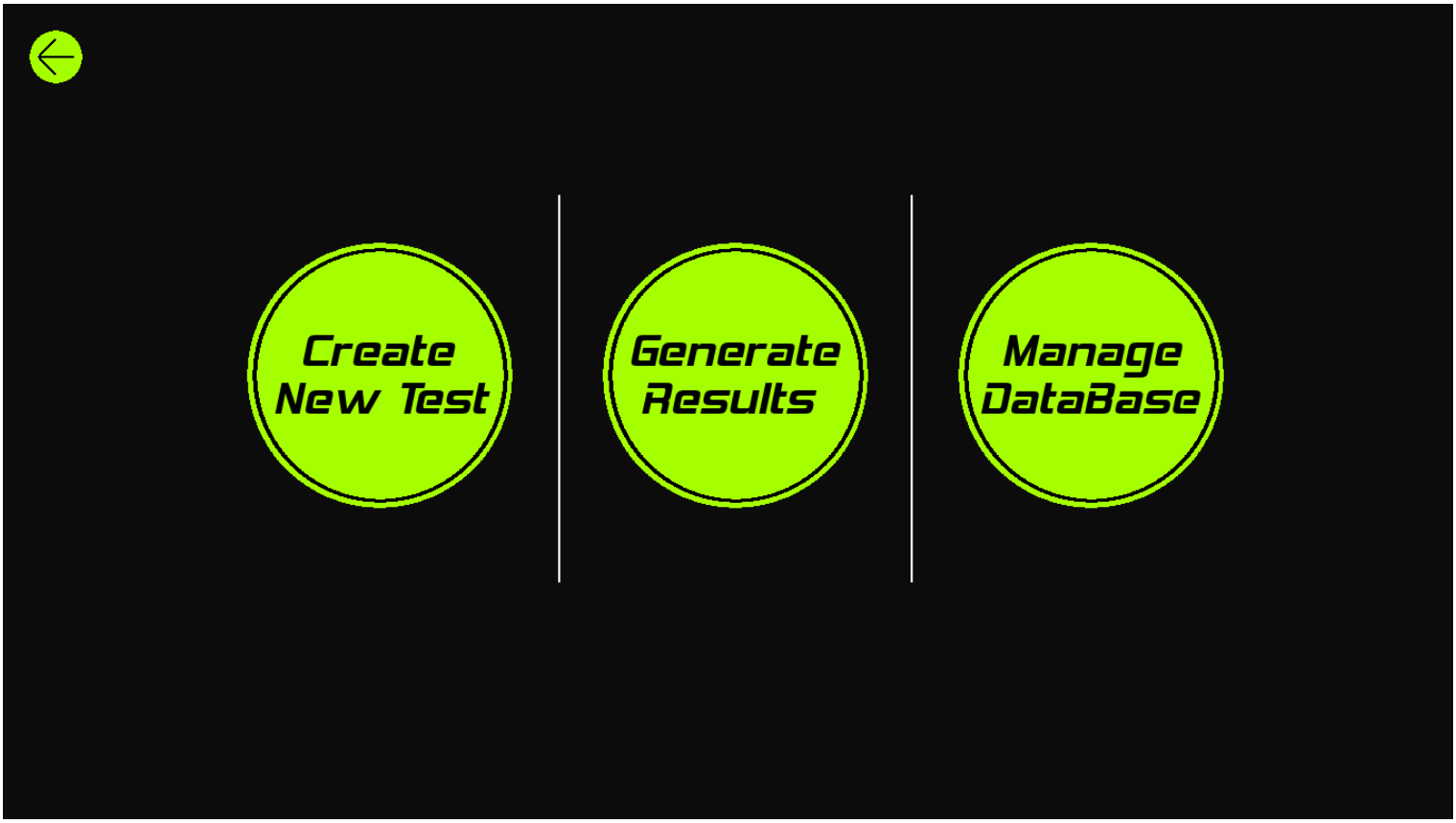
Homescreen>>StudentScreen>>WriteTest:



