Project Report

by CHRISTOPHER YATES

Submission date: 12-Sep-2021 09:53PM (UTC+0100)
Submission ID: 159709155
File name: 122808_CHRISTOPHER_YATES_Project_Report_1071986_1337435428.docx (1.68M)
Word count: 18576
Character count: 116362

RealTime Quiz App

MSc Information Technology Project Report

Software implementation project report on a quiz application built to improve cognitive function and to help people with memory difficulties

Prepared by Christopher Yates

Student Number: 13176981

Department of Computer Science and Information Systems,

Birkbeck College, University of London

2021

Supervisor: George Roussos

https://realtimed-quiz.herokuapp.com

This report is substantially the result of my own work, expressed in my own words, except where explicitly indicated in the text. I give my permission for it to be submitted to the JISC Plagiarism Detection Service.

The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

During the pandemic the ability to meet and interact with others has been severely limited, thereby reducing the opportunity to engage critical thinking. The human brain is like a high-performance muscle, and if it is not used it will weaken and deteriorate. Although this is an active research area, studies are increasingly finding a link between brain inactivity and memory problems.

This project will show how software can mitigate the impact of the pandemic on mental health, sharpen brain function and help people exercise their memory. It will be a fun way to improve cognitive function and overall mental agility between friends, and we will explore the potential of the application to prevent the onset of Dementia.

Acknowledgements

The prospect of completing this course was at times challenging. The author has a legal background and no prior experience in IT, programming, maths or any other related STEM subject. Special thanks go to all the staff at Birkbeck College for making me believe this was possible, including the programme director, Vladislav Ryzhikov, for allowing me to take the course despite my inexperience. The patience of Keith Mannock during the Applied Software Engineering module. All the help, support and useful guidance of my project supervisor, George Roussos. The insight and moral support of my friend Shoni Newell, a former mental health solicitor who has direct experience of the issues touched upon in this report. Finally, my employer, Edward Strickland of Thomas Legal Costs, for allowing me the time to focus exclusively on my studies.

List of Figures

Figure 1 - MindMate User Interface 9

Figure 2 - Lumosity User Interface 10

Figure 3 - Home Tab 12

Figure 4 - Rankings Table 12

Figure 5 - About Tab 13

Figure 6 - Password Reminder 13

Figure 7 - Registration Step One 13

Figure 8 - Registration Step Two 13

Figure 9 - Status Tab 14

Figure 10 - Attempts Tab 14

Figure 11 - Account Tab 14

Figure 12 - Example of Answer 15

Figure 13 - Example of Question 15

Figure 14 - Results Page 15

Figure 15 - Breakdown of Results 15

Figure 16 - Survey 15

Figure 17 - Non-Functional Requirements 16

Figure 18 - Functional Requirements 18

Figure 19 - Diagram of Overall Application Design 19

Figure 20 – Use Case Diagram 20

Figure 21 - Typical Flow of Execution 21

Figure 22 - ER Diagram of New Database Design 22

Figure 23 - Open Trivia URL Generator 23

Figure 24 – Code for Width of Headings and Tabs 24

Figure 25 – Code for Hiding and Showing Front End UI 25

Figure 26 – Diagram of Questions and Timer Functions 26

Figure 27 – Initialising Express and the App Object 27

Figure 28 – Routing Request for Home Page 27

Figure 29 – Creation of Rankings Table 27

Figure 30 – Scraping API for Questions Data and Storing in Database 29

Figure 31 - ER Diagram of Old Database Design 31

Figure 32 - User Testing Results 36

TABLE OF CONTENTS

Acknowledgements 3

List of Figures 4

Chapter 1: Introduction

1.1 Rationale for the Project 8
1.2 Background to Memory Loss and Dementia 8
1.3 Analysis of Brain Training Research 8
1.4 Other Brain Training Applications 9
1.5 Critique of Other Applications 10
1.6 Project Objective 10
1.7 Differences with Project Proposal 11

Chapter 2: Project Trailer and Features

2.1 Introduction to the RealTime Quiz Application 12
2.2 Home Page and Rankings System 12
2.3 Registration and Data Protection 13
2.4 User Profile Features and Quiz 14
2.5 Post-Quiz Features and Survey 15

Chapter 3: Design of Application

3.1 Requirements Gathering 16
3.2 Non-Functional Requirements 16
3.3 Functional Requirements 17
3.4 High Level Summary of the Design 19
3.5 RESTful Services 20
3.6 Database Structure 21
3.7 Trivia API 23

Chapter 4: Implementation

4.1 Methodology 24
4.2 Review of Implemented Technology 24
4.3 Security Issues 29
4.4 Specific System Improvements 30
4.5 Deployment 31
4.6 Choosing a Platform 31

4.7 Configuring the Remote System 32

Chapter 5: Testing

5.1 Manual Testing 33

5.2 Data Protection and Ethical Approval 33

5.3 Live Testing 33

5.4 User Testing 35

5.5 Data Analysis from User Testing 35

Chapter 6: Conclusions

6.1 How the Project Meets the Objective 37

6.2 Critical Comparisons with Similar Applications 37

6.3 Potential Improvements to Features 37

6.4 Overall Conclusion of the Project 38

Bibliography 39

Appendix 40

User Manual 40

Full Listing of Development Code 42

Surveys 92

Chapter 1

INTRODUCTION

1.1 Rationale for the Project

Around 40% of people aged 65 or older have age associated memory impairment, and 1% of them go on to develop Dementia each year [1]. This period of social isolation has been particularly challenging for the most vulnerable in our society, including the elderly, as they have faced more restrictions on their movements than any other group and could not accept visitors or go outside. This in turn has impacted their interactions with others in terms of conversation and engaging critical thinking.

Studies have shown that the risk of developing Alzheimer's disease is lower in people who have been intellectually active [2]. One method of fighting the onset of Dementia in recent years has been the emergence of brain training games; as an increasing body of research suggests that combined with a healthy lifestyle, brain training could be a viable method of preventing the disease.

1.2 Background to Memory Loss and Dementia

Although people can experience memory loss at any age, it is more likely that it will occur to people who are over 65 than any other age group. Because the average human life expectancy is increasing each year, memory loss and Dementia are key concerns.

Age related memory problems are commonplace in our society. As the human body gets older all parts of it deteriorate including the brain. Although age related memory loss is less debilitating than Dementia and Alzheimer's, it can still have a significant impact including raising alarm and embarrassment, as well as a detrimental impact on day to day living.

The most severe conditions in this area are Dementia and Alzheimer's. These diseases are devastating, both for the person suffering and their family members and carers. Sufferers are unable to communicate with their loved ones properly and often lose their autonomy in their personal affairs and are placed into specialist care homes for their own safety.

One of the leading methods to diagnose Dementia is the use of word list recall tests, where the patient is given numerous words to remember and then undergo a memory test [3].

1.3 Analysis of Brain Training Research

The effectiveness of brain training games to help with or prevent Dementia and memory loss is a contested issue, and an active research area.

In 2014, the Stanford Center on Longevity published an open letter signed by 69 international neuroscientists and cognitive psychologists stating that there was no compelling scientific evidence that playing brain games improved cognitive abilities in everyday life, although isolated benefits could exist [4].

A few months later in 2014, an open letter was sent in rebuttal by 100 neuroscientists, psychologists, and other experts in the field of neural plasticity. It was argued that the research used by the Stanford Center was fundamentally flawed [5].

One of the largest and most prestigious studies to ever take place in relation to brain training games was conducted in America and known as the Advanced Cognitive Training in Vital Elderly (ACTIVE) Study and the results were measured over a 10-year period and published in 2014 [6]. This study considered a total of 2,802 adults with an average age of 74.

The ACTIVE study used computer software for users to remember words and objects and aimed to improve the users' cognitive function. The training focused on memory, reasoning, and speed of processing because prior research indicated that these abilities show early age-related decline and are related to activities of daily living.

It was noted that memory and reasoning training only helped short term cognitive function and did not lower the risk of Dementia. The speed of processing training was found to be the most beneficial method as it had a direct link to lowering the risk of developing the disease. Also, it was noted that the benefits obtained by speed of processing training increased in correlation with how many training sessions had been undertaken.

The Active study team calculated that the speed of processing training resulted in a 29 percent reduced risk of Dementia over 10 years, and that each additional training session was associated with a 10 percent lower Dementia risk [7].

However, the researchers for the Active group concluded that speed of processing training was not a cure for existing Dementia sufferers and would only serve as a tool to help lower the risk of people developing the disease if they were otherwise healthy when undertaking the training.

1.4 Other Brain Training Applications

Further to the research carried out for the project proposal, more brain training resources have been analysed that have a specific emphasis on improving users' memory recall.

The most prominent of these are MindMate [8] and Lumosity [9]:

Mindmate concentrates on both memory and physical health. There is good evidence to suggest that a healthy lifestyle is essential in the fight against Dementia [10]. As shown in Figure 1 the user interface is appealing, and it also uses a timing mechanism for the exercises. The exercises themselves are based on remembering objects rather than word list recall. The app is free to play and invites user participation in ongoing research into this area, which is an excellent addition to their offering.

Figure 1 – MindMate User Interface

Lumosity is also involved in ongoing research in this area. There are limited free daily games that can be played, although it is necessary to pay to participate in much of the functionality of this application. The memory training focuses on remembering objects rather word list recall. The user interface is engaging and visually appealing and is shown in Figure 2 below.

Figure 2 – Lumosity User Interface

1.5 Critique of Other Applications

George Rebok, the professor who led the ACTIVE study was interviewed for an article featured in the Business Insider magazine. He was of the view that the current brain training games on the market do not correlate with the cognitive training undertaken in the ACTIVE study [11].

The activities in the study included listening to a list of unrelated words, identifying patterns in a series of letters and identifying targets in their field of view. In contrast games like Lumosity included activities like Bird Watching, where a player must identify a letter in the center of the screen whilst simultaneously finding the location of the bird.

Although a lot of other brain training applications are involved in research, this is mostly introspective in relation to how well the games existing functions are improving memory and are therefore subjective. These games would benefit by broadening the research and including recommendations from leading independent studies such as ACTIVE.

1.6 Project Objective

The project aim is to develop software which provides a direct measure of memory retention using general knowledge quiz questions. An emphasis will be placed on complying with the recommendations of the ACTIVE study and the use of word list recall and speed of processing training.

The software designed for this project will show the user a list of various questions and answers, and they will have a short time limit to remember as many answers as possible before the quiz begins, and their memory is tested.

The user will be able to track the progress of their memory retention through the attempts table provided in their user profile. This will show the history of all attempts at the quiz and the number of questions answered correctly.

All users will be able to see their progress on a leaderboard on the home page of the application, so that they can compare their progress with the other users of the quiz application; this addition should act as extra motivation for users to keep playing the quiz to further improve their memory.

1.7 Differences with Project Proposal

Although the main objective of the project has not changed, there have been changes made to the format of the questions and the data collection method for the quiz data.

Initially, the proposal idea was to use a small, fixed set of 25 questions with rapidly evolving answers, such as "*Who is the current leading goal scorer in the Premier League?*". It was intended that the answers would be collected by web-scraping targeted sites at daily intervals for the most up-to-date information.

However, following discussions with friends at the pre-development stage it was decided that such a small set of questions would become monotonous. The main purpose of the software is to improve memory and so it is vital that the users are engaged and motivated to keep playing the game.

The data pool for the questions and answers is now much larger, with a total of 1200 questions and answers stored in the database. The method of data collection has changed from scraping the web to collecting the data from an API, because the structure of websites are frequently updated, thereby necessitating changes to the web scraping code to comply with the updated website structure.

Scraping an API is much more reliable than web scraping and will involve less ongoing maintenance to ensure that the data collected is accurate. The new data collection method was achieved by scraping a Trivia API, which is a dedicated service for students who are preparing quiz applications for educational purposes.

There is a time limit for the user to remember each answer before the quiz begins. Furthermore, because there is 1200 questions and answers stored in the database, the user is constantly receiving new questions and answers which it is hoped will motivate them to keep playing the quiz. These additions were included as a direct result of the speed of processing findings by the leading ACTIVE study.

In the original proposal objective, the intention was to scrape the data at twice-daily intervals. This has also changed due to the size of the data obtained from the Trivia API. Rather than programming the application to run the scraping service autonomously at daily intervals, the data is simply scraped once before the application is either deployed to the cloud or being run locally on a home machine.

Chapter 2

PROJECT TRAILER AND FEATURES

2.1 Introduction to the RealTime Quiz Application

The RealTime Quiz is a simple to use memory and speed of processing test. It uses the word list recall method that was used in the ACTIVE study. Once registered and logged in the user can play the quiz as many times as they like and different questions and answers will be shown on each attempt.

Although the game is essentially a quiz, there is a twist insofar as the answers are shown to the user before the quiz begins and a short time is given for the user to remember each answer.

Before playing the quiz, it is necessary to register and login. A record of users' previous attempts is available to view, as well as a rankings table with the overall memory improvement score of all users.

It can be viewed on any device or screen size, albeit the multiple-choice options are slightly easier to select on a mobile phone or an I-Pad.

2.2 Home Page and Rankings System

The home page includes a split tabbed interface with options for either logging in (Figure 3), rankings (Figure 4) or the about section detailing a high-level overview of the application (Figure 5).

The login tab can only be completed if the user is already registered. If the user is not registered, they will need to complete the registration setup. If the user has forgotten their password, functionality is also included for the user to be emailed their password by clicking the forgotten password button and entering their email address (Figure 6).

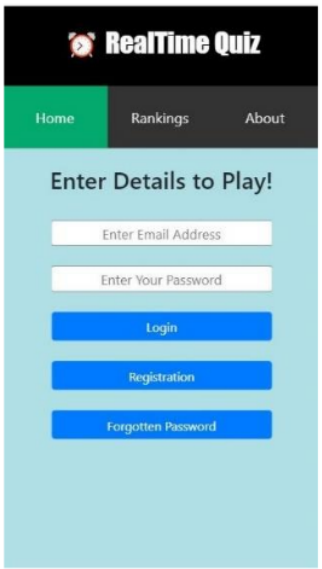

Figure 3 – Home Tab

POS	NAME	%IMP
1	Adam	111%
2	Francesca	100%
3	Caroline	78%
4	Shoni	50%
5	Brad	50%
6	Xin	30%
7	Penelope	28%
8	Matt Gooch	25%
9	tony r	20%

Figure 4 – Rankings Table

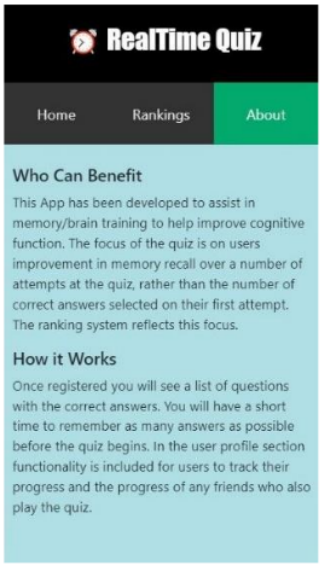

Figure 5 – About Tab

Figure 6 – Password Reminder

2.3 Registration and Data Protection

The registration section has two parts. First, the user is asked for their email address, a nickname and for their password which needs to be entered twice (Figure 7).

Second, the user is presented with a data protection disclaimer which specifies the data that will be stored, the purpose of storing the data and sets a time limit for when the user can be removed from the database (Figure 8).

Figure 7 – Registration Step One

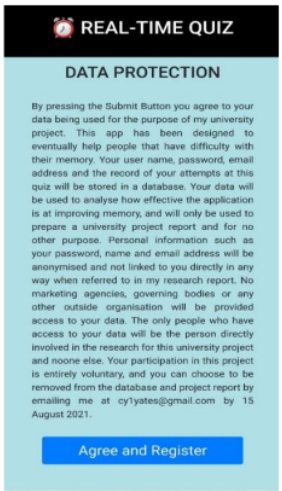

Figure 8 – Registration Step Two

2.4 User Profile Features and Quiz

Once the user logs in, they will see their profile section consisting of a split interface with three tabs (Figure 9). The first is the status tab which shows whether the user is a beginner or is improving in terms of memory recall. For a user to elevate from beginner status they need to play the quiz twice. Afterwards they will be shown whether they are improving or not. This is determined by a simple calculation of how many correct answers were made on the first attempt compared with the most recent attempt at the quiz.

The attempts tab shows the user all their previous attempts at the quiz, the date, and how many correct answers they remembered during each attempt (Figure 10).

On the account tab the user can set a new password or username (Figure 11).

Once the user starts the quiz, they are provided with a set of 20 questions and answers. Each question and answer is shown to the user on a single page design, with a time limit of 5 seconds for the user to remember each answer (Figure 12).

If the user already knows the answer to the specific question, they have the option to skip to the next answer by clicking the *Next* link. The progress bar is updated after each answer is shown until it reaches 100% when all the answers have been presented to the user.

After the user has seen the full set of 20 questions and answers, the quiz begins, and the same set of questions are presented with a choice of 3 options for the answer (Figure 13).

Once the user selects a multiple choice option, the page is automatically changed and replaced with the next question. The progress bar is also updated so that the users have a point of reference in relation to how many of the questions they have completed.

