**GCSE COMPUTING PROGRAMMING PROJECT**

Second Draft

Date Started: 27/6/20

Date Finished: 15/10/20

Contents

* Task Analysis 3
* Text Version
  + Assumptions 4
  + Pseudocode 5
  + Flowchart 9
  + Code 10
  + Variable list 15
  + Testing 16
  + Final evaluation 18
* Pygame Version
  + Assumptions 19
  + Code 20
  + Variable list 29
  + Testing 31

**Task Analysis**

In this project I am making a music quiz game in which the first letter of all the words of a song is shown alongside the artist and the player must guess the name of the song.

* A user must log in with a username and password which should be authenticated with an external file
* Another external file should hold a list of song names and artists
* A random song should be selected at the start of each round and the artist and first letter of each song word should be displayed to the user
* 2 attempts can be made by the user to guess the song name
* If the guess is correct on the first guess, +3 points
* If the guess is correct on the second guess. +1 point
* If the guess is wrong on the second guess, no points and the game ends
* When the game ends it should display the total number of points the player has won and how the top 5 winning scores that have been stored in an external file

**TEXT VERSION**

**Assumptions**

* The user should be able to create a new account if required
* Some users may have malicious intent and thus all inputs must be sanitised
  + It should be verified to be an actual string
  + Characters used must comply with an array of accepted characters, if the characters are not in the array then the input is discarded
* Discarded guesses should not count towards the total guess count
* The songs do not have to be comprehensive of currents trends and so there may be lesser-known songs included
* The player will want to see their high score at the end
* I do not have to store the result of every game but only the personal high score and a list of top 5
* When displaying the song name, blanks where there were letters should be replaced with underscores and spaces, e.g. Under Pressure -> U \_ \_ \_ \_ P \_ \_ \_ \_ \_ \_ \_
* It must not be allowed to display the same song twice and thus if the user has solved every song then the program must end.
* Usernames and passwords do not need to be encrypted/hashed before placing into the external file

**Pseudocode**

VAR valid\_characters = [array of all letters numbers and symbols valid]

VAR userIndex = int

FUNC newAccount():

VAR accountCreated = FALSE

WHILE accountCreated is not TRUE:

VAR valid = FALSE

WHILE valid is not TRUE:

TRY:

VAR newUsnm = INPUT("Enter username: ")

VAR valid = TRUE

FOR i in newUsnm:

IF i not in valid\_characters and valid:

VAR valid = FALSE

PRINT("invalid characters")

ENDIF

ENDFOR

EXCEPT:

PRINT("Not valid")

ENDTRY

ENDWHILE

VAR valid = FALSE

WHILE valid is not TRUE:

TRY:

VAR newPswd = INPUT("Enter password: ")

VAR valid = TRUE

FOR i in newPswd:

IF i not in valid\_characters and valid:

VAR valid = FALSE

PRINT("invalid characters")

ENDIF

ENDFOR

EXCEPT:

PRINT("Not valid")

ENDTRY

ENDWHILE

WITH open("Logins.json", "r") as VAR file:

VAR data = file data

VAR valid = TRUE

FOR i in data["Logins"]:

IF i["Usnm"] == newUsnm and valid is TRUE:

PRINT("Username already in use, please choose another")

VAR valid = FALSE

ENDIF

ENDFOR

IF valid is TRUE:

VAR newLogin = {"Usnm":newUsnm, "Pswd":newPswd, "highscore":0}

VAR data["Logins"].append(newLogin)

global VAR userIndex

VAR userIndex = data["Logins"].index({"Usnm":newUsnm, "Pswd":newPswd, "highscore":0})

WITH open("Logins.json", "w+") as file:

put data in file

PRINT("login created")

VAR accountCreated = TRUE

ENDWITH

ENDIF

ENDWITH

ENDWHILE

VAR newPswd = str

ENDFUNC

FUNC login():

VAR loggedIn = FALSE

WHILE loggedIn is not TRUE:

VAR valid = FALSE

WHILE valid is not TRUE:

TRY:

VAR Usnm = INPUT("Enter username: ")

VAR valid = TRUE

FOR i in Usnm:

IF i is not in valid\_characters and valid is TRUE:

VAR valid = FALSE

PRINT("invalid characters")

ENDIF

ENDFOR

EXCEPT:

PRINT("Not a possible username")

ENDTRY

ENDWHILE

VAR valid = FALSE

WHILE valid is not TRUE:

TRY:

VAR Pswd = INPUT("Enter password: ")

VAR valid = TRUE

EXCEPT:

PRINT("Not a possible password")

ENDTRY

ENDWHILE

WITH open("Logins.json", "r") as VAR file:

VAR data = file data

ENDWITH

FOR i in range(0,length(data["Logins"])):

IF data["Logins"][i]["Usnm"] == Usnm and data["Logins"][i]["Pswd"] == Pswd:

VAR loggedIn = TRUE

global VAR userIndex

VAR userIndex = i

VAR Pswd = ""

ENDIF

ENDFOR

IF loggedIn is not TRUE:

PRINT("Username or Password incorrect")

ENDIF

ENDFUNC

FUNC startup():

VAR valid = FALSE

WHILE valid is FALSE:

TRY:

VAR cPlayerCheck = INPUT("Do you have a login yet? (Y/N) ").uppercase()

EXCEPT:

PRINT()

PRINT("whatever you typed was not what you were supposed to")

ELSE:

VAR valid = TRUE

array options = ["Y","N"]

IF cPlayerCheck not in options:

PRINT("That wasn't one of the options")

VAR valid = FALSE

ELSE:

IF cPlayerCheck == "Y":

login()

ELSE:

newAccount()

ENDIF

ENDIF

ENDTRY

ENDFUNC

FUNC endGame(score,userIndex):

PRINT()

PRINT("GAME OVER")

WITH open("Logins.json", "r") as file:

VAR data = file data

IF data["Logins"][userIndex]["highscore"] <= score :

data["Logins"][userIndex]["highscore"] = score

IF score > data["Highscore"][0]["Score"]:

data["Highscore"].insert(0,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

data["Highscore"].pop(5)

ELIF score > data["Highscore"][1]["Score"]:

data["Highscore"].insert(1,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

data["Highscore"].pop(5)

ELIF score > data["Highscore"][2]["Score"]:

data["Highscore"].insert(2,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

data["Highscore"].pop(5)

ELIF score > data["Highscore"][3]["Score"]:

data["Highscore"].insert(3,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

data["Highscore"].pop(5)

ELIF score > data["Highscore"][4]["Score"]:

data["Highscore"].insert(4,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

data["Highscore"].pop(5)

ELSE:

pass

PRINT()

PRINT("HIGHSCORES:")

VAR place = 0

FOR i in data["Highscore"]:

VAR place += 1

PRINT(place,": ", data["Highscore"][place-1]["User"]," with ", data["Highscore"][place-1]["Score"]," points!")

