NEA COMPUTER SCIENCE PROJECT: DIGITAL QUIZ SYSTEM

Isaac Patrickson
Candidate No. 7507
Centre No. 39445
Whitburn Church of England Academy

Contents

Analysis

Project problem definition

Client: Helleni Tsentas and other people who enjoy partaking in online quizzes for leisure.

The Problem: To research methods of how quizzes are developed and conducted online, and to design a purpose-built system which optimizes the process of creating, taking, and monitoring quizzes. The system should be user-friendly, and less time consuming when conducting a quiz.

Background to the problem

During the Lockdown of 2020, quizzes became a popular pastime to those confined to their own homes. Using the video communications app “Zoom”, families and friends could virtually meet up and enjoy each other’s company. During these video calls, members of the call could host and take part in virtual pub quizzes as a form of entertainment.

These quizzes were displayed through Zoom’s screenshare function. This allowed the participants to see exactly what the host wanted them to see. Since there was no purpose-built software to handle quizzes, a quiz host would create quizzes on PowerPoints, Word documents and sometimes paper. This meant that the act of creating a quiz was not well optimised and was quite time consuming for the host.

The most popular method of participating in these quizzes relied on each participant writing their answers as a text message or on a piece of paper. The problem with this, is that once their answer was submitted, it could have easily been changed by the time the answers were read out. As well as this, the final submission of the answers via paper involved the participant sending a picture to the host. Submitted Answers were often scattered across multiple platforms which made it harder for the host to mark. In conclusion, this method is subject to cheating and can prove itself to be very time consuming when creating and marking.

Interview with Helleni Tsentas – A Representative of the Primary Client

What was your method for creating quizzes?

- I decided how many rounds and the topics for each round.
- I would decide how many questions I wanted per round, this would depend on how long I wanted the quiz to be and how many rounds I had selected. For example, if I had lots of rounds I might choose fewer questions per round, so the quiz wouldn't be too long.
- I would use the internet to research questions for each of my rounds.
- I would then use PowerPoint adding the questions to slides and using the formatting tools to create the look for each round.
- I added each question in the appropriate format. For example, there would be simple text-based questions, image questions where I would use a graphic with a transition to reveal an image slowly, music questions where I would insert an audio file, and for some I would insert video clips.
- After each round of questions, I would repeat the question slides but with the answers to each questions added. Normally I used a reveal function to bring up the answer to each question individually.

What were the benefits of using this method?

- Complete flexibility.
- By researching questions and using PowerPoint for the visuals, it allowed creative freedom and complete flexibility.
- Each quiz could be completely bespoke, different rounds, different questions, different format.

What were the drawbacks of this method?

- It was very time consuming.
- It took time to find appropriate questions and it took even longer to create the quiz in PowerPoint and do all the formatting.
- Had to mark the quizzes manually.

Which features would be the most important to you in quiz making software?

- An easy to use.
- Visually appealing user interface.
- Library of preloaded quiz questions that could be sorted by Topic so I could easily make multiple rounds.
- Option to add your own questions.
- Automatic marking.
- A leader board would be nice as it creates the element of competition and can be used to track progress.

Which existing features do you find the most useful?

- Leader board.
- Auto marking.
- Stores quiz question.

How would you like the participants of the quizzes to answer the created questions?

- Via an app on their phones or on a computer.

Do you think that a separate, purpose-built digital system is ideal for quiz creating/taking?

- Yes, a purpose built 'Quiz Builder' system would certainly make the process of creating quizzes much easier and quicker.
- It would allow you to create a quiz at short notice. It took me days to create a quiz with PowerPoint.

Observation of Current Methods

As well as recalling from past personal experience, I observed the current methods used by Lecturers, teachers, and family members to get a better understanding of how participants interact with and create online quizzes and identify the problems they face when doing so.

Due to privacy issues, the screenshots below are examples rather than a real scenario.

The most popular technique is as follows:

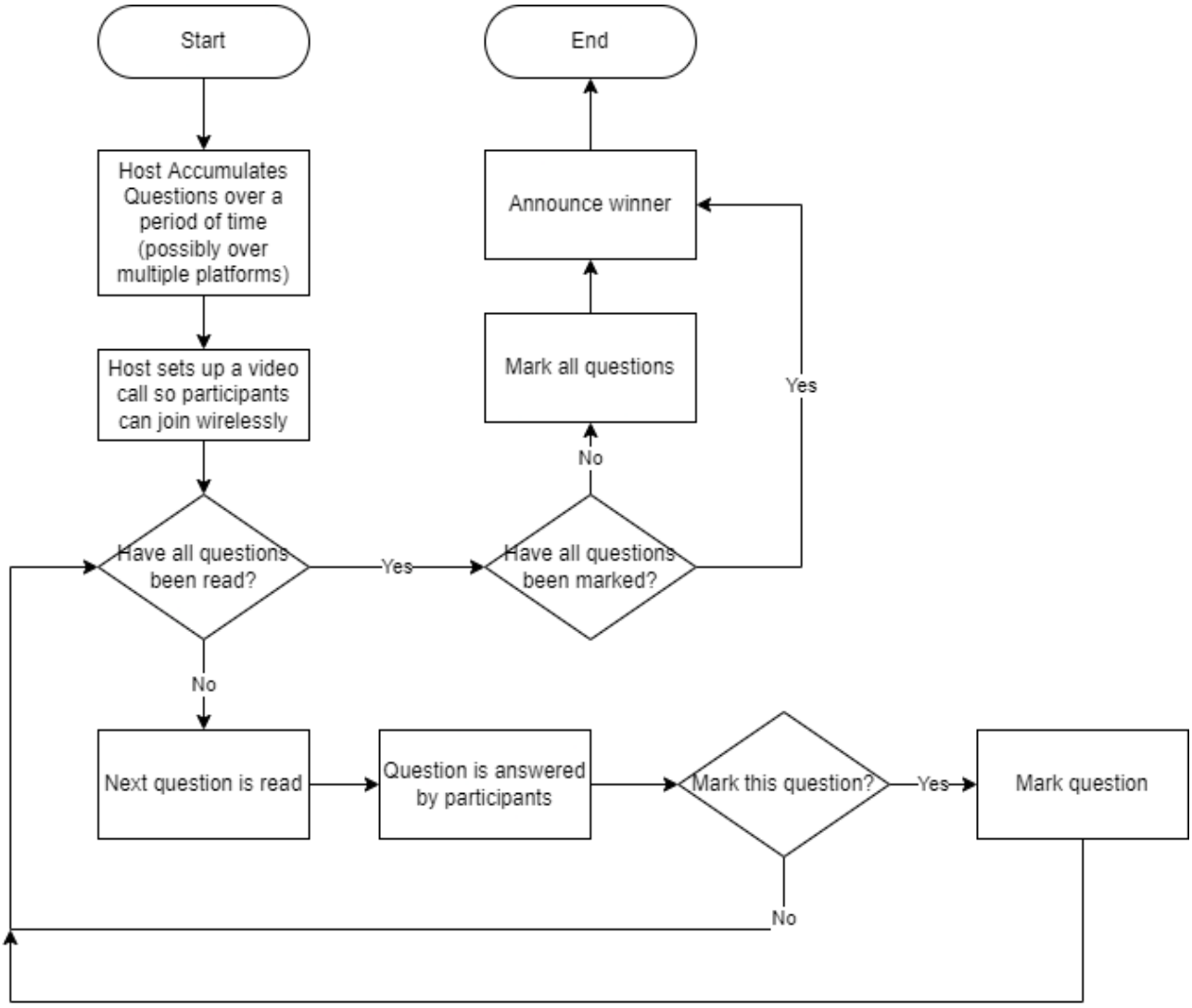
1. Questions are accumulated over a brief period
    a. The Host is likely to have a notebook and/or digital note keeping app in which questions are recorded throughout the period.
    b. In most cases, family members or co-hosts of the quiz contribute to the process by submitting question through text or word-of-mouth.
    c. This means questions were often spread across multiple platforms which can increase confusion for the Quiz Host when the quiz creating process begins.

2. Questions are then read out in a video call
    a. The quiz host sets up a video call where participants can join
    b. Depending on the video calling software, calls can be limited. When using the video calling software "Zoom," calls would be restricted to 40 minutes long unless the host pays the subscription fee
    c. This means that some participants might have to download/pay for software. If participants are not technically minded this could take effort
    d. The host or participants could drop out the call if there Wi-Fi connection is not stable.

3. Questions are answered and submitted through multiple methods
    a. After the host has read out the question, the participants answer by writing it down in a notebook or sending the answer to the host via text message
    b. Answers are usually submitted at the end and not after each question. Although this minimizes cheating, it would be very time consuming to check the answers of each participant after each question. This would slow the pace of the quiz and reduce enjoyment. When answers are submitted at the end, they are either sent to the host to mark them, or self-marked if the host reads the answers out.
    c. This method of marking all questions for all participants at the end can prove to be quite the workload and can leave the marking prone to mistakes due to the host trying to mark as quickly as possible. As for the other method, this produces the highest chance of cheating. Participants can change their answer to the correct one last minute to increase their chances of winning. This method purely relies on trusting each participant to be fair and a strict marker.

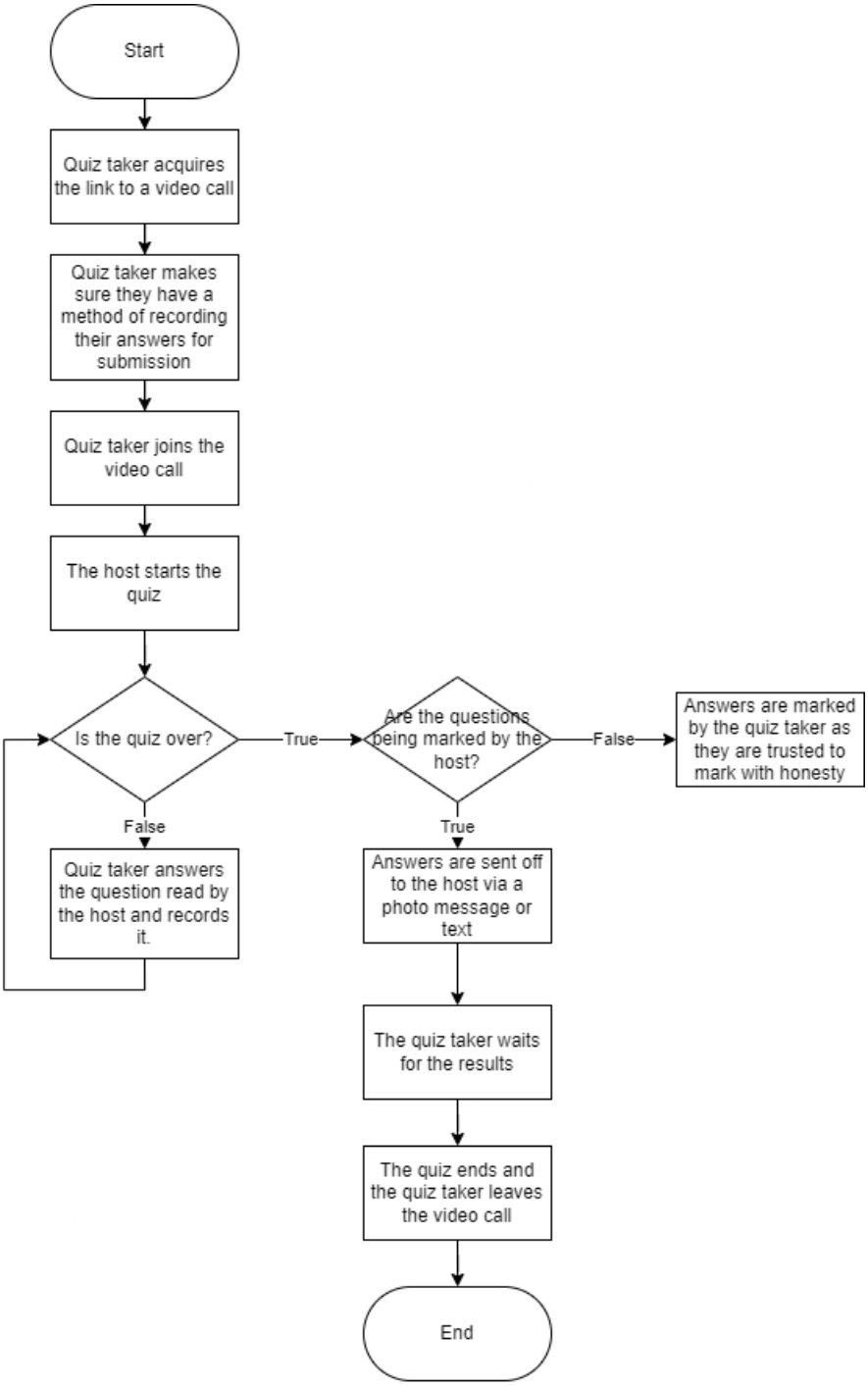
4. After questions have been marked, scores are tallied up and a winner is announced
    a. A number system (1st, 2nd, 3rd, 4th, etc.) is used to rank each participant
    b. If this were a quiz set in a lesson by a teacher during online school, ranking each student manually would consume valuable time
    c. Unless the participant records it themselves, there is usually no record of how many quizzes they have won or how many questions they have answered correctly
    d. This can cause the participant to lose interest in future quizzes as their efforts are not put towards anything
    e. This also means if this is used for educational purposes, there would be no true representation of the student's ability in the area that quiz covered

Flowchart of the researched quiz creating method



This flowchart models a generic system of creating and hosting online quizzes for recreational purposes. Most of the processes in this diagram do not rely on technology and therefore human error can arise. This can offer a lot more freedom when it comes to question types, points, and announcements. However, it can often create a larger workload for the host when carrying out these tasks.

Flow chart of the researched quiz taking method



This flowchart models a generic system for joining and taking part in an online quiz, hosted via an online video calling software. Most of the processing in this diagram also don't rely on technology, so the same issue of human error is present.

When it comes to designing my program, I will not be incorporating any video calling software. However, there are no restrictions for the user, and I don't see why my program and video calling software cannot be used simultaneously.

Prospective Users and Acceptable Limitations

This project is aimed at various groups and demographics, teachers, students, and quiz enjoyers. The primary user for this program will be people who enjoy taking part in online quizzes for leisure. Most online quiz takers, unless they are just discovering this pass-time, are computer literate which means their skills should not limit the complexity of the program. However, if a user is less technically minded this should not affect the program as ease-of-use and validation will be a priority.

Although students and teachers are not my primary clients, they could be affected if teachers decide to use the software to set homework or as a short warm-up task, so the question types should be like those seen in multiple-choice questions or short 1-mark exam questions. If the questions were longer, students could lose interest or struggle to recall substantial amounts of information.

The limitations to the system are as follows:

- Hardware and software constraints – Most online quizzes that occurred over lockdown took place on a family computer. Certain devices such as older laptops or phones were rarely used. However, my program will utilize databases created with MySQL. This means MySQL must be installed on any device which wishes to use this service. Helleni Tsentas is in possession of a modern functioning laptop which has the capacity to install MySQL.
- My Skills and knowledge – I have no experience in MySQL or how to connect MySQL to python. This is something I will have to teach myself so I can reach this project's aims. However, I do have extensive knowledge in programming with python so once connections have been made, I will be able to manipulate them into whatever form I'd like. As for the user, they would not need any prior coding knowledge in python or MySQL to operate the program. Only the knowledge to install MySQL is required.
- Quiz constraints – To make these quizzes dissimilar to full exam papers, I am limiting each question to be either multiple-choice or open-ended. Open-ended answers are more flexible, however each word must match for the answer to be correct, so short answers are advised. These short, snappy questions complement the 'Pub Quiz' element of the software.
- Time constraints – I must complete the program by Easter of 2022

Evidence of current method

Here are some screenshots and photographs of different stages of the current method

All screenshots are examples of PowerPoints or word documents that were displayed through the video communications software 'Zoom.'

Short quizzes with fewer rounds tend to be simpler and quicker to make. They were usually made in a Microsoft word document

There are 5 rounds, total number of points available 100

Round 1 - Picture Round – name the actor and the character, 20 points available

Round 2 – Brand Logos – name the brand - 20 points available

Round 3 – Sports with a difference – points available 5

1. What sporting event is held annually at Cooper's Hill, Gloucester? **– Cheese Rolling** **1**
2. Which country does the annual face slapping championship take place? – **Russia** **1**
3. How tall is the smallest mobile man in feet and inches? – **2ft 2.41 in** **1**
4. What is the record number of stairs climbed by a person balancing only on their head? – **36** **1**
5. What is the record time of the fastest 100m on a space hopper (female)? – **38.22sec** **1**

Round 4 – Music Round – Name the artist, Song and Decade it was released – 45 points available

Round 5 – Paranormal – 5 points available

1. What is the name of the apparition that is said to haunt Hilton Castle? – Card lad 1
2. What is the girl's name that gets possessed by the demon Pazuzu in the exorcist? – Regan MacNeil 2
3. What's the name of the seventeenth-century merchant ship said to haunt the high seas? – **The Flying Dutchman** **1**
4. What are the full names of the 4 ghost busters in the original film **– Peter Venkman, Ray Stanz, Winston Zeddemore, Egon Spengler.** **4**
5. What is the name of the ghostly character that John Cleese plays in the Harry Potter films? – **Nearly Headless Nick** **1**
6. What 2 North East locations where used in the filming of Harry Potter and the Philosopher's Stone? – **Alnwick Castle and Durham Cathedral.** **2**

TOTAL POINTS AVAILABLE – 20 + 20 +5 + 45 + 10 = 100

Each round is declared.

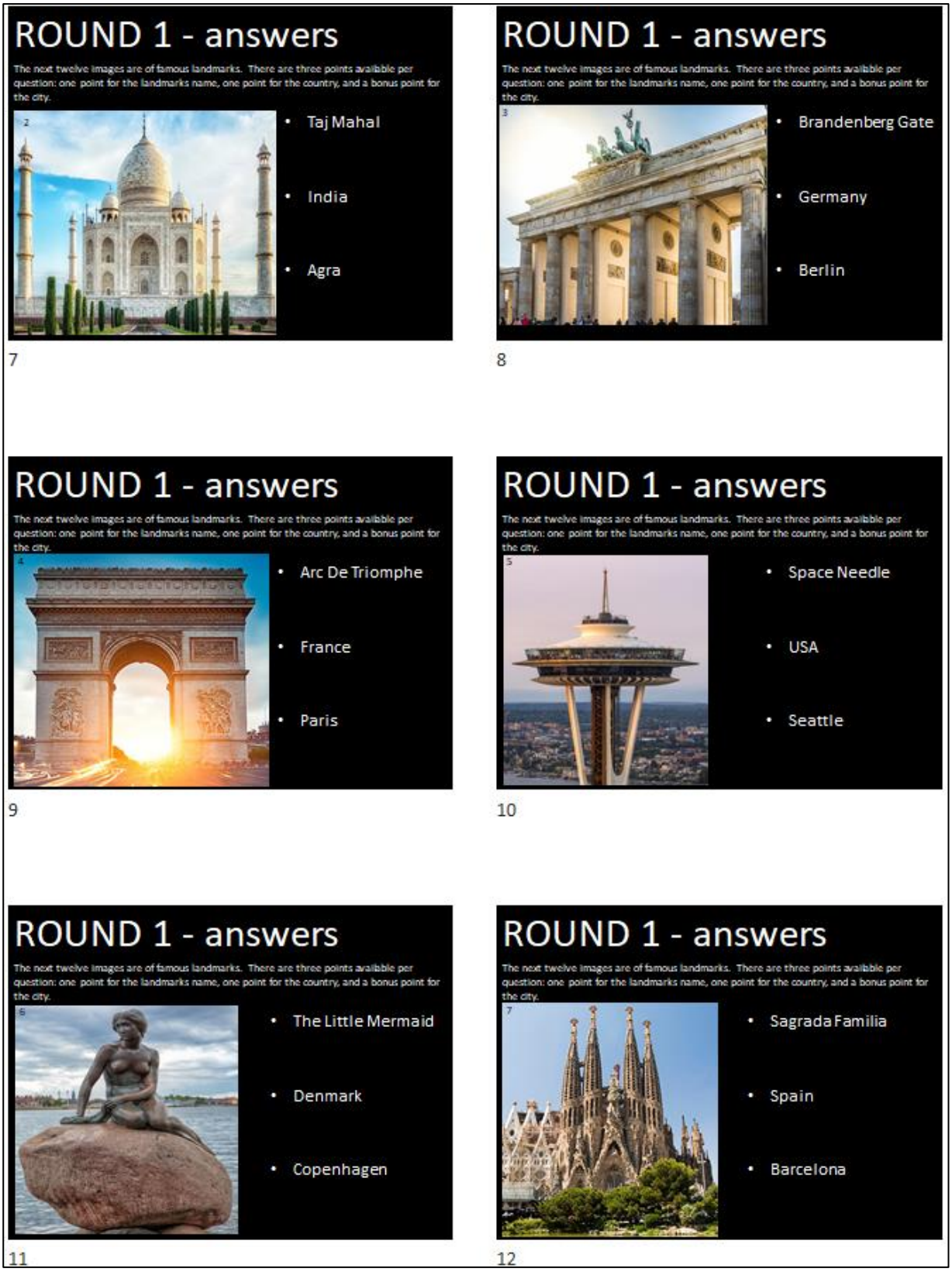
This is an example of a picture round. The quiz creator has used screenshots and a table in Microsoft word to display the images.

Longer quizzes are more complicated and are more time consuming to develop. They were usually made in PowerPoint.

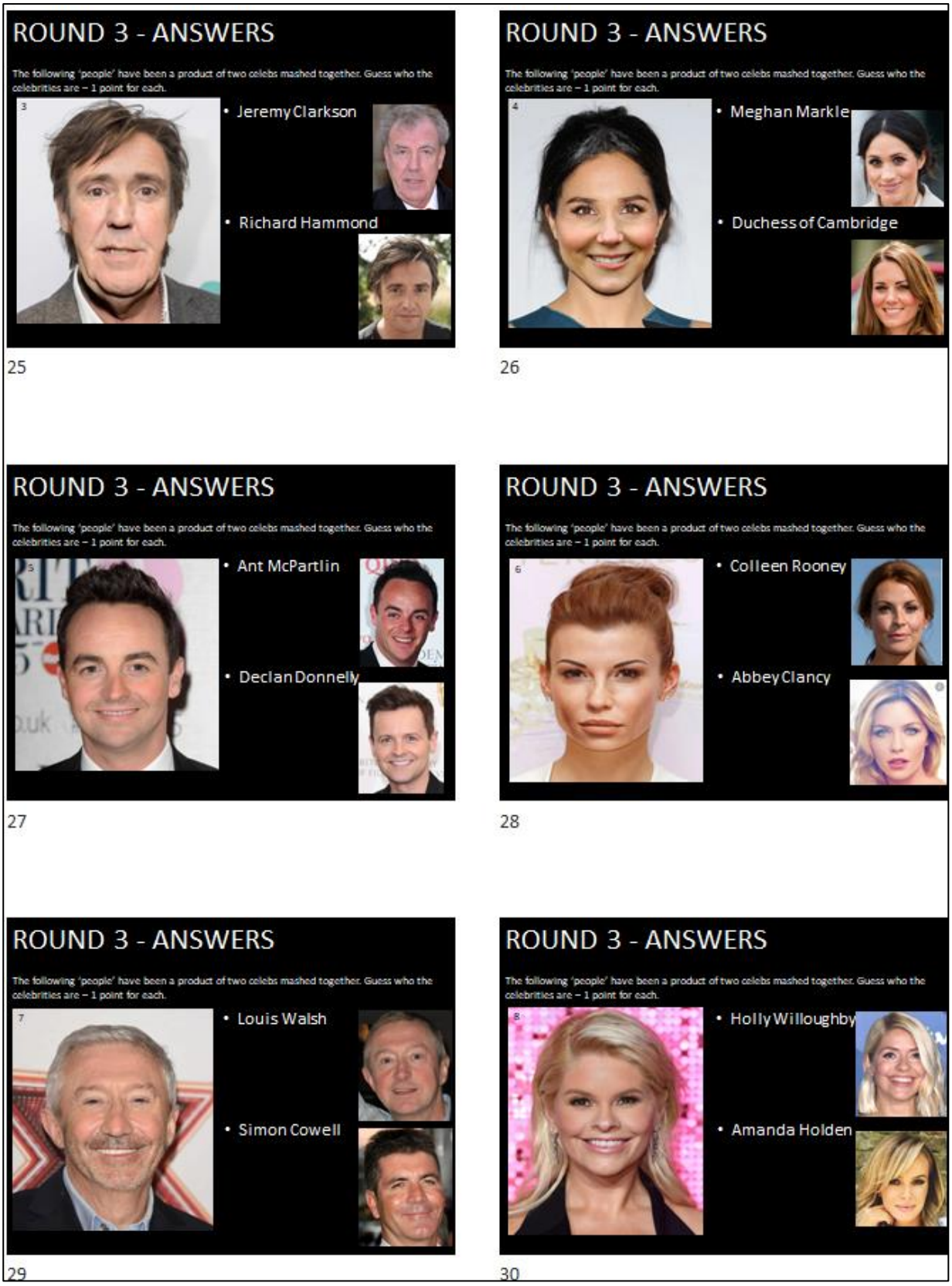
The more questions a user chooses to have, the more work needs to be done when marking.