Figure 9 – Status Tab

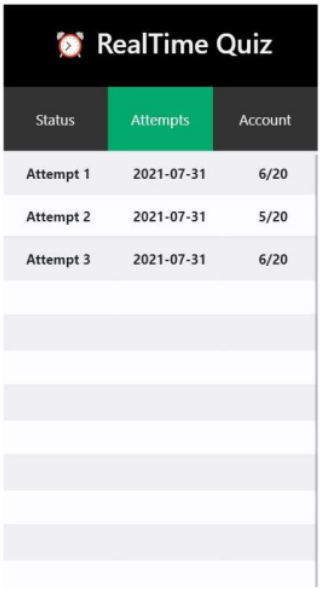

Figure 10 – Attempts Tab

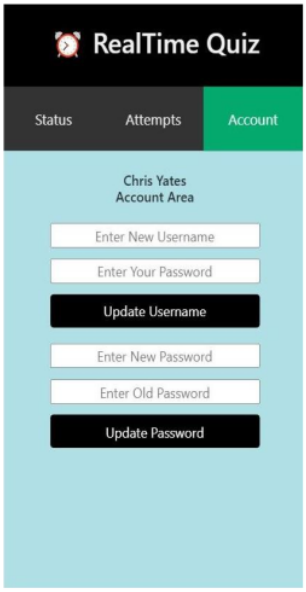

Figure 11 – Account Tab

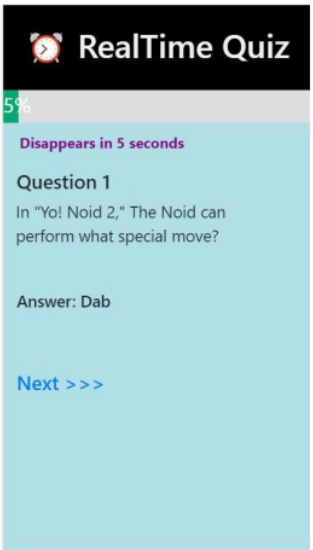

Figure 12 – Example of Answer

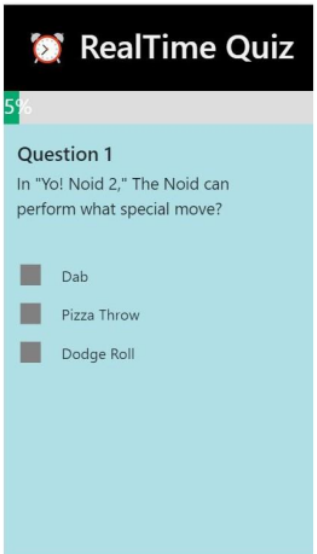

Figure 13 – Example of Question

2.5 Post-Quiz Features and Survey

Following all 20 questions being answered, the user is shown their score and has the option to see a breakdown of their incorrect answers, to take the survey or return to their user profile (Figure 14).

If the user chooses to see the breakdown, they will be presented with a list of correct and incorrect answers (Figure 15). The survey was used to gather the required user testing information to be included in this report. It includes some general questions about the quiz, the age of the user and whether the app could be improved in any way (Figure 16).

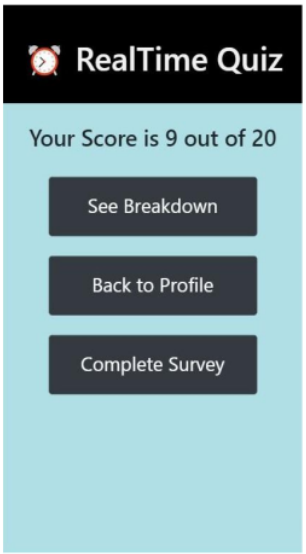

Figure 14 – Results Page

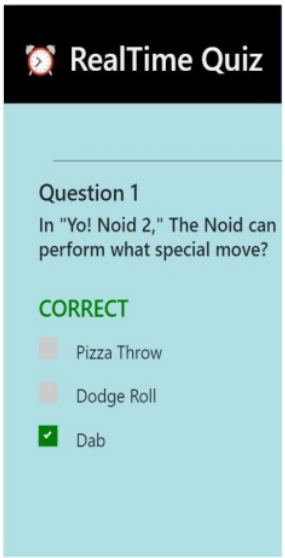

Figure 15 – Breakdown of Results

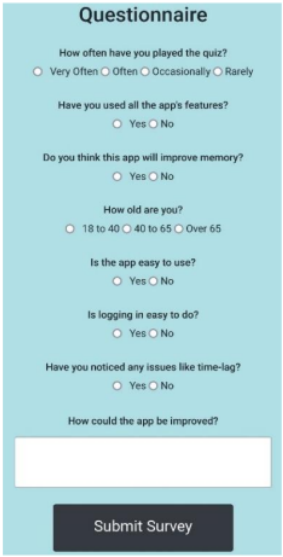

Figure 16 - Survey

Chapter 3

DESIGN OF APPLICATION

3.1 Requirements Gathering

During this phase consultations took place between friends regarding the requirements of the application. Most notably, a former mental health solicitor provided very useful insight into memory loss, capacity issues and other related issues during her time in this area.

It was noted that the application had to be easy to use for people of varying abilities, it had to be fun to use and it was beneficial for a record to be held on each attempt so that carers or family members could see a history of the memory training.

Furthermore, consideration was given to the findings of the large-scale research carried out by the ACTIVE [6] study including the following overarching findings:

- **Finding 1** - games that use speed of processing training were the most beneficial to lower the risk of Dementia

- **Finding 2** – speed of processing games that are played the most frequently have the greatest positive impact on cognitive ability and memory. Reference is made below to requirements that make the user experience more pleasurable or motivate the user to keep playing the quiz

3.2 Non-Functional Requirements

As this application is relatively simple, there was very few non-functional requirements apart from that the application worked, does not take too long to initially load in the browser, is compatible across any device and is easy to use for people of different capabiltities.

Figure 17 shows these non-functional requirements, and how they relate to the consultations that are mentioned in section 3.1 above, and the findings of the leading ACTIVE study [6].

Non-Functional Requirements

1. Performance
   1.1 System loads quickly **(Finding 2)**
2. Browser and Mobile Compatibility
   2.1 App can be viewed on all PC, Laptop and Mobile Devices **(Finding 2)**
3. Security
   3.1 User required to login and register to play
4. Usability
   4.1 System is easy to use **(Finding 2)**

Figure 17 – Non-Functional Requirements

3.3 Functional Requirements

The functional requirements are dominated by the overarching findings of the ACTIVE study mentioned in section 3.1. These are detailed in Figure 18 below.

A comprehensive use-case is described at Figure 18, which shows the complete functionality of the system and reference is made to how parts of the system functionality relate to the research carried out in relation to brain training games.

Finding 1 is the emphasis on speed of processing training. This is achieved using timers when the user views the questions. This was an essential part of the application which ensures that the user has a time limit in which to process the answer data and store it in their memory.

Any reference to Finding 2 in Figure 18 relates to encouraging the user to keep playing the quiz, so that they can adhere to the research carried out by the ACTIVE study.

It was noted in the study that the more times a user played a speed of processing game the better the results of the training [7]. This encompasses a wide variety of the functionality of the system including anything that makes the experience easier for the user or motivates them to keep playing the quiz.

Because the recommendations taken during the consultation phase outlined in section 3.1 are consistent with the overarching findings of the ACTIVE study, Figure 18 only show the link between the functional requirements and the main findings of the ACTIVE study.

Functional Requirements

1. View the Home Page
   - 1.1 View the rankings table **(Finding 2)**
   - 1.2 View the about section
2. Login to Quiz
   - 2.1 Login as existing user
   - 2.2 Register to create account
   - 2.3 Receive password reminder email **(Finding 2)**
3. View the User Profile Section
   - 3.1 View the status page to see if they are improving or not **(Finding 2)**
   - 3.2 View the attempts table **(Finding 2)**
   - 3.3 View the account section
     - 3.3.1 Change username or password
4. Play the Quiz
   - 4.1 See the answers
     - 4.1.1 Each answer is shown for 5 seconds **(Finding 1)**
     - 4.1.2 Each answer is shown on a single-answer-per-page basis **(Finding 2)**
     - 4.1.3 If answer is known user can use next button to fast-forward to see next answer rather than waiting for timer to expire **(Finding 1/Finding 2)**
     - 4.1.4 When next answer loads the previous answer disappears **(Finding1/Finding 2)**
     - 4.1.5 When the next answer loads the progress bar is updated **(Finding 2)**
   - 4.2 See the Questions
     - 4.2.1 Each question is shown with 3 multiple choices **(Finding 1)**
     - 4.2.2 Each question is shown on a single-question-per-page basis **(Finding 2)**
     - 4.2.3 When the selection is made a new question appears **(Finding 2)**
     - 4.2.4 The user cannot go back to re-attempt the question **(Finding 1)**
     - 4.2.5 When the next question loads the progress bar is updated **(Finding 2)**
5. View Post-Quiz Features
   - 5.1 See the Results
     - 5.1.1 Immediately following the quiz, the score is shown **(Finding 2)**
     - 5.1.2 Users can see incorrect answers, take survey or back to user profile **(Finding 2)**
   - 5.2 View the Breakdown
     - 5.2.1 A list of the questions is shown with the correct answers **(Finding 2)**
     - 5.2.2 At the end of the list the user can go back to the user profile section **(Finding 2)**
   - 5.3 Take the Survey
     - 5.3.1 A list of questions is shown, with space for comments on how it can be improved
     - 5.3.2 The survey is emailed to me on an anonymous basis when completed
     - 5.3.3 The user is taken back to their profile upon submitting the survey **(Finding 2)**

Figure 18 – Functional Requirements

3.4 High Level Summary of the Design

This application has been designed using an *n tier client-server architecture* which consists of the traditional 3 layered architecture for web applications, plus another layer to collect the data from an external API source.

As shown in Figure 19 below, the system has two sub-systems including the main application itself where clients can interact with the application server, view and store information in the database. And the scraping sub-system, which is responsible for storing the questions and answers in the database.

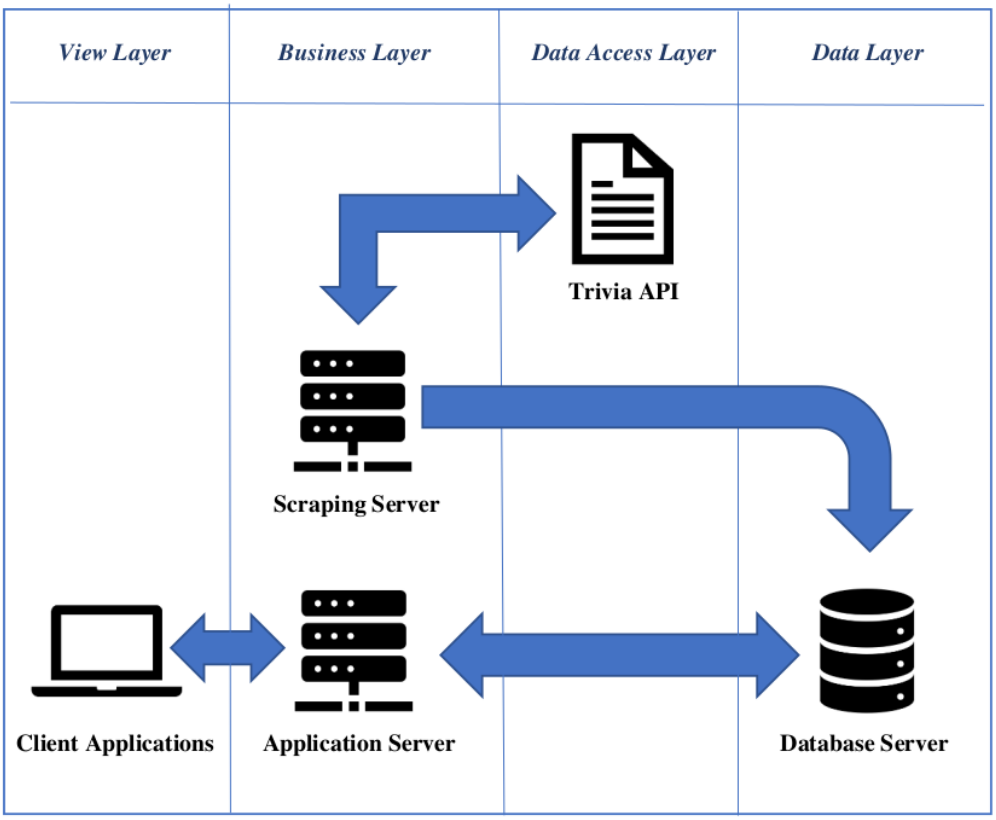

Figure 19 – Diagram of Overall Application Design

All communication to each layer is made by the business or logic layer, which stores data in the database, receives requests from users and collects information from the database and updates the database based on instructions from users.

Due to the design of the application, it can be easily managed and is scalable because each tier is independent from each other. Furthermore, the servers within the business tier are also independent of each other. Therefore, any programming language could be used for any tier or sub-system as they are not reliant on each other. The different sub-systems can be updated from different locations by different teams which makes them independently scalable, results in less code for each team to understand which in turn makes the different sub-systems simpler to maintain.

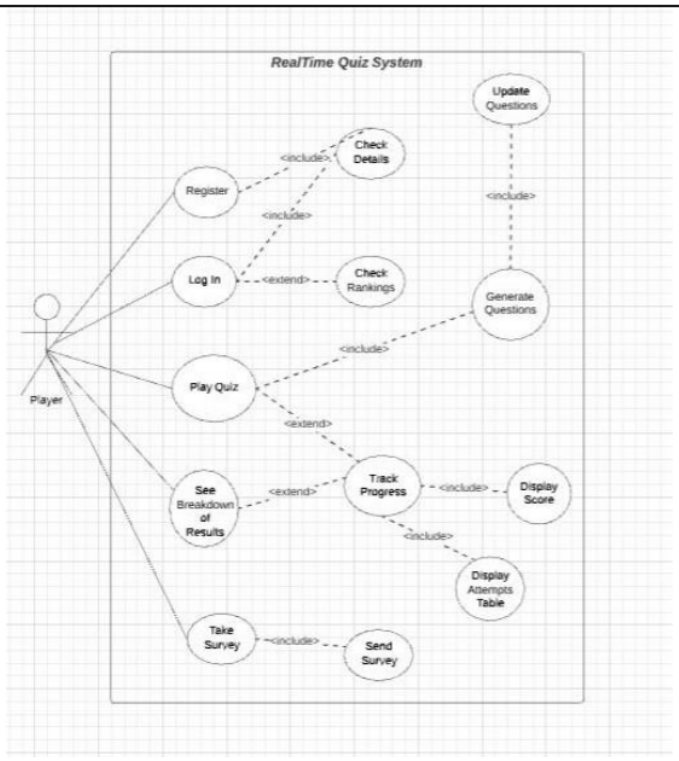

Figure 20 – Use Case Diagram

3.5 RESTful Services

This application revolves around the REST architectural style and has numerous APIs that either read or update information in the database.

The APIs used in the application include:

1) Registering as a new user (Figures 7 and 8)
2) Logging into the application (Figure 3)
3) Retrieving forgotten password (Figure 6)
4) Changing username or password (Figure 11)
5) Starting a new quiz (Figure 9)
6) Viewing incorrect answers (Figures 14 and 15)
7) Taking the survey (Figures 14 and 16)

For each API mentioned above, the client server sends a HTTP request to the application server, either in the form of a GET or POST request depending on whether the API is intended to read or update information in the database.

The application server then sends a request to the database server for information, and the database server sends an object with the data back to the application server for processing. Once processed, the application server sends a response to the client browser which is then handled by the code on the client side.

Figure 3 is an example of an API used in the application and its role is to log into the quiz. Once a user completes the logging in fields and presses the submit button, a HTTP request is sent from the client server to the application server. The application server then communicates with the database server and checks whether the user is indeed registered and if they are, it sends back a HTTP response to the browser to show the user profile. An overview of this process is shown in Figure 21 below.

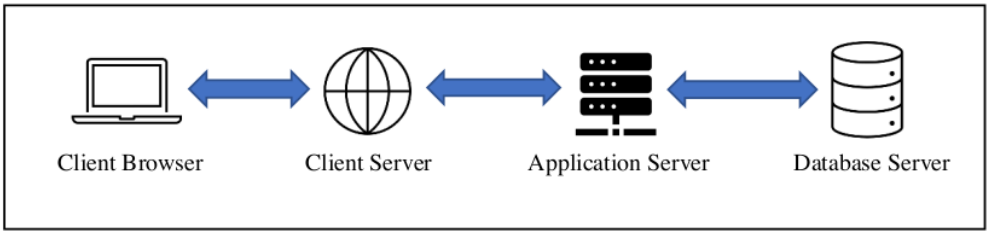

Figure 21 – Typical Flow of Execution

3.6 Database Structure

The overall focus on the design of the database was in relation to data redundancy and ensuring that the least amount of data was stored including keeping the number of rows to a minimum.

There are 3 tables in the database which hold the user login details, the questions and the scores for the rankings and attempts data (Figure 22).

As discussed in the implementation chapter, the original design of the database shown in Figure 24 was overly complex and would have resulted in unnecessary data being stored. For example, there was no need for a quiz, answers or attempts table. Instead, all these tables were replaced by a leaderboard table.

Scores are processed by the application server before they are sent to the database. This has resulted in an elegant and simple database structure which only holds the lowest amount of necessary data.