PRINT()

PRINT("Your highscore is", data["Logins"][userIndex]["highscore"])

ENDWITH

WITH open("Logins.json", "w") as file:

put data in file

ENDWITH

quit()

ENDFUNC

startup()

indexsUsed = []

WITH open("Songs.json", "r") as file:

songs = file data

ENDWITH

VAR score = 0

WHILE TRUE:

IF length(songs["Songs"]) == length(indexsUsed):

PRINT("You appear to have solved every song in our list, congrats.")

endGame(score,userIndex)

ENDIF

VAR songNotUsed = FALSE

WHILE songNotUsed is FALSE:

VAR index = randomintbetween(0,length(songs["Songs"])-1)

IF index not in indexsUsed:

VAR indexsUsed.append(index)

VAR songNotUsed = TRUE

ENDIF

ENDWHILE

VAR song = songs["Songs"][index]["name"]

VAR artist = songs["Songs"][index]["artist"]

VAR songWords = song.split(" ")

VAR displaySong = ""

FOR i in songWords:

VAR displaySong += songWords[songWords.index(i)][0]

FOR j in range(0,length(i)-1):

VAR displaySong += " \_"

ENDFOR

VAR displaySong += " "

ENDFOR

PRINT("song is ",displaySong)

PRINT("artist is ",artist)

VAR guesses = 0

VAR correct = FALSE

WHILE guesses < 2:

VAR valid = FALSE

WHILE valid is FALSE:

TRY:

VAR guess = INPUT("Enter guess: ")

EXCEPT:

PRINT("Not a valid guess")

ELSE:

VAR valid = TRUE

FOR i in guess:

IF i not in valid\_characters and valid:

VAR valid = FALSE

PRINT("invalid characters")

ENDIF

ENDFOR

VAR guesses += 1

ENDTRY

IF guess.uppercase() == song.uppercase():

IF guesses == 1:

PRINT("CORRECT, +3 POINTS")

VAR score += 3

ELSE:

PRINT("CORRECT, +1 POINTS")

VAR score += 1

ENDIF

PRINT(" YOU HAVE ", score, "POINTS")

VAR correct = TRUE

VAR guesses = 2

ELSE:

PRINT("INCORRECT")

ENDIF

IF correct is TRUE:

PRINT()

PRINT()

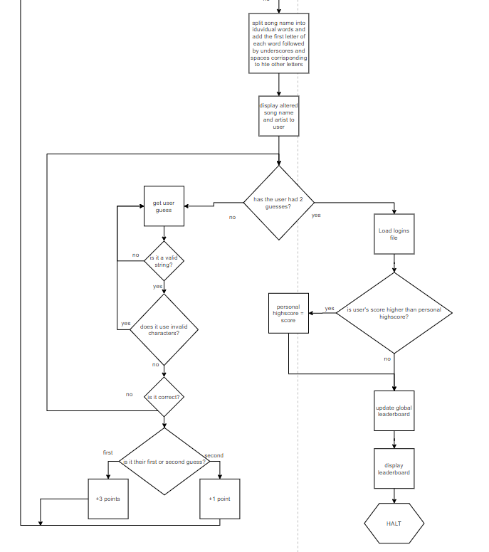
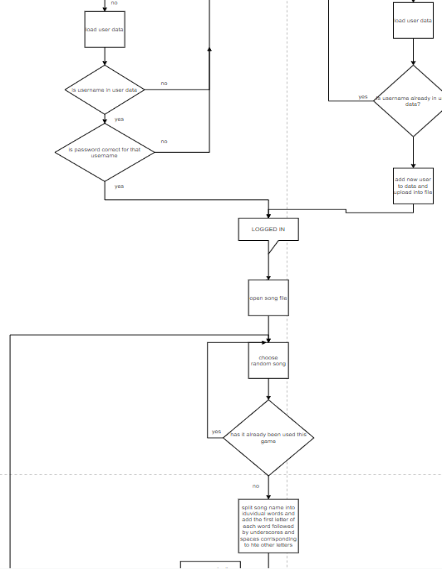
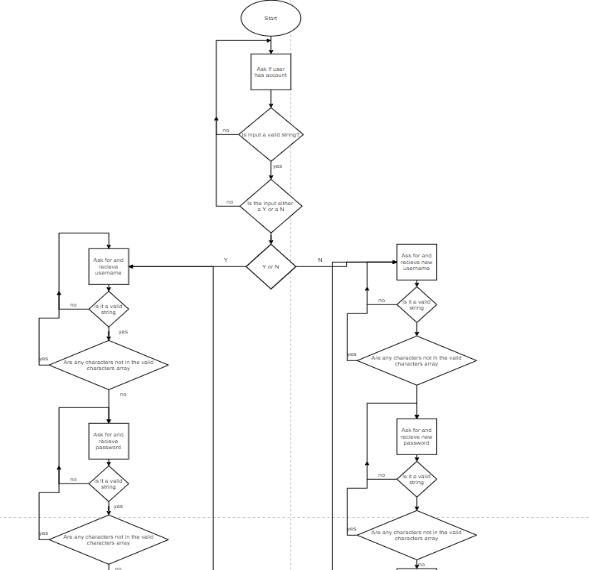
ELSE:

endGame(score,userIndex)

ENDIF

ENDWHILE

**Flowchart**



Link: https://drive.google.com/file/d/1B963G8Nrocq2BAUn4WUnurt7rAqnPvJg/view?usp=sharing

**Code**

**MAIN**

import time,json,random

valid\_characters = ["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z","a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z",

";",":",",",".","'","!","£","$","%","^","&","?","@","~","-","\_"," ","`","¬","0","1","2","3","4","5","6","7","8","9"]

userIndex = int

def newAccount():

accountCreated = False

while not accountCreated:

valid = False

while not valid:

try:

newUsnm = str(input("Enter username: "))

valid = True

for i in newUsnm:

if i not in valid\_characters and valid:

valid = False

print("invalid characters")

except:

print("Not valid")

valid = False

while not valid:

try:

newPswd = str(input("Enter password: "))

valid = True

for i in newPswd:

if i not in valid\_characters and valid:

valid = False

print("invalid characters")

except:

print("Not valid")

#load file dictionary

with open("Logins.json", "r") as file:

data = json.load(file)