If the user wants to make the quiz more visually appealing; images, fonts, animations, and sound can be added. This can give the quiz more character and increase the level fun experienced by the quizzers. However, the drawbacks of this are that adding more content to the quiz, takes more time and effort.

Answers can also be revealed after each round. This process relies on the quizzer being trusted to mark their own work fairly.

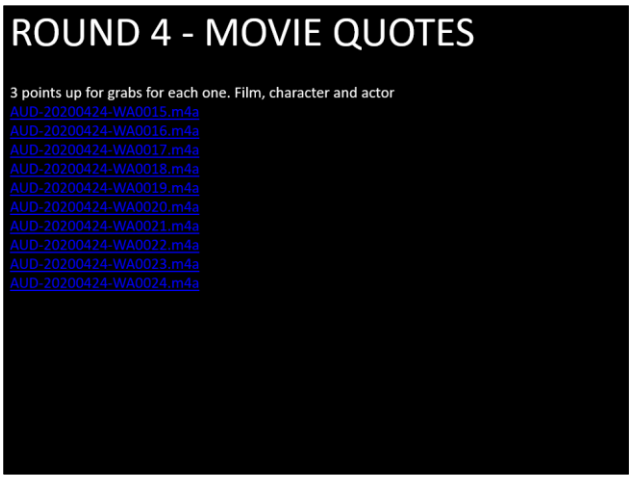


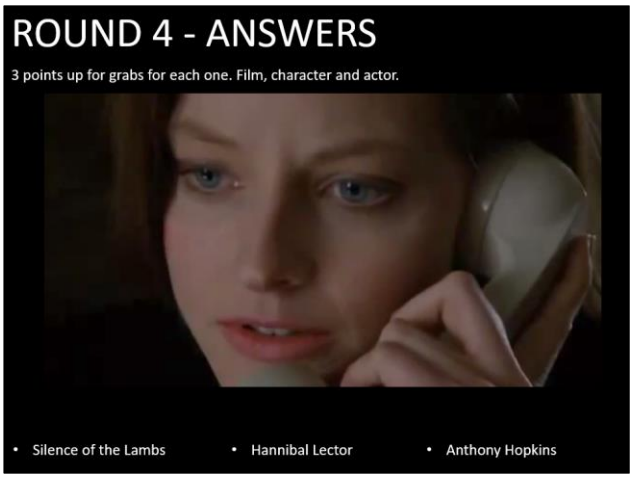
Picture rounds were commonly used to add variety to the quizzes. They were generally easier than the other rounds and could often amuse quizzers with strange visuals.



Movie trivia rounds were usually accompanied by video clips. The clips were shown first, and then the question was read out. In theory, video clips in questions sounds like a good idea, but in practise, it was common for a quizzer to miss the clip entirely due to an audio/visual issue or unstable connection to the zoom call.

Clips were stored as files and were linked to the PowerPoint. This gave the quiz host a reliable way of storing and accessing clips. This saved time as it meant the quiz host would not need to search their file explorer for the previously downloaded clips





After all rounds are complete and all answers are submitted and marked, the results are read out by the host. If the host has allowed the quizzers to mark their own answers, the quiz host would ask for their total score at the end. If the host must mark, the results would be read from an excel spreadsheet.

	QU	ANSWERS	POINTS					
ROUND 1 - LANDMARKS	1	Chichen Itza - Mexico - Yucatan	3					
	2	Taj Mahal - India - Agra	3					
	3	Brandenburg Gate - Germany - Berlin	3					
	4	Arc de Triomphe - France - Paris	3					
	5	Space Needle - USA - Seatle	3					
	6	Little Mermaid - Denmark - Copenhagen	3					
	7	Segrada Familia - Spain - Barcelona	3					
	8	CN Tower - Canada - Toronto	3					
	9	St Basil's Cathedral - Russia - Moscow	3					
	10	Forbidden City / tianeman square - China - Bejing	3					
	11	Christ the Redeemer - Brazil - Rio De Jinero	3					
	12	Acropolis/parthanon - Greece - Athens	3					
		TOTAL	**36**	**0**	**0**	**0**	**0**	**0**
ROUND 2 - ALI	1	Anne Bolyn	1					
	2	Nigeria	1					
	3	Adam Sandler	1					
	4	Remington	1					
	5	Nitrogen	1					
		TOTAL	**5**	**0**	**0**	**0**	**0**	**0**
ROUND 3 - CELEB MASHUP	1	Rosie Huntington-Whiteley and Cara Delevingne	2					
	2	Madonna and Lady Gaga	2					
	3	Jeremy Clarkson and Richard Hammond	2					
	4	Meghan Markle and the Duchess of Cambridge	2					
	5	Ant and Dec	2					
	6	Coleen Rooney and Abbey Clancy	2					
	7	Louis Walsh and Simon Cowell	2					
	8	Holly Willoughby and Amanda Holden	2					
	9	Louis Walsh and Simon Cowell	2					
	10	Angelina Jolie and Jennifer Aniston	2					

Requirements

The requirements, requested by the client, that my program needs to fulfil are as follows

- A user interface which is easy to operate and visually appealing
- An account which can save information between sessions. For example, user data, quiz points earned, position on the leaderboard, and quizzes taken
- A database that already contains quizzes for the user to take
- A method of creating quizzes
- A method to delete quizzes in case there is a mistake made in the creation process
- A way to access a leader board of all players who have taken a specific quiz

Objectives

- Install the free MySQL package containing the MySQL workbench
- Learn the fundamentals of the SQL
- Create a MySQL database which the user can connect to using their MySQL user
- Establish a connection between the computer and a MySQL database
- Design the architecture behind the database (table interconnectivity, primary keys, and foreign keys
- Connect Python 3.9 to the MySQL database and manipulate the database via queries
- Create a system where the user can create a profile and login to an account
- Use a hashing algorithm to encrypt each user's password incase the tables get hacked and data there is a data leak
- Provide an option to take a quiz where the user can select the quiz they want to take
    - The system must store all the relevant data about each take a user completes, i.e., who took the quiz, which quiz did they take, the score they achieved, the title of the quiz they took.
- Provide an option to create a quiz
    - The user can create three types of quizzes. Multiple-choice, open-ended, or mixed
    - The user can create a set number of questions within the quiz
    - Depending on the question type, the user can create the appropriate answers for each question
    - Any other relevant information about the quiz (quiz host, title, max score, and description) should be stored in the database
- Provide an option to delete a quiz from the database
    - Users can only delete quizzes they have made
- Provide an option to view a user's score
- Provide an option to view other user's scores via a leaderboard
- Every user input must be validation to prevent errors in the system and make my program more robust

Examples of online quiz services

As well as the homemade method of creating a quiz for friends and family and displaying them via video streaming software, there are other ways to create and take part in quizzes online.

Online quizzes are a popular form of entertainment and a common type of eLearning. Educational quizzes are one of the most common eLearning patterns for many online courses. Some companies and schools use online quizzes to educate their employees or students. Popular websites for this purpose include Quizlet, Revision Quiz maker and Kahoot!

Many online quizzes are set up to test knowledge or identify a person's attributes. Some companies use online quizzes as an efficient way of assessing a potential hire's knowledge without that candidate needing to travel. Online dating services often use personality quizzes to find a match between similar members.

Most online quizzes are to be taken lightly. The results do not often reflect the true level of understanding in a subject. They are also rarely psychometrically valid. However, they may provide a reflection and function as a springboard for a person to explore ways in which they can improve on a subject.

An example of a popular online quiz service is the game-based learning platform “Kahoot!” It is commonly used in school and other educational institutions. Its learning games, “kahoots,” are user-generated multiple-choice quizzes that can be accessed via a web browser or the Kahoot app. Kahoot! Can be used to review a students’ knowledge, for a formative assessment, or as a break from traditional classroom activities. Kahoot is also a commonplace for trivia quizzes about movies, music, and popular culture.

It was designed for social learning, with learners gathered around a common screen such as an interactive whiteboard, projector, or a computer monitor. The site can also be used through screen-sharing tools such as Skype or Google Hangouts. The gameplay is simple; all players connect using a generated game PIN shown on the common screen and use a device to answer questions created by a teacher, business leader or other quiz creator. These questions can be changed to award points. The creator can choose whether the players can get 0 points, up to 1000 or 2000 points. The points the player gets are calculated on how long it takes them to answer. The sooner they answer, the greater the points if they answer correctly. The player can also get a streak, meaning they answered more than one question correctly in sequence. The longer their streak is, the more points they get when answering a question correctly.

Kahoot can be played through different web browsers and mobile devices through its web interface. It can also be played through the downloadable application version.

Groups of students taking part in a Kahoot displayed on a projector

A Kahoot! Creator can use different question types. The first being a form of multiple-choice which requires a question and at least two options, one of which must be marked as the right answer. The second being an open-ended question, which requires a question and one answer. The player must type in the correct answer to get points. A puzzle question type is also available. Puzzle allows the player to align four options in order, which the creator sets as correct. For example: Align the countries by population from the least populated to the most populated.

At the end of the game, there is an animation of the three best players appearing on the winners' podium. The players can then rate the Kahoot based on their experience.

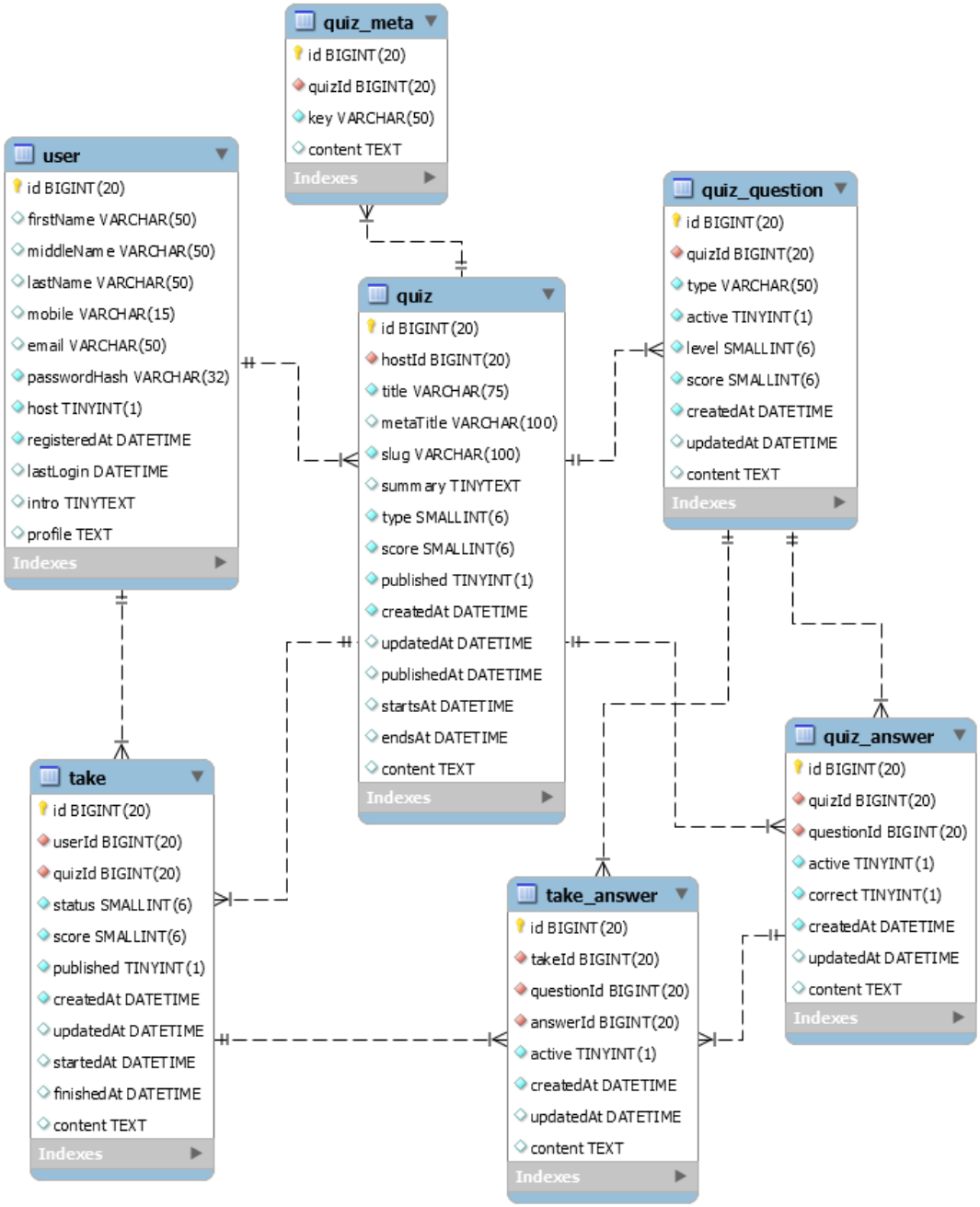
Modelling inspiration from example online quiz service

Taking inspiration from “Kahoot!” I will be incorporating some similar features in my project.

Features of my Program

- All the points a user earns during a quiz are accumulated at the end and their score for that take is shown.
- A podium will not be shown at the end of the quiz. Instead, a leader board can be viewed through a separate menu whenever the user chooses.
- Question order can be randomized or in order depending on the quiz takers choice.
- Only multiple-choice or open-ended question types will be available.
- The test will not be timed, and extra points will not be rewarded for completing the quiz under a limited time.
- Each question will award the user with one point.
- A players total points accumulated can be viewed through their profile

Modelling of a typical online quiz service



	Many-to-one
	One-to-many

If I choose to solve the problem using a database, the diagram above identifies the relationships between each table. A User can do many takes of many quizzes. Each quiz can contain many questions and each question can have many answers. Each time a user takes a quiz, they can submit many answers during that take.

Tasks to be Computerized and the benefits of Computerizing

Tasks to be Computerized

- Creating a quiz
- Taking a quiz
- Quiz deletion
- Keeping a record of results on a particular quiz
- Keeping a record of the user's score

Qualitative Benefits

- Creating the quiz follows the same formula each time, so no confusion surrounding the process is created
- A more user-friendly interface
- Each step will be instructed, so even people who are unfamiliar with computers can create a quiz
- When wanting to randomize question order, it is significantly easier to let a program do it for you than to do it manually

Quantitative Benefits

- It makes it possible to have infinite participants. Theoretically, if the system had enough storage, an infinite number of people could make an account
- Theoretically, without the limitations of storage, users could take an infinite number of quizzes and create an infinite number of quizzes

Design

Inputs, Outputs, Processing and Storage

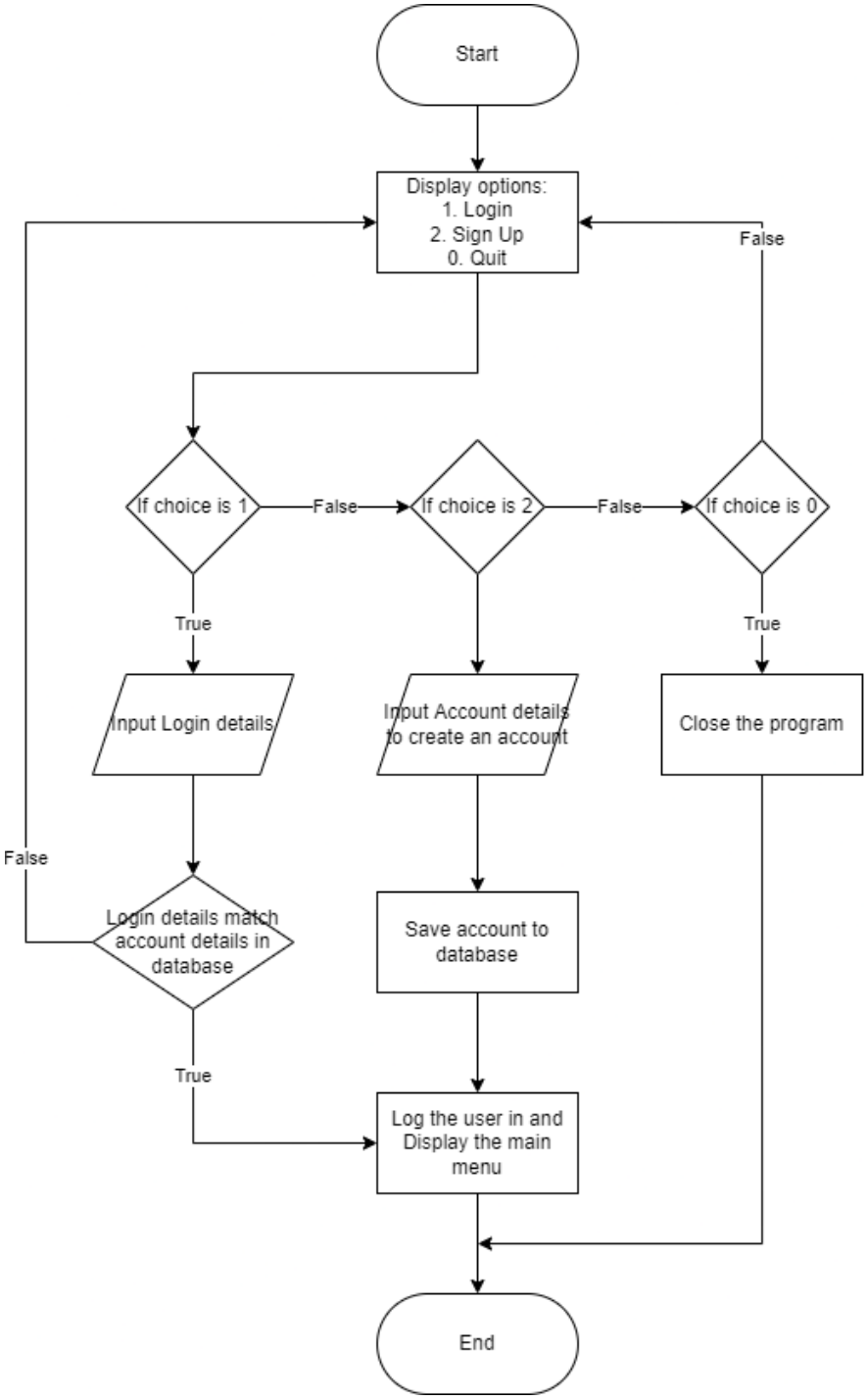
This table outlines what happens to the data put into my system at the most basic level.

Input	**Process**
User • First Name • Last Name • Email • Password Quiz • Title • Type • Number of questions • Description Quiz Question • Difficulty level • Type • Score • Content Quiz Answer • Correct • Content Take • Select Take ID Take Answer • Enter you answer	Creating a User or Logging In • Check email has not been used before. If it hasn't, create the account. If it has, inform the user • If the email hasn't been used and all other fields of input are valid, create a salt for the user. Use this salt and their password to hash their password • Permanently store the user's data in the "User" table in the Database and then log the user in • If the User Decides to log in, their email and password are required • If the email entered is in the database table, retrieve that users salt. Use the entered password and the salt to hash the password. If this results in the same value stored in the table, then the correct password has been entered. • Log the user in Creating a Quiz or Taking a Quiz • If the user wants to create a quiz, ask for a Title, the type of quiz (the user will select one of 3: Multiple-choice, Open-ended, or Mixed), the number of questions and a brief description of the quiz. • Then for the number of questions they wished to create (maximum 50), ask for the: • Difficulty level (1 easy, 3 hard) • Type would be set to either 1 (multiple-choice) or 2 (Open-ended) unless its mixed. If the quiz is mixed, the user will be able to choose each question type as they go along. • The score will be set to 1 as I will be using this to count the max score and count the number of questions in the quiz • The question contents • The questions' correct answer. If the question is multiple-choice, the question will require wrong answers too. • When taking a quiz, inputs for answers will be required from the user and their input will be compared against the correct answer. If they are the same, then the participant has answered the question correctly
Storage	**Output**
• A Database schema called "quiz" • User table • Quiz table • Quiz Question table • Quiz Answer table • Take table • Take Answer table	• Python Shell Window

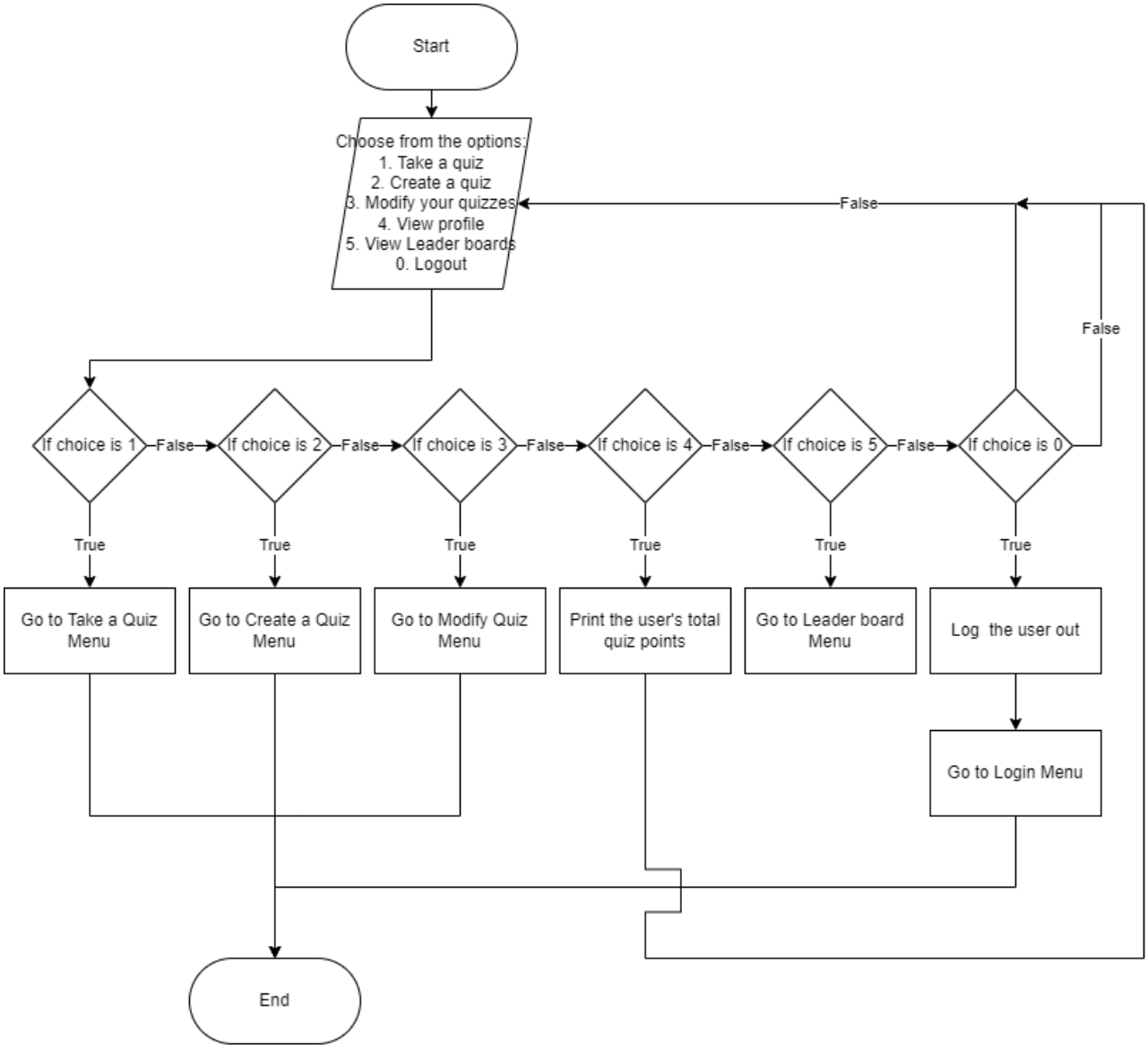
System Flowcharts

The following charts show how different sections of my project operate and interact with the database. Although the flowcharts are denser than that of the current system, most operations are automated, and the processes are less time consuming than the manual alternative.