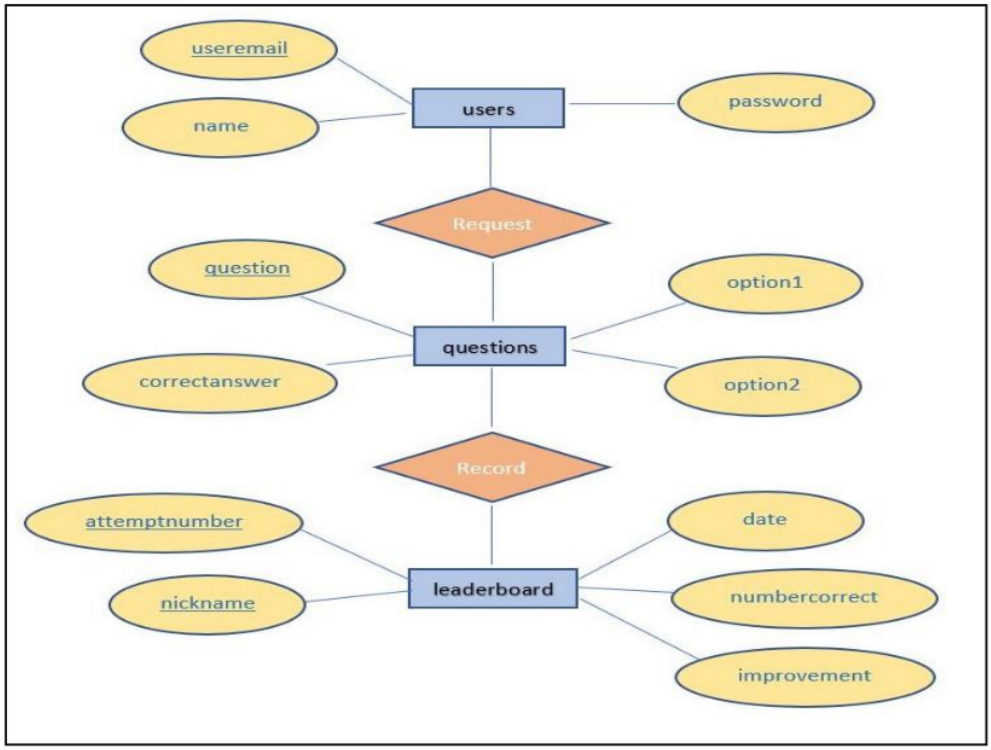

Figure 22 – ER Diagram of New Database Design

The users table is only updated once for each user when they initially register. Thereafter, it acts as a reference point to ensure that the login details are correct. Therefore, this ensures that a minimal number of rows are included in the users table to alleviate any data redundancy issues.

The questions table is completely independent and is only updated by the separate scraping sub-system and is not updated in any way by the application sub-system which only reads the data contained within the questions table.

Once a user is registered, the only table that is updated on each attempt is the leaderboard table. Rather than storing each question and answer in the leaderboard table, the calculations are instead completed within the node.js server-side script and the number of correct answers is posted to the leaderboard table, to reduce the amount of data held in the database.

The leaderboard table is responsible for updating both the main rankings table on the home page of the application, and the specific attempts table contained within the user profile. This dual purpose has obviated any requirement for different tables which in turn has resulted in less data being stored.

3.7 Trivia API

The Open Trivia API [12] was used to collect the questions, answers and multiple-choice options to be stored in the application database. It is free to use and is aimed at helping students gather data for programming exercises and projects. New questions, answers and multiple choice options can be added to the API by anyone, and it stores the data in JSON objects.

It is an example of a *third-party API* which is an external API managed by a third party and exists to allow anyone to obtain data to be used in their own programs.

To obtain the data from the Trivia API it is necessary to set up a connection from a server running in the student's project. To allow this connection the API provides a URL which is a direct link to obtain the JSON object with the requested data.

As shown in Figure 23, it is very simple to select the number of questions, category and difficulty level. Once the submit button is pressed an API URL is then provided which can be included in students programming scripts to collect data for their projects.

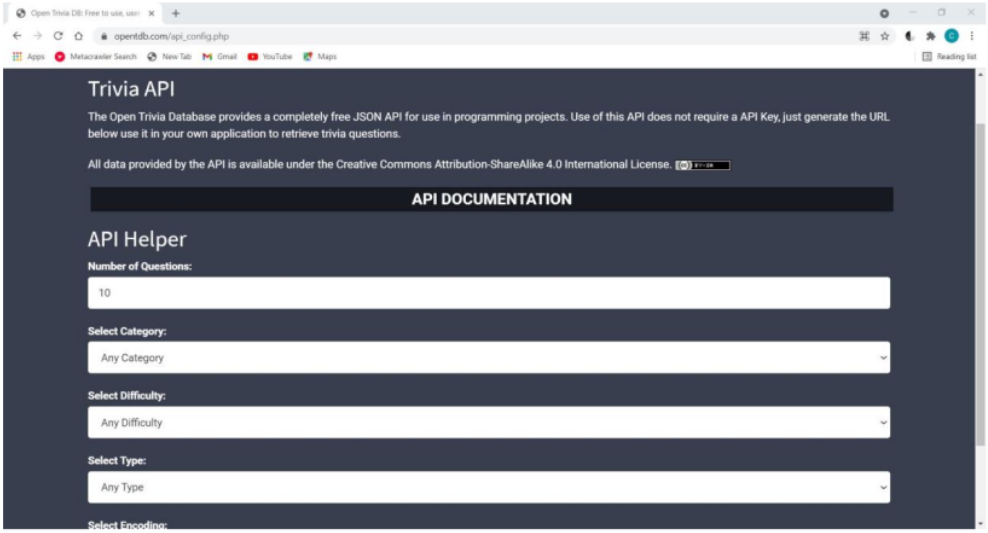

Figure 23 – Open Trivia URL Generator

Once the URL is generated, it is then necessary to copy and paste it into the programming script to establish the connection with the API and retrieve the data. As the data is in JSON format it needs to be parsed by the script code which adds some complexity to the process.

Overall, however, the attainment of data through the Trivia API is a very simple process and it is easy to use.

Chapter 4

IMPLEMENTATION

4.1 Methodology

The Waterfall Methodology was used for the development stage of this project. The reason is because the writer was working solo on the development and could not share any of the code with anyone else. Also, the requirements were fixed before work commenced on the implementation stage and so there was little that could change in terms of the features that were required and there was no customer to provide a changing set of requirements.

Work was completed in terms of milestones, and at each milestone a new draft of the code was saved and uploaded to the project github repository.

4.2 Review of Implemented Technology

The system has three main elements for which technology was used for the development, including the application front and back ends and the separate scraping service.

Application – Front End

HTML, CSS and Javascript were used for the front end of the application sub-system. The Bootstrap Framework was also used to deal with the styling of the web pages and to take advantage of the framework's buttons, tables and other useful tools.

An emphasis in this aspect of the development was for the application to be viewable across all devices including PC, mobile, I-Pad or differing screen sizes.

For example, the tabbed interface (Figures 3-5 and 9-11) was designed using custom CSS styling which allows percentages to be chosen for the width of the tabs rather than choosing pixels. This has ensured that the tabbed interface covers the entirety of the page on any device that it is viewed. An example of this code is shown below in Figure 24.

```
<a href="/" onclick='showHome();' style="width: 100"><a>
<a href="#status" onclick='showStatus();' style="width: 33%;"></a>
<a href="#history" onclick="showHistory();" style="width: 33%;"</a>
<a href="#account" onclick="showAccount();" style="width: 33%;"></a>
```

Figure 24 – Code for Width of Headings and Tabs

The code above ensures that the main headline for the RealTime Quiz is always set to 100% of any device that it is viewed. Furthermore, the 3 tabs underneath the headline tab are set to 1/3rd of the screen width ensuring they also cover 100% of any device or screen size.

On each database call, as much information was sent to the Front End for processing as possible, other than sensitive data (discussed in section 4.3 below). This was possible as Javascript has built in functionality to allow parts of the code to be hidden and not processed by the browser until an event is triggered by the user.

There are two main examples of this process including the tabbed interfaces for the home page and user profile, as well as the single question per page shown during the quiz. An example of how this was implemented is shown below in Figure 25.

HTML

```
<a id="rankings1" onclick="showRankings();" style="width: 33%;">Rankings</a>
```

Javascript

```
function showRankings(){
        document.getElementById("home").style.display = "none";
        document.getElementById("home1").className = "";
        document.getElementById("rankings").style.display ="block";
        document.getElementById("rankings1").className = "active";
        document.getElementById("about").style.display = "none";
        document.getElementById("about1").className = "";
    }
```

Figure 25 – Code for Hiding and Showing Front End UI

The rankings tab on the home page includes an onclick event which sends a request to the *showRankings* Javascript function. This in turn hides the home and about sections and instead only shows the user the rankings information.

Another challenging aspect of the front end of this application was to automatically change the answers during the quiz after the timer expires or when the user presses the next option. As shown in Figure 26, various functions were utilised for this functionality.

Upon the user selecting the next option during the phase when answers are being displayed, an onclick event is triggered by the HTML code and the answer number is parsed to the *fastForward* Javascript function. This function then resets the timer and calls the *getQuestion* function.

The *getQuestion* function is parsed the current question number and it sets the next question to be shown to the user and ensures all other questions are hidden. It also calls the *timer* function which sets the timer to 5 seconds for the next question.

If the user does not press the next button and the timer expires, then the *timer* function calls the *fastForward* function for the process to be reset and the next answer displayed. The whole process is then initiated again with the *fastForward* function calling the *getQuestion* function, which changes the answer and then calls the *timer* function again for the timer to be reset.

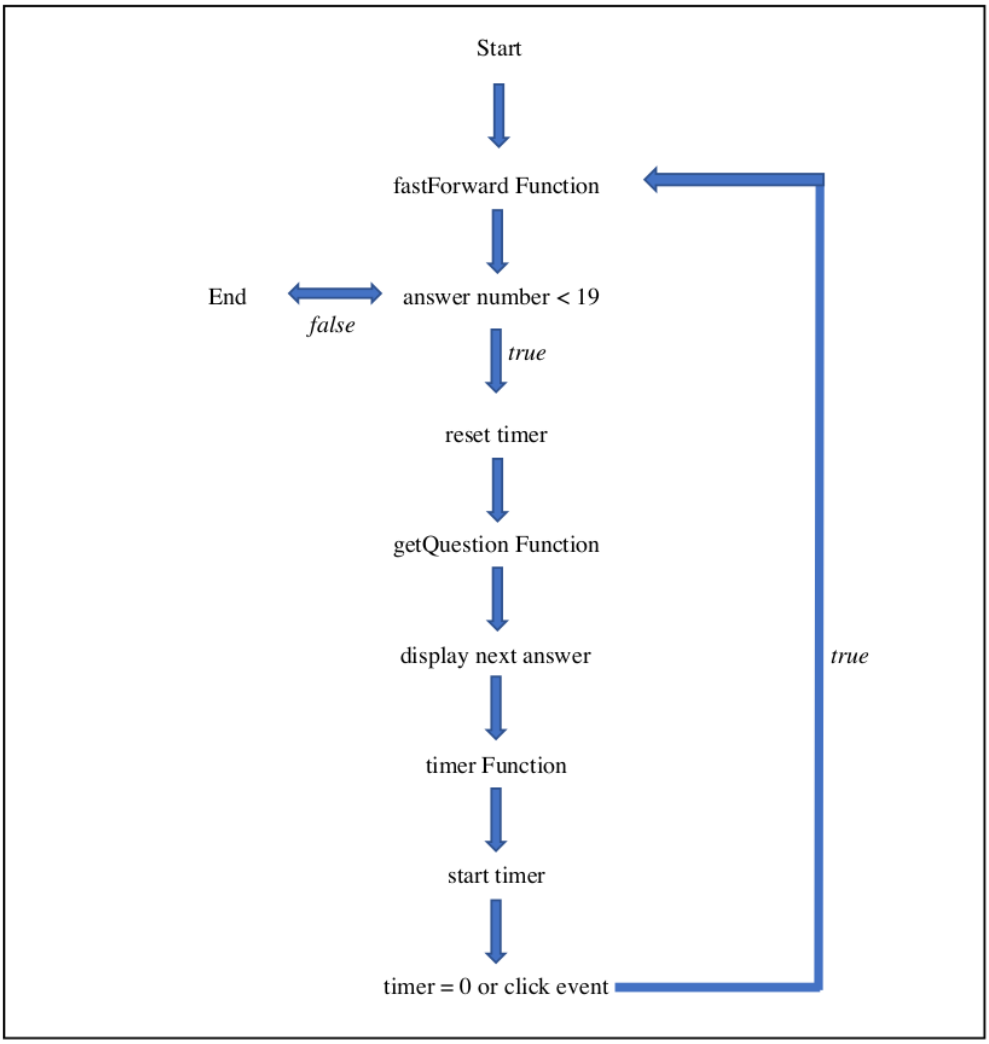

Figure 26 – Diagram of Questions and Timer Functions

Application – Back End

Javascript and the Node JS runtime environment were used for the application sub-system back end. Various modules from the Node.js package management system was used for the back end including the following:

- Express
- HTTP
- Body-Parser
- PG
- Nodemailer

Express is a Node.js web application framework that acts as middleware on the server-side of the application. It deals with the logic in relation to all API calls and sends requests to the database via the HTTP server. It also deals with all the routing that takes place which determines which web pages will be shown to the user.

Express is included in the application with the following code:

```
const express = require('express');
const app = express();
```

Figure 27 – Initialising Express and the App Object

The code in Figure 27 includes all the functionality of express into the app object. Upon each HTTP request and response that is made in the application, express has access to the objects being sent and can decide which route to send the request and all the server-side logic that is applied.

Upon the application first being requested in the browser, express is used to intercept the HTTP request using the following code snippet in Figure 28:

```
app.get('/', (req, res)=>{
```

Figure 28 – Routing Request for Home Page

The above function then gathers all the information from the database for the home page to be displayed with the updated rankings. It does this by compiling the data from the database into functions to apply logic to arrange the object to be included in the front-end script. For example, for the home page the object inserted into the front-end script is known as "j" and can be seen in the following snippet:

```
for (i = 0; i < size ; i++) {
        j+= '<tr style="text-align: center">';
        j+= '<td>' + '<b>' + (i+1) + '</b>' + '</td>';
        j+= '<td>' + '<b>' + resp.rows[i].nickname + '</b>' + '</td>';
        j+= '<td>' + '<b>' + resp.rows[i].avgimprovement + '%' + '</b>' + '</td>';
        j+= '</tr>';
    }
```

```
<tbody>
    {j}
</tbody>
```

Figure 29 – Creation of Rankings Table

The HTTP module is used to create the application server. It is built into the Node.js framework and allows Node.js to transfer data over the HTTP protocol. The Body Parser module is used to parse the JSON data in the objects received from the external Trivia API and the database.

Node.js also has a built in PG package which is a group of modules for interfacing with the PostgreSQL database. It provides functionality for a connection to be made with the database.

The Nodemailer module has been used to send automated emails when the user has forgotten their password and also to send the completed surveys. It is easy to set up and use and uses the application server to log into a dedicated RealTime Quiz email account and send emails to either the user if they have forgotten their password or to send the completed surveys to the author of this report.

The typical execution flow consists of requests being made by the client server and being sent to the HTTP application server. Then the HTTP server sends the request to the express middleware which then decides on the response, whether it is a logic or routing request, and sends it back to the HTTP server to be forwarded to the database server to either read data or store information.

Once a response is received from the database server, the application server then passes it back to the express app which then deals with the response and adds features to the response to be sent back to the client browser via the application server.

PostgreSQL was chosen for the database because of its compatibility with the Node JS environment and the node-postgres npm module which significantly eased the development process.

Scraping Service

The scraping service is included in a different directory to the main application. Although it also uses Javascript, Node JS and Express, it is not connected to the main application system in any way, and only serves to update the independent questions table in the database.

Although the main application does access the questions table it is only to read the data, and it does not in any way update the content of the questions table.

It is very similar to the flow of execution described for the application backend above, except there is no client involved or front end to this service. The processing all takes place on the backend with a separate HTTP scraping server passing a request to the Trivia API for data, handling the response object and storing the data inside the database.

Asynchronous Javascript was used to carry out the data collection process. As shown in the following code example at Figure 30, loops were used to scrape the data from each of the options offered by the API including the number of questions and difficulty levels.

On each API call the use of the ".then" and "await" methods ensure that the process of obtaining the data for each option was completed before any attempt was made at scraping the next option. This was important because without asynchronous javascript the program would have fired off another request before it had carried out the previous task and would lead to errors.

```
        for (p=0; p<3; p++){
            if (p==0){
                await fetch(`https://opentdb.com/api.php?amount=50&category=${i}&difficulty=
easy&type=multiple`).then(
                function(a){return a.json();}
            ).then(
                async function(general){
                    for (t=0; t< general.results.length;t++) {
                        var first = general.results[t].question;
                        var second = general.results[t].correct_answer;
                        var third = general.results[t].incorrect_answers[0];
                        var fourth = general.results[t].incorrect_answers[1];
                        var sixth = await pool.query(`insert into questions (question, correctanswer, option1, option2)
values ($1, $2, $3, $4)`,
                        [(`${first}`), (`${second}`), (`${third}`), (`${fourth}`)]);
                    }
                }


            )
            }
```

Figure 30 – Scraping API for Questions Data and Storing in Database

4.3 Security Issues

The focus here was on making sure no objects containing user data was sent to the front end for processing. If this happened, then it would be easy for anyone to intercept this and gain access to personal information such as passwords and email addresses.

To alleviate this concern all processing of sensitive information is completed on the server side, and then the answer itself is sent back to the front end. For example, rather than sending all passwords to the front end to be processed/matched by javascript functions on the client browser, this is all done on the server side instead.

The number of calls to the database could have been reduced significantly if sensitive information was sent to the front end when the browser initially loads. In this scenario there would be no need for any existing users to send a request to the database when logging into the application or to play the quiz, as the details in the users, questions and leaderboard tables could be packaged up and sent to the front end to be processed there.

However, this was not possible as the data contained in the users table is sensitive and could be easily intercepted and seen on the client side of the application. Thus, it was necessary to organise the application into many separate calls to the database which has an impact on the performance of the application.

4.4 Specific System Improvements

<u>Application and Scraping Service</u>

Once a working version of the application had been created the file size of the quiz.js application file and the api.js scraping file was as follows:

- Quiz.js – 133 kb
- Api.js – 100 kb

Analysis was made of the main application file with a view to increasing the load on the front end, whilst also ensuring that all processing of personal information was still completed on the server side.

Originally, separate calls to the database were made when a user requested the rankings table or attempts table. Consideration was given to reducing the number of database calls to send all the requisite information to the front end, and to only show the user specific aspects of the code and to hide the other parts of the code until an event was triggered and the front-end scripts could then change the interface in accordance with the request.

When the home page is loaded, all the information for the tabbed interface pages including the rankings and about section is sent to the front end at the same time from one database call. The pages are then hidden using front end Javascript until a click event is triggered.