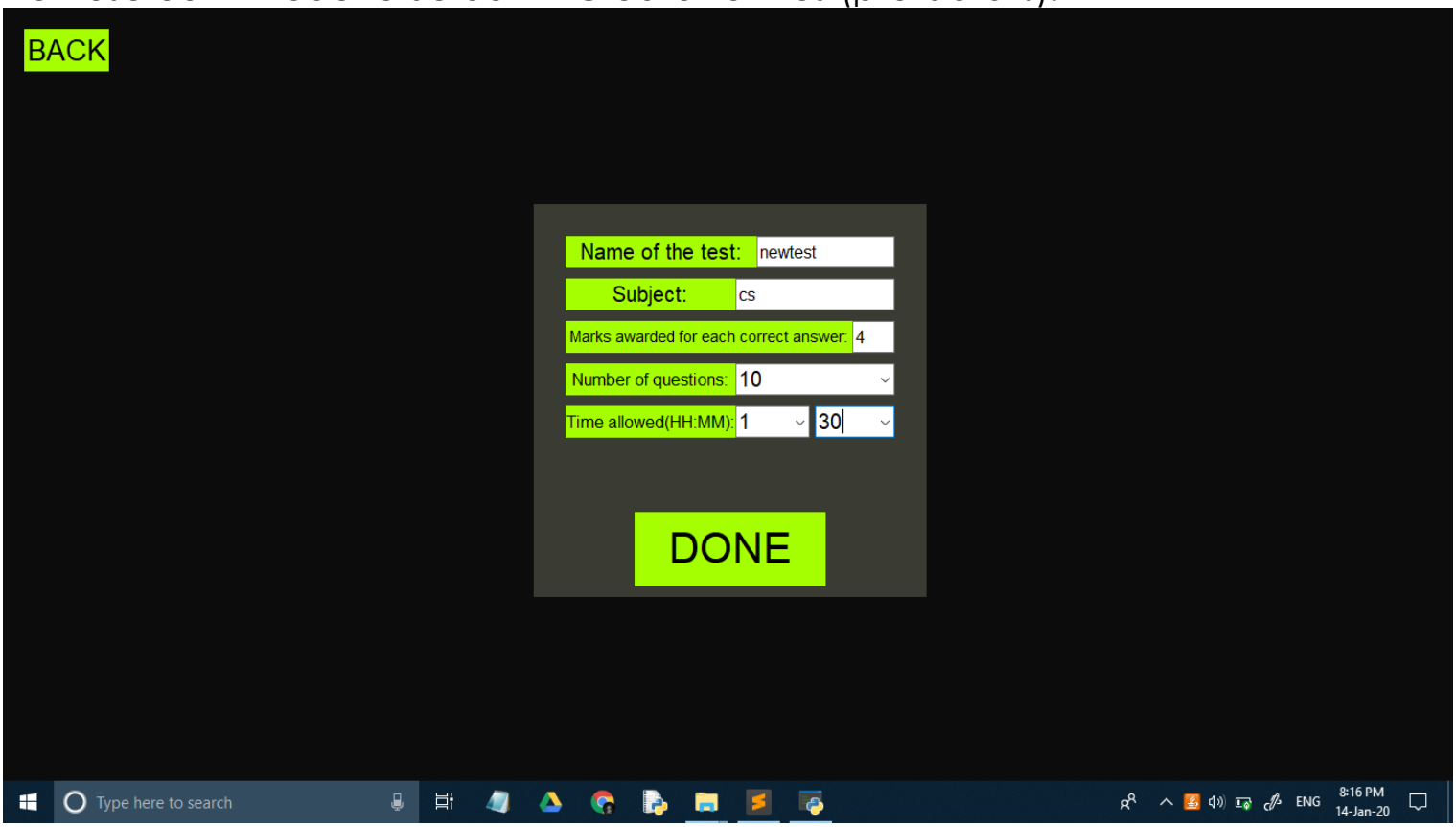
Homescreen>>TeacherScreen(authentication):



