

Skateboarding Progress Tracker

Ben Keppie

March 11, 2015

Contents

1 Analysis	7
1.1 Introduction	7
1.1.1 Client Identification	7
1.1.2 Define the current system	7
1.1.3 Describe the problems	8
1.1.4 Section appendix	8
1.2 Investigation	10
1.2.1 The current system	10
1.2.2 The proposed system	19
1.3 Objectives	28
1.3.1 General Objectives	28
1.3.2 Specific Objectives	29
1.3.3 Core Objectives	30
1.3.4 Other Objectives	30
1.4 ER Diagrams and Descriptions	31
1.4.1 ER Diagram	31
1.4.2 Entity Descriptions	31
1.5 Object Analysis	32
1.5.1 Object Listing	32
1.5.2 Relationship diagrams	33
1.5.3 Class definitions	33
1.6 Other Abstractions and Graphs	35
1.7 Constraints	35
1.7.1 Hardware	35
1.7.2 Software	36
1.7.3 Time	36
1.7.4 User Knowledge	36
1.7.5 Access restrictions	36
1.8 Limitations	37
1.8.1 Areas which will not be included in computerisation	37
1.8.2 Areas considered for future computerisation	37
1.9 Solutions	37
1.9.1 Alternative solutions	37

1.9.2	Justification of chosen solution	38
2	Design	40
2.1	Overall System Design	40
2.1.1	Short description of the main parts of the system	40
2.1.2	System flowcharts showing an overview of the complete system	46
2.2	User Interface Designs	54
2.3	Hardware Specification	60
2.4	Program Structure	61
2.4.1	Top-down design structure charts	61
2.4.2	Algorithms in pseudo-code for each data transformation process	64
2.4.3	Object Diagrams	65
2.4.4	Class Definitions	65
2.5	Prototyping	67
2.6	Definition of Data Requirements	72
2.6.1	Identification of all data input items	72
2.6.2	Identification of all data output items	73
2.6.3	Explanation of how data output items are generated	73
2.6.4	Data Dictionary	74
2.6.5	Identification of appropriate storage media	78
2.7	Database Design	79
2.7.1	Normalisation	79
2.7.2	SQL Queries	85
2.8	Security and Integrity of the System and Data	86
2.8.1	Security and Integrity of Data	86
2.8.2	System Security	87
2.9	Validation	87
2.10	Testing	89
2.10.1	Outline Plan	91
2.10.2	Detailed Plan	92
3	Testing	106
3.1	Test Plan	106
3.1.1	Original Outline Plan	108
3.1.2	Changes to Outline Plan	109
3.1.3	Original Detailed Plan	109
3.1.4	Retained Items From Detailed Plan	122
3.1.5	Changed Items From Detailed Plan	129
3.1.6	Removed Items From Detailed Plan	134
3.2	Test Data	139
3.2.1	Original Test Data	139
3.2.2	Changes to Test Data	139
3.3	Annotated Samples	139
3.3.1	Actual Results	139

3.3.2 Evidence	152
3.4 Evaluation	170
3.4.1 Approach to Testing	170
3.4.2 Problems Encountered	171
3.4.3 Strengths of Testing	171
3.4.4 Weaknesses of Testing	172
3.4.5 Reliability of Application	172
3.4.6 Robustness of Application	172
4 System Maintenance	173
4.1 Environment	173
4.1.1 Software	173
4.1.2 Usage Explanation	173
4.1.3 Features Used	175
4.2 System Overview	175
4.2.1 General User Interface	175
4.2.2 Profile Tab User Interface	176
4.2.3 Editing Profile Table Information	176
4.2.4 Tricks Tab User Interface	176
4.2.5 Editing Trick Table Information	176
4.2.6 Skateparks Tab User Interface	177
4.2.7 Editing Skatepark Table Information	177
4.2.8 Reviews Tab User Interface	177
4.2.9 Editing Review Table Information	177
4.2.10 Support Tab User Interface	177
4.2.11 Reporting a Bug	177
4.3 Code Structure	177
4.3.1 Main Window	178
4.3.2 Tabs	180
4.3.3 Menu Bar	180
4.3.4 Tool Bar	180
4.3.5 SQL Connections	181
4.3.6 Validation	181
4.3.7 Main	182
4.4 Variable Listing	182
4.5 System Evidence	186
4.5.1 User Interface	186
4.5.2 ER Diagram	194
4.5.3 Database Table Views	194
4.5.4 Database SQL	201
4.5.5 SQL Queries	204
4.6 Testing	204
4.7 Testing	208
4.7.1 Summary of Results	208
4.7.2 Known Issues	208
4.8 Code Explanations	209

4.8.1	Difficult Sections	209
4.8.2	Self-created Algorithms	215
4.9	Settings	215
4.10	Acknowledgements	215
4.11	Code Listing	216
4.11.1	Main Window	216
4.11.2	Main Tabbed Widget	219
4.11.3	Menu Bar	219
4.11.4	Profile Widget	222
4.11.5	Profile Picture	225
4.11.6	Profile Toolbar	225
4.11.7	Profile SQL Connections	227
4.11.8	Tricks Widget	229
4.11.9	Tricks Toolbar	236
4.11.10	Tricks SQL Connections	237
4.11.11	Skateparks Widget	238
4.11.12	Skateparks Map	243
4.11.13	Skateparks Toolbar	248
4.11.14	Skateparks SQL Connections	249
4.11.15	Reviews Widget	249
4.11.16	Reviews Toolbar	255
4.11.17	Reviews SQL Connections	256
4.11.18	Support Widget	257
4.11.19	CLI Menu	259
4.11.20	CLI Get Menu Option	260
4.11.21	CLI Database Table Menu	260
4.11.22	CLI Create Database	265
4.11.23	CLI Profile Edit Options	268
4.11.24	CLI Trick Edit Options	272
4.11.25	CLI Skatepark Edit Options	275
4.11.26	CLI Review Edit Options	276
4.11.27	CLI Make New Difficulty	279
4.11.28	CLI Make New Product	280
5	User Manual	282
5.1	Introduction	282
5.2	Installation	283
5.2.1	Prerequisite Installation	283
5.2.2	System Installation	283
5.2.3	Running the System	285
5.3	Tutorial	287
5.3.1	Introduction	287
5.3.2	Assumptions	287
5.3.3	Tutorial Questions	287
5.3.4	Saving	287
5.3.5	Limitations	287

5.4	Error Recovery	287
5.4.1	Error 1	287
5.4.2	Error 2	287
5.5	System Recovery	287
5.5.1	Backing-up Data	287
5.5.2	Restoring Data	287
6	Evaluation	288
6.1	Customer Requirements	288
6.1.1	Aesthetically pleasing, easy to navigate GUI.	288
6.1.2	Videos organised and filtering capabilities.	288
6.1.3	Correct and accurate mapping to the skate parks/spots. .	288
6.1.4	Correct directions from current location to skate park/ spot on the map.	288
6.1.5	Non-biased reviews.	289
6.1.6	Clear database with a list of tricks in.	289
6.1.7	Easy to filter through tricks known.	289
6.1.8	Display status bar messages at appropriate times to in- form the user of changes *NEW*	289
6.1.9	Allow for the user to contact the developer *NEW* . . .	289
6.1.10	Ensure that the profile picture can be changed easily *NEW*	289
6.1.11	Ensure that the profile name can be edited easily *NEW*	289
6.1.12	Ensure that the profile email can be edited easily *NEW*	289
6.1.13	Ensure that videos can be filtered by categories. e.g easy, medium, hard tricks.	289
6.1.14	Ensure that videos load correctly and are linked to the right video.	290
6.1.15	Ensure that videos are displayed at the correct size/reso- lution that the monitor of the computer is.	290
6.1.16	Ensure the database can add, edit and remove trick data (Name, description, image, completed status and tutorial link).	290
6.1.17	Ensure that the database is displayed correctly inside the application at all resolutions.	290
6.1.18	Ensure that the tricks are marked by how hard they are by a three way scale of: Easy, Medium or Hard.	290
6.1.19	Ensure a checkbox is by the side of a trick to represent whether the user has completed that trick or not.	290
6.1.20	Ensure there is a search bar for a specific trick name. . .	290
6.1.21	Ensure there are filters for tricks e.g Switch trick filters. .	291
6.1.22	Ensure that the map is accurate to current roads.	293
6.1.23	Ensure location of the user is not revealed to anyone else.	293
6.1.24	Ensure that the current location marker is accurate. . . .	293
6.1.25	Ensure that when giving directions to skate parks from your current location that the mapping route is correct and on viable roads.	293

6.1.26	Ensure that the program can mark skate park locations.	293
6.1.27	Ensure no biased reviews are posted to the app and that they're moderated before they are universally posted.	293
6.1.28	Ensure the program runs fast without lag when navigating between areas of the application.	293
6.2	Effectiveness	293
6.2.1	Objective Evaluation	293
6.3	Learnability	293
6.4	Usability	293
6.5	Maintainability	293
6.6	Suggestions for Improvement	293
6.7	End User Evidence	293
6.7.1	Questionnaires	293
6.7.2	Graphs	293
6.7.3	Written Statements	293

Chapter 1

Analysis

1.1 Introduction

1.1.1 Client Identification

My client is my brother, Stuart Keppie, he is a former computing student who is currently studying Biological Sciences at the University of East Anglia and takes a keen interest in the urban sport skateboarding. He has a Sony Vaio laptop that he takes with him everywhere and therefore has mini applications that aid him through daily life and wants an application that will be able to cater all of his skateboarding, social and shopping advice needs. He likes utilising technology and has requested a program so that his life can be made easier.

1.1.2 Define the current system

Currently there is no single system available to cater for Stuarts activities. To aid ones learning in skateboarding the majority of people watch YouTube videos, this is done for a veriety of reasons. One being the fact that you are able to see in slow motion all of the movements that the person is doing to perform the trick. This is extremely useful, especially for a biology student, as you can theoretically replicate these muscle movements to perform the desired trick. To keep a record of what tricks you can do the current system is a pad and pen. The reason it is useful to keep a note of all your tricks is so that you feel that you have accomplished something within the sport, showing your accomplishments to your friends and remembering what tricks you have to use in competitions or games of S.K.A.T.E. For skateboarders 'spots' are locations that are fun to skate and for people to find them you can google them. Some people have tried creating applications such as [www.skatespots](http://www.skatespots.com).

co.uk and www.extremesportsmap.com/uk/. For skateboard shopping advice one would have to research extensively the pros and cons of each product and then make a final descision based on what is the best product for the use. This can be extremely time consuming as all the reviews are not in the same place and therfore you have to not only read through all the reviews but navigate from different websites to get the best idea of what a skteboarders view is on that specific product.

1.1.3 Describe the problems

There is no uniformed program for the system, which in itself is the main problem. Having to use multiple systems to carry out tasks causes Stuart's laptop to waste power and ultimately battery. Due to multiple web pages needing to be opened at one time on top of navigating through the internet is not time efficient which ultimately will lead to more computer activity which would drain the battery of the laptop more quickly. This can be an issue as if you run out of battery at the skate park then you will have nowhere to charge the laptop. The current system isn't efficient in being able to easily access all the neccesary information. For example to find a place to go skate nearby and then to get inspiration of what tricks to do and then learn a trick you would need to have atleast 3 web pages open, two of which will heavily use the CPU power, thus draining the battery due to the video streaming and advertisements. This current method is very time consuming and is a waste of time. Using YouTube as a source of learning skateboarding tricks can be useful, but some of the videos aren't useful and therefore they can be a waste of time to watch. Baised reviews of products by people that are paid to give a good review is a big problem in this industry, and therefore people can make ill-informed decisions on which product to buy. This is due to companies paying people/automated review writers.

1.1.4 Section appendix

Analysis section interview

1. What is the current system used?

Google Maps is used for locating possible skate spots. YouTube is used for new trick learning. Have to manually google for items to purchase.

2. What problems does this system cause you?

Maps does not have a skatepark search feature, skate spots can generally only be found if their name is known or when using a different website. Some YouTube videos have location restrictions. Online skateboard reviews can have bias.

3. What data is being recorded to carry out your tasks with the current system?

Search inputs

4. What extra data do you need to store/not need to store?

Tricks completed will be a new variable for storage.

5. How frequently will you need to edit the data?

On a daily basis, whenever the software is accessed.

6. Will data be deleted/added frequently? If so, how often?

Stored data will probably be amended daily, or every few days.

7. What processes are performed by the current system?

Satellite view presentation, general location search feature, video streaming.

8. What processes would you like to see in the new system?

Specific skate spot searching, relevant filtering or categorisation of skateboard videos, unbiased reviews of products.

9. When should these new processes be used in the new system?

When searching for skate spots. Categorising videos in the help section.

10. Which processes should be manually completed?

When the user has to select the filtering options. Adding new tricks to the database.

11. What are the inputs/outputs to the current system?

Adding a skate spot.

12. Are there any new inputs/outputs needed for the new system?

Current location, trick names, trick description, product details.

13. Is the application purely computer based, or are hard copies of data needed?

Computer based.

14. What are your computer specifications (inc. Operating System)?

- Sony Vaio e15
- Microsoft Windows 7 Home Premium OS
- 500 GB HDD Memory
- 8GB RAM
- Intel Core i5 Quad Core Processor
- Intel HD3000 Graphics Card

15. Is security a problem?

Current location input shouldn't be let out without permission (privacy of whereabouts).

16. How should errors be reported in the new system?

GUI pop-up and error message sent to software developer.

17. Are there any constraints? (cost, time, data, software, hardware etc.)

The software needs to be time efficient, to maximise time available to spend on the activity the software aids.

18. How many people will be using the new system?

One user per system. One system initially, but if the software is good it will be recommended to other users for synchronisation.

19. If greater than one, what information should other users have about your account?

Progress level (how many tricks learnt etc.), skate spots visited.

20. What should the new system achieve?

Able to perform/navigate to all current tasks from one navigation menu. Not need separate programs for each task. Have social compatibility ie. Connectivity to peers.

21. Do you have a particular solution in mind to tackle any specific problems?

N/A

22. Is installing additional software an issue?

No.

23. Any extra notes?

N/A

24. How many hard coded tricks would you like in the database?

50 tricks in the database initially, and then allow for personal user additions.

1.2 Investigation

1.2.1 The current system

The current system is split into 4 sub systems. These systems are:

- YouTube - for learning tricks.
- Notepad - for tricks.
- Google maps and other websites - for finding skate parks and spots.
- googling reviews on the internet - for buying guidance.

Data sources and destinations

Some of these systems have multiple data sources and destinations and none of the systems overlap in data sources and destinations.

Data Source	Data	Data Example	Data Destination
User	Search keywords	How to kickflip	YouTube Servers
YouTube Servers	Server response with a list of videos relating to the search	How to kickflip tutorial video	User
User	Writing a tricks name that you have learnt	Kickflip	Notepad
Google Maps Server	Image of the location, coordinates, description	Image of Cambourne skatepark, 52.2200 N, 0.0700 W, Cambourne skatepark was established in 2002	user
User	Searching for a skateboard part review	Thunder skateboard truck reviews	Google Server
Google Server	Results of google search	5 star thunder review from Skate Blog	user

Algorithms

Algorithm 1 Algorithm to show deciding on a new trick to learn

```
1: Trick ← USERINPUT
2: IF Trick = True THEN
3:   OUTPUT "You can do this trick"
4:   OUTPUT "Write trick in note pad"
5: ELSE
6:   OUTPUT "You can't do this trick"
7: ENDIF
```

Algorithm 2 Deciding whether to search how to learn a trick

```
1: Trick ← USERINPUT
2: IF Trick = True THEN
3:   OUTPUT "Search for a YouTube video"
4: ELSE
5:   OUTPUT "Don't search for a YouTube video"
6: ENDIF
```

Algorithm 3 Algorithm for learning tricks

```
1: "Trick" ← USERINPUT
2: finished ← false
3:
4: WHILE notfinished
5:   OUTPUT Attempt trick
6:   IF Trick = False THEN
7:     OUTPUT "Try again"
8:   ELSE
9:     finished ← true
10:  ENDIF
11: ENDWHILE
12: OUTPUT "Trick completed"
```

Algorithm 4 Algorithm for watching videos

```

1: OUTPUT Open InternetBrowser
2: OUTPUT Load www.YouTube.com
3: Trick  $\leftarrow$  USERINPUT
4: OUTPUT Type Trick tutorial into YouTube Search Bar
5: OUTPUT Press the Enter key
6: OUTPUT Find appropriate tutorial link
7: OUTPUT Click the thumbnail
8: OUTPUT Watch the video

```

Algorithm 5 Finding Skate Spots

```

1: "Bored"  $\leftarrow$  USERINPUT
2: IF Bored = True THEN
3:   OUTPUT "Search for a skate spot"
4: ELSE
5:   OUTPUT "Don't search for a skate spot"
6: ENDIF

```

Algorithm 6 Finding Reviews and Deciding on a Purchase

```

1: finished  $\leftarrow$  false
2:
3: WHILE notfinished
4:   IF Skate part broken = True THEN
5:     OUTPUT "Search for a review"
6:     IF part_review = good THEN
7:       OUTPUT "Consider Purchasing"
8:       IF purchased = True THEN
9:         finished  $\leftarrow$  true
10:        ENDIF
11:      ELSE
12:        OUTPUT "Keep searching for a replacement part"
13:      ENDIF
14:    ENDIF
15:  ENDWHILE

```

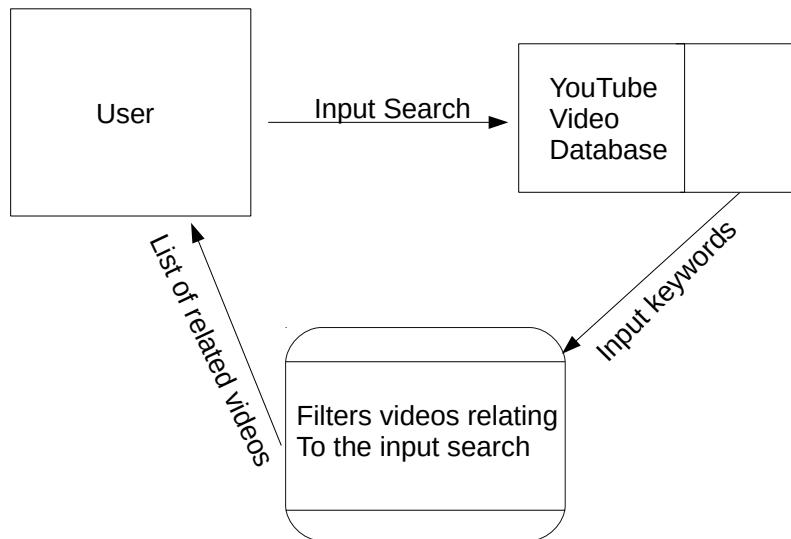
Data flow diagram

Figure 1.1: Data Flow Diagram of Searching for a YouTube Tutorial

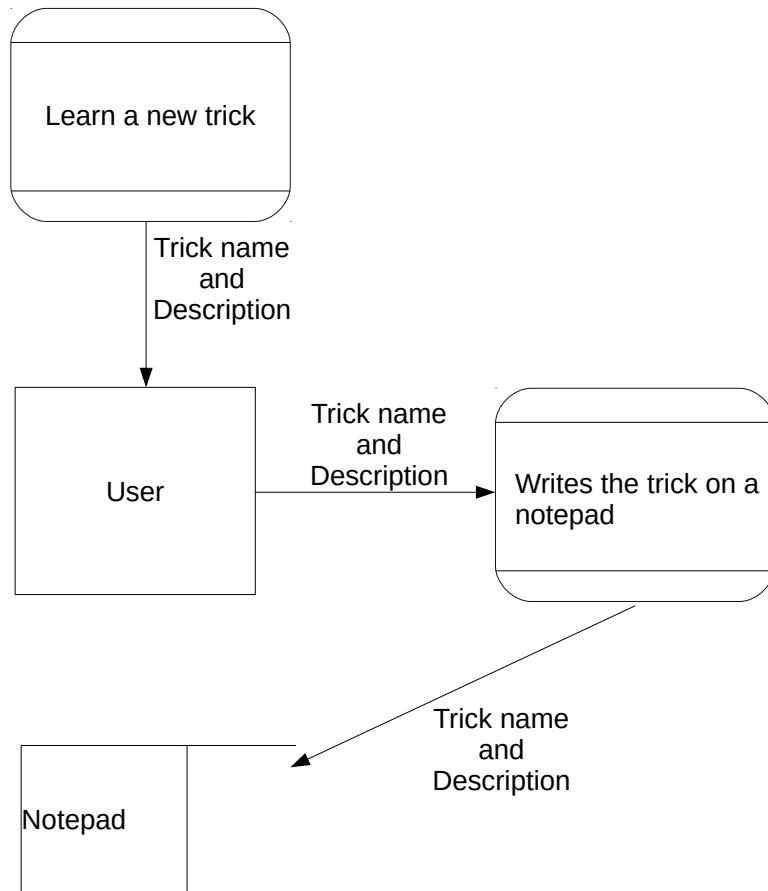


Figure 1.2: Data Flow Diagram of writing recently learnt tricks on a note pad

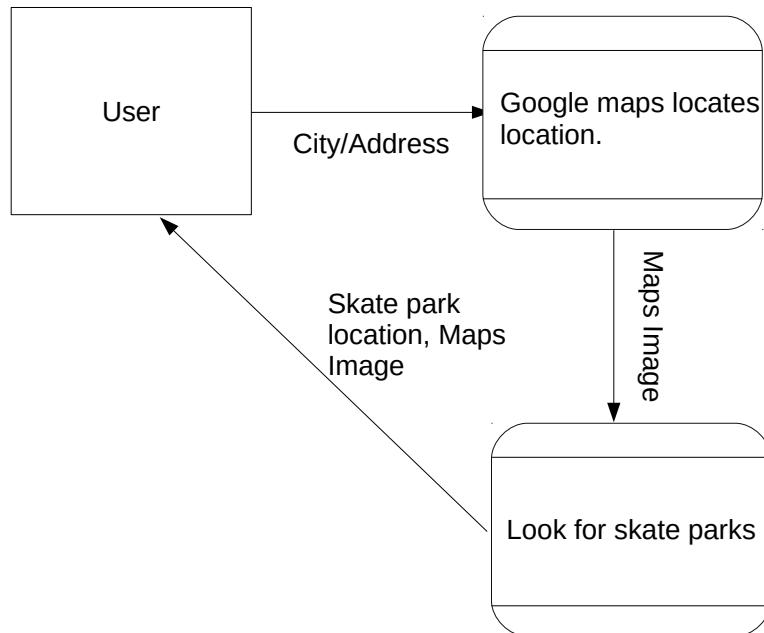


Figure 1.3: Data Flow Diagram of Searching for a skate park

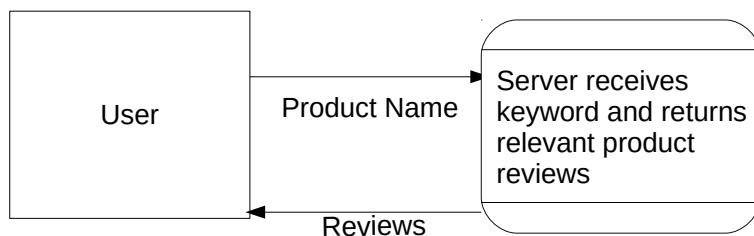


Figure 1.4: Data Flow Diagram of Searching for reviews of a product

Input Forms, Output Forms, Report Formats

The only input form in the current system is Stu's notepad which contains data about his tricks that he has learnt. I have taken a page from his notepad (see image below) of details about his time at Saffron Walden skate park on Friday the 26th of September. His input form contains data about the obstacles at the skate park, the tricks he learnt on that day and tricks that he saw and possibly wants to try and learn.

S Rate Notepad

Saffron Walden Skate Park: Fri, 26th September

Obstacles:

- 3x bowls
- Street Section
- Stairs
- Cylinder Rail

Tricks Learnt:

- Drop in 1st bowl
- Kickflip off the Street Sections pad
- Ollie the Stairs

Cool Tricks Seen by Others:

- Tre Flip
- Laser Flip

Figure 1.5: A page from Stuarts notepad

The only output forms in the current system would be the YouTube video links at redirect you to the YouTube video. A couple of these output links are listed below:

- How To Ollie Tutorial - https://www.youtube.com/watch?v=FuyYBWuV7VU&index=1&list=PLIZKb9hZiA_uFdK_zu9d_E_8gydHx5kwy
- How To Kickflip Tutorial - https://www.youtube.com/watch?v=_7fEsZG1xuI&index=2&list=PLIZKb9hZiA_uFdK_zu9d_E_8gydHx5kwy

1.2.2 The proposed system

Data sources and destinations

The new system keeps some of the same data sources and destinations as the current system. For example YouTube will still be the source of the tutorial videos and google maps will still be used as the basis for mapping. But all of the other data will be stored internally within the system to increase the ease of access.

Data Source	Data	Data Example	Data Destination
User	Searching for a skatepark name	Cambourne skatepark	Google Maps Servers
Google Maps Server	Image of the location, coordinates, description	Image of Cambourne skatepark, 52.2200 N, 0.0700 W, Cambourne skatepark was established in 2002	user
User	Trick	Kickflip	Trick Database
User	Trick Description	Board rotating 360 degrees on a horizontal axis	Trick Database
User	Trick Image	Kickflip.jpeg	Trick Database
User	Trick Tutorial Link	http://www.youtube.com/watch?v=1082h	Trick Database
Trick Database	Trick	Kickflip	User
Trick Database	Trick Description	Board rotating 360 degrees on a horizontal axis	User
Trick Database	Trick Image	Kickflip.jpeg	User
Trick Database	Trick Tutorial Link	http://www.youtube.com/watch?v=1082h	User
User	ProductName	Trucks	Review Database
User	Product Type	Trucks	Review Database
User	Product Size	5.0	Review Database
User	Product Brand	Thunder	Review Database
User	Product Review	Best Trucks I've owned	Review Database
User	Product Rating	1	Review Database
Review Database	Product Name	Spec ops	User
Review Database	Product Type	Trucks	User
Review Database	Product Size	5.0	User
Review Database	Product Brand	Thunder	User
Review Database	Product Review	Best Trucks I've owned	User
Review Database	Product Rating	1	User

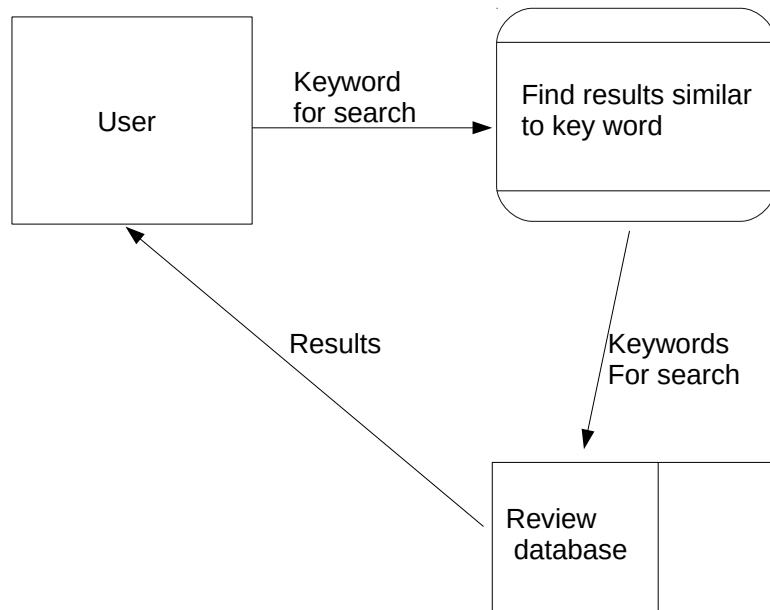
Data flow diagram

Figure 1.6: Data flow diagram for the new systems review search

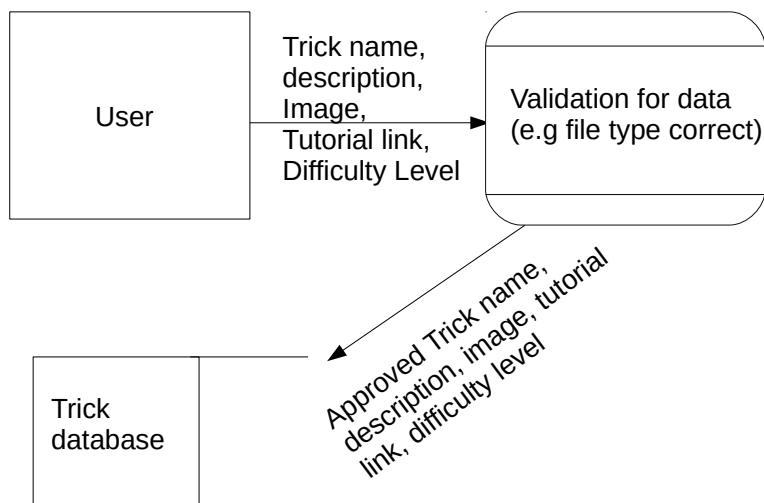


Figure 1.7: Data flow diagram for adding new tricks to the database

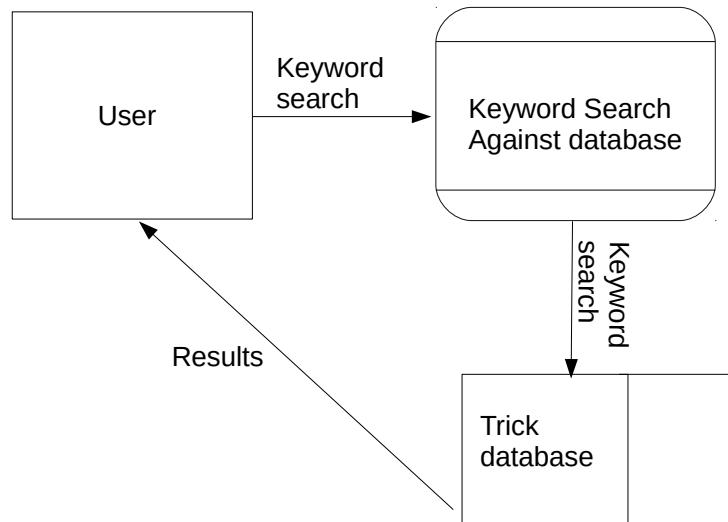


Figure 1.8: Data flow diagram for reading tricks from the database

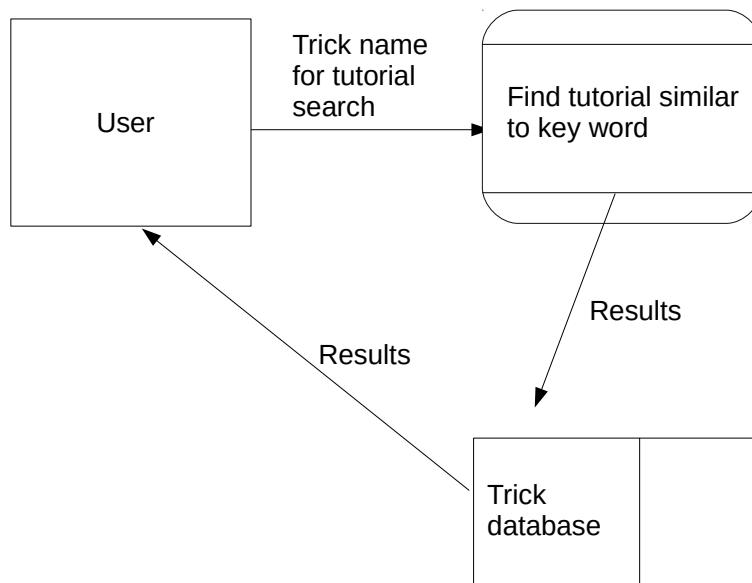


Figure 1.9: Data flow diagram for the new systems tutorial search

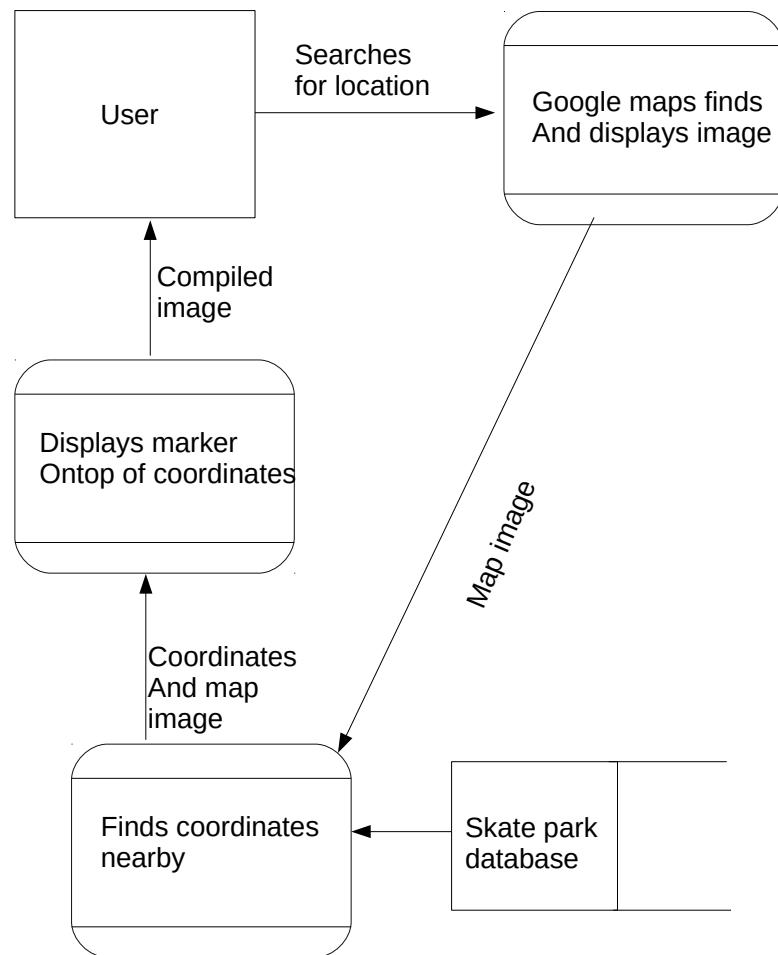


Figure 1.10: Data flow diagram for the new systems skate park search

Ben Keppie

Candidate No. 4609

Centre No. 22151

Data dictionary

Name	Data Type	Length	Validation	Example Data	Comment
TrickName	String	25 characters	None	Ollie	Linked to Description, image and tutorial link
TrickDescription	String	100 characters	None	Board is turned around 180 degrees	Linked to trick, image and tutorial link
TrickImage	Image	N/A	670 x 503	Ollie.jpeg	None
TrickTutorialLink	String	100 characters	Correct link	http://www.youtube.com/watch?v=3809	Linked to trick, description and image
TrickDifficulty	string	6 characters	easy, medium, hard	easy	colour coded
TrickCompleted	Boolean	True/False	None	True	None
SkateparkName	String	25 characters	Correct Name	Cambourne Skatepark	None
SkateparkCoordinates	Float	20 characters	Correct coordinates	52.2200 N, 0.0700 W	None
SkateparkDescription	String	200 characters	Accurate description	Halfpipe only	None
ProductBrand	String	20 characters	None	ZERO	Moderated
ProductType	String	20 characters	None	Deck	Moderated
ProductName	String	25 characters	None	Cosmic Tiger	Moderated
ProductSize	String	20 characters	None	7.875"	Moderated
ProductReview	String	500 characters	Non-biased	These trucks are the best I have owned	Moderated
ProductRating	integer	range 1-5	Non-biased	1	Moderated

Volumetrics

For the initial size of the proposed system I chose to add 50 standard skate boarding tricks as there are limitless tricks and the user is able to add tricks to his own individual database of tricks and my client requested it (See the section appendix question 24). The maximum length of a name for a skateboarding trick is 25 characters, this is because they range from words such as "shuv" to "triple dolphin late flip". With the initial program as there will be 30 standard skateboarding tricks the names of them alone would take up 750 bytes as a string takes up 1 byte per character. The tickbox next to the trick stating whether you have completed the trick or not would take a boolean value and therefore take up 60 bytes of storage as boolean values take up 2 bytes each. The description of a trick would approximately be 100 characters, for example the description of a kickflip would be:

- Flipping the board 360 ° along the axis that extends from the nose to the tail of the deck.

This will add a further 3000 bytes to the program. The location coordinates of the skatepark will have to be stored, and the skateparks and spots around Cambridge is roughly 20 and each skatepark will contain 2 integers (the coordinates) and as integers take up 4 bytes of storage each the stored coordinates will initially be 160 bytes. The maximum length of YouTube link would be 100 characters and as there are 50 tricks already implemented there will be 50 links, this ultimately adds up to a further 5000 bytes. images will be 670x503 which totals to 337010 bytes each and a total of 50 images will be needed which means in total 16850500 bytes if memory will be needed for images.

Adding up all of the bytes of data would be calculated by the sum:

$$750 + 60 + 3000 + 160 + 5000 + 16850500 = 16859470 \text{ Bytes}$$

To get this unit in KB you would divide the number of bytes by 1024 which equals 16464.3 KB (Rounded to 1 d.p)

To get this unit in MB you would divide by a further 1024 which equals 16.1 MB (Rounded to 1 d.p)

As this system will be ever expanding in the number of tricks that are added to the database the actual systems data size will be larger as time goes on.

1.3 Objectives

1.3.1 General Objectives

- Aesthetically pleasing, easy to navigate GUI.
- Videos organised and filtering capabilities.

- Correct and accurate mapping to the skate parks/spots.
- Correct directions from current location to skate park/ spot on the map.
- Non-biased reviews.
- Clear database with a list of tricks in.
- Easy to filter through tricks known.

1.3.2 Specific Objectives

- Ensure that videos can be filtered by categories. e.g easy, medium, hard tricks.
- Ensure that videos load correctly and are linked to the right video.
- Ensure that videos are displayed at the correct size/resolution that the monitor of the computer is.
- Ensure the database can add, edit and remove trick data (Name, description, image, completed status and tutorial link).
- Ensure that the database is displayed correctly inside the application at all resolutions.
- Ensure that the tricks are marked by how hard they are by a three way scale of: Easy, Medium or Hard.
- Ensure a checkbox is by the side of a trick to represent whether the user has completed that trick or not.
- Ensure there is a search bar for a specific trick name.
- Ensure there are filters for tricks e.g Switch trick filters.
- Ensure that the map is accurate to current roads.
- Ensure location of the user is not revealed to anyone else.
- Ensure that the current location marker is accurate.
- Ensure that when giving directions to skate parks from your current location that the mapping route is correct and on viable roads.
- Ensure that the program can mark skate park locations.
- Ensure no biased reviews are posted to the app and that they're moderated before they are universally posted.
- Ensure the program runs fast without lag when navigating between areas of the application.

1.3.3 Core Objectives

- Ensure that videos can be filtered by categories. e.g easy, medium, hard tricks.
- Ensure that videos load correctly and are linked to the right video.
- Ensure the database can add, edit and remove trick data (Name, description, image, completed status and tutorial link).
- Ensure that the tricks are marked by how hard they are by a three way scale of: Easy, Medium or Hard.
- Ensure a checkbox is by the side of a trick to represent whether the user has completed that trick or not.
- Ensure there is a search bar for a specific trick name.
- Ensure there are filters for tricks e.g Switch trick filters.
- Ensure that the program can mark skate park locations.
- Ensure the program runs fast without lag when navigating between areas of the application.

1.3.4 Other Objectives

- Ensure that videos are displayed at the correct size/resolution that the monitor of the computer is.
- Ensure that the database is displayed correctly inside the application at all resolutions.
- Ensure that the map is accurate to current roads.
- Ensure location of the user is not revealed to anyone else.
- Ensure that the current location marker is accurate.
- Ensure that when giving directions to skate parks from your current location that the mapping route is correct and on viable roads.
- Ensure no biased reviews are posted to the app and that they're moderated before they are universally posted.