#Check username hasn't already been chosen

valid = True

for i in data["Logins"]:

if i["Usnm"] == newUsnm and valid:

print("Username already in use, please choose another")

valid = False

if valid:

#assemble new account dict for file insertion

newLogin = {"Usnm":newUsnm, "Pswd":newPswd, "highscore":0}

#edit file dict

data["Logins"].append(newLogin)

#set global variables of username for later file editing

global userIndex

userIndex = data["Logins"].index({"Usnm":newUsnm, "Pswd":newPswd, "highscore":0})

#insert edited dictionary into file

with open("Logins.json", "w+") as file:

json.dump(data, file, indent = 5)

print("login created")

accountCreated = True

#erase usnm and password variables for security

newPswd = str

def login():

loggedIn = False

while not loggedIn:

valid = False

while not valid:

try:

Usnm =str(input("Enter username: "))

valid = True

for i in Usnm:

if i not in valid\_characters and valid:

valid = False

print("invalid characters")

except:

print("Not a possible username")

valid = False

while not valid:

try:

Pswd = str(input("Enter password: "))

valid = True

except:

print("Not a possible password")

#load accounts file

with open("Logins.json", "r") as file:

data = json.load(file)

#loop through logins in file

for i in range(0,len(data["Logins"])):

if data["Logins"][i]["Usnm"] == Usnm and data["Logins"][i]["Pswd"] == Pswd:

loggedIn = True

#set global variables of username for later file editing

global userIndex

userIndex = i

#reset password for security reasons

Pswd = ""

#failed login handling

if not loggedIn:

print("Username or Password incorrect")

def startup():

valid = False

while not valid:

try:

cPlayerCheck = str(input("Do you have a login yet? (Y/N) ")).upper()

except:

print()

print("whatever you typed was not what you were supposed to")

else:

valid = True

options = ["Y","N"]

if cPlayerCheck not in options:

print("That wasn't one of the options")

valid = False

else:

if cPlayerCheck == "Y":

login()

else:

newAccount()

def endGame(score,userIndex):

print()

print("GAME OVER")

with open("Logins.json", "r") as file:

data = json.load(file)

if data["Logins"][userIndex]["highscore"] <= score :

data["Logins"][userIndex]["highscore"] = score

if score > data["Highscore"][0]["Score"]:

data["Highscore"].insert(0,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

data["Highscore"].pop(5)

elif score > data["Highscore"][1]["Score"]:

data["Highscore"].insert(1,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

data["Highscore"].pop(5)

elif score > data["Highscore"][2]["Score"]:

data["Highscore"].insert(2,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

data["Highscore"].pop(5)

elif score > data["Highscore"][3]["Score"]:

data["Highscore"].insert(3,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

data["Highscore"].pop(5)

elif score > data["Highscore"][4]["Score"]:

data["Highscore"].insert(4,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

data["Highscore"].pop(5)

else:

pass

print()

print("HIGHSCORES:")

place = 0

for i in data["Highscore"]:

place += 1

print(place,": ", data["Highscore"][place-1]["User"]," with ", data["Highscore"][place-1]["Score"]," points!")

print()

print("Your highscore is", data["Logins"][userIndex]["highscore"])

with open("Logins.json", "w") as file:

json.dump(data, file, indent = 5)

quit()

startup()

indexsUsed = []

with open("Songs.json", "r") as file:

songs = json.load(file)

score = 0

playing = True

while playing:

#select song

if len(songs["Songs"]) == len(indexsUsed):

print("You appear to have solved every song in our list, congrats.")

endGame(score,userIndex)

quit()

songNotUsed = False #Variable that keeps track of if we have found an unused song

while not songNotUsed:

index = random.randint(0,len(songs["Songs"])-1)

if index not in indexsUsed:

indexsUsed.append(index)

songNotUsed = True

song = songs["Songs"][index]["name"]

artist = songs["Songs"][index]["artist"]

#edit song name to display to player

songWords = song.split(" ")

displaySong = ""

for i in songWords:

displaySong += songWords[songWords.index(i)][0]

for j in range(0,len(i)-1):

displaySong += " \_"

displaySong += "   "

print("song is ",displaySong)

print("artist is ",artist)

#Guessing cycle

guesses = 0

correct = False

while guesses < 2:

valid = False

while not valid:

try:

guess = str(input("Enter guess: "))

except:

print("Not a valid guess")

else:

valid = True

for i in guess:

if i not in valid\_characters and valid:

valid = False

print("invalid characters")

guesses += 1

if guess.upper() == song.upper():

if guesses == 1:

print("CORRECT, +3 POINTS")

score += 3

else:

print("CORRECT, +1 POINTS")

score += 1

print(" YOU HAVE ", score, "POINTS")

correct = True

guesses = 2 #janky way of breaking guess loop

else:

print("INCORRECT")

if correct:

print()

print()

#breakpoints to seperate questions

else:

endGame(score,userIndex)

**Logins file**

{

     "Logins": [

          {

               "Usnm": "Violet Baudelaire",

               "Pswd": "protect siblings",

               "highscore": 20

          },

          {

               "Usnm": "Klaus Baudelaire",

               "Pswd": "Books4Lyfe",

               "highscore": 27

          },

          {

               "Usnm": "Sunny Baudelaire",

               "Pswd": "I'm not a baby",

               "highscore": 15

          },

          {

               "Usnm": "Count Olaf",

               "Pswd": "IHateBaudelaires",

               "highscore": 4

          },

          {

               "Usnm": "Royaljames99",

               "Pswd": "Hello",

               "highscore": 0

          }

     ],

     "Highscore": [

          {

               "User": "Klaus Baudelaire",

               "Score": 27

          },

          {

               "User": "Klaus Baudelaire",

               "Score": 25

          },

          {

               "User": "Violet Baudelaire",

               "Score": 20

          },

          {

               "User": "Violet Baudelaire",

               "Score": 17

          },

          {

               "User": "Sunny Baudelaire",

               "Score": 15

          }

     ]

}

**Songs file**

To save you the space of that massive file the entire file is in the format:

{

“Songs”:[

{

“name”: “songname”

“artist”: “artistname”

}

]