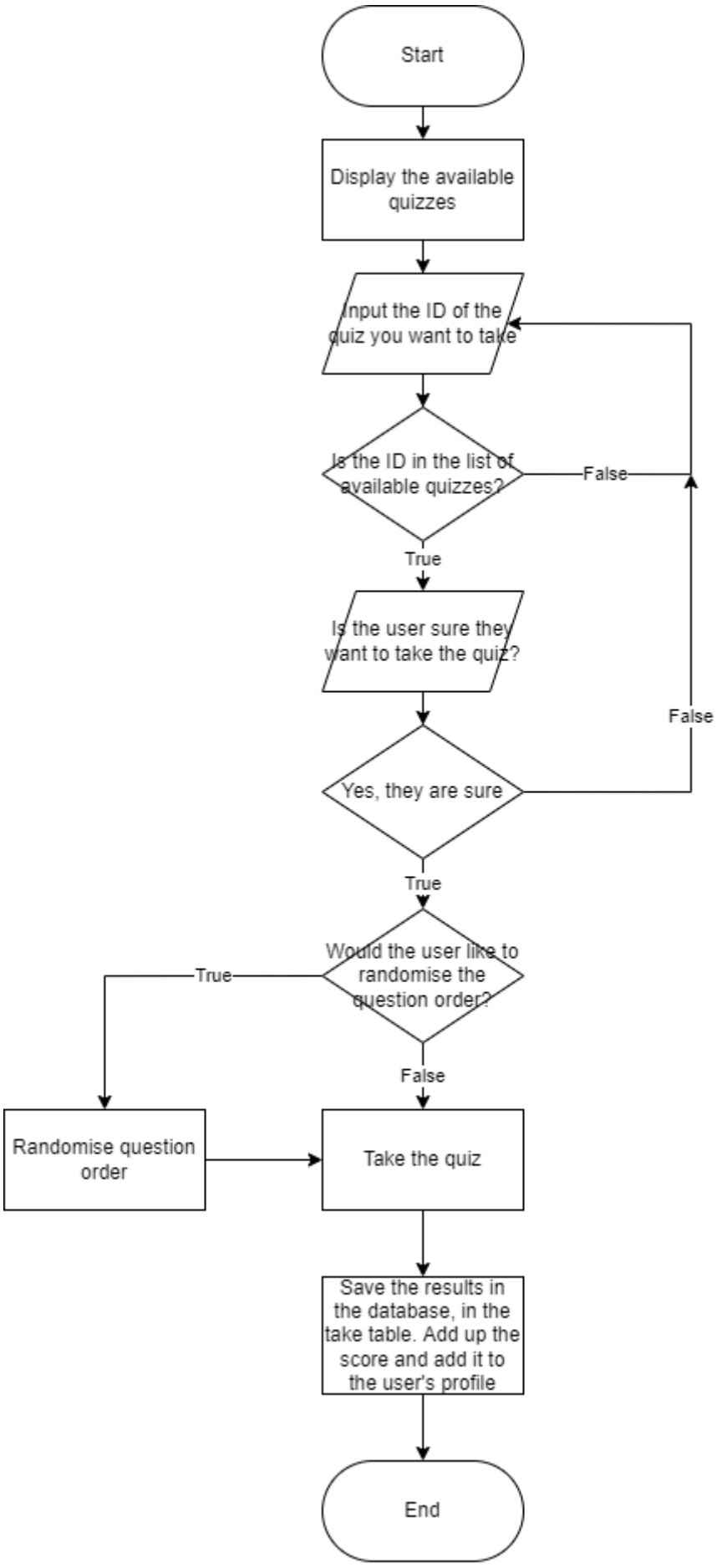
Login Menu



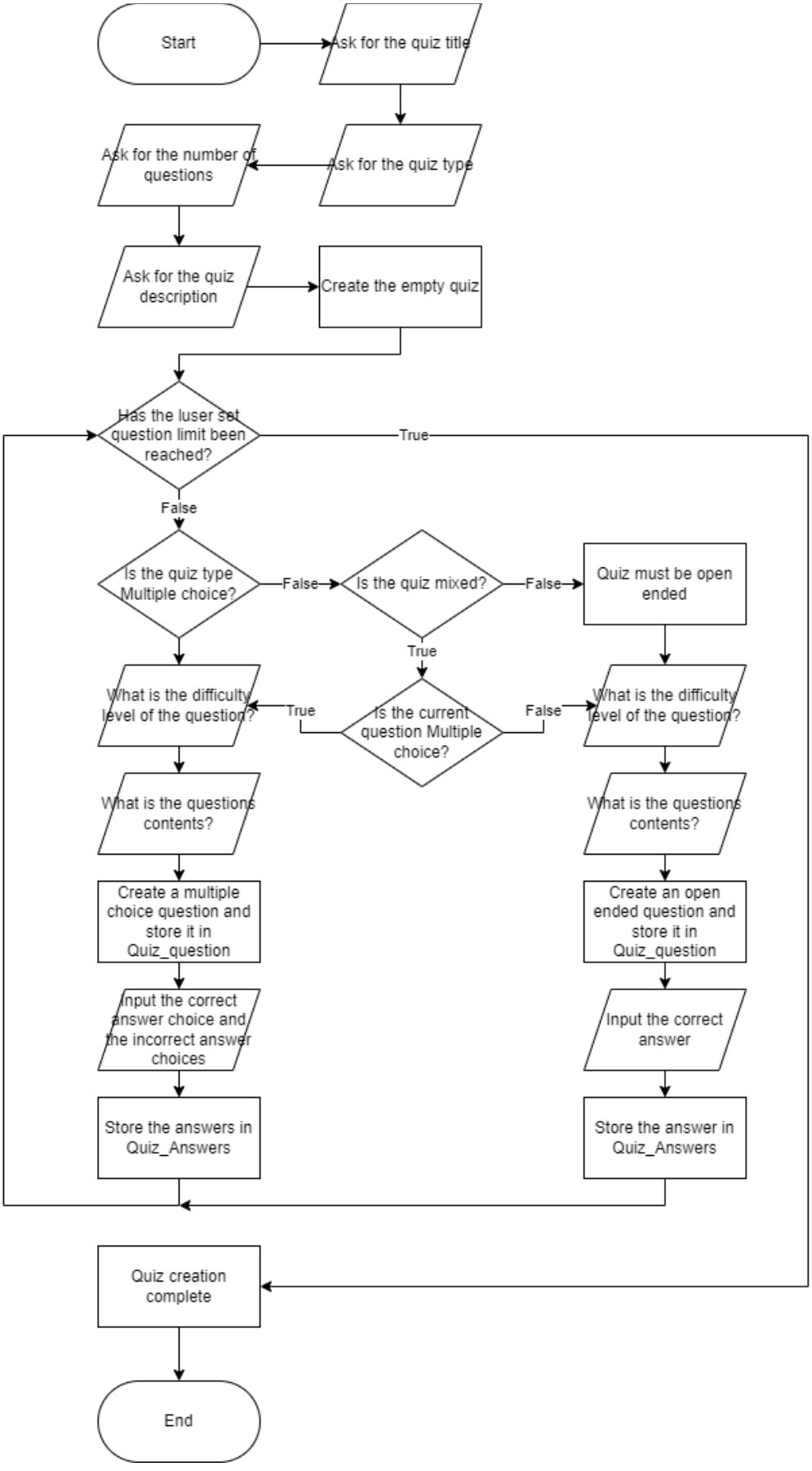
Main Menu


Start
Choose from the options:
1. Take a quiz
2. Create a quiz
3. Modify your quizzes
4. View profile
5. View Leader boards
0. Logout
False
If choice is 1
False
If choice is 2
False
If choice is 3
False
If choice is 4
False
If choice is 5
False
If choice is 0
False
True
True
True
True
True
True
Go to Take a Quiz Menu
Go to Create a Quiz Menu
Go to Modify Quiz Menu
Print the user's total quiz points
Go to Leader board Menu
Log the user out
Go to Login Menu
End

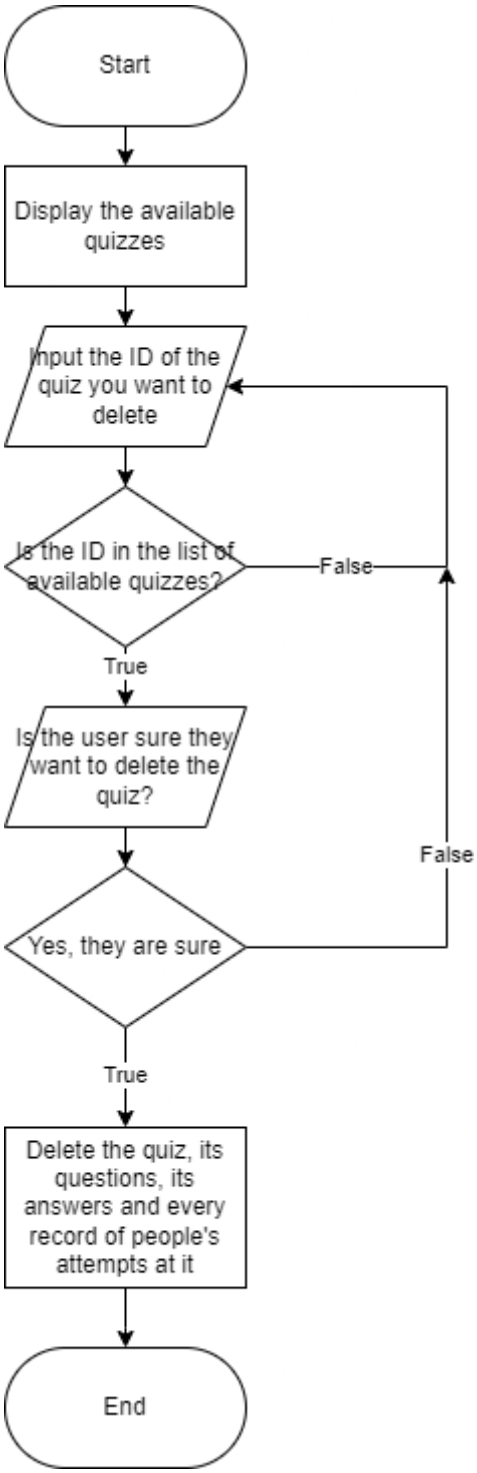
Take a Quiz Menu



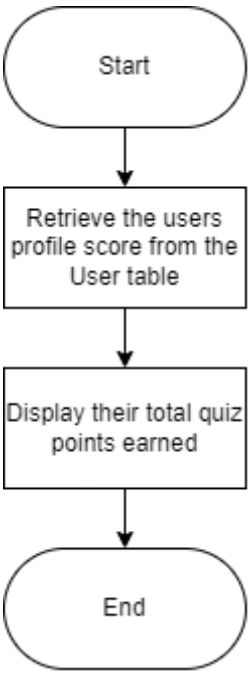
Create a quiz menu



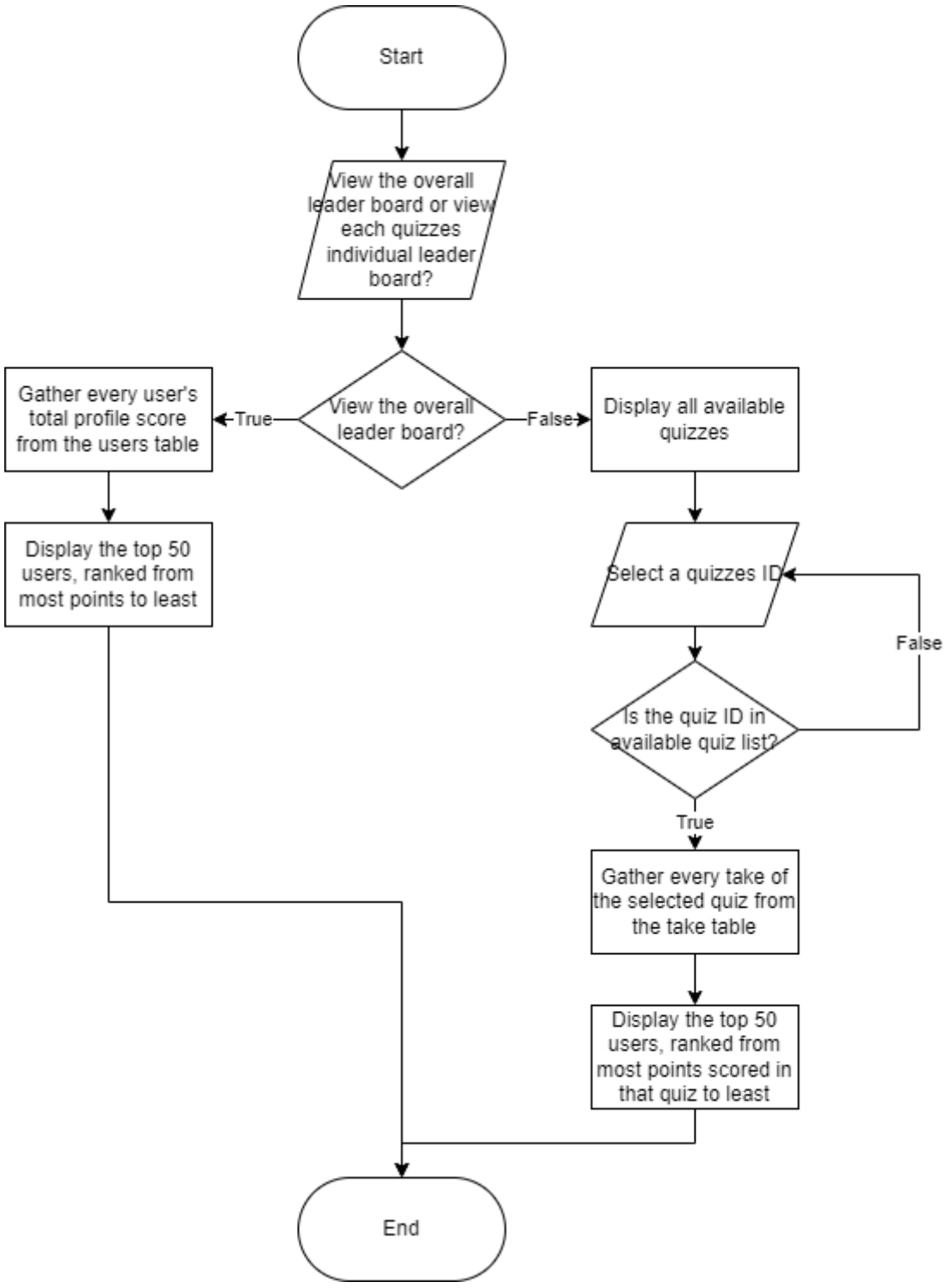
Modify quiz menu



View profile

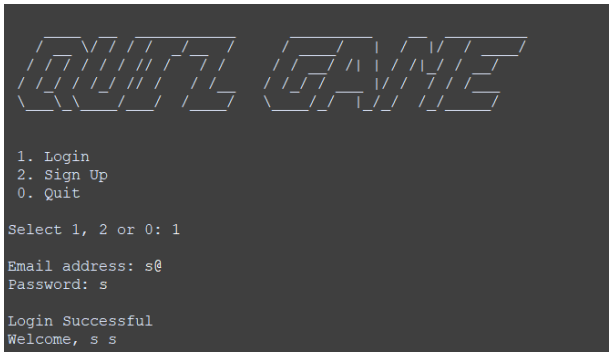


Leader boards menu



User Interface

The user interface for my program will be displayed in the python shell. The screenshots below show the menus I created and some small sections of the program working. I programmed the menus first so I could have a brief checklist of different functionalities. This checklist helped me prioritize different sections of the program and helped me choose my starting point.



```
 1. Login
 2. Sign Up
 0. Quit

Select 1, 2 or 0: 1

Email address: s@
Password: s

Login Successful
Welcome, s s
```

This is the opening menu the user is presented with once they run the program. The user has a choice of "1", "2" or "0".

"1" – Prompts the user to enter their login details as shown (the example email and password used are "s@" and "s")

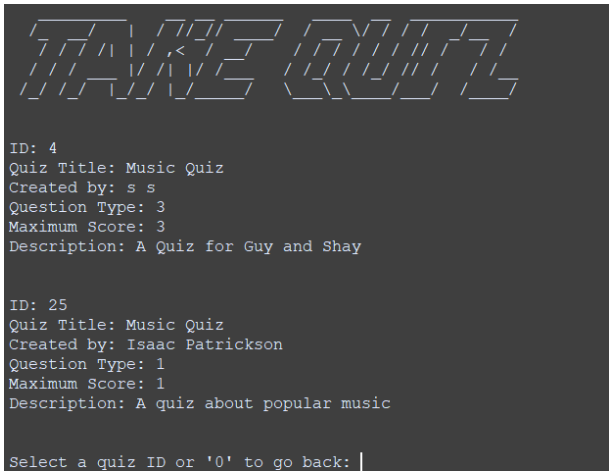
"0" – Closes the program

```
 1. Login
 2. Sign Up
 0. Quit

Select 1, 2 or 0: 2

First Name: Isaac
Last Name: Patrickson
Email address: isaac@patrickson.com
Password: 4a5secure9password1
Your Account has been created
```

"2" – Prompts the user to enter their details for their account creation. These details will be saved in the database.

The example data entered is "Isaac", "Patrickson", "isaac@patrickson.com", "4a5secure9password1"

All the data meets the requirements for the database, so the account has been created. The user would then be taken to the Main menu after this process.

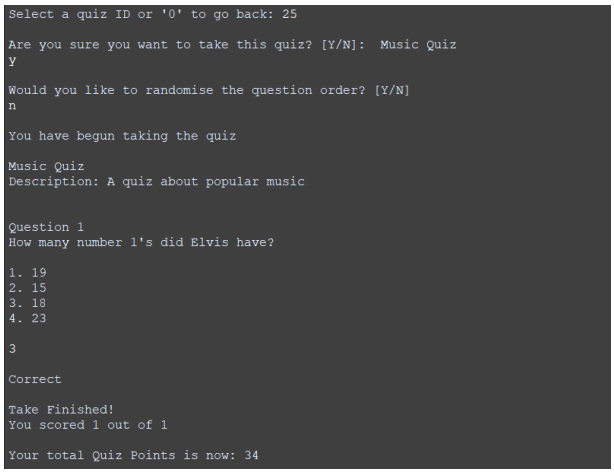


```
ID: 4
Quiz Title: Music Quiz
Created by: s s
Question Type: 3
Maximum Score: 3
Description: A Quiz for Guy and Shay


ID: 25
Quiz Title: Music Quiz
Created by: Isaac Patrickson
Question Type: 1
Maximum Score: 1
Description: A quiz about popular music


Select a quiz ID or '0' to go back:
```

This is the quiz selection screen where every quiz in the database is displayed on screen. For the screenshot, there are two quizzes in the database. If there were one hundred quizzes, they would all be displayed one after the other. A disadvantage of displaying quizzes like this would be that they would fly up the page and would require the user to scroll to find the ID of their desired quiz. This is impractical.

Each quiz has its important details displayed alongside it.

If this screen was done with Tkinter, there would be a scroll bar in a frame.



```
Select a quiz ID or '0' to go back: 25

Are you sure you want to take this quiz? [Y/N]:  Music Quiz
y

Would you like to randomise the question order? [Y/N]
n

You have begun taking the quiz

Music Quiz
Description: A quiz about popular music


Question 1
How many number 1's did Elvis have?

1. 19
2. 15
3. 18
4. 23

3

Correct

Take Finished!
You scored 1 out of 1

Your total Quiz Points is now: 34
```

The quiz with ID "25" has been selected. The user is then asked if they are sure they want to take this quiz.
"Y" – Proceeds to the quiz

"N" – Takes the user back to the Take Quiz menu

The user is asked if they would like to randomise the question order. One that choice has been made, the quizzes title and description are displayed.

Each question of the quiz is displayed one at a time. The system only prints the next question if the current question has been answered.

In this case, option "3" has been selected which is choice 18

Justification of Included Question Types

Below are the types of questions which appeared during my research of online quizzes completed over lockdown and my justification for including or not including them in the design of the new system. My design is dependent on which one's I choose to develop into my project.

Question type	To be included?	Justification
Open-ended	Yes	They are one of the most common types of pub quiz question and can be applied to every topic
Multiple-choice	Yes	They're a common format of online quiz question and can be answered quickly.
Audio file	No	It is possible to store audio files, but it is more common to store their file location or metadata. I will not be doing this because it would require the user to download image files in a specific folder each time, they would want to create a question containing an audio file.
Video file	No	It is possible to store a video file's file location as a reference but like audio files, they would need to be inserted into a specific file location for that reference to be consistent. As well as this, vides could go on forever unless a max file size is incorporated. Having these necessary features could cause frustration among quiz creators as files could be inserted incorrectly and then could be too large due to high quality. Putting effort into this question type that barely came up when doing my research wouldn't be sensible as there are a lot of variables that wouldn't under the programs control.
Picture file	No	I will be planning to use a MySQL database to store each question. Although possible to store images in the database, it is not advisable, and it isn't general practice. A general practice would be to store images in directories on the system and store the references to the images in the database. These references would have to stay constant for the stored image otherwise it could cause errors.

Database Normalization and Design

If my program uses a database, it will need to normalize it to avoid inconsistencies and the duplication of data. It will also save space by eliminating non-atomic data.

These are the tables of my database in third normal form

Bold – Primary key

user

id	firstName	lastName	email	salt	passwordHash	profile

quiz

id	hostId	title	type	maxScore	content

quiz_question

id	quizId	type	level	score	content

quiz_answer

id	quizId	questionId	correct	content

take

id	userId	quizId	score	content

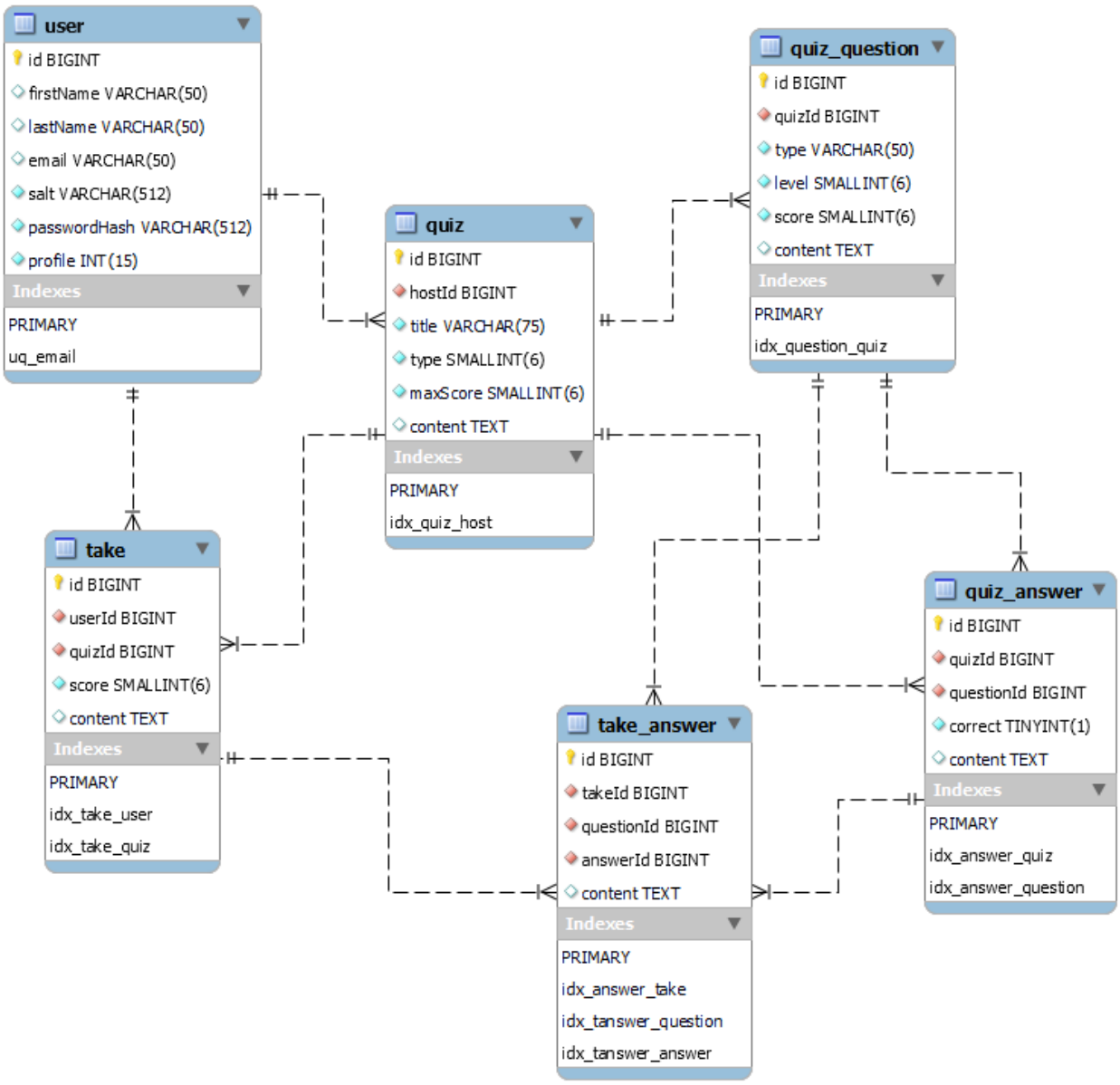
take_answer

id	takeId	questionId	answerId	content

My system's table relationship diagram

This diagram illustrates the relationships between the normalized tables in my program. We can tell the tables have been fully normalized as there are no many-to-many relationships.

Unlike the previous table relationship diagram for a typical quiz service, I am not including date and time. I did not deem this feature necessary as I do not want my system to be used for deadline tasks such as homework. As stated previously, this program is for people who enjoy partaking in quizzes for leisure.



Justification for using databases

I am using a MySQL database so I can permanently store essential information across different sessions. This is important because each user is linked to every quiz they have made and every quiz they have taken.

Not storing this data permanently means that between sessions, the user's data, and any actions they performed would be erased. I need this data to be saved between session, so users can login, view previously created quizzes, and view leader boards.

If I used a text file to store the data, there would be no structures for indexing or recognizing relationships between records. This would make handling the stored data impossible.

An advantage of a MySQL database would be that multiple users could connect to the database simultaneously. If the user has allowed remote access to the database, other computers from around the world can connect. This allows players to create quizzes for each other, build a community and compete for the top spot on the leader board.

Sample of Possible SQL Queries

Adding a new user to the database

```
"INSERT INTO user"
"(firstName, lastName, email, salt, passwordHash, profile) "
"VALUES (%s, %s, %s, %s, %s, %s)")
```

Displaying the user's full name

```
"SELECT firstName, lastName FROM user WHERE email = %s"
```

Adding a user's take to the database

```
"INSERT INTO take"
"(userId, quizId, score, content) "
"VALUES (%s, %s, %s, %s)")
```

Updating the user's take with their score once they have finished the take

```
"UPDATE take SET score = " + str(scoreCount) + " WHERE id = " + takeId
```

Displaying a user's profile score

```
"SELECT profile FROM user WHERE id = " + str(id)
```

Deleting a quiz from the database

```
"DELETE FROM quiz WHERE id = " + deleteSelection
```

Displaying the questions in the quiz

```
"SELECT content FROM quiz_question WHERE quizId = " + quizSelection
```

Displaying the answers for a multiple-choice question

```
"SELECT content FROM quiz_answer WHERE questionId = " + questionIds[i]
```

Displaying users in a leader board for a specific quiz

```
"SELECT userId FROM take WHERE quizId = " + quizSelection
```

Displaying all quizzes

```
"SELECT * FROM quiz"
```

Security

I will be using a hashing algorithm to encrypt user passwords. This is to protect user data.