1.4 ER Diagrams and Descriptions

1.4.1 ER Diagram

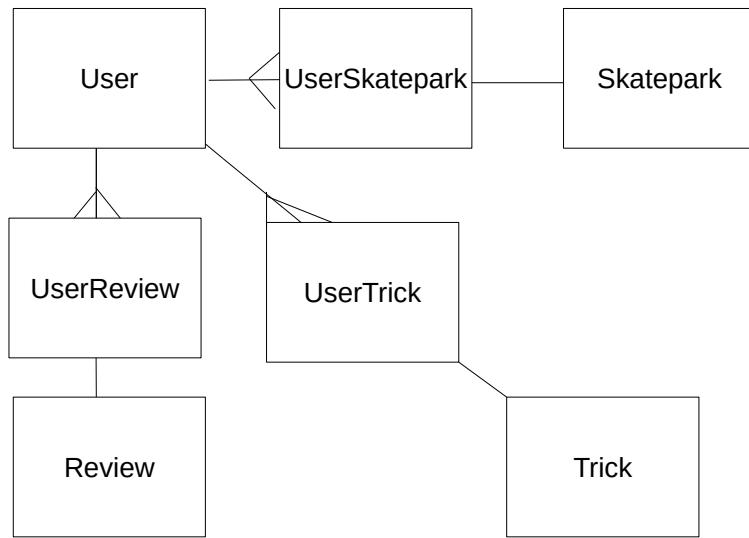


Figure 1.11: Entity-Relationship Diagram

1.4.2 Entity Descriptions

User(UserID, Username)

Trick(TrickName, UserID, Description, Difficulty, Completed, Image, TutorialLink)

Skatepark(SkateparkID, UserID, SkateParkName, Coordinates, Description)

Review(ReviewID, UserID, ReviewRating, ProductName, ProductType, ProductSize, ProductBrand, Review)

UserTrick(UserTrickID, UserID, Description, Difficulty, Completed, Image, TutorialLink)

UserSkatepark(UserSkateparkID, *UserID*, SkateParkName, Coordinates, Description)

UserReview(UserReviewID, *UserID*, ReviewRating, ProductName, ProductType, ProductSize, ProductBrand, Review)

1.5 Object Analysis

1.5.1 Object Listing

- User
- Trick
- SkatePark
- Review
- Product

1.5.2 Relationship diagrams

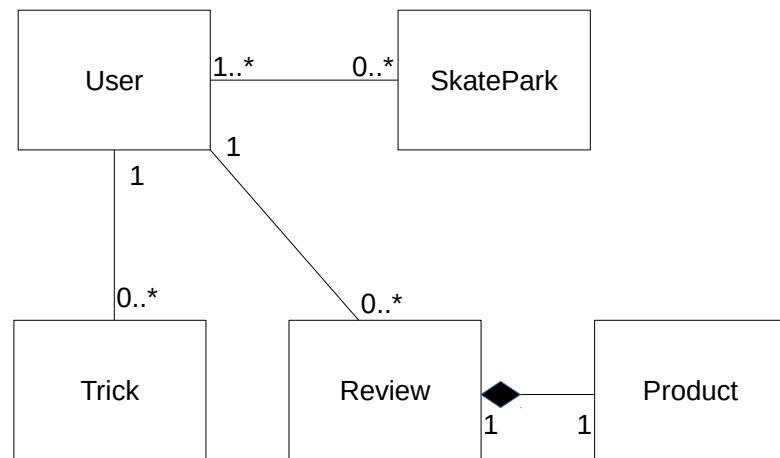


Figure 1.12: Relationship Diagram

1.5.3 Class definitions

User
UserID
Username
get_user_id
get_username

Trick
TrickName
TrickDescription
TrickDifficulty
TrickCompleted
TrickImage
TrickTutorialLink
get_trick_name
get_trick_difficulty
get_trick_state
get_trick_image
get_trick_tutorial_link
SkatePark
SkateParkID
SkateParkName
SkateparkCoordinates
SkateparkDescription
get_skatepark_id
get_skatepark_name
get_skatepark_coordinates
get_skatepark_description
Review
ReviewID
get_review_id
Product
ProductName
ProductSize
ProductBrand
ProductType
ProductReview
get_product_name
get_product_size
get_product_brand
get_product_type
get_product_review

1.6 Other Abstractions and Graphs

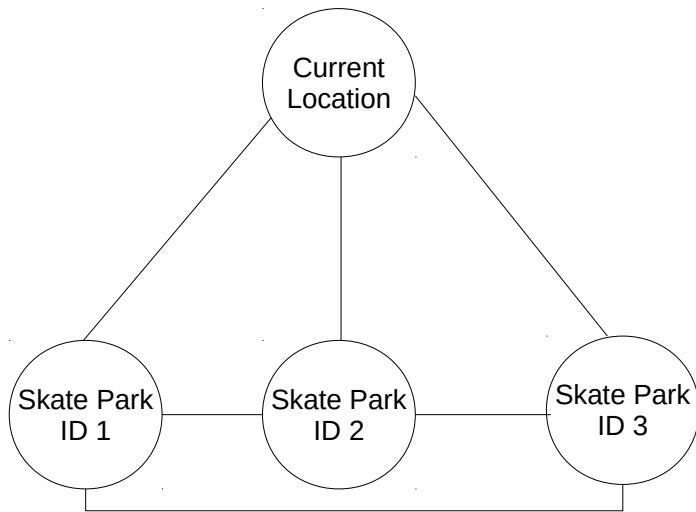


Figure 1.13: Graph to show how the user can map their current location to skateparks.

1.7 Constraints

1.7.1 Hardware

The new system will need to be able to run on Stuart's computer, the current specification of Stuart's laptop is:

- 15.6" HD 1366x768 Screen
- i5-2450M Dual Core Processor (Sandy Bridge) 2.5GHz (overclocked to 3.1GHz) 3MB Cache
- 8GB DDR3 RAM
- 500GB HDD Memory
- Intel HD3000 Graphics Card

Stuarts laptop has more than enough processing power in order to run the new system, this will allow Stuart to run several applications whilst also using the new system. Stuart will be able to take the application with him wherever he takes his laptop. Therefore portability of the application isn't limited as he works on a laptop.

The only constraints will be the screen resolution and battery of his laptop, therefore optimising the program for his specific resolution will be a task to overcome and making the program universal for other users to run the program.

1.7.2 Software

Stuarts laptop is currently running on Windows 7 Home Premium, he does not want to have to install a virtual machine to run the application; however he does not mind installing external software to aid the systems running capability and ease. The new system however was initially thought to run on Windows 7 Home Premium and therefore this is not an issue. Stuart being willing to install extra programs gives some extra flexibility with how to overcome other problems if they arise. Therefore currently there are no foreseeable constraints regarding software.

1.7.3 Time

Currently the only time restriction is the project deadline, this is Friday 13th February 2015 for the implementation section of the coursework, this was set by my teacher. Stuart is extremely flexible with time and isn't under any time constraints as long as the final product works.

1.7.4 User Knowledge

As an ex-computing student and a person regularly around computers, Stuart's knowledge about computers and the way that they work is good and therefore he is capable of understanding and explaining complex processes. This means that in the designing process of the new system I will be able to plan for keyboard shortcuts and more complex features which will essentially .

1.7.5 Access restrictions

Restricting the data about the individual's location is an important feature of the application, other than that the information that is inputted into the system is general and therefore it does not need its access restricted. This lack of access restriction is in place as other users should be able to benefit from the skate parks and spots that another user has found. Due to the system not

containing private information about a living individual, the new system will have no problems with any current legislation or law.

1.8 Limitations

1.8.1 Areas which will not be included in computerisation

The process of learning tricks will still have to be done physically as it cannot be completed any other way, the same goes for seeing other people's tricks and being inspired to learn a new trick. Apart from that, all of the information will be stored in the system electronically and processed by the computer.

1.8.2 Areas considered for future computerisation

- A forum for users of the app to discuss problems, help each other and recommend products.
- Phone app.

1.9 Solutions

1.9.1 Alternative solutions

Solution	Advantages	Disadvantages
Web-Based Application	<ul style="list-style-type: none"> • Can access anywhere with an internet connection/LAN connection. • No installation required. • Nice formatting and extremely versatile. • Can use a lot of different programming languages to accomplish a task (HTML, CSS, JavaScript, PHP etc.) 	<ul style="list-style-type: none"> • Lack of experience in web based programming. • More complex security for hiding locations. • Web Hosting costs money. • More extensive knowledge to fix problems.

	<ul style="list-style-type: none"> Making the current system more efficient 	<ul style="list-style-type: none"> No need for the client to learn anything new. No expense. 	<ul style="list-style-type: none"> Problems with the current system will still exist. The system won't be as efficient.
Command-Line Application		<ul style="list-style-type: none"> Runs extremely fast. Uses minimal system resources. 	<ul style="list-style-type: none"> Client will need to learn code. Security would be hard to keep as there are 'hidden' codes. No GUI. Coding error could break the computer.
SQL Database		<ul style="list-style-type: none"> Runs complex queries fast. Could store all the information in a compressed form. Information easy to access. 	<ul style="list-style-type: none"> No GUI. Client would have to learn SQL code. I am not very good at SQL code. Debugging would be difficult.
Python Application with GUI (PyQt4)		<ul style="list-style-type: none"> Python is my primary programming language. Easy to use for the client. Nice, clean GUI. Versatile GUI and Python allows for the program to work on all types of systems. Easy to format data. 	<ul style="list-style-type: none"> Uses system resources more than other solutions. Programming can be complex (GUI is harder than command-line).

1.9.2 Justification of chosen solution

I have chosen to complete the new system using a Python application with a PyQt Graphical User Interface. I have chosen to create the system in this way as python is a programming language that is extremely versatile and will be able to carry out all of the tasks whilst supplying the client with a smooth and efficient experience. A web-based solution would not be appropriate as neither

the client nor I am willing to pay to set up a server and also I do not have extensive knowledge and experience with working with programming languages such as: HTML, CSS, JavaScript and PHP. Making the current system more suitable would not be suitable either as the main problems with the current system would still appear in the new system, the client and I do not see this to be a worthy way to tackle this system. A command-line application would be too complicated for everyday use due to its steep learning curve and additionally the security threats are too high. Finally an SQL database doesn't provide a clean GUI like the python solution does and isn't as versatile in the ways that you can input and read data, which leaves me to believe that using the Python application is the most suitable for undertaking the new system. Python also contains extensive forums with information to help the client add any additional features he wants when my project is completed whilst also aiding me to find the best way to tackle the problems that I will encounter when programming the new system.

Chapter 2

Design

2.1 Overall System Design

2.1.1 Short description of the main parts of the system

Start-Up Wizard

- General User Interface
- Adding a Profile

General User Interface for the Start-Up Wizard

The general user interface for the start-up wizard will consist of a paragraph of text, containing information on how to proceed with setting up a profile and 3 text boxes to enter your name and email. An additional 'browse' button is available to select a profile picture. A save button at the bottom of the window is there to save all the changes.

Adding a Profile

The start-up wizard appears if no profile information can be found in the database. The start-up wizard allows you to add your name and email and to select a profile picture. Once all the information has been filled in the changes will be saved and the actual application will load up, personalised with the information that you have entered in the start-up wizard.

Profile

- General User Interface
- Editing Profile Information

General User Interface for the Profile

The general user interface for the profile will consist of a picture, name, email and recent completed tricks, above the main window there will be tabs containing the other areas of the system and above that will be an option to edit your profile. Additionally a progress bar showing the percentage of tricks completed will be displayed at the bottom of the window.

Editing Profile Information

Once the 'Edit profile' button is clicked on the menu bar a drop down appears with the options:

- Change Profile Picture
- Change Name
- Change Email

Once the 'Change Profile Picture' button is pressed you will be redirected to browse your documents for a picture, the picture will be resized to 160x160 pixels. When the 'Change Name' button is pressed a pop-out dialogue box will appear with the opportunity to change your name and a 'save' button below that to save your new name. When the 'Change Email' button is pressed a pop-out dialogue box will appear and present you with a text box to enter a new email, this is validated to ensure the email is correct, A 'save' button is displayed below and when clicked it will save your new email.

Trick Table

- General User Interface
- Adding a Trick
- Deleting a Trick
- Editing a Trick
- Completing a Trick
- Progress Tracker

General User Interface for the Trick Table

The general user interface for the trick table will consist of a table in the middle of the application, search filters will be placed on the side of the application and options to add a trick at the top of the application. By the side of each trick there a choice to delete or edit existing tricks. The columns of the table will consist of:

- Trick Creator (The Trick Creator contain the first and last name of the user who added the trick to the database)
- Trick Name (The Trick Name contains the name of the skateboard trick)
- Trick Description (The Trick Description will contain a short description of the trick)

- Trick Obstacle (The Trick obstacle will say if a specific obstacle is needed for the trick)
- Trick Image (The Trick Image will be contain a 670x503 pixel image of the trick)
- Trick Tutorial (The Trick Tutorial will contain a YouTube tutorial link to the trick)
- Trick Difficulty (The Trick Difficulty will contain either: Easy, Medium or hard depending on how difficult the trick is)
- Trick Completed (The Trick Copleted will contain a tick box along with the date that the tick box became ticked)

Below the option to add a new trick will be tabs containing other areas of the system.

Adding a Trick to the Trick Table

When the addition button (+) is pressed, a pop out will appear. This will automatically fill in the Trick Creator's name (first name and last name). Whilst the rest of the information will be readily available to edit. For example, Trick Name, Description, Obstacle and Tutorial will all have a text box to fill in freely, whilst the trick image will have an 'upload' button where you will be able to search your computer for an image which will automatically be re-sized to 670x503 pixels. The Trick Difficulty will be selected via a drop box with the three options: Easy, Medium and Hard and the Trick Completed will be a tick box. The Trick Tutorial text box will be checked for a correct youtube link. Once all the information has been added the trick will be added to the database and the trick will be able to be seen inside the table when on the Trick Database page.

Deleting a Trick from the Trick Table

By the side of every trick in the Trick Table there will be a 'bin' icon which gives you the option to delete a trick from your table. Once this is clicked a confirmation will pop up to ensure that you want to permanently delete that trick. Once that trick is deleted it wil be removed from the database and you will no longer be able to view it in your table of tricks.

Editing a Trick in the Trick Table

By the side of every trick in the Trick Table there will be a 'pencil' icon which gives you the option to edit a trick in your table. Once this is clicked a pop up identical to the one that you are given when you click on the (+) button comes up; however all of the information is already filled in with the information from that trick. From this pop up you can edit that specific tricks information, just as you would if you were adding a trick.

Completing a Trick in the Trick Table

Once the user has ticked a trick to its completed state, the tick box will display a tick and below it will have the date that the trick has been ticked. This date will be generated via the computers date.

Progress Tracker

At the bottom of the application a bar containing the status of the user's progress is displayed. This will contain information of how many tricks you have completed out of the tricks in the trick table.

Skatepark Map Marker

- General User Interface
- Adding a Skatepark
- Deleting a Skatepark
- Editing a Skatepark
- Mapping From Location to a Skatepark

General User Interface for the Skatepark Map Marker

The general interface for the Skatepark Map Marker is a Google maps image with markers locating skateparks and skate spots around the UK. Below the Google maps graph will be two text boxes where you will be able to type in two locations and a 'Map Journey' button to the right of both boxes. When a marker on the map is clicked on information about that skatepark is given in a dialogue box. Also in the dialogue box will be two options to edit and delete the skatepark, the symbols for these are a pencil and a bin, respectively. Above the graph will be an option to add a skatepark, shown by an addition sign. Below the option to add a new Skatepark will be tabs containing other areas of the system.

Adding a Skatepark

In the top menu bar of the Skatepark Map Marker window there will be an addition symbol (+), identical to that of the one in the Trick Table window with the functionality of adding a skatepark to the map. Once the symbol is pressed the user will be prompted with a pop-up which contains 3 text boxes and a confirm button. The three text boxes will allow the user to add the Name, Coordinates and Description of the skatepark that they are adding. The coordinates are validated by being in the correct format. The Name and Description are freely entered by the user. Once the confirm button is pressed the information for the skatepark is stored and a marker is placed on the map.

Deleting a Skatepark

When the bin symbol is pressed inside the marker dialogue box a pop-up will be displayed asking the user if they want to permanently delete that skatepark. Once the skatepark is deleted you will no longer be able to view the marker or information on the map.

Editing a Skatepark

Once the pencil button is clicked a pop up identical to the one that you are given when you click on the (+) button comes up; however all of the information is already filled in with the information from that skatepark. From this pop up you can edit that specific skateparks information, just as you would if you were adding a skatepark.

Mapping From a Location to a Skatepark

Below the map there are two text boxes where you can enter two addresses and then click on the 'Map Route' button to the right of both of these which will then show the route on the Google maps image above.

Review Window

- General User Interface
- Add a Review
- Editing a Review
- Deleting a Review
- Filtering Reviews

General User Interface for the Review Window

The general user interface for the Review table will consist of a table in the middle of the application with search filters on the side of the application and options to add a review at the top of the application and if you're the creator of a review then a pencil will be beside your review so that you can edit the details of it and a bin so that you can delete your review. The columns of the table will consist of:

- Product Type
- Product Size
- Product Brand
- Product Name
- Rating
- Review
- Review Creator

Below the option to add a new review will be tabs containing other areas of the system.

Adding a Review

When the addition button (+) is pressed (at the top of the window), a pop out will appear. This will automatically fill in the Review Creator (first name and

last name). Below this drop down boxes allowing you to choose the: Product Type, Product Size and Product Brand. Below the drop down boxes the information is inserted via a text box, the user can give a product a rating restricted to 1-5 and type out a review of up to 500 characters. Once all the information has been added the review will be added to the database and the review will be able to be seen inside the table when on the Review Database page.

Editing a Review

By the side of any review that you have created there will be a 'pencil' icon which gives you the option to edit your in your table. Once this is clicked a pop up identical to the one that you are given when you click on the (+) button comes up; however all of the information is already filled in with the information from that review. From this pop up you can edit that specific reviews information, just as you would if you were adding a review.

Deleting a Review

By the side of any review that you have created there will be a 'bin' icon which gives you the option to delete a review from your table. Once this is clicked a confirmation will pop up to ensure that you want to permanently delete that review. Once that trick is deleted it wil be removed from the database and you will no longer be able to view it in your table of reviews.

Filtering Reviews

At the top of the application there are search filters represented by the drop down box, you can then click on each individual filter and select the appropriate values. There will be 3 search filters, Brand (the company that makes the product), Type (the part of the skateboard), Size (the size of the product). When the filters are selected it will systematically reduce the number of items in the table in response to the filters put in place.

2.1.2 System flowcharts showing an overview of the complete system

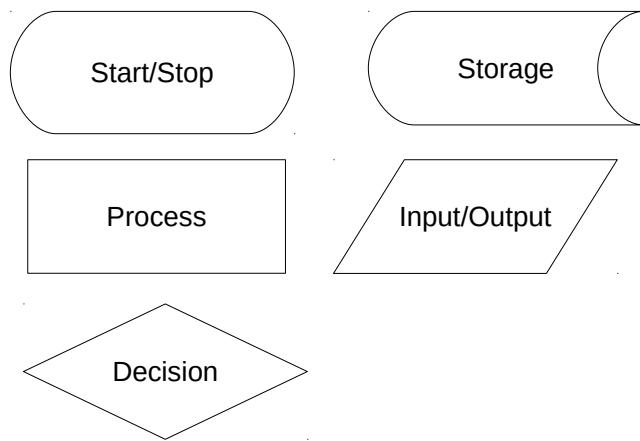


Figure 2.1: System Flowchart Key

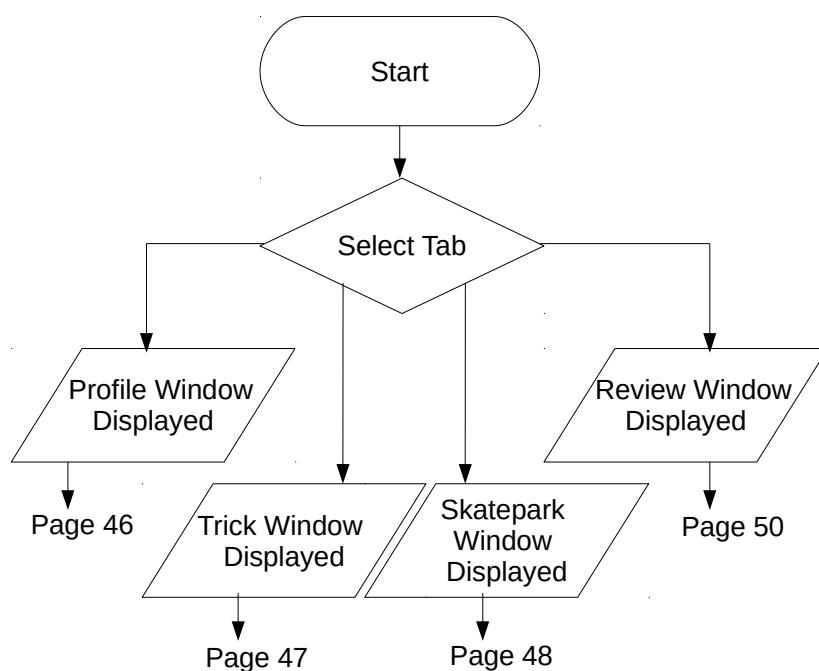


Figure 2.2: Profile Window Flowchart

The flowchart above shows the flow of operations between tabs.

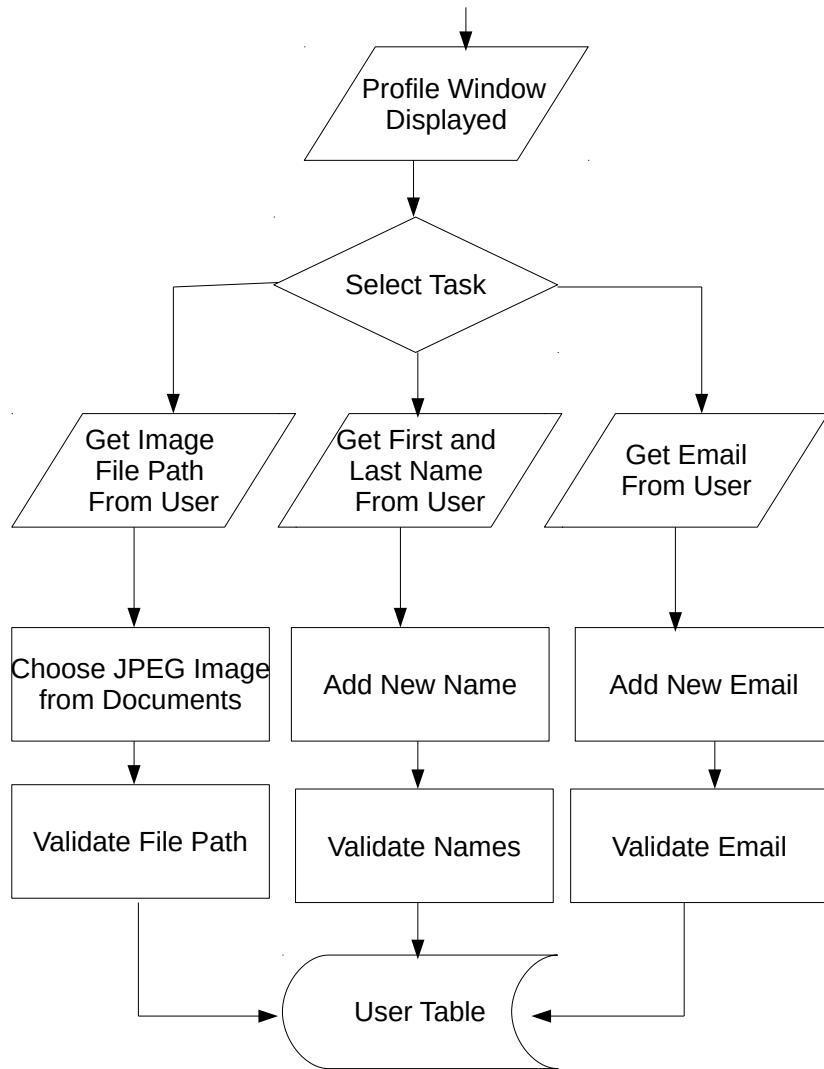


Figure 2.3: Profile Window Flowchart

The flowchart above displays the profile windows flow of operations. This shows the user is able to change their information, such as: Name, email and picture.

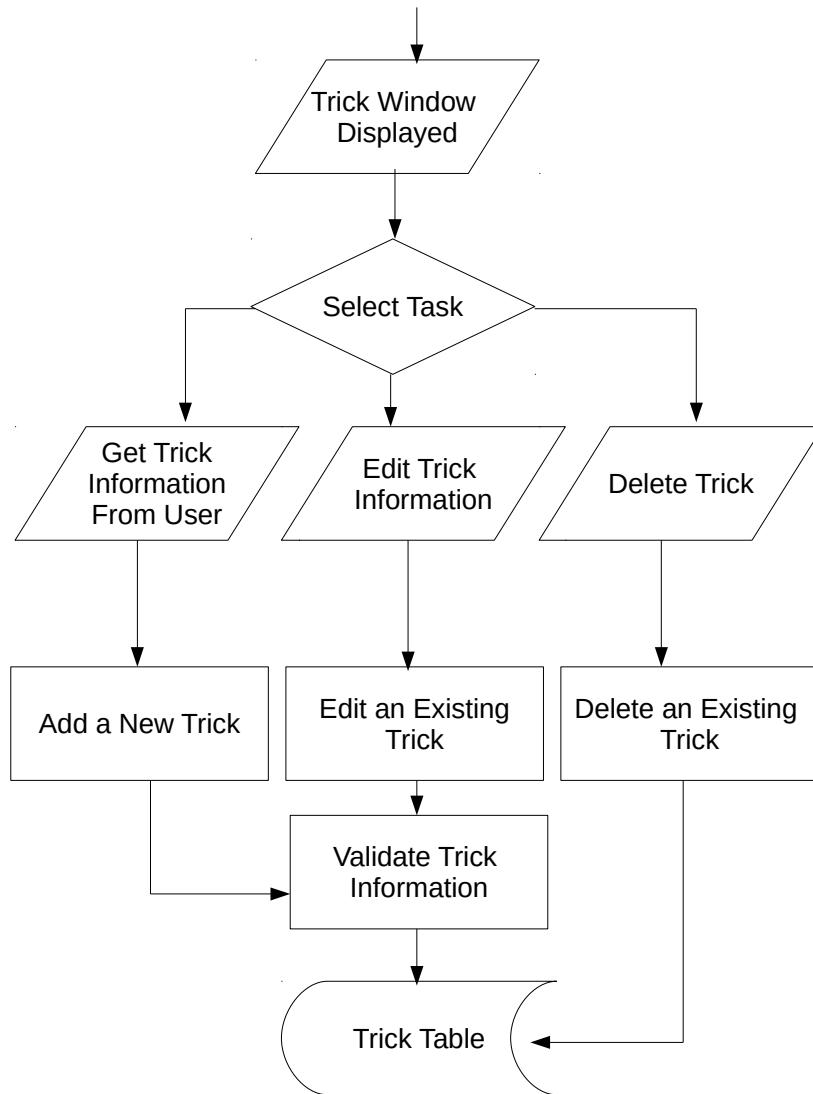


Figure 2.4: Trick Window Flowchart

The flowchart above displays the trick windows flow of operations. This shows the user is able to: add, edit and delete tricks.

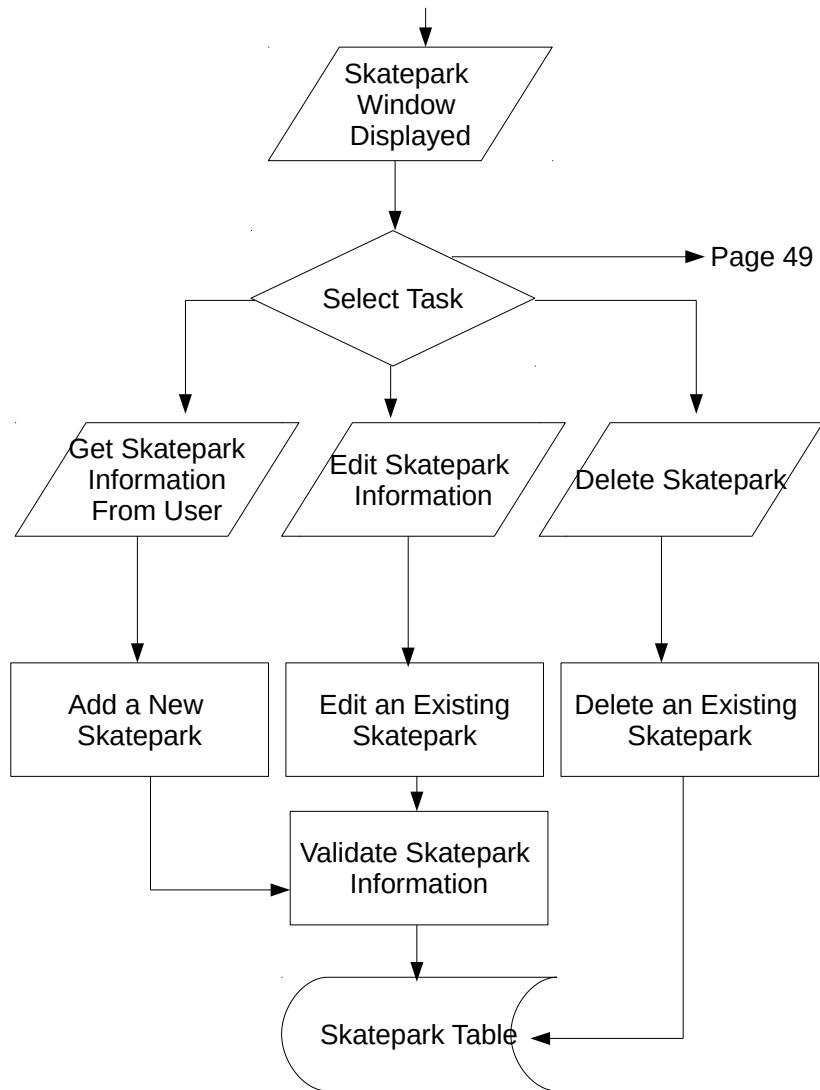


Figure 2.5: Skatepark Window Flowchart

The flowchart above displays the skatepark windows flow of operation. This shows the user can: add, edit and delete skateparks on and off of their map.

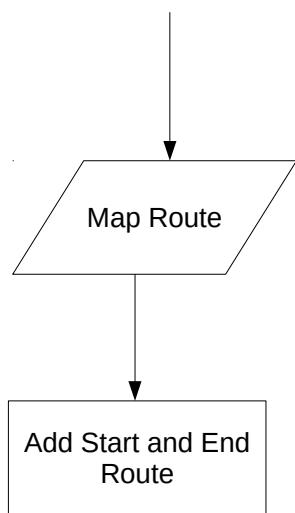


Figure 2.6: Skatepark Window Flowchart

The flowchart above is continued on from the previous flowchart and shows the user can map a route from location to location.

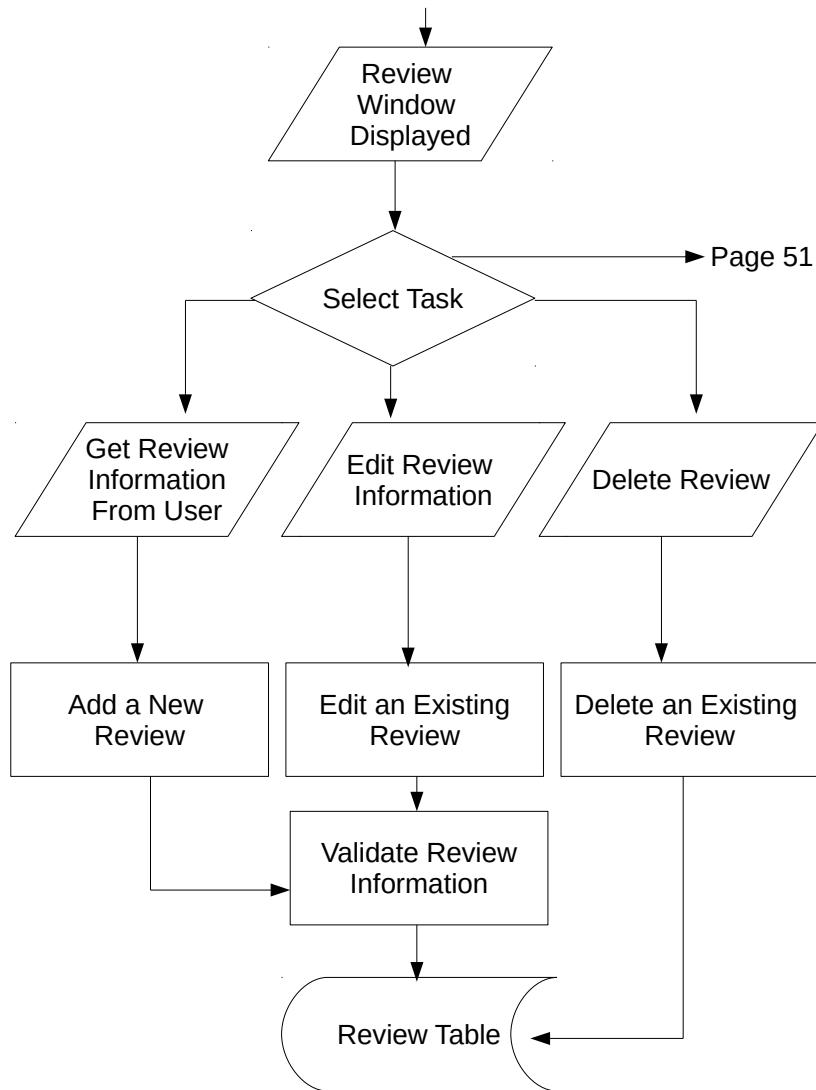


Figure 2.7: Review Window Flowchart

The flowchart above displays the review windows flow of operation. This shows the user can: add, edit and delete reviews.

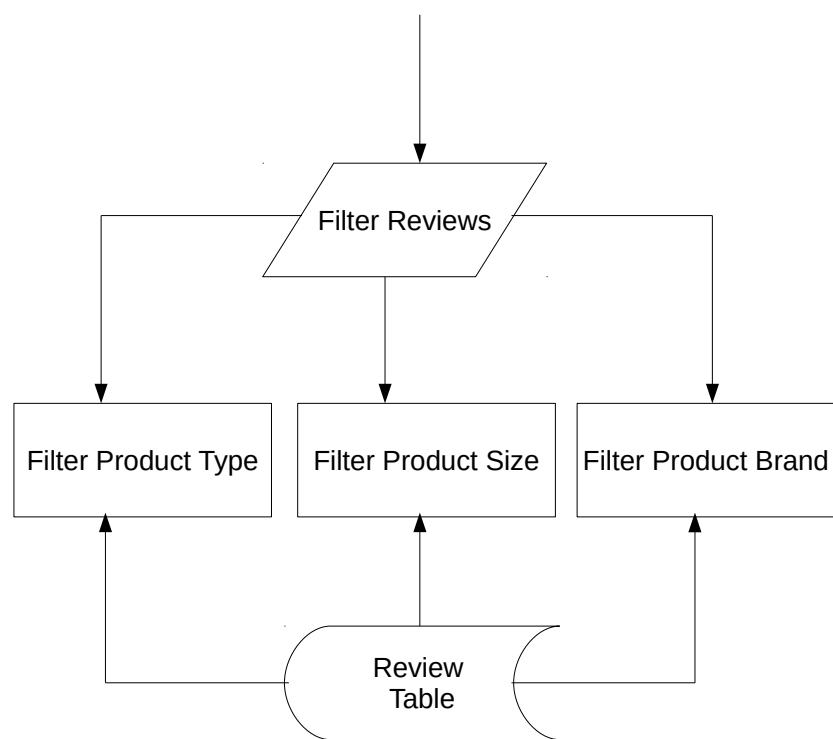


Figure 2.8: Review Window Flowchart

The flowchart above is continued on from the previous flowchart and shows the user can filter reviews via: brands, size and type.

2.2 User Interface Designs

The User Interface shown below occurs on a one off occasion when no profile information is found in the database. This screen allows the user to add a profile. This allows you to add your name, email and picture to your profile. A introductory message is also included to guide the user through the set-up process.

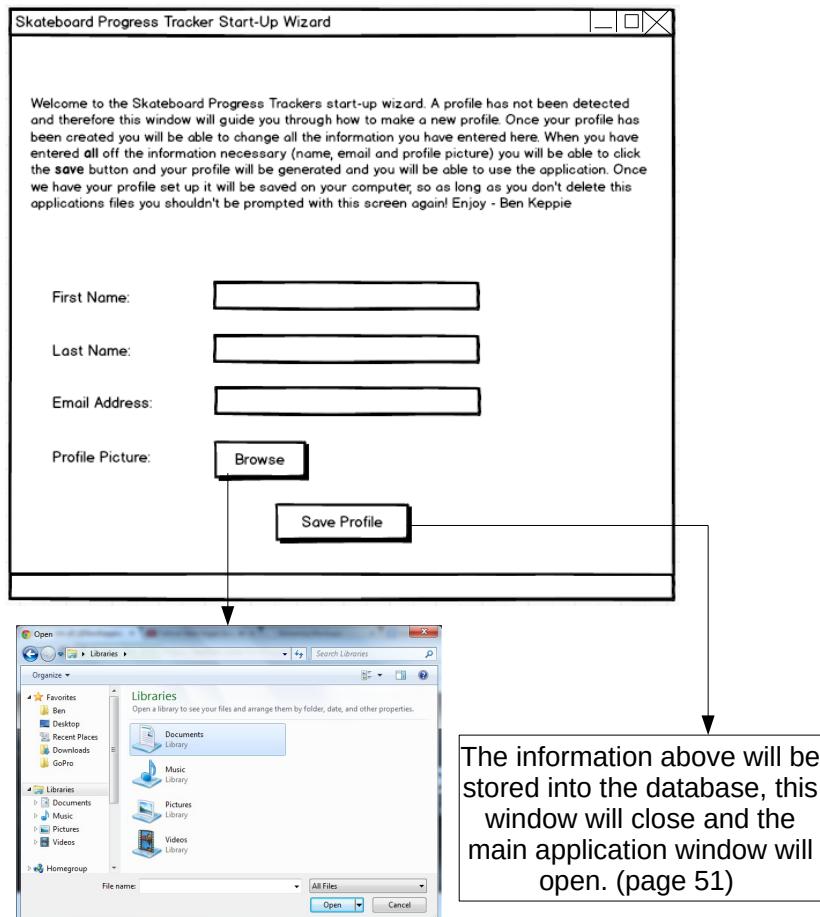


Figure 2.9: The User Interface for the Start-Up Wizard

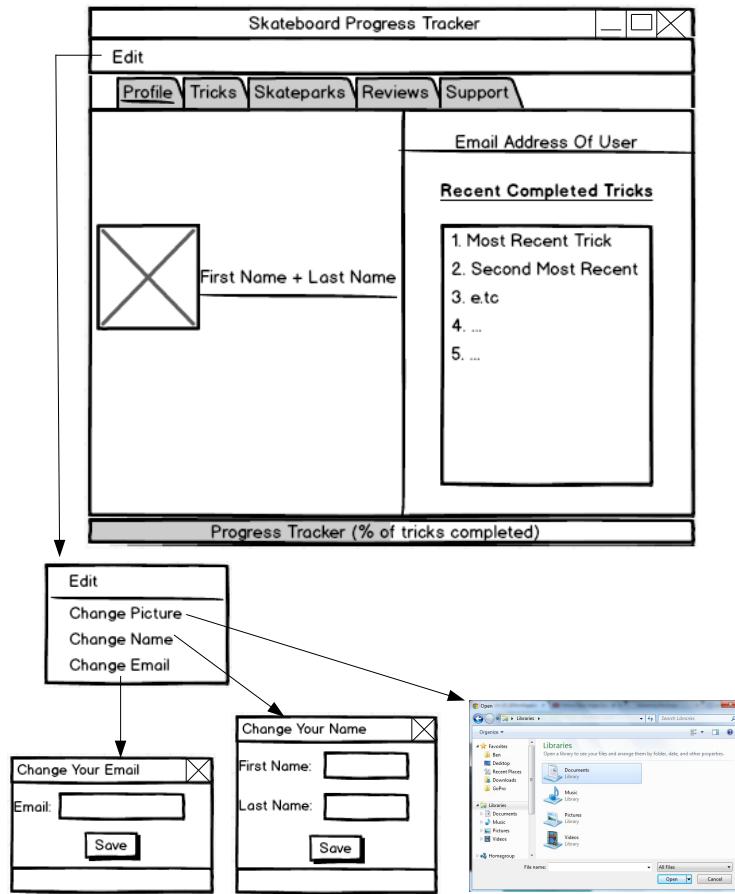


Figure 2.10: The User Interface for the profile section

This is the start-up page (the profile) of the application once a profile has been created. It contains the users profile with an image, name, email, progress tracker and a list of recently completed tricks. All of the information can be edited by the 'Edit' menu bar which contains 3 options (change profile picture, change name and change the email address). The tabs below the menu bar can be used to navigate between the windows of the application. These are displayed on each window and kept in the same position for ease of use. There is also a progress tracker at the bottom of the window where the user can see how many tricks they have completed out of the tricks in there table.