}

**Variable list**

|  |  |
| --- | --- |
| VARIABLE | USE |
| userIndex | Integer, Stores index of user in relation to the main logins file for ease of location |
| accountCreated | Boolean, used to maintain while loop for the creation of a new user account |
| valid | Boolean, used throughout program for small while loops relating to user inputs and whether they are acceptable |
| newUsnm | String, user inputted new username to be validated and placed into logins file |
| valid\_characters | Array, all allowed characters for user input |
| newPswd | String, user inputted new password to be validated and placed into logins file |
| data | Dictionary, stores full data from whichever file is currently being edited |
| newLogin | Dictionary, new login to be added to logins file comprised of newUsnm and newPswd |
| loggedIn | Boolean, used to maintain while loop for the login subroutine |
| Usnm | String, user inputted username to be checked against Logins file |
| Pswd | String, user inputted password to be checked against Logins file |
| cPlayerCheck | String, user inputted Y/N to say whether they have an account yet |
| score | Integer, player’s score |
| place | Integer, used when shuffling leaderboard, holds index within array for ease of editing |
| indexsUsed | Array, stores the indexs of every song within the Songs file that has so far been used |
| songNotUsed | Boolean, keeps track of if we have found an unused song yet |
| index | Integer, random number used as index for a song |
| song | String, the songname of currently used song |
| artist | String, the artist of currently used song |
| songWords | Array, stores the individual words of the songname |
| displaySong | String, the edited songname after letters have been removed, is the songname shown to the player |
| guesses | Integer, number of guesses the user has had on current song |
| correct | Boolean, whether the user has guessed the song correctly |
| guess | String, the user’s inputted guess on what the current song is |

**Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No. | Test Details | Expected Result | Actual Result | Working/Needs fix |
| 1 | Testing Y/N check with values: “y”, “Y”, “n”, “N”, “;”,”a”,”P” for a full analysis of its validation | Values “y”,”Y”,”n”,”N” should be accepted whilst the others should be rejected | “y” = accepted  “Y” = acepted  “n” = accepted  “N” = accepted  “;” = rejected with explanatory message  “a” = rejected with explanatory message  “P” = rejected with explanatory message | Working correctly |
| 2 | Testing Account Setup: will attempt to create new account with name “BaRrY” and password “yRrAb” | Account should create successfully and should be visible in the Logins file for eay conformation | ##console##  Enter username: BaRrY  Enter password: yRrAb  login created  ##Logins file##  {     "Usnm": "BaRrY",     "Pswd": "yRrAb",     "highscore": 0  } | Working correcctly |
| 3 | Testing Account Setup: will attempt to create account with name “#IHATETRUMP”  And if that succeeds I will use the password “#BidenSuks2” | The username should be rejected for the use of invalid characters before getting to the password which would also be rejected on the same grounds |  | Working correctly |
| 4 | Testing Account Setup: will attempt to create account with name “IHATETRUMP” and if that succeeds I will use the password “#BidenSuks2” | The username should be accepted but the password rejected on the use of invalid characters |  | Working correctly |
| 5 | Testing Account Login: I will attempt to login to the account created in test 2 with username “BaRrY” and password “yRrAb” | It should log me in successfully and initialise the game |  | Working correctly |
| 6 | Testing Account Login: I’ll attempt to login to the account created in test 2 but I will put a capital where every lowercase letter is and vice versa for both the username and the password | Login attempt should be rejected as both fields are case sensitive |  | Working correctly |
| 7 | Testing Account Login: I’ll attempt to login to the account created in test 2 but I will use the correct username but use “asdf” as the password | Login attempt should fail |  | Working correctly |
| 8 | Testing Account Login: I’ll attempt to login to the account created in test 2 but I will use the correct password but the username “asdf” | Login attempt should fail |  | Working correctly |
| 9 | I will login to the account from test 2 and complete the first song with my guess using only lowercase letters | Guess should be marked as correct and 3 points should be added to the score |  | Working correctly |
| 10 | I will login to the account from test 2 and complete the first song with my guess using only uppercase letters | Guess should be marked as correct and 3 points should be added to the score |  | Working correctly |
| 11 | I will login to the account from test 2 and fail the first guess before getting the second guess first | Second guess should be marked as correct and 1 point should be added to the score |  | Working correctly although when the score is one it still says plural points rather than point |
| 12 | I will login to the account from test 2 and fail both guesses | Game should end and display leaderboard that currently contains placeholder accounts |  | Working correctly |
| 13 | I will get to a score of 20 and then fail 2 guesses in a row | Game should end and my score should appear on the leaderboard |  | Working correctly |

**Evaluation**

My program is successful in fulfilling its requirements thus:

It is able to create a new account with an inputted username and password or login to an already existing account. It can randomly select 1 song from a list of thirty with no chance of it repeating itself. It can check the user’s guesses whilst preventing injection attacks and ignoring capitalisation. It can keep track of if the user guessed correctly on the first or second attempt and give points accordingly. It can detect whether a user has failed both attempts and then can successfully update and display highscores.

It does have some key issues, however:

Security: With the entire program running in the command line, there is nothing that prevents other people from scrolling up and seeing the active user’s username and password. Granted there is an unencrypted json file with them all but the point stands. Speaking of that text file, with none of the passwords using hashing or even at the very least basic encryption, the passwords of every user are exposed. Whilst this is just a simple game where it doesn’t really matter it sets a bad example.

Interface: The interface of this game is, to be blunt, appalling. It is entirely in the command line and thus creates many problems, for example, if it is run in the regular windows command line then it automatically closes before users can see the final messages. This means that the user would preferably have to use something like VSCode to run the program which is less than ideal.

Overall, this program fulfils the requirements set out by the original brief however that brief did not cover many important areas of the game which has left serious vulnerabilities and a lack of polish.

**PYGAME VERSION**

**Assumptions**

* The user should be able to create a new account if required
* Some users may have malicious intent and thus all inputs must be sanitised
  + It should be verified to be an actual string
  + Characters used must comply with an array of accepted characters, if the characters are not in the array then the input is discarded
* Discarded guesses should not count towards the total guess count
* The songs do not have to be comprehensive of currents trends and so there may be lesser-known songs included
* The player will want to see their high score at the end
* I do not have to store the result of every game but only the personal high score and a list of top 5
* When displaying the song name, blanks where there were letters should be replaced with underscores and spaces, e.g. Under Pressure -> U \_ \_ \_ \_ P \_ \_ \_ \_ \_ \_ \_
* It must not be allowed to display the same song twice and thus if the user has solved every song then the program must end.
* Usernames and passwords do not need to be encrypted/hashed before placing into the external file
* The user will not have access to the python raw input console or the program files and thus injection attacks through them are not an issue.

This pygame version is only being made for my own personal satisfaction and thus I will not be reiterating the flowcharts and pseudocode that were seen in the text version as they will be largely exactly the same but with more complicated input and output statements.