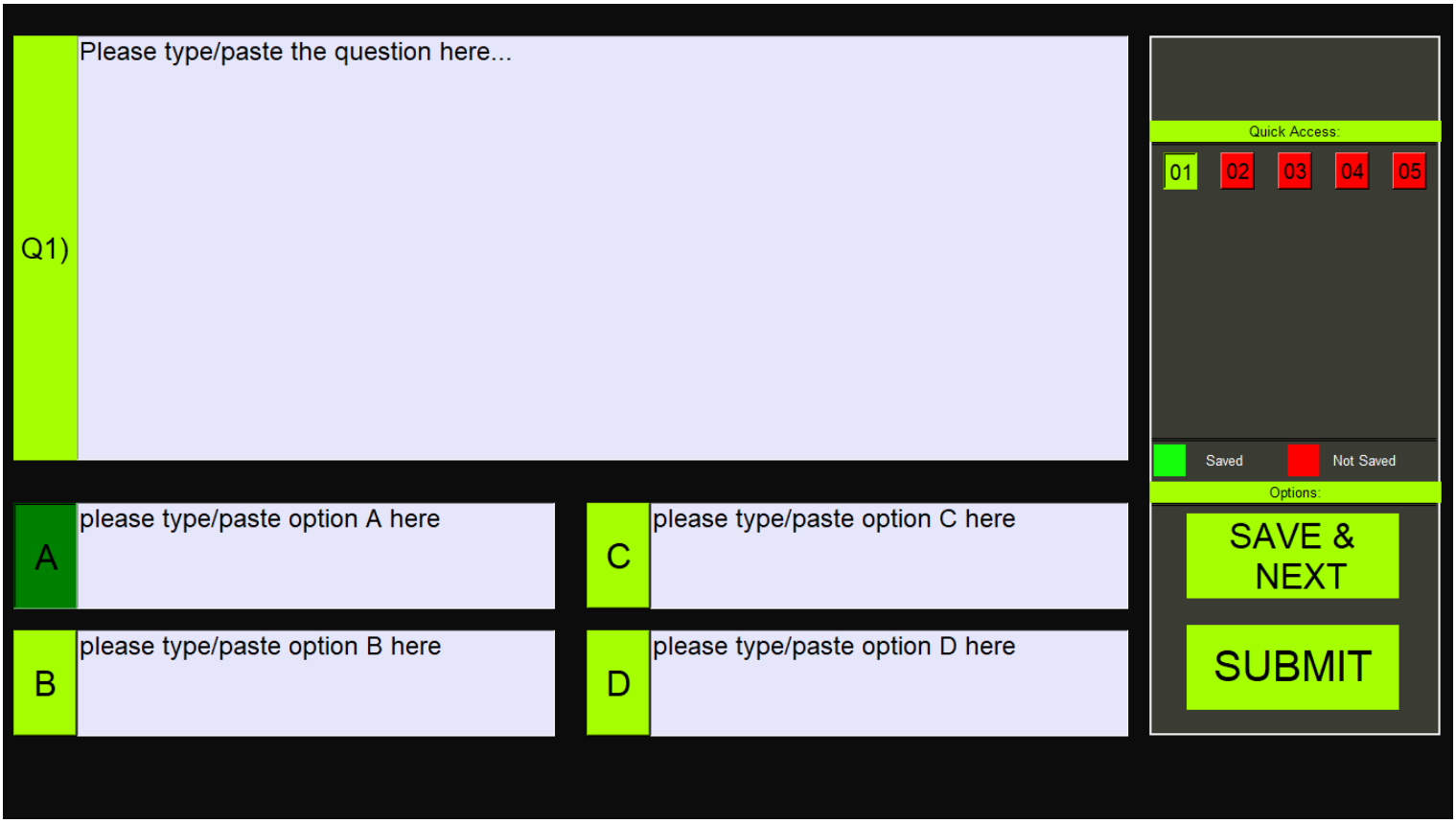
Homescreen>>TeacherScreen:



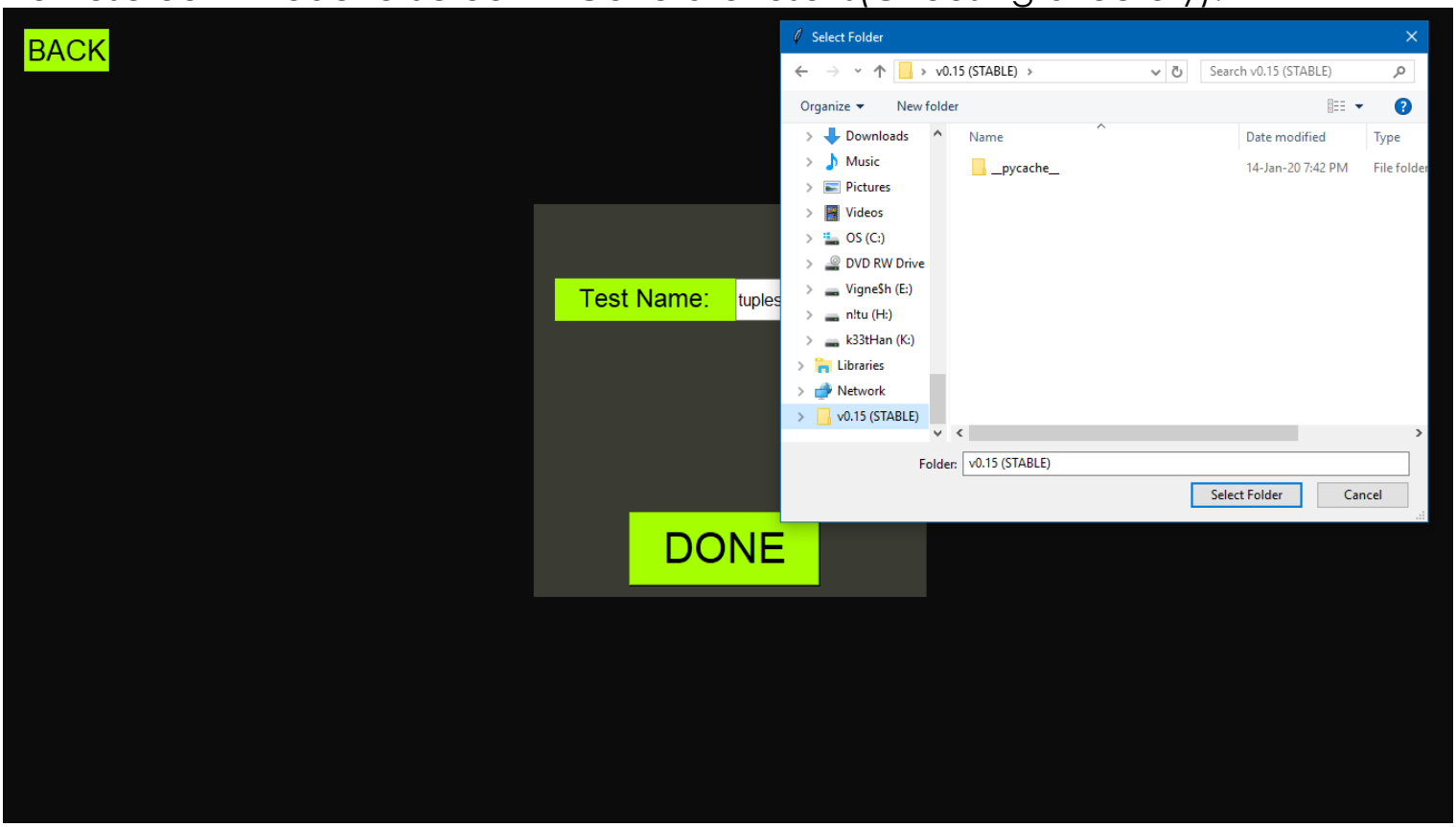
Homescreen>>TeacherScreen>>CreateNewTest(pre-details):