The same approach was then taken with the rest of the application files, by minimising the amount of database calls and sending as much information to the front end as possible on each database call and showing/hiding pages by using front end Javascript.

The size of the scraping script was reduced by using an API rather than scraping the web, and by using a loop to iterate through each request.

Furthermore, the size of the files was then reduced more by minifying the code and creating new files called quiz.min.js and api.min.js to be used as the main application and scraping files. The size of the improved files is as follows:

- Quiz.min.js – 70 kb
- Api.min.js – 2 kb

<u>Database</u>

The original database plan consisted of 5 tables and a total of 17 attributes. This can be seen at Figure 31 below.

Due to the layout of this original schema plan, it was noted that issues relating to data redundancy would arise.

For example, there was no need to include the quiz table, the quiz ID, Question ID or user ID attributes. The reason for this is because the processing could be done on the application server and then placed into a more efficient and elegant table design in the database.

The new schema design is shown in Figure 22 above, which shows there are now only 3 tables and 12 attributes.

This reduction to the number of tables and attributes in the database has resulted in a more efficient and better performing database structure, which holds the minimum amount of data and rows as possible.

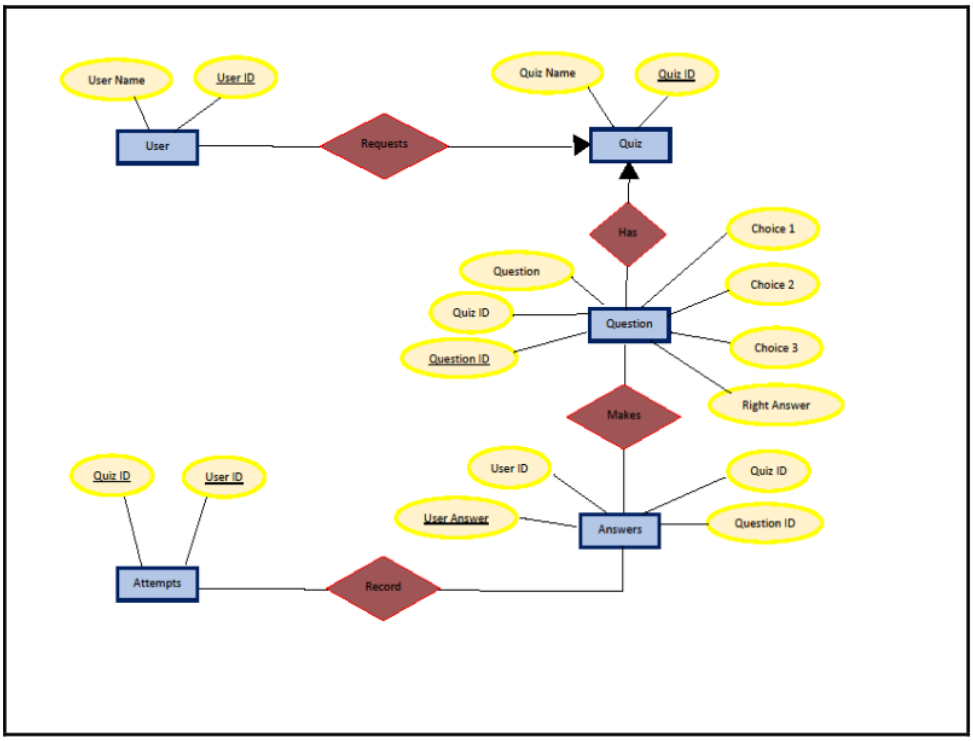

Figure 31 – ER Diagram of Old Database Design

4.5 Deployment

After the development was completed, attention then turned to deploying the application on the internet so that user testing could be undertaken, and surveys obtained for the preparation of this report.

Although the application could have been run locally on users' machines by following the instructions in the user manual appended to this report, it was thought that it would be much easier for any non-technical users to simply visit the application using a web browser.

4.6 Choosing a Platform

Various platforms were considered including paid-for platforms like Amazon Web Services (AWS), and the writer considered running a server on his home machine. However, this approach was not pursued as it would have involved keeping the machine switched on 24 hours a day and this was not viable.

Eventually it was decided that the application would be run for free on the Heroku platform. Heroku is a cloud application platform which is described as a Platform as a Service (PAAS). They enable applications to run entirely in the cloud and support the Node JS development language.

A plug in is available on the platform called Heroku-Postgres which allows users to obtain a free Postgres database for the application.

For the purposes of this report the free hobbyist plan was chosen, as it still allows up to 20 concurrent server calls at any time. As the number of people used for testing was relatively small the concurrent connections are adequate for the purposes of this report. However, because this is a free service the application goes to sleep after 30 minutes of inactivity. Thereafter, any new requests to load the home page in the browser have a time delay of around 5 to 10 seconds.

4.7 Configuring the Remote System

Because Heroku was compatible with the programming languages and database used in the development of the application, the configuration of the remote system was straightforward.

It is necessary to sign up for an account on Heroku, and then on the user profile section a new application can be created, and a name is chosen. The plug in for the Heroku-Postgres database can then be linked to the new project.

Once the database is created the credentials can be viewed on the Heroku site. These need to be included in the server side scripts to log into and update the database.

In order to log into the remote PostgreSQL database or to upload the application scripts for deployment, it is necessary to download the Heroku CLI. Once downloaded it can be viewed on the computer terminal and the username and password needs to be entered.

The tables for the database were then created by logging in to the database remotely. Following this process, the scraper script was run to update the questions table and a new user was created for the user table and an entry was also included in the leaderboard table. All of this was done using the Heroku Cli on the terminal.

The application and scraper scripts were then pushed to the remote cloud system by using the Heroku Cli and GIT. Once this was complete the deployment process was run and could be viewed in the client terminal. Afterwards the progress of the deployment is shown, and it can be seen whether it was successful. Then the application can be viewed by including the URL provided by the Heroku CLI on the terminal.

Chapter 5

TESTING

5.1 Manual Testing

A system of manual testing was used throughout the development due to the following factors:

- To gain a human perspective of the overall system performance, as if the writer was a user to gain an insight into the user experience
- As the user interface was an important part of this application, manual testing was used to improve the overall customer experience
- To gain a broader perspective of the overall system
- The size of the project was small, and the cost associated with manual testing was not overly large and could be jusitified

After the end of each milestone, consideration was given to the list of functional and non-functional requirements, and test cases were prepared for the functionality that had been added to the system.

These tests were carried out and any issues arising from the changes were fixed immediately. Once the changes were working correctly, the entire application was tested to ensure that the changes had not unexpectedly affected another part of the system.

This process of manual testing continued throughout the lifecycle of the development process until all the required features had been included in the application and it was ready for deployment.

5.2 Data Protection and Ethical Approval

Before approaching any people to carry out user testing it was necessary to consider the terms and constraints imposed by the UK General Data Protection Regulations (UK GDPR).

As data revolves around the issue of ethics and individuals' right to privacy, the author was required to gain the approval of the GDPR Officer at Birkbeck University before deploying the application on the internet and inviting people to participate in the project.

Under the GDPR it is necessary to be fully transparent with users about why you need their data, what it will be used for, who will have access to the data and how the user can be removed from the database and their details deleted. The full GDPR compliant statement is shown in Figure 8 which explicitly sets this out.

Before agreeing to register to play the application, users must agree to the data protection statement and so they are fully aware of why their data is captured, who will have access to it and how they can be removed from the database.

5.3 Live Testing

After the project had been deployed to the cloud platform and the remote database was created and configured, it was necessary to test the online application before approaching anyone to take the quiz.

All the testing that had been carried out in the development phase were repeated when this application was deployed online, to ensure there were no hidden errors or compatibility issues.

This stage of testing consisted of checking whether the application worked and was viewable across different web browsers and screen sizes, and whether the time taken to retrieve information from the Database was overly long and tedious.

Browser Compatibility Tests

As different browsers read and interpret the same code differently, it was necessary to check whether the application was fully functional and viewable on any given browser. As the author developed the application on a windows device, the following windows compatible browsers were used to carry out the browser compatibility tests:

- Google Chrome
- Edge Internet Explorer
- Mozilla Firefox

The testing at this stage involved viewing thte application in each of the browsers, playing quiz and checking whether the application was viewable and whether the core functionality remained consistent on a cross-browser basis.

On each browser used in the compatibility testing, the application was fully functional, and the only difference recorded was the display of the font for the main logo for the RealTime Quiz.

Different Devices and Screen Sizes

It was essential the application could be viewed on any device including PC, mobile and I-Pads. Numerous screen sizes were tested on the Google Chrome Developer Platform which allows any webpage to be viewed across a wide spectrum of mobile devices including I-Pads, I-Phones, Samsung Galaxy and many other mobile devices.

In addition, the application was also viewed on a 28-inch second monitor and a 15-inch laptop. The different screens for the entire application were viewed and cross-referenced to detect any differences between the devices and screen sizes.

There were many issues with how the application was presented on different screen sizes and during this aspect of the testing a continuous iterative approach was undertaken in amending the code, and going back through each device and if necessary, making further amendments until the display was satisfactory.

Database Calls and Speed

The speed of the application was a paramount concern due to the main requirement of the application that users should enjoy playing the quiz and be motivated to keep playing it repeatedly **(Finding 2)**. Once the application had been placed on the Heroku Cloud Platform and a remote Database had been set up, the concern was that this would significantly increase the amount of time taken to retrieve information from the Database and display the result to the user.

It was already known that the application would take some time to initially load in the browser (discussed in the implementation chapter). This was unavoidable and so the testing for speed focussed instead for any activity after the application had been initially rendered on the user's client device.

This included the time taken to check passwords, login, register, display the questions and answers, calculate the result of the quiz and changing the username or password. All these functions were tested to ensure that the wait time was within milliseconds to ensure a pleasurable user experience.

The results of the speed testing were very positive, with all parts of the system (apart from the initial upload of the application), returning and displaying database call results almost immediately.

5.4 User Testing

A total of 35 people were asked to undertake the quiz, and the web address was provided along with some background regarding the use of brain training games to improve memory. The subjects used during the testing were chosen to ensure a wide spectrum of different people to have a broad range of opinions in the user surveys. There are young (18-40), middle aged (40-60) and older (60+) people that have used the application.

All the people chosen to take part in the testing of the application are known to the author as being friends, direct relatives or relatives of friends. This approach did carry some risk of the feedback being mostly subjective as friends and relatives could be seen as not completely impartial.

However, the messages that were sent out explicitly asked for the application to be played and honest feedback given, as the application was not finalised, and the aim was to use the feedback to improve the application so that it could help people with memory difficulties. Furthermore, to encourage completely honest feedback, no user details are sent with the surveys and so any feedback remains anonymous apart from the user's age.

Due to the research carried out for this project, it became apparent that the most benefit of brain training games was for people who may suffer Dementia in the future. As this can potentially happen to anyone as they get older, it was beneficial to have a broad range of opinions and feedback.

Although almost all the people did register and play the quiz itself, not everyone completed the survey. 18 completed surveys were received, which is an uptake of around 56% of users who played the quiz. In retrospect it may have been better to have made the survey mandatory after the first attempt at the quiz, rather than including an option to take the survey (Figure 14).

The survey is included at Figure 16 above. The questions themselves were mostly yes or no answers as the purpose was to make the survey as easy as possible to ensure a high uptake. Direct questions such as 'Do you think this app will improve memory' were used and a text box is included at the end of the survey for any comments or suggestions to improve the application.

All submitted results from the surveys were emailed to me automatically, as the system was programmed to send emails with this information directly to my personal email account. All user information such as their name was excluded from this process, so the surveys are completely anonymous.

5.5 Data Analysis from User Testing

All survey results are annexed to this report. 100% of the surveyed users said that they thought the application would improve memory, which is a very encouraging sign that this application could potentially be useful if developed further. Most of the other answers were also positive in the survey form. There were some issues and suggestions made to improve the application, and these are shown in Figure 32 below.

	Number of People	Percentage
The timers for the answers were too fast	6	33%
The home page is slow to load	4	22%
Inclusion of other non-quiz memory games	2	11%
Inclusion of quiz categories	2	11%
Inclusion of timers for questions	1	5.5%
Search functionality for the rankings	1	5.5%
Difficulty option for the quiz	1	5.5%

Figure 32 – User Testing Results

Most of the suggestions to improve the application related to the length of time allocated to remember each question and answer. The application only allocates 5 seconds for users to digest and remember the question and answer, which is a short amount of time especially for elderly users. It is also noted however, that there were no users who criticised the timers in principle.

As shown in Figure 32 above, a total of 4 people criticised the speed in which the home page is loaded. However, as mentioned in the implementation chapter, this was a compromise that was taken to obtain the free hosting for the application, and something that can be fixed very easily if the application is to be developed and extended by opting for a paid-for plan.

The rest of the suggestions in Figure 32 relate to the introduction of further features, and all of them are useful suggestions and could be introduced if the application was extended and developed further.

Overall, the results from the user testing were very positive, and several issues were highlighted which could be taken forward with this application.

Chapter 6

CONCLUSIONS

6.1 How the Project Meets its Objective

Overall, the objective has been met as the application uses word-list recall to provide a quiz and a time limit for users to remember answers. The history of users' attempts is reflected in the rankings and attempts tables and users can check on the progress of friends that also play the quiz.

It is easy to use and all the people who played the quiz and completed a survey believed that the application would improve memory.

Despite the project meeting its overall objective, it is not perfect and has various limitations which are discussed below.

6.2 Critical Comparisons with Similar Applications

Compared to other applications of this type the quiz application is quite narrow in scope and does not for example provide any other memory games or information such as health and fitness, which are also helpful to improve memory and lower the risk of Dementia. The user interface of the application is basic in comparison to similar applications of this type.

However, it has been found that similar applications are lacking in relation to their adherence to the approach taken by the leading ACTIVE study. This was commented upon by the leading scientist involved in the study [11] insofar as there was no use of the word list recall or other methods used during the study.

The RealTime Quiz does adhere to the approach of ACTIVE as it combines the use of speed of processing training with word list recall. It also has made a significant effort to be as easy as possible for users to play and to motivate the users to keep playing. The quiz is completely free to play, unlike Lumosity and many other brain training games.

6.3 Potential Improvements to Features

This project could be extended further by including more of the functionality used by similar brain training applications, whilst keeping the emphasis on the word list recall aspect of the quiz which is not used by other brain training apps.

Variations of word list recall could be used to introduce more games so that the application is multi-dimensional. This would make the user experience more pleasurable and would add more variety to the application and encourage people to keep training their memory.

It would be beneficial to introduce categories for the questions, and this would help keep the user interested in the quiz if they like a particular subject. Also, an option to select a difficulty level for the questions as well as categories of the questions would make the user experience more pleasurable. These features could be introduced quite easily as the Trivia API provides functionality to select categories and difficulty level.

The data for the questions and answers is only collected once and stored in the database. The process of collecting the data should be set to run automatically at daily intervals if this application was to be taken forward with the suggestions in this report.

Furthermore, the rankings table is not sustainable if the quiz is used more extensively by people as it would render the table too long and impossible for the user to look up their name or the name of any

friends. Therefore, a search tool for the rankings table is a necessity if the application is to be developed further.

The timer for the answers could be increased to 10 seconds, and there could be a timer for the questions as well. Also, some functionality to promote better health and fitness could be included in the application.

6.4 Overall Conclusion of the Project

The quiz application is overall a successful resource that could be used to help with brain training and for people who are liable to suffer from Dementia. It complies with the approach taken by the ACTIVE study and offers functionality for word list recall not available in other brain training applications.

The quiz has limitations such as performance, and the features are very limited. There are other applications on the market that are discussed above, that offer a wider variety of memory games, and provide access to health and fitness functionality which has been noted to also help with overall brain health and lowers the risk of developing Dementia.

Although there is no conclusive evidence that brain training games prevent the onset of Dementia, the results obtained by the leading ACTIVE study are compelling.

As we have seen from the research, a significant improvement to the risk of developing Dementia was achieved using speed of processing training for people who are not suffering from Dementia but are at risk of developing it in the future without the use of mental exercise.

The current brain training applications on the market have not fully adopted the approach taken during the ACTIVE study and there is the potential for significant improvement to improve the applications to fight against onset of Dementia.

The RealTime Quiz could seek to bridge that gap by keeping the existing features and adopting the functionality included in other applications such as promoting health and fitness and introducing a wider variety of games for users to enjoy.

BIBLIOGRAPHY

[1] Small G, Parlow-Solomon (2002): What we need to know about age related memory loss. BMJ. 22; 324(7352): 1502–1505. doi: 10.1136/bmj.324.7352.1502

[2] Friedland RP, Fritsch T, et al (2001). Patients with Alzheimer's disease have reduced activities in midlife compared with healthy control-group members. Proc Natl Acad Sci USA. 2001;98:3440–3445

[3] Lezak MD. *Neuropsychological assessment.* 3th ed. New York: Oxford University Press; 1995. pp. 366–368.

[4] Stanford Center on Longevity (2014): A Consensus on the Brain Training Industry from the Scientific Community (Summary). https://longevity.stanford.edu/a-consensus-on-the-brain-training-industry-from-the-scientific-community-/

[5] Cognitive Training Data (2014): Cognitive Training Data Response Letter. https://www.cognitivetrainingdata.org/the-controversy-does-brain-training-work/response-letter/

[6] Rebok G W et al (2014): Ten-year effects of the ACTIVE study on cognition and everyday functioning in older adults: Journal of the American Geriatrics Society.