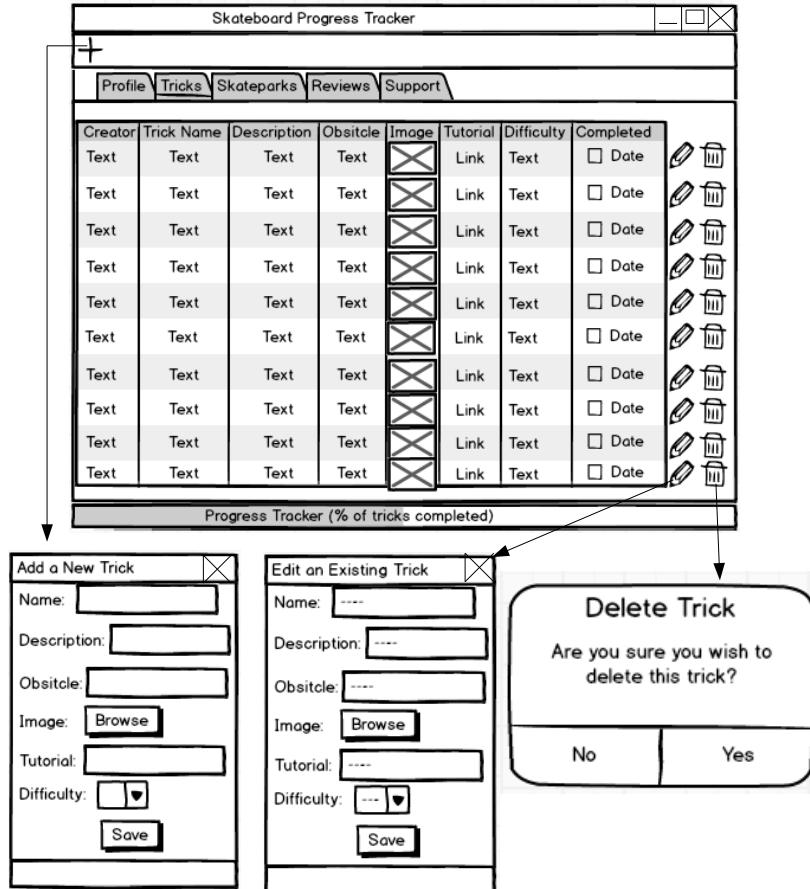


Figure 2.11: The User Interface for the trick section

The Tricks window of the application contains the same progress tracker as discussed in the profile user interface and a table in the main window full of tricks and their information. By the side of each trick there are icons including a pencil and a bin which represent 'edit' and 'delete'. I have decided to use these icons as they're recognisable, aesthetically pleasing and don't use up as much space as words, this allows for the table to be bigger. To add a trick the user can click the (+) symbol from the menu bar.

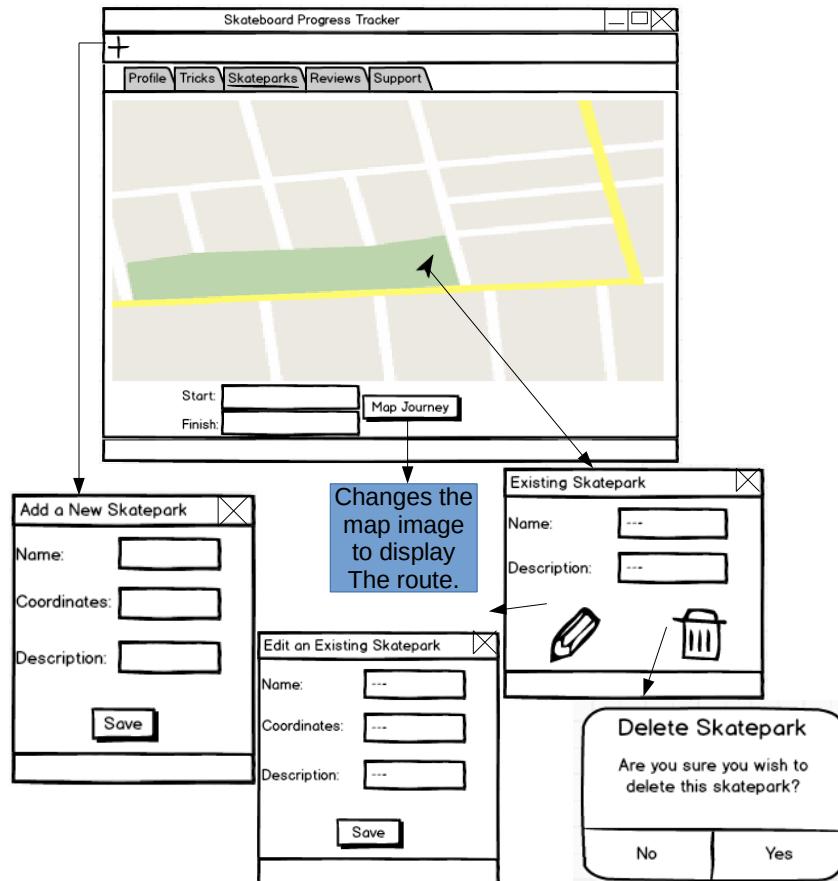


Figure 2.12: The User Interface for the skatepark section

The skatepark window contains a Google maps image in the centre of the window with a start and finish destination which can be used to map a route on the Google maps image. Once a skatepark is clicked on the maps information about that skatepark is given and there are options to edit and delete it, represented by a pencil and bin. I have used these symbols continuously through my application so that the user knows what the symbols mean. In the menu bar there is a (+) symbol which is used to add a new skatepark.

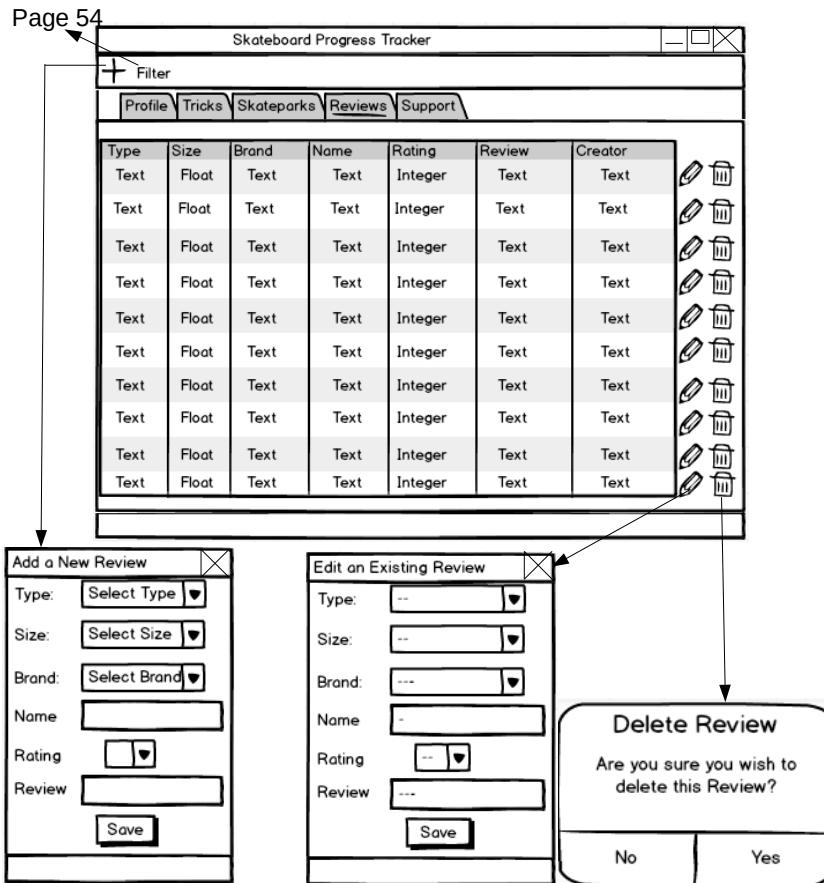


Figure 2.13: The User Interface for the review section

Like the trick window the review window has a table in the main window with a pencil and bin next to each row, and a (+) symbol is used to add a review. This continuity and re-use of symbols is all in place for ease of use. See the next page for filtering the review table.

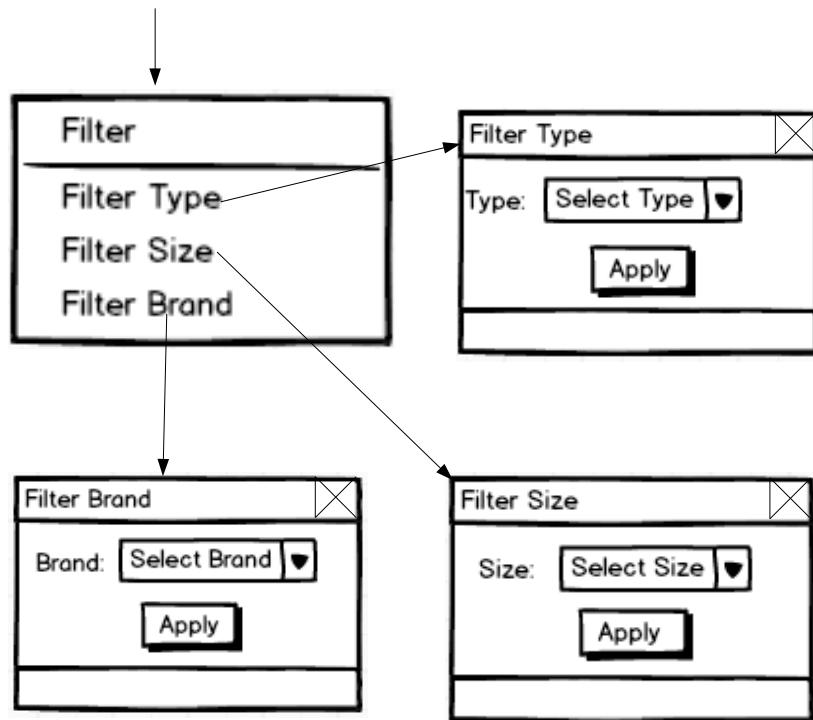


Figure 2.14: The continued User Interface for the review section

In the menu bar there is a 'Filter' option which allows for the user to filter the table for: Type, Brand and Size. These filters are in place so that the user can easily filter and narrow down the table to find the reviews that you want.

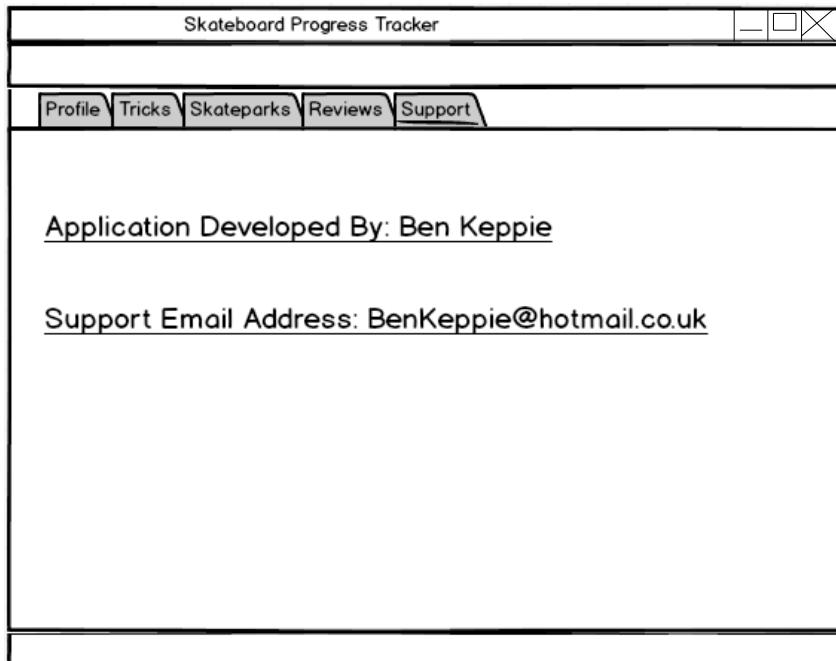


Figure 2.15: The User Interface for the support section

This window is available so that if there are any problems any user can contact the developer to fix them.

2.3 Hardware Specification

The system will need to run on Stuart's laptop. This means the program will have to work with a 1360x768, 16:9 aspect ratio screen and also windows 7. This is important as I have to make sure my program will fit on this screen size as the program is being built to Stuarts laptop specifications. This is an important

factor as all of the applications features will need to be aesthetically pleasing in many areas of the application such as the tables of information and the skatepark mapping section. A keyboard will be needed for inputting the information to the program, such as adding tricks or skateparks to the database. A track pad/mouse will be used to navigate around the program and the laptop screen will be used for the output of the program. The database and application data will be stored on the hard drive of the user. The cost of extra hardware totals to £0 as my client has already purchased the necessary equipment. This is beneficial as no extra hardware needs to be purchased which makes it readily available and suitable for purpose.

2.4 Program Structure

2.4.1 Top-down design structure charts

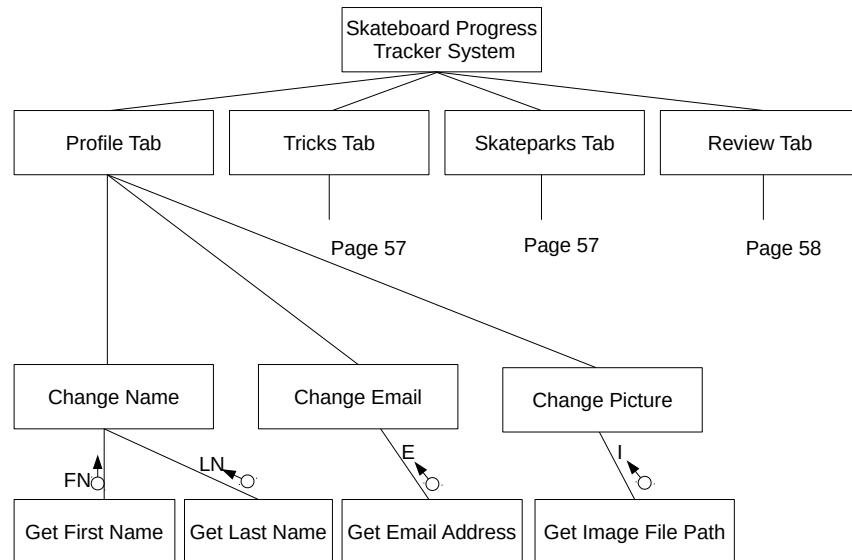


Figure 2.16: Profile Top-Down Design Chart

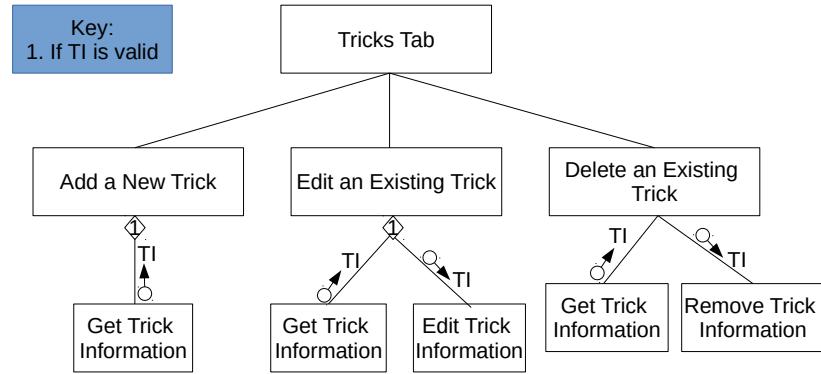


Figure 2.17: Tricks Top-Down Design Chart

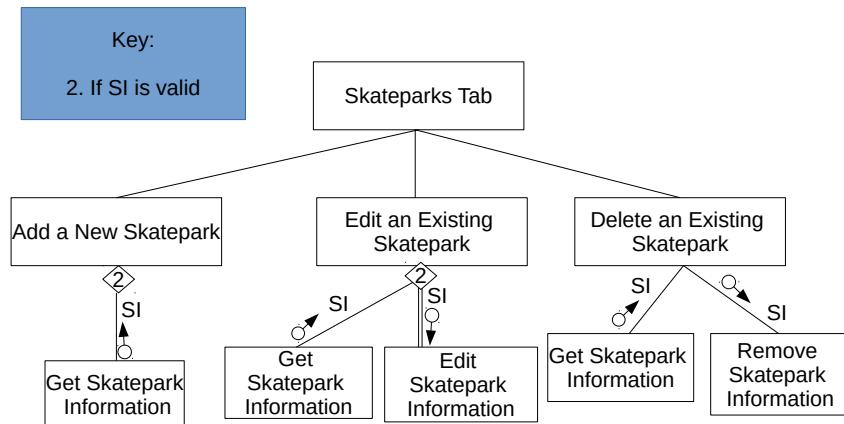


Figure 2.18: Skateparks Top-Down Design Chart

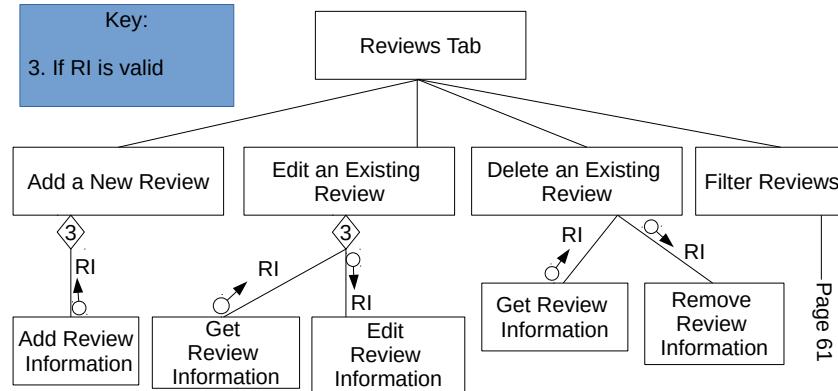


Figure 2.19: Reviews Top-Down Design Chart

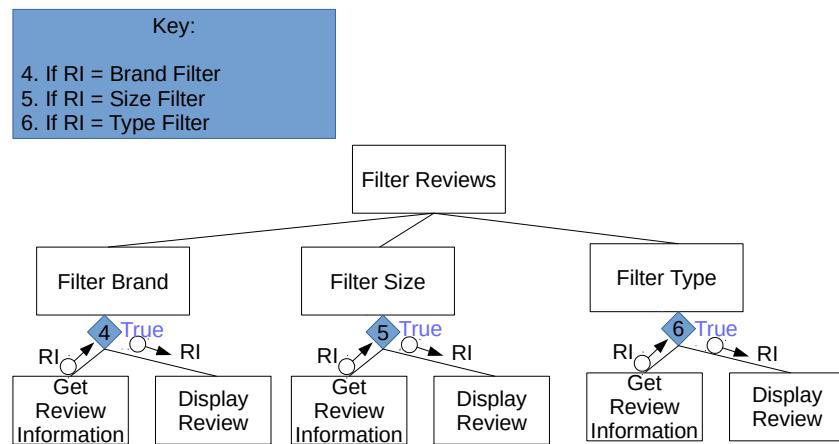


Figure 2.20: Review Filters Top-Down Design Chart

2.4.2 Algorithms in pseudo-code for each data transformation process

Algorithm 7 Algorithm For The Progress Tracker Bar

```
1: FUNCTION PROFILE_TRACKER(CompletedTricks,AllTricks)
2:   LengthCompletedTricks ← LEN(CompletedTricks)
3:   LengthAllTricks ← LEN(AllTricks)
4:   ProgressPercentage ← LengthCompletedTricks/LengthAllTricks*100
5: ENDFUNCTION
```

Algorithm 8 Algorithm For Mapping a Route

```
1: FUNCTION MAP_ROUTE(StartLocation,EndLocation)
2:   StartLocationCoordinates ← GEOCODING(StartLocation)
3:   EndLocationCoordinates ← GEOCODING(EndLocation)
   MAPROUTE(StartLocationCoordinates,EndLocationCoordinates)
4: ENDFUNCTION
```

Algorithm 9 Algorithm For Adding a Skatepark Marker to the Map

```
1: FUNCTION SKATEPARK_MARKER(SkateparkLongitude,SkateparkLatitude)
2:   marker ← Google.maps.marker(SkateparkLongitude,SkateparkLatitude)
3: ENDFUNCTION
```

2.4.3 Object Diagrams

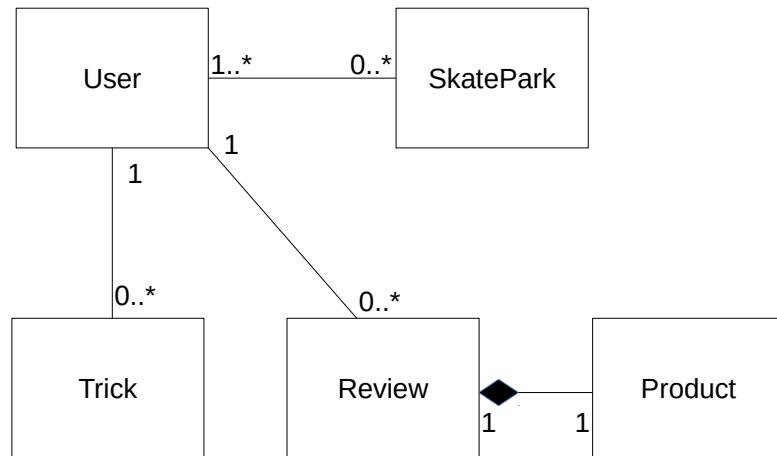


Figure 2.21: Relationship Diagram

2.4.4 Class Definitions

User
UserID
UserPicture
UserEmail
Username
get_userid
get_user_picture
get_user_email
get_username

Trick
TrickName
TrickDescription
TrickDifficulty
Trickobstacle
TrickCompleted
TrickImage
TrickTutorialLink
get_trick_name
get_trick_description
get_trick_obstacle
get_trick_difficulty
get_trick_state
get_trick_image
calculate_tricks_completed
calculate_tricks_progress_percentage
get_trick_tutorial_link

SkatePark
SkateparkID
SkateparkName
SkateparkCoordinates
SkateparkDescription
get_skatepark_id
get_skatepark_name
get_skatepark_coordinates
get_skatepark_description
add_new_skatepark
edit_existing_skatepark
delete_existing_skatepark
set_skatepark_marker
map_skatepark_route

Review
ReviewID
get_review_id
add_new_review
edit_existing_review
delete_existing_review

Product
ProductName
ProductSize
ProductBrand
ProductType
ProductReview
get_product_name
get_product_size
get_product_brand
get_product_type
get_product_review
filter_product_brand
filter_product_type
filter_product_size

2.5 Prototyping

Inserting a Webpage into PyQt

For the 'Skateparks' section of my system I would need to be able to add Google maps into my application. For this to work I would need to be able to view a web page in the main window of my PyQt application. I successfully managed to integrate the web page into the main window, this can be seen below.

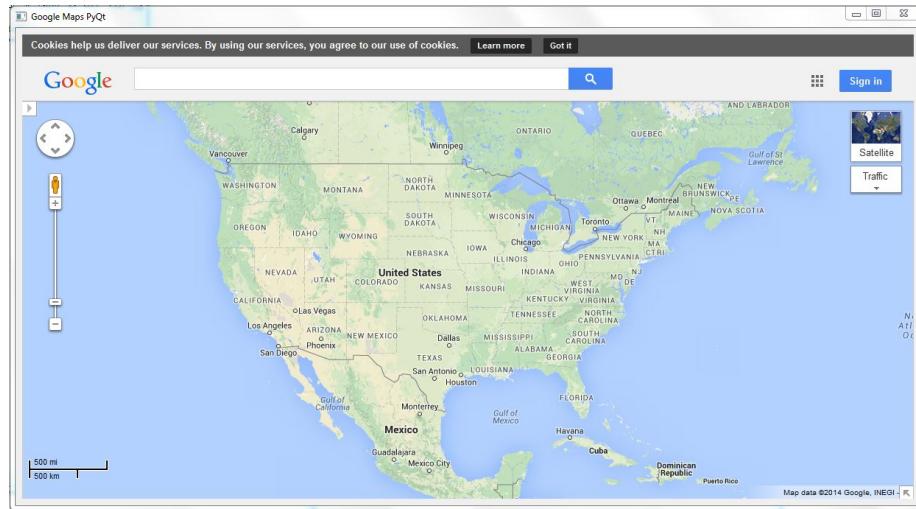


Figure 2.22: Google Maps in Python Application

My code for this is shown below.

```

1 import sys
2 from PyQt4.QtGui import *
3 from PyQt4.QtCore import *
4 from PyQt4.QtWebKit import *

5
6 class MainWindow(QMainWindow):
7     """The main window for my application"""
8     def __init__(self):
9         super().__init__()
10        self.setWindowTitle("Google Maps PyQt")
11        self.create_layout()

12    def create_layout(self):
13        self.label=QWebView()
14        self.label.load(QUrl("http://www.Google.com/maps"))
15        self.label.show()

16        self.layout=QVBoxLayout()
17        self.layout.addWidget(self.label)

18        self.widget=QWidget()
19        self.widget.setLayout(self.layout)
20        self.setCentralWidget(self.widget)

21
22 if __name__ == "__main__":
23     application= QApplication(sys.argv)
24     window=MainWindow()
25     window.show()
26     window.raise_()
27     application.exec_()

```

Google Maps API

For my skateparks section of my program I had to think about a way to represent all the skateparks on the map. I found out after researching about Google maps API, that embedding Google maps into my program using HTML would provide a better user interface then the whole web page as it cuts out the parts of the web page which are not needed. I also found out a way using HTML to place markers which is a possible way of representing the individual skateparks on the map. My code for this is shown below.

```

1 self.Google_maps=QWebView()
2 self.html=( '''<iframe width="100%" height="100%"
3           frameborder="0" style="border:0"
4           src="https://www.Google.co.uk/maps/embed/v1/place?
5           key=AIzaSyC5RcJ7vLSEYF32KqDusnuRcLJiHW8EbDg

```

```

5      &q=long+road+sixth+form+college
6      &attribution_source=Google+Maps+Embed+API
7      &attribution_web_url=http://www.butchartgardens.com/
8      &attribution_ios_deep_link_id=comGooglemaps://?daddr=long+road+sixth
9      +form+college"> </iframe> ''')
10 self.Google_maps.setHtml(self.html)

```

This places a marker on the map at my college, Long Road Sixth Form. The code produces an embedded map with a pin marker located at Long Road Sixth Form. This is shown below.

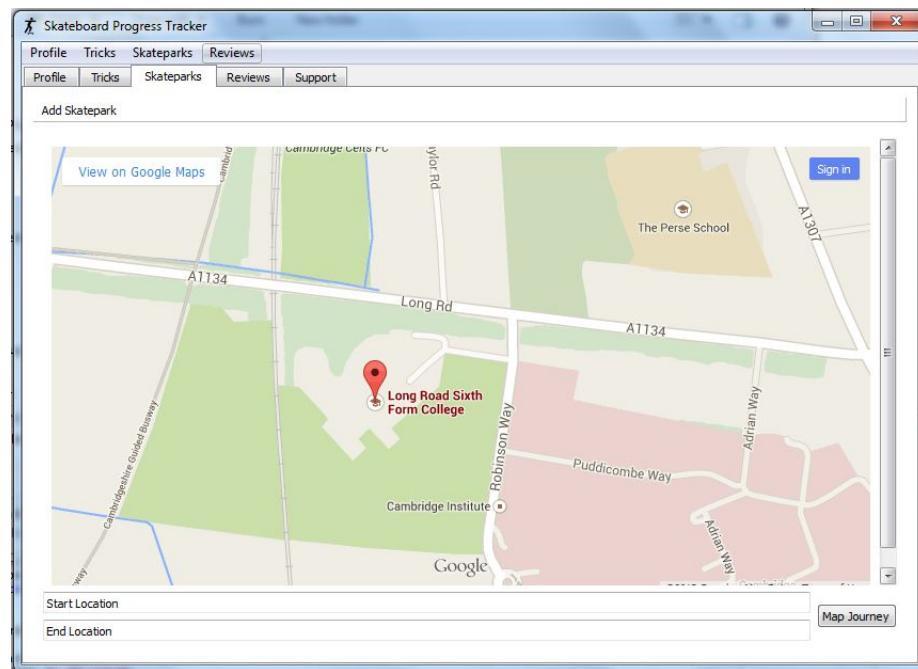


Figure 2.23: Pin Marker on Embedded Google Maps

Using Tabs To Navigate Through Windows

For my system I have decided to use a tabbed interface to navigate through my application. I have never used this form of navigation before and have decided to try and use this aesthetically pleasing and easy to use form of navigation. I investigated tabs and found that this was possible by using a QTabWidget. From this I then used my existing knowledge on how to add widgets to layouts and layouts to windows to produce this code:

```
1 import sys
2 from PyQt4.QtGui import *
3 from PyQt4.QtCore import *
4
5 class MainWindow(QMainWindow):
6     def __init__(self):
7         super().__init__()
8         self.setWindowTitle("Tabbed Interface")
9         self.create_tabs()
10
11     def create_tabs(self):
12
13         self.tabs=QTabWidget()
14
15         #Create Widgets
16         self.profile_tab=QWidget()
17         self.tricks_tab=QWidget()
18         self.skateparks_tab=QWidget()
19         self.reviews_tab=QWidget()
20         self.support_tab=QWidget()
21
22         #Add Tabs
23         self.tabs.addTab(self.profile_tab, "Profile")
24         self.tabs.addTab(self.tricks_tab, "Tricks")
25         self.tabs.addTab(self.skateparks_tab,
26                         "Skateparks")
27         self.tabs.addTab(self.reviews_tab, "Reviews")
28         self.tabs.addTab(self.support_tab, "Support")
29
30         self.setCentralWidget(self.tabs)
31
32 if __name__=="__main__":
33     application=QApplication(sys.argv)
34     window=MainWindow()
35     window.show()
36     window.raise_()
37     application.exec_()
```

Figure 2.24: Tab Navigation Code

This code then produced the window below. This program allowed me to navigate through tabs.

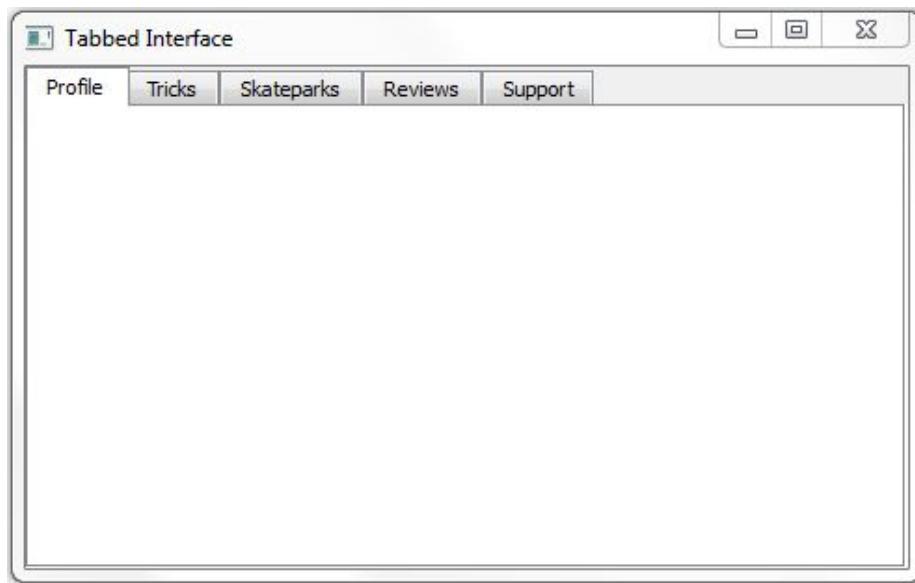


Figure 2.25: Tabbed Navigation in Python Application

Displaying a Database Table into a window

For two areas of my system tables from my database have to be displayed, as this feature is key for my tricks and reviews section I have decided to prototype it. I had previous experience in reading databases and displaying tables but I had never designed a program to read the database automatically from program start-up and display the able instantly. To do this I found out that all I needed to do was manually code the file path to the database; however I needed this to work on every computer in order to make it possible to I used the code below in order to do this.

```
1 self.path = ("{}{}".format(os.getcwd(), "\skateboard_progress_tracker.db"))
```

2.6 Definition of Data Requirements

2.6.1 Identification of all data input items

Data	Description
FirstName	The first name of the user
LastName	The last name of the user
UserPicture	The picture selected by the user for a profile picture
UserEmail	The email address of the user
TrickName	The name of a trick being added to the trick table
TrickDescription	The description of the trick being added to the trick table
TrickObstacle	Any obstacle needed to perform the trick being added to the trick table
TrickImage	The picture selected by the user for the trick being added to the trick table
TrickTutorialLink	The video link for a tutorial for the trick being added to the trick table
TrickDifficulty	The difficulty of the trick being added to the trick table
SkateparkName	The name of a skatepark being added to the skatepark table
SkateparkCoordinates	The coordinates of the skatepark being added to the skatepark table
SkateparkDescription	The description of the skatepark being added to the skatepark table
ReviewDescription	The written review for a product
ProductBrand	The brand of the product that is being reviewed
ProductName	The name of the product being reviewed
ProductSize	The size of the product being reviewed
ReviewRating	The rating of the product being reviewed

2.6.2 Identification of all data output items

Data	Description
UserPicture	The picture selected by the user for a profile picture
TrickImage	The picture selected by the user for the trick being added to the trick table
TrickCompleted	A checkbox indicating a trick is completed
TrickCompletedDate	A date indicating when the trick was completed
ReviewDescription	The written review for a product will be displayed when the review filter fits the reviews criteria
ProductBrand	The brand of the product that is being reviewed will be displayed when the review filter fits the reviews criteria
ProductName	The name of the product being reviewed will be displayed when the review filter fits the reviews criteria
ProductSize	The size of the product being reviewed will be displayed when the review filter fits the reviews criteria
ReviewRating	The rating of the product being reviewed will be displayed when the review filter fits the reviews criteria

2.6.3 Explanation of how data output items are generated

The UserPicture and TrickImage is displayed to the user by a file path which is selected by the user in the setting up of the profile and when the user is adding a trick.

The TrickCompleted is generated by the user clicking the checkbox to show that they have completed that trick. This will then display the checkbox as being checked.

The TrickCompletedDate is generated by the users clock on their computer. The date will be generated by the python function to call the time now. This date will then be displayed in the same column as the TrickCompleted checkbox.

The ReviewDescription, ProductBrand, ProductName, ProductSize and ReviewRating are all placed through a query to determine whether the review fits the criteria of the specific filter. If the review data fits the review filter then the data will be displayed to the user.

2.6.4 Data Dictionary

My Data Dictionary is displayed below, this contains quite a few modifications since my analysis section. This is because I have realised that for filtering through information I would need more attributes so that users can't spell things such as brand names wrong. Additionally I have decided to add some more user features such as a user picture so that the user interface will be better to look at and more user friendly.

Data dictionary

Ben Keppie

Candidate No. 4609

Centre No. 22151

Name	Data Type	Length	Validation	Example Data	Comment
UserID	Integer	10 Numbers	None	1	Unique identifier for a user
FirstName	String	20 Characters	Presence, no numbers, no special characters	Ben	None
LastName	String	20 Characters	Presence, no numbers, no special characters	Keppie	None
UserPicture	Image	N/A	160x160 pixels	UserPicture.jpeg	None
UserEmail	string	55 characters	contains @ and .com/.co.uk	BenKeppie@hotmail.co None	None
TrickCreator	String	41 Characters	Adds First and last name	Ben Keppie	None
TrickID	Integer	10 numbers	None	1	Unique identifier for a trick
TrickName	String	25 characters	None	Ollie	Linked to Description, image and tutorial link
TrickDescription	String	100 characters	None	Board is turned around 180 degrees	Linked to trick, image and tutorial link
TrickObstacle	String	25 characters	None	Half Pipe	None
TrickImage	Image	N/A	670 x 503 pixels	Ollie.jpeg	None
TrickTutorialLink	String	100 characters	Correct link	http://www.youtube.com/watch?v=3809	Linked to trick, description and image
TrickDifficulty	string	6 characters	easy, medium, hard	easy	colour coded
TrickCompleted	Boolean	True/False	None	True	None
TrickCompletedDate	String	10 characters	DD/MM/YYYY	15/07/2014	None

SkateparkID	interger	10 numbers	None	1	Unique identifier for a skatepark
SkateparkName	String	25 characters	Correct Name	Cambourne Skatepark	None
SkateparkCoordinates	Float	20 characters	Correct coordinates	52.2200 N, 0.0700 W	None
SkateparkDescription	String	200 characters	Accurate description	Halfpipe only	None
ReviewID	interger	10 numbers	None	1	Unique identifier for a review
ReviewDescription	String	500 characters	Non-biased	These trucks are the best I have owned	Moderated
ReviewRating	interger	range 1-5	Non-biased	1	Moderated
ReviewCreator	String	41 Characters	Adds First and last name	Ben Keppie	None
ProductID	interger	10 numbers	None	1	Unique identifier for a product
ProductBrandID	interger	10 numbers	None	1	Unique identifier for a brand
ProductBrand	String	20 characters	None	ZERO	Moderated
ProductTypeID	interger	10 numbers	None	1	Unique identifier for a skate board part
ProductType	String	20 characters	None	Deck	Moderated
ProductName	String	25 characters	None	Cosmic Tiger	Moderated
ProductSizeID	interger	10 numbers	None	1	Unique identifier for a size.
ProductSize	String	20 characters	None	7.875"	Moderated

2.6.5 Identification of appropriate storage media

A Hard Drive Disk (HDD) will be an appropriate storage media as the system and its data (database) needs to be stored in a way which is easily accessible for the system to use. As Stuart's laptop has a HDD built in as its main source of storage, this is the only suitable storage media for his situation. An external HDD will also allow for long term memory storage and allow for syncing between two computers, whilst also suiting the purpose of storing a back up. This is due to the external HDD's portability and security properties.

2.7 Database Design

2.7.1 Normalisation

ER Diagrams

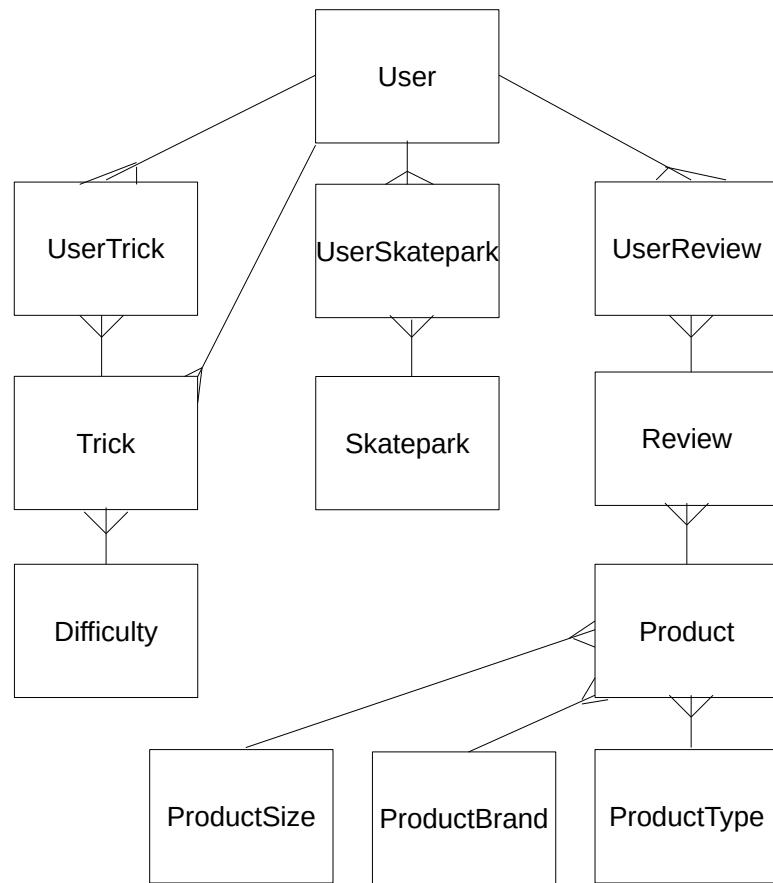


Figure 2.26: Entity-Relationship Diagram

Entity Descriptions

User(UserID, FirstName, LastName, UserPicture, UserEmail)

UserTrick(UserID, TrickID)

Trick(TrickID, DifficultyID, TrickCreator, TrickName, TrickDescription, TrickObstacle, TrickImage, TrickTutorialLink, TrickCompleted, TrickCompletedDate)

Difficulty(DifficultyID, TrickDifficulty, DifficultyDescription)

UserReview(UserID, ReviewID)

Review(ReviewID, ProductID, ReviewCreator, ReviewDescription, ReviewRating)

Product(ProductID, ProductBrandID, ProductTypeID, ProductSizeID, ProductName)

ProductBrand(ProductBrandID, ProductBrand)

ProductType(ProductTypeID, ProductType)

ProductSize(ProductSizeID, ProductSize)

UserSkatepark(UserID, SkateparkID)

Skatepark(SkateparkID, SkateparkName, SkateparkCoordinates, SkateparkDescription)

1NF to 3NF

The stages below show how my data has gone from UNF to 3NF via the process of normalisation.