**CODE**

**Pygame\_Classes**

import pygame

import sys

from pygame.locals import \*

screenClicked = False

keyPressed = []

keyBeenPressed = False

#Stores info about pygame widnow and deals with initialisation

class window():

    def \_\_init\_\_(self, ww, wh, name):

        self.ww = ww

        self.wh = wh

        self.name = name

    def initScreen(self):

        pygame.init()

        self.windowSurface = pygame.display.set\_mode((self.ww,self.wh),0,32)

        pygame.display.set\_caption(self.name)

#Collects and sorts various pygame events

def eventGet():

    global screenClicked

    global keyPressed

    global keyBeenPressed

    screenClicked = False

    keyBeenPressed = False

    for event in pygame.event.get():

        #Quit if user presses x button

        if event.type == pygame.QUIT:

            pygame.quit()

            quit()

        #detect if user clicks screen

        elif event.type == pygame.MOUSEBUTTONDOWN:

            screenClicked = True

        #detect keyboard presses and identifies key

        elif event.type == pygame.KEYDOWN:

            keyBeenPressed = True

            keyPressed = []

            keyPressed.append(event.unicode)

            keyPressed.append(event.key)

#Text for putting on screen

class Text():

    def \_\_init\_\_(self, font, fontsize, words, colour = (0,0,0)):

        self.font = str(font)

        self.fontsize = fontsize

        self.words = str(words)

        self.colour = colour

        #use user variables to generate txt

        self.font = pygame.font.SysFont(self.font, self.fontsize)

        self.text = self.font.render(self.words,True,self.colour)

        self.textRect = None

    #generate location of text

    def createSquare(self, left, top):

        self.textRect = self.text.get\_rect()

        self.textRect.left = left

        self.textRect.top = top

    #show text on screen

    def showText(self,window):

        window.windowSurface.blit(self.text, self.textRect)

#User input text boxes

class textBox():

    def \_\_init\_\_(self,width,height,x,y,prompt,fontsize = 25, colour = (255,255,255)):

        self.prompt = prompt #shown when the text box is empty

        self.complete = False #set as True when enter key is pressed

        self.text = "" #inputted text

        self.fontsize = fontsize #fontsize

        self.font = "arial" #font

        self.colour = colour #colour as RGB

        self.width = width #width of textbox

        self.height = height #height of textbox

        self.x = x #x position

        self.y = y #y position

        self.rect = pygame.Rect(self.x,self.y,self.width,self.height) #rect object of textbox

        self.active = False

    #checks if player is using textbox by detecting if clicks are on box

    def checkActive(self):

        global screenClicked

        if screenClicked:

            x, y = pygame.mouse.get\_pos()

            if self.rect.collidepoint((x, y)):

                self.active = True

            else:

                self.active = False

    def keypressed(self):

        global keyPressed

        global keyBeenPressed

        if self.active:

            if keyBeenPressed:

                if keyPressed[1] == pygame.K\_RETURN:

                    self.complete = True

                elif keyPressed[1] == pygame.K\_BACKSPACE:

                    self.text = self.text[:-1]

                else:

                    self.text += keyPressed[0]

    def showTextBox(self, window):

        if self.text == "":

            screenText = Text(self.font, self.fontsize, self.prompt)

        else:

            screenText = Text(self.font, self.fontsize, self.text)

        screenText.createSquare(self.x + 5, self.y + (self.height/2) - (self.fontsize/2))

        pygame.draw.rect(window.windowSurface, self.colour, self.rect)

        if self.text == "" and self.active != True:

            screenText.showText(window)

        elif screenText != "":

            screenText.showText(window)

    def reset(self):

        self.text = ""

        self.complete = False

**MAIN**

import time, json, random, pygame, Pygame\_Classes

valid\_characters = ["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z","a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z",

";",":",",",".","'","!","£","$","%","^","&","?","@","~","-","\_"," ","`","¬","0","1","2","3","4","5","6","7","8","9"]

userIndex = int

def newAccount(window):

    global valid\_characters

    loggedIn = False

    while not loggedIn:

        valid = False

        while not valid:

            #create and show text input box until user hits enter

            newUsnmBx = Pygame\_Classes.textBox(400, 50, window.ww/2 - 200, window.wh/2 - 25, "Enter new username")

            while not newUsnmBx.complete:

                window.windowSurface.fill((0,0,0))

                Pygame\_Classes.eventGet()

                newUsnmBx.checkActive()

                newUsnmBx.keypressed()

                newUsnmBx.showTextBox(window)

                pygame.display.update()

            #Check if entered text is a valid string and does not contain banned characters

            try:

                newUsnm = newUsnmBx.text

                valid = True

                for i in newUsnm:

                    if i not in valid\_characters and valid:

                        valid = False

                        charMessage = "You can only use the following characters: "

                        for i in valid\_characters:

                            charMessage += i

                        invalidText = Pygame\_Classes.Text("arial",25,charMessage, colour = (255,255,255))

            except:

                invalidText = Pygame\_Classes.Text("arial",25,"INVALID USERNAME", colour = (255,255,255))

            #Show invalid messige if necissary

            if not valid:

                invalidText.createSquare(100,600)

                invalidText.showText(window)

                pygame.display.update()

                time.sleep(2)

        #repeat previous cycle for password

        valid = False

        while not valid:

            newPswdBx = Pygame\_Classes.textBox(400,50, window.ww/2 - 200, window.wh/2 - 25, "Enter new password")

            while not newPswdBx.complete:

                window.windowSurface.fill((0,0,0))

                Pygame\_Classes.eventGet()

                newPswdBx.checkActive()

                newPswdBx.keypressed()

                newPswdBx.showTextBox(window)

                pygame.display.update()

            try:

                newPswd = newPswdBx.text

                valid = True

                for i in newPswd:

                    if i not in valid\_characters and valid:

                        valid = False

                        charMessage = "You can only use the following characters: "

                        for i in valid\_characters:

                            charMessage += i

                        invalidText = Pygame\_Classes.Text("ariel",25,charMessage, colour = (255,255,255))

            except:

                invalidText = Pygame\_Classes.Text("arial",25,"INVALID PASSWORD", colour = (255,255,255))

            #Display error message if needed

            if not valid:

                invalidText.createSquare(100,600)

                invalidText.showText(window)

                pygame.display.update()

                time.sleep(2)

        ##Assemble data and place into file

        #Assemble data into dict

        newLogin = {"Usnm":newUsnm, "Pswd":newPswd, "highscore":0}

        #load login file

        with open("Logins.json", "r") as file:

            data = json.load(file)

        for i in data["Logins"]:

            if i["Usnm"] == newUsnm:

                loggedIn = False

                invalidText = Pygame\_Classes.Text("ariel",25,"Username already in use", colour = (255,255,255))

                invalidText.createSquare(100,600)

                invalidText.showText(window) #display error message

                pygame.display.update()

                time.sleep(2)

            else:

                #if its the final loop

                if data["Logins"].index(i) == (len(data["Logins"])-1):

                    #set loggedin to true so as to exit main loop

                    loggedIn = True

                    #compile and deposit new login

                    data["Logins"].append(newLogin)

                    global userIndex

                    userIndex = data["Logins"].index({"Usnm":newUsnm, "Pswd":newPswd, "highscore":0})

                    #insert data into file

                    with open("Logins.json", "w+") as file:

                        json.dump(data, file, indent = 5)

                    break

def login(window):

    loggedIn = False

    while not loggedIn:

        #Create username box and wait for user to enter

        usnmBox = Pygame\_Classes.textBox(400,50,window.ww/2 - 200, window.wh/2 - 25, "Enter Username")

        while not usnmBox.complete:

            window.windowSurface.fill((0,0,0))

            Pygame\_Classes.eventGet()

            usnmBox.checkActive()

            usnmBox.keypressed()

            usnmBox.showTextBox(window)

            pygame.display.update()

        #Test for valid characters

        try:

            usnm = usnmBox.text

            valid = True

            for i in usnm:

                if i not in valid\_characters and valid:

                    valid = False

                    charMessage = "You can only use the following characters: "

                    for i in valid\_characters:

                        charMessage += i

                    invalidText = Pygame\_Classes.Text("arial",25,charMessage, colour = (255,255,255))

        except:

            invalidText = Pygame\_Classes.Text("arial",25,"IMPOSSIBLE USERNAME", colour = (255,255,255))

        #Display error message if needed

        if not valid:

            invalidText.createSquare(100,600)

            invalidText.showText(window)

            pygame.display.update()

            time.sleep(3)

        ##Repeat for password

        pswdBox = Pygame\_Classes.textBox(400,50,window.ww/2 - 200, window.wh/2 - 25, "Enter password")

        while not pswdBox.complete:

            window.windowSurface.fill((0,0,0))

            Pygame\_Classes.eventGet()

            pswdBox.checkActive()

            pswdBox.keypressed()

            pswdBox.showTextBox(window)

            pygame.display.update()

        try:

            pswd = pswdBox.text

            valid = True

            for i in pswd:

                if i not in valid\_characters and valid:

                    valid = False

                    charMessage = "You can only use the following characters: "

                    for i in valid\_characters:

                        charMessage += i

                    invalidText = Pygame\_Classes.Text("ariel",25,charMessage, colour = (255,255,255))

        except:

            invalidText = Pygame\_Classes.Text("arial",25,charMessage, colour = (255,255,255))

        if not valid:

            invalidText.createSquare(100,600)

            invalidText.showText(window)

            pygame.display.update()

            time.sleep(3)

        #CHECK USNM AND PASSWORD

        with open("Logins.json", "r") as file:

            data = json.load(file)

        for i in range(0,len(data["Logins"])):

            if data["Logins"][i]["Usnm"] == usnm and data["Logins"][i]["Pswd"] == pswd:

                loggedIn = True

                global userIndex

                userIndex = i

                pswd = ""

        if not loggedIn:

            errorText = Pygame\_Classes.Text("arial", 25, "Username or Password incorrect", colour = (255,255,255))

            errorText.createSquare(100,600)

            errorText.showText(window)

            pygame.display.update()

            time.sleep(2)

def startup(window):

    valid = False

    while not valid:

        #make textbod that asks for Y/N and waits for answer

        checkBox = Pygame\_Classes.textBox(400,50,window.ww/2 - 200, window.wh/2 - 25, "Do you have an account yet Y/N: ")

        while not checkBox.complete:

            window.windowSurface.fill((0,0,0))

            Pygame\_Classes.eventGet()

            checkBox.checkActive()

            checkBox.keypressed()

            checkBox.showTextBox(window)

            pygame.display.update()

        try:

            playerCheck = checkBox.text

            valid = True

            if playerCheck.upper() not in ["Y", "N"]:

                errorMessage = Pygame\_Classes.Text("arial",25,"NOT A VALID OPTION", colour = (255,255,255))

                valid = False

        except:

            errorMessage = Pygame\_Classes.Text("arial",25,"NOT A VALID OPTION", colour = (255,255,255))

        if not valid:

            errorMessage.createSquare(550,600)

            errorMessage.showText(window)

            pygame.display.update()

            time.sleep(2)

    if playerCheck.upper() == "Y":

        login(window)

    else:

        newAccount(window)

def endgame(window):

    with open("Logins.json", "r") as file:

        data = json.load(file)

        if data["Logins"][userIndex]["highscore"] <= score :

            data["Logins"][userIndex]["highscore"] = score

        if score > data["Highscore"][0]["Score"]:

            data["Highscore"].insert(0,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

            data["Highscore"].pop(5)

        elif score > data["Highscore"][1]["Score"]:

            data["Highscore"].insert(1,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

            data["Highscore"].pop(5)

        elif score > data["Highscore"][2]["Score"]:

            data["Highscore"].insert(2,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

            data["Highscore"].pop(5)

        elif score > data["Highscore"][3]["Score"]:

            data["Highscore"].insert(3,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

            data["Highscore"].pop(5)

        elif score > data["Highscore"][4]["Score"]:

            data["Highscore"].insert(4,{"User": data["Logins"][userIndex]["Usnm"], "Score":score})

            data["Highscore"].pop(5)

        else:

            pass

    with open("Logins.json", "w") as file:

        json.dump(data, file, indent = 5)

    pygame.quit()

    quit()

window = Pygame\_Classes.window(1500,700, "COURSEWORK")

window.initScreen()

startup(window)

indexsUsed = []

score = 0

with open("Songs.json", "r") as file:

    songs = json.load(file)

while True:

    ##Generate song

    if len(songs["Songs"]) == len(indexsUsed):

        userMessage = Pygame\_Classes.Text("arial",25,"You appear to have used all our songs",colour = (255,255,255))

        userMessage.createSquare(100, 600)

        userMessage.showText(window)

        time.sleep(2)

        endgame(window)

    songNotUsed = False #Variable that keeps track of if we have found an unused song

    while not songNotUsed:

        index = random.randint(0,len(songs["Songs"])-1)

        if index not in indexsUsed:

            indexsUsed.append(index) #Add index of chosen song to used list

            songNotUsed = True

    song = songs["Songs"][index]["name"]

    artist = songs["Songs"][index]["artist"]