[7] Edwards J, Xu Huiping, O.Clark D et al. (2017) Speed of processing training results in lower risk of dementia. Alzheimer's & Dementia: Translational Research & Clinical Interventions 3 (2017) 603-611

[8] https://www.mindmate-app.com

[9] https://www.lumosity.com

[10] Klodian Dhana et al (2020): Healthy lifestyle and the risk of Alzheimer dementia: Neurology Jul 2020, 95 (4) e374-e383; DOI: 10.1212/WNL. 0000000000009816

[11] Lewis Tanya (2016): The scientist behind the most comprehensive study of brain training says apps like Lumosity are based on shaky science. Business Insider. https://www.businessinsider.com/science-of-brain-training-2016-1?r=US&IR=T

[12] https://opentdb.com/api_config.php

USER MANUAL

Hopefully, when this project is examined, it will still be possible to visit the application on the web by entering the URL in method 1 below. However, if the application server is down for any reason, then the more long-winded method 2 can be followed to view the application locally.

There are two folders that have been created on Github to run this application.

(1) Deployment – files that have been deployed to Heroku
(2) Development – files that can be used to run the application locally

Method 1 - Internet

To view the application running on the internet, the following web address is placed into any browser on a PC, I-Pad, Mobile Phone or any other device.

https://realtimed-quiz.herokuapp.com/

As the free hobby tier was chosen from the Heroku platform, it is likely there will be a wait of around 5 to 10 seconds for the application to initially load as discussed in the implementation chapter.

Method 2 - Locally

Any steps that require running commands below refer to the development directory of the project and not the deployment directory.

Step 1

Clone the Github repository by running the following terminal command *git clone https://github.com/Birkbeck/msc-information-technology-project-2020_21---files-Chris-Y-ates.git*

Step 2

NodeJS needs to be downloaded onto your local machine. If you do not already have node installed, you can download it by visiting https://nodejs.org/en/download/

Step 3

Download PostgreSQL and PGAdmin by visiting https://www.postgresql.org/download/

Step 4

During the setup choose the following credentials for the database:

Host: localhost

User Name: postgres

Password: password

Port: 5432

Step 5

Open the PG Admin GUI and create a database by right-clicking on the Databases option. Save the database as *realtime*.

Step 6

Once NodeJS and PostgreSQL have been installed it is necessary to initialise the npm packages by using the *npm init* terminal command within the development directory. Press enter for all options until the directory has been initialised.

Step 7

Now the node modules need to be uploaded. To do this run the *npm install* command in the terminal within the development directory.

Step 8

Now the questions table can be updated by scraping the Trivia API. To do this change directory into the scraping sub-directory and run the following terminal command:

nodemon api.min.js

Step 9

Once the countdown is finished in the terminal, close the scraper server with the *Control + C* command

Step 10

Now the directory needs to be changed again to the main development directory in the terminal.

Step 11

Run this terminal command to start the application server:

nodemon quiz.min.js

Step 12

The application can now be viewed in the browser at http://localhost:8080/

Full Listing of Code

package.json, Procfile, api.js, quiz.js

package.json

```
{
  "name": "msc-project1",
  "version": "1.0.0",
  "description": "Real-Time Quiz App",
  "main": "quiz.js",
  "scripts": {
    "test": "node quiz.js",
    "start": "nodemon quiz.min.js"
  },
  "author": "Chris Yates",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "dotenv": "^10.0.0",
    "express": "^4.17.1",
    "node-fetch": "^2.6.1",
    "nodemailer": "^6.6.2",
    "nodemon": "^2.0.9",
    "pg": "^8.6.0"
  }
}
```

Procfile

```
"web: nodemon quiz.min.js"
```

api.js

```
const express = require('express');
const app = express();
const {Pool} = require('pg');
const fetch = require('node-fetch');
require('dotenv').config();
app.use(express.json());

let pool = new Pool();
const port = process.env.PORT;

var count = 32;

async function start (){
    try {
        await pool.connect()
        await pool.query(`TRUNCATE public.questions`)

        for (i=1; i<33; i++){
            console.log(`Countdown ${count}`);
            count--;
            for (p=0; p<3; p++){
                if (p==0){
                    await fetch(`https://opentdb.com/api.php?amount=50&categor
y=${i}&difficulty=easy&type=multiple`).then(

                    function(a){return a.json();}
                ).then(
                    async function(general){
                        for (t=0; t< general.results.length;t++) {
                            var first = general.results[t].question;
                            var second = general.results[t].correct_answer;
                            var third = general.results[t].incorrect_answers[0
];
                            var fourth = general.results[t].incorrect_answers[
1];
                            var sixth = await pool.query(`insert into public.q
uestions (question, correctanswer, option1, option2) values ($1, $2, $3, $4)`,

                            [(`${first}`), (`${second}`), (`${third}`), (`${fo
urth}`)]);
                        }
                    }

                )
                }
                else if (p==1){
```

```
                    await fetch(`https://opentdb.com/api.php?amount=50&categor
y=${i}&difficulty=medium&type=multiple`).then(

                        function(a){return a.json();}
                    ).then(
                        async function(general){
                            for (t=0; t< general.results.length;t++) {
                                var first = general.results[t].question;
                                var second = general.results[t].correct_answer
;
                                var third = general.results[t].incorrect_answe
rs[0];
                                var fourth = general.results[t].incorrect_answ
ers[1];
                                var sixth = await pool.query(`insert into publ
ic.questions (question, correctanswer, option1, option2) values ($1, $2, $3, $
4)`,
                                [(`${first}`), (`${second}`), (`${third}`), (`
${fourth}`)]);
                            }
                        }

                    )
                }
                else {
                    await fetch(`https://opentdb.com/api.php?amount=50&categor
y=${i}&difficulty=hard&type=multiple`).then(

                        function(a){return a.json();}
                    ).then(
                        async function(general){
                            for (t=0; t< general.results.length;t++) {
                                var first = general.results[t].question;
                                var second = general.results[t].correct_answer
;
                                var third = general.results[t].incorrect_answe
rs[0];
                                var fourth = general.results[t].incorrect_answ
ers[1];
                                var sixth = await pool.query(`insert into publ
ic.questions (question, correctanswer, option1, option2) values ($1, $2, $3, $
4)`,
                                [(`${first}`), (`${second}`), (`${third}`), (`
${fourth}`)]);
                            }
                        }

                    )
```

```
                }

            }
        }
        console.log("Fetching Has Now Finished");
    }catch(error){
        console.log(error);
    }
}


start();


app.listen(port, ()=>{
    console.log(`server started on port ${port}`);
})
```