If the user's password is the same password for other services, if the database is hacked, their password could be used on more important services such as their email or bank.

This hashing algorithm will create a random string of characters for each user. This will be the user's "salt". This salt and the user's password will be combined and hashed using python's hash library. The final hashed password will be stored in the user's table alongside the user's records.

To login to that user's account, the user's unique email and password must be entered. Once the email is entered, the corresponding salt is retrieved. The password entered is then hashed with the retrieved salt. If this hashed password matches the stored hashed password, the user is granted access.

To access the database, the user must enter their MySQL username and password. Only users who know the username and password for the MySQL account can proceed to the game. The user can create multiple MySQL users which have access to this database however I will be using the root user when testing my project.

To protect the integrity of the store data, all data entry will be controlled by strict validation rules. Since all entries will be free text input, I will be doing validation for every input. This will eliminate typographic errors which may cause the system to crash.

Temporary Storage with OOP

```
7  class Take():
8
9      def __init__(self):
10         self.takeId = ""
11         self.userId = ""
12         self.quizId = ""
13         self.score = 0
14         self.content = ""
15
16     def set_take_id(self):
17         takeId = cursor.lastrowid
18         self.takeId = takeId
19
20     def set_user_id(self, userId):
21         self.userId = userId
22
23     def set_quiz_id(self, quizId):
24         self.quizId = quizId
25
26     def set_score(self, score):
27         self.score
28
29     def set_content(self, content):
30         self.content = content
31
32
33     def get_take_id(self):
34         return self.takeId
35
36     def get_user_id(self):
37         return self.userId
38
39     def get_quiz_id(self):
40         return self.quizId
41
42     def get_score(self):
43         return self.score
44
45     def get_content(self):
46         return self.content
```

Object Oriented Programming

Before data is stored in the database, it will be stored in object variables.

Each variable will represent the data which will be inserted into each column.

This data is set by the user through the "setter" functions.

That data is then added to the database.

But if the data of an object variable needs to be retrieved and put into another variable, the "getter" function can be used.

For example.

The variable "QuizId" is present in both the quiz and the quizzes' questions.

To assign this "QuizId" to its questions, I can get the "QuizId" from the quiz class and store it in the question class.

```
question.set_quiz_id(quiz.get_quiz_id())
```

This is a simpler method than retrieving the "QuizId" from the database each time.

Algorithm Design

Random Ordering of Multiple-Choice Answers

Explanation
Every multiple-choice question needs to have a different order of answers each time a quiz is taken, i.e., the first option shown cannot always be the correct answer. This algorithm randomizes the indexes of the array of four answers. One correct answer, and one/three incorrect answers depending on the multiple-choice question

Code

```
for i in range(len(questions)):

    if questionTypes[i] == "1":
        multipleOptions = []
        correctMCValues = []
        multipleOptions = select_all_strings_query_executor("SELECT content FROM quiz_answer WHERE questionId = " + questionIds[i])
        correctMCValues = select_all_integers_query_executor("SELECT correct FROM quiz_answer WHERE questionId = " + questionIds[i]

        for j in range(len(multipleOptions)-1, 0, -1):
            k = random.randint(0, j)
            multipleOptions[j], multipleOptions[k] = multipleOptions[k], multipleOptions[j]
            correctMCValues[j], correctMCValues[k] = correctMCValues[k], correctMCValues[j]

    print()
    print()
    print("Question", i+1)
    print(questions[i])
    print()

    if questionTypes[i] == "1":

        for j in range(len(multipleOptions)):
            print(str(j+1) + ".", multipleOptions[j])
```

User's Quizzes Search

Explanation
Users should be able to search for their own quizzes in case they want to delete them. Quizzes might need to be deleted due to errors in answers or spelling mistakes within the quiz details. A similar algorithm is used for searching for all quizzes.

Code

```
def select_users_quizzes(id):
    availableRecordIds = []
    availableHostIds = []
    availableTitles = []
    availableTypes = []
    availableMaxScores = []
    avaiableContents = []
    query = "SELECT * FROM quiz WHERE hostId = " + str(id)
    cursor.execute(query)
    quizTable = cursor.fetchall()
    if quizTable == []:
        print("No quizzes have been made yet")
        print()
    else:
        for record in quizTable:
            recordId = record[0]
            recordHostId = id
            recordTitle = record[2]
            recordType = record[3]
            recordMaxScore = record[4]
            recordContent = record[5]
            quizCreatorName = name_selector(recordHostId)
            availableRecordIds.append(recordId)
            availableHostIds.append(recordHostId)
            availableTitles.append(recordTitle)
            availableTypes.append(recordType)
            availableMaxScores.append(recordMaxScore)
            avaiableContents.append(recordContent)

            if recordType == "1":
                recordType = "Multiple Choice"

            elif recordType == "2":
                recordType = "Open Ended"

            elif recordType == "3":
                recordType = "Mixed"

    return availableRecordIds, availableHostIds, availableTitles, availableTypes, availableMaxScores, avaiableContents
```

Validation of Text Free Entries

Explanation
Entries need to be controlled by strict validation rules to minimize crashing. This validation will also filter the data before it goes into the database. If a record's datatype is integer but a string is entered, an error will occur. The algorithm below will check if the user's Quiz Id input is available to access. It will also check if they would like to go back to the main menu. If this wasn't validated, the database might try and retrieve a quiz with a non-existing Quiz Id (which cannot be done.)
Code

```
valid4 = False
while valid4 is False:

    while True:
        try:
            quizSelection = int(input("Select a quiz ID or '0' to go back: "))
            break
        except:
            print("That's not an ID")

    if quizSelection in availableRecordIds:
        valid4 = True
        quizSelection = str(quizSelection)
        scoreCount = 0
        takeTitle = select_query_executor("SELECT title FROM quiz WHERE id = " + quizSelection)


    elif quizSelection == 0:
        valid4 = True

    else:
        print("Please select an existing ID")
```

Technical Solution

Database Tables

Creating the schema 'quiz' so the tables have somewhere to reside

```
CREATE SCHEMA `quiz` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

CREATE TABLE `quiz`.`user` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `firstName` VARCHAR(50) NULL DEFAULT NULL,
  `lastName` VARCHAR(50) NULL DEFAULT NULL,
  `email` VARCHAR(50) NULL,
  `salt` VARCHAR(512) NOT NULL,
  `passwordHash` VARCHAR(512) NOT NULL,
  `profile` INT(15) NOT NULL DEFAULT 0,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `uq_email` (`email` ASC) );

CREATE TABLE `quiz`.`quiz` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `hostId` BIGINT NOT NULL,
  `title` VARCHAR(75) NOT NULL,
  `type` SMALLINT(6) NOT NULL DEFAULT 0,
  `maxScore` SMALLINT(6) NOT NULL DEFAULT 0,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_quiz_host` (`hostId` ASC),
  CONSTRAINT `fk_quiz_host`
    FOREIGN KEY (`hostId`)
    REFERENCES `quiz`.`user` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

CREATE TABLE `quiz`.`quiz_question` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `quizId` BIGINT NOT NULL,
  `type` VARCHAR(50) NOT NULL,
  `level` SMALLINT(6) NOT NULL DEFAULT 0,
  `score` SMALLINT(6) NOT NULL DEFAULT 0,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_question_quiz` (`quizId` ASC),
  CONSTRAINT `fk_question_quiz`
    FOREIGN KEY (`quizId`)
    REFERENCES `quiz`.`quiz` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Creating the User table

- All columns except 'profile' need to be able to store variable characters (both integers and strings)
- 'salt' and 'passwordHash' need to have a larger capacity for the encrypted passwords
- 'profile' is an integer as it stores the user's total points accumulated

'id' will be set as the primary key for each table

'hostId' is being used as a foreign key. If someone tried to delete the user, they would have to delete all records linked to the host's ID first

Setting 'hostId' as a foreign key

If 'hostId' were to be deleted or updated, deny the action

When a quiz is created, if the fields with 'NOT NULL' are left empty, the quiz will not be created.

```
CREATE TABLE `quiz`.`quiz_answer` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `quizId` BIGINT NOT NULL,
  `questionId` BIGINT NOT NULL,
  `correct` TINYINT(1) NOT NULL DEFAULT 0,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_answer_quiz` (`quizId` ASC),
  CONSTRAINT `fk_answer_quiz`
    FOREIGN KEY (`quizId`)
    REFERENCES `quiz`.`quiz` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

ALTER TABLE `quiz`.`quiz_answer`
ADD INDEX `idx_answer_question` (`questionId` ASC);
ALTER TABLE `quiz`.`quiz_answer`
ADD CONSTRAINT `fk_answer_question`
  FOREIGN KEY (`questionId`)
  REFERENCES `quiz`.`quiz_question` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;

CREATE TABLE `quiz`.`take` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `userId` BIGINT NOT NULL,
  `quizId` BIGINT NOT NULL,
  `score` SMALLINT(6) NOT NULL DEFAULT 0,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_take_user` (`userId` ASC),
  CONSTRAINT `fk_take_user`
    FOREIGN KEY (`userId`)
    REFERENCES `quiz`.`user` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

ALTER TABLE `quiz`.`take`
ADD INDEX `idx_take_quiz` (`quizId` ASC);
ALTER TABLE `quiz`.`take`
ADD CONSTRAINT `fk_take_quiz`
  FOREIGN KEY (`quizId`)
  REFERENCES `quiz`.`quiz` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;
```

'id' must be a big integer (an integer which can take up 8 bytes of storage). This is because, theoretically there could be many users, quizzes, questions, answers, etc.

'correct' only needs to be a tiny integer as the value only needs to be '1' or '0'

I am using 'InnoDB' as my storage engine

The 'InnoDB' storage engine maintains its own buffer pool that caches table and index data in main memory as data is accessed. This cache speeds up processing.

I made alterations to the 'quiz' table and the 'take' table. I made a new index called 'idx_take_quiz' which is linked to the 'quizId' of the taken quiz.

Setting 'quizId' as the foreign key in the 'take' table.

If the quiz is to be deleted, every take with its quizId must be deleted first.

If the quiz where to change its ID, all takes of that quiz must change first.

If these delete or update actions are attempted without doing the instructions stated above, the quiz will not delete

```
CREATE TABLE `quiz`.`take_answer` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `takeId` BIGINT NOT NULL,
  `questionId` BIGINT NOT NULL,
  `answerId` BIGINT NOT NULL,
  `content` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `idx_answer_take` (`takeId` ASC),
  CONSTRAINT `fk_answer_take`
    FOREIGN KEY (`takeId`)
    REFERENCES `quiz`.`take` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

ALTER TABLE `quiz`.`take_answer`
ADD INDEX `idx_tanswer_question` (`questionId` ASC);
ALTER TABLE `quiz`.`take_answer`
ADD CONSTRAINT `fk_tanswer_question`
  FOREIGN KEY (`questionId`)
  REFERENCES `quiz`.`quiz_question` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;

ALTER TABLE `quiz`.`take_answer`
ADD INDEX `idx_tanswer_answer` (`answerId` ASC);
ALTER TABLE `quiz`.`take_answer`
ADD CONSTRAINT `fk_tanswer_answer`
  FOREIGN KEY (`answerId`)
  REFERENCES `quiz`.`quiz_answer` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;
```

I made alterations to the take answer table

- Set foreign keys
- Set constraints

These constraints are vital to maintaining the integrity of the data within the database.

Without them, data could be updated or deleted, leaving other records possibly linked to nothing.

This unlinked data could clog up the system and cause errors if accessed

Database Connection

To connect python to the MySQL database I needed to import the MySQL connector. This library was automatically installed to pip when I downloaded the MySQL package.

The library allows me to make a connection through my MySQL account which has access to the 'quiz' database. As well as making a connection, it allows me to make a cursor. This cursor is a tool which allows me to update, delete and apply other MySQL commands to my database through python.

```
import mysql.connector
```

This import retrieves the 'errorcode' function within 'mysql.connector' library. This function allows for exception handling when dealing with database errors. Errors can be formatted using the '.format(err)' command. The error message will appear in the form of an error code followed by a brief explanation on why the error occurred.

```
from mysql.connector import errorcode
```

```
Something went wrong: 1045 (28000): Access denied for user 'fnd'@'localhost'
(using password: YES)
```

The algorithm for logging into a user's MySQL account

```
# This while loop will ask the user to enter their details of their MySQL user
so they can connect to the database 'quiz'
while True:
    print()
    global cnxUser
    global cnxPassword
    cnxUser = input("Enter your MySQL username: ")
    cnxPassword = input("Enter your MySQL password: ")
    try:
        cnx = mysql.connector.connect(user=cnxUser, password=cnxPassword,
                                      database='quiz')
        cursor = cnx.cursor()
        break
    # If the user's details are incorrect the error will be handled with
      exception handling
    except mysql.connector.Error as err:
        print()
        print("Something went wrong: {}".format(err))
```

Exception handling in case the entered username and password are incorrect

```
quiz_connection(cnxUser, cnxPassword)
quiz_answer_connection(cnxUser, cnxPassword)
quiz_question_connection(cnxUser, cnxPassword)
sql_executor_connection(cnxUser, cnxPassword)
take_connection(cnxUser, cnxPassword)
take_answer_connection(cnxUser, cnxPassword)
user_connection(cnxUser, cnxPassword)
```

'cnxUser' and 'cnxPassword' are made global so they can be used in establishing the connections for all the other imported python files.

Every database change throughout the program is done through the MySQL user signed in at the start

Hashing Algorithm

The hashing algorithm I used to encrypt each user's password

```
import hashlib, binascii, os

def salt():
    salt = hashlib.sha256(os.urandom(60)).hexdigest()
    return salt

def hash_password(password, salt):
    salt = salt.encode('ascii')
    passwordHash = hashlib.pbkdf2_hmac('sha512', password.encode('utf-8'),
                   salt, 100000)
    passwordHash = binascii.hexlify(passwordHash)
    return(salt+passwordHash).decode('ascii')

def check_password(salt, storedPasswordHash, enteredPassword):
    passwordHash = hashlib.pbkdf2_hmac('sha512', enteredPassword.encode('utf-
                                                                        8'),
                   salt.encode('ascii'), 100000)
    passwordHash = binascii.hexlify(passwordHash).decode('ascii')
    return passwordHash == storedPasswordHash

# salt = salt()
# print(salt)
# storedPasswordHash = hash_password(input('Set your password: '), salt)
# print(check_password(storedPasswordHash, input('Enter your password: ')))

# Compares the entered password with the stored one
```

I imported python's inbuilt hash library, binascii (converts between binary and ASCII) and os is used to generate random bytes.

I created a salt for each user using a random list of 60 bytes and sha-256. 'Hexdigest' returns the encoded data in hexadecimal format.

The salt and the entered password are both used to create the hashed password

The entered login password and the salt are used to create a hashed password. If this hashed password matches the one stored in the user's account, this function returns True

Annotated Code

Main.py

```
from __future__ import print_function
# Changes the behavior of the 'print' funnction for older versions of python

import mysql.connector
# Importing the MySQL python connector (must have MySQL downnloaded)
import random
import sys
import time
# Importing python's premade libraries

from Hashing_Algorithm import *
from mysql.connector import errorcode
from Quiz import *
from Quiz_Answer import *
from Quiz_Question import *
from SQL_Query_Executor import *
from Take import *
from Take_Answer import *
from User import *
# Importing each part of my program so it can be accessed in the 'main'

# All subroutines, functions and imported files are in Snake case (login_menu),
# whereas variable names are in Camel case (cnxPassword)
# This is to help differentiate between variables and processes

# This while loop will ask the user to enter their details of their MySQL user so they can connect to the database 'quiz'
while True:
    print()
    cnxUser = input("Enter your MySQL username: ")
    cnxPassword = input("Enter your MySQL password: ")
    try:
        cnx = mysql.connector.connect(user=cnxUser, password=cnxPassword, database='quiz')
        cursor = cnx.cursor()
        print()
        print("MySQL account has been accepted")
        quiz_connection(cnxUser, cnxPassword)
        quiz_answer_connection(cnxUser, cnxPassword)
        quiz_question_connection(cnxUser, cnxPassword)
        sql_executor_connection(cnxUser, cnxPassword)
        take_connection(cnxUser, cnxPassword)
        take_answer_connection(cnxUser, cnxPassword)
        user_connection(cnxUser, cnxPassword)
        break
    # If the user's details are incorrect the error will be handled with exception handling
    except mysql.connector.Error as err:
        print()
        print("Something went wrong: {}".format(err))


# This is a subroutine which loads the Login Menu for the system
def login_menu():
    # I have used time.sleep to create an animation for the title's appearancee.
    # This style choice mimmicks that of an old computer and gives my program more character
    print()
    time.sleep(0.4)
    print()
    print("  ____  __  ___________      _________    __  ________")
    time.sleep(0.4)
    print("   / __ \/ / / /  _/__  /     / ____/   |  /  |/  / ____/")
    time.sleep(0.4)
    print("  / / / / / / // /   / /     / / __/ /| | / /|_/ / __/   ")
    time.sleep(0.4)
    print(" / /_/ / /_/ // /   / /__   / /_/ / ___ |/ /  / / /___   ")
    time.sleep(0.4)
    print(" \___\_\____/___/  /____/   \____/_/  |_/_/  /_/_____/   ")
    time.sleep(0.4)
    print()
    print()
    print(" 1. Login")
    print(" 2. Sign Up")
    print(" 0. Quit")
    print()

    # This while loop will validate the user's choice of action at the Login Menu
    loginMenuValid = False
    while loginMenuValid is False:
        choice = input("Select 1, 2 or 0: ")

        if choice == "1":
            loginMenuValid = True
```

```
            while True:
                try:
                    login = User()
                    print()
                    login.set_email(input("Email address: "))
                    userEmail = login.get_email()
                    query = "SELECT salt FROM user WHERE email = %s"
                    cursor.execute(query, (userEmail,))
                    for salt in cursor:
                        userSalt = ("{}".format(salt))
                        userSalt = userSalt.replace("'","")
                        userSalt = userSalt.replace("(","")
                        userSalt = userSalt.replace(")","")
                        userSalt = userSalt.replace(",","")
                    query = "SELECT passwordHash FROM user WHERE email = %s"
                    cursor.execute(query, (userEmail,))
                    for passwordHash in cursor:
                        userHPassword = ("{}".format(passwordHash))
                        userHPassword = userHPassword.replace("'","")
                        userHPassword = userHPassword.replace("(","")
                        userHPassword = userHPassword.replace(")","")
                        userHPassword = userHPassword.replace(",","")
                    check = check_password(userSalt, userHPassword, input("Password: "))
                    break
                # If the email entered does not exist and a salt cannot be selected,
                # the error will be dealt with using exception handling
                except:
                    print("Error! An account with this email does not exist")
                    print()
                    login_menu()

            # If the email does exist in the database and the passwords match,
            # the user is sent to the User Menu
            if check == True:
                email = login.get_email()
                login.set_user_id(email)
                name = name_selector(login.get_user_id())
                print()
                print("Login Successful")
                print("Welcome,", name)
                user_menu(login.get_email(), login.get_user_id())

            #If the password does not match, the user is sent back to the Login Menu
            else:
                print("Error! Incorrect password")
                print()
                login_menu()

        # The user has chosen to create an account
        elif choice == "2":
            loginMenuValid = True

            while True:
                # The program will go through each field required to create an account.
                # If one of the fields causes an error, the user is sent back to the Login Menu
                try:
                    print()
                    new = User()
                    new.set_first_name(input("First Name: "))
                    new.set_last_name(input("Last Name: "))
                    new.set_email(input("Email address: "))
                    new.set_salt()
                    new.set_password_hash(input("Password: "), new.get_salt())
                    new.set_profile(0)
                    new.add_user()
                    new.set_user_id(new.get_email())
                    break
                except:
                    print()
                    print("Error! An account with that email already exists")
                    login_menu()

            user_menu(new.get_email(), new.get_user_id())

        # If the User has chosen to exit the program, the program will close
        elif choice == "0":
            loginMenuValid = True
            exit()

        else:
            print("Error! Input must be an integer preceding one of the options above")

def user_menu(signedInUserEmail, signedInUserId):
    time.sleep(0.4)
```

```
    print()
    print("      __ ______    _____   __   __ __________  ____  __")
    print("     /  |/  /   |  /  _/ | / /  /  |/  / ____/ | / / / /")
    print("    / /|_/ / /| |  / //  |/ /  / /|_/ / __/ /  |/ / / / ")
    print("  / /  / / ___ |_/ // /|  /  / /  / / /___/ /|  / /_/ /  ")
    print(" /_/  /_/_/  |_/___/_/ |_/  /_/  /_/_____/_/ |_/\____/   ")
    print()
    print()
    time.sleep(0.4)
    print(" 1. Take A Quiz")
    print(" 2. Create A Quiz")
    print(" 3. Delete A Quiz")
    print(" 4. View Profile")
    print(" 5. View Leaderboards")
    print(" 0. Logout")
    print()

    # This while loop will validate the user's choice of action at the Main Menu
    mainMenuValid = False
    while mainMenuValid is False:
        choice = input("Select 1, 2, 3, 4, 5 or 0: ")

        # This choice will take the user to the Take Menu
        if choice == "1":
            mainMenuValid = True
            take_quizzes_menu(signedInUserEmail, signedInUserId)

        # This choice will take the user to the Create Quiz Menu
        elif choice == "2":
            mainMenuValid = True
            create_quiz_menu(signedInUserEmail, signedInUserId)

        # This choice will take the user to the Modify Menu
        elif choice == "3":
            mainMenuValid = True
            delete_menu(signedInUserEmail, signedInUserId)

        # This choice will retrieve the user's total accumulated quiz points and display them on screen
        elif choice == "4":
            mainMenuValid = True
            userTotalScore = select_query_executor("SELECT profile FROM user WHERE id = " + signedInUserId)
            print()
            print("Total Quiz Points Earned:", userTotalScore)
            print()

            # A while loop to check if the user wants to go back to the Main Menu
            goBackFromProfile = False
            while goBackFromProfile is False:
                print("0. Go Back")
                goBack = input()

                if goBack == "0":
                    goBackFromProfile = True
                    time.sleep(0.5)
                    user_menu(signedInUserEmail, signedInUserId)

                else:
                    print("Error! Must type '0' to go back to the Menu")
                    print()


        # This choice will take the user to the Leader boards Menu
        elif choice == "5":
            mainMenuValid = True
            leaderboard_menu(signedInUserEmail, signedInUserId)

        # This choice will take the user back to the Login Menu by breaking the loop causing 'Login_menu()' to trigger
        elif choice == "0":
            mainMenuValid = True

        else:
            print("Error! Input must be an integer preceding one of the options above")
            print()

    login_menu()

def take_quizzes_menu(signedInUserEmail, signedInUserId):
    time.sleep(0.4)
    print()
    print("    ________  __ __ ______  ____ __ __ ___________")
    print("   /_  __/   |  / //_// ____/  / __ \/ / / /  _/__  /")
    print("    / / / /| | / ,<  / __/    / / / / / / // /   / / ")
    print("   / / / ___ |/ /| |/ /___   / /_/ / /_/ // /   / /__")
    print("  /_/ /_/  |_/_/ |_/_____/   \___\_\____/___/  /____/")
    print()
    print()

    # 'select_all_quizzes()' returns all the necessary information for printing all quizzes in the quiz table
    availableRecordIds, availableHostIds, availableTitles, availableTypes, availableMaxScores, availableContents = select_all_quizzes()

    # This for loop prints every quiz found with 'select_all_quizzes()'
    for i in range(len(availableRecordIds)):
        time.sleep(0.25)
        HostName = name_selector(availableHostIds[i])
        print("ID:", availableRecordIds[i], "\n"
              "Quiz Title:", availableTitles[i], "\n"
              "Created by:", HostName, "\n"
              "Question Type:", availableTypes[i], "\n"
              "Maximum Score:", availableMaxScores[i], "\n"
              "Description:", availableContents[i])
        print("\n")

    # This while loop will check if the user's quiz selection is in the list of quizzes displayed
    quizSelectionValid = False
    while quizSelectionValid is False:

        # This while loop makes sure that the ID entered is an integer
        while True:
            try:
                quizSelection = int(input("Select a quiz ID or '0' to go back: "))
                break
            except:
                print("Errror! That's not an ID")
                print()

        # This if statement checks that the integer is in the list of available Quiz IDs
        if quizSelection in availableRecordIds:
            quizSelectionValid = True
            quizSelection = str(quizSelection)
            scoreCount = 0
            takeTitle = select_query_executor("SELECT title FROM quiz WHERE id = " + quizSelection)
```