    #edit song name to display to player

    songWords = song.split(" ")

    displaySong = "Song is: "

    for i in songWords:

        displaySong += songWords[songWords.index(i)][0]

        for j in range(0,len(i)-1):

            displaySong += " \_"

        displaySong += "   "

    songDisplay = Pygame\_Classes.Text("arial", 25, displaySong, colour = (255,255,255))

    songDisplay.createSquare(550,200)

    artistDisplay = Pygame\_Classes.Text("arial",25,artist, colour = (255,255,255))

    artistDisplay.createSquare(550,250)

    #Get LeaderBoard and assemble as text

    with open("Logins.json", "r") as file:

        leaderboard = json.load(file)

    leaderboardTitle = Pygame\_Classes.Text("arial",25,"HIGHSCORES",colour = (255,255,255))

    leaderboard1 = Pygame\_Classes.Text("arial",25, leaderboard["Highscore"][0]["User"] + " with a score of " + str(leaderboard["Highscore"][0]["Score"]),colour = (255,255,255))

    leaderboard2 = Pygame\_Classes.Text("arial",25, leaderboard["Highscore"][1]["User"] + " with a score of " + str(leaderboard["Highscore"][1]["Score"]),colour = (255,255,255))

    leaderboard3 = Pygame\_Classes.Text("arial",25, leaderboard["Highscore"][2]["User"] + " with a score of " + str(leaderboard["Highscore"][2]["Score"]),colour = (255,255,255))

    leaderboard4 = Pygame\_Classes.Text("arial",25, leaderboard["Highscore"][3]["User"] + " with a score of " + str(leaderboard["Highscore"][3]["Score"]),colour = (255,255,255))

    leaderboard5 = Pygame\_Classes.Text("arial",25, leaderboard["Highscore"][4]["User"] + " with a score of " + str(leaderboard["Highscore"][4]["Score"]),colour = (255,255,255))

    leaderboardTitle.createSquare(20,10)

    leaderboard1.createSquare(20,55)

    leaderboard2.createSquare(20,85)

    leaderboard3.createSquare(20,115)

    leaderboard4.createSquare(20,145)

    leaderboard5.createSquare(20,175)

    #Get playerscore for display

    displayScore = Pygame\_Classes.Text("arial",25,"Score = " + str(score), colour = (255,255,255))

    displayScore.createSquare(1400, 25)

    #Guess loop

    guesses = 0

    correct = False

    while guesses < 2:

        valid = False

        while not valid:

            guessBox = Pygame\_Classes.textBox(400,50,window.ww/2 - 200, window.wh/2 - 25,"Enter Guess")

            while not guessBox.complete:

                window.windowSurface.fill((0,0,0))

                Pygame\_Classes.eventGet()

                #show current leaderboard statistics

                leaderboardTitle.showText(window)

                leaderboard1.showText(window)

                leaderboard2.showText(window)

                leaderboard3.showText(window)

                leaderboard4.showText(window)

                leaderboard5.showText(window)

                #show score

                displayScore.showText(window)

                songDisplay.showText(window) #Show songname

                artistDisplay.showText(window) #Show artist

                guessBox.checkActive() #Check if user has clicked on guessbox

                guessBox.keypressed() # Take key input

                guessBox.showTextBox(window) #Show guessbox

                pygame.display.update()

            #validate user input

            try:

                guess = guessBox.text

                valid = True

                for i in guess:

                    if i not in valid\_characters and valid:

                        valid = False

                        userMessage = Pygame\_Classes.Text("arial",25,"Invalid guess", colour = (255,255,255))

            except:

                userMessage = Pygame\_Classes.Text("arial",25,"Invalid guess", colour = (255,255,255))

            if not valid:

                userMessage.createSquare(550,600)

                userMessage.showText(window)

                pygame.display.update()

                time.sleep(2)

        valid = False

        guesses += 1

        if guess.upper() == song.upper():

            correct = True

            if guesses == 1:

                userMessage = Pygame\_Classes.Text("arial", 25, "CORRECT, +3 Points", colour = (255,255,255))

                score += 3

            else:

                userMessage = Pygame\_Classes.Text("arial", 25, "CORRECT, +1 Point", colour = (255,255,255))

                score += 1

            guesses = 2

        else:

            userMessage = Pygame\_Classes.Text("arial", 25, "INCORRECT", colour = (255,255,255))

            userMessage.createSquare(550,600)

            userMessage.showText(window)

            pygame.display.update()

            time.sleep(1)

        #Endgame setup

        if guesses == 2:

            window.windowSurface.fill((0,0,0))

            #show the guessbox

            guessBox.showTextBox(window)

            #show leaderboard

            leaderboardTitle.showText(window)

            leaderboard1.showText(window)

            leaderboard2.showText(window)

            leaderboard3.showText(window)

            leaderboard4.showText(window)

            leaderboard5.showText(window)

            #show output message

            userMessage.createSquare(550,600)

            userMessage.showText(window)

            #show song and artist

            songDisplay = Pygame\_Classes.Text("arial",25,song,colour = (255,255,255))

            songDisplay.createSquare(550,200)

            songDisplay.showText(window)

            artistDisplay.showText(window)

            #show score

            displayScore.showText(window)

            pygame.display.update()

            time.sleep(2)

        #Endgame message and activation of engame subroutine

        if guesses == 2 and not correct:

            endgameMessage = Pygame\_Classes.Text("arial",25,("GAME OVER, YOUR SCORE WAS: " + str(score)), colour = (255,255,255))

            endgameMessage.createSquare(550,650)

            endgameMessage.showText(window)

            pygame.display.update()

            time.sleep(2)

            endgame(window)