quiz.js

```
 "const express = require('express');
const app = express();
var server = require('http').createServer(app);
const bodyParser = require("body-parser");
const {Pool} = require('pg');
var nodemailer = require('nodemailer');

let pool = new Pool({
    user: "postgres",
    password: "corbett99",
    host: "localhost",
    port: 5432,
    database: "realtime"
});

var regcompleted = false;
var regagain = false;
var isntreg = false;
var passwrong = false;
var userpass= false;
var passpass = false;
var passsent = false;
var newdetails = false;

const port = 8080;
app.use(express.json());
app.use(bodyParser.urlencoded({ extended: true }));

let date_ob = new Date();
let date = ("0" + date_ob.getDate()).slice(-2);
let month = ("0" + (date_ob.getMonth() + 1)).slice(-2);
let year = date_ob.getFullYear();

var userDate = year + "-" + month + "-" + date;

function shuffle(array) {
    var currentIndex = array.length,  randomIndex;

    // While there remain elements to shuffle...
    while (0 !== currentIndex) {

      // Pick a remaining element...
      randomIndex = Math.floor(Math.random() * currentIndex);
```

```
      currentIndex--;

      // And swap it with the current element.
      [array[currentIndex], array[randomIndex]] = [
        array[randomIndex], array[currentIndex]];
    }

    return array;
  }

  app.get('/', (req, res)=>{
    try{
        list = [];
        userChoice=[];
        firstchoice = [];
        correctchoice = [];
        secondchoice = [];
        arr = [];
        counts = 0;
        userEmail = "";

    pool.connect(async (error, client, release)=>{
        let desp = await client.query(`SELECT nickname, MAX(attemptnumber) as
maxattempt FROM public.leaderboard
        GROUP BY nickname ORDER BY maxattempt DESC`)

        var size = desp.rows.length;

        let resp = await client.query(`SELECT nickname,
ROUND(avg(improvement)) as avgimprovement
        FROM public.leaderboard
        GROUP BY nickname
        ORDER BY avgimprovement DESC`);

        var j = '';

        for (i = 0; i < size ; i++) {
            j+= '<tr style="text-align: center">';
            j+= '<td>' + '<b>' + (i+1) + '</b>' + '</td>';
            j+= '<td>' + '<b>' + resp.rows[i].nickname + '</b>' + '</td>';
            j+= '<td>' + '<b>' + resp.rows[i].avgimprovement + '%' + '</b>' +
'</td>';
            j+= '</tr>';
        }
        client.release();

        res.send(`<!DOCTYPE html>
        <html lang="en">
```

```
        <head>
            <meta charset="UTF-8">
            <meta http-equiv="X-UA-Compatible" content="IE=edge">
            <meta name="viewport" content="width=device-width, initial-
scale=1.0">
            <title>Real-Time Quiz App</title>
            <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
            crossorigin="anonymous">
            <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scrip
t>
            <script>

            var nowreg = "You Are Now Registered";
            var alreadyreg = "You Are Registered";
            var notreg = "You Are Not Registered";
            var wrongcode = "Incorrect Password";
            var userwrong = "Incorrect Password <br> Login Again";
            var passemailed = "Password Sent";
            var changedetails = "Details Changed <br> Login Again";

            </script>
            <style>

                h1 {
                    font-family: Fantasy;
                }
                body {
                    text-align: center;
                    background: powderblue;
                }

                table {
                    border-collapse: collapse;
                    border: none !important;

                }

                h2 {
                    text-align: center;
                    color: white;
                    font-family: Fantasy;
                }

                th {
```

```
                text-align: center;
            }

            form {
                display: inline-block;
            }

            .achievements-wrapper {
                height: 100%;
                width: 100%;
                overflow-y: auto;
                text-align: center;
                position: fixed;
            }

            input{
                text-align:center;
             }

             th {
                position: sticky;
                top: 0px;
                background: white;
                color: black;
              }

              /* Add a black background color to the top navigation */
                .topnav {
                background-color: #333;
                overflow: hidden;
                }

                /* Style the links inside the navigation bar */
                .topnav a {
                float: left;
                color: #f2f2f2;
                text-align: center;
                padding: 24px 5%;
                text-decoration: none;
                font-size: 17px;
                }

                /* Add a color to the active/current link */
                .topnav a.active {
                background-color: #04AA6D;
                color: white;
                }
        </style>
```

```
        </head>
        <body>
        <div class="topnav">
            <a href="/" onclick='showHome();' style="width: 100%; background-
color: black;"><h2>&#9200 RealTime Quiz</h2></a>
            <a id="home1" href="/" onclick='showHome();' class="active"
style="width: 33%;">Home</a>
            <a id="rankings1" href="#rankings" onclick="showRankings();"
style="width: 33%;">Rankings</a>
            <a id="about1" href="#about" onclick="showAbout();" style="width:
33%;">About</a>
        </div>
                    <div id="home" style="padding-top: 20px;">
                        <h3 id="homeMessage">Enter Details to Play!</h3>
                        <form action="/info/add" method="POST"><br>
                            <p id="manyemail" style="color: red;
display:none;"><b>Max Characters 40</b></p>
                            <input type="email" name="email" id="email"
maxlength="40" placeholder='Enter Email Address' style="margin-left: 10px
font-size: 18pt; height: 30px; width:260px;" required>
                            <br><br>
                            <p id="manypassword" style="color: red;
display:none;"><b>Max Characters 20</b></p>
                            <input type="password" name="password"
id="password" maxlength="20" placeholder='Enter Your Password' style="margin-
left: 5px font-size: 18pt; height: 30px; width:260px;" required><br><br>

                            <div style="width:250px;" align="center">
                                <button type="submit" class="btn btn-primary
btn-block" style="font-size: 11pt; width:260px; height: 34px;">Login</button>
                            </div><br>
                        </form><br>
                        <form action="/registration" method="GET">
                        <div style="width:260px; align: center;">
                                <button type="submit" class="btn btn-primary
btn-block" style="font-size: 11pt; width:260px; height:
34px;">Registration</button>
                            </div><br>
                        </form><br>
                        <form action="/forgotten/password" method="GET">
                        <div style="width:260px; align: center;">
                                <button type="submit" class="btn btn-primary
btn-block" style="font-size: 11pt; width:260px; height: 34px;">Forgotten
Password</button>
                            </div><br>
                        </form>
                    </div>
```

```
            <div id="rankings" class="span3 achievements-wrapper"
style="display:none;">

                        <table class="table table-striped table-light">
                        <th colspan="1" class="table-grey"><h4>POS</h4></th>
                        <th colspan="1" class="table-grey"><h4>NAME</h4></th>
                        <th colspan="1" class="table-grey"><h4>%IMP</h4></th>
                            <tbody>
                                ${j}
                            </tbody>
                        </table>
                    </div>

                    <div class="achievements-wrapper" id="about"
style="display:none;"><br>
                    <h5 style="text-align: left; margin-left:10px;margin-
right:10px;">Who Can Benefit</h5>
                    <p style="text-align: left;margin-left:10px;margin-
right:10px;">This App has been developed to assist in memory/brain training
                    to help improve cognitive function. The focus of the quiz
is on users improvement in memory recall over a number of attempts at the
quiz,
                     rather than the number of correct answers selected on
their first attempt. The ranking system reflects this focus.</p>
                    <h5 style="text-align: left;margin-left:10px;margin-
right:10px;">How it Works</h5>
                    <p style="text-align: left;margin-left:10px;margin-
right:10px;">Once registered you will see a list of questions with the correct
answers.
                    You will have a short time to remember as many answers as
possible before the quiz begins.
                    In the user profile section functionality is included for
users to track their progress and the progress of any friends who also play
the quiz.
                    </p>
                    </div>

        </body>
        <script>
                    if (${regcompleted} == true){
                        document.getElementById("homeMessage").innerHTML =
nowreg;
                    }
                    if (${regagain} == true){
                        document.getElementById("homeMessage").innerHTML =
alreadyreg;
                    }
                    if (${isntreg} == true){
```

```
                        document.getElementById("homeMessage").innerHTML =
notreg;
                    }
                    if (${passwrong} == true){
                        document.getElementById("homeMessage").innerHTML =
wrongcode;
                    }

                    if (${userpass} == true || ${passpass} == true){
                        document.getElementById("homeMessage").innerHTML =
userwrong;
                    }

                    if (${passsent} == true){
                        document.getElementById("homeMessage").innerHTML =
passemailed;
                    }

                    if (${newdetails} == true){
                        document.getElementById("homeMessage").innerHTML =
changedetails;
                    }

                    $("#email").on('input', function() {
                        if ($(this).val().length >= 40) {
                            document.getElementById("manyemail").style.display
= "block";
                        } else{
                            document.getElementById("manyemail").style.display
= "none";
                        }
                    });

                    var $input = $('input')
                    $input.keyup(function(e) {
                        var max = 40;
                        if ($input.val().length > max) {
                            $input.val($input.val().substr(0, max));
                        }
                    });

                    $("#password").on('input', function() {
                        if ($(this).val().length >= 20) {

document.getElementById("manypassword").style.display = "block";
                        } else{

document.getElementById("manypassword").style.display = "none";
```

```
                        }
                    });

                    function showHome(){
                        document.getElementById("home").style.display =
"block";
                        document.getElementById("home1").className = "active";
                        document.getElementById("rankings").style.display =
"none";
                        document.getElementById("rankings1").className = "";
                        document.getElementById("about").style.display =
"none";
                        document.getElementById("about1").className = "";
                    }
                    function showRankings(){
                        document.getElementById("home").style.display =
"none";
                        document.getElementById("home1").className = "";
                        document.getElementById("rankings").style.display =
"block";
                        document.getElementById("rankings1").className =
"active";
                        document.getElementById("about").style.display =
"none";
                        document.getElementById("about1").className = "";
                    }
                    function showAbout(){
                        document.getElementById("home").style.display =
"none";
                        document.getElementById("home1").className = "";
                        document.getElementById("rankings").style.display =
"none";
                        document.getElementById("rankings1").className = "";
                        document.getElementById("about").style.display =
"block";
                        document.getElementById("about1").className =
"active";
                    }
        </script>
        </html>`)
        regagain = false;
        regcompleted = false;
        isntreg = false;
        passwrong = false;
        userpass = false;
        passpass = false;
        newdetails = false;
    })
```

```
    }catch(error){
        console.log(error);
    }
});

app.get('/registration', (req, res)=>{
    try{

        res.send(`<!DOCTYPE html>
        <html lang="en">
        <head>
            <meta charset="UTF-8">
            <meta http-equiv="X-UA-Compatible" content="IE=edge">
            <meta name="viewport" content="width=device-width, initial-
scale=1.0">
            <title>Real-Time Quiz App</title>
            <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
            crossorigin="anonymous">
            <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scrip
t>

            <style>

                body {
                    text-align: center;
                    background-color: powderblue;
                }

                form {
                    display: inline-block;
                }

                h2 {
                    text-align: center;
                    color: white;
                    font-family: Fantasy;
                }

                input{
                    text-align:center;
                 }

                 .topnav {
                 background-color: #333;
```

```
                overflow: hidden;
                position: sticky;
                }

                .topnav a {
                float: left;
                color: #f2f2f2;
                text-align: center;
                padding: 24px 5%;
                text-decoration: none;
                font-size: 17px;
                }

                .topnav a.active {
                background-color: #04AA6D;
                color: white;
                width: 25%;
                }

                .login  {
                   background-color: #0057fc;
                   color: white;
               }

            </style>
        </head>
        <body>
        <div class="topnav fixed-top">
            <a href="/" style="width: 100%; background-color:
black;"><h2>&#9200 REAL-TIME QUIZ</h2></a>
        </div>
        <div id="disclaimer" style="margin-left: 10%; margin-right: 10%;
text-align: justify;text-justify: inter-word; display:none;">
            <h3 style="margin-top: 3%; text-align: center;">DATA
PROTECTION</h3><br>
            By pressing the Submit Button you agree to your data being used
for the purpose of my university project. This app has
            been designed to eventually help people that have difficulty with
their memory. Your
            user name, password, email address and the record of your attempts
at this quiz will be stored in a database. Your data will be
            used to analyse how effective the application is at improving
memory, and will only be used to prepare a
            university project report and for no other purpose. Personal
information such as your password, name and email address will be
            anonymised and not linked to you directly in any way when referred
to in my research report. No marketing agencies, governing bodies
```

```
            or any other outside organisation will be provided access to your
data. The only people who have access to your data will be the
            person directly involved in the research for this university
project and noone else. Your participation in this project is entirely
            voluntary, and you can choose to be removed from the database and
project report by emailing me at cy1yates@gmail.com by 15 August 2021.

        </div>
            <div id= "form" style="padding-top: 1%;">
                        <div id="form1">
                        <h4>Step 1 - Enter Details</h4>
                        </div><br>
                        <form action="/adduser" method="POST">
                            <div>
                            <label for="email" id="manyemail" style="color:
red; display: none;"><b>Max 40 Email Characters</b></label>
                            </div>
                            <div id="form2">
                            <input type="email" name="email" id="email"
maxlength="40" placeholder='Enter Email Address' style="margin-left: 10px
                            font-size: 18pt; height: 30px; width:260px;"
required><br>
                            </div>
                            <div>
                            <label for="nickname" id="manynickname"
style="color: red; display: none; margin-bottom: -10px;"><b>Max 14 Name
Characters</b></label>
                            </div>
                            <div id="form3">
                            <input type="text" name="nickname" id="nickname"
maxlength="14" placeholder='Enter Your Name' style="margin-left: 10px
                            font-size: 18pt; height: 30px; width:260px;margin-
top: 5%;" required><br>
                            </div>
                            <div>
                            <label for="password" id="manypassword"
style="color: red;display: none;"><b>Max 20 Password Characters</b></label>
                            </div>
                            <div id="form4">
                            <input type="password" name="password"
id="password" maxlength="20" placeholder='Enter Your Password' style="margin-
left:
                            5px font-size: 18pt; height: 30px;
width:260px;margin-top: 5%;" required><br>
                            </div>
                            <div id="form5">
                            <input type="password" name="password_confirm"
id="password_confirm" maxlength="20" placeholder='Re-Enter Your Password'
```

```
                            style="margin-left: 5px font-size: 18pt; height:
30px; width:260px;margin-top: 5%;"
                            oninput="check(this)" required><br>
                            </div>
                            <script language='javascript'
type='text/javascript'>
                                function check(input) {
                                    if (input.value !=
document.getElementById('password').value) {
                                        input.setCustomValidity('Password Must
be Matching.');
                                    } else {
                                        // input is valid -- reset the error
message
                                        input.setCustomValidity('');
                                    }
                                }
                            </script>
                            <div id="button" class="row justify-content-
center" style="display:none;">
                                <button for="form" type="submit" class="btn
btn-primary btn-block" style="font-size: 18pt; width:300px;">Agree and
Register</button>
                            </div><br>

                        </form>
                    </div>
                    <div id="button2" class="row justify-content-center">
                        <button type="submit" class="btn btn-primary btn-
block" onclick="step2();" style="height: 12%;font-size: 18pt;
width:100%;position:
                        absolute; bottom:0;margin-right:20%;margin-
left:20%;">Go to Step 2</button>
                    </div><br>

              <script>
              var $input = $('input')
                    $input.keyup(function(e) {
                        var max = 40;
                        if ($input.val().length > max) {
                            $input.val($input.val().substr(0, max));
                        }
                    });

              $("#email").on('input', function() {
                if ($(this).val().length >= 40) {
                    document.getElementById("manyemail").style.display =
"block";
```

```
                } else{
                    document.getElementById("manyemail").style.display =
"none";
                }
            });

            $("#password").on('input', function() {
                if ($(this).val().length >= 20) {
                    document.getElementById("manypassword").style.display =
"block";
                } else{
                    document.getElementById("manypassword").style.display =
"none";
                }
            });

            $("#nickname").on('input', function() {
                if ($(this).val().length >= 14) {
                    document.getElementById("manynickname").style.display =
"block";
                } else{
                    document.getElementById("manynickname").style.display =
"none";
                }
            });

            function step2(){
                document.getElementById("form1").style.display = "none";
                document.getElementById("form2").style.display = "none";
                document.getElementById("form3").style.display = "none";
                document.getElementById("form4").style.display = "none";
                document.getElementById("form5").style.display = "none";
                document.getElementById("button2").style.display = "none";
                document.getElementById("disclaimer").style.display = "block";
                document.getElementById("button").style.display = "block";
            }
              </script>
        </body>
        </html>`)

    }catch(error){
        console.log(error);
    }
});

var register = "";

app.post('/adduser', (req, res)=>{
```

```
    try {
        pool.connect(async(error,client,release)=>{
            let resp = await client.query(`SELECT name, useremail from
public.user`)

            registeragain = false;

            nickName = req.body.nickname;

            for (i=0;i<resp.rows.length;i++){
                if(req.body.email == resp.rows[i].useremail){
                    registeragain = true;

                } if (req.body.nickname == resp.rows[i].name) {
                    registeragain = true;
                }
            }
            if (registeragain == true) {
                client.release();
                regagain = true;
                res.redirect('/')
            } else{
                let addregister = await client.query(`insert into public.user
(useremail, name, password) values ($1, $2, $3)`, [(req.body.email),
(req.body.nickname), (req.body.password)]);
                client.release();
                regcompleted = true;
                res.redirect('/');
            }
        })
        register = '';

    }catch(error){
        console.log(error);
    }
});

app.get('/forgotten/password', (req, res)=>{
    try{
        res.send(`<!DOCTYPE html>
        <html lang="en">
        <head>
            <meta charset="UTF-8">
            <meta http-equiv="X-UA-Compatible" content="IE=edge">
            <meta name="viewport" content="width=device-width, initial-
scale=1.0">
            <title>Real-Time Quiz App</title>
```

```
            <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
            crossorigin="anonymous">
            <script src='quiz.js'></script>

            <style>
                body {
                    text-align: center;
                    background: powderblue;
                }

                form {
                    display: inline-block;
                }

                h1 {
                    text-align: center;
                }

                input{
                    text-align:center;
                 }
                    .topnav {
                    background-color: #333;
                    overflow: hidden;
                    }

                    .topnav a {
                    float: left;
                    color: #f2f2f2;
                    text-align: center;
                    padding: 24px 5%;
                    text-decoration: none;
                    font-size: 17px;
                    }

                    .topnav a.active {
                    background-color: #04AA6D;
                    color: white;
                    }

                    .login  {
                        background-color: #0057fc;
                        color: white;
                    }
            </style>
```

```
        </head>
        <body>
        <div class="topnav">
            <a href="/" onclick='showHome();' style="width: 100%; background-
color: black;"><h1>&#9200 RealTime Quiz</h1></a>
        </div>
            <div style="padding-top: 5%;">

                        <h4>Get Password Reminder</h4><br>
                        <form action="/get/password" method="POST">

                            <input type="email" name="email" id="email"
maxlength="40" placeholder='Enter Email Address' style="margin-left: 10px
font-size: 18pt; height: 40px; width:260px;" required><br><br>
                            <div style="width:250px;" align="center">
                                <button type="submit" class="btn btn-block
login" style="font-size: 18pt; width:260px;">Retrieve Password</button>
                            </div><br>
                        </form>
                    </div>
        </body>
        </html>`)

    }catch(error){
        console.log(error);
    }
});

app.post('/get/password', async(req,res)=>{

    try{
        pool.connect(async(error,client,release)=>{
            let checkpassword = await client.query(`select password from
public.user where useremail = '${req.body.email}'`);

            if (checkpassword.rows.length > 0){
                var transporter = nodemailer.createTransport({
                    service: 'outlook',
                    auth: {
                      user: 'real_time_quiz@outlook.com',
                      pass: 'Realtimequiz'
                    }
                  });

                  var mailOptions = {
                    from: 'real_time_quiz@outlook.com',
                    to: `${req.body.email}`,
                    subject: 'Forgotten Password',
```

```
                    text: `Your Password is:
${checkpassword.rows[0].password}`
                  };

                  transporter.sendMail(mailOptions, function(error, info){
                    if (error) {
                      console.log(error);
                    }
                  });
                client.release();
                passsent = true;
                res.redirect('/')
            }
        })
    }catch(error){
        console.log(error);
    }
})

app.post('/info/add', async (req, res)=>{
    nickName = "";
    userEmail = "";
    var userexists = 0;
    var passwordexists = false;

    try {
        pool.connect(async(error,client,release)=>{
            let checkuser = await client.query(`select * from public.user`);

            for (i=0;i<checkuser.rows.length;i++){

                if(checkuser.rows[i].useremail == req.body.email &&
checkuser.rows[i].password == req.body.password){
                    userexists = 0;
                    passwordexists = true;
                    break
                }
                else if (checkuser.rows[i].useremail == req.body.email &&
checkuser.rows[i].password != req.body.password){
                    userexists = 0;
                    passwordexists = false;
                    break
                } else {
                    userexists = 1;

                }
            }
```

```
            let findemail = await client.query(`select * from public.user,
public.leaderboard where public.leaderboard.nickname
            = public.user.name and public.user.useremail =
'${req.body.email}'`);

            if (userexists == 1){
                client.release();
                isntreg = true;
                res.redirect('/')
            }else if (passwordexists == false){
                client.release();
                passwrong = true;
                res.redirect('/')
            }else if (findemail.rows.length > 0){
                nickName += findemail.rows[0].name;
                userEmail += findemail.rows[0].useremail;
                client.release();
                res.redirect('/user/profile');
            } else{
                pool.connect(async (error, client, release)=>{
                    var notontable = await client.query(`select * from
public.user`);
                    var lengthnoton = notontable.rows.length;
                    nickName += notontable.rows[lengthnoton - 1].name;
                    userEmail += notontable.rows[lengthnoton - 1].useremail;
                    client.release();
                    res.redirect('/user/profile');
                })
            }
        })
    }catch(error){
        console.log(error);
    }
})

app.get('/user/profile', async(req, res) => {

    try {
        list = [];
        userChoice=[];
        firstchoice = [];
        correctchoice = [];
        secondchoice = [];
        arr = [];
        counts = 0;
        var status = "";

        pool.connect(async (error, client, release)=>{
```

```
            let leader = await client.query(`select * from public.leaderboard
where nickname = '${nickName}' order by attemptnumber asc`);
            var getleader = "";

            for (i = 0; i < leader.rows.length; i++) {
                getleader+= '<tr>';
                getleader+= '<td style="width: 35%">' + '<b>' + 'Attempt' + '
' + leader.rows[i].attemptnumber + '</b>' + '</td>';
                getleader+= '<td style="width: 35%">' + '<b>' +
leader.rows[i].date + '</b>' + '</td>';
                getleader+= '<td style="width: 35%">' + '<b>' +
leader.rows[i].numbercorrect + '/20' + '</b>' + '</td>';
                getleader+= '</tr>';
        } if(leader.rows.length < 15){
            for (i=0; i<15; i++) {
                getleader+= '<tr height="40px">';
                getleader+= '<td style="width: 35%">' + '</td>';
                getleader+= '<td style="width: 35%">' + '</td>';
                getleader+= '<td style="width: 35%">' + '</td>';
                getleader+= '</tr>';
            }
        }

        if (leader.rows.length <= 1){
            status += '<div style="text-align: center;">';
            status += '<h1>&#128373</h1><h4>BEGINNER</h4><br><br><h5
style="text-align: center;">Your Journey is just beginning.<br><br>';
            status += 'Good luck!</h5></div>';
        }

        else if ((leader.rows[leader.rows.length-1].improvement) >
(leader.rows[leader.rows.length-2].improvement)){
            status += '<div style="text-align: center;"><h1>&#128077;&#127999;
&#127942';
            status += '&#x1F44D</h1><h4>IMPROVER</h4><br><br><h5 style="text-
align: center;">Your Memory is Improving. Keep Going!</h5><br></div>';
        }

        else if ((leader.rows[leader.rows.length-1].improvement) <
(leader.rows[leader.rows.length-2].improvement)) {
            status += '<div style="text-align: center;">';
            status += '<h1>&#128078 &#128530
&#128078;&#127998;</h1><br><h4>POORER</h4><br><br><h5 style="text-align:
center;">';
            status += 'Keep playing to improve your memory.</h5><br></div>';
        }
```

```
        else if ((leader.rows[leader.rows.length-1].improvement) <
(leader.rows[leader.rows.length-2].improvement) &&
leader.rows[leader.rows.length-2].improvement == 20){
            status += '<div style="text-align: center;">';
            status += '<h1>&#128580 &#129335
&#128580;</h1><br><h4>STEADY</h4><br><br><h5 style="text-align: center;">';
            status += 'Your Memory has remained consistent.<br><br>Keep Going
to Maintain your Memory.</h5><br></div>';
        }
        else{
            status += '<div style="text-align: center;">';
            status += '<h1>&#128580 &#129335
&#128580;</h1><br><h4>STEADY</h4><br><br><h5 style="text-align: center;">';
            status += 'Your Memory has remained consistent.<br><br>Play more
to improve further.</h5><br></div>';
        }
        client.release();

            res.send(`<!DOCTYPE html>
            <html lang="en">
            <head>
                <meta charset="UTF-8">
                <meta http-equiv="X-UA-Compatible" content="IE=edge">
                <meta name="viewport" content="width=device-width, initial-
scale=1.0">
                <title>Real-Time Quiz App</title>
                <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
                crossorigin="anonymous">
                <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scrip
t>
                <script src="https://kit.fontawesome.com/758fbfa7b4.js"
crossorigin="anonymous"></script>

                <style>

                body{
                    background-color:powderblue;
                }

                p {
                    text-align: center;
                }
                h5 {
                    text-align: center;
```

```
        color: #34282C;
        font-family: Fantasy;
    }

    h4 {
        text-align: center;
    }

    th {
        text-align: center;
    }

    .achievements-wrapper {
        height: 100%;
        width: 100%;
        overflow-y: auto;
        text-align: center;
        position: fixed;
    }

     th {
        background: white;
      }

    .topnav {
    background-color: #333;
    overflow: hidden;
    position: sticky;
    top: 0px;
    }

    .topnav a {
    float: left;
    color: #f2f2f2;
    text-align: center;
    padding: 24px 5%;
    text-decoration: none;
    font-size: 17px;
    }

    .topnav a.active {
    background-color: #04AA6D;
    color: white;
    }

    .login  {
        background-color: black;
        color: white;
```

```
                }

                </style>
            </head>

            <body>
            <div class="topnav">
            <a href="/" onclick='showHome();' style="width: 100%; background-
color: black;"><h2>&#9200 RealTime Quiz</h2></a>
            <a id="status1" href="#status" onclick='showStatus();'
class="active"  style="width: 33%;">Status</a>
            <a id="history1" href="#history" onclick="showHistory();"
style="width: 33%;">Attempts</a>
            <a id="account1" href="#account" onclick="showAccount();"
style="width: 33%;">Account</a>
        </div>
        <div id="status">
        <br>
                ${status}
            <form action="/info/get" method="GET">
                <div class="row justify-content-center">
                    <button type="submit" class="btn btn-primary btn-block"
style="padding-top: 15px !important;padding-bottom: 55px !important;font-size:
20pt; width: 100%; height: 10%; position: absolute; bottom:0;">Play the Quiz
Now!</button>
                </div><br>
            </form>
        </div>

        <div id="history" class="span3 achievements-wrapper"
style="display:none;">
            <table class="table table-striped table-light">
                <tbody>
                    ${getleader}
                </tbody>
            </table>
         </div>

         <div id="account" style="text-align:center; display: none;"><br>
         <h6 style="text-align: center;">${nickName}<br>Account Area </h6>

         <div class="row justify-content-center" id="typeuser">

         <form action="/change/user" method="POST">
             <div>
             <div>
             <label for="nickname" id="manynickname" style="color: red;
display: none; margin-bottom: -10px;"><b>Max 14 Name Characters</b></label>
```

```
            </div>
            <input type="text" name="nickname" id="nickname" maxlength="14"
placeholder='Enter New Username' style="margin-left: 10px font-size: 6pt;
height: 28px; width:260px;margin-top: 5%; text-align: center;" required><br>
            <div>
            <label for="password" id="manypassword" style="color:
red;display: none;"><b>Max 20 Password Characters</b></label>
            <div>
            <input type="password" name="password" id="password"
maxlength="20" placeholder='Enter Your Password' style="margin-left: 5px font-
size: 6pt; height: 28px; width:260px;margin-top: 5%;text-align: center;"
required><br>

            <div style="width:250px;" align="center">
                <button type="submit" class="btn btn-block login"
style="font-size: 12pt; width:260px; height: 38px;margin-top: 5%;">Update
Username</button>
            </div>
        </form>