```
288
289             # Here, the user has a chance to back out of taking the quiz.
290             # Their answer to the [Y/N] question is then validated
291             takeQuizYNvalid = False
292             while takeQuizYNvalid is False:
293                 print()
294                 print("Are you sure you want to take this quiz? [Y/N]: ", takeTitle)
295                 choice = input()
296                 choice = choice.upper()
297
298                 # If they have entered 'Y' they will go on to take the quiz
299                 if choice == "Y":
300                     takeQuizYNvalid = True
301
302                     # The user is asked if they want the question order randomised.
303                     # Their answer to the [Y/N] question is then validated
304                     validYNRandOrder = False
305                     while validYNRandOrder is False:
306                         print()
307                         print("Would you like to randomise the question order? [Y/N]")
308                         randOrderChoice = input()
309                         randOrderChoice = randOrderChoice.upper()
310
311                         if randOrderChoice == "Y":
312                             validYNRandOrder = True
313                             randOrder = True
314
315                         elif randOrderChoice == "N":
316                             validYNRandOrder = True
317                             randOrder = False
318
319                         else:
320                             print("Error! Please enter Y or N")
321
322                     # The object "takeQuiz" is created and its variables are set with the take setter subroutines
323                     takeQuiz = Take()
324                     takeQuiz.set_quiz_id(quizSelection)
325                     takeQuiz.set_user_id(id)
326                     takeQuiz.set_quiz_id(quizSelection)
327                     takeQuiz.set_score(0)
328                     content = select_query_executor("SELECT content FROM quiz WHERE id = " + quizSelection)
329                     takeQuiz.set_content(content)
330                     takeQuiz.add_take()
331                     takeQuiz.set_take_id()
332                     scoreCount = 0
333
334                     # The user is presented with the title and the description of the quiz they are about to take
335                     print()
336                     print(takeTitle)
337                     print("Description:", content)
338
339                     # All question IDs, types and contents are retrieved from the table (in order) and saved to lists
340                     questions = []
341                     questionTypes = []
342                     questionIds = []
343                     questions = select_all_strings_query_executor("SELECT content FROM quiz_question WHERE quizId = " + quizSelection)
344                     questionTypes = select_all_strings_query_executor("SELECT type FROM quiz_question WHERE quizId = " + quizSelection)
345                     questionIds = select_all_integers_query_executor("SELECT id FROM quiz_question WHERE quizId = " + quizSelection)
346
347                     # If the user has selected to randomise the question order, this for loop will randomise each in the same way
348                     if randOrder == True:
349                         for i in range(len(questions)-1, 0, -1):
350                             j = random.randint(0, i)
351                             questions[i], questions[j] = questions[j], questions[i]
352                             questionTypes[i], questionTypes[j] = questionTypes[j], questionTypes[i]
353                             questionIds[i], questionIds[j] = questionIds[j], questionIds[i]
354
355                     # For each question (Multiple-Choice or Open ended) the answers are retrieved from the database,
356                     # alongside their corresponding correct value
357                     # "1" - Correct
358                     # "0" - Incorrect
359                     for i in range(len(questions)):
360                         if questionTypes[i] == "1":
361                             multipleOptions = []
362                             correctMCValues = []
363                             multipleOptions = select_all_strings_query_executor("SELECT content FROM quiz_answer WHERE questionId = " + questionIds[i])
364                             correctMCValues = select_all_integers_query_executor("SELECT correct FROM quiz_answer WHERE questionId = " + questionIds[i])
365
366                             # Even if the question order is not random, the choices for a multiple choice question will always be random
367                             # This will randomise the answer contents and the values determining if the answer choice is correct or not
368                             # Both lists will be randomised the same way so their values still match up
369                             for j in range(len(multipleOptions)-1, 0, -1):
370                                 k = random.randint(0, j)
371                                 multipleOptions[j], multipleOptions[k] = multipleOptions[k], multipleOptions[j]
372                                 correctMCValues[j], correctMCValues[k] = correctMCValues[k], correctMCValues[j]
373
374                         print()
375                         print()
376                         print("Question", i+1)
377                         print(questions[i])
378                         print()
```

```
                    # If the question type is multiple-choice, the answer options are printed
                    if questionTypes[i] == "1":
                        for j in range(len(multipleOptions)):
                            print(str(j+1) + ".", multipleOptions[j])

                        # This while loop will make sure the user's choice is valid
                        # Checking to see if the choice is an integer
                        # Checking if the choice is one of the available options
                        while True:
                            try:
                                print()
                                takeOption = int(input("Enter your choice here: "))
                                if takeOption > 0 and takeOption <= len(multipleOptions):
                                    takeAnswer = Take_Answer()
                                    takeAnswer.set_take_id(takeQuiz.get_take_id())
                                    takeAnswer.set_question_id(questionIds[i])
                                    quizId = takeQuiz.get_quiz_id()
                                    answerId = select_query_executor("SELECT id FROM quiz_answer WHERE quizId = " + quizId
                                                                     + " AND questionId = " + questionIds[i]
                                                                     + " AND correct = 1")
                                    takeAnswer.set_answer_id(answerId)
                                    takeAnswer.set_answer_content(multipleOptions[takeOption - 1])
                                    takeAnswer.add_take_answer()
                                    break
                                else:
                                    print("Error! Input must be an integer preceding one of the options above")
                                    print()
                            except:
                                print("Error! Input must be an integer")
                                print()

                        # If the value at position "takeOption - 1" in "correctMCValues" is 1, the chosen option is correct
                        # The user's score for the take is increased by one
                        if correctMCValues[takeOption - 1] == "1":
                            scoreCount += 1
                            print()
                            print("Correct")

                        # If the value is not "1" (is "0") the chosen option is incorrect
                        # The user's score for the take remains the same
                        else:
                            print()
                            print("Incorrect")

                    # If the question type for the question is open ended, all the necessary data is collected and stored in the "openEndedOption" object
                    if questionTypes[i] == "2":
                        openEndedOption = []
                        openEndedOption = select_query_executor("SELECT content FROM quiz_answer WHERE questionId = " + questionIds[i])
                        takeOption = input("Enter your answer here: ")
                        takeAnswer = Take_Answer()
                        takeAnswer.set_take_id(takeQuiz.get_take_id())
                        takeAnswer.set_question_id(questionIds[i])
                        quizId = takeQuiz.get_quiz_id()
                        answerId = select_query_executor("SELECT id FROM quiz_answer WHERE quizId = " + quizId
                                                         + " AND questionId = " + questionIds[i]
                                                         + " AND correct = 1")
                        takeAnswer.set_answer_id(answerId)
                        takeAnswer.set_answer_content(takeOption)
                        takeAnswer.add_take_answer()

                        # If the user input matches the answer's content, the entered answer is correct
                        # This enetered answer is not case sensitive
                        # The user's score for the take is increased by one
                        if takeOption.upper() == openEndedOption.upper():
                            scoreCount += 1
                            print()
                            print("Correct")

                        # If the user's input does not match the answer's content, the entered answer is incorrect
                        # The user's score for the take remains the same
                        else:
                            print()
                            print("Incorrect")

                # All the questions have been answered so the take is finished
                print()
                print("Take Finished!")
                takeId = str(takeQuiz.get_take_id())
                update_query_executor("UPDATE take SET score = " + str(scoreCount) + " WHERE id = " + takeId)
                profileScore = select_query_executor("SELECT profile FROM user WHERE id = " + str(signedInUserId))
                update_query_executor("UPDATE user SET profile = " + str(int(profileScore) + scoreCount) + " WHERE id = " + str(signedInUserId))
                maxPossibleScore = select_query_executor("SELECT maxScore FROM quiz WHERE id = " + quizSelection)
                profileScore = select_query_executor("SELECT profile FROM user WHERE id = " + str(signedInUserId))

                print("You scored", scoreCount, "out of", maxPossibleScore)
                print()
                print("Your total Quiz Points is now:", profileScore)
                user_menu(signedInUserEmail, signedInUserId)

            # If the user does not want to take this test, they can back out and the take quizzes menu will display itself once more
            elif choice == "N":
                takeQuizYNvalid = True
                take_quizzes_menu(signedInUserEmail, signedInUserId)

            # If the user does not input either 'Y' or 'N' they are prompted to enter the input again
            else:
                print("Error! Please enter Y or N")

        # If the user wants to go back to the main menu the loop will break
        elif quizSelection == 0:
            quizSelectionValid = True

        else:
            print("Error! Please select an existing ID")
            print()

    user_menu(signedInUserEmail, signedInUserId)
```

```
487
488
489 def create_quiz_menu(signedInUserEmail, signedInUserId):
490     time.sleep(0.4)
491     print()
492     print("    _________ _________ _____________   ____ __ ___________")
493     print("   / ____/ __ \/ ____/   |/_  __/ ____/  / __ \/ / / /  _/__  /")
494     print("  / /   / /_/ / __/ / /| | / / / __/    / /_/ / / / // /   / / ")
495     print(" / /___/ _, _/ /___/ ___ |/ / / /___   / /_/ / /_/ // /   / /__")
496     print(" \____/_/ |_/_____/_/  |_/_/ /_____/   \___\_\____/___/  /____/")
497     print()
498     print()
499     # The user is immediately put into the process of creating a quiz
500     # The user's inputs are recorded and temporarily stored in the 'quiz' object variables
501     quiz = Quiz()
502     quiz.set_host_id(signedInUserId)
503     quiz.set_title(input("Quiz title: "))
504     quiz.set_quiz_type()
505     quiz.set_max_score()
506     print()
507     quiz.set_content(input("Quiz description: "))
508     quiz.add_quiz() # Quiz is added to the database
509     quiz.set_quiz_id()
510
511     # Since each question is always worth one point, the max score is euqal to the number of questions in the quiz
512     # So repeat the question create process a 'max score' amount of times
513     for i in range(quiz.get_max_score()):
514         # The user's inputs are recorded and temporarily stored in the 'question' object variables
515         print()
516         print("Question", i+1)
517         question = Question()
518         question.set_quiz_id(quiz.get_quiz_id())
519         question.set_question_type(quiz.get_quiz_type())
520         question.set_level(input("Easy[1], Medium[2] or Hard[3]: "))
521         question.set_score()
522         print()
523         question.set_question_content(input("Question content: "))
524         question.add_question() # Question is added to the database
525         question.set_question_id()
526
527         # If the user has chosen to create a multiple-choice question,
528         # the user must input wrong answers as well as one right answer
529         # The user can choose either two options (one right, one wrong) or four options (one right and three wrong)
530         if question.get_question_type() == "1":
531             # The user's inputs are recorded and temporarily stored in the 'multipleChoice' object variables
532             multipleChoice = Answer()
533             multipleChoice.set_quiz_id(quiz.get_quiz_id())
534             multipleChoice.set_question_id(question.get_question_id())
535             numberOfOptions = multiple_choice_handling()
536             for j in range(numberOfOptions):
537                 print()
538                 print("Option",j+1)
539                 if str(j) == "0":
540                     print("You can only have one correct option. Enter your correct option first.")
541                     multipleChoice.set_correct()
542                     multipleChoice.set_answer_content(input("Question Content: "))
543                     multipleChoice.add_answer() # The correct answer is added to the database
544                 else:
545                     print("Enter your incorrect option(s) now.")
546                     multipleChoice.set_incorrect()
547                     multipleChoice.set_answer_content(input("Question Content: "))
548                     multipleChoice.add_answer() # The incorrect answer is added to the database
549                 print("Option",j+1,"has been created")
550
551         # If the user has chosen to create an open ended question, the user must input just the right answer
552         elif question.get_question_type() == "2":
553             # The user's inputs are recorded and temporarily stored in the 'openEnded' object variables
554             openEnded = Answer()
555             openEnded.set_correct()
556             openEnded.set_quiz_id(quiz.get_quiz_id())
557             openEnded.set_question_id(question.get_question_id())
558             openEnded.set_answer_content(input("Answer Content: "))
559             openEnded.add_answer() # The correct answer is added to the database
560
561         # If the user has chosen to create a quiz with mixed question types,
562         # the user gets to choose whether each questions is open ended or multiple choice
563         elif question.get_question_type() == "3":
564             # This while loop will validate the user's input
565             # Their choice must either be 'M' or 'O'
566
567             mOrOValid = False
568             while mOrOValid is False:
569
570                 # If the user has chosen 'M', they go through the process for creating a multiple choice question
571                 if multipleOrOpen.upper() == "M":
572                     mOrOValid = True
573                     update_query_executor("UPDATE quiz_question SET type = 1 WHERE type = 3") # Since there is no 'question type 3',
574                     multipleChoice = Answer()                                                 # the type needs to be set to multiple choice (type 1)
575                     multipleChoice.set_quiz_id(quiz.get_quiz_id())                            # Type 3 is used as a palceholder for qeustions within a mixed quiz
576                     multipleChoice.set_question_id(question.get_question_id())
577                     numberOfOptions = multiple_choice_handling()
578                     for j in range(numberOfOptions):
579                         print()
580                         print("Option",j+1)
581                         if str(j) == "0":
582                             print("You can only have one correct option. Enter your correct option first.")
583                             multipleChoice.set_correct()
584                             multipleChoice.set_answer_content(input("Content: "))
585                             multipleChoice.add_answer()
586                         else:
587                             print("Enter your incorrect option(s) now.")
588                             multipleChoice.set_incorrect()
589                             multipleChoice.set_answer_content(input("Content: "))
590                             multipleChoice.add_answer()
591                         print("Option",j+1,"has been created")
592
593                 elif multipleOrOpen.upper() == "O":
594                     mOrOValid = True
595                     update_query_executor("UPDATE quiz_question SET type = 2 WHERE type = 3") # Since there is no 'question type 3',
596                     openEnded = Answer()                                                      # the type needs to be set to open ended (type 2)
597                     openEnded.set_correct()                                                   # Type 3 is used as a placeholder for questions withing a mixed quiz
598                     openEnded.set_quiz_id(quiz.get_quiz_id())
599                     openEnded.set_question_id(question.get_question_id())
600                     openEnded.set_answer_content(input("Answer Content: "))
601                     openEnded.add_answer()
602
603                 else:
604                     multipleOrOpen = input("Multiple choice question or Open ended [M/O]: ")
605
606     # Once the quiz has been created, the user is notified that the creation process is over,
607     # and is taken back to the user menu
608     print()
609     print("Quiz Creation Complete")
610     user_menu(signedInUserEmail, signedInUserId)
```

```
613
614 def delete_menu(signedInUserEmail, signedInUserId):
615     time.sleep(0.4)
616     print()
617     print("      ____  ________    _____________________")
618     print("    / __ \/ ____/ /   / ____/ _ __/ ____/")
619     print("   / / / / __/ / /   / __/   / / / __/    ")
620     print("  / /_/ / /___/ /___/ /___  / / / /___    ")
621     print(" /_____/_____/_____/_____/ /_/ /_____/    ")
622     print()
623     print()
624     print("Select a quiz you want to delete")
625     print()
626     time.sleep(0.4)
627     print("YOU CAN ONLY DELETE THE QUIZZES YOU HAVE MADE")
628     time.sleep(2)
629     # All the details of every quiz the user has created is collected for future use
630     # The user can only modify their quizzes and no one elses
631     availableRecordIds, availableHostIds, availableTitles, availableTypes, availableMaxScores, availableContents = select_users_quizzes(signedInUserId)
632
633     # Using the retrieved data from each quiz the user has made,
634     # print each quiz
635     for i in range(len(availableRecordIds)):
636         time.sleep(0.2)
637         HostName = name_selector(availableHostIds[i])
638         print()
639         print("ID:", availableRecordIds[i], "\n"
640               "Quiz Title:", availableTitles[i], "\n"
641               "Created by:", HostName, "\n"
642               "Question Type:", availableTypes[i], "\n"
643               "Maximum Score:", availableMaxScores[i], "\n"
644               "Description:", availableContents[i])
645         print("\n")
646
647     # This while loop checks if the ID entered is in the list of quiz IDs
648     isIdInQuizIdList = False
649     while isIdInQuizIdList is False:
650
651         # This while loop checks that the ID entered is an integer
652         while True:
653                 try:
654                     deleteSelection = int(input("Select a quiz ID or '0' to go back: "))
655                     break
656                 except:
657                     print("Error! Please enter an integer")
658                     print()
659
658
659         # If the entered ID is in the list of available quizzes,
660         #
661         if deleteSelection in availableRecordIds:
662             isIdInQuizIdList = True
663             deleteSelection = str(deleteSelection)
664             takeTitle = select_query_executor("SELECT title FROM quiz WHERE id = " + deleteSelection)
665             print()
666
667             # This while loop ask the user if they are sure they want to delete the quiz
668             # The user's input is also validated and can only be 'Y' or 'N' (not case sensitive)
669             deleteQuizYNValid = False
670             while deleteQuizYNValid is False:
671                 print("Are you sure you want to delete this quiz [Y/N]: ", takeTitle)
672                 choice = input()
673                 choice = choice.upper()
674
675                 # If they are sure they want to delete the quiz, the quiz is deleted
676                 # However not just the quiz is deleted
677                 # To delete the quiz, all 'take answers', 'takes', 'quiz answers' and 'quiz questions' must be deleted first
678                 # The quiz is then deleted
679                 if choice == "Y":
680                     deleteQuizYNValid = True
681                     selectedTakeIds = select_all_integers_query_executor("SELECT id FROM take WHERE quizId = " + deleteSelection)
682
683                     for TakeId in selectedTakeIds:
684                         update_query_executor("DELETE FROM take_answer WHERE takeId = " + TakeId)
685
686                     update_query_executor("DELETE FROM take WHERE quizId = " + deleteSelection)
687                     update_query_executor("DELETE FROM quiz_answer WHERE quizId = " + deleteSelection)
688                     update_query_executor("DELETE FROM quiz_question WHERE quizId = " + deleteSelection)
689                     update_query_executor("DELETE FROM quiz WHERE id = " + deleteSelection)
690                     # The user is notified of the quiz's deletion,
691                     # and is taken back to the modify menu
692                     print()
693                     print("Your Quiz has been deleted")
694                     delete_menu(signedInUserEmail, signedInUserId)
695
696                 elif choice == "N":
697                     deleteQuizYNValid = True
698                     print("Your Quiz has not been deleted")
699                     delete_menu(signedInUserEmail, signedInUserId)
700
701                 else:
702                     print("Error! Please enter either 'Y' or 'N'")
703                     print()
704
705         # If the user decided to go back to the user menu, the while loop will end
706         elif deleteSelection == 0:
707             isIdInQuizIdList = True
708
709         else:
710             print("Error! That ID doesn't exist")
711
712     user_menu(signedInUserEmail, signedInUserId)
```

```python



def leaderboard_menu(signedInUserEmail, signedInUserId):
    time.sleep(0.4)
    print()
    print("     __    _________    ____  __________  ____  ____  ___    ____  ____  _____")
    print("    / /   / ____/   |  / __ \/ ____/ __ \/ __ )/ __ \/   |  / __ \/ __ \/ ___/")
    print("   / /   / __/ / /| | / / / / __/ / /_/ / __  / / / / /| | / /_/ / / / /\__ \ ")
    print("  / /___/ /___/ ___ |/ /_/ / /___/ _, _/ /_/ / /_/ / ___ |/ _, _/ /_/ /___/ / ")
    print(" /_____/_____/_/  |_/_____/_____/_/ |_/_____/\____/_/  |_/_/ |_/_____//____/  ")
    print()
    print()
    print("1. View overall Leaderboard")
    print("2. View leaderboards per quiz")
    print("0. Go back")
    print()

    # This while loop validates the user's choice of action at the leaderboard menu
    leaderboardChoiceValid = False
    while leaderboardChoiceValid is False:
        choice = input("Select 1, 2 or 0: ")

        # If choice is '1' all user IDs are retreived from the database
        if choice == "1":
            leaderboardChoiceValid = True
            userIdList = select_all_integers_query_executor("SELECT id FROM user")
            userNameList = []
            scoreList = []
            userIdNameScoreList = []

            # For each user ID, their full name and profile score is retrieved
            for userId in userIdList:
                userName = name_selector(userId)
                userNameList.append(userName)
                profileScore = select_query_executor("SELECT profile FROM user WHERE id = " + userId)
                scoreList.append(profileScore)

            # For each user ID, their data is concatenated and added to a list as one item
            for i in range(len(userIdList)):
                userIdNameScore = ("USER ID: " + userIdList[i] + "  " + "USERNAME: " + userNameList[i] + "  " + "SCORE: " + scoreList[i])
                userIdNameScoreList.append(userIdNameScore)

            # All the user scores are ordered from highest to lowest
            # The concatenated user IDs, names and scores are ordered in the same way
            # This makes sure each user profile is matched up with their own score
            scoreList, userIdNameScoreList = zip(*sorted(zip(scoreList, userIdNameScoreList)))
            scoreList = scoreList[::-1]
            userIdNameScoreList = userIdNameScoreList[::-1]

            time.sleep(0.5)
            print()
            print()
            print("TOP 50 PLAYERS")
            print()

            # The top 50 highest scoring users are displayed one user at a time
            for i in range(len(scoreList[:50])):
                time.sleep(0.25)
                print(str(i+1) + ". " + str(userIdNameScoreList[i]))

            # After the top 50 users have been displayed, the user is asked if they want to return to the leaderboard menu
            # Their choice is validated
            print()
            goBackToLeaderboardsValid = False
            while goBackToLeaderboardsValid is False:
                print("0. Go Back")
                goBack = input()

                if goBack == "0":
                    goBackToLeaderboardsValid = True
                    leaderboard_menu(signedInUserEmail, signedInUserId)

                else:
                    print("Error! Must enter '0' to go back")
                    print()
                    pass

        # If the user wants to view a specific leaderboard for a certain quiz,
        # all quizzes are displayed for the user to choose which one they want to select
        elif choice == "2":
            leaderboardChoiceValid = True
            availableRecordIds, availableHostIds, availableTitles, availableTypes, availableMaxScores, availableContents = select_all_quizzes()
            for i in range(len(availableRecordIds)):
                time.sleep(0.25)
                HostName = name_selector(availableHostIds[i])
                print()
                print("ID:", availableRecordIds[i], "\n"
                      "Quiz Title:", availableTitles[i], "\n"
                      "Created by:", HostName, "\n"
                      "Question Type:", availableTypes[i], "\n"
                      "Maximum Score:", availableMaxScores[i], "\n"
                      "Description:", availableContents[i])
                print("\n")
```

```
807
808             # This while loop will check if the entered quiz ID is in the list of available quizIDs
809             quizLeaderboardSelectionValid = False
810             while quizLeaderboardSelectionValid is False:
811
812                 # This while loop will check if the entered ID is an integer
813                 while True:
814                         try:
815                             quizSelection = int(input("Select a quiz ID or '0' to go back: "))
816                             break
817                         except:
818                             print("Error! Please enter an integer.")
819                             print()
820
821                 # If the entered ID is in the list of quiz IDs,
822                 # Select the user ID and score for each user who has taken this quiz
823                 if quizSelection in availableRecordIds:
824                         quizSelection = str(quizSelection)
825                         userIdsOfQuizTakers = select_all_integers_query_executor("SELECT userId FROM take WHERE quizId = " + quizSelection)
826                         scoreForEachTake = select_all_integers_query_executor("SELECT score FROM take WHERE quizId = " + quizSelection)
827                         userNameList = []
828                         userIdNameScoreList = []
829
830                         # Gather each user's full name
831                         for userId in userIdsOfQuizTakers:
832                             userName = name_selector(userId)
833                             userNameList.append(userName)
834
835                         # If no one has taken the quiz, the following message is displayed
836                         if userIdsOfQuizTakers == []:
837                             time.sleep(0.5)
838                             print()
839                             print("No one has completed this quiz")
840                             print()
841
842                         # If other users have taken the quiz, each user is ordered from highest score to lowest score
843                         else:
844                             scoreForEachTake, userIdsOfQuizTakers, userNameList = zip(*sorted(zip(scoreForEachTake, userIdsOfQuizTakers, userNameList)))
845                             userIdsOfQuizTakers = userIdsOfQuizTakers[::-1]
846                             scoreForEachTake = scoreForEachTake[::-1]
847                             userNameList = userNameList[::-1]
848
849                             for i in range(len(userIdsOfQuizTakers)):
850                                 userIdNameScore = ("USER ID: " + userIdsOfQuizTakers[i] + "  " + "USERNAME: " + userNameList[i] + "  " + "SCORE: " + scoreForEachTake[i])
851                                 userIdNameScoreList.append(userIdNameScore)
852
853                             leaderBoardQuizTitle = select_query_executor("SELECT title FROM quiz WHERE id = " + quizSelection)
854
855                             time.sleep(0.5)
858                             print()
859                             print()
860                             print("TOP 50 PLAYERS WHO COMPLETED:", leaderBoardQuizTitle)
861                             print()
862
863                             for i in range(len(userIdsOfQuizTakers[:50])):
864                                 time.sleep(0.25)
865                                 print(str(i+1) + ". " + str(userIdNameScoreList[i]))
866
867                             print()
868                             goBackToLeaderboardsValid = False
869                             while goBackToLeaderboardsValid is False:
870                                 print("0. Go Back")
871                                 goBack = input()
872                                 if goBack == "0":
873                                     goBackToLeaderboardsValid = True
874                                     leaderboard_menu(signedInUserEmail, signedInUserId)
875
876                                 else:
877                                     print("Error! Must enter '0' to go back")
878                                     print()
879
880                 elif quizSelection == 0:
881                     quizLeaderboardSelectionValid = True
882                     leaderboard_menu(signedInUserEmail, signedInUserId)
883
884                 else:
885                     print("Error! That ID doesn't exist")
886                     print()
887
888         elif choice == "0":
889             break
890
891         else:
892             print("Error! Input must be an integer preceding one of the options above")
893             print()
894
895     user_menu(signedInUserEmail, signedInUserId)
896
897 # After the user has logged in with their MySQL account, the program runs by starting at the login_menu
898 login_menu()
899
```

Hashing_Algorithm.py