Un-Normalised
UserID
FirstName
LastName
UserPicture
UserEmail
TrickCreator (UserID)
TrickID
TrickName
TrickDescription
Trickobstacle
TrickImage
TrickTutorialLink
DifficultyID
TrickDifficulty
DifficultyDescription
TrickCompleted
TrickCompletedDate
SkateparkID
SkateparkName
SkateparkCoordinates
SkateparkDescription
ReviewID
ReviewDescription
ProductID
ProductBrand
ProductType
ProductName
ProductSize
ReviewCreator (UserID)
ReviewRating

1NF	
Repeating	Non-Repeating
<u>UserID</u>	<u>UserID</u>
<u>TrickID</u>	FirstName
TrickName	LastName
TrickCreator (UserID)	UserPicture
TrickDescription	UserEmail
Trickobstacle	
TrickImage	
TrickTutorialLink	
DifficultyID	
DifficultyDescription	
TrickDifficulty	
TrickCompleted	
TrickCompletedDate	
SkateparkID	
SkateparkName	
SkateparkCoordinates	
SkateparkDescription	
ReviewID	
ReviewDescription	
ProductID	
ProductBrandID	
ProductTypeID	
ProductSizeID	
ProductBrand	
ProductType	
ProductName	
ProductSize	
ReviewCreator (UserID)	
ReviewDescription	
ReviewRating	

2NF
<u>UserID</u> FirstName LastName UserPicture UserEmail
<u>UserID</u> <i>TrickID</i>
<u>TrickID</u> TrickCreator (UserID) TrickName TrickDescription TrickObsitcle TrickImage TrickTutorialLink TrickDifficulty DifficultyID DifficultyDescription TrickCompleted TrickCompletedDate
<u>UserID</u> SkateparkID SkateparkName SkateparkCoordinates SkateparkDescription ReviewID ReviewDescription ProductID ProductBrandID ProductSizeID ProductTypeID ProductBrand ProductType ProductName ProductSize ReviewCreator (UserID) ReviewRating

3NF	
<u>UserID</u>	FirstName
	LastName
	UserPicture
	UserEmail
<u>UserID</u>	<i>TrickID</i>
<u>TrickID</u>	<i>DifficultyID</i>
	TrickCreator (UserID)
	TrickName
	TrickDescription
	Trickobstacle
	TrickImage
	TrickTutorialLink
	TrickCompleted
	TrickCompletedDate
<u>DifficultyID</u>	TrickDifficulty
	DifficultyDescription
<u>UserID</u>	<i>ReviewID</i>
<u>UserID</u>	<i>SkateparkID</i>
<u>SkateparkID</u>	SkateparkName
	SkateparkCoordinates
	SkateparkDescription
<u>ProductID</u>	<i>ProductBrandID</i>
	<i>ProductTypeID</i>
	<i>ProductSizeID</i>
	ProductName

<u>ReviewID</u>
<i>ProductID</i>
ReviewDescription
ReviewRating
ReviewCreator (UserID)
<u>ProductBrandID</u>
ProductBrand
<u>ProductTypeID</u>
ProductType
<u>ProductSizeID</u>
ProductSize

2.7.2 SQL Queries

For all of my SQL queries I will be using Python to format the SQL query text strings.

Query to Show Filtering the Product Type

The query below shows the SQL query that will be used to filter the reviews for a specific product type. This takes all the information from a review (in the Review table) and displays it if the ProductTypeID (from the ProductType table) equals the filter that is set. The filter will be selected via a drop down box in the 'Filter reviews' pop-out.

```

1 SELECT *
2 FROM Review, Product
3 WHERE Product.ProductTypeID=?

```

Query to Show Filtering the Product Size

The query below shows the SQL query that will be used to filter the reviews for a specific product size. This takes all the information from a review (in the Review table) and displays it if the ProductSizeID (from the ProductSize table) equals the filter that is set. The filter will be selected via a drop down box in the 'Filter reviews' pop-out.

```

1 SELECT *
2 FROM Review, Product
3 WHERE Product.ProductSizeID=?

```

Query to Show Filtering the Product Brand

The query below shows the SQL query that will be used to filter the reviews for a specific product brand. This takes all the information from a review (in the Review table) and displays it if the ProductBrandID (from the ProductBrand table) equals the filter that is set. The filter will be selected via a drop down box in the 'Filter reviews' pop-out.

```
1 SELECT *
2 FROM Review, Product
3 WHERE Product.ProductBrandID=?
```

Query to Show How Many Tricks Have Been Completed

The query below shows the SQL query that will be used to find how many tricks have been completed. This SQL query generates the basis for my progress tracker algorithm shown in a previous section.

```
1 SELECT TrickID
2 FROM Trick
3 WHERE TrickCompleted=True
```

Query to Show How Many Tricks are in the Trick Table

The query below shows the SQL query that will be used to find out how many tricks are in the trick table. This SQL query also generates the basis for my progress tracker algorithm shown in a previous section.

```
1 SELECT TrickID
2 FROM Trick
```

Query to Order the Trick Database in Alphabetical Order

The query below shows how I will order the trick QTableView in my program to display all the tricks in alphabetical order.

```
1 SELECT *
2 FROM Trick
3 ORDER BY TrickName ASC
```

2.8 Security and Integrity of the System and Data

2.8.1 Security and Integrity of Data

Due to the system containing some private information about a living individual (name and email), the new system will have to abide by the data protection act.

Location data about the user will need to be secured as that information could be used to find out where a living person is going. To make sure that the data that is stored is also valid, at the input stage, drop down menus will be used when necessary e.g reviews brand. Wherever the user types in the information via the keyboard, the data will be checked to make sure that it is acceptable by the validation discussed in the next section. I will also need to make sure that I keep referential integrity in my database. I have desided to stick with the default: ON UPDATE RESTRICT ON DELETE RESTRICT as this will prevent users of my system from mistakenly altering the database in an unexpected way.

2.8.2 System Security

It is important that the system is protected from data theft, corruption and tampering. The database will be encrypted to avoid people accessing the information without the use of the system. As my program must abide by the data protection act I must ensure that the data:

- Will not be transferred to other countries.
- Will be secured securely so only authorised users can access it. To enforce this my database will be encrypted.
- Will be destroyed after 11 years of collection. To enforce this after 11 years the user will be forced to re-enter the personal data that the program stores before being able to access the program (name and email address).
- Will be accurate and up to date. To enforce this, periodically the program will display pop-ups reminding the user to ensure the information stored on the database is correct.
- Will be necessary. To enforce this, as the programmer I will only use the data for the specific purposes for which it was collected, e.g Profile Picture to display on the individuals users profile page.

2.9 Validation

To avoid any incorrect data entries from being added to the database the system needs to carry out some validation searches to ensure that each piece of information being added to the database is in acceptable parameters.

Item	Example	Validation	Justification
FirstName	Ben	Presence, no numbers, no special characters	To ensure a first name is entered and with only acceptable characters
LastName	Keppie	Presence, no numbers, no special characters	To ensure a last name is entered and with only acceptable characters
UserPicture	Picture.jpeg	JPEG image (will be re-sized to 160x160)	To ensure a standard file type and picture size
UserEmail	BenKeppie@hotmail.com	Ensure a standard format of email address	So only valid email addresses are entered
TrickName	Ollie	Presence check	To ensure a trick name is entered
TrickDescription	Board lifts off the ground	Presence check	To ensure a trick description is entered
Trickobstacle	Flat Ground	Presence check	To ensure a trick obstacle is entered
TrickImage	Ollie.jpeg	JPEG image (will be re-sized to 670x503)	To ensure a standard file type and picture size
TrickTutorialLink	http://www.youtube.com/watch?v=1	Presence, ensure the text is a web address	To ensure a link to a trick tutorial is valid
TrickDifficulty	Easy	Ensure an option is selected	To ensure that a difficulty is available for a trick
TrickCompletedDate	15/08/2014	Date is in the DD/MM/YYYY format	So a universal date format is available for completed tricks
SkateparkName	Cambourne	Presence	So a name is entered for a skatepark
Skatepark Coordinates	52.2200 N, 0.0700 W	Presence and correct coordinate format	So a usable coordinate is entered
SkateparkDescription	Halfpipe only	Presence	To ensure a skatepark description is entered
ReviewDescription	Amazing trucks, best I have owned	Presence	To ensure a review description is entered
ReviewRating	1 88	Presence, and only numbers 1-5 allowed	To ensure a correct value is entered for a rating

ProductBrand	ZERO	Presence	To ensure a brand is selected for a review
ProductType	Trucks	Presence	To ensure a type is selected for a review
ProductName	Spec Ops	Presence	To ensure a name is selected for the product of the review
ProductSize	5.0"	Presence	To ensure a size is selected for a review

2.10 Testing

Ben Keppie

Candidate No. 4609

Centre No. 22151

2.10.1 Outline Plan

Test Series	Purpose of Test Series	Testing Strategy	Strategy Rationale
1	Test the flow of control between user interfaces	Top-down testing	I have chosen top-down testing as the flow of user interfaces is hierarchical due to the fact there are multiple interfaces which stem from an original, main interface
2	Validation of input data performed correctly	Bottom-up Testing	I have chosen bottom-up testing as I need to test the lower levels of data input to ensure the information has been entered into the database. Only then I will be able to test other areas that use that information from the database
3	Test information input is stored in the correct place	White box testing	I have chosen white box testing as I will have to look into the database after I have inputted the data using the program to see that the data has been entered in the correct place
4	Test algorithms and SQL Queries to ensure the output is correct	Black box testing	I have chosen black box testing as I will see whether or not the algorithm/query has returned the correct values, without looking at the internal structure of the code
5	Test that the system fulfils the specification	Acceptance testing	I have chosen acceptance testing as this test is conducted to determine if the specification is met

2.10.2 Detailed Plan

Test Series	Purpose of Test	Test Description	Test Data	Test Data Type (Normal/ Erroneous/ Boundary)	Expected Result	Actual Result	Evidence
1.00	Test that the 'Profile' tab functions properly	This should load the profile window	Click the 'Profile' tab in the application	Normal	The profile window should be displayed		
1.01	Test the Change Name button on the profile window functions properly	A pop-up with two text boxes should display prompting you to enter your first and last name.	Click 'Edit' followed by 'Change Name'	Normal	A pop-up with two text boxes should display prompting you to enter your first and last name.		

1.02	Test the Change Email button on the profile window functions properly	A pop-up with a text box should display prompting you to enter your first and last name	Click 'Edit' followed by 'Change Email'	Normal	A pop-up with a text box should display prompting you to enter your first and last name	
1.03	Test the Change Picture button on the profile window functions properly	The default file browser for the system should open, allowing the user to select a jpeg image	click the 'Edit' button followed by the 'Change Picture' button	Normal	Default file browser should appear	
1.04	Test that the 'Tricks' tab functions properly	This should load the tricks window	Click the 'Tricks' tab in the application	Normal	The Tricks window should be displayed	
1.05	Test the add trick button functions properly	This should load a pop-up to add a trick	Click the (+) icon at the top left corner of the application	Normal	A pop-up prompting you to add a trick should appear	

1.06	Test the Edit Trick button (pencil next to a trick) functions properly	This should load a pop-up to edit a trick	Click the pencil icon next to a trick	Normal	A pop-up prompting you to edit a trick should appear		
1.07	Test the Delete Trick button (bin next to a trick) functions properly	This should load a pop-up to delete a trick	Click the bin icon next to a trick	Normal	A pop-up should ask you whether you wish to delete that trick		
1.08	Test that the 'Skateparks' tab functions properly	This should load the skateparks window	Click the 'Skateparks' tab in the application	Normal	The Skateparks window should be displayed		
1.09	Test the Add Skatepark button functions properly	This should load a pop-up to add a skatepark	Click the (+) icon at the top left corner of the application	Normal	A pop-up prompting you to add a skatepark should appear		
1.10	Test the Skatepark Location button functions properly	This should load a pop-up giving details about the skatepark	Click a location on a map	Normal	A pop-up giving you information about a skatepark		

1.11	Test the Edit Skatepark button (pencil icon in the existing skatepark pop-up) functions properly	This should load a pop-up to edit a skatepark	Click the pencil icon in the existing skatepark pop-up	Normal	A pop-up prompting you to edit a skatepark should appear		
1.12	Test the Delete Skatepark button (bin icon in the existing skatepark pop-up) functions properly	This should load a pop-up to delete a skatepark	Click the bin icon in the existing skatepark pop-up	Normal	A pop-up prompting you to delete a skatepark should appear		
1.13	Test the 'Map Journey' button functions properly	This should map a route on the map from the start and finish location	Click the 'Map Journey' icon	Normal	A route will be displayed on the map		
1.14	Test that the 'Reviews' tab functions properly	This should load the reviews window	Click the 'Reviews' tab in the application	Normal	The Reviews window should be displayed		

1.15	Test the Add Review button functions properly	This should load a pop-up to add a review	Click the (+) icon at the top left corner of the application	Normal	A pop-up prompting you to add a review should appear		
1.16	Test the Edit Review button (pencil next to a review) functions properly	This should load a pop-up to edit a review	Click the pencil icon next to a review	Normal	A pop-up prompting you to edit a review should appear		
1.17	Test the Delete Trick button (bin next to a review) functions properly	This should load a pop-up to delete a review	Click the bin icon next to a review	Normal	A pop-up should ask you whether you wish to delete that review		
1.18	Test the Filter Type button functions properly	This could load a pop-up to filter the type	Click the 'Filter' button then from the list select 'Filter Type'	Normal	A pop-up should ask you to select a type		

1.19	Test the Filter Brand button functions properly	This sould load a pop-up to filter the brand	Click the 'Filter' button then from the list select 'Filter Brand'	Normal	A pop-up should ask you to select a brand		
1.20	Test the Filter Size button functions properly	This sould load a pop-up to filter the size	Click the 'Filter' button then from the list select 'Filter Size'	Normal	A pop-up should ask you to select a size		
2.00	Verify an appropriate name is entered to the 'Change Name' pop-out.	Should not accept the name if it is not valid	1.Ben 2.Keppie 3. 4.12345 5.Ben10	1.Normal 2.Normal 3.Erroneous 4.Erroneous 5.Erroneous	1.Accept 2.Accept 3.Error (Presence) 4.Error (Numbers) 5.Error (Numbers)		
2.01	Verify an appropriate picture is selected in the 'Change Picture' pop-out	Should only accept JPEG images	1.Picture.JPG 2.Pic-ture.PNG 3.Picture.txt	G1.Normal 2.Erroneous 3.Erroneous	1.Accept 2.Error (File Type) 3.Error (File Type)		

2.02	Verify a valid email is entered to the 'Change Email' pop-out	Should only accept a correct email format	1.BenKeppie@h8tJih290ocukuk 2.BenKep-pieEmail.com		1. Normal 2. Erroneous 3. Erroneous	1. Accept 2. Error(Format) 3.Error(Format)	
2.03	Verify presence for adding a tricks name	Checks something is entered	1.Ollie 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.04	Verify presence for adding a trick description	Checks something is entered	1.Flips 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.04	Verify presence for adding a trick obstacle	Checks something is entered	1.Flat Ground 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.04	Verify presence for adding a trick tutorial link	Checks something is entered and that it is a website link	1.http://www.youtube.com/watch?v=1 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.05	Verify an appropriate picture is selected in the 'add a trick' pop-out	Should only accept JPEG images	1.Picture.JPG 2.Pic-ture.PNG 3.Picture.txt	1.Normal 2.Erroneous 3.Erroneous	1.Accept 2.Error (File Type) 3.Error (File Type)		

2.06	Verify a difficulty is selected	Drop down box with 3 options	1.Easy 2.Medium 3.Hard 4.	1.Normal 2.Normal 3.Normal 4.Erroneous	1.Accept 2.Accept 3.Accept 4.Error(Presence)		
2.07	Verify the date is in the correct format	Format=DD/MM/YY/2014 2.10/12/2014 3/12/15/2014		1.Erroneous 2.Normal 3.Erroneous	1.Error(Format) 2.Accept 3.Error(Format)		
2.08	Verify presence for adding a skatepark name	Checks something is entered	1.Cambourne 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.09	Verify the correct format of coordinates are entered	Check that the coordinates are correct	1.52.2200,0.0700 2. 3.30480839	1.Normal 2.Erroneous 3.Erroneous	1.Accept 2.Error(Presence) 3.Error(Format)		
2.10	Verify presence for a skatepark description	Checks something is entered	1.Halfpipe only 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.11	Verify presence for a review description	Checks something is entered	1.Amazing 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		

2.12	Verify presence and correct number range	Checks something is entered and the values are between 1 and 5	1.3 2.0 3. 4.r	1.Normal 2.Boundary 3.Erroneous 4.Erroneous	1.Accept 2.Error(Range) 3.Error(Presence) 4.Error(Character)		
2.13	Verify a product brand is selected	Checks a value is selected	1.ZERO 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.14	Verify a product type is selected	Checks a value is selected	1.Trucks 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.15	Verify a product size is selected	Checks a value is selected	1. 5.0" 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.16	Verify a product name is selected	Checks a value is selected	1.SpecOps 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
3.00	Verify the first and last name are inputted into the database	The first and last name should be added to the database	1.FirstName 2.LastName	1.Normal 2.Normal	1.Accept 2.Accept		

3.01	Verify the profile picture is inputted into the database	A jpeg image should be added to the database	JPEG image	Normal	Accept		
3.02	Verify an email is inputted into the database	An email should be added to the database	BenKeppie@hotmail.uk	Normal	Accept		
3.03	Verify a trick name is inputted into the database	A trick name should be added to the database	Ollie	Normal	Accept		
3.04	Verify a trick description is inputted into the database	A trick description should be added to the database	Board Rotates 360	Normal	Accept		
3.05	Verify a trick obstacle is inputted into the database	A trick obstacle should be added to the database	Flat ground	Normal	Accept		
3.06	Verify a trick image is inputted into the database	A trick image should be added to the database	JPEG Image	Normal	Accept		
3.07	Verify a trick tutorial link is inputted into the database	A trick tutorial link should be added to the database	www.youtube.com/watch?v=?	Normal	Accept		

3.08	Verify a trick difficulty is inputted into the database	A trick difficulty should be added to the database	Easy	Normal	Accept		
3.09	Verify a skatepark name is inputted into the database	A skatepark name should be added to the database	Cambourne Skatepark	Normal	Accept		
3.10	Verify skatepark coordinates are inputted into the database	Skatepark coordinates should be added to the database	52.2200,0.0700	Normal	Accept		
3.11	Verify a skatepark description is inputted into the database	A skatepark description should be added into the database	Half pipe	Normal	Accept		
3.12	Verify a review description is inputted into the database	A review description should be entered into the database	Amazing product	Normal	Accept		
3.13	Verify a product brand is inputted into the database	A product brand should be entered into the database	Product Brand (ZERO)	Normal	Accept		

3.14	Verify a product size is inputted into the database	A product size should be entered into the database	Product Size (5.0")	Normal	Accept		
3.15	Verify a product name is inputted into the database	A product name should be entered into the database	Product Name (Spec Ops)	Normal	Accept		
3.16	Verify a product type is inputted into the database	A product type should be entered into the database	Product Type (Truck)	Normal	Accept		
4.00	Verify that the product brand filter correctly returns the right reviews	Reviews with the product brand should be displayed	Select a brand filter (ZERO)	Normal	Only reviews that relate to the filter are displayed		
4.01	Verify that the product type filter correctly returns the right reviews	Reviews with the product type should be displayed	Select a type filter (Trucks)	Normal	Only reviews that relate to the filter are displayed		
4.02	Verify that the product size filter correctly returns the right reviews	Reviews with the product size should be displayed	Select a size filter (5.0")	Normal	Only reviews that relate to the filter are displayed		

4.03	Verify that the progress tracker returns the correct amount of completed tricks	Tricks which are completed will be displayed	Length of tricks completed	Normal	Only tricks that are completed will be displayed		
4.04	Verify that the progress tracker returns the correct amount of overall tricks	All tricks will be displayed	Length of tricks	Normal	All tricks will be displayed		
4.05	Verify that the skatepark is added to the correct location on the map	Longitude and latitude will correspond to map location	1.52.2200,0.0700	Normal	Skatepark will be displayed on the map		
4.06	Verify that the progress tracker displayed the correct percentage	Completed tricks divided by all tricks multiplied by 100	Tricks	Correct percentage will be displayed			
4.07	Verify that the route is correct	A correct route should be displayed on the map	Start Location, End Location	Normal	A correct route is displayed		

5	Verify the program fulfils the specification	Run through the program, testing the different aspects to make sure they fit the objectives in the specification	Add some information to the program, start a student test, and view the results of the test	Normal	Program fulfils the specification		
---	----------------------------------------------	------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------	--------	-----------------------------------	--	--

Chapter 3

Testing

3.1 Test Plan

Ben Keppie

Candidate No. 4609

Centre No. 22151

3.1.1 Original Outline Plan

Test Series	Purpose of Test Series	Testing Strategy	Strategy Rationale
1	Test the flow of control between user interfaces	Top-down testing	I have chosen top-down testing as the flow of user interfaces is hierarchical due to the fact there are multiple interfaces which stem from an original, main interface
2	Validation of input data performed correctly	Bottom-up Testing	I have chosen bottom-up testing as I need to test the lower levels of data input to ensure the information has been entered into the database. Only then I will be able to test other areas that use that information from the database
3	Test information input is stored in the correct place	White box testing	I have chosen white box testing as I will have to look into the database after I have inputted the data using the program to see that the data has been entered in the correct place
4	Test algorithms and SQL Queries to ensure the output is correct	Black box testing	I have chosen black box testing as I will see whether or not the algorithm/query has returned the correct values, without looking at the internal structure of the code
5	Test that the system fulfills the specification	Acceptance testing	I have chosen acceptance testing as this test is conducted to determine if the specification is met

3.1.2 Changes to Outline Plan

There were no changes made to my outline plan.

3.1.3 Original Detailed Plan

Test Series	Purpose of Test	Test Description	Test Data	Test Data Type (Normal/ Erroneous/ Boundary)	Expected Result	Actual Result	Evidence
1.00	Test that the 'Profile' tab functions properly	This should load the profile window	Click the 'Profile' tab in the application	Normal	The profile window should be displayed		
1.01	Test the Change Name button on the profile window functions properly	A pop-up with two text boxes should display prompting you to enter your first and last name.	Click 'Edit' followed by 'Change Name'	Normal	A pop-up with two text boxes should display prompting you to enter your first and last name.		

1.02	Test the Change Email button on the profile window functions properly	A pop-up with a text box should display prompting you to enter your first and last name	Click 'Edit' followed by 'Change Email'	Normal	A pop-up with a text box should display prompting you to enter your first and last name	
1.03	Test the Change Picture button on the profile window functions properly	The default file browser for the system should open, allowing the user to select a Trick.JPG	click the 'Edit' button followed by the 'Change Picture' button	Normal	Default file browser should appear	
1.04	Test that the 'Tricks' tab functions properly	This should load the tricks window	Click the 'Tricks' tab in the application	Normal	The Tricks window should be displayed	
1.05	Test the add trick button functions properly	This should load a pop-up to add a trick	Click the (+) icon at the top left corner of the application	Normal	A pop-up prompting you to add a trick should appear	

1.06	Test the Edit Trick button (pencil next to a trick) functions properly	This should load a pop-up to edit a trick	Click the pencil icon next to a trick	Normal	A pop-up prompting you to edit a trick should appear	
1.07	Test the Delete Trick button (bin next to a trick) functions properly	This should load a pop-up to delete a trick	Click the bin icon next to a trick	Normal	A pop-up should ask you whether you wish to delete that trick	
1.08	Test that the 'Skateparks' tab functions properly	This should load the skateparks window	Click the 'Skateparks' tab in the application	Normal	The Skateparks window should be displayed	
1.09	Test the Add Skatepark button functions properly	This should load a pop-up to add a skatepark	Click the (+) icon at the top left corner of the application	Normal	A pop-up prompting you to add a skatepark should appear	
1.10	Test the Skatepark Location button functions properly	This should load a pop-up giving details about the skatepark	Click a location on a map	Normal	A pop-up giving you information about a skatepark	

1.11	Test the Edit Skatepark button (pencil icon in the existing skatepark pop-up) functions properly	This should load a pop-up to edit a skatepark	Click the pencil icon in the existing skatepark pop-up	Normal	A pop-up prompting you to edit a skatepark should appear	
1.12	Test the Delete Skatepark button (bin icon in the existing skatepark pop-up) functions properly	This should load a pop-up to delete a skatepark	Click the bin icon in the existing skatepark pop-up	Normal	A pop-up prompting you to delete a skatepark should appear	
1.13	Test the 'Map Journey' button functions properly	This should map a route on the map from the start and finish location	Click the 'Map Journey' icon	Normal	A route will be displayed on the map	
1.14	Test that the 'Reviews' tab functions properly	This should load the reviews window	Click the 'Reviews' tab in the application	Normal	The Reviews window should be displayed	

1.15	Test the Add Review button functions properly	This should load a pop-up to add a review	Click the (+) icon at the top left corner of the application	Normal	A pop-up prompting you to add a review should appear		
1.16	Test the Edit Review button (pencil next to a review) functions properly	This should load a pop-up to edit a review	Click the pencil icon next to a review	Normal	A pop-up prompting you to edit a review should appear		
1.17	Test the Delete Trick button (bin next to a review) functions properly	This should load a pop-up to delete a review	Click the bin icon next to a review	Normal	A pop-up should ask you whether you wish to delete that review		
1.18	Test the Filter Type button functions properly	This could load a pop-up to filter the type	Click the 'Filter' button then from the list select 'Filter Type'	Normal	A pop-up should ask you to select a type		

1.19	Test the Filter Brand button functions properly	This sould load a pop-up to filter the brand	Click the 'Filter' button then from the list select 'Filter Brand'	Normal	A pop-up should ask you to select a brand	
1.20	Test the Filter Size button functions properly	This sould load a pop-up to filter the size	Click the 'Filter' button then from the list select 'Filter Size'	Normal	A pop-up should ask you to select a size	
2.00	Verify an appropriate name is entered to the 'Change Name' pop-out.	Should not accept the name if it is not valid	1.Ben 2.Keppie 3. 4.12345 5.Ben10	1.Normal 2.Normal 3.Erroneous 4.Erroneous 5.Erroneous	1.Accept 2.Accept 3.Error (Presence) 4.Error (Numbers) 5.Error (Numbers)	
2.01	Verify an appropriate picture is selected in the 'Change Picture' pop-out	Should only accept Trick.JPGs	1.Picture.JPG 2.Pic-ture.PNG 3.Picture.txt	G1.Normal 2.Erroneous 3.Erroneous	1.Accept 2.Error (File Type) 3.Error (File Type)	

2.02	Verify a valid email is entered to the 'Change Email' pop-out	Should only accept a correct email format	1.BenKeppie@hdmNormal2. 2.BenKep-pieEmail.com 3.Ji1290.co.uk	1.BenKeppie@hdmNormal12. 2.Erroneous 3.Erroneous	1. Accept 2. Error(Format) 3.Error(Format)		
2.03	Verify presence for adding a tricks name	Checks something is entered	1.Ollie 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.04	Verify presence for adding a trick description	Checks something is entered	1.Flips 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.04	Verify presence for adding a trick obstacle	Checks something is entered	1.Flat Ground 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.04	Verify presence for adding a trick tutorial link	Checks something is entered and that it is a website link	1.http://www.youtube.com/watch?v=1 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.05	Verify an appropriate picture is selected in the 'add a trick' pop-out	Should only accept Trick.JPGs	1.Picture.JPG 2.Pic-ture.PNG 3.Picture.txt	1.Normal 2.Erroneous 3.Erroneous	1.Accept 2.Error (File Type) 3.Error (File Type)		

2.06	Verify a difficulty is selected	Drop down box with 3 options	1.Easy 2.Medium 3.Hard 4.	1.Normal 2.Normal 3.Normal 4.Erroneous	1.Accept 2.Accept 3.Accept 4.Error(Presence)		
2.07	Verify the date is in the correct format	Format=DD/MM/YY/2014 2.10/12/2014 3/12/15/2014		1.Erroneous 2.Normal 3.Erroneous	1.Error(Format) 2.Accept 3.Error(Format)		
2.08	Verify presence for adding a skatepark name	Checks something is entered	1.Cambourne 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.09	Verify the correct format of coordinates are entered	Check that the coordinates are correct	1.52.2200,0.0700 2. 3.30480839	1.Normal 2.Erroneous 3.Erroneous	1.Accept 2.Error(Presence) 3.Error(Format)		
2.10	Verify presence for a skatepark description	Checks something is entered	1.Halfpipe only 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.11	Verify presence for a review description	Checks something is entered	1.Amazing 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		

2.12	Verify presence and correct number range	Checks something is entered and the values are between 1 and 5	1.3 2.0 3. 4.r	1.Normal 2.Boundary 3.Erroneous 4.Erroneous	1.Accept 2.Error(Range) 3.Error(Presence) 4.Error(Character)		
2.13	Verify a product brand is selected	Checks a value is selected	1.ZERO 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.14	Verify a product type is selected	Checks a value is selected	1.Trucks 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.15	Verify a product size is selected	Checks a value is selected	1. 5.0" 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.16	Verify a product name is selected	Checks a value is selected	1.SpecOps 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
3.00	Verify the first and last name are inputted into the database	The first and last name should be added to the database	1.FirstName 2.LastName	1.Normal 2.Normal	1.Accept 2.Accept		

3.01	Verify the profile picture is inputted into the database	A Trick.JPG should be added to the database	Trick.JPG	Normal	Accept		
3.02	Verify an email is inputted into the database	An email should be added to the database	BenKeppie@hotmail.uk	Normal	Accept		
3.03	Verify a trick name is inputted into the database	A trick name should be added to the database	Ollie	Normal	Accept		
3.04	Verify a trick description is inputted into the database	A trick description should be added to the database	Board Rotates 360	Normal	Accept		
3.05	Verify a trick obstacle is inputted into the database	A trick obstacle should be added to the database	Flat ground	Normal	Accept		
3.06	Verify a trick image is inputted into the database	A trick image should be added to the database	Trick.JPG	Normal	Accept		
3.07	Verify a trick tutorial link is inputted into the database	A trick tutorial link should be added to the database	www.youtube.com/watch?v=?	Normal	Accept		

3.08	Verify a trick difficulty is inputted into the database	A trick difficulty should be added to the database	Easy	Normal	Accept		
3.09	Verify a skatepark name is inputted into the database	A skatepark name should be added to the database	Cambourne Skatepark	Normal	Accept		
3.10	Verify skatepark coordinates are inputted into the database	Skatepark coordinates should be added to the database	52.2200,0.0700	Normal	Accept		
3.11	Verify a skatepark description is inputted into the database	A skatepark description should be added into the database	Half pipe	Normal	Accept		
3.12	Verify a review description is inputted into the database	A review description should be entered into the database	Amazing product	Normal	Accept		
3.13	Verify a product brand is inputted into the database	A product brand should be entered into the database	Product Brand (ZERO)	Normal	Accept		

3.14	Verify a product size is inputted into the database	A product size should be entered into the database	Product Size (5.0")	Normal	Accept		
3.15	Verify a product name is inputted into the database	A product name should be entered into the database	Product Name (Spec Ops)	Normal	Accept		
3.16	Verify a product type is inputted into the database	A product type should be entered into the database	Product Type (Truck)	Normal	Accept		
4.00	Verify that the product brand filter correctly returns the right reviews	Reviews with the product brand should be displayed	Select a brand filter (ZERO)	Normal	Only reviews that relate to the filter are displayed		
4.01	Verify that the product type filter correctly returns the right reviews	Reviews with the product type should be displayed	Select a type filter (Trucks)	Normal	Only reviews that relate to the filter are displayed		
4.02	Verify that the product size filter correctly returns the right reviews	Reviews with the product size should be displayed	Select a size filter (5.0")	Normal	Only reviews that relate to the filter are displayed		

4.03	Verify that the progress tracker returns the correct amount of completed tricks	Tricks which are completed will be displayed	Length of tricks completed	Normal	Only tricks that are completed will be displayed		
4.04	Verify that the progress tracker returns the correct amount of overall tricks	All tricks will be displayed	Length of tricks	Normal	All tricks will be displayed		
4.05	Verify that the skatepark is added to the correct location on the map	Longitude and latitude will correspond to map location	1.52.2200, 0.0700	Normal	Skatepark will be displayed on the map		
4.06	Verify that the progress tracker displayed the correct percentage	Completed tricks divided by all tricks multiplied by 100	Tricks	Normal	Correct percentage will be displayed		
4.07	Verify that the route is correct	A correct route should be displayed on the map	Start Location, End Location	Normal	A correct route is displayed		

5	Verify the program fulfills the specification	Run through the program, testing the different aspects to make sure they fit the objectives in the specification	Add some information to the program, start a student test, and view the results of the test	Normal	Program fulfills the specification		
---	-----------------------------------------------	------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------	--------	------------------------------------	--	--

3.1.4 Retained Items From Detailed Plan

122

Test Series	Purpose of Test	Test Description	Test Data	Test Data Type (Normal/ Erroneous/ Boundary)	Expected Result	Actual Result	Evidence
1.00	Test that the 'Profile' tab functions properly	This should load the profile window	Click the 'Profile' tab in the application	Normal	The profile window should be displayed		

1.03	Test the Change Picture button on the profile window functions properly	The default file browser for the system should open, allowing the user to select a Trick.JPG	click the 'Edit' button followed by the 'Change Picture' button	Normal	Default file browser should appear		
1.04	Test that the 'Tricks' tab functions properly	This should load the tricks window	Click the 'Tricks' tab in the application	Normal	The Tricks window should be displayed		
1.08	Test that the 'Skateparks' tab functions properly	This should load the skateparks window	Click the 'Skateparks' tab in the application	Normal	The Skateparks window should be displayed		
1.14	Test that the 'Reviews' tab functions properly	This should load the reviews window	Click the 'Reviews' tab in the application	Normal	The Reviews window should be displayed		
2.01	Verify an appropriate picture is selected in the 'Change Picture' pop-out	Should only accept Trick.JPGs	1.Picture.JPG 2.Picture.PNG 3.Picture.txt	G1.Normal 2.Erroneous 3.Erroneous	1.Accept 2.Error (File Type) 3.Error (File Type)		

2.03	Verify presence for adding a tricks name	Checks something is entered	1.Ollie 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.04	Verify presence for adding a trick description	Checks something is entered	1.Flips 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.04	Verify presence for adding a trick obstacle	Checks something is entered	1.Flat Ground 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.04	Verify presence for adding a trick tutorial link	Checks something is entered and that it is a website link	1.http://www.youtube.com/watch?v=1 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.05	Verify an appropriate picture is selected in the 'add a trick' pop-out	Should only accept Trick.JPGs	1.Picture.JPG 2.Picture.PNG 3.Picture.txt	1.Normal 2.Erroneous 3.Erroneous	1.Accept 2.Error (FileType) 3.Error (FileType)		
2.06	Verify a difficulty is selected	Drop down box with 3 options	1.Easy 2.Medium 3.Hard 4.	1.Normal 2.Normal 3.Normal 4.Erroneous	1.Accept 2.Accept 3.Accept 4.Error(Presence)		
2.08	Verify presence for adding a skatepark name	Checks something is entered	1.Cambourne 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		

2.10	Verify presence for a skatepark description	Checks something is entered	1.Halfpipe only 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.11	Verify presence for a review description	Checks something is entered	1.Amazing 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.12	Verify presence and correct number range	Checks something is entered and the values are between 1 and 5	1.3 2.0 3. 4.r	1.Normal 2.Boundary 3.Erroneous 4.Erroneous	1.Accept 2.Error(Range) 3.Error(Presence) 4.Error(Character)		
2.13	Verify a product brand is selected	Checks a value is selected	1.ZERO 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.14	Verify a product type is selected	Checks a value is selected	1.Trucks 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.15	Verify a product size is selected	Checks a value is selected	1. 5.0" 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		
2.16	Verify a product name is selected	Checks a value is selected	1.SpecOps 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)		

3.00	Verify the first and last name are inputted into the database	The first and last name should be added to the database	1.FirstName 2.LastName	1.Normal 2.Normal	1.Accept 2.Accept		
3.01	Verify the profile picture is inputted into the database	A Trick.JPG should be added to the database	Trick.JPG	Normal	Accept		
3.02	Verify an email is inputted into the database	An email should be added to the database	BenKeppie@hotmail.uk	Normal	Accept		
3.03	Verify a trick name is inputted into the database	A trick name should be added to the database	Ollie	Normal	Accept		
3.04	Verify a trick description is inputted into the database	A trick description should be added to the database	Board Rotates 360	Normal	Accept		
3.05	Verify a trick obstacle is inputted into the database	A trick obstacle should be added to the database	Flat ground	Normal	Accept		

3.06	Verify a trick image is inputted into the database	A trick image should be added to the database	Trick.JPG	Normal	Accept		
3.07	Verify a trick tutorial link is inputted into the database	A trick tutorial link should be added to the database	www.youtube.com/watch?v=?	Normal	Accept		
3.08	Verify a trick difficulty is inputted into the database	A trick difficulty should be added to the database	Easy	Normal	Accept		
3.09	Verify a skatepark name is inputted into the database	A skatepark name should be added to the database	Cambourne Skatepark	Normal	Accept		
3.10	Verify skatepark coordinates are inputted into the database	Skatepark coordinates should be added to the database	52.2200,0.0700	Normal	Accept		
3.11	Verify a skatepark description is inputted into the database	A skatepark description should be added into the database	Half pipe	Normal	Accept		

3.12	Verify a review description is inputted into the database	A review description should be entered into the database	Amazing product	Normal	Accept		
3.13	Verify a product brand is inputted into the database	A product brand should be entered into the database	Product Brand (ZERO)	Normal	Accept		
3.14	Verify a product size is inputted into the database	A product size should be entered into the database	Product Size (5.0")	Normal	Accept		
3.15	Verify a product name is inputted into the database	A product name should be entered into the database	Product Name (Spec Ops)	Normal	Accept		
3.16	Verify a product type is inputted into the database	A product type should be entered into the database	Product Type (Truck)	Normal	Accept		
4.05	Verify that the skatepark is added to the correct location on the map	Longitude and latitude will correspond to map location	1.52.2200, 0.0700	Normal	Skatepark will be displayed on the map		

5	Verify the program fulfills the specification	Run through the program, testing the different aspects to make sure they fit the objectives in the specification	Add some information to the program, start a student test, and view the results of the test	Normal	Program fulfils the specification		
---	-----------------------------------------------	------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------	--------	-----------------------------------	--	--

3.1.5 Changed Items From Detailed Plan

129

Test Series	Purpose of Test	Test Description	Test Data	Test Data Type (Normal/ Erroneous/ Boundary)	Expected Result	Actual Result	Evidence
1.01	Test the Change Name button on the profile window functions properly	The line edit will be available to edit and then once save is clicked, it will be read only	Click 'Edit' followed by 'Change Name', and then 'save'	Normal	The two name line edits should become available to edit		

1.02	Test the Change Email button on the profile window functions properly	The line edit will be available to edit and then once save is clicked, it will be read only	Click 'Edit' followed by 'Change Email', and then 'save'	Normal	The email line edit should be available to edit	
1.05	Test the add trick button functions properly	This should load a side form to add a trick	Click the add trick button at the top left corner of the application	Normal	A side form prompting you to add a trick should appear	
1.06	Test the Edit Trick function	CLI interface runs you through editing a selected trick	select edit trick in the CLI	Normal	The CLI will run through options to edit a selected trick	
1.07	Test the Delete process functions properly	Once a row is selected and the delete button is pressed the row should be deleted	Select a row, press delete and click save	Normal	A pop-up should ask you whether you wish to delete that trick and once save is clicked the row will be deleted	

1.09	Test the Add Skatepark button functions properly	This should load a side form to add a skatepark	Click the add skatepark button at the top left corner of the application	Normal	A side form prompting you to add a skatepark should appear		
1.10	Test the Skatepark Location process functions properly	This should load a pop-up giving details about the skatepark	Hover over a location on a map	Normal	A pop-up giving you information about a skatepark		
1.11	Test the Edit Skatepark process functions properly	CLI interface runs you through editing a selected skatepark	Select a skatepark to edit and enter new details	Normal	The CLI will run through options to edit a selected skatepark		
1.12	Test the Delete skatepark process functions properly	CLI interface runs you through deleting a selected skatepark	Select a skatepark to delete and confirm	Normal	The CLI will run through options to delete a selected skatepark		
1.15	Test the Add Review process functions properly	CLI interface runs you through adding a review	Run through the add skatepark CLI	Normal	The CLI will run through fields to add a new review		

1.16	Test the Edit Review process functions properly	CLI interface runs you through editing a review	Select a review to edit and enter new details	Normal	The CLI will run through options to edit a selected skatepark	
1.17	Test the Delete Review process functions properly	CLI interface runs you through deleting a review	Select a review to delete and confirm	Normal	The CLI will run through options to delete a selected skatepark	
2.00	Verify an appropriate name is entered to the 'Change Name' line edit.	Should not accept the name if it is not valid	1.Ben 2.Keppie 3. 4.12345 5.Ben10	1.Normal 2.Normal 3.Erroneous 4.Erroneous 5.Erroneous	1.Accept 2.Accept 3.Error (Presence) 4.Error (Numbers) 5.Error (Numbers)	
2.02	Verify a valid email is entered to the 'Change Email' line edit	Should only accept a correct email format	1.BenKeppie@hdtmNormal12. 2.BenKep- pieEmail.com 3.Ji1290.co.uk	1.Normal 2.Erroneous 3.Erroneous	1. Accept 2. Error(Format) 3.Error(Format)	

Justification for Changed Items

- Test 1.01 - I changed the details of the test as I have changed my user interface of my program to contain line edits which become read only and editable rather than a pop-out form that you fill in as this made the program more aesthetically pleasing.
- Test 1.02 - I changed the details of the test as I have changed my user interface of my program to contain line edits which become read only and editable rather than a pop-out form that you fill in as this made the program more aesthetically pleasing.
- Test 1.05 - I changed the details of this test as I have changed my user interface of my program to contain line edits in a side form which becomes available once the 'add trick' button is pressed. I felt this was more aesthetically pleasing than a pop-out.
- Test 1.06 - I changed the details of this test as I have have not implemented an edit trick functionality to my user interface, therefore I have used my old CLI program to make the changed to the database.
- Test 1.07 - I changed the details of this test as I have changed my user interface of my program to select a row and press delete to delete a trick.
- Test 1.09 - I changed the details of this test as I have changed my user interface of my program to contain line edits in a side form which becomes available once the 'add skatepark' button is pressed. I felt this was more aesthetically pleasing than a pop-out.
- Test 1.10 - I changed the details of this test as instead of clicking on the skatepark marker, all you need to do is hover over the marker to receive information about the skatepark.
- Test 1.11 - I changed the details of this test as I have have not implemented an edit skatepark functionality to my user interface, therefore I have used my old CLI program to make the changed to the database.
- Test 1.12 - I changed the details of this test as I have have not implemented an delete skatepark functionality to my user interface, therefore I have used my old CLI program to make the changed to the database.