    </div>

    <div class="row justify-content-center" id="typepassword" style="padding-
top: 2%;">

        <form action="/change/password" method="POST">
            <div>
            <div>
            <label for="password" id="manypassword" style="color: red;text-
align: center; display: none;"><b>Max 20 Password Characters</b></label>
            <div>
            <input type="password" name="newpassword" id="newpassword"
maxlength="20" placeholder='Enter New Password' style="margin-left: 5px font-
size: 6pt; height: 28px; width:260px;margin-top: 5%;text-align: center;"
required><br>
            <input type="password" name="password" id="password"
maxlength="20" placeholder='Enter Old Password' style="margin-left: 5px font-
size: 6pt; height: 28px; width:260px;margin-top: 5%;text-align: center;"
required><br>

            <div style="width:250px;" align="center">
                <button type="submit" class="btn btn-block login"
style="font-size: 12pt; width:260px; height: 38px;margin-top: 5%;">Update
Password</button>
            </div><br>
        </form>
    </div>
    </div>
    <script>
```

```
            var $input = $('input')
                $input.keyup(function(e) {
                    var max = 40;
                    if ($input.val().length > max) {
                        $input.val($input.val().substr(0, max));
                    }
                });

            $("#email").on('input', function() {
            if ($(this).val().length >= 40) {
                document.getElementById("manyemail").style.display = "block";
            } else{
                document.getElementById("manyemail").style.display = "none";
            }
            });

            $("#password").on('input', function() {
            if ($(this).val().length >= 20) {
                document.getElementById("manypassword").style.display =
"block";
            } else{
                document.getElementById("manypassword").style.display =
"none";
            }
            });

            $("#nickname").on('input', function() {
            if ($(this).val().length >= 14) {
                document.getElementById("manynickname").style.display =
"block";
            } else{
                document.getElementById("manynickname").style.display =
"none";
            }
            });

            function showStatus(){
                document.getElementById("status").style.display = "block";
                document.getElementById("status1").className = "active";
                document.getElementById("history").style.display = "none";
                document.getElementById("history1").className = "";
                document.getElementById("account").style.display = "none";
                document.getElementById("account1").className = "";
            }
            function showHistory(){
                document.getElementById("status").style.display = "none";
                document.getElementById("status1").className = "";
                document.getElementById("history").style.display = "block";
```

```
                document.getElementById("history1").className = "active";
                document.getElementById("account").style.display = "none";
                document.getElementById("account1").className = "";
            }
            function showAccount(){
                document.getElementById("status").style.display = "none";
                document.getElementById("status1").className = "";
                document.getElementById("history").style.display = "none";
                document.getElementById("history1").className = "";
                document.getElementById("account").style.display = "block";
                document.getElementById("account1").className = "active";
            }
        </script>
        </body>
        </html>`);
        });

    } catch(error){
        console.log(error);
    }
});

app.post('/change/user', (req, res)=>{
    try{
        pool.connect(async(error,client,release)=>{
            let checkuser = await client.query(`select * from public.user`);
            var passwordmatch = 0;

            for (i=0;i<checkuser.rows.length;i++){

                if(checkuser.rows[i].name == nickName &&
checkuser.rows[i].password == req.body.password){
                    passwordmatch ++;
                }
            }
            if (passwordmatch == 1){
                let user = await client.query(`UPDATE public.user SET name =
'${req.body.nickname}' WHERE name = '${nickName}'`)
                let user2 = await client.query(`UPDATE public.leaderboard SET
nickname = '${req.body.nickname}' WHERE nickname = '${nickName}'`)
                newdetails = true;
                res.redirect('/')
            } else {
                userpass = true;
                res.redirect('/')
            }
            client.release();
        })
```

```
        newdetails = false;
    } catch(error){
        console.log(error)
    }
})

app.post('/change/password', (req, res)=>{
    try{
        pool.connect(async(error,client,release)=>{
            let checkuser = await client.query(`select * from public.user`);
            var passwordmatch = 0;

            for (i=0;i<checkuser.rows.length;i++){

                if(checkuser.rows[i].name == nickName &&
checkuser.rows[i].password == req.body.password){
                    passwordmatch ++;
                }
            }

            if (passwordmatch == 1){
                let user = await client.query(`UPDATE public.user SET password
= '${req.body.newpassword}' WHERE name = '${nickName}'`)
                client.release();
                newdetails = true;
                res.redirect('/')
            } else {
                client.release();
                passpass = true;
                res.redirect('/')
            }
        })
        newdetails = false;
    } catch(error){
        console.log(error)
    }
})

app.get('/info/get', (req, res)=>{

    list = [];
    firstchoice = [];
    correctchoice = [];
    secondchoice = [];

    try{
    pool.connect(async (error, client, release)=>{
        let resp = await client.query(`SELECT * FROM public.questions`);
```

```
        let choices = [];

        for (i = 0; i < 20; i++) {
            x = Math.floor(Math.random() * (1200 - 1 + 1)) + 1;
            choices.push(x);
            list.push(resp.rows[x].question)
        }
        for (i = 0; i < 20; i++) {
            firstchoice.push(resp.rows[choices[i]].option1)
        }

        for (i = 0; i < 20; i++) {
            secondchoice.push(resp.rows[choices[i]].option2)
        }

        for (i = 0; i < 20; i++) {
            correctchoice.push(resp.rows[choices[i]].correctanswer)
        }

        var j = "";

        for (i = 0; i < list.length; i++) {

            arr = [];
            for (t = 0; t < 1; t++) {
                arr.push(correctchoice[i]);

                j+=`<div id=${i} class="my-container" style="margin-left: -
3.2em; margin-right: -1.2em; display:none">`;
                j+=`<h4 style="margin-left: 1.2em;">Question ${i+1}<br></h4><p
style="font-size: 20px;margin-left: 1.4em;">${list[i]}</p>`+'<br>';
                j+=`<label class="container"><h5 style="margin-left: -
0.8em;">Answer: ${arr[0]}</h5>
                </label><br>`;
                j+=`<a href="#" onclick="fastForward(${i})">><h4
style="margin-left: 1.2em;">Next >>></h4></a>`;
                j+=`</div>`;
            };
        };

        var q = "";

        for (i = 0; i < 20; i++) {

            arr2 = [];
            for (t = 0; t < 1; t++) {
                arr2.push(firstchoice[i]);
```

```
                arr2.push(secondchoice[i]);
                arr2.push(correctchoice[i]);

                shuffle(arr2);

                q+=`<div id=${21+i} class="my-container" style="margin-left: -
3.2em; margin-right: -1.2em; display:none">`;
                q+=`<h4 style="margin-left: 1.2em;">Question ${i+1}</h4><p
style="font-size: 20px;margin-left: 1.4em;">${list[i]}</p>`+'<br>';
                q+=`<label class="container" style="margin-left:
1.8em;"><p>${arr2[0]}</p>
                <input type="checkbox" id="checkvalue2" name="option"
value='${arr2[0]}' onclick="seeNext(${21+i});">
                <span class="checkmark"></span>
                </label>`;
                q+=`<label class="container" style="margin-left:
1.8em;"><p>${arr2[1]}</p>
                <input type="checkbox"  id="checkvalue2" name="option"
value='${arr2[1]}' onclick="seeNext(${21+i});">
                <span class="checkmark"></span>
                </label>`;
                q+=`<label class="container" style="margin-left:
1.8em;"><hp>${arr2[2]}</hp>
                <input type="checkbox"  id="checkvalue2" name="option"
value='${arr2[2]}' onclick="seeNext(${21+i});">
                <span class="checkmark"></span>
                </label>`;
                q+=`</div>`;

            };
        };

        client.release();

        res.send(`<!DOCTYPE html>
        <html lang="en">
        <head>
            <meta charset="UTF-8">
            <meta http-equiv="X-UA-Compatible" content="IE=edge">
            <meta name="viewport" content="width=device-width, initial-
scale=1.0">
            <title>Document</title>
            <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
            crossorigin="anonymous">
```

```
            <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scrip
t>
            <script>
            $('.container').click(function() {
                $("#checkvalue").prop("checked", true);
              });
            </script>

            <style>

            body {
                background-color:powderblue;
            }

            .my-container {
                height: 340px;
                justify-content: space-around;
                margin: 20px 0px 0px 20px;
            }

            .container {
                display: block;
                position: relative;
                padding-left: 50px;
                padding-right: 50px;
                margin-bottom: 18px;
                width: 100%;
                cursor: pointer;
                font-size: 18px;
                -webkit-user-select: none;
                -moz-user-select: none;
                -ms-user-select: none;
                user-select: none;
                margin: 10px 20px 20px -5px;
              }

            .container input {
              position: absolute;
              opacity: 0;
              cursor: pointer;
              height: 0;
              width: 0;
            }

            .checkmark {
              position: absolute;
              top: 0;
```

```
    left: 0;
    height: 25px;
    width: 25px;
    background-color: grey;
  }

  .container:hover input ~ .checkmark {
    background-color: #ccc;
  }

  .container input:checked ~ .checkmark {
    background-color: #2196F3;
  }

  .checkmark:after {
    content: "";
    position: absolute;
    display: none;
  }

  .container input:checked ~ .checkmark:after {
    display: block;
  }

  .container .checkmark:after {
    left: 9px;
    top: 5px;
    width: 5px;
    height: 10px;
    border: solid white;
    border-width: 0 3px 3px 0;
    -webkit-transform: rotate(45deg);
    -ms-transform: rotate(45deg);
    transform: rotate(45deg);
  }

  .jumbotron {
      height: 40px;
      padding-top: 25px !important;
      padding-bottom: 75px !important;
      margin: 0px 15px 30px 30px;
  }

  button {
      height: 50px;
      width: 290px;
      margin-top: 5%;
  }
```

```
            h2 {
                margin-left: 20%;
            }
            #myProgress {
                width: 100%;
                background-color: #ddd;
                vertical-align:middle;

              }

            #myBar {
                width: 0%;
                height: 40px;
                background-color: #04AA6D;
                text-align: center;
                vertical-align: middle;
                line-height: 35px;
                font-size: 25px;
                color: white;
              }
                    .topnav {
                    background-color: #333;
                    overflow: hidden;
                    }

                    .topnav a {
                    float: left;
                    color: #f2f2f2;
                    text-align: center;
                    padding: 24px 5%;
                    text-decoration: none;
                    font-size: 17px;
                    }

                    /* Add a color to the active/current link */
                    .topnav a.active {
                    background-color: #04AA6D;
                    color: white;
                    }
            </style>
        </head>
        <body>
        <div class="topnav">
        <a href="/" onclick='showHome();' style="width: 100%; background-
color: black;"><h1>&#9200 RealTime Quiz</h1></a>

        </div>
```

```
                <div id="myProgress">
                <div id="myBar"></div>
             </div>
            <div id="checkvalue" class="container" style="width: 100%;">

            <form>
                <div id="form"><p style="color:purple;margin-left: -
1.4em;"><b> Disappears in <span id="countdowntimer">5 </span>
seconds</b></p>${j}</div>
                <div id="form2"><p style="color:purple;margin-left: -
1.4em;display:none;"><b> Question Disappears in <span id="countdowntimer2">5
</span> Seconds</b></p></div>
            </form>
            </div>

            <div id="checkvalue2" class="container" ">

            <form action="/answers" method="POST">

            <div id="form">${q}</div>

            <div id="button2" style="display: none;"><br><br>
                <label for="answers"><h4>Press Button to See
Results</h4></label><br>
                <button type="submit" id="answers" class="btn btn-primary btn-
lg">Submit Answers</button>
            </div>
            </form>
            </div>
            <div id="button" style="display: none;margin-left: 5%;"><br><br>

                    <label for="answers"><h4>Push Button Below to
Begin</h4></label><br>
                    <button id="answers" class="btn btn-primary btn-lg"
onclick="seeFirstofAll(21);">See the Questions</button>

                </div>
            <script>
            var width = 0;
            var length = 0;
            let downloadTimer;
            let downloadTimer2;
            let myTimeOut;
            let myTimeOut2;
            let timeleft = 5;

            function getQuestion(number) {
                myTimeOut = setTimeout(function(){
```

```
                if (number != 19){
                    clearInterval(downloadTimer);
                    notnext = document.getElementById(number);
                    notnext.style.display = "none";
                    nextNumber = number + 1;
                    passnext = document.getElementById(nextNumber);
                    passnext.style.display = "block";
                    nobutton = document.getElementById("button");
                    nobutton.style.display = "none";
                    var elem = document.getElementById("myBar");
                    width += 5;
                    elem.style.width = width + "%";
                    elem.innerHTML = width + "%";
                    timeleft = 5;
                    timer(nextNumber);
                }

            }, 100);
        }

        function fastForward(number){
            timeleft = 0;
            console.log("fastForward")
            console.log(number);
            clearTimeout(myTimeOut);
            if (number < 19){
                getQuestion(number);
            } else{
                notnext = document.getElementById(number);
                notnext.style.display = "none";
                passnext = document.getElementById("button");
                passnext.style.display = "block";
                document.getElementById("form").style.display = "none";
            }

        }

        var size = document.getElementById("checkvalue");
        for(i=0; i<size.innerHTML.length; i++){
            if(i==0){
                document.getElementById("button").style.display = "none";
                document.getElementById(0).style.display = "block";
                var elem = document.getElementById("myBar");
                var width = 5;
                elem.style.width = width + "%";
                elem.innerHTML = width +  "%";
                timer(0);
            } else{
```

```
                    document.getElementById(i).style.display = "none";
                }
            }
            async function timer(number){
                clearTimeout(myTimeOut);
                console.log("timer started");
                downloadTimer = setInterval(function(){
                document.getElementById("countdowntimer").textContent =
timeleft;
                if(number != 30 && timeleft == 0){
                    clearInterval(downloadTimer);
                    fastForward(number);
                }
                timeleft -= 1;
            }, 1500);
            }

            async function seeFirstofAll(digit){
                document.getElementById("button").style.display = "none";
                seeFirst(digit);
            }
            function seeFirst(digit){
                if (digit == 21){
                    console.log("seeFirst digit is" + digit);
                    document.getElementById(digit).style.display = "block";
                    var elem = document.getElementById("myBar");
                    width = 5;
                    elem.style.width = width + "%";
                    elem.innerHTML = width + "%";
                    console.log("seeFirst != 40");
                }
            }

            function seeNext(digit){
                setTimeout(function(){
                    if (digit != 40) {
                        document.getElementById(digit).style.display = "none";
                        passnext = document.getElementById("button");
                        passnext.style.display = "none";
                        document.getElementById(digit+1).style.display =
"block";
                        var elem = document.getElementById("myBar");
                        width += 5;
                        elem.style.width = width + "%";
                        elem.innerHTML = width + "%";
                        console.log("seeNext != 40");
                    } else if (digit == 40){
                        document.getElementById(digit).style.display = "none";
```

```
                        document.getElementById("button2").style.display =
"block";
                        width += 10;
                        console.log("seeNext == 40");
                    }
                }, 100);
            }
                </script>
        </body>
        </html>`)
    })
    }catch(error){
        console.log(error);
    }
});

app.post('/answers', async(req, res) => {

    var oneoption = "";

    const body = (req.body.option);

    if(req.body.option[0].length == 1) {
        for(i=0;i<req.body.option.length; i++){
            oneoption += req.body.option[i]
        }
        userChoice.push(oneoption);
    } else{
        for (i=0; i < body.length; i++) {
            userChoice.push(body[i])
        };

    }
    var counts = 0;
    var difference = [];
    for (i=0; i < userChoice.length; i++) {
        for (p=0; p < correctchoice.length; p++){
            if (userChoice[i] == correctchoice[p]) {
                difference.push(userChoice[i]);
            }
        }
    }

    counts = difference.length;
    var j = "";

    for (i = 0; i < userChoice.length; i++) {
```

```
        j+='<br>';
        j+=`<hr style="height:1px;border:none;color:#333;background-
color:#333;">`;
        j+=`<div class="my-container" style="margin-left: -10px;">`;
        j+=`<h3 style="margin-left: -10px;">Question ${i+1}</h3>`;
        j+= `<h4 style="margin-left: -10px;">${list[i]}</h4>`+'<br>';

        if(userChoice[i]==correctchoice[i]){
            j+=`<h3 style="margin-left: -10px; color:green;">CORRECT</h3>`;
            j+=`<label class="container" style="margin-left: -
10px;">${firstchoice[i]}
            <input type="checkbox" id="correct1" name="option"
value='${firstchoice[i]}'>
            <span class="checkmark"></span>
            </label>`;
            j+=`<label class="container" style="margin-left: -
10px;">${secondchoice[i]}
            <input type="checkbox"  id="correct2" name="option"
value='${secondchoice[i]}'>
            <span class="checkmark"></span>
            </label>`;
            j+=`<label class="container" style="margin-left: -
10px;">${correctchoice[i]}
            <input type="checkbox" checked="checked"  id="correct3"
name="option" value='${correctchoice[i]}'>
            <span class="checkmark"></span>
            </label>`;
            j+=`<br></div>`;
        } else if(userChoice[i]==firstchoice[i]){
            j+=`<h3 style="margin-left: -10px;color:red;">INCORRECT</h3>`
            j+=`<label class="container2" style="margin-left: -
10px;">${firstchoice[i]}
            <input type="checkbox" checked="checked" id="incorrect1"
name="option" value='${firstchoice[i]}'>
            <span class="checkmark2"></span>
            </label>`;
            j+=`<label class="container" style="margin-left: -
10px;">${secondchoice[i]}
            <input type="checkbox"  id="incorrect2" name="option"
value='${secondchoice[i]}'>
            <span class="checkmark"></span>
            </label>`;
            j+=`<label class="container" style="margin-left: -
10px;">${correctchoice[i]}
            <input type="checkbox" checked="checked" id="incorrect3"
name="option" value='${correctchoice[i]}'>
            <span class="checkmark"></span>
            </label>`;
```

```
            j+=`</div>`;
        }
        else{
            j+=`<h3 style="margin-left: -10px; color:red;">INCORRECT</h3>`
            j+=`<label class="container" style="margin-left: -
10px;"">${firstchoice[i]}
            <input type="checkbox" id="incorrect1" name="option"
value='${firstchoice[i]}'>
            <span class="checkmark"></span>
            </label>`;
            j+=`<label class="container2" style="margin-left: -
10px;">${secondchoice[i]}
            <input type="checkbox" checked="checked" id="incorrect2"
name="option" value='${secondchoice[i]}'>
            <span class="checkmark2"></span>
            </label>`;
            j+=`<label class="container" style="margin-left: -
10px;">${correctchoice[i]}
            <input type="checkbox" checked="checked" id="incorrect3"
name="option" value='${correctchoice[i]}'>
            <span class="checkmark"></span>
            </label>`;
            j+=`</div>`;
        }
    };

    try {
        pool.connect(async (error, client, release)=>{

            let desp = await client.query(`select * from public.leaderboard
where
            public.leaderboard.nickname = '${nickName}'`);

            var firstattempt = 0;
            var base = 0;
            var howimproved = 0;

            if (desp.rows.length < 1) {
                userAttempt = 1;
                howimproved = 0;
            } else {
                userAttempt = desp.rows.length + 1;
                firstattempt = desp.rows[0].numbercorrect;
                base = counts - firstattempt;
                howimproved = Math.round((base / firstattempt)*100);
            }
```

```
            let resp = await client.query(`insert into public.leaderboard
(attemptnumber, nickname, date, numbercorrect, improvement) values ($1, $2,
$3, $4, $5)`, [(userAttempt), (nickName), (userDate), (counts),
(howimproved)]);

            client.release();

            res.send(`<!DOCTYPE html>
            <html lang="en">
            <head>
                <meta charset="UTF-8">
                <meta http-equiv="X-UA-Compatible" content="IE=edge">
                <meta name="viewport" content="width=device-width, initial-
scale=1.0">
                <title>Real-Time Quiz App</title>
                <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
                crossorigin="anonymous">
                <style>
                body{
                    background-color:powderblue;
                }
                h1 {
                    text-align: center;
                    padding-top: 15px !important;
                }

                th {
                    text-align: center;
                }

                .my-container {
                    height: 440px;
                    justify-content: space-around;
                    margin: 20px 0px 0px 20px;
                }

                .container {
                  display: block;
                  position: relative;
                  padding-left: 50px;
                  padding-right: -50px;
                  margin-bottom: 18px;
                  cursor: pointer;
                  font-size: 22px;
                  -webkit-user-select: none;
```

```
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
    margin: 20px -20px 20px 20px;
  }

  .container input {
    position: absolute;
    opacity: 0;
    cursor: pointer;
    height: 0;
    width: 0;
  }

  .checkmark {
    position: absolute;
    top: 0;
    left: 0;
    height: 25px;
    width: 25px;
    background-color: grey;

  }

  .container:hover input ~ .checkmark {
    background-color: #ccc;
  }

  .container input:checked ~ .checkmark {
    background-color: green;
  }

  .checkmark:after {
    content: "";
    position: absolute;
    display: none;
  }

  .container input:checked ~ .checkmark:after {
    display: block;
  }

  .container .checkmark:after {
    left: 9px;
    top: 5px;
    width: 5px;
    height: 10px;
    border: solid white;
```

```
    border-width: 0 3px 3px 0;
    -webkit-transform: rotate(45deg);
    -ms-transform: rotate(45deg);
    transform: rotate(45deg);
  }

  .container2 {
      display: block;
      position: relative;
      padding-left: 65px;
      margin-bottom: -18px;
      cursor: pointer;
      font-size: 22px;
      -webkit-user-select: none;
      -moz-user-select: none;
      -ms-user-select: none;
      user-select: none;
      margin: 20px -20px 20px 20px;
    }

    .container2 input {
      position: absolute;
      opacity: 0;
      cursor: pointer;
      height: 0;
      width: 0;
    }

    .checkmark2 {
      position: absolute;
      top: 0;
      left: 0;
      height: 25px;
      width: 25px;
      background-color: grey;

    }

    .container2:hover input ~ .checkmark2 {
      background-color: #ccc;
    }

    .container2 input:checked ~ .checkmark2 {
      background-color: red;
    }

    .checkmark2:after {
      content: "";
```

```
                position: absolute;
                display: none;
              }

              .container2 input:checked ~ .checkmark2:after {
                display: block;
              }
              .container2 .checkmark2:after {
                left: 9px;
                top: 5px;
                width: 5px;
                height: 10px;
                border: solid white;
                border-width: 0 3px 3px 0;
                -webkit-transform: rotate(45deg);
                -ms-transform: rotate(45deg);
                transform: rotate(45deg);
              }

            .jumbotron {
                height: 100px;
                padding-top: 15px !important;
                padding-bottom: 105px !important;
            }
            .achievements-wrapper {
                height: 310px;
                overflow: auto;
            }

             th {
                position: sticky;
                top: 0px;
                background: black;
              }
              button {
                height: 30px;
                padding-top: 15px !important;
                padding-bottom: 55px !important;
            }
          .topnav {
          background-color: #333;
          overflow: hidden;
          }
          .topnav a {
          float: left;
          color: #f2f2f2;
          text-align: center;
          padding: 24px 5%;
```

```
              text-decoration: none;
              font-size: 17px;
              }

              .topnav a.active {
              background-color: #04AA6D;
              color: white;
              }
                </style>
            </head>
            <body>
            <div class="topnav">
                    <a href="/" onclick='showHome();' style="width: 100%;
background-color: black;"><h1>&#9200 RealTime Quiz</h1></a></div>
            <div id="showincorrect" class="container" style="display:none;">
            ${j}
            <form action="/user/profile" method="GET">
                <button type="submit" class="btn btn-primary btn-lg btn-block"
style="margin-bottom: 180px;">Back To Profile</button>
                </form>
            </div>
            <div id="takesurvey" class="row justify-content-center"
style="display: none;text-align: center;">
            <div>
            <h2 style="text-align: center;">Questionnaire</h2></div><br>
            <form action="/survey" method="POST">
            <h6 id="question1" style="text-align: center;">How often have you
played the quiz?</h6>
            <input type="radio" id="howoften" name="howoften" value="very
often" required>
            <label for="howoften">Very Often</label>
            <input type="radio" id="howoften" name="howoften" value="often">
            <label for="howoften">Often</label>
            <input type="radio" id="howoften" name="howoften"
value="occasionally">
            <label for="howoften">Occasionally</label>
            <input type="radio" id="howoften" name="howoften" value="rarely">
            <label for="howoften">Rarely</label><br><br>
            <h6 id="question2" style="text-align: center;">Have you used all
the app's features?</h6>
            <input type="radio" id="usedfeatures" name="usedfeatures"
value="yes" required>
            <label for="usedfeatures">Yes</label>
            <input type="radio" id="usedfeatures" name="usedfeatures"
value="no">
            <label for="usedfeatures">No</label><br><br>
            <h6 id="question3" style="text-align: center;">Do you think this
app will improve memory?</h6>
```

```
            <input type="radio" id="willimprove" name="willimprove"
value="yes" required>
            <label for="willimprove">Yes</label>
            <input type="radio" id="willimprove" name="willimprove"
value="no">
            <label for="willimprove">No</label><br><br>
            <h6 id="question4" style="text-align: center;">How old are
you?</h6>
            <input type="radio" id="howold" name="howold" value="18 to 40"
required>
            <label for="howold">18 to 40</label>
            <input type="radio" id="howold" name="howold" value="40 to 65">
            <label for="howold">40 to 65</label>
            <input type="radio" id="howold" name="howold" value="over 65">
            <label for="howold">Over 65</label><br><br>
            <h6 id="question5" style="text-align: center;">Is the app easy to
use?</h6>
            <input type="radio" id="easyuse" name="easyuse" value="yes"
required>
            <label for="easyuse">Yes</label>
            <input type="radio" id="easyuse" name="easyuse" value="no">
            <label for="easyuse">No</label><br><br>
            <h6 id="question6" style="text-align: center;">Is logging in easy
to do?</h6>
            <input type="radio" id="logeasy" name="logeasy" value="yes"
required>
            <label for="logeasy">Yes</label>
            <input type="radio" id="logeasy" name="logeasy" value="no">
            <label for="logeasy">No</label><br><br>
            <h6 id="question7" style="text-align: center;">Have you noticed
any issues like time-lag?</h6>
            <input type="radio" id="timelag" name="timelag" value="yes"
required>
            <label for="timelag">Yes</label>
            <input type="radio" id="timelag" name="timelag" value="no">
            <label for="timelag">No</label><br><br>
            <label for="explain"><h6 id="question8">How could the app be
improved?</h6></label><br>
            <textarea id="explain" name="explain" rows="3" cols="40" required>
            </textarea><br><br>
            <button type="submit" class="btn btn-dark btn-block" style="font-
size: 18pt; width:60%;margin-left:20%;">Submit Survey</button><br>
          </form>
            </div><br>

            <div id="showscore" class="row justify-content-center">
                <h3>Your Score is ${counts} out of 20</h3>
            </div><br>
```

```
            <div id="seebreakdown" class="row justify-content-center">
                <button class="btn btn-dark btn-block" onclick="incorrect()"
style="font-size: 18pt; width:260px;">See Breakdown</button>
            </div><br>

            <div id="backprofile" class="row justify-content-center">
            <form action="/user/profile" method="GET">
                <button type="submit" class="btn btn-dark btn-block"
style="font-size: 18pt; width:260px;">Back to Profile</button>
            </form>
            </div><br>

            <div id="showsurvey" class="row justify-content-center">
            <div class="text-center">
                <button type="submit" class="btn btn-dark btn-block"
onclick="survey();" style="font-size: 18pt; width:260px;">
                Complete Survey</button>
            </div>
            </div>
            <script>
            function survey(){
                document.getElementById("showscore").style.display = "none";
                document.getElementById("seebreakdown").style.display =
"none";
                document.getElementById("backprofile").style.display = "none";
                document.getElementById("showsurvey").style.display = "none";
                document.getElementById("takesurvey").style.display = "block";
            }
            function incorrect(){
                document.getElementById("showscore").style.display = "none";
                document.getElementById("seebreakdown").style.display =
"none";
                document.getElementById("backprofile").style.display = "none";
                document.getElementById("showsurvey").style.display = "none";
                document.getElementById("takesurvey").style.display = "none";
                document.getElementById("showincorrect").style.display =
"block";
            }
            </script>
            </body>
            </html>`);
        });
    } catch(error){
        console.log(error);
    }
});
```

```
app.post('/survey', async (req, res)=>{

    if (2 > 1){
        var transporter = nodemailer.createTransport({
            service: 'outlook',
            auth: {
              user: 'real_time_quiz@outlook.com',
              pass: 'Realtimequiz'
            }
          });
          var mailOptions = {
            from: 'real_time_quiz@outlook.com',
            to: `cy1yates@gmail.com`,
            subject: 'RealTime Quiz Survey',
            html: `
            <table style="width:100%; border: 1px solid black;">
            <tr>
              <td style="border: 1px solid black; width: 50%;">How often have
you played the quiz?</td>
              <td style="border: 1px solid black;width: 50%;text-align:
center;">${req.body.howoften}</td>
            </tr>
            <tr>
              <td style="border: 1px solid black; width: 50%;">Have you used
all the app's features?</td>
              <td style="border: 1px solid black;width: 50%;text-align:
center;">${req.body.usedfeatures}</td>
            </tr>
            <tr>
              <td style="border: 1px solid black; width: 50%;">Do you think
this app will improve memory?</td>
              <td style="border: 1px solid black;width: 50%;text-align:
center;">${req.body.willimprove}</td>
            </tr>
            <tr>
              <td style="border: 1px solid black; width: 50%;">How old are
you?</td>
              <td style="border: 1px solid black;width: 50%;text-align:
center;">${req.body.howold}</td>
            </tr>
            <tr>
              <td style="border: 1px solid black; width: 50%;">Is the app easy
to use?</td>
              <td style="border: 1px solid black;width: 50%;text-align:
center;">${req.body.easyuse}</td>
            </tr>
            <tr>
```

```
              <td style="border: 1px solid black; width: 50%;">Is logging in
easy to do?</td>
              <td style="border: 1px solid black;width: 50%;text-align:
center;">${req.body.logeasy}</td>
            </tr>
            <tr>
              <td style="border: 1px solid black; width: 50%;">Have you
noticed any issues like time-lag?</td>
              <td style="border: 1px solid black;width: 50%;text-align:
center;">${req.body.timelag}</td>
            </tr>
            <tr>
              <td style="border: 1px solid black; width: 50%;">How could the
app be improved?</td>
              <td style="border: 1px solid black;width: 50%;text-align:
center;">${req.body.explain}</td>
            </tr>
          </table>
            `
          };
          transporter.sendMail(mailOptions, function(error, info){
            if (error) {
              console.log(error);
            }
          });
    }
    res.redirect('/user/profile');
})

server.listen(process.env.PORT || 8080);
console.log(`Server listening on Port 8080 `);
```