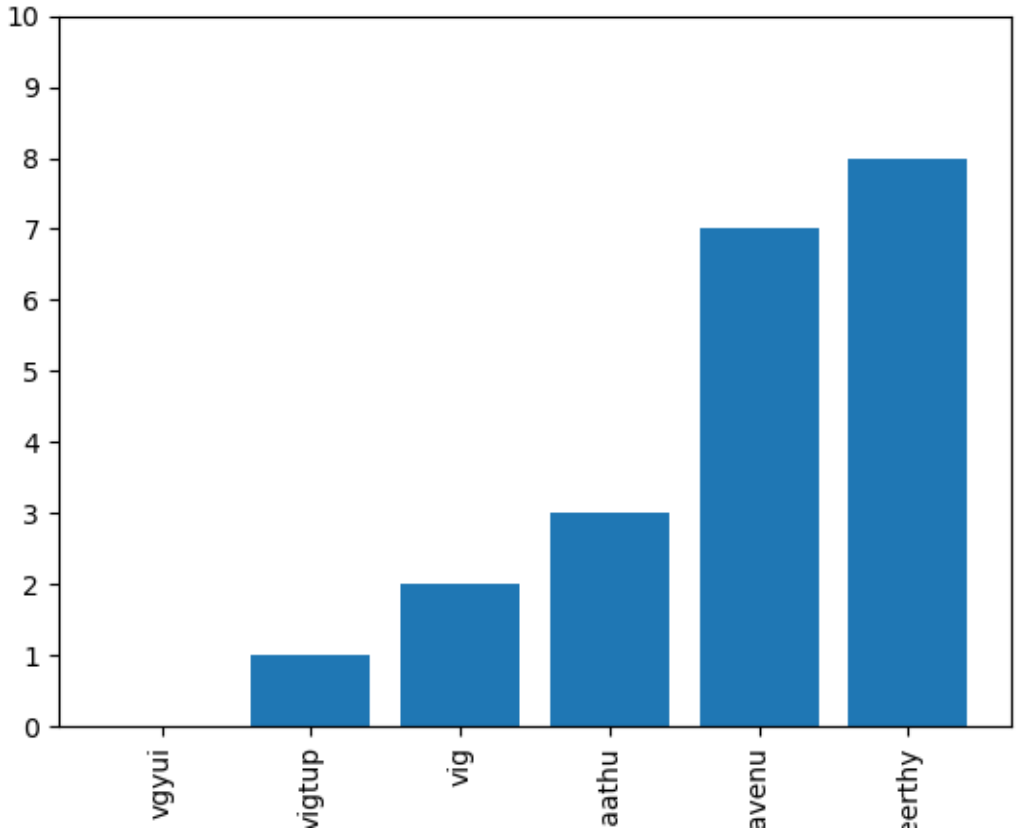
Homescreen>>TeacherScreen>>CreateNewTest:



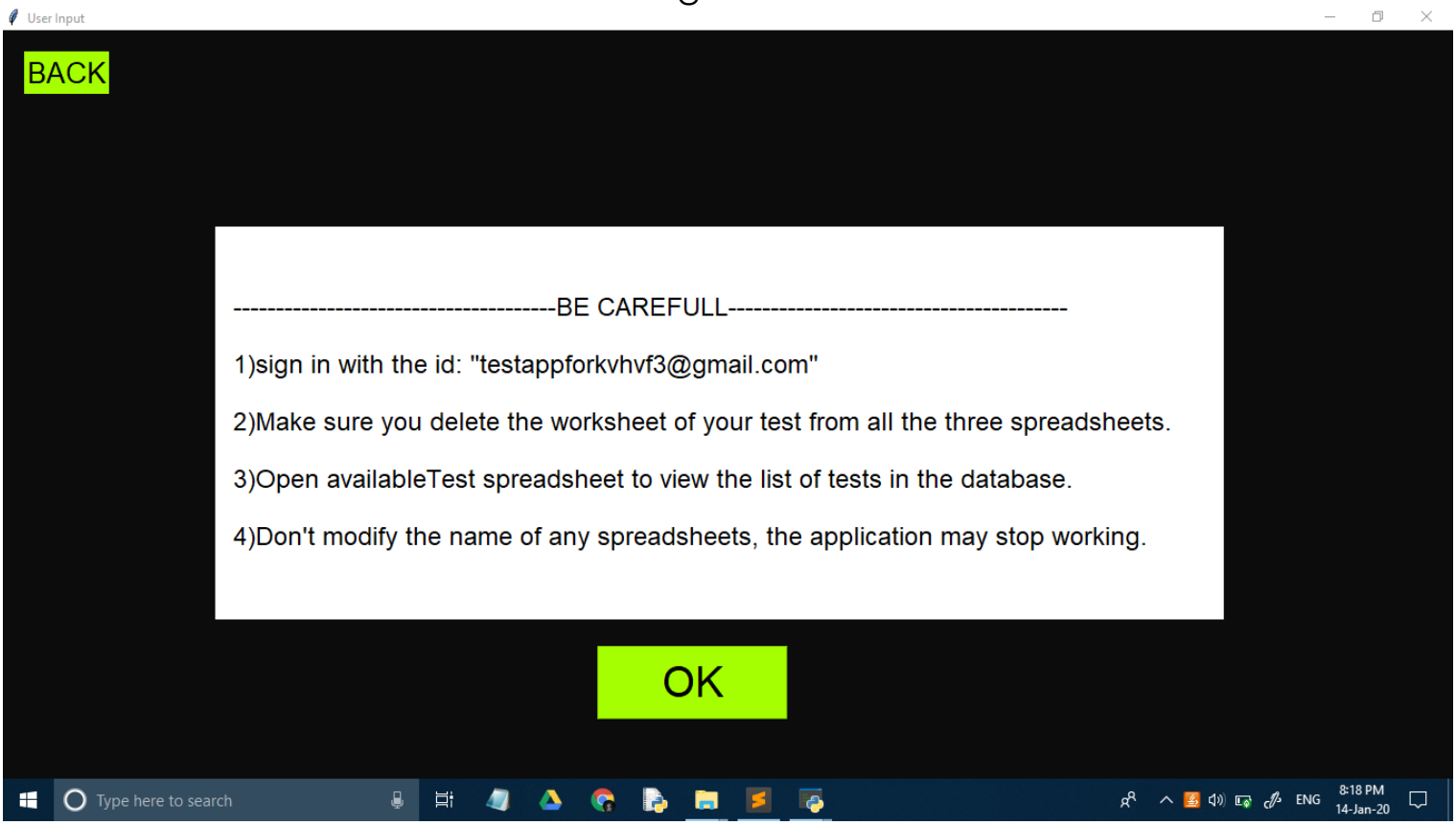
Homescreen>>TeacherScreen>>GenerateResults(Choosing directory):