- Test 1.15 - I changed the details of this test as I have have not implemented an add review functionality to my user interface, therefore I have used my old CLI program to make the changed to the database.
- Test 1.16 - I changed the details of this test as I have have not implemented an edit review functionality to my user interface, therefore I have used my old CLI program to make the changed to the database.
- Test 1.17 - I changed the details of this test as I have have not implemented an delete review functionality to my user interface, therefore I have used my old CLI program to make the changed to the database.
- Test 2.00 - I changed the details of the test as I have changed my user interface of my program to contain line edits which become read only and editable rather than a pop-out form that you fill in as this made the program more aesthetically pleasing.
- Test 2.02 - I changed the details of the test as I have changed my user interface of my program to contain line edits which become read only and editable rather than a pop-out form that you fill in as this made the program more aesthetically pleasing.

3.1.6 Removed Items From Detailed Plan

Test Se-ries	Purpose of Test	Test Description	Test Data	Test Data Type (Nor-mal/ Er-roneous/ Boundary)	Expected Result	Actual Re-sult	Evidence
1.13	Test the 'Map Journey' button functions properly	This should map a route on the map from the start and finish location	Click the 'Map Journey' icon	Normal	A route will be displayed on the map		

1.18	Test the Filter Type button functions properly	This should load a pop-up to filter the type	Click the 'Filter' button then from the list select 'Filter Type'	Normal	A pop-up should ask you to select a type		
1.19	Test the Filter Brand button functions properly	This should load a pop-up to filter the brand	Click the 'Filter' button then from the list select 'Filter Brand'	Normal	A pop-up should ask you to select a brand		
1.20	Test the Filter Size button functions properly	This should load a pop-up to filter the size	Click the 'Filter' button then from the list select 'Filter Size'	Normal	A pop-up should ask you to select a size		
2.07	Verify the date is in the correct format	Format = DD/MM/YYYY	1.1/2/2014 2.10/12/2014 3/12/15/2014	1.Erroneous 2.Normal 3.Erroneous	1.Error(Format) 2.Accept 3.Error(Format)		
2.09	Verify the correct format of coordinates are entered	Check that the coordinates are correct	1.52.2200,0.0700 2. 3.30480839	1.Normal 2.Erroneous 3.Erroneous	1.Accept 2.Error(Presence) 3.Error(Format)		

4.00	Verify that the product brand filter correctly returns the right reviews	Reviews with the product brand should be displayed	Select a brand filter (ZERO)	Normal	Only reviews that relate to the filter are displayed		
4.01	Verify that the product type filter correctly returns the right reviews	Reviews with the product type should be displayed	Select a type filter (Trucks)	Normal	Only reviews that relate to the filter are displayed		
4.02	Verify that the product size filter correctly returns the right reviews	Reviews with the product size should be displayed	Select a size filter (5.0")	Normal	Only reviews that relate to the filter are displayed		
4.03	Verify that the progress tracker returns the correct amount of completed tricks	Tricks which are completed will be displayed	Length of tricks completed	Normal	Only tricks that are completed will be displayed		
4.04	Verify that the progress tracker returns the correct amount of overall tricks	All tricks will be displayed	Length of tricks	Normal	All tricks will be displayed		

4.06	Verify that the progress tracker displayed the correct percentage	Completed tricks divided by all tricks multiplied by 100	Tricks	Normal	Correct percentage will be displayed		
4.07	Verify that the route is correct	A correct route should be displayed on the map	Start Location, End Location	Normal	A correct route is displayed		

Justification for Removed Items

- 137
- Test 1.13 - I have removed this test as this functionality is not present in my program.
 - Test 1.18 - I have removed this test as this functionality is not present in my program.
 - Test 1.19 - I have removed this test as this functionality is not present in my program.
 - Test 1.20 - I have removed this test as this functionality is not present in my program.
 - Test 2.07 - I have removed this test as this functionality is not present in my program.
 - Test 2.09 - I have removed this test as the coordinates are now entered automatically, corresponding to the users click on the Google map.
 - Test 4.00 - I have removed this test as this functionality is not present in my program.
 - Test 4.01 - I have removed this test as this functionality is not present in my program.
 - Test 4.02 - I have removed this test as this functionality is not present in my program.

- Test 4.03 - I have removed this test as this functionality is not present in my program.
- Test 4.04 - I have removed this test as this functionality is not present in my program.
- Test 4.06 - I have removed this test as this functionality is not present in my program.
- Test 4.07 - I have removed this test as this functionality is not present in my program.

3.2 Test Data

3.2.1 Original Test Data

Please see column 'Test Data' in subsection 'Original Detailed Plan' and for justifications see the text below each table.

3.2.2 Changes to Test Data

Please see column 'Test Data' in subsection 'Changed Items From Detailed Plan' and for justifications see the text below each table.

3.3 Annotated Samples

3.3.1 Actual Results

The table below contains my finalised test plan, including the retained and changed test series. In the 'actual results' column, the text in bold are tests that failed.

Test Series	Purpose of Test	Test Description	Test Data	Test Data Type (Normal/ Erroneous/ Boundary)	Expected Result	Actual Result	Evidence
1.00	Test that the 'Profile' tab functions properly	This should load the profile window	Click the 'Profile' tab in the application	Normal	The profile window should be displayed	The profile tab was displayed	Figure 3.1 on page 152
1.01	Test the Change Name button on the profile window functions properly	The line edit will be available to edit and then once save is clicked, it will be read only	Click 'Edit' followed by 'Change Name', and then 'save'	Normal	The two name line edits should become available to edit	The two name line edits became available to edit	
1.02	Test the Change Email button on the profile window functions properly	The line edit will be available to edit and then once save is clicked, it will be read only	Click 'Edit' followed by 'Change Email', and then 'save'	Normal	The email line edit should be available to edit	The email line edit became available to edit	

	1.03	Test the Change Picture button on the profile window functions properly	The default file browser for the system should open, allowing the user to select a Trick.JPG	click the 'Edit' button followed by the 'Change Picture' button	Normal	Default file browser should appear	The default file browser appeared allowing you to pick a file	
141	1.05	Test the add trick button functions properly	This should load a side form to add a trick	Click the add trick button at the top left corner of the application	Normal	A side form prompting you to add a trick should appear	A side form appeared on the left hand side prompting the user to add a trick	Figure 3.2 on page 153
	1.06	Test the Edit Trick function	CLI interface runs you through editing a selected trick	select edit trick in the CLI	Normal	The CLI will run through options to edit a selected trick	The CLI ran through a series of input statements to edit a trick	

1.07	Test the Delete process functions properly	Once a row is selected and the delete button is pressed the row should be deleted	Select a row, press delete and click save	Normal	A pop-up should ask you whether you wish to delete that trick and once save is clicked the row will be deleted	Row that was selected is deleted.	Figure 3.3 on page 154, Figure 3.4 on page 155
1.04	Test that the 'Tricks' tab functions properly	This should load the tricks window	Click the 'Tricks' tab in the application	Normal	The Tricks window should be displayed	Tricks window was displayed	
1.08	Test that the 'Skateparks' tab functions properly	This should load the skateparks window	Click the 'Skateparks' tab in the application	Normal	The Skateparks window should be displayed	The skateparks window was displayed	
1.09	Test the Add Skatepark button functions properly	This should load a side form to add a skatepark	Click the add skatepark button at the top left corner of the application	Normal	A side form prompting you to add a skatepark should appear	A side form appeared on the left hand side, prompting the user to add a skatepark	

1.10	Test the Skatepark Location process functions properly	This should load a pop-up giving details about the skatepark	Hover over a location on a map	Normal	A pop-up giving you information about a skatepark	An information window appeared giving information about that skatepark	
1.11	Test the Edit Skatepark process functions properly	CLI interface runs you through editing a selected skatepark	Select a skatepark to edit and enter new details	Normal	The CLI will run through options to edit a selected skatepark	The CLI ran through a series of input statements to edit a skatepark	
1.12	Test the Delete skatepark process functions properly	CLI interface runs you through deleting a selected skatepark	Select a skatepark to delete and confirm	Normal	The CLI will run through options to delete a selected skatepark	The CLI ran through a series of statements to delete a skatepark	Figure 3.5 on page 156
1.14	Test that the 'Reviews' tab functions properly	This should load the reviews window	Click the 'Reviews' tab in the application	Normal	The Reviews window should be displayed	The review window was displayed	

1.15	Test the Add Review process functions properly	CLI interface runs you through adding a review	Run through the add skatepark CLI	Normal	The CLI will run through fields to add a new review	The CLI ran through a series of input statements to add a review	
1.16	Test the Edit Review process functions properly	CLI interface runs you through editing a review	Select a review to edit and enter new details	Normal	The CLI will run through options to edit a selected skatepark	The CLI ran through a series of input statements to edit a review	
1.17	Test the Delete Review process functions properly	CLI interface runs you through deleting a review	Select a review to delete and confirm	Normal	The CLI will run through options to delete a selected skatepark	The CLI ran through a series of statements to edit a review	
2.00	Verify an appropriate name is entered to the 'Change Name' line edit.	Should not accept the name if it is not valid	1.Ben 2.Keppie 3. 4.12345 5.Ben10	1.Normal 2.Normal 3.Erroneous 4.Erroneous 5.Erroneous	1.Accept 2.Accept 3.Error (Presence) 4.Error (Numbers) 5.Error (Numbers)	1.Passed 2.Passed 3.Failed 4.Failed 5.Failed	Figure 3.6 on page 158 and Figure3.7 on page 159

2.01	Verify an appropriate picture is selected in the 'Change Picture' pop-out	Should only accept Trick.JPGs	1.Picture.JPG 2.Picture.PNG 3.Picture.txt	G1.Normal 2.Erroneous 3.Erroneous	1.Accept 2.Error (File Type) 3.Error (File Type)	1.Passed 2.Failed 3.Failed	
2.02	Verify a valid email is entered to the 'Change Email' line edit	Should only accept a correct email format	1.BenKeppie@hdtnNormal12. 2.BenKep- pieEmail.com 3.Ji1290.co.uk	N1Normal12. Erroneous 3. Erroneous	1. Accept 2. Error(Format) 3.Error(Format)	1.Passed 2.Failed 3.Failed	
2.03	Verify presence for adding a tricks name	Checks something is entered	1.Ollie 2.	1.Normal 2.Erroneous	1.Accept 2.Err- or(Presence)	1.Passed 2.Passed	Figure 3.8 on page 160, Figure 3.9 on page 161
2.04	Verify presence for adding a trick description	Checks something is entered	1.Flips 2.	1.Normal 2.Erroneous	1.Accept 2.Err- or(Presence)	1.Passed 2.Passed	
2.04	Verify presence for adding a trick obstacle	Checks something is entered	1.Flat Ground 2.	1.Normal 2.Erroneous	1.Accept 2.Err- or(Presence)	1.Passed 2.Passed	

2.04	Verify presence for adding a trick tutorial link	Checks a valid link is entered, and is allowed to be left empty	1.http://www.youtube.com/watch?V=1 2.http://www.google.com 3.	1.Normal 2.Erroneous	1.Accept 2.Error(Format) 3.	1.Passed 2.Passed 3.Passed		
2.05	Verify an appropriate picture is selected in the 'add a trick' pop-out	Should only accept Trick.JPGs	1.Picture.JPG 2.Picture.PNG 3.Picture.txt	G1.Normal 2.Erroneous 3.Erroneous	1.Accept 2.Error (FileType) 3.Error (FileType)	1.Passed 2.Failed 3.Failed		
2.06	Verify a difficulty is selected	Drop down box with 3 options	1.Easy 2.Medium 3.Hard 4.	1.Normal 2.Normal 3.Normal 4.Erroneous	1.Accept 2.Accept 3.Accept 4.Error(Presence)	1.Passed 2.Passed 3.Passed 4.Passed	Figure 3.10 on page 162, Figure 3.11 on page 163, Figure 3.12 on page 164	
2.08	Verify presence for adding a skatepark name	Checks something is entered	1.Cambourne 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)	1.Passed 2.Passed		
2.10	Verify presence for a skatepark description	Checks something is entered	1.Halfpipe only 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)	1.Passed 2.Passed		

2.11	Verify presence for a review description	Checks something is entered	1.Amazing 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)	1.Passed 2.Passed	
2.12	Verify presence and correct number range	Checks something is entered and the values are between 1 and 5	1.3 2.0 3. 4.r	1.Normal 2.Boundary 3.Erroneous 4.Erroneous	1.Accept 2.Error(Range) 3.Error(Presence) 4.Error(Character)	1.Passed 2.Passed 3.Passed 4.Passed	
2.13	Verify a product brand is selected	Checks a value is selected	1.ZERO 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)	1.Passed 2.Passed	
2.14	Verify a product type is selected	Checks a value is selected	1.Trucks 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)	1.Passed 2.Passed	
2.15	Verify a product size is selected	Checks a value is selected	1. 5.0" 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)	1.Passed 2.Passed	
2.16	Verify a product name is selected	Checks a value is selected	1.SpecOps 2.	1.Normal 2.Erroneous	1.Accept 2.Error(Presence)	1.Passed 2.Passed	
3.00	Verify the first and last name are inputted into the database	The first and last name should be added to the database	1.FirstName 2.LastName	1.Normal 2.Normal	1.Accept 2.Accept	1.Passed 2.Passed	Figure 3.13 on page 165

3.01	Verify the profile picture is inputted into the database	A Trick.JPG should be added to the database	ProfilePicture.JP	Normal	Accept	File path was added to the database	
3.02	Verify an email is inputted into the database	An email should be added to the database	BenKeppie@hotmail.uk	Normal	Accept	Email was added to the database	
3.03	Verify a trick name is inputted into the database	A trick name should be added to the database	Ollie	Normal	Accept	Trick name was added to the database	Figure 3.14 on page 166
3.04	Verify a trick description is inputted into the database	A trick description should be added to the database	Board Rotates 360	Normal	Accept	Trick description was added to the database	
3.05	Verify a trick obstacle is inputted into the database	A trick obstacle should be added to the database	Flat ground	Normal	Accept	Trick obstacle was added to the database	
3.06	Verify a trick image is inputted into the database	A trick image should be added to the database	TrickImage.JP	Normal	Accept	Trick image file path was added to the database	
3.07	Verify a trick tutorial link is inputted into the database	A trick tutorial link should be added to the database	www.youtube.com/watch?v=?	Normal	Accept	YouTube link was added to the database	

3.08	Verify a trick difficulty is inputted into the database	A trick difficulty should be added to the database	Easy	Normal	Accept	The trick difficulty was added to the database	
3.09	Verify a skatepark name is inputted into the database	A skatepark name should be added to the database	Cambourne Skatepark	Normal	Accept	The skatepark name was added to the database	Figure 3.15 on page 167
3.10	Verify skatepark coordinates are inputted into the database	Skatepark coordinates should be added to the database	52.2200,0.0700	Normal	Accept	The skatepark coordinates were added to the database	
3.11	Verify a skatepark description is inputted into the database	A skatepark description should be added into the database	Half pipe	Normal	Accept	The skatepark description was added to the database	
3.12	Verify a review description is inputted into the database	A review description should be entered into the database	Amazing product	Normal	Accept	Review description was added to the database	

3.13	Verify a product brand is inputted into the database	A product brand should be entered into the database	Product Brand (ZERO)	Normal	Accept	Product brand was added to the database	
3.14	Verify a product size is inputted into the database	A product size should be entered into the database	Product Size (5.0")	Normal	Accept	Product size was added to the database	
3.15	Verify a product name is inputted into the database	A product name should be entered into the database	Product Name (Spec Ops)	Normal	Accept	Product name was added to the database	
3.16	Verify a product type is inputted into the database	A product type should be entered into the database	Product Type (Truck)	Normal	Accept	Product type was added to the database	
4.05	Verify that the skatepark is added to the correct location on the map	Longitude and latitude will correspond to map location	1.52.2200, 0.0700	Normal	Skatepark will be displayed on the map	A google maps marker was placed correctly on the map	Figure 3.16 on page 168, Figure 3.17 on page 169

5	Verify the program fulfills the specification	Run through the program, testing the different aspects to make sure they fit the objectives in the specification	Ask the client if the program fulfills the specification, and pass all the test series in the detailed plan.	Normal	Program fulfills the specification	Program partially fulfills the specification, some areas do not work.	Please see all annotated samples below and Stuarts' email (Figure 3.18 on page 170).
---	-----------------------------------------------	------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------	--------	------------------------------------	------------------------------------------------------------------------------	--------------------------------------------------------------------------------------

3.3.2 Evidence

Test 1.00 Evidence

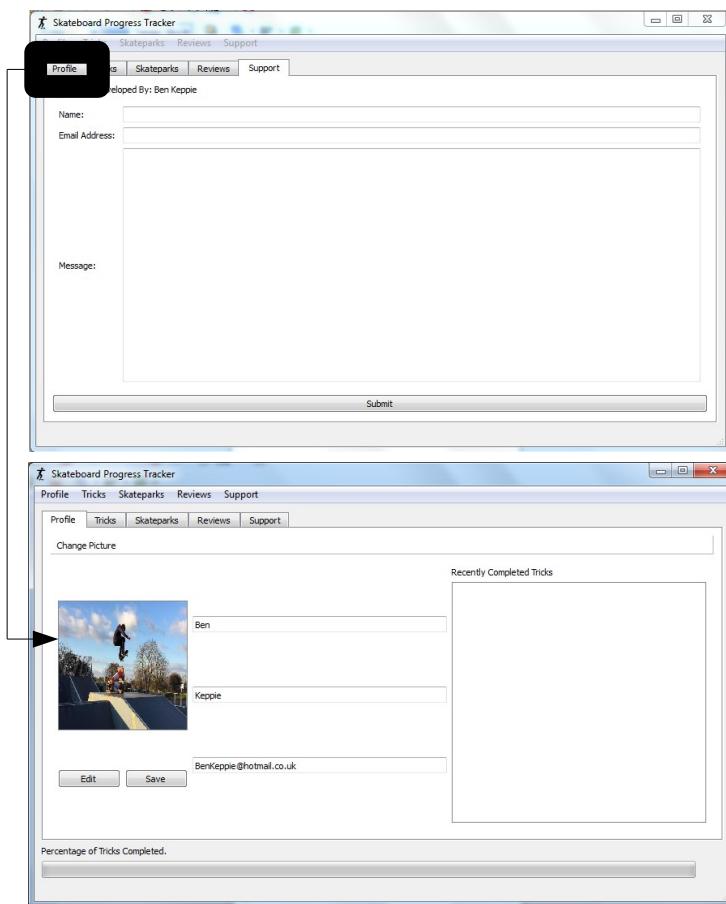


Figure 3.1: Evidence for Test 1.00

This test shows that when the 'profile' tab is clicked from a different tab, the profile window is displayed. This test was successful.

Test 1.05 Evidence

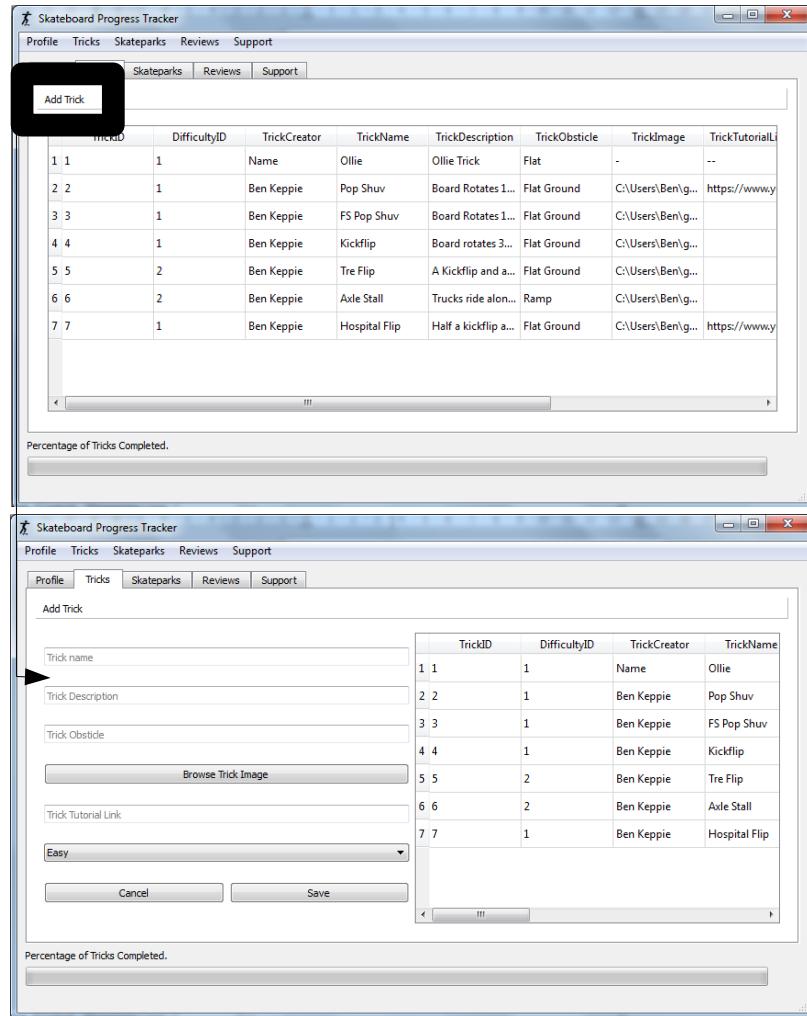


Figure 3.2: Evidence for Test 1.05

This test shows that when the 'add trick' button is pressed on the tool bar, the side form appears on the left hand side. This test was successful.

Test 1.07 Evidence

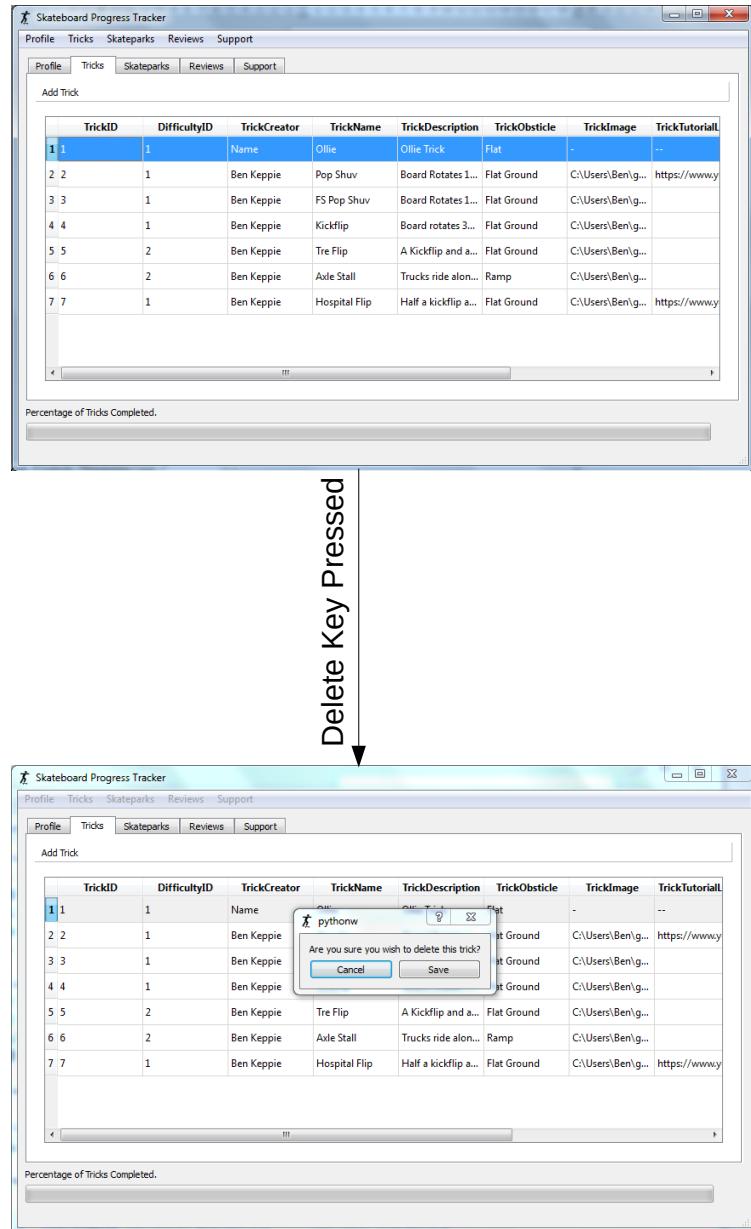


Figure 3.3: Evidence for Test 1.07
154

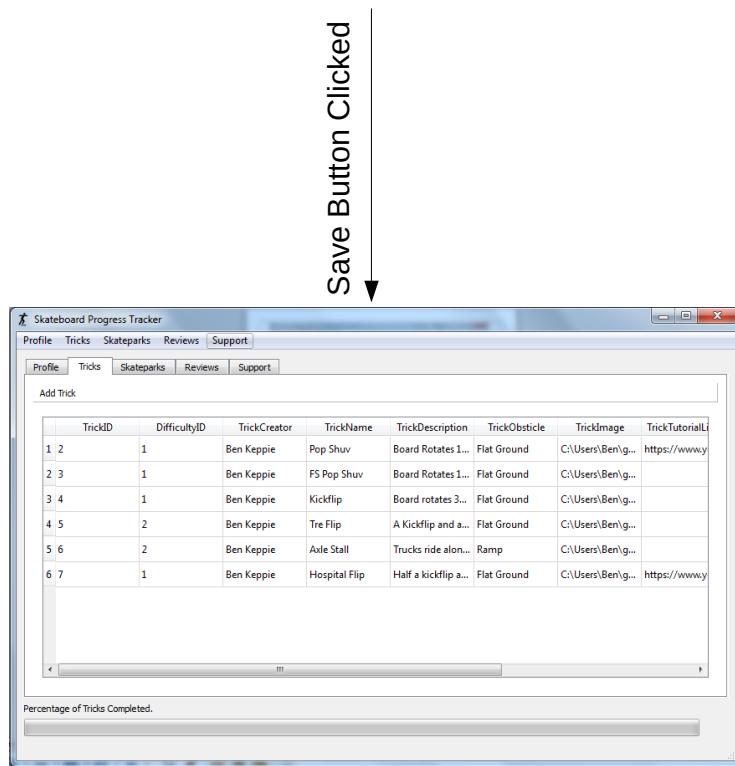


Figure 3.4: Evidence for Test 1.07 Part 2

This test shows that when a row is selected and the delete key is pressed a confirmation message is displayed asking if you wish to delete the selected trick and then if 'save' is clicked then the trick is deleted. This is shown by the table screen shot with the original selected row missing. This test was successful.

Test 1.12 Evidence

The screenshot shows the SQLite Inspector interface. The title bar says "SQLite Inspector". The main window has tabs: "Entity Descriptions", "Browse Data", and "Execute Query". The "Browse Data" tab is selected. A table named "Skatepark" is displayed with the following data:

SkateparkID	SkateparkName	SkateparkLongitude	SkateparkLatitude	SkateparkDescription
1	Cambourne	-0.0609773	52.2212	Cambourne Skatepark

Skateboard Progress Tracker Database Management

1. (Re)Create Database
2. Edit Profile Table
3. Edit Trick Table
4. Edit Skatepark Table
5. Edit Review Table
0. Exit

Please select an option: 4

Skatepark Table Management

1. Add a New Skatepark
2. Edit an Existing Skatepark
3. Delete an Existing Skatepark
0. Exit

Please select an option: 3

Please enter the SkateparkID of the skatepark you wish to delete: 1

Skatepark Successfully Deleted.

The screenshot shows the SQLite Inspector interface. The title bar says "SQLite Inspector". The main window has tabs: "Entity Descriptions", "Browse Data", and "Execute Query". The "Browse Data" tab is selected. A table named "Skatepark" is displayed with the following data:

SkateparkID	SkateparkName	SkateparkLongitude	SkateparkLatitude	SkateparkDescription

Figure 3.5: Evidence for Test 1.12

This test shows the command line interface process of deleting a skatepark within the skatepark table of the database. This test was successful.

Ben Keppie

Candidate No. 4609

Centre No. 22151

Test 2.00 Evidence

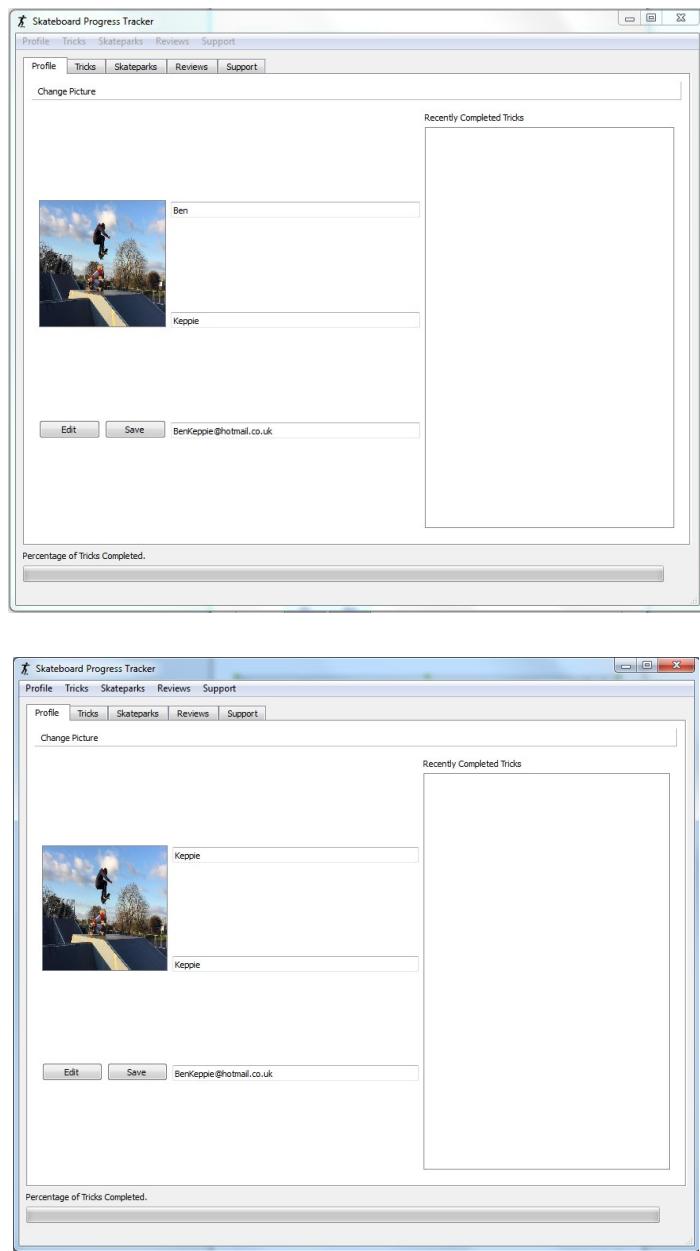


Figure 3.6: Evidence for Test 2.00 Part 1

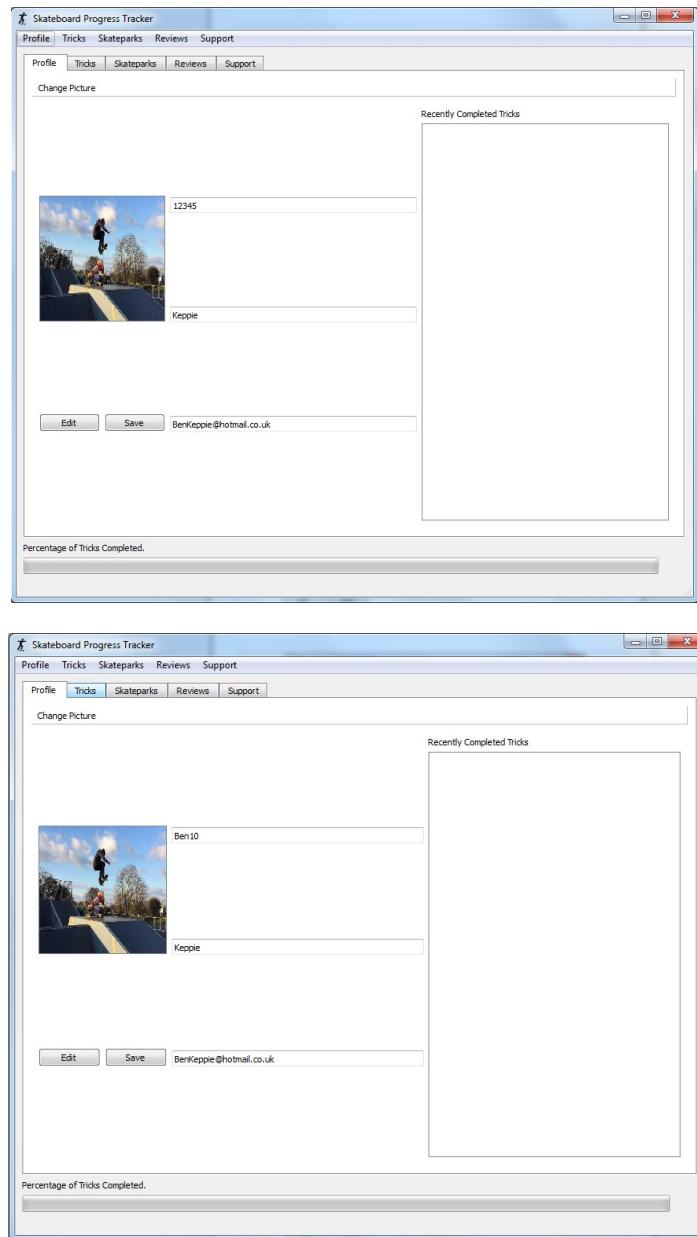


Figure 3.7: Evidence for Test 2.00 Part 2

This test shows how different names are accepted into the name line edits. Unfortunately the validation used was not present and therefore the erroneous values were accepted which means that this test failed.

Test 2.03 Evidence

TrickID	DifficultyID	TrickCreator	TrickName
1 1	1		Ollie
2 2	1	Ben Keppie	Pop Shuv
3 3	1	Ben Keppie	FS Pop Shuv
4 4	1	Ben Keppie	Kickflip
5 5	2	Ben Keppie	Tre Flip
6 6	2	Ben Keppie	Axle Stall
7 7	1	Ben Keppie	Hospital Flip

Save Button Clicked.

TrickID	DifficultyID	TrickCreator	TrickName	TrickDescription	TrickObstacle	TrickImage	TrickTutorialLink
1 1	1		Ollie	Ollie Trick	Flat	-	...
2 2	1	Ben Keppie	Pop Shuv	Board rotates ...	Flat Ground	C:\Users\Ben..._...	https://www.yo...
3 3	1	Ben Keppie	FS Pop Shuv	Board rotates ...	Flat Ground	C:\Users\Ben..._...	...
4 4	1	Ben Keppie	Kickflip	Board rotates ...	Flat Ground	C:\Users\Ben..._...	...
5 5	2	Ben Keppie	Tre Flip	A Kickflip and a...	Flat Ground	C:\Users\Ben..._...	...
6 6	2	Ben Keppie	Axle Stall	Tricks ride along...	Ramp	C:\Users\Ben..._...	...
7 7	1	Ben Keppie	Hospital Flip	Tricks ride along...	Flat Ground	C:\Users\Ben..._...	...
8 8	1	Ben Keppie	Ollie	Board lifts off t...	Flat Ground	U:\light CompAC...	...

Figure 3.8: Evidence for Test 2.03 Part 1

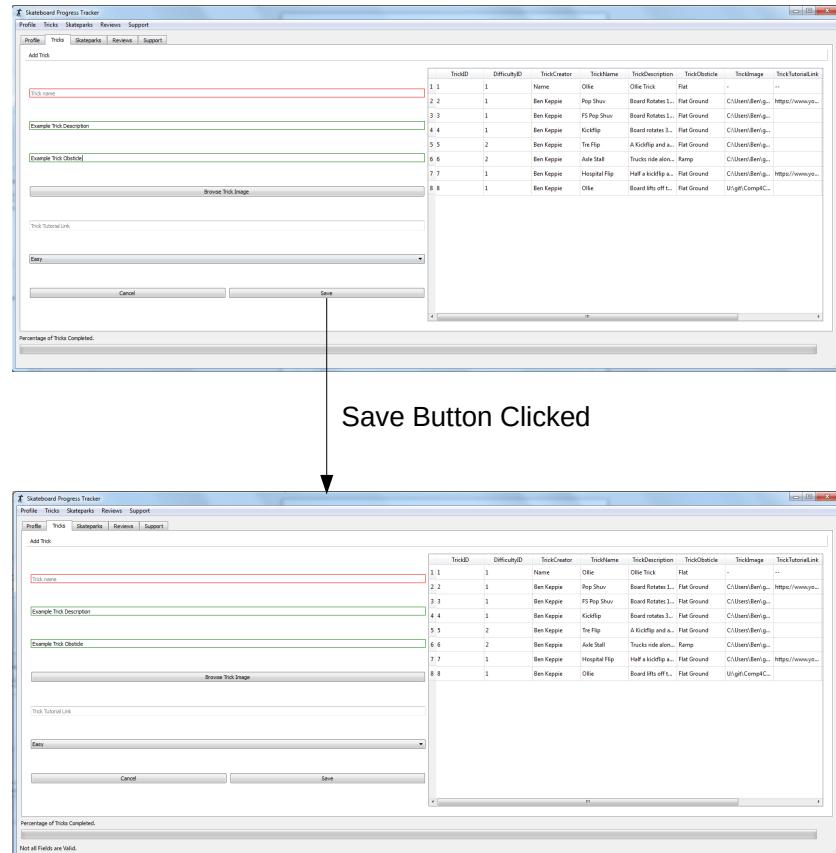


Figure 3.9: Evidence for Test 2.03 Part 2

The screen shots above show that when adding a trick, the trick gets successfully added to the database, therefore this test was successful.