```
import hashlib, binascii, os

# Generates a salt for combining with the account password
def salt():
    salt = hashlib.sha256(os.urandom(60)).hexdigest()
    return salt

# Uses the hashing algorithm SHA-512 to create a hashed password,
# out of the account password and the salt
def hash_password(password, salt):
    salt = salt.encode('ascii')
    passwordHash = hashlib.pbkdf2_hmac('sha512', password.encode('utf-8'),
                   salt, 100000)
    passwordHash = binascii.hexlify(passwordHash)
    return(passwordHash).decode('ascii')

# Uses the retrieved salt and an entered password to create a hashed password
# If that hashed password matches the stored hashed password,
# the function returns True
def check_password(salt, storedPasswordHash, enteredPassword):
    passwordHash = hashlib.pbkdf2_hmac('sha512', enteredPassword.encode('utf-8'),
                   salt.encode('ascii'), 100000)
    passwordHash = binascii.hexlify(passwordHash).decode('ascii')
    return passwordHash == storedPasswordHash


#salt = salt()
#print(salt)
#storedPasswordHash = hash_password(input('Set your password: '), salt)
#print(storedPasswordHash)
#print(check_password(salt, storedPasswordHash, input('Enter your password: ')))

# Compares the entered password with the stored one

```

Quiz.py

```
import mysql.connector

# Database connection established
def quiz_connection(cnxUser, cnxPassword):
    global cnx
    global cursor
    cnx = mysql.connector.connect(user=cnxUser,
                                  password=cnxPassword,
                                  database='quiz')
    cursor = cnx.cursor()

# The class used for creating 'quiz' objects
class Quiz():

    # Each variable represents a field in the 'quiz' table in the database
    def __init__(self):
        self.quizId    = ""
        self.hostId    = ""
        self.title     = ""
        self.type      = ""
        self.maxScore  = ""
        self.content   = ""


    # The setters used for setting all the data for the object variables

    def set_quiz_id(self):
        quizId = cursor.lastrowid
        self.quizId = quizId

    def set_host_id(self, hostId):
        self.hostId = hostId

    def set_title(self, title):
        titleValid = False
        while titleValid is False:
            if len(title) > 75:
                print("Error! Limit of 75 characters!")
                title = input("Title: ")
            else:
                titleValid = True
        self.title = title
```

```
43
44      def set_quiz_type(self):
45          print()
46          print("Quiz type:")
47          print()
48          print(" 1. Multiple Choice")
49          print(" 2. Open-ended Answer")
50          print(" 3. Mixed")
51          print()
52          quizTypeValid = False
53          quizType = input("Select 1, 2 or 3: ")
54          while quizTypeValid is False:
55              if quizType == "1":
56                  quizTypeValid = True
57                  self.quizType = quizType
58              elif quizType == "2":
59                  quizTypeValid = True
60                  self.quizType = quizType
61              elif quizType == "3":
62                  quizTypeValid = True
63                  self.quizType = quizType
64              else:
65                  quizType = input("Error! Please enter 1, 2 or 3: ")
66
67      def set_max_score(self):
68          print()
69          numberOfQuestionsValid = False
70          while numberOfQuestionsValid is False:
71              while True:
72                  try:
73                      maxScore = int(input("Number of questions in your quiz [1-50]: "))
74                      break
75                  except ValueError:
76                      print("Error! Input must be an integer")
77
78              if maxScore > 0 and maxScore <= 50:
79                  numberOfQuestionsValid = True
80                  self.maxScore = maxScore
81              else:
82                  print("Error! Integer must be between 1 and 50")
83
84
85      def set_content(self, content):
86          valid = False
87          while valid is False:
88              if len(content) > 50:
89                  print("Error! Limit of 50 characters!")
90                  content = input("Quiz description: ")
91              else:
92                  valid = True
93                  break
94          self.content = content
95
```

```
87
88
89      # The getters, for retrieving the data from the object variables
90
91      def get_quiz_id(self):
92          return self.quizId
93
94      def get_hostId(self):
95          return self.hostId
96
97      def get_title(self):
98          return self.title
99
100     def get_quiz_type(self):
101         return self.quizType
102
103     def get_max_score(self):
104         return self.maxScore
105
106     def get_content(self):
107         return self.content
108
109
110     # This is the function which uses the object variables
111     # to add a quiz to the 'quiz' table
112
113     def add_quiz(self):
114         query = ("INSERT INTO quiz"
115                  "(hostId, title, type, maxScore, content) "
116                  "VALUES (%s, %s, %s, %s, %s)")
117         quizData = (self.hostId, self.title, self.quizType, self.maxScore, self.content)
118         cursor.execute(query, quizData)
119         cnx.commit()
120         emp_no = cursor.lastrowid
121         print()
122         print("Quiz created. Now for the questions.")
123
```

Quiz_Answer.py

```
import mysql.connector
from mysql.connector import errorcode

# Database connection established
def quiz_answer_connection(cnxUser, cnxPassword):
    global cnx
    global cursor
    cnx = mysql.connector.connect(user=cnxUser,
                                  password=cnxPassword,
                                  database='quiz')
    cursor = cnx.cursor()

# The class used for creating quiz answer objects
class Answer():

    # Each variable represents a field in the 'quiz_answer' table in the database
    def __init__(self):
        self.quizId           = ""
        self.questionId       = ""
        self.correct          = 0  # This variable can only store integers (1 or 0)
        self.answerContent    = ""


    # The setters used for setting all the data for the object variables

    def set_quiz_id(self, quizId):
        self.quizId = quizId

    def set_question_id(self, questionId):
        self.questionId = questionId

    def set_answer_content(self, answerContent):
        valid = False
        while valid is False:
            if len(answerContent) > 50:
                print("Error! Limit of 50 characters!")
                answerContent = input("Answer content: ")
            else:
                valid = True
                break
        self.answerContent = answerContent

    def set_correct(self):
        self.correct = 1

    def set_incorrect(self):
        self.correct = 0


```

```
50      def get_correct(self):
51          return self.correct
52
53      def get_answer_content(self):
54          return self.answerContent
55
56
57      # This is the function which uses the object variables
58      # to add a quiz to the 'quiz_answer' table
59
60      def add_answer(self):
61          query = ("INSERT INTO quiz_answer"
62                   "(quizId, questionId, correct, content) "
63                   "VALUES (%s, %s, %s, %s)")
64          answerData = (self.quizId, self.questionId, self.correct, self.answerContent)
65          cursor.execute(query, answerData)
66          cnx.commit()
67          emp_no = cursor.lastrowid
68
69
70  # This is a function used to determine whether a user wants to have
71  # 2 multiple choice option or 4
72  # The function vaidates the process and returns the number of chosen options
73  def multiple_choice_handling():
74      print()
75      numberOfOptionsValid = False
76      numberOfOptions = input("Number of options [2 or 4]: ")
77      while numberOfOptionsValid == False:
78          if numberOfOptions == "2":
79              numberOfOptionsValid = True
80              numberOfOptions = int(numberOfOptions)
81          elif numberOfOptions == "4":
82              numberOfOptionsValid = True
83              numberOfOptions = int(numberOfOptions)
84          else:
85              numberOfOptions = input("Number of options [2 or 4]: ")
86      return numberOfOptions
87
```

Quiz_Question.py

```
import mysql.connector
from mysql.connector import errorcode

# Database connection established
def quiz_question_connection(cnxUser, cnxPassword):
    global cnx
    global cursor
    cnx = mysql.connector.connect(user=cnxUser,
                                  password=cnxPassword,
                                  database='quiz')
    cursor = cnx.cursor()

# The class used for creating question objects
class Question():

    # Each variable represents a field in the 'quiz_question' table in the database
    def __init__(self):
        self.questionId       = ""
        self.quizId           = ""
        self.questionType     = ""
        self.level            = ""
        self.score            = ""
        self.questionContent  = ""


    # The setters used for setting all the data for the object variables

    def set_question_id(self):
        questionId = cursor.lastrowid
        self.questionId = questionId

    def set_quiz_id(self, quizId):
        self.quizId = quizId

    def set_question_type(self, questionType):
        self.questionType = questionType

    def set_level(self, level):
        levelValid = False
        while levelValid is False:
            if level == "1":
                levelValid = True
                self.level = level
            elif level == "2":
                levelValid = True
                self.level = level
            elif level == "3":
                levelValid = True
                self.level = level
            else:
                print("Error! Input 1, 2 or 3")
                print()
                level = input("Easy[1], Medium[2] or Hard[3]: ")

    def set_score(self):
        self.score = 1

    def set_question_content(self, content):
        valid = False
        while valid is False:
            if len(content) > 50:
                print("Error! Limit of 50 characters!")
                content = input("Question content: ")
            else:
                valid = True
                break
        self.questionContent = content
```

```
60
61
62      # The getters, for retrieving the data from the object variables
63
64      def get_question_id(self):
65          return self.questionId
66
67      def get_quiz_id(self):
68          return self.quizId
69
70      def get_question_type(self):
71          return self.questionType
72
73      def get_level(self):
74          return self.level
75
76      def get_score(self):
77          return self.score
78
79      def get_question_content(self):
80          return self.questionContent
81
82
83      # This is the function which uses the object variables
84      # to add a quiz question to the 'quiz_question' table
85
86      def add_question(self):
87          query = ("INSERT INTO quiz_question"
88                   "(quizId, type, level, score, content) "
89                   "VALUES (%s, %s, %s, %s, %s)")
90          questionData = (self.quizId, self.questionType, self.level, self.score,
91                          self.questionContent)
92          cursor.execute(query, questionData)
93          cnx.commit()
94          emp_no = cursor.lastrowid
95          print()
96          print("Your Question has been created")
97
```

SQL_Query_Executor.py

```
import mysql.connector
from mysql.connector import errorcode

# Database connection established
def sql_executor_connection(cnxUser, cnxPassword):
    global cnx
    global cursor
    cnx = mysql.connector.connect(user=cnxUser,
                                  password=cnxPassword,
                                  database='quiz')
    cursor = cnx.cursor()

# This function takes a select query as a parameter and executes it
# The select query should select one string or integer
# The value selected is then formatted so it can be used in other algorithms
def select_query_executor(query):
    cursor.execute(query)
    for data in cursor:
        data = ("{}".format(data))
        data = data.replace("'","")
        data = data.replace("(","")
        data = data.replace(")","")
        data = data.replace(",","")
    return data

# This function takes a select query as a parameter and executes it
# The select query should select more than one string
# All strings selected are formatted so they can be used in other algorithms
def select_all_strings_query_executor(query):
    cursor.execute(query)
    records = cursor.fetchall()
    recordList = []
    for record in records:
        record = ("{}".format(record))
        record = record[2:]
        record = record[:-3]
        recordList.append(record)
    return recordList

# This function takes a select query as a parameter and executes it
# The select query should select more than one integer
# All integers selected are formatted so they can be used in other algorithms
def select_all_integers_query_executor(query):
    cursor.execute(query)
    integers = cursor.fetchall()
    integerList = []
    for integer in integers:
        integer = str(integer)
        integer = integer.replace("(","")
        integer = integer.replace(")","")
        integer = integer.replace(",","")
        integerList.append(integer)
    return integerList

# This function is used for executing UPDATE and DELETE queries
def update_query_executor(query):
    cursor.execute(query)
    cnx.commit()
```

```
59
60  # This function retrieves all data about each quiz
61  # Each list represent a field in a table
62  # Since each field contains a values, the indexes should match up
63  # Therefore the data in index 5 of each list will be the data about quiz 5
64  # Full names are also retrieved, using the quiz's host ID
65  def select_all_quizzes():
66      availableRecordIds = []
67      availableHostIds = []
68      availableTitles = []
69      availableTypes = []
70      availableMaxScores = []
71      avaiableContents = []
72      query = "SELECT * FROM quiz"
73      cursor.execute(query)
74      quizTable = cursor.fetchall()
75      if quizTable == []:
76          print("No quizzes have been made yet")
77          print()
78      else:
79          for record in quizTable:
80              recordId = record[0]
81              recordHostId = record[1]
82              recordTitle = record[2]
83              recordType = record[3]
84              if recordType == 1:
85                  recordType = "Multiple Choice"
86
87              elif recordType == 2:
88                  recordType = "Open Ended"
89
90              elif recordType == 3:
91                  recordType = "Mixed"
92
93              recordMaxScore = record[4]
94              recordContent = record[5]
95              quizCreatorName = name_selector(recordHostId)
96              availableRecordIds.append(recordId)
97              availableHostIds.append(recordHostId)
98              availableTitles.append(recordTitle)
99              availableTypes.append(recordType)
100             availableMaxScores.append(recordMaxScore)
101             avaiableContents.append(recordContent)
102
103     return availableRecordIds, availableHostIds, availableTitles, availableTypes, availableMaxScores, avaiableContents
104
105 # This function retrieves all data about each quiz the logged in user has made
106 # Each list represent a field in a table
107 # Since each field contains a values, the indexes should match up
108 # Therefore the data in index 5 of each list will be the data about quiz 5
109 # Full names are also retrieved, using the quiz's host ID
110 # Since this function is only retrieving quizzes made by the logged in user,
111 # All quizzes have the same Host ID and therefore the same creator name
112 def select_users_quizzes(signedInUserId):
113     availableRecordIds = []
114     availableHostIds = []
115     availableTitles = []
116     availableTypes = []
117     availableMaxScores = []
118     avaiableContents = []
119     query = "SELECT * FROM quiz WHERE hostId = " + str(signedInUserId)
120     cursor.execute(query)
121     quizTable = cursor.fetchall()
122     if quizTable == []:
123         print("No quizzes have been made yet")
124         print()
125     else:
126         for record in quizTable:
127             recordId = record[0]
128             recordHostId = signedInUserId
129             recordTitle = record[2]
130             recordType = record[3]
131             if recordType == 1:
132                 recordType = "Multiple Choice"
133
134             elif recordType == 2:
135                 recordType = "Open Ended"
136
137             elif recordType == 3:
138                 recordType = "Mixed"
139
140             recordMaxScore = record[4]
141             recordContent = record[5]
142             quizCreatorName = name_selector(recordHostId)
143             availableRecordIds.append(recordId)
144             availableHostIds.append(recordHostId)
145             availableTitles.append(recordTitle)
146             availableTypes.append(recordType)
147             availableMaxScores.append(recordMaxScore)
148             avaiableContents.append(recordContent)
149
150     return availableRecordIds, availableHostIds, availableTitles, availableTypes, availableMaxScores, avaiableContents
```

```
151
152
153 # This function uses the id of a user to retireve the user's full name
154 # Their first name and last name are concatenated with a space in the middle,
155 # to create their full name
156 def name_selector(signedInUserId):
157     name = ""
158     query = "SELECT firstName, lastName FROM user WHERE id = %s"
159     cursor.execute(query, (signedInUserId,))
160     for firstName, lastName in cursor:
161         name = str(("{}".format(firstName) + " " + "{}".format(lastName)))
162     return name
163
```

Take.py

```
import mysql.connector
from mysql.connector import errorcode

# Database connection established
def take_connection(cnxUser, cnxPassword):
    global cnx
    global cursor
    cnx = mysql.connector.connect(user=cnxUser,
                                  password=cnxPassword,
                                  database='quiz')
    cursor = cnx.cursor()

# The class used for creating take objects
class Take():

    # Each variable represents a field in the 'take' table in the database
    def __init__(self):
        self.takeId  = ""
        self.userId  = ""
        self.quizId  = ""
        self.score   = 0    # The score variable only stores integers
        self.content = ""


    # The setters used for setting all the data for the object variables

    def set_take_id(self):
        takeId = cursor.lastrowid
        self.takeId = takeId

    def set_user_id(self, userId):
        self.userId = userId

    def set_quiz_id(self, quizId):
        self.quizId = quizId

    def set_score(self, score):
        self.score = score

    def set_content(self, content):
        self.content = content
```

```
42
43
44      # The getters, for retrieving the data from the object variables
45
46      def get_take_id(self):
47          return self.takeId
48
49      def get_user_id(self):
50          return self.userId
51
52      def get_quiz_id(self):
53          return self.quizId
54
55      def get_score(self):
56          return self.score
57
58      def get_content(self):
59          return self.content
60
61
62      # This is the function which uses the object variables
63      # to add a take to the 'take' table
64
65      def add_take(self):
66          query = ("INSERT INTO take"
67                   "(userId, quizId, score, content) "
68                   "VALUES (%s, %s, %s, %s)")
69          takeData = (self.userId, self.quizId, self.score, self.content)
70          cursor.execute(query, takeData)
71          cnx.commit()
72          emp_no = cursor.lastrowid
73          print()
74          print("You have begun taking the quiz")
75
```

Take_Answer.py

```
import mysql.connector
from mysql.connector import errorcode

# Database connection established
def take_answer_connection(cnxUser, cnxPassword):
    global cnx
    global cursor
    cnx = mysql.connector.connect(user=cnxUser,
                                  password=cnxPassword,
                                  database='quiz')
    cursor = cnx.cursor()

# The class used for creating take answers
class Take_Answer():

    # Each variable represents a field in the 'take_answer' table in the database
    def __init__(self):
        self.takeId               = ""
        self.questionId           = ""
        self.answerId             = ""
        self.takeAnswerContent    = ""


    # The setters used for setting all the data for the object variables

    def set_take_id(self, takeId):
        self.takeId = takeId

    def set_question_id(self, questionId):
        self.questionId = questionId

    def set_answer_id(self, answerId):
        self.answerId = answerId

    def set_answer_content(self, answerContent):
        valid = False
        while valid is False:
            if len(answerContent) > 50:
                print("Error! Limit of 50 characters!")
                answerContent = input("Enter your answer here: ")
            else:
                valid = True
                break
        self.answerContent = answerContent
```

```
37
38
39      # The getters, for retrieving the data from the object variables
40
41      def get_quiz_id(self):
42          return self.quizId
43
44      def get_question_id(self):
45          return self.questionId
46
47      def get_answer_id(self):
48          return self.answerId
49
50      def get_take_answer_content(self):
51          return self.takeAnswerContent
52
53
54      # This is the function which uses the object variables
55      # to add a take answer to the 'take_answer' table
56
57      def add_take_answer(self):
58          query = ("INSERT INTO take_answer"
59                   "(takeId, questionId, answerId, content) "
60                   "VALUES (%s, %s, %s, %s)")
61          takeAnswerData = (self.takeId, self.questionId, self.answerId, self.answerContent)
62          cursor.execute(query, takeAnswerData)
63          cnx.commit()
64          emp_no = cursor.lastrowid
65
```

User.py

```
1  import mysql.connector
2  import hashlib, binascii, os
3
4  # Database connection established
5  def user_connection(cnxUser, cnxPassword):
6      global cnx
7      global cursor
8      cnx = mysql.connector.connect(user=cnxUser,
9                                    password=cnxPassword,
10                                   database='quiz')
11     cursor = cnx.cursor()
12
13 # The class used for creating 'user' objects
14 class User():
15
16     # Each variable represents a field in the 'user' table in the database
17     def __init__(self):
18         self.userId       = ""
19         self.firstName    = ""
20         self.lastName     = ""
21         self.email        = ""
22         self.salt         = ""
23         self.passwordHash = ""
24         self.profile      = 0
25
26
27     # The setters used for setting all the data for the object variables
28
29     def set_user_id(self, email):
30         query = "SELECT id FROM user WHERE email = %s"
31         cursor.execute(query, (email,))
32         for selectedId in cursor:
33             selectedId = ("{}".format(selectedId))
34             selectedId = selectedId.replace("'","")
35             selectedId = selectedId.replace("(","")
36             selectedId = selectedId.replace(")","")
37             selectedId = selectedId.replace(",","")
38         self.userId  = selectedId
```

```
39
40      def set_first_name(self, firstName):
41          valid = False
42          while valid is False:
43              if not str(firstName).isalpha():
44                  print("Error! Only letters a-z allowed!")
45                  firstName = input("First Name: ")
46              elif len(firstName) > 50:
47                  print("Error! Limit of 50 characters!")
48                  firstName = input("First Name: ")
49              else:
50                  valid = True
51                  break
52          self.firstName = firstName
53
54      def set_last_name(self, lastName):
55          valid = False
56          while valid is False:
57              if not str(lastName).isalpha():
58                  print("Error! Only letters a-z allowed!")
59                  lastName = input("Last Name: ")
60              elif len(lastName) > 50:
61                  print("Error! Limit of 50 characters!")
62                  lastName = input("Last Name: ")
63              else:
64                  valid = True
65                  break
66          self.lastName = lastName
67
68      def set_email(self, email):
69          valid = False
70          while valid == False:
71              if "@" not in email:
72                  print("Error! Must include '@' symbol!")
73                  email = input("Email address: ")
74              elif len(email) > 50:
75                  print("Error! Limit of 50 characters!")
76                  email = input("Email address: ")
77              elif " " in email:
78                  print("Error! No spaces allowed!")
79                  email = input("Email address: ")
80              else:
81                  valid = True
82                  break
83          self.email = email
84
85      # The salt is set in the user class. If I were to instead call the 'salt()' function,
86      # there would be an UnboundLocalError
87      def set_salt(self):
88          salt = hashlib.sha256(os.urandom(60)).hexdigest()
89          self.salt = salt
90
91      # The hashed password is set in the user class. If I were to instead call the 'hash_password()' function,
92      # there would be an UnboundLocalError
93      # The entered password has to be at least 8 characters
94      def set_password_hash(self, password, salt):
95          valid = False
96          while valid is False:
97              if len(password) < 8:
98                  print("Error! Password must be at least 8 characters")
99                  print()
100                 password = input("Password: ")
101             else:
102                 valid = True
103         salt = salt.encode('ascii')
104         passwordHash = hashlib.pbkdf2_hmac('sha512', password.encode('utf-8'),
105                                            salt, 100000)
106         passwordHash = binascii.hexlify(passwordHash)
107         (passwordHash).decode('ascii')
108         self.passwordHash = passwordHash
109
110     def set_profile(self, profile):
111         self.profile = profile
112
113
114     # The getters, for retrieving the data from the object variables
115
116     def get_user_id(self):
117         return self.userId
118
119     def get_first_name(self):
120         return self.firstName
121
122     def get_last_name(self):
123         return self.lastName
124
125     def get_email(self):
126         return self.email
127
128     def get_salt(self):
129         return self.salt
130
131     def get_password_hash(self):
132         return self.passwordHash
```