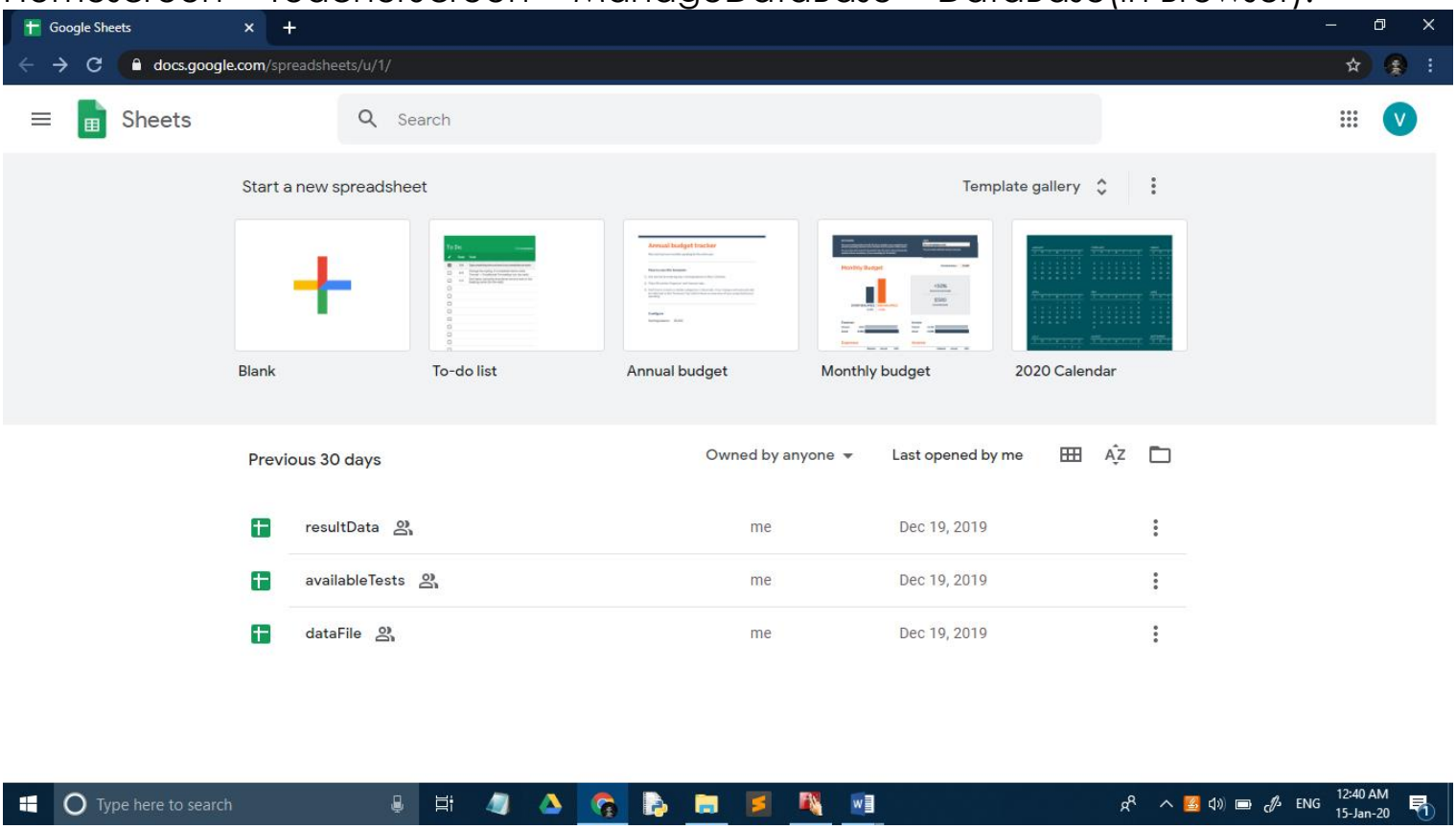
Saved result(sample):