Test 2.06 Evidence

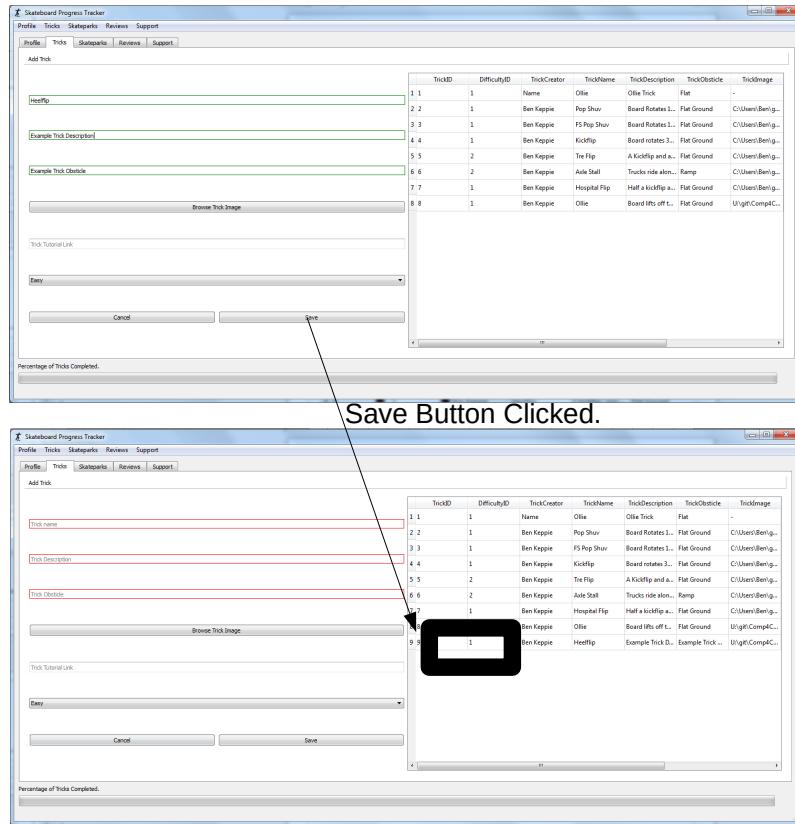


Figure 3.10: Evidence for Test 2.06 Part 1

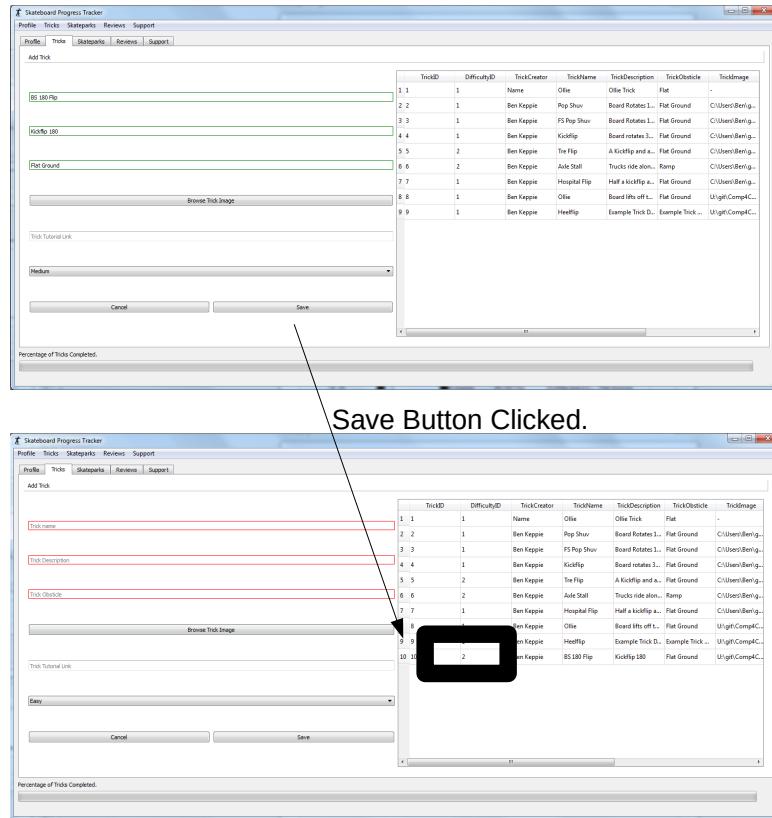


Figure 3.11: Evidence for Test 2.06 Part 2

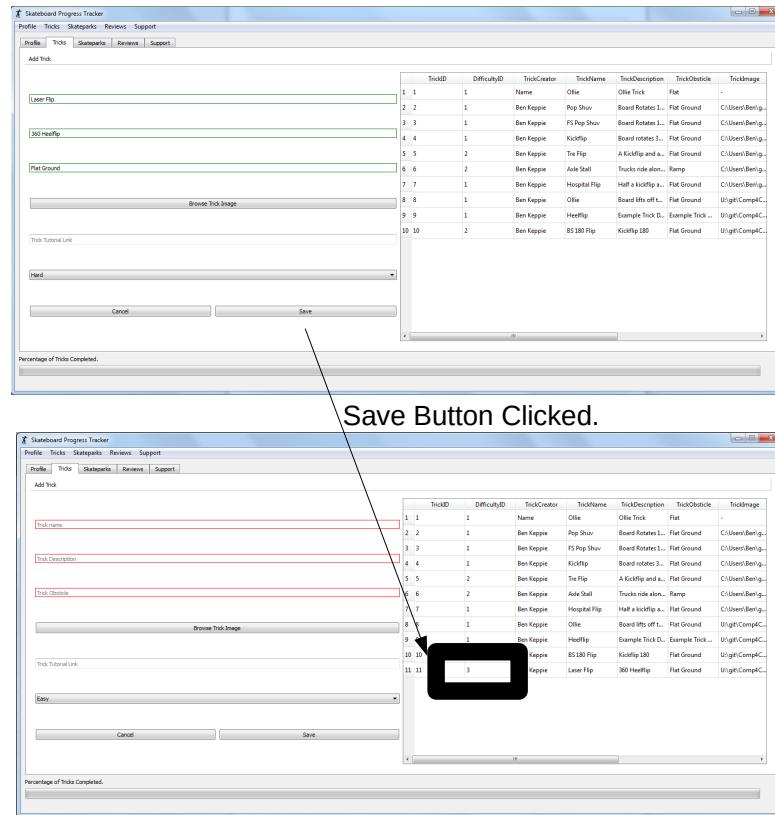


Figure 3.12: Evidence for Test 2.06 Part 3

The screen shots above show that the 'easy', 'medium' and 'hard' tricks have a corresponding integer value (1, 2 and 3 respectively) and when the trick is saved, the integer value is shown in the table. This test was therefore successful.

Test 3.00 Evidence

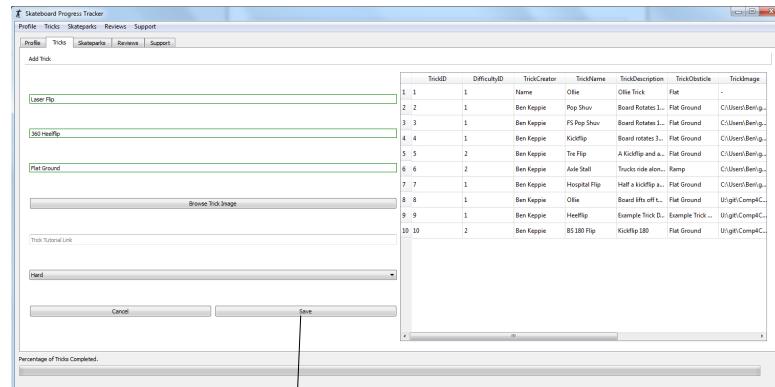
The screenshot shows two windows. The top window is titled 'Skateboard Progress Tracker' and has tabs for Profile, Tricks, Skateparks, Reviews, and Support. The Profile tab is selected. It displays a placeholder image of a skateboarder performing a trick, and fields for FirstName and LastName. Below these are 'Edit' and 'Save' buttons, and an email address 'BenKeppie@hotmail.co.uk'. To the right is a large empty area labeled 'Recently Completed Tricks'. The bottom window is titled 'User' and contains a table with columns: UserID, FirstName, LastName, UserPicture, and UserEmail. One row is visible, showing UserID 1, FirstName Ben, LastName Keppie, a placeholder image path C:\Users\Ben\g..., and the email BenKeppie@hotmail.co.uk.

UserID	FirstName	LastName	UserPicture	UserEmail
1	Ben	Keppie	C:\Users\Ben\g...	BenKeppie@ho...

Figure 3.13: Evidence for Test 3.00

The screen shot above shows that when a name is saved in the line edits on the 'profile' tab, the values are placed into the database. This test was successful.

Test 3.03 Evidence



Save Button Clicked.

TrickID	DifficultyID	TrickCreator	TrickName	TrickDescription	TrickObstacle	TrickImage	TrickTutorialLink
1 1	1	Ben Keppie	Ollie	Ollie Trick	Flat	-	..
2 2	1	Ben Keppie	Pop Shuv	Board Rotates 180 degrees	Flat Ground	C:\Users\Ben\g...	https://www.yo...
3 3	1	Ben Keppie	FS Pop Shuv	Board Rotates 180 degrees front side	Flat Ground	C:\Users\Ben\g...	
4 4	1	Ben Keppie	Kickflip	Board rotates 360 degrees on its x-axis	Flat Ground	C:\Users\Ben\g...	
5 5	2	Ben Keppie	Tre Flip	A Kickflip and a 360 shuv in a single motion	Flat Ground	C:\Users\Ben\g...	
6 6	2	Ben Keppie	Axle Stall	Trucks ride along the coping	Ramp	C:\Users\Ben\g...	
7 7	1	Ben Keppie	Hospital Flip	Half a kickflip and then a shuv	Flat Ground	C:\Users\Ben\g...	
8 8	1	Ben Keppie	Ollie	Board lifts off the ground.	Flat Ground	U:\git\Comp4C...	
9 9	1	Ben Keppie	Heelflip	Example Trick Description	Example Trick ...	U:\git\Comp4C...	
11 11	3	Ben Keppie	Laser Flip	360 Heelflip	Flat Ground	U:\git\Comp4C...	

Figure 3.14: Evidence for Test 3.03

The screen shot above shows that when a trick is saved, the values are placed into a database and this is shown by a status bar message that is displayed. This test was successful.

Test 3.09 Evidence

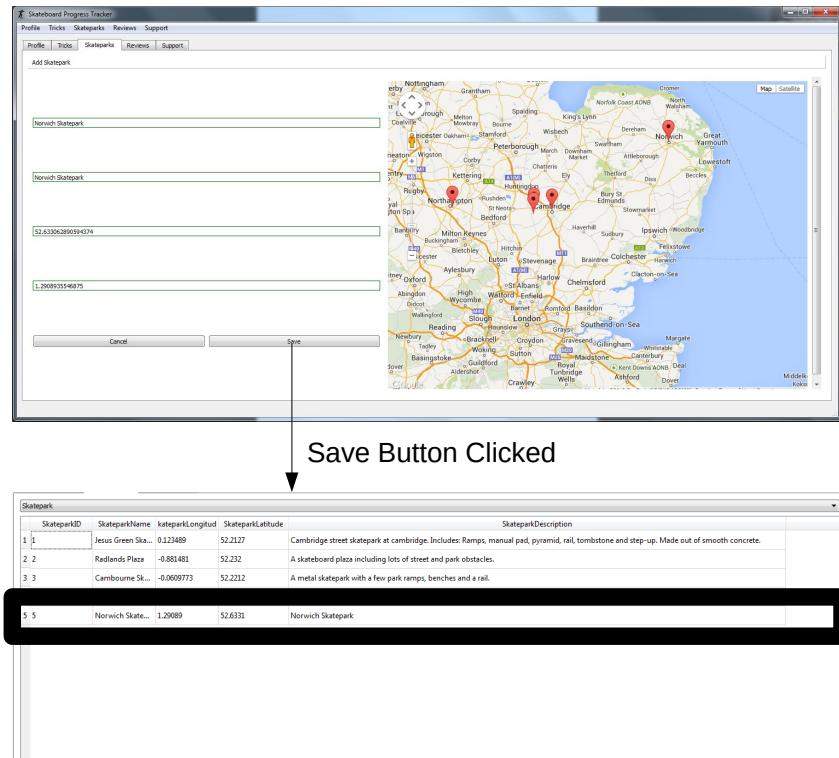


Figure 3.15: Evidence for Test 3.09

The screen shot above shows that when a skatepark is saved, the values are placed into a database and this is shown by a status bar message that is displayed. This test was successful.

Test 4.05 Evidence

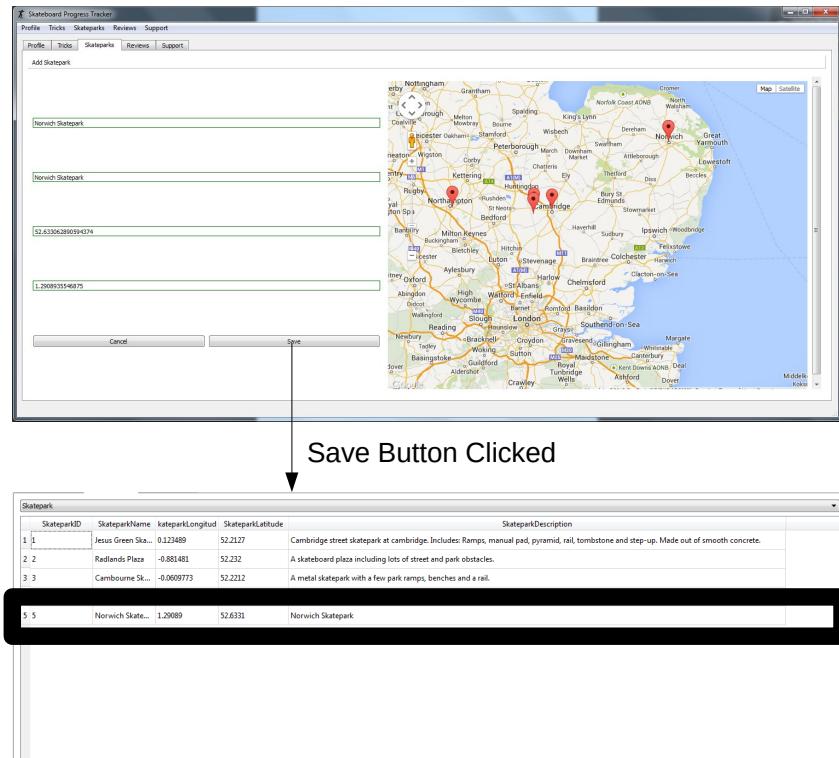


Figure 3.16: Evidence for Test 4.05 Part 1

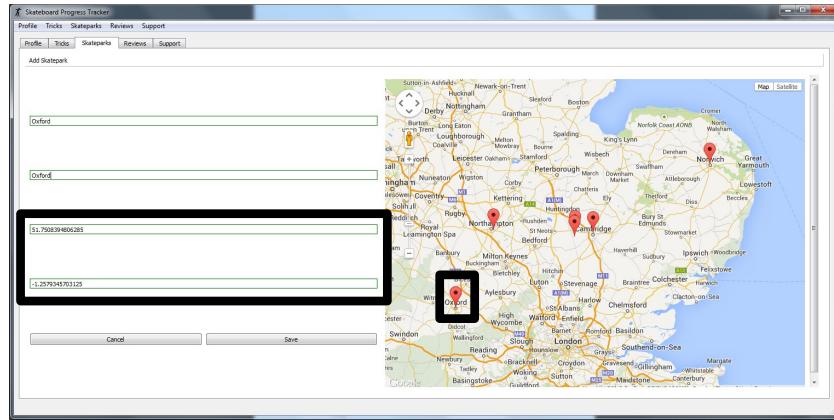


Figure 3.17: Evidence for Test 4.05 Part 2

The screen shot above shows that the values of the database for the skatepark correspond to the location of the marker on the google map image.

Test 5.00 Evidence

All of the previous annotated samples contribute to Test 5.00. There are elements of my program which work as intended e.g The name line edit did not contain the correct validation which lead to the failure of that test series (Figure 3.6 on page 158 and Figure 3.7 on page 159). On the other hand most of my tests passed e.g saving tricks (Figure 3.14 on page 166). This was also brought to my attention by my client as shown by an email conversation below.

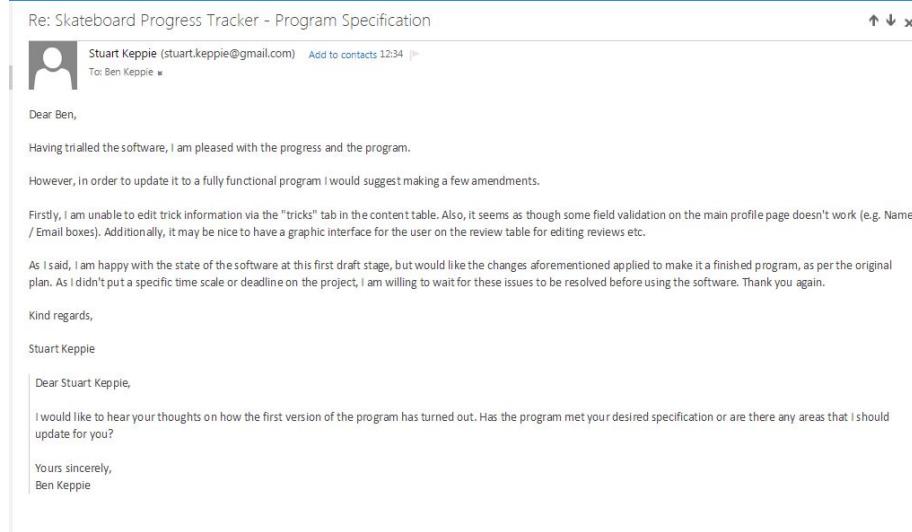


Figure 3.18: Evidence for Test 5.00

3.4 Evaluation

3.4.1 Approach to Testing

For each of my test series I used a different approach to testing. For my first test series I chose to use top-down testing as the flow of user interfaces was hierarchical. This was the best option as there are multiple interfaces which stem from the original interface. Top-down testing was the most suitable approach as I could test the integration between the different tabs in my user interface. For my second test series I chose bottom-up testing as I needed to test the lower levels of data input to ensure the information had been entered into the database. Following this, it allows me to test other areas of my program which use the information from the database which is needed in order to ensure that the program works correctly. For my third test series I chose to use white box testing as for the individual tests I have to look inside the database after inputting information into the program, which then adds the data to the database. This was the most appropriate approach as it's the only way to ensure that the fundamental programming aspects of my program actually carry out the correct function. For my fourth test series I chose black box testing as I was checking to see if algorithms returned the correct value without looking at the internal structure of the code, this is important as I needed to make sure that the algorithms in place worked from a working aspect rather than dry-running the algorithms and therefore allowing for assumptions and human-errors to be

made. Finally, for my fifth test series I chose acceptance testing as this is conducted to determine if the specification is met. This is the most important test as my program is being built for my client and therefore in order to pass this test I needed my clients approval.

3.4.2 Problems Encountered

Testing my program allowed me to identify areas of the system which did not work as intended. These tests are identified below, with an explanation. I will endeavour to fix all of the errors for the final release of my program which I will give to my client. You can view my full testing table with results, this can be found as Figure 3.5 on page 151.

Test 2.00 and Test 2.02

The line edits in the 'profile' tab which allow you to edit your first name, last name and email did not include any validation on them. This is a minor issue that could easily be fixed by a short validation method.

Test 2.01 and Test 2.05

When uploading pictures for the program, the file type accepted was supposed to be limited to a .jpeg file; however the program accepted any file type. This is a minor problem, but can be annoying as file types that aren't .jpeg will not be displayed. For example, if a .txt file is uploaded for the profile picture, the profile picture will appear to be blank. This could easily be fixed by a short validation method and then a message being displayed on the status bar.

Test 5.00

The build up of minor errors, along with the fact that the graphical user interface is not complete within the reviews tab, has lead to the failure of this test. For this test to pass the whole program would have to be completed at a usable graphical user interface level, along with every test series passing. This means I am not that far off passing this test as all that needs to be done is a few validation methods in certain input areas and the review table input form functionality.

3.4.3 Strengths of Testing

I feel that my testing methods were particularly strong. This was partnered with the large amount of individual tests in each test series to show which parts of my program worked, and which parts didn't. The use of multiple different testing types allowed for my system to be tested in many different aspects which then gave a rigorous analysis of the functionality of my program. My testing allowed me to see if the functions made the correct changes to the database and user interface.

3.4.4 Weaknesses of Testing

The weakness of my testing is the fact that not every single aspect of my program was identified, therefore there could be some areas of my program which have errors in, of which I do not know about. My testing also doesn't look at the internal structure of the code. This means that there could be parts of my code which are inefficient and therefore could be coded in a much more efficient way which would lead to less processing power needing to be used as well as a faster program.

3.4.5 Reliability of Application

The reliability of my program is questionable. It carries out most of the initial functions that I set for it to do; however some key features are missing and my testing has highlighted those areas. With a few minor tweaks, these issues would be rectified. The two main problems with the reliability of my program lie within the validation of some fields (for example, test series 2.00, 2.01, and 2.02 which can be found on table 3.5 on page 151 and the mixed program usage (graphical user interface and command line interface). As I didn't have time to complete the command line interface, some of the functionality (the review tab) is only available to use in a command line interface as outlined by tests 2.11-2.16 on table 3.5 on page 151. This is not a problem within the functionality, but for my client, this form of information editing is not acceptable. None of the image parts of my program validate the file type which is a key contributing factor to the decreases reliability of my program. Looking back on my program I should have changed some of the entry field to fixed combo boxes as the some of the information that can be entered into the database could be inaccurate, and therefore my program is only as reliable if the data input is accurate.

3.4.6 Robustness of Application

Even though my application failed a few of its test series, for example test series 2.00, 2.01, and 2.02 which can be found on table 3.5 on page 151. I would still deem my program robust. Regardless of whether parts of the program didn't work, at no time did this cause the program to crash, lose any data or start an infinite loop which would leave the program unable to use. With some of my input fields, even if data is not designed to go into an input field, an error message is displayed and the program continues as normal, for example test series 2.03 in table 3.5 on page 151. This is a good quality of my program as the support section will allow for users to report errors that happen as the program doesn't crash due to the errors. This will then allow for me to identify the error, fix it and send out a new release of the program.

Chapter 4

System Maintenance

4.1 Environment

4.1.1 Software

During the implementation of my program I used a variety of software items to help me create the program for my client. The software I used is detailed in the bullet points below.

- Python 3.4
- IDLE (python GUI)
- PyQt 4
- SQLite3
- smtplib
- pdb
- SQLite Inspector
- Notepad ++ v6.6.7
- Google Chrome

4.1.2 Usage Explanation

The table below gives details of why I decided to use the software I used.

Software	Justification for Use
Python 3.4	Python is the programming language I am most confident with as I have been learning it through the past two years at college. Python is also the most supported program at my college.
IDLE (python GUI)	This programming editor comes with the free installation of Python, and is the only programming environment available at my college.
PyQt 4	This software is an add on to the Python programming language which allowed for me to create a graphical user interface for my program.
SQLite 3	This software came along with the Python 3.4 library and I also had some previous experience of using it, therefore I used it to handle my SQL queries.
smtplib	This module came along with the Python 3.4 library and allowed me to send emails to me (the developer) about any bugs in the program.
pdb	This module came along with the Python 3.4 library and was readily available to be imported
SQLite Inspector	This piece of software allowed me to look inside my database and test SQL queries. This was available for me to use at college and home as my teacher created it.
Notepad ++ v6.6.7	This piece of software allowed me to test my JavaScript code and allowed me to debug any formatting errors as JavaScript isn't formatted nicely in IDLE.
Google Chrome	I am familiar with using this web browser and it also has plenty of compatibility with lots of programming languages therefore I used this to view my Javascript script.

Another reason for using all the programs I have used is the fact that they are free to download from the internet, which therefore creates a free application for my client to use.

4.1.3 Features Used

Software	Features Used
Python 3.4	I used python to run my program which allowed me to test the graphical user interface of my program. I also used the code libraries which came with the installation of Python to code my system.
IDLE (python GUI)	I used IDLE to write out my code and save it as a python file. I took advantage of the colour coded syntax and also the code predictor. I also used IDLE to run the python file.
PyQt 4	I used PyQt extensively, from using pre-programmed classes such as QVBoxLayout to rewriting some classes such as the QWebPage class. PyQt was used to create the graphical user interface of my program.
SQLite 3	I used this piece of software to write SQL queries that would allow me to add, edit, delete and retrieve data from my database and ensure referential integrity was enforced.
smtplib	I used the email sending capabilities of this module.
pbd	I used this module as an interactive debugger to help debug my code
SQLite Inspector	I used SQLite inspector for two functions. First of all I used it to check that data had been added/edited/deleted properly. Secondly I used it to check that my SQL statements were correct.
Notepad ++ v6.6.7	I used this piece of software to debug and write the Javascript for my google maps integration found in the 'skatepark' tab of my program.
Google Chrome	I used Google Chrome to check that my Javascript code functioned properly inside a web browser. I also took advantage of the 'developer' features of google chrome to debug my Javascript.

4.2 System Overview

4.2.1 General User Interface

On every part of my user interface a QMenuBar is at the top, allowing you to access functionality of any tab from anywhere in the program, for example if you are on the profile tab you can click on the 'support' part of the QMenuBar and

click 'contact support' on the drop down options and the view of the program will change to the support tab and load the correct widgets to allow for the user to contact support. A QStatusBar is also available on every page which displays messages at appropriate times, informing the user about changes that have occurred.

4.2.2 Profile Tab User Interface

The profile tab consists of a QToolBar at the top of the tab labeled 'Change Picture' widget allowing you to change your profile picture which is displayed in a QGraphicsScene. Below the profile picture there are 2 QPushButtons labelled 'Edit' and 'Save'. To the right of this there are three QLineEdit's showing the users first name, last name and email address, to the right of these QLineEdit's is a Recently completed tricks list. Below the tabbed interface a QProgressBar which shows the percentage of completed tricks.

4.2.3 Editing Profile Table Information

Once the edit button is clicked the QLineEdit's containing the first name, last name and email address of the user become available to edit. Once the QLineEdit's have been changed you may click the 'save' button to save the changes. Once the 'Change Picture' button is pressed a QFileDialog appears allowing you to choose an image from your documents to set as your profile picture.

4.2.4 Tricks Tab User Interface

The tricks tab also contains the QProgressBar below the tabbed interface. There is also a QToolBar with an option to add a trick. Below the QToolBar a QTableView displays all of the items in the tricks table of the database. Once the 'Add Trick' button is pressed a side form appears on the left hand side with QLineEdit's, QPushButtons and QComboBoxes which allow you to fill in information about a trick.

4.2.5 Editing Trick Table Information

Once the 'Add Trick' button has been pressed you can fill in information about a trick, once the 'save' button below the form has been pressed the trick will be saved to the database if all the fields are valid. If a field is invalid then the invalid fields will be highlighted red. To delete a trick you select the row you wish to delete and then press the delete key, a confirmation message will appear and you click the 'save' button to accept the delete. To edit a trick you have to run through the CLI menu and then run through the appropriate steps.

4.2.6 Skateparks Tab User Interface

The Skateparks tab interface is similar to that of the Tricks tab, but the table is replaced with a QWebView of the Google map.

4.2.7 Editing Skatepark Table Information

To add a skatepark you click on the Google Maps object, the program will then automatically fill in the latitude and longitude of the marker. Then you need to fill in the skatepark name and description. Then click save to save the skatepark to the database. To edit or delete a skatepark you have to run through the CLI menu and run through the appropriate steps

4.2.8 Reviews Tab User Interface

The Review user interface is similar to the tricks tab. But the table is replaced with information about reviews.

4.2.9 Editing Review Table Information

To edit the review table you have to run through the CLI menu and run through the appropriate steps.

4.2.10 Support Tab User Interface

The support tab consists of a QLabel containing my details as the application developer and a series of QLineEdit's allowing the user to enter information to send a bug. A 'submit' QPushButton is below this form.

4.2.11 Reporting a Bug

A series of QLineEdit's must be filled in, in order to send a message about a bug in the program. This includes: Users name and email address, as well as the actual message saying what the bug is.

4.3 Code Structure

My general code was structured around a graphical user interface where I have incorporated PyQt functions and object orientated programming concepts which

I have developed over the past two years of learning python. A sample of coding structures are shown below.

4.3.1 Main Window

The full main window code can be found in subsection 4.11.1.

```
1  class MainWindow(QMainWindow):
2      """Class for the main window of my program"""
3      def __init__(self):
4          super().__init__()
5          self.setWindowTitle("Skateboard Progress
6              Tracker")
7          self.app = QtGui.QApplication([])
8          self.set_icon()
9          self.main_VBoxLayout=QVBoxLayout()
10         self.central_widget=QWidget()
11         self.ProgressBar=QProgressBar()
12         self.ProgressBarLabel=QLabel("Percentage of
13             Tricks Completed.")
14
15         #create tabs
16         self.create_tabs()
17
18         #Create Menu Bar
19         self.Menu=Menu(self)
20         self.setMenuBar(self.Menu.MenuBar)
21
22         #Create Statusbar
23         self.StatusBar=QStatusBar()
24         self.setStatusBar(self.StatusBar)
25
26         self.tabs.currentChanged.connect(self.progress_bar_hide)
27
28     def progress_bar_hide(self):
29         if (self.tabs.currentIndex() >=2):
30             self.ProgressBarLabel.hide()
31             self.ProgressBar.hide()
32         else:
33             self.ProgressBar.show()
34             self.ProgressBarLabel.show()
35
36
```

```
37
38
39
40     def set_icon(self):
41         self.app_icon=QtGui.QIcon()
42         self.app_icon.addFile("ProgramIcon.png",QSize(16,16))
43         self.app.setWindowIcon(self.app_icon)
44
45
46
47
48     def create_tabs(self):
49
50
51         self.tabs=QTabWidget()
52
53         #Create Widgets
54         self.profile_tab=DisplayProfileWidget(self)
55
56         self.tricks_tab=DisplayTricksWidget(self)
57         self.skateparks_tab=DisplaySkateparksWidget(self)
58         self.reviews_tab=DisplayReviewsWidget(self)
59         self.support_tab=DisplaySupportWidget(self)
60
61
62         #Add Tabs
63         self.tabs.addTab(self.profile_tab, "Profile")
64         self.tabs.addTab(self.tricks_tab, "Tricks")
65         self.tabs.addTab(self.skateparks_tab,
66                         "Skateparks")
67         self.tabs.addTab(self.reviews_tab, "Reviews")
68         self.tabs.addTab(self.support_tab, "Support")
69
70         self.main_VBoxLayout.addWidget(self.tabs)
71         self.main_VBoxLayout.addWidget(self.ProgressBarLabel)
72         self.main_VBoxLayout.addWidget(self.ProgressBar)
73         self.central_widget.setLayout(self.main_VBoxLayout)
```

My main window sets up the basis for my application. I have structured it to be a class which means that it can be used and edited easily with broken up methods. My code structure splits up every different function of the main window into each method which allows for easy debugging and the ability to individually operate certain functions. My class extends QMainWindow and calls the super class to get all of the functionality of the PyQt object 'QMainWindow'.

4.3.2 Tabs

The code for the tabbed interface of my program can be found in subsection 4.11.1, line numbers 77-112. And subsection 4.11.2.

```

1 class CustomQTabWidget(QTabWidget):
2     """A class for my custom QTabWidget"""
3     def __init__(self, parent):
4         super().__init__()
5         self.parent=parent
6
7     def currentChanged(self):
8         print("Change in tab")

```

I created my own custom QTabWidget which allowed me to easily debug the problems which I was having in the early stages of programming my program. I pass 'parent' into the class so that when the CustomQTabWidget is instantiated I could call attributes and methods from the main window to aid the debugging of my program.

4.3.3 Menu Bar

The code for my whole menu bar can be found in subsection 4.11.3.

My menu bar allowed for a shortcut to any functionality in the system from any page.

4.3.4 Tool Bar

My code for all the toolbars can be found in subsections: 4.11.6, 4.11.9, 4.11.13 and 4.11.16.

```

1 class DisplayTricksToolbar(QToolBar):
2     def __init__(self, parent):
3         super().__init__()
4         self.parent=parent
5
6         self.add_trick= QAction("Add Trick", self)
7
8         self.addAction(self.add_trick)
9
10        #connections
11        self.add_trick.triggered.connect(self.add_trick_connection)
12
13

```

```

14     def add_trick_connection(self):
15         self.parent.add_trick_stacked()

```

The code shown above is the code for my tricks toolbar. The structured approach I had to creating a toolbar was to first create an action, and then link that action to a connection which would carry out a particular function. In this case, clicking the 'add trick' action would display the 'add trick' stacked layout of the tricks tab. All my toolbar files took the same structured approach to allow for a universal, and easy to replicate piece of code.

4.3.5 SQL Connections

My code for all the SQL connections can be found in subsections: 4.11.7, 4.11.10, 4.11.14 and 4.11.17.

```

1     def show_all_tricks(self):
2         query = QSqlQuery()
3         query.prepare(""" SELECT * FROM Trick""")
4         query.exec_()
5         return query

```

The SQL method shown above is taken from the tricks SQL connection code. This code shows the structured process for preparing a query and passing the executed query back down to the widget, therefore allowing for the query results to be displayed in a QTableView.

4.3.6 Validation

My code for validation can be found in subsection 4.11.8, line numbers 102 - 169. And subsection 4.11.11, line numbers 50-87.

```

1     def validate_trick_name(self):
2         Text=self.trick_name.text()
3         TrickNameExpression=re.compile("^(?!\\s*$).+")
4         Match=TrickNameExpression.match(Text.upper())
5         if Match:
6             self.trick_name.setStyleSheet(self.GreenBorder)
7             return True
8         else:
9             self.trick_name.setStyleSheet(self.RedBorder)
10            return False

```

This validation method is taken from the tricks widget (subsection 4.11.8). This code shows the structured approach I took to validating each field. First I

gathered the text to be validated from the appropriate widget and then compiled a regular expression to compare it to. If the text fit the regular expression the boolean value True was returned and the field was coloured green. If the text did not fit the regular expression the boolean value False was returned and the field was coloured red.

4.3.7 Main

My code for every main module can be found on every subsection of the Code Listing section, at the bottom of each pieces of code.

```

1 def main():
2     application=QApplication(sys.argv)
3     window=MainWindow()
4     #splash_screen()
5
6     window.show()
7     window.raise_()
8     application.exec_()
9     print()
10
11
12
13
14 if __name__=="__main__":
15     main()

```

This piece of code shows the main module for my main window. This is an example of the IF statement which is run at the start of almost every module. This allows the module to be run individually which is extremely useful for implementation and testing purposes. It is also used to launch the application once the main window file is run.

4.4 Variable Listing

My data dictionary can be found on Figure 2.6.4 on page 76. My other variables that I implemented in my program can be found in the table below.

Variable	Purpose	Reference
splash_pix	Stores a QPixmap image of the programs splash screen.	Subsection 4.10.1, line number 115

replace	A string to replace another variables string	Subsection 4.10.3, line numbers: 78,79,80. Subsection 4.10.6, line numbers: 35,36,37. Subsection 4.10.8, line numbers: 302,303
FirstName	A variable to temporarily hold the first name of the user	Subsection 4.10.7, line numbers:17,19,22.
LastName	A variable to temporarily hold the last name of the user	Subsection 4.10.7, line numbers: 18,19,22.
Email	A variable to temporarily hold the email address of the user	Subsection 4.10.7, line numbers: 32,36.
cursor	a variable to act as the database cursor	subsection 4.10.7, line numbers: 24, 38, 52, 61, 68, 74.
self.RedBorder	A style sheet used to give a QWidget a red border	Subsection 4.10.8, line numbers: 15, 133, 146, 157.
self.GreenBorder	A style sheet used to give a QWidget a green border	Subsection 4.10.8, line numbers: 16, 130, 143, 154.
query	holds a QSqlQuery object	Subsection 4.10.8, line numbers: 94, 95, 116.
Text	Holds the text of a QLineEdit whilst the data entry is being validated	Subsection 4.10.8, line numbers: 126, 128, 139, 141, 150, 152. Subsection 4.10.11, line numbers: 62, 64, 72, 73, 82, 83.
TrickNameExpression	Holds the compiled regular expression for the trick name	Subsection 4.10.8, line numbers:127, 128.
TrickDescriptionExpression	Holds the compiled regular expression for the trick description	Subsection 4.10.8, line numbers: 140, 141
TrickObstacleExpression	Holds the compiled regular expression for the trick obstacle	Subsection 4.10.8, line numbers: 151, 152.
TrickTutorialExpression	Holds the compiled regular expression for the trick tutorial link	Subsection 4.10.8, line numbers: 162, 163.
SkateparkNameExpression	Holds the compiled regular expression for the skatepark name	Subsection 4.10.11, line numbers: 63, 64.
SkateparkDescriptionExpression	Holds the compiled regular expression for the skatepark description	Subsection 4.10.11, line numbers: 52 ,53.

Match	Matches the Text of a QLineEdit against the appropriate regular expression	Subsection 4.10.8, line numbers: 128, 141, 152, 163. Subsection 4.10.11, line numbers: 53, 54, 64, 65.
sql	holds the sql text for sqlite3 queries	Subsection 4.10.10, line numbers: 15, 43. Subsection 4.10.12, line numbers: 78. Subsection 4.10.14, line numbers: 13.
self.LastMarker	Holds the coordinates of the last marker placed on the google map	Subsection 4.10.12, line numbers: 55, 57.
self.html	Holds all of the HTML/Javascript for my google maps object	Subsection 4.10.12, line number: 100.
var markers	A Javascript list for holding the markers coordinates	Subsection 4.10.12, line numbers: 118, 164, 187, 188, 195.
var ContentString	A Javascript variable for holding the text description for each marker	Subsection 4.10.12, line number 166.
Send	A variable to compile the SMTP protocol for sending an email	Subsection 4.10.18, line numbers: 32, 33, 34.
msg	A variable to format the email to send	Subsection 4.10.18, line numbers: 26, 27, 28, 29, 30.
result	Stores the result of an sqlite3 query	Subsection 4.10.22, line numbers: 8, 11
db_name	The name of the programs database	Subsection 4.11.22, line numbers: 5, 30, 32, 36, 38, 43, 46.
Choice	Stores the users choice whilst navigation the CLI	Subsection 4.11.19, line numbers: 22, 23, 26, 30, 33, 36, 39
Finished	A while loop condition that changes to True once the conditions are satisfied	Subsection 4.11.21, line numbers: 29, 30, 47, 91, 92, 98, 101, 103, 105, 107, 120, 121.