```
133
134     def get_profile(self):
135         return self.profile
136
137     def get_user_salt(self):
138         return self.selectedSalt
139
140     def get_user_hashed_password(self):
141         return self.selectedPassword
142
143     # This is the function which uses the object variables
144     # to add a user to the 'user' table
145
146     def add_user(self):
147         query = ("INSERT INTO user"
148                  "(firstName, lastName, email, salt, passwordHash, profile) "
149                  "VALUES (%s, %s, %s, %s, %s, %s)")
150         userData = (self.firstName, self.lastName, self.email, self.salt,
151                     self.passwordHash, self.profile)
152         cursor.execute(query, userData)
153         cnx.commit()
154         emp_no = cursor.lastrowid
155         print("Your Account has been created")
156
```

Completeness

Although I met all my objectives, an additional objective could have been set when considering the graphical user interface.

The client stated that a more attractive user interface would make the system easier to operate and more fun to play.

I would have liked to learn how to use python's inbuilt GUI library 'Tkinter'. By utilizing Tkinter, I could have developed an attractive graphical user interface which could encourage more quizzers to use the system.

Creating a GUI with Tkinter would have refined some features of my system.

- Instead of text moving up and off the screen (if the screen fills up) it would be displayed in a box with a scroll bar.
- 'Go Back' buttons would remove the need to type in '0' to return to the previous screen
- Less validation would be needed as some inputs would be tied to buttons.
- The title of each screen would be at the top-middle of the screen instead of at the bottom.
- More attractive colours could have been used. The limitations of the shell are essentially stripped away.
- Error messages could have appeared in a message box, notifying the user instead of just telling them through text.
- Images, videos, and sounds could have been part of question contents.

Testing

Test Plan

To ensure that all the system's algorithms function correctly, and that all user inputs (correct and incorrect) are handled, I have planned to test the system.

To meet my objectives

- All inputs must be validated and not cause crashes
- All inputs that go into the database must be successfully added to the database

Test Plan Table

Test ID	Description	Test Data	Expected Results
Testing that a user can sign in with their MySQL user, to connect to the database			
01	Check the user can enter the correct details to connect to the database. Correct Inputs.	root Isaac250mysql!	• The user should be accepted into the database • User taken to the login menu.
02	Check the user can enter the correct details to connect to the database. Incorrect Inputs.	Groot Pineapple300	• The user is denied access to the database and the system. • The user is shown an error message and prompted to enter the details again.
03	Check the user can enter the correct details to connect to the database. Empty field is entered.	[None] Isaac250mysql!	• The user is denied access to the database and the system. • The user is shown an error message and prompted to enter the details again.
Testing that the user can enter a value corresponding to an action at the login menu			
04	Check the user has entered either "1", "2", or "0"	"1"	• The user is taken through to the login process
05	Check the user has entered either "1", "2", or "0"	"2"	• The user is taken through to the sign-up process
06	Check the user has entered either "1", "2", or "0"	"0"	• The user exits the program, and a message box pops up • Message box asks "Your program is still running! Do you want to kill it?" • If "yes" is selected, the shell window closes • If "cancel" is selected, the shell window remains open but the program ends

07	Check the user has entered either “1”, “2”, or “0”	“F”	• The user is prompted to type either “1”, “2” or “0” again
08	Check the user has entered either “1”, “2”, or “0”	[None]	• The user is prompted to type either “1”, “2” or “0” again
Testing that the user can login to the system			
09	Check that the email and password are valid. Correct inputs	[Make sure the user with email and password “test@” and “test” has been created on the system beforehand] test@ testpassword	• The user is greeted by a welcome message • “Welcome, [first name + last name] • The user is then taken to the main menu
10	Check that the email and password are valid. Incorrect data entered (no email)	[None]	• The user is told that there are no accounts with that email • The user is taken back to the login menu
11	Check that the email and password are valid. Incorrect data entered (no password)	test@ [None]	• The user is told that that is the incorrect password for that email address • The user is taken back to the login menu
12	Check that the email and password are valid. Incorrect data entered (email does not have “@” symbol)	test	• The user is told that the email must contain an “@” symbol • The user is prompted to enter the email again
Testing that a new user can register on the system			
13	Check the user can register an account. Correct Inputs	Barry White Barry@White.com Barrysecurepassword56!	• The user is told their account has been created • They are taken to the main menu
14	Check if all fields are entered	[None] White Barry@White.com Barrysecurepassword56!	• If the username is blank, the user is told that only letters a-z are allowed • They are then prompted to enter the field in again • This is the same for the last name • The user is told that the email must contain an “@” symbol if the email is left blank • The user is prompted to enter the email again • The password must be at least 8 characters • The user is prompted to enter the password again

15	Check if fields meet length requirements	A first name, last name, or email longer than 50 characters	• The system provides an error message which tells the user to enter data which is less than 50 characters long • The user is prompted to input data again
16	Check if fields meet length requirements	[A password with length 7 or under] Cake23	• The system provides an error message which tells the user to enter a password which has a minimum of 8 characters • The user is prompted to input data again
Testing that a user can take a quiz			
17	Check that the user is provided with options of all the created quizzes	[The user has selected the take quiz option from the main menu] 1	• The user is presented with every quiz in the quiz table
18	Check the user can select a quiz. Correct ID entered	[Assuming there is already a quiz created with the ID "4"] 4	• The user is asked if they are sure they want to take the quiz
19	Check if the user wishes to go back to the main menu (If the quiz ID "0" is entered) Correct Inputs	0	• The user is returned to the main menu
20	Check the user can select a quiz. Not an ID entered	[Quiz IDs can only be integers] @sdasd'q2	• The user is told that this isn't an ID
21	Check the user can select a quiz. Incorrect ID entered	[Assuming there is no quiz with the ID 89] 89	• The user is told that the ID entered is not in the list of available quizzes
22	Check the user is sure they want to take the quiz. Correct inputs	[Not case sensitive] Y	• The user is then asked if they want to randomize the question order
23	Check the user is sure they want to take the quiz. Correct inputs	[Not case sensitive] N	• The user is taken back to the take quiz menu • All the available quizzes are displayed again
24	Check the user is sure they want to take the quiz. Incorrect inputs	[None] or not Y or not N None	• The user is prompted again to enter either Y or N until Y or N is entered
25	Check if the user wants to randomize question order. Correct inputs	[Not case sensitive] Y	• Question order is randomized • The quiz begins and the first question is displayed

26	Check if the user wants to randomize question order. Correct inputs	[Not case sensitive] N	• The quiz begins and the first question is displayed
27	Check if the user wants to randomize question order. Incorrect inputs	[None] or not Y or not N None	• The user is prompted again to enter either Y or N until Y or N is entered
28	Check that each question in the selected quiz is displayed and the user has answered each one	User inputs for each question	• Each question is displayed one at a time • The user must enter their answer before the next question is displayed
29	Check if the question is multiple-choice (MC), the options [either 2 or 4] are displayed in a random order	User inputs for each question	• Each MC question has its choices displayed in a random order each take
30	Check if the user's input for selecting a choice (for an MC question) is valid. Correct inputs	[Answer is option 3] 3	• The user is presented with a message that says "correct" • One point is added to their score • The next question is displayed
31	Check if the user's input for selecting a choice (for an MC question) is valid. Incorrect inputs	[Answer is option 1] 1	• The user is presented with a message that says "incorrect" • The user's score stays the same • The next question is displayed
32	Check if the user's input for selecting a choice (for an MC question) is valid. Incorrect inputs	The input is not 1,2,3 or 4 D56gsfe## 54	• If the user has entered a non-integer, they are told to enter an integer and asked for the input again • If the user has entered an integer but not one of the integers preceding an option, they are asked to enter an integer preceding an option. They are asked for the input again
33	Check if the open-ended (OE) question is valid. Correct inputs	[Answer is "Blue Whale"] [Not case sensitive] Blue Whale	• The user is told that they are correct • The user's score increases by one • The next question is displayed (or quiz ends if it was the last question)
34	Check if the open-ended (OE) question is valid. Incorrect inputs	[Answer is "Blue Whale"] [Not case sensitive] Not Blue Whale [The incorrect answer could be anything] Moby Dick	• The user is told that they are incorrect • The user's score stays the same • The next question is displayed (or quiz ends if it was the last question)

35	Check when the quiz is finished. Correct Inputs	Inputs for all the questions in the quiz	• The user is told the quiz is finished • Their score for the quiz is displayed • Their new, increased profile quiz points is also displayed • All data collected from the take has been recorded and stored in the take table and the take_answer table • The user is then taken back to the Main Menu
Testing that a user can create a quiz			
36	Check that the user is provided an option to create a quiz. Correct Inputs	[The user has selected the create quiz option from the main menu] 2	• The user is put through the quiz making process • This starts with the user being asked to enter the quiz title
37	Check the entered quiz title is valid. Correct inputs	[a string of characters less than 75 characters] General knowledge	• The title is saved to an object variable • The user is then asked what type their quiz is
38	Check the entered quiz title is valid. Incorrect inputs	[a string of characters more than 75 characters]	• Error message saying the title is too long • User is prompted to enter the title again
39	Check the entered quiz type is valid. Correct inputs	1, 2 or 3	• 1 – MC type quiz • 2 – OE type quiz • 3 – Mixed type quiz • Type is saved to an object variable • The user is then asked what the max score of their quiz will be
40	Check the entered quiz type is valid. Incorrect inputs	Not 1, not 2 or not 3 34	• Error, please enter 1, 2 or 3 • User prompted to enter the quiz type again
41	Check the entered Max Score is valid. Correct Inputs	[A value between 1 to 50] 2	• The max score is saved to an object variable • The user is then asked to create a question the number of times equal to the max score • Since each question is always worth one point, the max score is equivalent to the number of questions per quiz • The user is then asked to enter the quiz description
42	Check the entered Max Score is valid. Incorrect inputs	[A value outside 1 to 50 or a non-integer] 100 hello	• Error, integer must be between 1 and 50 • Error, input must be an integer • The user is prompted to enter the max score again

43	Check the quiz description	[Quiz description can be anything] A quite interesting quiz	• The quiz description is saved to an object variable • Every variable is now filled so the quiz can now be added to the database • The user is then prompted to create the questions
Testing that a user can create a question			
44	Check the entered question difficulty is valid. Correct inputs	1, 2 or 3	• This process repeats for a number which is equal to max score • 1 – Easy • 2 – Medium • 3 – Hard • The difficulty level is stored in an object variable • The user is then asked to enter the question content
45	Check the entered question difficulty is valid. Incorrect inputs	Not 1, not 2 or not 3 Hard	• Error, input 1, 2 or 3 • The user is prompted to enter the difficulty level again
46	Check the entered question content. Correct inputs	[Question content can be anything]	• The question content is stored in an object variable • All question object variables are satisfied so the quiz can be added to the database • The user is then asked to create an answer for this question • MC – create multiple answers • OE – create one answer
Test that a user can create an answer			
47	If OE check the answer content for one input If MC check the answer input for all the choices. Either 2 or 4 Correct inputs	[Answer content can be anything]	• If OE, the one answer inputted is set as the correct answer • The OE answer content is stored to an object variable • The OE answer is added to the database • The system asks for either 2 or 4 inputs for answer contents. • The first input will always be the correct answer • The rest are incorrect • The answer contents and their corresponding correct values are stored to an object variable • They are then added to the database • The quiz creation process is then over • The user is returned to the main menu

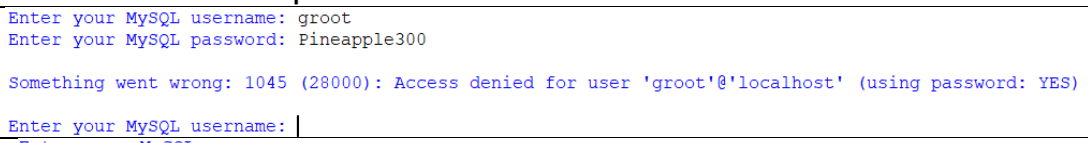
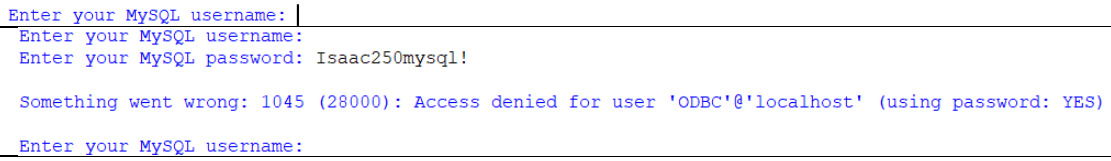
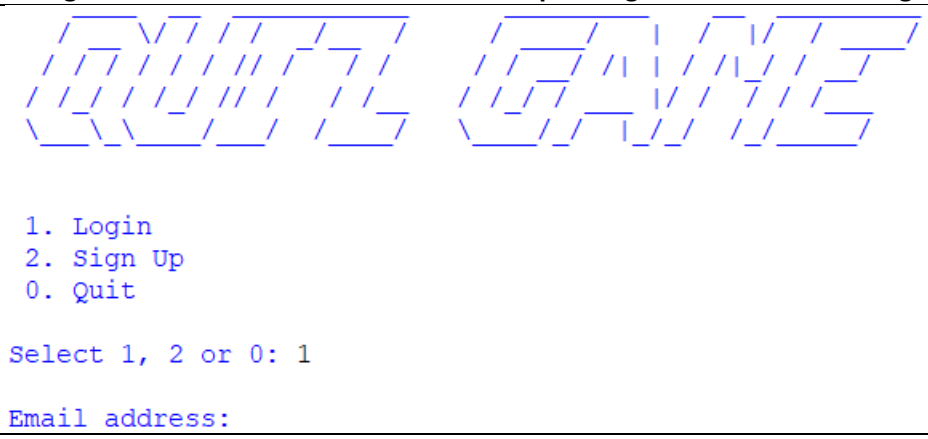
Test that a user can delete a quiz			
48	Check that the user is provided with options of the created quizzes they have made. Correct Inputs	[The user has selected the delete quiz option from the main menu] 3	• The user is presented with the quizzes they have made, retrieved from the quiz table
49	Check the user can select a quiz to delete. Correct ID entered	[Assuming there is already a quiz created with the ID "50"] 50	• The user is asked if they are sure they want to delete the quiz
50	Check if the user wishes to go back to the main menu (If the quiz ID "0" is entered)	0	• The user is returned to the main menu
51	Check the user can select a quiz. Not an ID entered	[Quiz IDs can only be integers] @sdasd'q2	• The user is told that this isn't an ID
52	Check the user can select a quiz. Incorrect ID entered	[Assuming there is no quiz with the ID 64] 64	• The user is told that the ID entered is not in the list of available quizzes
53	Check the user is sure they want to delete the quiz. Correct inputs	[Not case sensitive] Y	• All the takes of the quiz and the quiz itself are deleted
54	Check the user is sure they want to delete the quiz. Correct Inputs	[Not case sensitive] N	• The user is taken back to the delete quiz menu • All the available quizzes are displayed again
55	Check the user is sure they want to delete the quiz. Incorrect inputs	[None] or not Y or not N YES!	• The user is prompted again to enter either Y or N until Y or N is entered
Test that a user can view their own score			
56	Check that the user can view their own quiz points Correct inputs	[The user has selected the Profile option from the main menu] 4	• The user is presented with their total quiz points earned • They are then given the option to return to the main menu
57	Check that the user wishes to go back to the main menu. Correct inputs	0	• The user has been returned to the main menu
58	Check that the user wishes to go back to the main menu. Incorrect inputs	Not 0 Go back please	• The user is prompted with the option to go back to the main menu again

Test that a user can view another user's scores through a leader board			
59	Check that the user can view another user's score through a leader board Correct inputs	[The user has selected the Leader boards option from the main menu] 5	• The user is brought to the leader board menu • The user is presented with an option to view an overall leader board, view a leader board per quiz or go back to the main menu
60	Check that the user wants to view the overall leader board. Correct inputs	1	• A leader board with the top 50 players ranked from best to worst will be displayed • The user is then given the option to go back to the leader board menu (0. Go back)
61	Check that the user wants to view the leader boards per quiz. Correct inputs	2	• All quizzes are displayed • The user must select a quiz ID or select "0" to go back
62	Check that the user wants to go back to the main menu. Correct inputs	0	• The user is returned to the main menu
63	Check that the user wants to make a valid decision at the leader board menu Incorrect inputs	Not 1, not 2, or not 0 49	• The user is prompted to enter 1, 2 or 0 again
64	Check if the user wants to go back to the leader board menu from the overall leader board. Correct inputs	0	• The user is returned to the leader board menu
65	Check if the user wants to go back to the leader board menu from the overall leader board. Incorrect inputs	Not 0 Sw3d3	• The user is prompted to enter the go back input again
Testing that a user can view leader boards per quiz			
66	Check that the user wants to view the leader boards per quiz. Correct inputs	2	• All quizzes are displayed • The user must select a quiz ID or select "0" to go back
67	Check that the user wants to view the leader boards per quiz. Incorrect inputs	[Not 2] 567ytyh*	• The user is prompted to enter their input again • Enter either 1,2 or 0
68	Check the user can select a quiz from which they can view a leader board from. Correct ID entered	[Assuming there is already a quiz created with the ID "48"] 48	• The user is shown the top 50 takes for the selected quiz • They are then asked if they would like to return to the leader board menu

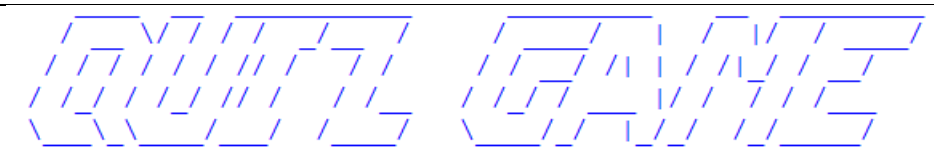
69	Check the user can select a quiz. Not an ID entered	[Quiz IDs can only be integers] @sdasd'q2	• The user is told that this isn't an ID
70	Check the user can select a quiz from which they can view a leader board from. Incorrect ID entered	[Assuming there is no quiz with the ID 89] 89	• The user is told that the ID entered is not in the list of available quizzes
71	Check if the user wishes to go back to the main menu from the leader board menu (If the quiz ID "0" is entered) Correct inputs	0	• The user is returned to the main menu

Testing Evidence

The following table is an overview of the basic input and output test conducted on the system. This is to make sure that the expected outcome of every action is the actual outcome when using the system.

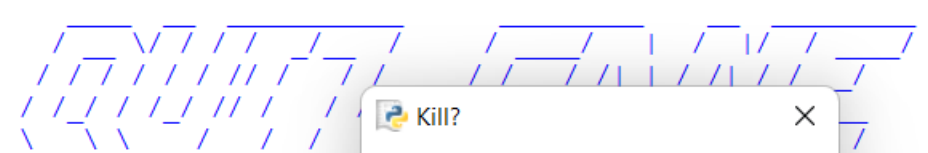
Test ID	Actual Outcome
Testing that a user can sign in with their MySQL user, to connect to the database	
01	
02	
03	
Testing that the user can enter a value corresponding to an action at the login menu	
04	

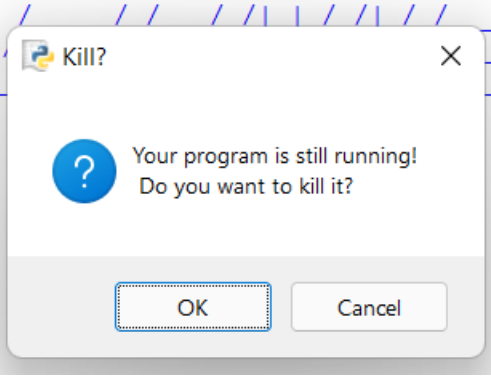
05



```
1. Login
2. Sign Up
0. Quit

Select 1, 2 or 0: 2

First Name:
```

06



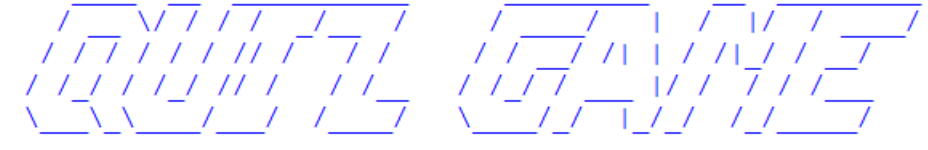



```
1. Login
2. Sign Up
0. Quit

Select 1, 2 or 0: 0
```

07

```
1. Login
2. Sign Up
0. Quit

Select 1, 2 or 0: F
Select 1, 2 or 0:
```

08



```
1. Login
2. Sign Up
0. Quit

Select 1, 2 or 0:
Select 1, 2 or 0:
```

Testing that the user can login to the system	
09	Email address: test@ Password: testpassword Login Successful Welcome, TestFirstName TestLastName  1. Take A Quiz 2. Create A Quiz 3. Delete A Quiz 4. View Profile 5. View Leaderboards 0. Logout Select 1, 2, 3, 4, 5 or 0:
10	Email address: Error! Must include '@' symbol! Email address:
11	Email address: test@ Password: Error! Incorrect password  1. Login 2. Sign Up 0. Quit Select 1, 2 or 0:
12	Email address: test Error! Must include '@' symbol! Email address:

Testing that a new user can register on the system

13

```
First Name: Barry
Last Name: White
Email address: Barry@White.com
Password: Barrysecurepassword56!
Your Account has been created


    __  ___      _            __  ___                
   /  |/  /___ _(_)___       /  |/  /__  ____  __  __
  / /|_/ / __ `/ / __ \     / /|_/ / _ \/ __ \/ / / /
 / /  / / /_/ / / / / /    / /  / /  __/ / / / /_/ / 
/_/  /_/\__,_/_/_/ /_/    /_/  /_/\___/_/ /_/\__,_/  


 1. Take A Quiz
 2. Create A Quiz
 3. Delete A Quiz
 4. View Profile
 5. View Leaderboards
 0. Logout

Select 1, 2, 3, 4, 5 or 0:
```

14

```
First Name:
Error! Only letters a-z allowed!
First Name: |
```

15

```
First Name: mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
Error! Limit of 50 characters!
First Name:
```

16

```
Password: Cake23
Error! Password must be at least 8 characters

Password:
```

Testing that a user can take a quiz	
17	Select 1, 2, 3, 4, 5 or 0: 1 ID: 4 Quiz Title: Music Quiz Created by: s s Question Type: Mixed Maximum Score: 3 Description: A Quiz for Guy and Shay ID: 25 Quiz Title: Music Quiz Created by: Isaac Patrickson Question Type: Multiple Choice Maximum Score: 1 Description: A quiz about popular music ID: 31 Quiz Title: Science Quiz Created by: s s Question Type: Open Ended Maximum Score: 1 Description: A quiz about general science knowledge
18	Select a quiz ID or '0' to go back: 4 Are you sure you want to take this quiz? [Y/N]: Music Quiz
19	Select a quiz ID or '0' to go back: 0 1. Take A Quiz 2. Create A Quiz 3. Delete A Quiz 4. View Profile 5. View Leaderboards 0. Logout Select 1, 2, 3, 4, 5 or 0:
20	Select a quiz ID or '0' to go back: @sdasd'q2 Errror! That's not an ID Select a quiz ID or '0' to go back:

21	Select a quiz ID or '0' to go back: 89 Error! Please select an existing ID Select a quiz ID or '0' to go back:
22	Are you sure you want to take this quiz? [Y/N]: Music Quiz y Would you like to randomise the question order? [Y/N]
23	Are you sure you want to take this quiz? [Y/N]: Music Quiz n ID: 4 Quiz Title: Music Quiz Created by: s s Question Type: Mixed Maximum Score: 3 Description: A Quiz for Guy and Shay ID: 25 Quiz Title: Music Quiz Created by: Isaac Patrickson Question Type: Multiple Choice Maximum Score: 1 Description: A quiz about popular music
24	Are you sure you want to take this quiz? [Y/N]: Music Quiz Error! Please enter Y or N Are you sure you want to take this quiz? [Y/N]: Music Quiz
25	Would you like to randomise the question order? [Y/N] y You have begun taking the quiz Music Quiz Description: A Quiz for Guy and Shay Question 1 What year did Kanye release the song Runaway? 1. 2011 2. 2013 3. 2012 4. 2010

26	Would you like to randomise the question order? [Y/N] n You have begun taking the quiz Music Quiz Description: A Quiz for Guy and Shay Question 1 Who is the lead singer of the Arctic Monkeys? 1. Alex Turner 2. Miles Kane
27	Would you like to randomise the question order? [Y/N] Error! Please enter Y or N Would you like to randomise the question order? [Y/N]

28

```
You have begun taking the quiz

Music Quiz
Description: A Quiz for Guy and Shay


Question 1
Who is the lead singer of the Arctic Monkeys?

1. Alex Turner
2. Miles Kane

1

Correct


Question 2
What year did Kanye release the song Runaway?

1. 2013
2. 2012
3. 2010
4. 2011

3

Correct


Question 3
What year was 2 pac shot?


1996

Correct

Take Finished!
You scored 3 out of 3

Your total Quiz Points is now: 61
```

29

```
Question 1
Who is the lead singer of the Arctic Monkeys?

1. Alex Turner
2. Miles Kane

Question 2
What year did Kanye release the song Runaway?