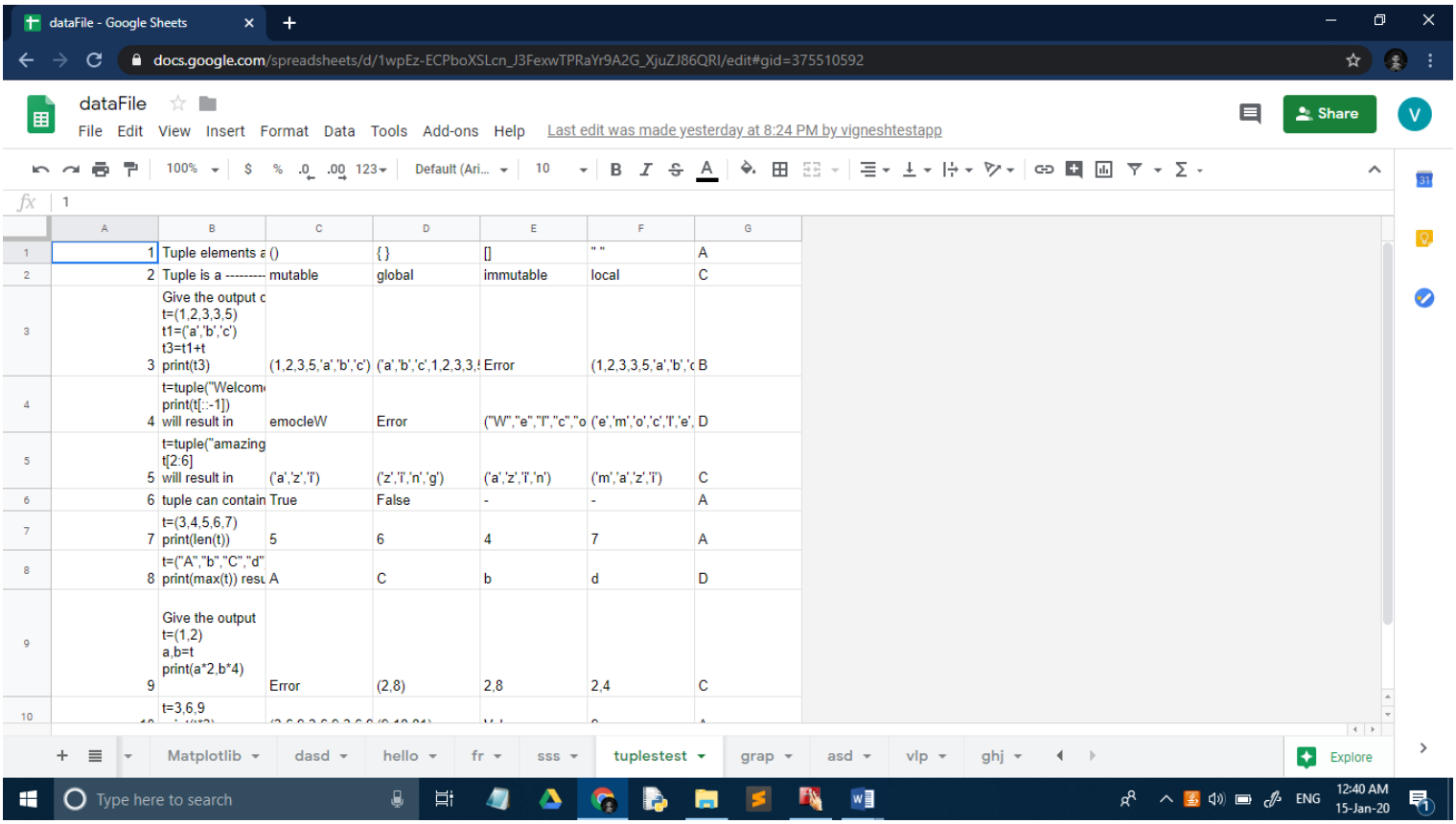
Homescreen>>TeacherScreen>>ManageDataBase:



Homescreen>>TeacherScreen>>ManageDataBase>>DataBase(in Browser):



Homescreen>>TeacherScreen>>ManageDataBase>>DataBase>>dataFile:



	A	B	C	D	E	F	G
1	1	Tuple elements a	()	{ }	[]	" "	A
2	2	Tuple is a ---------	mutable	global	immutable	local	C
3	3	Give the output c t=(1,2,3,3,5) t1=('a','b','c') t3=t1+t print(t3)	(1,2,3,5,'a','b','c')	('a','b','c',1,2,3,3,5	Error	(1,2,3,3,5,'a','b','c	B
4	4	t=tuple("Welcom print(t[::-1]) will result in	emocleW	Error	("W","e","l","c","o	('e','m','o','c','l','e',	D
5	5	t=tuple("amazing t[2:6] will result in	('a','z','i')	('z','i','n','g')	('a','z','i','n')	('m','a','z','i')	C
6	6	tuple can contain	True	False	-	-	A
7	7	t=(3,4,5,6,7) print(len(t))	5	6	4	7	A
8	8	t=("A","b","C","d" print(max(t)) resu	A	C	b	d	D
9	9	Give the output t=(1,2) a,b=t print(a*2,b*4)	Error	(2,8)	2,8	2,4	C
10		t=3,6,9					

BIBLIOGRAPHY

- www.stackoverflow.com
- www.pygame.org
- www.youTube.com
- Mrs: Sri Keerthy madam's classnotes
- Python with Sumitha Arora

With ♥

VIGNESH BALAJI S

Class XII A

KENDRIYA VIDYALAYA

HVF, AVADI, CH-600054