User Surveys

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 04 August 2021 22:38
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	often
Have you used all the app's features?	yes
Do you think this app will improve memory?	yes
How old are you?	over 65
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	no
How could the app be improved?	

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 04 August 2021 17:16
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	often
Have you used all the app's features?	yes
Do you think this app will improve memory?	yes
How old are you?	over 65
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	yes
How could the app be improved?	The home page takes a while to load. The time limit is a bit quick and could be longer. Also maybe other memory games could be included

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 04 August 2021 00:07
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	occasionally
Have you used all the app's features?	yes
Do you think this app will improve memory?	yes
How old are you?	40 to 65
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	yes
How could the app be improved?	The home page takes a while to load. Also the timer of 5 seconds is too short.

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 01 August 2021 22:45
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	very often
Have you used all the app's features?	yes
Do you think this app will improve memory?	yes
How old are you?	18 to 40
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	yes
How could the app be improved?	

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 31 July 2021 22:15
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	occasionally
Have you used all the app's features?	yes
Do you think this app will improve memory?	yes
How old are you?	over 65
Is the app easy to use?	yes
Is logging in easy to do?	no
Have you noticed any issues like time-lag?	no
How could the app be improved?	If you would include timers on the questions also.

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 31 July 2021 18:36
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	rarely
Have you used all the app's features?	no
Do you think this app will improve memory?	yes
How old are you?	18 to 40
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	no
How could the app be improved?	

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 31 July 2021 18:23
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	occasionally
Have you used all the app's features?	no
Do you think this app will improve memory?	yes
How old are you?	over 65
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	no
How could the app be improved?	Time limits could be higher

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 31 July 2021 16:06
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	occasionally
Have you used all the app's features?	no
Do you think this app will improve memory?	yes
How old are you?	18 to 40
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	no
How could the app be improved?	More options for question difficulty and maybe some quiz categories.

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 31 July 2021 13:08
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	occasionally
Have you used all the app's features?	yes
Do you think this app will improve memory?	yes
How old are you?	over 65
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	no
How could the app be improved?	There could be more games other than just quizzes

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 30 July 2021 17:58
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	occasionally
Have you used all the app's features?	no
Do you think this app will improve memory?	yes
How old are you?	40 to 65
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	no
How could the app be improved?	It does not have any way to choose a difficulty level, or to have a timer with more time.

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 30 July 2021 17:30
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	often
Have you used all the app's features?	yes
Do you think this app will improve memory?	yes
How old are you?	18 to 40
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	no
How could the app be improved?	The timers are quite fast, and the leaderboard could have a search tool

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 30 July 2021 16:05
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	occasionally
Have you used all the app's features?	no
Do you think this app will improve memory?	yes
How old are you?	over 65
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	no
How could the app be improved?	There could be categories for the types of questions

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 30 July 2021 14:31
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	often
Have you used all the app's features?	yes
Do you think this app will improve memory?	yes
How old are you?	40 to 65
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	no
How could the app be improved?	

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 29 July 2021 07:45
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	often
Have you used all the app's features?	no
Do you think this app will improve memory?	yes
How old are you?	18 to 40
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	no
How could the app be improved?	

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 28 July 2021 12:12
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	very often
Have you used all the app's features?	yes
Do you think this app will improve memory?	yes
How old are you?	40 to 65
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	yes
How could the app be improved?	The timers could be a little slower

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 28 July 2021 07:35
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	rarely
Have you used all the app's features?	no
Do you think this app will improve memory?	yes
How old are you?	18 to 40
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	no
How could the app be improved?	

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 27 July 2021 19:41
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	very often
Have you used all the app's features?	yes
Do you think this app will improve memory?	yes
How old are you?	40 to 65
Is the app easy to use?	yes
Is logging in easy to do?	yes
Have you noticed any issues like time-lag?	no
How could the app be improved?	Its really great

From: real_time_quiz@outlook.com <real_time_quiz@outlook.com>
Sent: 24 July 2021 21:37
To: cy1yates@gmail.com
Subject: RealTime Quiz Survey

How often have you played the quiz?	very often
Have you used all the app's features?	yes
Do you think this app will improve memory?	yes
How old are you?	18 to 40
Is the app easy to use?	yes
Is logging in easy to do?	no
Have you noticed any issues like time-lag?	yes
How could the app be improved?	Good

Project Report

GRADEMARK REPORT

FINAL GRADE

73 /100

GENERAL COMMENTS

Instructor

1s marker:

All aspects of this project were carried out effectively and with good attention to detail. The project developed a web app implementing a real-time quiz with web harvesting elements. A detailed and appropriately comprehensive design was developed with particular emphasis on visual features. The implementation is competent with a complete set of functionalities delivered with immediate practical use in the application domain. Extensive testing was conducted including with a variety of end users. High quality professional presentation throughout the report with good signposting. Future extensions could be considered and explored in the discussions section.

2nd marker:

This is a very thorough and high-quality MSc IT project. The report is well-written. There are some functions that could be added to the system (e.g., the system could store the history of learning and, in each new session, suggest questions induced by the previous session). The validation part is very good, although a little more could have been said about the scalability (what happens if the number of users/questions increases).

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

PAGE 62

PAGE 63

PAGE 64

PAGE 65

PAGE 66

PAGE 67

PAGE 68

PAGE 69

PAGE 70

PAGE 71

PAGE 72

PAGE 73

PAGE 74

PAGE 75

PAGE 76

PAGE 77

PAGE 78

PAGE 79

PAGE 80

PAGE 81

PAGE 82

PAGE 83

PAGE 84

PAGE 85

PAGE 86

PAGE 87

PAGE 88

PAGE 89

PAGE 90

PAGE 91

PAGE 92

PAGE 93

PAGE 94

PAGE 95

PAGE 96

PAGE 97

PAGE 98

PAGE 99

PAGE 100