1. 2013
2. 2012
3. 2010
4. 2011
```

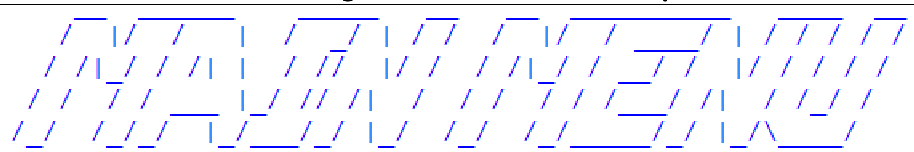
30	Question 2 What year did Kanye release the song Runaway? 1. 2013 2. 2012 3. 2010 4. 2011 3 Correct Question 3 What year was 2 pac shot?
31	Question 2 What year did Kanye release the song Runaway? 1. 2012 2. 2013 3. 2010 4. 2011 2 Incorrect Question 3 What year was 2 pac shot?
32	Question 2 What year did Kanye release the song Runaway? 1. 2010 2. 2013 3. 2012 4. 2011 54 Error! Input must be an integer preceding one of the options above
33	Question 1 What is the largest mammal to have ever lived? Blue Whale Correct Take Finished! You scored 1 out of 1
34	Question 1 What is the largest mammal to have ever lived? Enter your answer here: Moby Dick Incorrect Take Finished! You scored 0 out of 1

<table>
<tr><td>35</td><td>Take Finished!
You scored 0 out of 1

Your total Quiz Points is now: 72

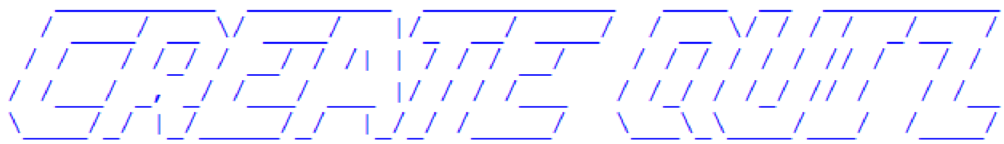
 

1. Take A Quiz
2. Create A Quiz
3. Delete A Quiz
4. View Profile
5. View Leaderboards
0. Logout

Select 1, 2, 3, 4, 5 or 0:</td></tr>
<tr><td colspan="2">Testing that a user can create a quiz</td></tr>
<tr><td>36</td><td> 

1. Take A Quiz
2. Create A Quiz
3. Delete A Quiz
4. View Profile
5. View Leaderboards
0. Logout

Select 1, 2, 3, 4, 5 or 0: 2

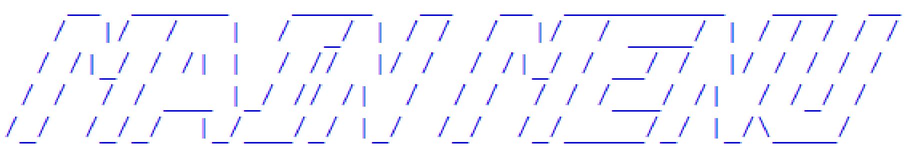
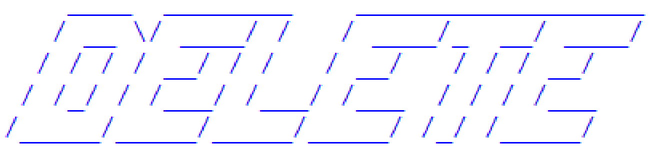
 

Quiz title:</td></tr>
<tr><td>37</td><td>Quiz title: General knowledge

Quiz type:

1. Multiple Choice
2. Open-ended Answer
3. Mixed</td></tr>
<tr><td>38</td><td>Quiz title: mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
Error! Limit of 75 characters!
Title:</td></tr>
</table>

39	Quiz type: 1. Multiple Choice 2. Open-ended Answer 3. Mixed Select 1, 2 or 3: 2 Number of questions in your quiz [1-50]:
40	Quiz type: 1. Multiple Choice 2. Open-ended Answer 3. Mixed Select 1, 2 or 3: 34 Error! Please enter 1, 2 or 3:
41	Number of questions in your quiz [1-50]: 2 Quiz description:
42	Number of questions in your quiz [1-50]: 100 Error! Integer must be between 1 and 50 Number of questions in your quiz [1-50]: hello Error! Input must be an integer Number of questions in your quiz [1-50]:
43	Quiz description: A quite interesting quiz Quiz created. Now for the questions.
Testing that a user can create a question	
44	Question 1 Easy[1], Medium[2] or Hard[3]: 1 Question content:
45	Question 1 Easy[1], Medium[2] or Hard[3]: Hard Error! Input 1, 2 or 3 Easy[1], Medium[2] or Hard[3]:
46	Question content: Who is regarded as the father of computing? Your Question has been created Multiple choice question or Open ended [M/O]:
Test that a user can create an answer	
47	Question content: Which war was the enigma code used in? Your Question has been created Multiple choice question or Open ended [M/O]: O Answer Content: World War 2

	Question content: Who is regarded as the father of computing? Your Question has been created Multiple choice question or Open ended [M/O]: m Number of options [2 or 4]: 2 Option 1 You can only have one correct option. Enter your correct option first. Content: Charles Babbage Option 1 has been created Option 2 Enter your incorrect option(s) now. Content: Bill Gates Option 2 has been created
Test that the user can delete a quiz	
48	 1. Take A Quiz 2. Create A Quiz 3. Delete A Quiz 4. View Profile 5. View Leaderboards 0. Logout Select 1, 2, 3, 4, 5 or 0: 3  Select a quiz you want to delete YOU CAN ONLY DELETE THE QUIZZES YOU HAVE MADE ID: 4 Quiz Title: Music Quiz Created by: s s Question Type: Mixed Maximum Score: 3 Description: A Quiz for Guy and Shay ID: 31 Quiz Title: Science Quiz Created by: s s Question Type: Open Ended Maximum Score: 1 Description: A quiz about general science knowledge

49	ID: 50 Quiz Title: Tester Created by: s s Question Type: Mixed Maximum Score: 8 Description: tester Select a quiz ID or '0' to go back: 50 Are you sure you want to delete this quiz [Y/N]: Tester
50	Select a quiz ID or '0' to go back: 0  1. Take A Quiz 2. Create A Quiz 3. Delete A Quiz 4. View Profile 5. View Leaderboards 0. Logout Select 1, 2, 3, 4, 5 or 0:
51	Select a quiz ID or '0' to go back: @sdasd'q2 Error! Please enter an integer Select a quiz ID or '0' to go back:
52	Select a quiz ID or '0' to go back: 64 Error! That ID doesn't exist Select a quiz ID or '0' to go back:

53

```
Are you sure you want to delete this quiz [Y/N]:  Tester
y

Your Quiz has been deleted
```

```
Select a quiz you want to delete

YOU CAN ONLY DELETE THE QUIZZES YOU HAVE MADE

ID: 4
Quiz Title: Music Quiz
Created by: s s
Question Type: Mixed
Maximum Score: 3
Description: A Quiz for Guy and Shay



ID: 31
Quiz Title: Science Quiz
Created by: s s
Question Type: Open Ended
Maximum Score: 1
Description: A quiz about general science knowledge
```

54

```
Are you sure you want to delete this quiz [Y/N]:  Blue Whale Quiz
n
You Quiz has not been deleted
```

```
Select a quiz you want to delete

YOU CAN ONLY DELETE THE QUIZZES YOU HAVE MADE

ID: 4
Quiz Title: Music Quiz
Created by: s s
Question Type: Mixed
Maximum Score: 3
Description: A Quiz for Guy and Shay



ID: 31
Quiz Title: Science Quiz
Created by: s s
Question Type: Open Ended
Maximum Score: 1
Description: A quiz about general science knowledge
```

55

```
Are you sure you want to delete this quiz [Y/N]:  Blue Whale Quiz
YES!
Error! Please enter either 'Y' or 'N'

Are you sure you want to delete this quiz [Y/N]:  Blue Whale Quiz
```

Test that a user can view their own score

56

```
 1. Take A Quiz
 2. Create A Quiz
 3. Delete A Quiz
 4. View Profile
 5. View Leaderboards
 0. Logout

Select 1, 2, 3, 4, 5 or 0: 4

Total Quiz Points Earned: 72

0. Go Back
```

57

```
0. Go Back
0
```

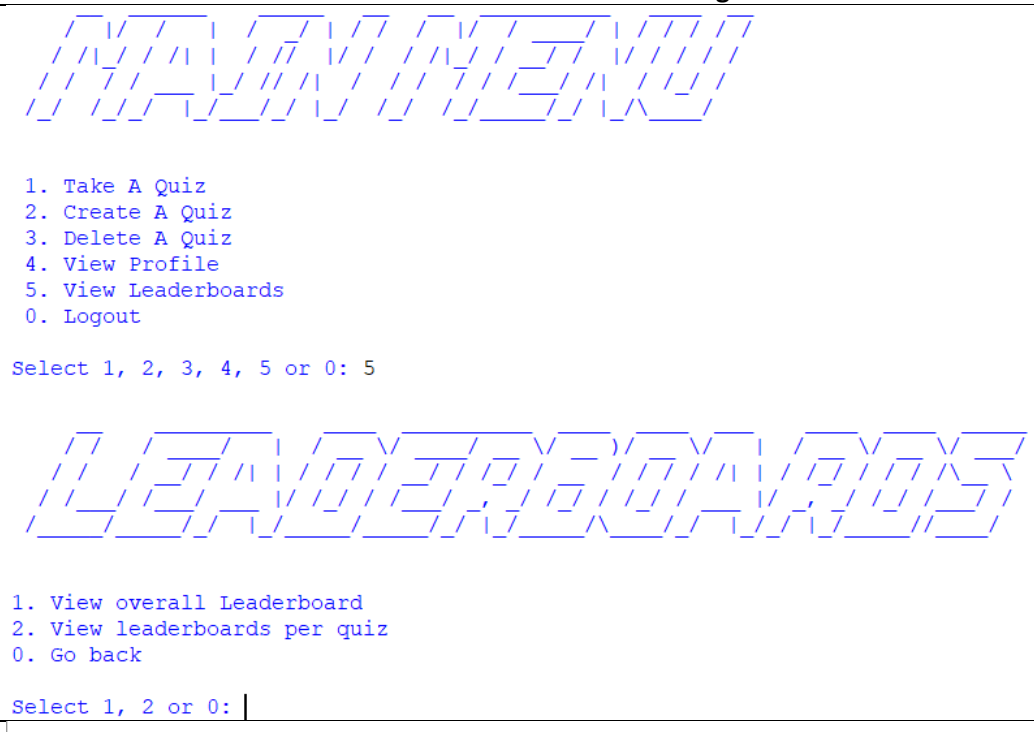
```
 1. Take A Quiz
 2. Create A Quiz
 3. Delete A Quiz
 4. View Profile
 5. View Leaderboards
 0. Logout

Select 1, 2, 3, 4, 5 or 0:
```

58

```
0. Go Back
Go back please
Error! Must type '0' to go back to the Menu

0. Go Back
```

Test that a user can view another user's scores through a leader board

59

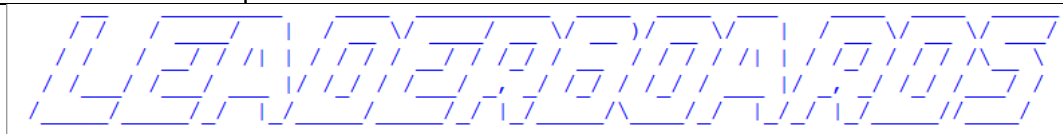


```
 1. Take A Quiz
 2. Create A Quiz
 3. Delete A Quiz
 4. View Profile
 5. View Leaderboards
 0. Logout

Select 1, 2, 3, 4, 5 or 0: 5
```



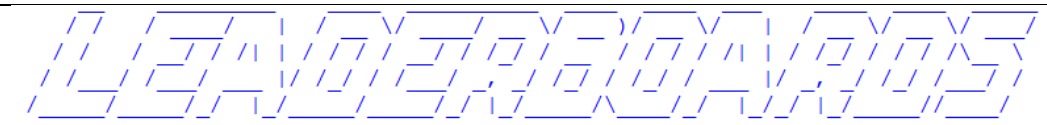
```
1. View overall Leaderboard
2. View leaderboards per quiz
0. Go back

Select 1, 2 or 0:
```

60



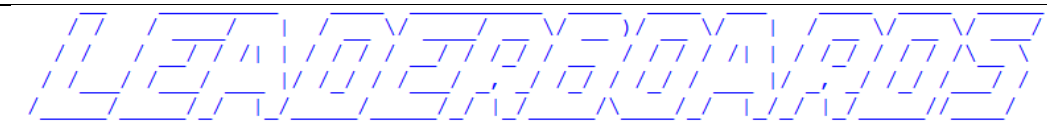
```
1. View overall Leaderboard
2. View leaderboards per quiz
0. Go back

Select 1, 2 or 0: 1


TOP 50 PLAYERS

1. USER ID: 1  USERNAME: s s  SCORE: 72
2. USER ID: 6  USERNAME: Isaac Patrickson  SCORE: 3
3. USER ID: 2  USERNAME: Guy Johnson  SCORE: 3
4. USER ID: 3  USERNAME: Shay Oleary  SCORE: 2
5. USER ID: 9  USERNAME: Jane Doe  SCORE: 0
6. USER ID: 8  USERNAME: John Doe  SCORE: 0
7. USER ID: 7  USERNAME: John Doe  SCORE: 0
8. USER ID: 5  USERNAME: Helleni Tsentas  SCORE: 0
9. USER ID: 28  USERNAME: Barry White  SCORE: 0
10. USER ID: 26  USERNAME: TestFirstName TestLastName  SCORE: 0
11. USER ID: 23  USERNAME: Clive Morrow  SCORE: 0
12. USER ID: 21  USERNAME: Ryan Johnson  SCORE: 0
13. USER ID: 19  USERNAME: George Harrison  SCORE: 0
14. USER ID: 18  USERNAME: Zack Monroe  SCORE: 0
15. USER ID: 17  USERNAME: Zack Monroe  SCORE: 0
16. USER ID: 16  USERNAME: Charlotte Pollard  SCORE: 0
17. USER ID: 10  USERNAME: Forest Louis  SCORE: 0

0. Go Back
```

61



```
1. View overall Leaderboard
2. View leaderboards per quiz
0. Go back

Select 1, 2 or 0: 2

ID: 4
Quiz Title: Music Quiz
Created by: s s
Question Type: Mixed
Maximum Score: 3
Description: A Quiz for Guy and Shay



ID: 25
Quiz Title: Music Quiz
Created by: Isaac Patrickson
Question Type: Multiple Choice
Maximum Score: 1
Description: A quiz about popular music



ID: 31
Quiz Title: Science Quiz
Created by: s s
Question Type: Open Ended
Maximum Score: 1
Description: A quiz about general science knowledge
```

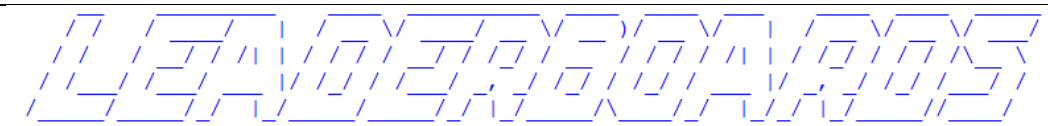
62



```
1. View overall Leaderboard
2. View leaderboards per quiz
0. Go back

Select 1, 2 or 0: 0
```

```
 1. Take A Quiz
 2. Create A Quiz
 3. Delete A Quiz
 4. View Profile
 5. View Leaderboards
 0. Logout

Select 1, 2, 3, 4, 5 or 0:
```

63



```
1. View overall Leaderboard
2. View leaderboards per quiz
0. Go back

Select 1, 2 or 0: 49
Select 1, 2 or 0:
```

64

```
TOP 50 PLAYERS

1. USER ID: 1  USERNAME: s s  SCORE: 72
2. USER ID: 6  USERNAME: Isaac Patrickson  SCORE: 3
3. USER ID: 2  USERNAME: Guy Johnson  SCORE: 3
4. USER ID: 3  USERNAME: Shay Oleary  SCORE: 2
5. USER ID: 9  USERNAME: Jane Doe  SCORE: 0
6. USER ID: 8  USERNAME: John Doe  SCORE: 0
7. USER ID: 7  USERNAME: John Doe  SCORE: 0
8. USER ID: 5  USERNAME: Helleni Tsentas  SCORE: 0
9. USER ID: 28  USERNAME: Barry White  SCORE: 0
10. USER ID: 26  USERNAME: TestFirstName TestLastName  SCORE: 0
11. USER ID: 23  USERNAME: Clive Morrow  SCORE: 0
12. USER ID: 21  USERNAME: Ryan Johnson  SCORE: 0
13. USER ID: 19  USERNAME: George Harrison  SCORE: 0
14. USER ID: 18  USERNAME: Zack Monroe  SCORE: 0
15. USER ID: 17  USERNAME: Zack Monroe  SCORE: 0
16. USER ID: 16  USERNAME: Charlotte Pollard  SCORE: 0
17. USER ID: 10  USERNAME: Forest Louis  SCORE: 0

0. Go Back
0
```

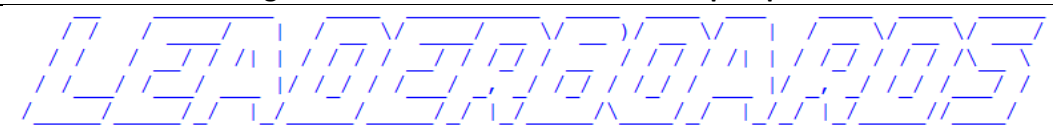
```
1. View overall Leaderboard
2. View leaderboards per quiz
0. Go back

Select 1, 2 or 0:
```

65

```
TOP 50 PLAYERS

1. USER ID: 1  USERNAME: s s  SCORE: 72
2. USER ID: 6  USERNAME: Isaac Patrickson  SCORE: 3
3. USER ID: 2  USERNAME: Guy Johnson  SCORE: 3
4. USER ID: 3  USERNAME: Shay Oleary  SCORE: 2
5. USER ID: 9  USERNAME: Jane Doe  SCORE: 0
6. USER ID: 8  USERNAME: John Doe  SCORE: 0
7. USER ID: 7  USERNAME: John Doe  SCORE: 0
8. USER ID: 5  USERNAME: Helleni Tsentas  SCORE: 0
9. USER ID: 28  USERNAME: Barry White  SCORE: 0
10. USER ID: 26  USERNAME: TestFirstName TestLastName  SCORE: 0
11. USER ID: 23  USERNAME: Clive Morrow  SCORE: 0
12. USER ID: 21  USERNAME: Ryan Johnson  SCORE: 0
13. USER ID: 19  USERNAME: George Harrison  SCORE: 0
14. USER ID: 18  USERNAME: Zack Monroe  SCORE: 0
15. USER ID: 17  USERNAME: Zack Monroe  SCORE: 0
16. USER ID: 16  USERNAME: Charlotte Pollard  SCORE: 0
17. USER ID: 10  USERNAME: Forest Louis  SCORE: 0

0. Go Back
Sw3d3
Error! Must enter '0' to go back

0. Go Back
```

Testing that a user can view leader boards per quiz

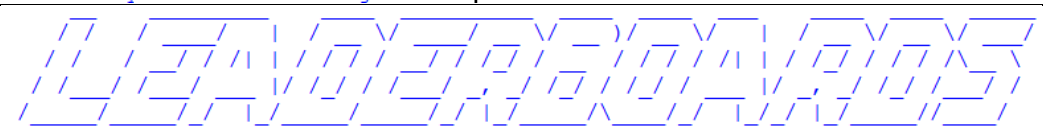
66



```
1. View overall Leaderboard
2. View leaderboards per quiz
0. Go back

Select 1, 2 or 0: 2

ID: 4
Quiz Title: Music Quiz
Created by: s s
Question Type: Mixed
Maximum Score: 3
Description: A Quiz for Guy and Shay



ID: 25
Quiz Title: Music Quiz
Created by: Isaac Patrickson
Question Type: Multiple Choice
Maximum Score: 1
Description: A quiz about popular music
```

```
ID: 45
Quiz Title: Star Wars Quiz
Created by: George Harrison
Question Type: Multiple Choice
Maximum Score: 1
Description: A short Star Wars Quiz



ID: 48
Quiz Title: Blue Whale Quiz
Created by: s s
Question Type: Open Ended
Maximum Score: 1
Description: A quiz about Blue Whales


Select a quiz ID or '0' to go back:
```

67



```
1. View overall Leaderboard
2. View leaderboards per quiz
0. Go back

Select 1, 2 or 0: 567ytyh*
Select 1, 2 or 0:
```

68

```
Select a quiz ID or '0' to go back: 48


TOP 50 PLAYERS WHO COMPLETED: Blue Whale Quiz

1. USER ID: 1  USERNAME: s s  SCORE: 1
2. USER ID: 1  USERNAME: s s  SCORE: 0
3. USER ID: 1  USERNAME: s s  SCORE: 0

0. Go Back
```

69

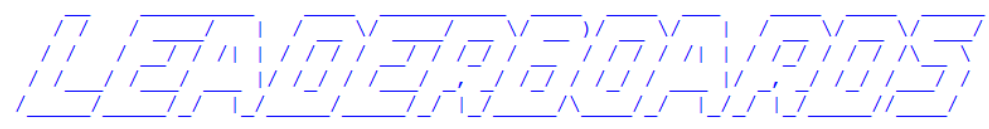
```
Select a quiz ID or '0' to go back: @sdasd'q2
Error! Please enter an integer.
Select a quiz ID or '0' to go back:
```

70

```
Select a quiz ID or '0' to go back: 89
Error! That ID doesn't exist
Select a quiz ID or '0' to go back:
```

71

```
Select a quiz ID or '0' to go back: 0
```



```
1. View overall Leaderboard
2. View leaderboards per quiz
0. Go back

Select 1, 2 or 0:
```

Testing Video

Demonstrating the robustness of the solution

In the video, I have demonstrated that every user input is fully validated before it gets processed. I did this by entering in normal data, erroneous data, and boundary data.

Evaluation

How well does the project meet its requirements? - Improvements

I believe my program fulfils the requirements set by the client. The system allows users to create, take and track progress of quizzes by permanently storing data within database tables. In testing, this proved a reliable way to store data for retrieval later. Manipulation of this data through queries, made it possible to insert, update, select, and delete records. These queries form the basis of every function in my program and allow the user to interact with the database.

For the user to create an account, an email needs to be entered. This email is unique, and in conjunction with a password, provides a means of logging in once the account has been created. The validation requirements for the entered email are that it must contain the '@' symbol, no spaces are allowed, and the email can be no longer than 50 characters. However, there is no validation to check whether the email exists or not – this can be dealt with as a future improvement. Another future improvement could consist of the user verifying their email address by clicking a link on a received email sent by the program.

Having an account stored in the database, is vital to the programs key functions. Each quiz created, needs to be linked with the user who created it. Similarly, each take the user does, needs to be traced back to the user so they can be displayed on leader boards. This is done through foreign keys and cross table parameterisation. In testing, this proved to be a viable way of linking users with their activity on the system. User feedback suggested that there should be an option to delete your user account and remove all records containing that user. If I were to improve the program in the future, I would consider including this feature.

Each quiz has a max score. This dictated the maximum points a user could earn from completing the quiz. However, it also controlled how many questions were in the quiz as each question was only worth one point. For example, a quiz could have fifty available points and therefore it would contain 50 questions. As a future improvement, I could introduce a new field into the quiz table called "numberOfQuestions" which would separate the points from the number of questions. This could also help me utilise the difficulty field in the question table, tying different question difficulties to different point rewards. For example, easy rewards ten points, medium thirty points, and hard fifty points.

The user interface is run through the python shell. In testing, it offered a simple way of receiving inputs and displaying outputs trough the text entry and display window. However, feedback from the client suggested that the user interface could be easier to navigate. This could be improved by implementing python's Tkinter library to create an attractive, easy-to-use GUI. Navigation could be improved by using Tkinter buttons. These buttons could take the user to and from different menus un the program.

Another objective that I would have liked to give myself would be to create a function that modifies individual quiz records. This function would allow the user to change a quiz's title and description as well as each individual question. This could save time, as user would not have to delete the quiz if they wished to make a change. For example, if a user created a quiz with fifty questions and they wanted to change one of them, they would not have to enter the content for all fifty questions again. This could also be applied to the user records; first name and last name.

Independent Feedback – Helleni Tsentas (Client)

I love taking and making quizzes and produced quite a few quizzes during lockdown, so I was excited to try out this new quizzing programme.

The interface is very clear, simple and easy to use, which is a good thing, but I think I would like something more graphical. I think a graphical interface would make it more engaging and exciting for users. Not being a computer programmer myself, I'm used to using software and apps that are more visually orientated, so this programme seemed a little old fashioned looking even though it did everything it needed to.

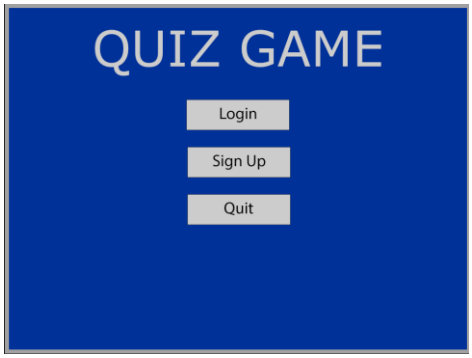
I really like the fact that there was a database of prepared quizzes because in the past it's been very time consuming to create a new quiz, so to have some to hand was great. I also liked the fact that I could create my own quizzes that would be saved so I could build my bank of quizzes and include different topic-based question. Also being able to delete quizzes was very useful, because sometimes you can make a mistake with the questions or maybe the topic was no longer relevant, and you don't want these clogging up your database.

Having a function that automatically stored user's results and even produced a leader board for each quiz was great. Before this system marking quizzes and then collating the points was time consuming for me, so having instant access to that information was useful. I like the fact that people can use the leader board to check how they are doing, it helps create a little friendly competition and may spur people on to retake quizzes or do more quizzes to improve their ranking.

Discuss the Independent Feedback

When considering different methods of displaying a user interface, I concluded that the python shell would suffice. My client stated that a more appealing interface could encourage more people to play the game. They also mentioned that the average player might not appreciate the work put into the program if it doesn't look visually appealing. Acting on this, I designed a few windows which demonstrate what the graphical user interface would have looked like if I decided to improve the program in the future.

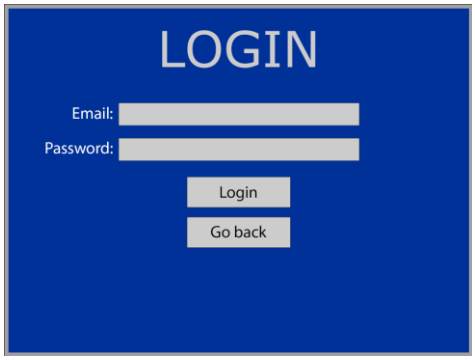
This user interface would be implemented using the python library "Tkinter."



This would be the opening window which would be the first thing the user sees once they run the program.

Pressing the "Quit" button would close the program.

This would have been a part of the Login Menu process.



If the user were to press the "Login" button on the opening window, they would be taken to this screen, where they can enter their login details.

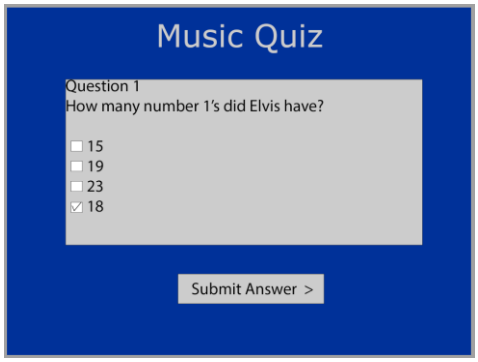
If the details typed in the boxes match that of a user in the "User" table, the user would proceed to the main menu. If the details are incorrect, an error message box would pop up and display an error message.

This would have been a part of the Login Menu process.

If the user were to press the "Sign Up" button on the opening window, they would be taken to this screen, where they can enter their account details.

If the "Create Account" button is then pressed, their details would then be saved to the "User" table in the database, unless there is an error in one or more fields entered i.e., their email does not contain an "@" symbol.

This would have been a part of the Login Menu process.



This is an example of when a user is taking part in a quiz.

They are taking part in the quiz titled "Music Quiz" and they are on the first question. This question is a multiple-choice question with four answer choices. The user has selected the fourth choice. The answer will only be submitted once the "submit answer" button has been pressed.

Once the "submit answer" button has been pressed, the screen will display the next question. The user will not be allowed to go back to a previous question.