Ben Keppie

Candidate No. 4609

Centre No. 22151

4.5 System Evidence

4.5.1 User Interface

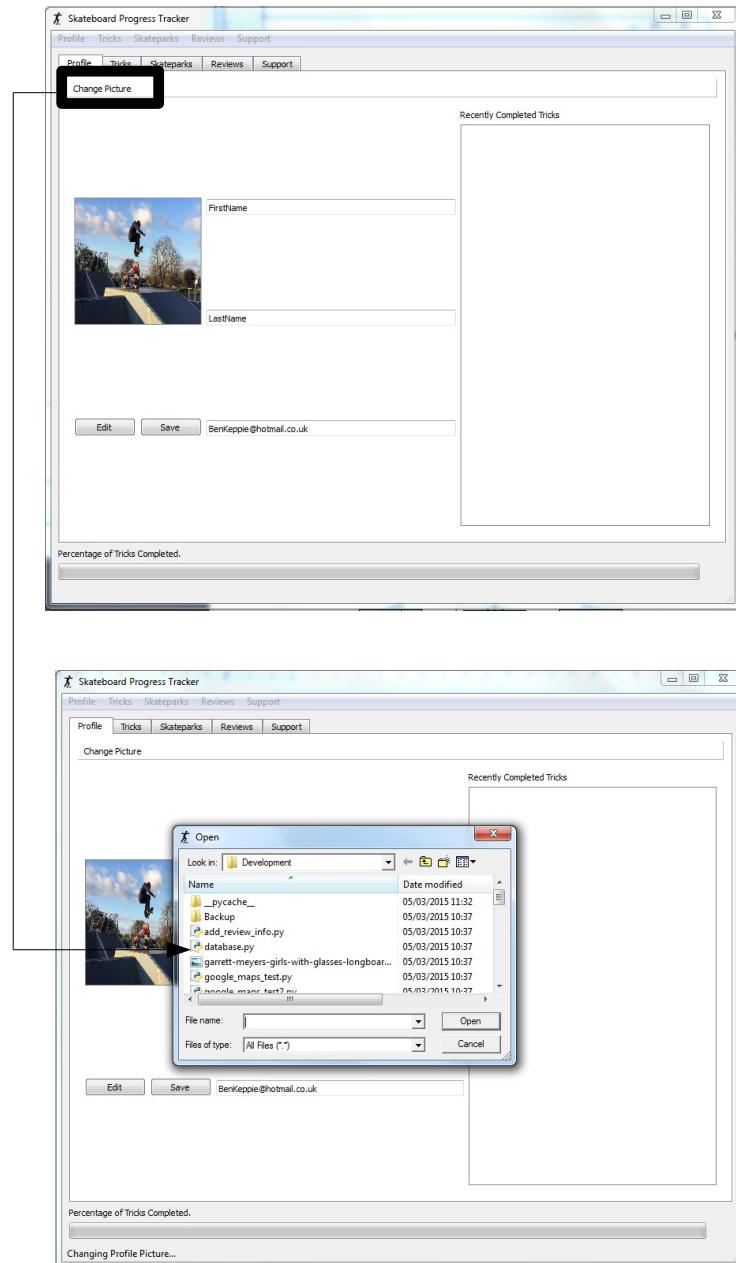


Figure 4.1: User interface, changing profile picture

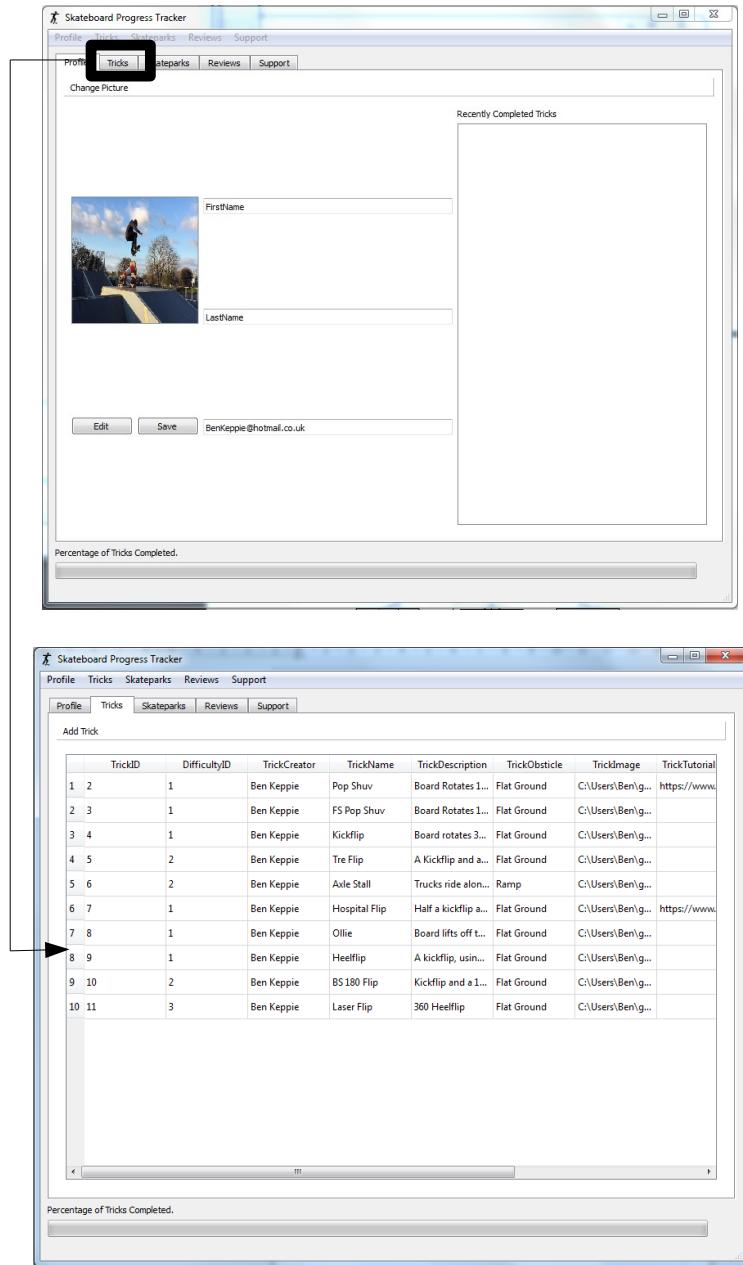


Figure 4.2: User interface, switching to the tricks tab

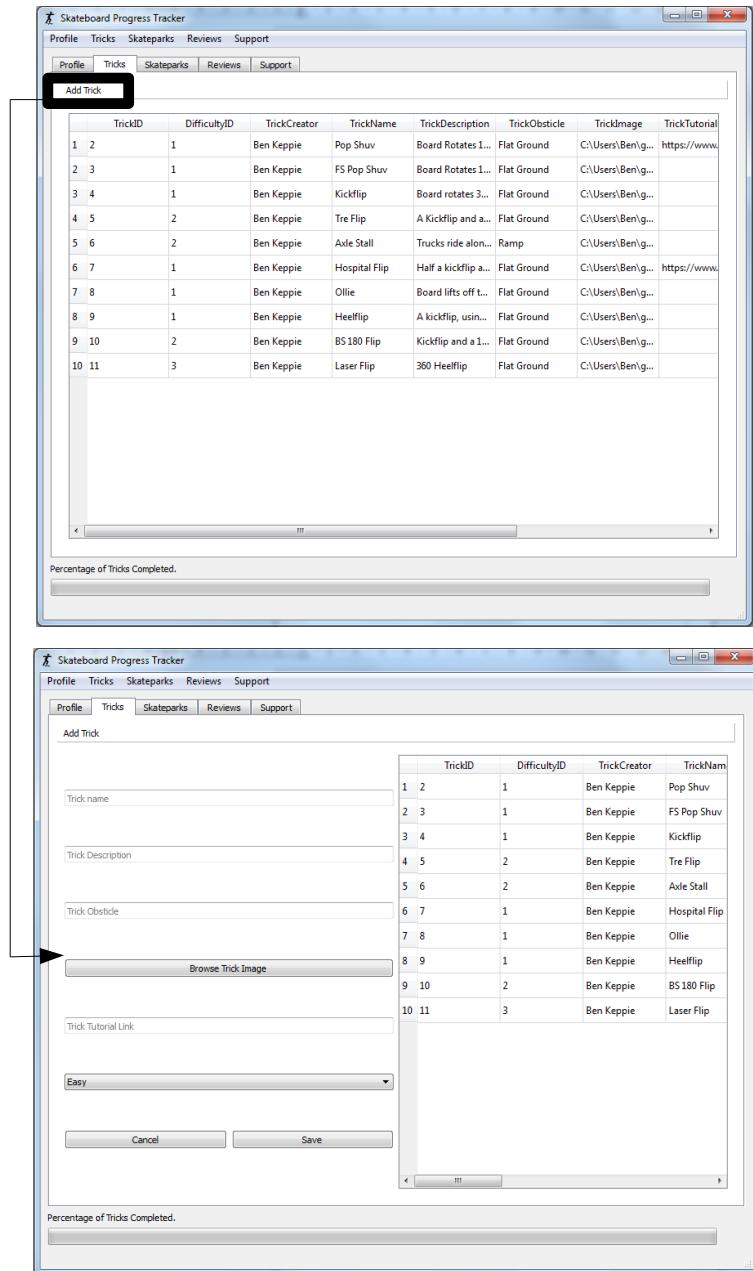


Figure 4.3: User interface, adding a trick

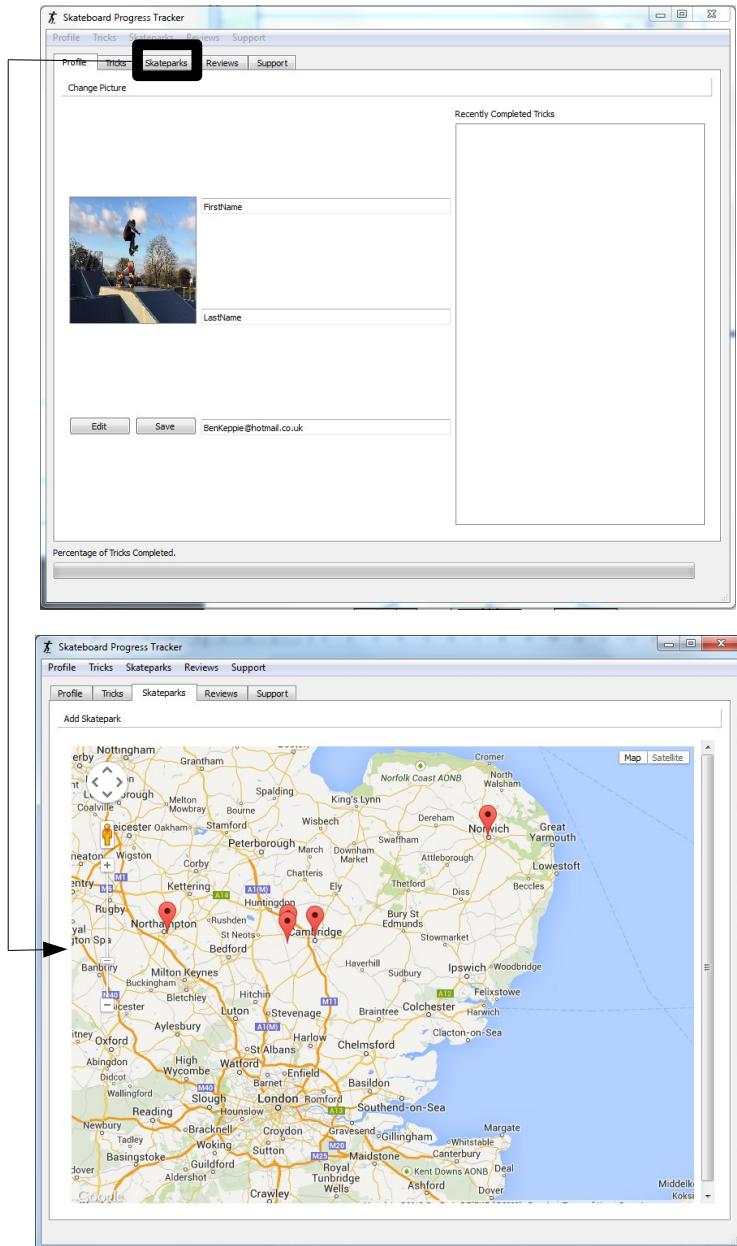


Figure 4.4: User interface, switching to the skateparks tab

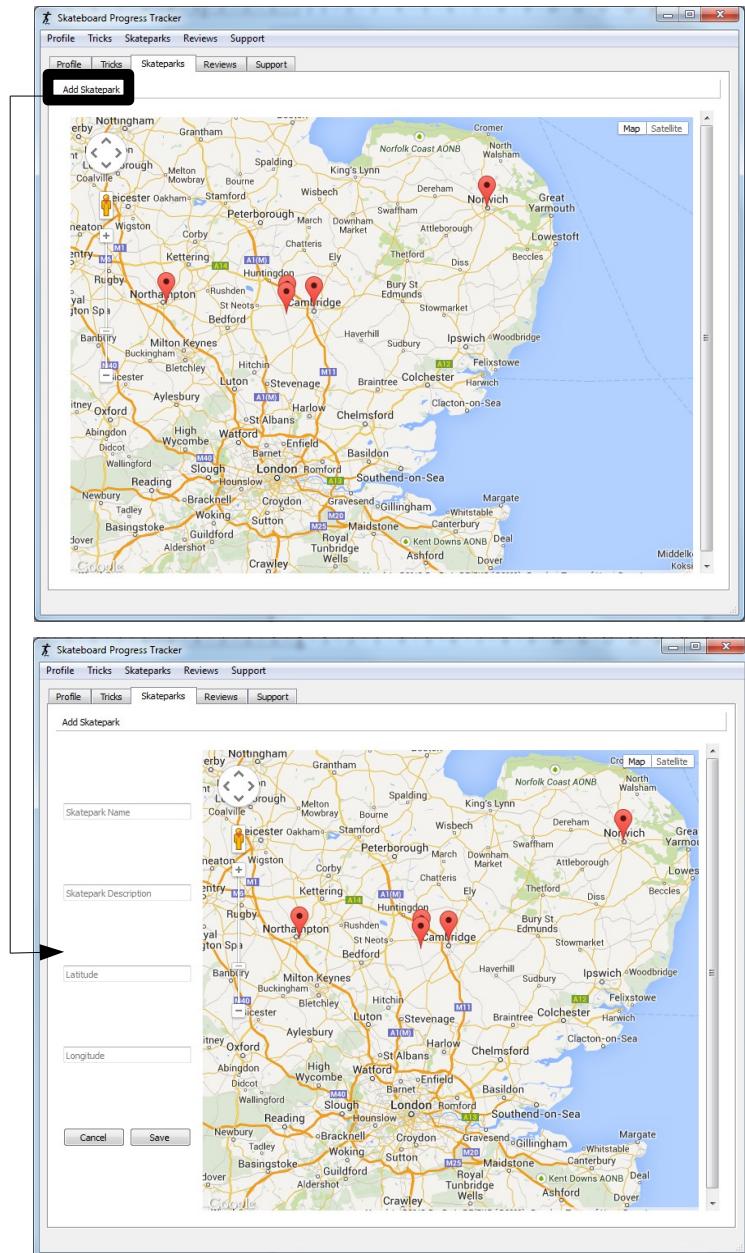


Figure 4.5: User interface, adding a skatepark

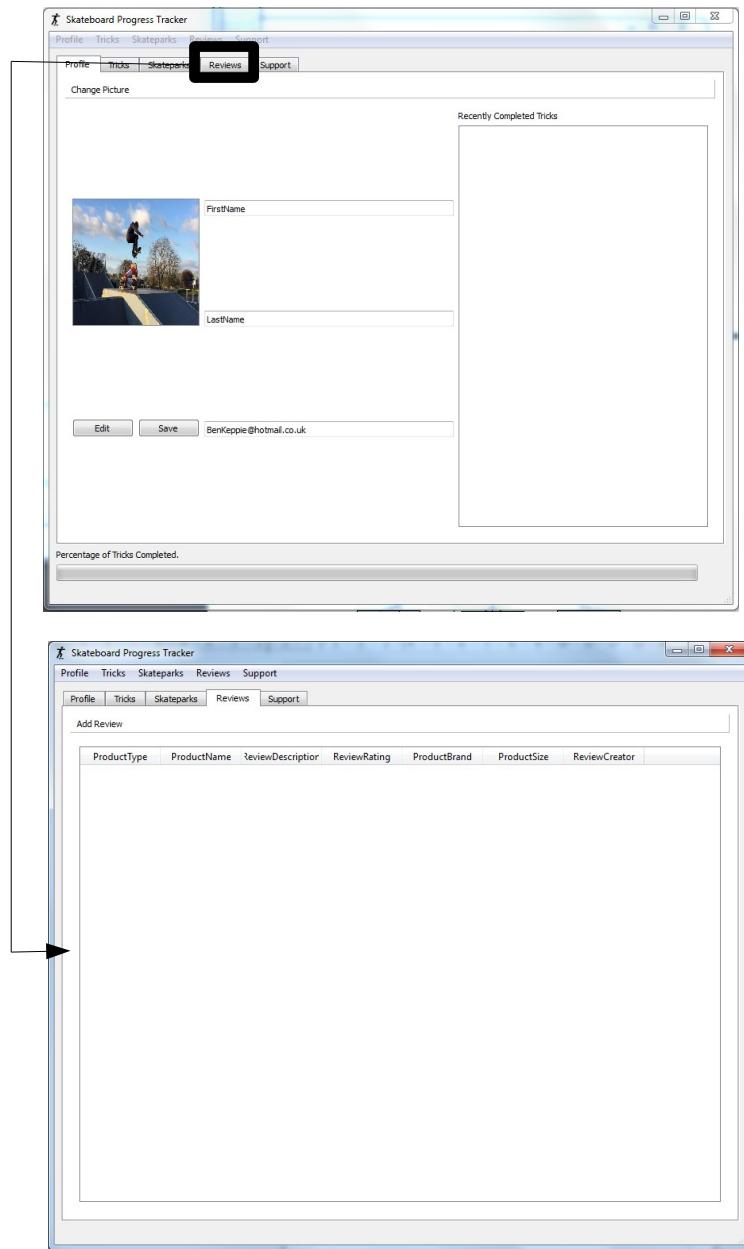


Figure 4.6: User interface, switching to the reviews tab

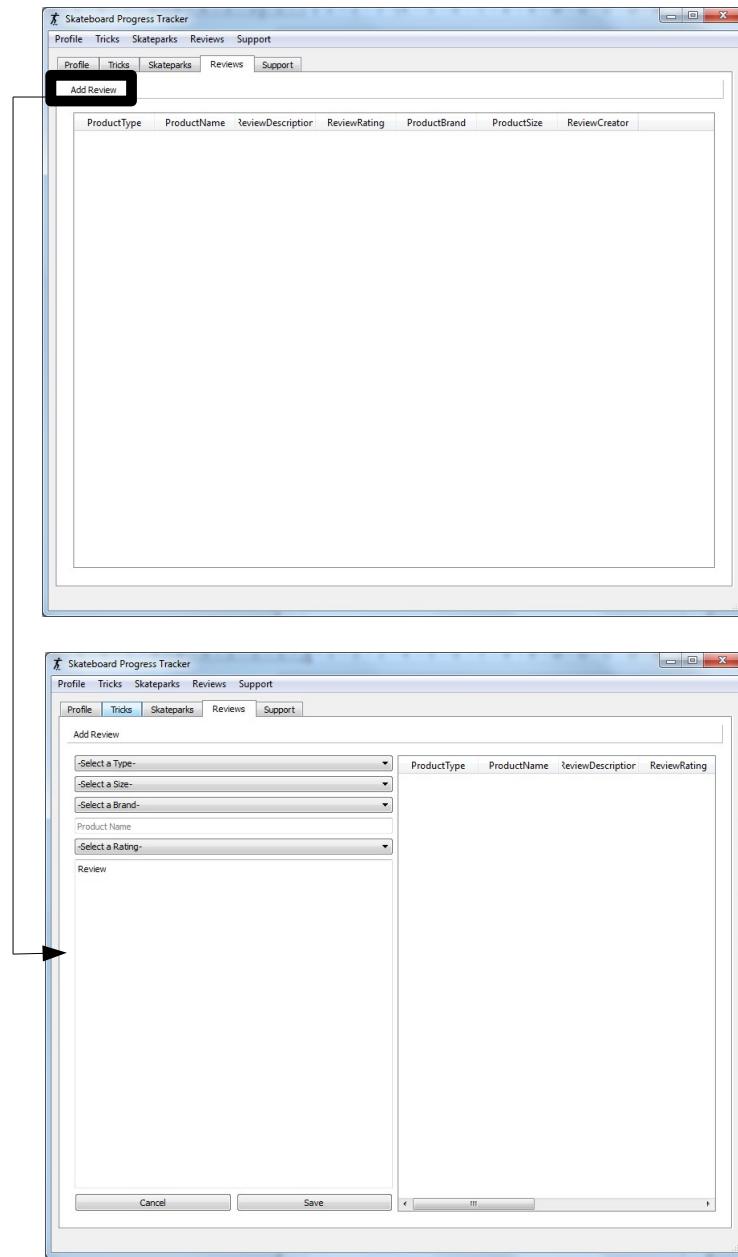


Figure 4.7: User interface, adding a review

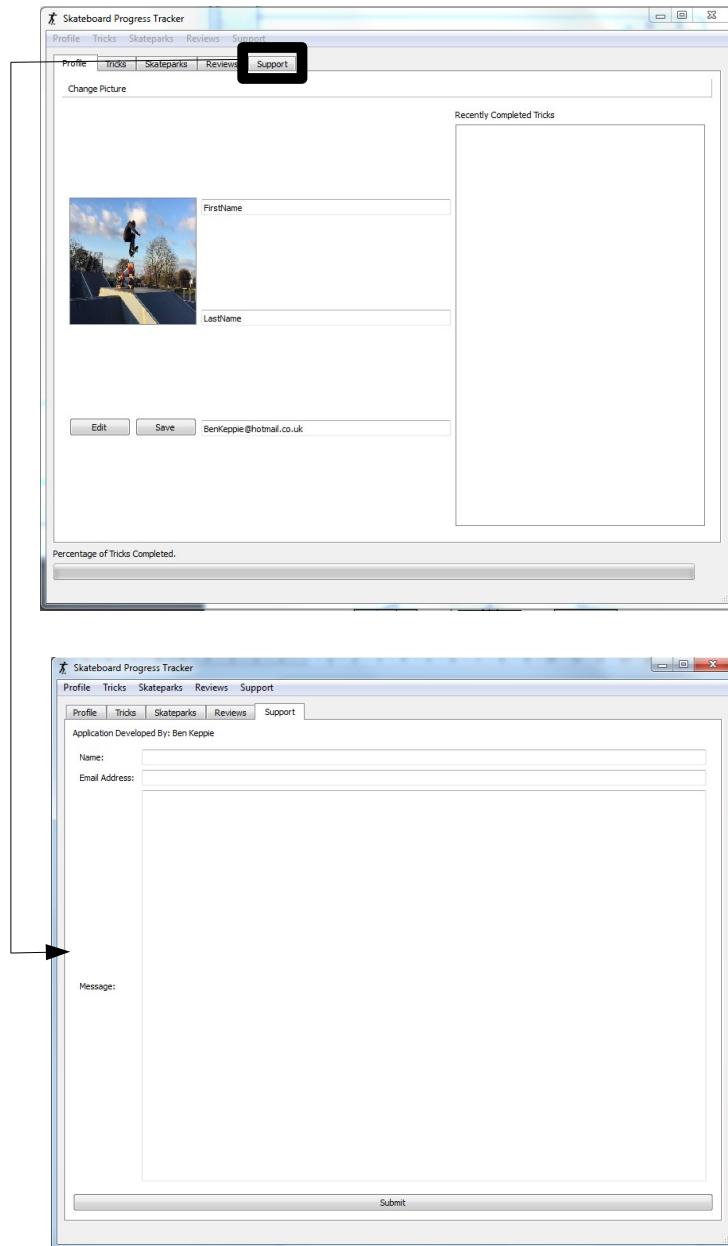


Figure 4.8: User interface, switching to the support tab

4.5.2 ER Diagram

This ER diagram is identical to the ER diagram shown in my design section (Figure 2.26 on page 79)

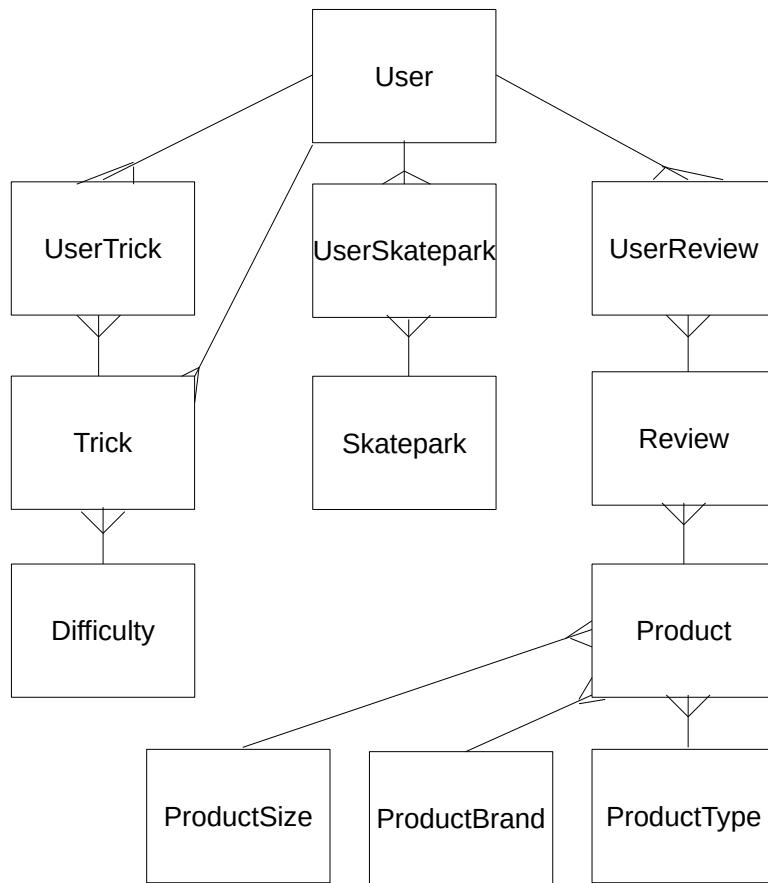
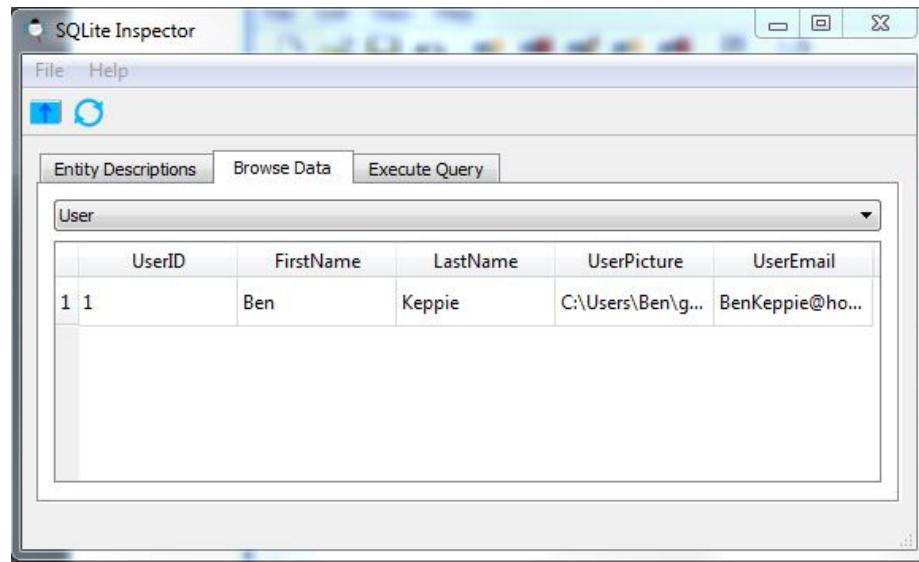


Figure 4.9: Entity-Relationship Diagram

4.5.3 Database Table Views

The figures below show all the entities and tables that I used in the creation of my program so far.

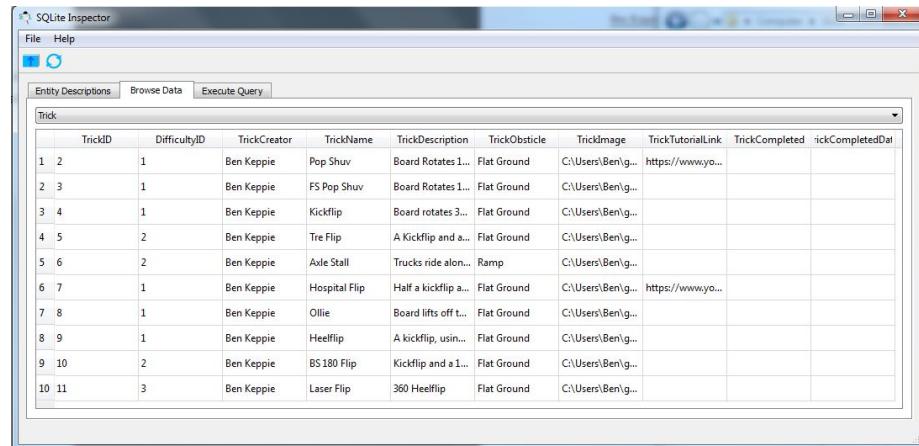


The screenshot shows the SQLite Inspector application window. The title bar says "SQLite Inspector". The menu bar has "File" and "Help". Below the menu is a toolbar with two icons. The main area has three tabs: "Entity Descriptions", "Browse Data" (which is selected), and "Execute Query". A dropdown menu under "Entity Descriptions" shows "User". The data grid displays the "User" table with columns: UserID, FirstName, LastName, UserPicture, and UserEmail. There is one row of data: UserID 1, FirstName Ben, LastName Keppie, UserPicture C:\Users\Ben\g..., and UserEmail BenKeppie@ho... .

UserID	FirstName	LastName	UserPicture	UserEmail
1	Ben	Keppie	C:\Users\Ben\g...	BenKeppie@ho...

Figure 4.10: The user table of the database

The screen shot above shows the user table which contains information about the user.



The screenshot shows the SQLite Inspector application window. The title bar says "SQLite Inspector". The menu bar has "File" and "Help". Below the menu is a toolbar with two icons. The main area has three tabs: "Entity Descriptions", "Browse Data" (which is selected), and "Execute Query". A dropdown menu under "Entity Descriptions" shows "Trick". The data grid displays the "Trick" table with columns: TrickID, DifficultyID, TrickCreator, TrickName, TrickDescription, TrickObstacle, TrickImage, TrickTutorialLink, TrickCompleted, and TrickCompletedDate. There are 10 rows of data, each representing a different trick created by Ben Keppie.

TrickID	DifficultyID	TrickCreator	TrickName	TrickDescription	TrickObstacle	TrickImage	TrickTutorialLink	TrickCompleted	TrickCompletedDate
1	2	Ben Keppie	Pop Shuv	Board Rotates 1...	Flat Ground	C:\Users\Ben\g...	https://www.yo...		
2	3	Ben Keppie	FS Pop Shuv	Board Rotates 1...	Flat Ground	C:\Users\Ben\g...			
3	4	Ben Keppie	Kickflip	Board rotates 3...	Flat Ground	C:\Users\Ben\g...			
4	5	Ben Keppie	Tre Flip	A Kickflip and a...	Flat Ground	C:\Users\Ben\g...			
5	6	Ben Keppie	Axle Stall	Trucks ride along...	Ramp	C:\Users\Ben\g...			
6	7	Ben Keppie	Hospital Flip	Half a kickflip a...	Flat Ground	C:\Users\Ben\g...			
7	8	Ben Keppie	Ollie	Board lifts off the...	Flat Ground	C:\Users\Ben\g...	https://www.yo...		
8	9	Ben Keppie	Heelflip	A kickflip, using t...	Flat Ground	C:\Users\Ben\g...			
9	10	Ben Keppie	BS 180 Flip	Kickflip and a 180...	Flat Ground	C:\Users\Ben\g...			
10	11	Ben Keppie	Laser Flip	360 Heelflip	Flat Ground	C:\Users\Ben\g...			

Figure 4.11: The trick table of the database

The screen shot above shows the tricks table which contains information about multiple tricks.

The screenshot shows the SQLite Inspector application window. The title bar says "SQLite Inspector". The menu bar has "File" and "Help". Below the menu is a toolbar with three icons. The main area has three tabs: "Entity Descriptions", "Browse Data" (which is selected), and "Execute Query". A dropdown menu shows "Skatepark". The data grid displays the following information:

	SkateparkID	SkateparkName	SkateparkLongitude	SkateparkLatitude	SkateparkDescription
1	1	Jesus Green Ska...	0.123489	52.2127	Cambridge stre...
2	2	Radlands Plaza	-0.881481	52.232	A skateboard pl...
3	3	Cambourne Sk...	-0.0609773	52.2212	A metal skat...
4	4	Bourne Church	-0.0639224	52.1904	3 stair set.
5	5	Norwich Skate...	1.29089	52.6297	Norwich Skate...

Figure 4.12: The skatepark table of the database

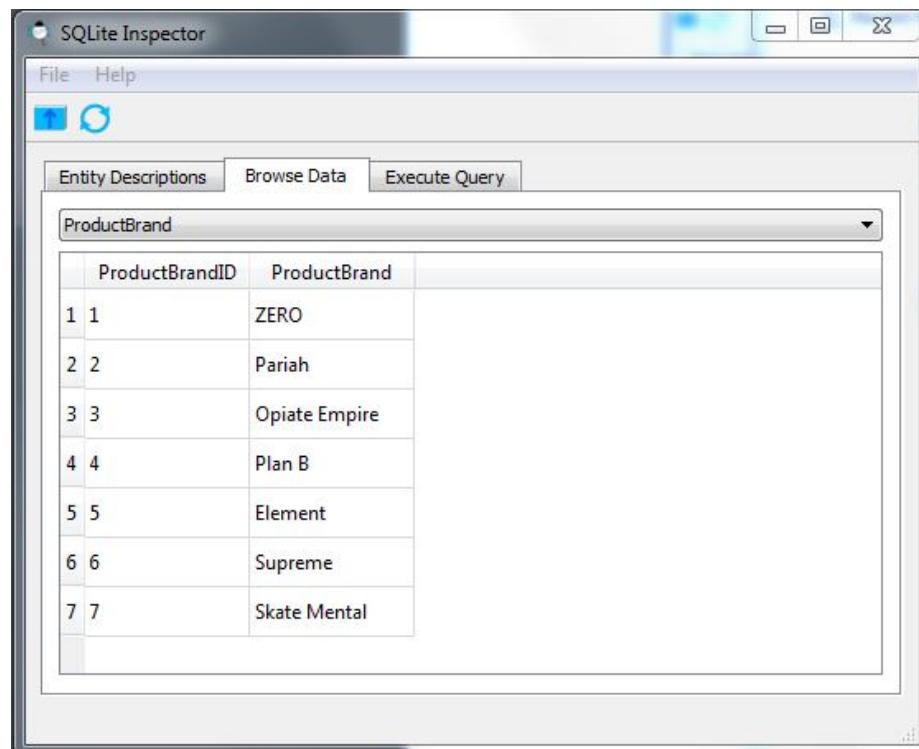
The screen shot above shows the skatepark table which contains information about multiple skateparks.

The screenshot shows the SQLite Inspector application window. The title bar says "SQLite Inspector". The menu bar has "File" and "Help". Below the menu is a toolbar with three icons. The main area has three tabs: "Entity Descriptions", "Browse Data" (which is selected), and "Execute Query". A dropdown menu shows "Review". The data grid displays the following information:

	ReviewID	ProductID	ReviewCreator	ReviewDescription	ReviewRating

Figure 4.13: The review table of the database

The screen shot above shows the review table which contains information about multiple reviews.

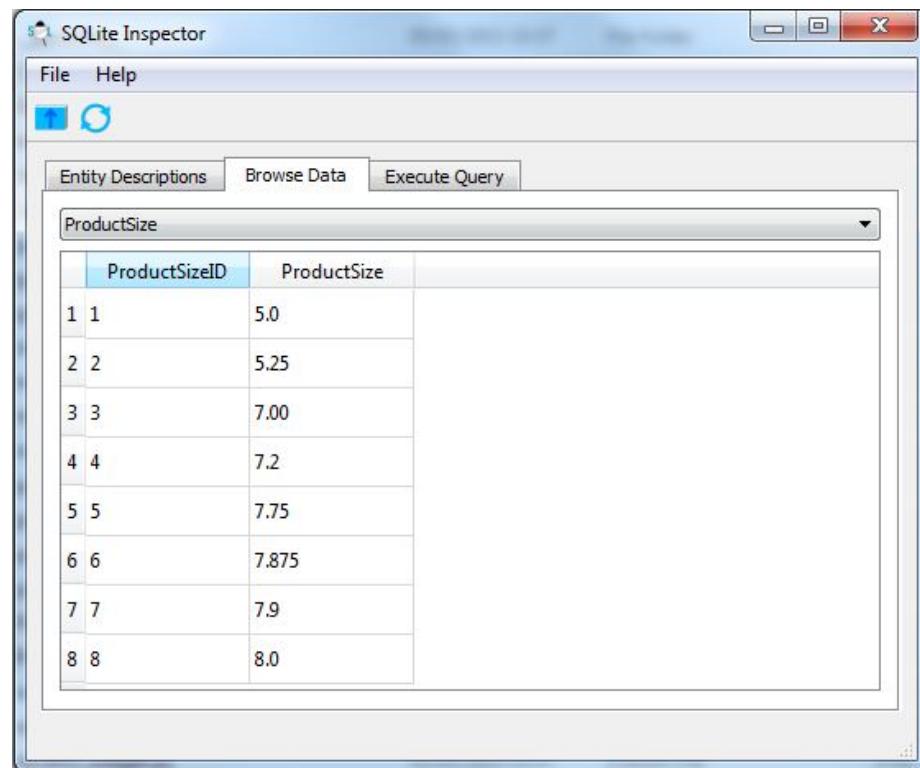


The screenshot shows the SQLite Inspector application window. The title bar reads "SQLite Inspector". The menu bar includes "File" and "Help". Below the menu is a toolbar with three icons: a blue square, a blue circle, and a magnifying glass. The main area has three tabs: "Entity Descriptions", "Browse Data", and "Execute Query". The "Browse Data" tab is selected, showing a table named "ProductBrand". The table has two columns: "ProductBrandID" and "ProductBrand". The data is as follows:

ProductBrandID	ProductBrand
1	ZERO
2	Pariah
3	Opiate Empire
4	Plan B
5	Element
6	Supreme
7	Skate Mental

Figure 4.14: The product brand table of the database

The screen shot above shows the product brand table which contains information about different product brands which you can review.



The screenshot shows the SQLite Inspector application window. The title bar reads "SQLite Inspector". The menu bar includes "File" and "Help". Below the menu is a toolbar with three icons: a blue square, a blue circle, and a magnifying glass. A tab bar at the top has three tabs: "Entity Descriptions", "Browse Data", and "Execute Query". The "Browse Data" tab is selected. A dropdown menu next to it shows "ProductSize". The main area displays a table with two columns: "ProductSizeID" and "ProductSize". The data rows are:

ProductSizeID	ProductSize
1	5.0
2	5.25
3	7.00
4	7.2
5	7.75
6	7.875
7	7.9
8	8.0

Figure 4.15: The product size table of the database

The screen shot above shows the product size table which contains information about different product sizes which you can review.

The screenshot shows the SQLite Inspector application window. The title bar reads "SQLite Inspector". The menu bar includes "File" and "Help". Below the menu is a toolbar with three icons: a magnifying glass, a blue square, and a blue circle. A tab bar at the top has three tabs: "Entity Descriptions", "Browse Data", and "Execute Query", with "Browse Data" being the active tab. A dropdown menu next to the tabs shows "ProductType". The main area is a table view with the following data:

	ProductTypeID	ProductType
1	1	Deck
2	2	Trucks
3	3	Wheels
4	4	Bearings
5	5	Bolts
6	6	Grip Tape

Figure 4.16: The product type table of the database

The screen shot above shows the product type table which contains information about different product types which you can review.

Unused Database Table Views

The figures below show the tables that I will use in the future to implement a multi-user system. However, currently the program is a single user system and therefore they do not have any data inside the tables.

The screenshot shows the SQLite Inspector application window. The title bar reads "SQLite Inspector". The menu bar includes "File" and "Help". Below the menu is a toolbar with three icons: a blue square, a blue circle, and a magnifying glass. A navigation bar at the top has three tabs: "Entity Descriptions" (selected), "Browse Data", and "Execute Query". A dropdown menu next to the tabs shows "UserTrick". The main area is a table view with three columns: "UserTrickID", "UserID", and "TrickID". The table is currently empty.

UserTrickID	UserID	TrickID

Figure 4.17: The user trick table of the database

The screenshot shows the SQLite Inspector application window. The title bar reads "SQLite Inspector". The menu bar includes "File" and "Help". Below the menu is a toolbar with three icons: a blue square, a blue circle, and a magnifying glass. A navigation bar at the top has three tabs: "Entity Descriptions" (selected), "Browse Data", and "Execute Query". A dropdown menu next to the tabs shows "UserSkatepark". The main area is a table view with three columns: "UserSkateparkID", "UserID", and "SkateparkID". The table is currently empty.