**Variable list**

|  |  |
| --- | --- |
| **Variable** | **Use** |
| Pygame\_Classes | |
| screenClicked | Boolean, keeps track of if the user is currently clicking the screen as determined by eventGet |
| keyPressed | Array, raw output from pygame saying what exact key has been pressed on the keyboard |
| keyBeenPressed | Boolean, says whether a key on the leyboard has just been pressed |
| Class Window | |
| ww | Integer, window width, exactly what it sounds like |
| wh | Integer, window height, exactly what it sounds like |
| name | String, name displayed as pygame window caption |
| windowSurface | Object, pygame display that all pygame objects are blitted onto |
| Class Text | |
| font | String, font used in text, later, Object, pygame font based of inputted original font |
| fontsize | Integer, size of the font used |
| words | String, the words to be displayed |
| colour | Group of integers, RGB colour e.g. (255,255,255) |
| text | Object, Rendered text by pygame ready to be displayed |
| textRect | Object, prgame rect storing dimensions of text and used to align to correct coordinates |
| Class textBox | |
| prompt | String, prompt shown to user when textbox is empty |
| complete | Boolean, says whether user has pressed return to submit input |
| text | String, user inputted text |
| fontsize | Integer, the size of the font in the textbox |
| font | String, the font used in the textbox, defaults as ariel |
| colour | Group of integers, RGB colour e.g. (255,255,255) |
| width | Integer, the width of the textbox |
| height | Integer, the height of the textbox |
| x | Integer, the x coordinate of the top left of the textbox |
| y | Integer, the y coordinate of the top left of the textbox |
| rect | Object, a pygame rect that shows the box of the inputbox |
| active | Boolean, says whether the user is currently using the textbox |
| x, y | 2 integers, used to store current mouse position when determining active |
| screenText | Object, Text object that stores the inputted text to be shown to user |
| Main | |
| valid\_characters | Array, all allowed characters for user input |
| userIndex | Integer, stores index of user in relation to the main logins file for ease of location |
| loggedIn | Boolean, used to maintain while loop for the login and newAccount subroutines |
| newUsnmBox | Object, textbox used for when the user inputs their username for their new account |
| window | Object, stores all variables about the pygame window |
| newUsnm | String, user inputted new username to be validated and placed into logins file |
| charMessage | String, a message to be displayed to the user that tells the user which characters are allowed |
| invalidText | Object, text to be displayed on screen telling about the user’s input being an invalid string |
| newPswdBox | Object, textbox used for when the user inputs their password for their new account |
| newPswd | String, user inputted new password to be validated and placed into logins file |
| newLogin | Dictionary, new login to be added to logins file comprised of newUsnm and newPswd |
| data | Dictionary, stores full data from whichever file is currently being edited |
| usnmBox | Object, textbox used for when the user inputs their username for their account |
| pswdBox | Object, textbox used for when the user inputs their password for their account |
| usnm | String, user inputted username to be checked against Logins file |
| pswd | String, user inputted password to be checked against Logins file |
| errorText | Object, text to be displayed to user on screen saying that their username or password is incorrect |
| checkBox | Object, textbox used for when the user responds to whether they have an account yet |
| playerCheck | String, the output from checkBox |
| errorMessage | Object, text to be displayed to user telling them that they have not answered a valid option |
| score | Integer, the player’s score |
| songs | Dictionary, stores the entire raw json text from the Songs file |
| userMessage | Object, message to be displayed to screen to tell the user they have solved every song |
| songNotUsed | Boolean, keeps track of if we have found an unused song |
| index | Integer, random number used as index for a song |
| song | String, the songname of currently used song |
| artist | String, the artist of currently used song |
| songWords | Array, stores the individual words of the songname |
| displaySong | String, the edited songname after letters have been removed, is the songname shown to the player |
| songDisplay | Object, the formatted displaySong text to be shown to the player |
| artistDisplay | Object, the formatted artist text to be shown to the player |
| leaderboard | Dictionary, the contents of the Logins file when retrieving the leaderboard |
| leaderboardTitle | Object, the formatted text of the leaderboard title |
| leaderboard1 | Object, the formatted text of the first row of the leaderboard |
| leaderboard2 | Object, the formatted text of the second row of the leaderboard |
| leaderboard3 | Object, the formatted text of the third row of the leaderboard |
| leaderboard4 | Object, the formatted text of the fourth row of the leaderboard |
| leaderboard5 | Object, the formatted text of the fourth row of the leaderboard |
| displayScore | Object, the formatted text of the player’s score |
| guesses | Integer, number of guesses the user has had on current song |
| guessBox | Object, textbox which the user inputs their guess into |
| guess | String, the user’s inputted guess on what the current song is |

**TESTING**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No. | Test Details | Expected Result | Actual Result | Working/Needs fix |
| 1 | Testing Y/N check with values: “y”, “Y”, “n”, “N”, “;”,”a”,”P” for a full analysis of its validation | Values “y”,”Y”,”n”,”N” should be accepted whilst the others should be rejected | “y” = Accepted  “Y” = Accepted  “n” = Accepted  “N” = Accepted  “;” = Rejected  “a” = Rejected  “P” = Rejected | Working Correctly |
| 2 | Testing Account Setup: will attempt to create new account with name “BaRrY” and password “yRrAb” | Account should create successfully and should be visible in the Logins file for easy conformation |  | Working Correctly |
| 3 | Testing Account Setup: will attempt to create account with name “#IHATETRUMP”  And if that succeeds I will use the password “#BidenSuks2” | The username should be rejected for the use of invalid characters before getting to the password which would also be rejected on the same grounds |  | Working Correctly |
| 4 | Testing Account Setup: will attempt to create account with name “IHATETRUMP” and if that succeeds I will use the password “#BidenSuks2” | The username should be accepted but the password rejected on the use of invalid characters |  | Working Correctly |
| 5 | Testing Account Login: I will attempt to login to the account created in test 2 with username “BaRrY” and password “yRrAb” | It should log me in successfully and initialise the game | Accepted | Working Correctly |
| 6 | Testing Account Login: I’ll attempt to login to the account created in test 2 but I will put a capital where every lowercase letter is and vice versa for both the username and the password | Login attempt should be rejected as both fields are case sensitive |  | Working Correctly |
| 7 | Testing Account Login: I’ll attempt to login to the account created in test 2 but I will use the correct username but use “asdf” as the password | Login attempt should fail |  | Working Correctly |
| 8 | Testing Account Login: I’ll attempt to login to the account created in test 2 but I will use the correct password but the username “asdf” | Login attempt should fail |  | Working Correctly |
| 9 | I will login to the account from test 2 and complete the first song with my guess using only lowercase letters | Guess should be marked as correct and 3 points should be added to the score |  | Working Correctly |
| 10 | I will login to the account from test 2 and complete the first song with my guess using only uppercase letters | Guess should be marked as correct and 3 points should be added to the score |  | Working Correctly |
| 11 | I will login to the account from test 2 and fail the first guess before getting the second guess first | Second guess should be marked as correct and 1 point should be added to the score |  | Working Correctly |
| 12 | I will login to the account from test 2 and fail both guesses | Game should end and display leaderboard that currently contains placeholder accounts |  | Working Correctly |
| 13 | I will get to a score of 20 and then fail 2 guesses in a row | Game should end and after logging in again my score should be on the leaderboard |  | Working Correctly |

This report is the product of James Richardson 2020 and many nights of toil, tea and twitch. Please mark generously.