UserSkateparkID	UserID	SkateparkID

Figure 4.18: The user skatepark table of the database

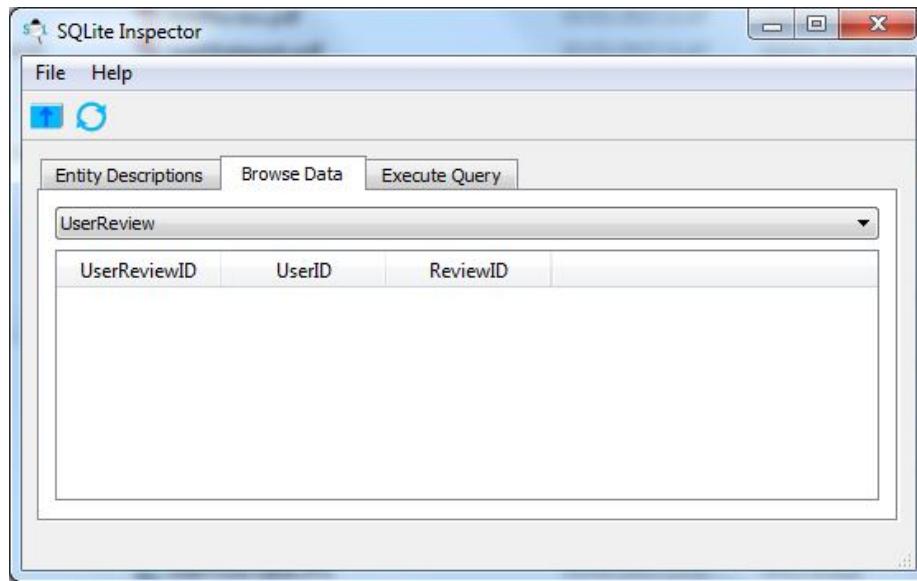


Figure 4.19: The user review table of the database

4.5.4 Database SQL

To create the entities I used the following SQL code:

```
1
2 def create_user_table():
3     db_name="skateboard_progress_tracker.db"
4     sql= """create table User (UserID integer,
5             FirstName text, LastName text, UserPicture
6             image, UserEmail text, Primary Key(UserID))"""
7     create_table(db_name,"User",sql)
8     print("Blank User Table Created.")
9
10 def create_user_trick_table():
11     db_name="skateboard_progress_tracker.db"
12     sql= """create table UserTrick (UserTrickID
13             integer, UserID integer, TrickID integer,
14             Primary Key(UserTrickID), Foreign Key(UserID)
15             references User(UserID), Foreign Key(TrickID)
16             references Trick(TrickID))"""
17     create_table(db_name,"UserTrick",sql)
18     print("Blank UserTrick Table Created.")
```

```
14
15 def create_trick_table():
16     db_name="skateboard_progress_tracker.db"
17     sql= """create table Trick (TrickID integer,
18                             DifficultyID integer, TrickCreator text,
19                             TrickName text, TrickDescription text,
20                             TrickObstacle text, TrickImage image,
21                             TrickTutorialLink text, TrickCompleted boolean,
22                             TrickCompletedDate text, Primary Key(TrickID),
23                             Foreign Key(DifficultyID) references
24                             Difficulty(DifficultyID))"""
25     create_table(db_name,"Trick", sql)
26     print("Blank Trick Table Created.")
27
28
29 def create_difficulty_table():
30     db_name="skateboard_progress_tracker.db"
31     sql= """create table Difficulty (DifficultyID
32                             integer, TrickDifficulty text,
33                             DifficultyDescription text, Primary
34                             Key(DifficultyID))"""
35     create_table(db_name,"Difficulty", sql)
36     print("Blank Difficulty Table Created.")
37
38
39 def create_user_review_table():
40     db_name="skateboard_progress_tracker.db"
41     sql= """create table UserReview (UserReviewID
42                             integer, UserID integer, ReviewID integer,
43                             Primary Key(UserReviewID), Foreign Key(UserID)
44                             references User(UserID), Foreign Key(ReviewID)
45                             references Review(ReviewID))"""
46     create_table(db_name,"UserReview", sql)
47     print("Blank UserReview Table Created.")
48
49
50 def create_review_table():
51     db_name="skateboard_progress_tracker.db"
52     sql= """create table Review (ReviewID integer,
53                             ProductID integer, ReviewCreator text,
54                             ReviewDescription text, ReviewRating integer ,
55                             Primary Key(ReviewID), Foreign Key(ProductID)
56                             references Product(ProductID))"""
57     create_table(db_name,"Review", sql)
58     print("Blank Review Table Created.")
59
60
61 def create_product_table():
62     db_name="skateboard_progress_tracker.db"
```

```
43     sql= """create table Product (ProductID integer,
44             ProductBrandID integer, ProductTypeID integer,
45             ProductSizeID integer, ProductName text ,
46             Primary Key(ProductID), Foreign
47                 Key(ProductBrandID) references
48                 ProductBrand(ProductBrandID),Foreign
49                 Key(ProductSizeID) references
50                 ProductSize(ProductSizeID),Foreign
51                 Key(ProductTypeID) references
52                 ProductType(ProductTypeID))"""
53
54     create_table(db_name,"Product", sql)
55     print("Blank Product Table Created.")
56
57
58
59
60
61
62
63
64
65
66
67
68
69
```

```

    Skatepark(SkateparkID))"""
70     create_table(db_name,"UserSkatepark", sql)
71     print("Blank UserTrick Table Created.")
72
73 def create_skatepark_table():
74     db_name="skateboard_progress_tracker.db"
75     sql= """create table Skatepark (SkateparkID
76         integer, SkateparkName text,
77         SkateparkLongitude integer, SkateparkLatitude
78         integer, SkateparkDescription text, Primary
79         Key(SkateparkID))"""
80     create_table(db_name,"Skatepark", sql)
81     print("Blank ProductType Table Created.")

```

To see the full code please look at Subsection 4.10.22.

4.5.5 SQL Queries

Change Name SQL Query

```

1     def change_name(self,FirstName,LastName):
2         FirstName=FirstName
3         LastName=LastName
4         if (FirstName=="") or (LastName==""):
5             print("Name Not Changed")
6         else:
7             values=(FirstName,LastName, 1)
8             with
9                 sqlite3.connect("skateboard_progress_tracker.db")
10                    as db:
11                        cursor = db.cursor()
12                        sql="update User set FirstName=?,
13                            LastName=? where UserID=?"
14                        cursor.execute(sql,values)
15                        db.commit()

```

4.6 Testing

Change Email SQL Query

```

1     def change_email(self,Email):
2         Email=Email

```

```
3     if (Email==""):
4         print("Email Not Changed")
5     else:
6         values=(Email,1)
7         with
8             sqlite3.connect("skateboard_progress_tracker.db")
9                 as db:
10                    cursor = db.cursor()
11                    sql="update User set UserEmail=?"
12                        where UserID=?"
13                    cursor.execute(sql,values)
14                    db.commit()
```

Change Picture SQL Query

```
1     def change_picture(self,FilePath):
2         FilePath=FilePath
3
4         values=(FilePath,1)
5         with
6             sqlite3.connect("skateboard_progress_tracker.db")
7                 as db:
8                    cursor = db.cursor()
9                    sql="update User set UserPicture=? where
10                        UserID=?"
11                    cursor.execute(sql,values)
12                    db.commit()
13                    print("Picture Changed")
```

Select First Name SQL Query

```
1     def get_first_name(self):
2         with
3             sqlite3.connect("skateboard_progress_tracker.db")
4                 as db:
5                     cursor=db.cursor()
6                     cursor.execute("select FirstName from
7                         User where UserID=?",(1,))
8                     FirstName=cursor.fetchone()
9                     print(FirstName)
10                    return FirstName
```

Select Last Name SQL Query

```

1     def get_last_name(self):
2         with
3             sqlite3.connect("skateboard_progress_tracker.db")
4                 as db:
5                     cursor=db.cursor()
6                     cursor.execute("select LastName from
7                         User where UserID=?",(1,))
8                     LastName=cursor.fetchone()
9                     return LastName

```

Delete Trick Row SQL Query

```

1     def delete_row(self,Row):
2         values=(Row+1,)
3         with
4             sqlite3.connect("skateboard_progress_tracker.db")
5                 as db:
6                     cursor = db.cursor()
7                     sql="DELETE FROM Trick WHERE TrickID=?"
8                     cursor.execute(sql,values)
9                     db.commit()

```

Select All Tricks From the Trick Table SQL Query

```

1     def show_all_tricks(self):
2         query = QSqlQuery()
3         query.prepare(""" SELECT * FROM Trick""")
4         query.exec_()
5         return query

```

Add Trick to Database SQL Query

```

1     def
2         add_trick_to_database(self,DifficultyID,TrickName,TrickDescription,TrickImage,
3                               TrickTutorialLink)
4
5         print("hi2")
6
7         values=(DifficultyID, "Ben Keppie",
8                 TrickName, TrickDescription,
9                 TrickObstacle, TrickImage,
10                TrickTutorialLink)

```

```

5      with
6          sqlite3.connect("skateboard_progress_tracker.db")
7              as db:
8                  cursor = db.cursor()
9                  sql="insert into Trick(DifficultyID,
10                      TrickCreator,TrickName,TrickDescription,TrickObstacle,TrickIm
11                      values (?,?,?,?,?,?)"
12                      cursor.execute(sql,values)
13                      db.commit()
14                      print()
15                      print("Trick Successfully Created.")
16                      print()

```

Add Skatepark to Database SQL Query

```

1      def
2          add_skatepark(self,SkateparkName,SkateparkDescription,Latitude,Longitude
3                          values=(SkateparkName,SkateparkDescription,Latitude,Longitude)
4                          with
5                              sqlite3.connect("skateboard_progress_tracker.db")
6                                  as db:
7                                      cursor = db.cursor()
8                                      sql="insert into
9                                          Skatepark(SkateparkName,SkateparkDescription,
10                                              SkateparkLatitude, SkateparkLongitude)
11                                              values (?,?,?,?,?)"
12                                              cursor.execute(sql,values)
13                                              db.commit()
14                                              print()
15                                              print("Skatepark Successfully Created.")
16                                              print()

```

Select all Reviews Fom the Reviews Table SQL Query

```

1      def show_all_reviews(self):
2          query = QSqlQuery()
3          query.prepare(""" SELECT
4              ProductType,ProductName,ReviewDescription,ReviewRating,ProductBr
5              FROM review, product,
6                  productbrand,producttype,productsizes""")
6          query.exec_()
7          return query

```

4.7 Testing

To view my full test plan look at Figure ?? on page ??

4.7.1 Summary of Results

My testing showed that my program was not a fully GUI operated program. It also identified key, minor issues with validation which I will fix then the next version in my program.

4.7.2 Known Issues

There are a few known issues with my system. These are outlined below.

Flashing Tables on Start-Up

When starting up my program, the QSqlTableView objects load individually before loading the whole program (trick and review table). This doesn't cause a problem with the functionality of the program; however the start-up time is increased. Through interactive debugging using the pdb debugger I managed to find that the problem occurs when the model is set; however I do not know how to resolve this issue. I have attempted to move around the order of setting layouts as the problem would appear to be a problem in the order of setting the layout. However, doing this did not resolve the issue.

Validation

The validation involved with all of the file paths is not present in the code and the validation involved with the QLineEdit's on the profile tab are also not present. These would be fixed by a simple validation method similar to that of the 'add trick' and 'add skatepark' validation lines.

Incomplete GUI

My GUI is incomplete for my program as the review tabs functionality is not present. This would be fixed by an extended period of time for working on my program.

4.8 Code Explanations

4.8.1 Difficult Sections

Creating a Custom Webpage For Debugging

```
1 class CustomQWebPage(QWebPage):
2     """A class to act as the webpage for the google
3         maps module"""
4     def __init__(self):
5         super().__init__()
6         print("QWebPage constructor")
7
8     def
9         javaScriptConsoleMessage(self,message,lineNumber,sourceID):
10             #An overridden method to display a javascript
11                 console message
12             print()
13             print(message,lineNumber,sourceID)
14             print("javascript console message^")
```

As I was having issues with the javascript functioning correctly within the python webpage, I decided to override the QWebPage class in order to activate the Javascript console message method. From this I was able to print the error message along with the line number the error is on and the error messages source ID which helped me debug my Javascript code problem.

Google Maps Javascript

```
1         self.html='''<!DOCTYPE html>
2 <html>
3     <head>
4         <meta name="viewport" content="initial-scale=1.0,
5             user-scalable=no">
6         <meta charset="utf-8">
7         <title>Simple markers</title>
8         <style>
9             html, body, #map-canvas {
10                 height: 100%;
11                 width: 100%
12                 margin: 0px;
13                 padding: 0px
14             }
15         </style>
```

```
15      <script
16          src="https://maps.googleapis.com/maps/api/js?key=AIzaSyC5RcJ7vLSEYF32
17      </script>
18
19      var map;
20      var markers = [];
21      var results = [];
22      var coords = [];
23      var highestLevel;
24
25      function initialize() {
26
27
28          var Centre = new
29              google.maps.LatLng(52.20255705185695,0.1373291015625);
30          var mapOptions = {
31              zoom: 8,
32              minZoom: 3,
33              center: Centre,
34          }
35          map = new
36              google.maps.Map(document.getElementById('map-canvas'),mapOptions)
37
38          google.maps.event.addListener(map, 'click',
39              function(event) {
40                  AddMarker(event.latLng);
41              });
42
43          function
44              MarkersFromDatabase(SkateparkLat,SkateparkLng,SkateparkDescripti
45              SkateparkName) {
46
47              var Skatepark = new
48                  google.maps.LatLng(SkateparkLat,SkateparkLng);
49
50              AddMarker(Skatepark,SkateparkDescription,
51                  SkateparkName); }
```

```
52     function AddMarker(Location,Description,
53                           SkateparkName) {
54
55     var marker = new google.maps.Marker({
56       title: 'Test',
57       position: Location,
58       animation: google.maps.Animation.DROP,
59       map: map
60     });
61     //markers.push(marker);
62     var lat = marker.getPosition().lat();
63     var lng = marker.getPosition().lng();
64     markers.push({Object:marker,"Lat":lat,"Lng":lng,
65                   "Desc":Description});
66
67     var contentString = ('<div id="content"><div
68       id="siteNotice"></div> <h1
69       id="firstHeading" class="firstHeading">' +
70       SkateparkName + '</h1> <div
71       id="bodyContent"><p>' + Description
72       +'</p></div></div>');
73
74     var infowindow = new google.maps.InfoWindow({
75       content: contentString
76     });
77
78     google.maps.event.addListener(marker,
79       'rightclick', function(event) {
80       marker.setMap(null);
81     });
82     google.maps.event.addListener(marker,
83       'mouseover', function(event) {
84       infowindow.open(map,marker);
85     });
86     google.maps.event.addListener(marker,'mouseout',
87       function(event){
88       infowindow.close(map,marker)
89     });
90
91   }
92
93
94   function GetMarkers(){
```

```

88     var Longitude=markers[markers.length - 1]["Lng"]
89     var Latitude=markers[markers.length - 1]["Lat"]
90     var coors=[Latitude ,Longitude]
91     return coors;
92   }
93
94
95   function DeleteMarkers(){
96     markers = [];
97     initialize();
98   }
99
100  google.maps.event.addDomListener(window, 'load',
101    initialize);
102    </script>
103  </head>
104  <body>
105    <div id="map-canvas"></div>
106  </body>
107 </html> '',
108           self.setHtml(self.html)

```

For my program, I used multiple programming languages. For example: Python, Javascript, HTML and SQL. In my program I created an interactive google maps object in Javascript. I had to activate my own API key (AIzaSyC5RcJ7vLSEYF32KqDusnuRcLJiHW8EbDg) to send requests to the Google maps server. My code above shows the HTML, Javascript hybrid I used to create the interactive google maps object. This code contains all of the functions involved in the processes that my map object can carry out. For example: adding markers, hiding markers and adding an info window to a marker.

Running Javascript From Python

```

1           self.LastMarker=
2             self.CustomPage.mainFrame().evaluateJavaScript("GetMarkers()")

```

To run specific Javascript functions within my python file I had to access the custom pages' main frame and then run an 'evaluateJavaScript' function which then analyses my Javascript code and runs the corresponding function.

Mouse Press Event To Allow for Coordinates To Be Calculated

```

1   def mousePressEvent(self,event):
2     super().mousePressEvent(event)

```

```

3     if not
4         self.parent.skatepark_name.isReadOnly():
5             self.get_last_marker()
6     else:
7         print()
8         print("In view only mode.")
9         print()
10    def get_last_marker(self):
11        self.LastMarker=
12            self.CustomPage.mainFrame().evaluateJavaScript("GetMarkers()")
13        print(self.LastMarker)
14        self.parent.fill_line_edits(self.LastMarker)

```

My code above shows the python code for listening for a mouse click on the google maps object. If 'add skatepark' form on the side of the tab was open then the coordinates would be filled into the longitude and latitude QLineEdits.

Deleting Rows From QTableView

Through setting the tricks result table selection behaviour to selecting rows (shown below)

```

1     self.results_table.setSelectionBehavior(QAbstractItemView.SelectRows)

```

I was able to select the whole row and therefore gather all the information I needed to delete the row from the database.

```

1     def keyPressEvent(self,event):
2         indexes=self.results_table.selectionModel().selectedRows()
3         for index in indexes:
4             Row=index.row()
5             if event.key()==Qt.Key_Delete:
6                 self.delete_trick_warning(Row)

```

The code above shows an overridden class that allowed me to monitor if the delete key had been pressed. If the delete key had been pressed the variable row would be assigned all of the relevant row information, and this is then passed into a QDialog method which allows the user to decide whether they wish to delete that row forever (QDialog code shown below).

```

1     def delete_trick_warning(self,Row):
2         self.delete_dialog=QDialog()
3         self.dialog_VBoxLayout=QVBoxLayout()
4         self.dialog_button_layout=QHBoxLayout()
5         if not hasattr(self,"delete_message"):

```

```

6             self.delete_message=QLabel("Are you sure
7                 you wish to delte this trick?")
8             self.dialog_VBoxLayout.addWidget(self.delete_message)
9             if not hasattr(self,"delete_cancel"):
10                self.delete_cancel=QPushButton("Cancel")
11                self.dialog_button_layout.addWidget(self.delete_cancel)
12                if not hasattr(self,"delete_save"):
13                    self.delete_save=QPushButton("Save")
14                    self.dialog_button_layout.addWidget(self.delete_save)
15                    self.dialog_VBoxLayout.addLayout(self.dialog_button_layout)
16                    self.delete_dialog.setLayout(self.dialog_VBoxLayout)
17                    self.delete_dialog.show()
18                    self.delete_cancel.clicked.connect(self.delete_cancel_clicked)
19                    self.delete_save.clicked.connect(self.delete_save_clicked,Row)
20
21        def delete_cancel_clicked(self):
22            self.delete_dialog.close()
23            self.parent.StatusBar.showMessage("Trick Not
24            Deleted",2000)
25
26        def delete_save_clicked(self,Row):
27            self.delete_dialog.close()
28            self.connection.delete_row(Row)
29            query = self.connection.show_all_tricks()
30            self.model.setQuery(query)
31            self.parent.StatusBar.showMessage("Trick
32            Successfully Deleted",2000)

```

Finally the code below shows the SQL query executed to delete the row. The results table automatically gets updated with any changes to the database.

```

1     def delete_row(self,Row):
2         values=(Row+1,)
3         with
4             sqlite3.connect("skateboard_progress_tracker.db")
5                 as db:
6                     cursor = db.cursor()
7                     sql="DELETE FROM Trick WHERE TrickID=?"
8                     cursor.execute(sql,values)
9                     db.commit()
10                    print()

```

Sending Emails

```
1     def send_email(self):
2         msg=MIMEText(self.form_email_line_edit.text()+
3                     self.form_message_line_edit.toPlainText())
4         print(msg)
5         msg["Subject"]="Skateboard Progress Tracker
6                         Support"
7         msg["From"]="SkateboardProgressTracker@gmail.com"
8         msg["To"] = "BenKeppie@hotmail.co.uk"
9
10        Send=smtplib.SMTP("smtp.gmail.com")
11        Send.sendmail(msg["From"],msg["To"],msg)
12        Send.quit
```

The code above shows the python code I used to allow for an email to be sent to me with details about bugs in my program. First of all I set the MIMEText of the email to include the message that the user set, followed by their email address. I then set a preset subject of the email as 'Skateboard Progress Tracker Support' and then send the email from a specially make google email account, to my personal email. I then use Googles' SMTP protocols to automatically send the email from my google mail account to my personal email.

4.8.2 Self-created Algorithms

My program doesn't contain any complex, self-created algorithms.

4.9 Settings

No settings need to be changed on the client's computer to run my program. All of the neccesary modules required are supplied with python, PyQt and my public API key is used to run the google maps object.

4.10 Acknowledgements

- Acknowledgment 1 - YouTube link regular expression - Found on <http://stackoverflow.com/questions/3717115/regular-expression-for-youtube-links> by Stack Overflow user <http://stackoverflow.com/users/3652125/fanmade>
- Acknowledgment 2 - Google Maps JavaScript API - Gained from Google's APIs Console. - <https://developers.google.com/maps/documentation/javascript/tutorial>

- Acknowledgment 3 - Javascript help on Stack Overflow - <http://stackoverflow.com/questions/28253168/running-a-javascript-function-from-qwebview-google-maps-api-pyqt>
I posted a question to attempt to resolve an issue I had with my Javascript code.

4.11 Code Listing

4.11.1 Main Window

```
1 import sys
2
3 import time
4 import subprocess
5 import os.path
6 from PyQt4.QtGui import *
7 from PyQt4 import QtGui
8 from PyQt4.QtCore import *
9 from PyQt4.QtWebKit import *
10
11 from menu_bar import *
12
13
14
15 from profile_widget import *
16 from profile_toolbar import *
17
18 from tricks_widget import *
19 from tricks_toolbar import *
20
21 from skateparks_widget import *
22 from skateparks_toolbar import *
23
24 from reviews_widget import *
25
26 from support_widget import *
27
28
29 class MainWindow(QMainWindow):
30     """Class for the main window of my program"""
31     def __init__(self):
32         super().__init__()
33         self.setWindowTitle("Skateboard Progress
34             Tracker")
35         self.app = QtGui.QApplication([])
```

```
35         self.set_icon()
36         self.main_VBoxLayout=QVBoxLayout()
37         self.central_widget=QWidget()
38         self.ProgressBar=QProgressBar()
39         self.ProgressBarLabel=QLabel("Percentage of
              Tricks Completed.")
40
41         #create tabs
42         self.create_tabs()
43
44         #Create Menu Bar
45         self.Menu=Menu(self)
46         self.setMenuBar(self.Menu.MenuBar)
47
48         #Create Statusbar
49         self.StatusBar=QStatusBar()
50         self.setStatusBar(self.StatusBar)
51
52         self.tabs.currentChanged.connect(self.progress_bar_hide)
53
54
55     def progress_bar_hide(self):
56         if (self.tabs.currentIndex() >=2):
57             self.ProgressBarLabel.hide()
58             self.ProgressBar.hide()
59         else:
60             self.ProgressBar.show()
61             self.ProgressBarLabel.show()
62
63
64
65
66
67     def set_icon(self):
68         self.app_icon=QtGui.QIcon()
69         self.app_icon.addFile("ProgramIcon.png", QSize(16,16))
70         self.app.setWindowIcon(self.app_icon)
71
72
73
74
75
76     def create_tabs(self):
77
78
79
```

```
80         self.tabs=QTabWidget()
81
82     #Create Widgets
83     self.profile_tab=DisplayProfileWidget(self)
84
85     self.tricks_tab=DisplayTricksWidget(self)
86     self.skateparks_tab=DisplaySkateparksWidget(self)
87     self.reviews_tab=DisplayReviewsWidget(self)
88     self.support_tab=DisplaySupportWidget(self)
89
90
91     #Add Tabs
92     self.tabs.addTab(self.profile_tab, "Profile")
93     self.tabs.addTab(self.tricks_tab, "Tricks")
94     self.tabs.addTab(self.skateparks_tab,
95                     "Skateparks")
96     self.tabs.addTab(self.reviews_tab, "Reviews")
97     self.tabs.addTab(self.support_tab, "Support")
98
99     self.main_VBoxLayout.addWidget(self.tabs)
100    self.main_VBoxLayout.addWidget(self.ProgressBarLabel)
101    self.main_VBoxLayout.addWidget(self.ProgressBar)
102    self.central_widget.setLayout(self.main_VBoxLayout)
103
104
105
106
107
108
109
110
111     #Add all to the main window
112     self.setCentralWidget(self.central_widget)
113
114 def splash_screen():
115     splash_pix = QPixmap('SplashScreen1.png')
116     splash = QSplashScreen(splash_pix,
117                           Qt.WindowStaysOnTopHint)
118     splash.setMask(splash_pix.mask())
119     splash.show()
120     time.sleep(2)
121     splash.finish(splash)
122
123 def main():
124     application=QApplication(sys.argv)
```

```
124     window=MainWindow()
125     #splash_screen()
126
127     window.show()
128     window.raise_()
129     application.exec_()
130     print()
131
132
133
134
135 if __name__=="__main__":
136     main()
```

4.11.2 Main Tabbed Widget

```
1 from PyQt4.QtGui import *
2 from PyQt4 import QtGui
3 from PyQt4.QtCore import *
4
5 class CustomQTabWidget(QTabWidget):
6     """A class for my custom QTabWidget"""
7     def __init__(self,parent):
8         super().__init__()
9         self.parent=parent
10
11    def currentChanged(self):
12        print("Change in tab")
```

4.11.3 Menu Bar

```
1 from PyQt4.QtGui import *
2 from PyQt4 import QtGui
3 from PyQt4.QtCore import *
4 import shutil
5
6
7 from profile_sql_connections import *
8
9 class Menu(QMenu):
10     """A class to represent the menu bar for the
profile"""
```

```
11     def __init__(self, parent):
12         super().__init__()
13         self.parent=parent
14         self.MenuBar=QMenuBar()
15         #create actions
16
17         self.change_picture= QAction("Change
18             Picture", self)
19
20         self.add_trick= QAction("Add Trick", self)
21
22         self.add_skatepark= QAction("Add
23             Skatepark", self)
24
25         self.add_review= QAction("Add Review", self)
26
27
28         self.contact_support= QAction("Contact
29             Support", self)
30
31
32         #create options
33         self.profile_menu=self.MenuBar.addMenu("Profile")
34         self.tricks_menu=self.MenuBar.addMenu("Tricks")
35         self.skateparks_menu=self.MenuBar.addMenu("Skateparks")
36         self.reviews_menu=self.MenuBar.addMenu("Reviews")
37         self.support_menu=self.MenuBar.addMenu("Support")
38
39         #add actions to menu
40
41         self.profile_menu.addAction(self.change_picture)
42
43         self.tricks_menu.addAction(self.add_trick)
44
45         self.skateparks_menu.addAction(self.add_skatepark)
46
47         self.reviews_menu.addAction(self.add_review)
48
49         self.support_menu.addAction(self.contact_support)
50
51         #connections
52
53         self.change_picture.triggered.connect(self.change_picture_connection)
54         self.add_trick.triggered.connect(self.add_trick_connection)
55         self.add_skatepark.triggered.connect(self.add_skatepark_connection)
```

```
54         self.add_review.triggered.connect(self.add_review_connection)
55         self.contact_support.triggered.connect(self.contact_support_connecti
56
57
58     def contact_support_connection(self):
59         self.parent.tabs.setCurrentIndex(4)
60         print("Contact Support")
61
62     def add_review_connection(self):
63         self.parent.tabs.setCurrentIndex(3)
64         self.parent.reviews_tab.add_review_stacked()
65         print("add Trick")
66
67
68     def change_picture_connection(self):
69         self.parent.StatusBar.showMessage("Changing
70             Profile Picture... ")
71         self.parent.tabs.setCurrentIndex(0)
72         print("Find Picture")
73         path=QFileDialog.getOpenFileName()
74
75         if path=="":
76             print("Picture not changed.")
77             self.parent.StatusBar.showMessage("Profile
78                 Picture Not Changed.",2000)
79         else:
80             replace="\."
81             path=path.replace("/",replace[0])
82             destination="{0}{1}{2}".format(os.getcwd(),replace[0],"ProfileP
83             print(destination)
84             print(path)
85             shutil.copy2(path,destination)
86             self.connection=ProfileSQLConnections()
87             self.connection.change_picture(destination)
88             print(path)
89             self.parent.StatusBar.showMessage("Profile
90                 Picture Successfully Changed.",2000)
91
92     def add_trick_connection(self):
93         self.parent.tabs.setCurrentIndex(1)
94         self.parent.tricks_tab.add_trick_stacked()
95         print("add Trick")
96
97     def add_skatepark_connection(self):
98         self.parent.tabs.setCurrentIndex(2)
99         self.parent.skateparks_tab.view_add_skatepark()
```

4.11.4 Profile Widget

```
1 import sys
2 from PyQt4.QtGui import *
3 from PyQt4 import QtGui
4 from PyQt4.QtCore import *
5 from PyQt4.QtSql import *
6
7 from main_window import *
8 from profile_toolbar import *
9 from profile_picture import *
10 from profile_sql_connections import *
11
12
13 class DisplayProfileWidget(QWidget):
14     """A class to display the model and represent a
15         view on the profile tab"""
16
17     def __init__(self, parent):
18         super().__init__()
19         self.parent=parent
20
21         self.ProfileSQLConnections=ProfileSQLConnections()
22         selfToolBarWidgetLayout=QVBoxLayout()
23         self.LayoutWidget=QWidget()
24
25         self.HBoxLayout=QHBoxLayout()
26         self.LeftVBoxLayout=QVBoxLayout()
27         self.LeftHBoxLayout=QHBoxLayout()
28         self.MiddleVBoxLayout=QVBoxLayout()
29         self.RightVBoxLayout=QVBoxLayout()
30
31
32         self.display_profile_layout()
33         self.display_profile_toolbar_widget()
34         self.setLayout(selfToolBarWidgetLayout)
35         self.model=None
36
37         self.edit_button.clicked.connect(self.edit_button_clicked)
38         self.save_button.clicked.connect(self.save_button_clicked)
39
```

```
40
41
42
43
44
45     def display_profile_layout(self):
46         if not hasattr(self, "profile_picture"):
47             self.profile_picture=ProfilePicture()
48             self.profile_picture.setHorizontalScrollBarPolicy(1)
49             self.profile_picture.setVerticalScrollBarPolicy(1)
50             self.profile_picture.setMinimumSize(QSize(160,160))
51             self.profile_picture.setMaximumSize(QSize(160,160))
52             self.LeftVBoxLayout.addWidget(self.profile_picture)
53         if not hasattr(self, "edit_button"):
54             self.edit_button=QPushButton("Edit")
55             self.LeftHBoxLayout.addWidget(self.edit_button)
56         if not hasattr(self, "save_button"):
57             self.save_button=QPushButton("Save")
58             self.LeftHBoxLayout.addWidget(self.save_button)
59             self.LeftVBoxLayout.addLayout(self.LeftHBoxLayout)
60             self.HBoxLayout.addLayout(self.LeftVBoxLayout)
61
62
63
64
65         if not hasattr(self, "first_name"):
66             self.FirstName=self.ProfileSQLConnections.get_first_name()
67
68             self.first_name=QLineEdit(self.FirstName[0])
69             self.first_name.setReadOnly(True)
70             self.MiddleVBoxLayout.addWidget(self.first_name)
71         if not hasattr(self, "last_name"):
72             self.LastName=self.ProfileSQLConnections.get_last_name()
73
74             self.last_name=QLineEdit(self.LastName[0])
75             self.last_name.setReadOnly(True)
76             self.MiddleVBoxLayout.addWidget(self.last_name)
77         if not hasattr(self, "user_email"):
78             self.UserEmail=self.ProfileSQLConnections.get_email()
79             self.user_email=QLineEdit("{0}".format(self.UserEmail[0]))
80             self.user_email.setReadOnly(True)
81             self.MiddleVBoxLayout.addWidget(self.user_email)
82
83             self.HBoxLayout.addLayout(self.MiddleVBoxLayout)
84
85
```

```
86         if not hasattr(self,"recent_tricks"):
87             self.recent_tricks=QLabel("Recently
88             Completed Tricks")
89             self.RightVBoxLayout.addWidget(self.recent_tricks)
90             if not hasattr(self,"recent_tricks_list"):
91                 self.recent_tricks_list= QListWidget()
92                 #self.recent_tricks_list.addItem("Ollie")
93                 self.RightVBoxLayout.addWidget(self.recent_tricks_list)

94             self.HBoxLayout.addLayout(self.RightVBoxLayout)
95             self.LayoutWidget.setLayout(self.HBoxLayout)
96
97
98
99             def display_profile_toolbar_widget(self):
100                 if not hasattr(self,"profile_tool_bar"):
101                     self.profile_tool_bar=DisplayProfileToolbar(self)
102                     self.profile_tool_bar.changedPicture.connect(self.refresh_picture)
103                     self.ToolBarWidgetLayout.addWidget(self.profile_tool_bar)
104                     self.ToolBarWidgetLayout.addWidget(self.LayoutWidget)

105
106             def refresh_picture(self):
107                 print("Refresh Picture")
108                 self.profile_picture.picture()
109                 self.parent.StatusBar.showMessage("Profile
110                     Picture Successfully Changed.", 2000)

111             def edit_button_clicked(self):
112                 self.parent.StatusBar.showMessage("Edit Mode")
113                 self.change_name_edit()
114                 self.change_email_edit()

115
116             def change_name_edit(self):
117                 self.first_name.setReadOnly(False)
118                 self.last_name.setReadOnly(False)

119
120             def change_email_edit(self):
121                 self.user_email.setReadOnly(False)

122
123
124             def save_button_clicked(self):
125                 self.ProfileSQLConnections.change_name(self.first_name.text(),self.l
126                 self.ProfileSQLConnections.change_email(self.user_email.text())
127                 self.first_name.setReadOnly(True)
128                 self.last_name.setReadOnly(True)
129                 self.user_email.setReadOnly(True)
```

```
130     self.parent.StatusBar.clearMessage()
```

4.11.5 Profile Picture

```
1 import sys
2 import os
3 import sqlite3
4
5 from PyQt4.QtGui import *
6 from PyQt4 import QtGui
7 from PyQt4.QtCore import *
8 from profile_sql_connections import *
9
10 class ProfilePicture(QGraphicsView):
11     """This class provides a graphics view for the
12         profile picture"""
13     def __init__(self):
14         super().__init__()
15         self.PictureSQLConnection=ProfileSQLConnections()
16         self.scene=QGraphicsScene()
17
18         self.picture()
19
20
21
22     def picture(self):
23         self.FilePath=self.PictureSQLConnection.get_picture()
24         Picture=QPixmap("{0}".format(self.FilePath))
25         Picture=Picture.scaled(QSize(160,160))
26
27         self.profile_picture=(Picture)
28         self.scene.addPixmap(self.profile_picture)
29
30         print(self.scene.items())
31
32
33
34
35         self.setScene(self.scene)
```

4.11.6 Profile Toolbar

```
1 import sys
2 from PyQt4.QtGui import *
3 from PyQt4 import QtGui
4 from PyQt4.QtCore import *
5 from PyQt4.QtSql import *
6
7 from profile_widget import *
8 from profile_sql_connections import *
9 from main_window import *
10
11
12 class DisplayProfileToolbar(QToolBar):
13     """A class to create the profile tabs' toolbar"""
14     changedPicture=pyqtSignal()
15     def __init__(self,parent):
16         super().__init__()
17         self.parent=parent
18
19         self.change_picture=QAction("Change
20             Picture",self)
21
22
23         self.change_picture.triggered.connect(self.change_picture_connection)
24
25
26     def change_picture_connection(self):
27         self.parent.parent.StatusBar.showMessage("Changing
28             Profile Picture...")
29
30         print("Find Picture")
31         path=QFileDialog.getOpenFileName()
32         if path=="":
33             print("Picture not changed.")
34             self.parent.parent.StatusBar.showMessage("Profile
35                 Picture Not Changed.", 2000)
36         else:
37             replace="\."
38             path=path.replace("/",replace[0])
39             destination="{0}{1}{2}".format(os.getcwd(),replace[0],"ProfileP
40             rint(destination)
41             print(path)
42             shutil.copy2(path,destination)
43             self.connection=ProfileSQLConnections()
44             self.connection.change_picture(destination)
```

```
43         print(path)
44         self.changedPicture.emit()
```

4.11.7 Profile SQL Connections

```
1 import sqlite3
2 import shutil
3 import os
4 import sys
5
6
7
8
9
10 class ProfileSQLConnections:
11     """Handles the connection to the SQL database for
12     the profile tab"""
13
14     def __init__(self):
15         print("Profile SQL Connection")
16
17     def change_name(self, FirstName, LastName):
18         FirstName=FirstName
19         LastName=LastName
20         if (FirstName=="") or (LastName==""):
21             print("Name Not Changed")
22         else:
23             values=(FirstName, LastName, 1)
24             with
25                 sqlite3.connect("skateboard_progress_tracker.db")
26                 as db:
27                     cursor = db.cursor()
28                     sql="update User set FirstName=?,
29                         LastName=? where UserID=?"
30                     cursor.execute(sql,values)
31                     db.commit()
32
33
34     def change_email(self, Email):
35         Email=Email
36         if (Email==""):
37             print("Email Not Changed")
38         else:
```

```
36             values=(Email ,1)
37             with
38                 sqlite3.connect("skateboard_progress_tracker.db")
39                 as db:
40                     cursor = db.cursor()
41                     sql="update User set UserEmail=?"
42                         where UserID=?"
43                     cursor.execute(sql,values)
44                     db.commit()
45
46
47
48     def change_picture(self,FilePath):
49         FilePath=FilePath
50
51         values=(FilePath ,1)
52         with
53             sqlite3.connect("skateboard_progress_tracker.db")
54             as db:
55                 cursor = db.cursor()
56                 sql="update User set UserPicture=? where
57                     UserID=?"
58                 cursor.execute(sql,values)
59                 db.commit()
60                 print("Picture Changed")
61
62
63
64
65     def get_first_name(self):
66         with
67             sqlite3.connect("skateboard_progress_tracker.db")
68             as db:
69                 cursor=db.cursor()
70                 cursor.execute("select FirstName from
71                     User where UserID=?",(1,))
72                 FirstName=cursor.fetchone()
73                 print(FirstName)
74                 return FirstName
75
76     def get_last_name(self):
77         with
78             sqlite3.connect("skateboard_progress_tracker.db")
79             as db:
80                 cursor=db.cursor()
81                 cursor.execute("select LastName from
82                     User where UserID=?",(1,))
```

```
70             LastName=cursor.fetchone()
71             return LastName
72     def get_email(self):
73         with
74             sqlite3.connect("skateboard_progress_tracker.db")
75             as db:
76                 cursor=db.cursor()
77                 cursor.execute("select UserEmail from
78                     User where UserID=?",(1,))
79                 UserEmail=cursor.fetchone()
80                 return UserEmail
81     def get_picture(self):
82         FilePath = "{0}{1}".format(os.getcwd(),"\ProfilePicture.jpeg"))
83         print(FilePath)
84         return FilePath
```

4.11.8 Tricks Widget

```
1  from tricks_toolbar import *
2  from tricks_sql_connections import *
3
4  import os
5  import sys
6  import re
7
8  class DisplayTricksWidget(QWidget):
9      """A class to display the Tricks widget"""
10
11
12      def __init__(self,parent):
13          super().__init__()
14          self.parent=parent
15          self.RedBorder="border: 1px solid red;"
16          self.GreenBorder="border: 1px solid green;"
17          self.TrickFilePath="{0}\ProgramIcon.png".format(os.getcwd())
18          self.main_stacked_layout=QStackedLayout()
19          self.add_trick_VBoxLayout=QVBoxLayout()
20          self.add_trick_button_layout=QHBoxLayout()
21          self.add_trick_HBoxLayout=QHBoxLayout()
22          self.add_trick_widget=QWidget()
23          self.add_trick_table=QTableView()
24
25
26          self.stacked_layout=QStackedLayout()
```

```
27         self.results_table=QTableView()
28         self.results_table.setSelectionBehavior(QAbstractItemView.SelectRows)
29         self.results_layout=QVBoxLayout()
30         self.results_widget=QWidget()
31
32
33
34         self.LayoutWidget=QWidget()
35         selfToolBarWidgetLayout=QVBoxLayout()
36
37
38         self.open_connection()
39         self.display_results()
40         self.show_tricks_layout()
41
42
43
44         self.add_tricks()
45
46         self.display_tricks_layout()
47         self.display_tricks_toolbar_widget()
48         self.setLayout(self.ToolBarWidgetLayout)
49
50
51         self.trick_image.clicked.connect(self.image_button_clicked)
52         self.cancel_trick.clicked.connect(self.cancel_button_clicked)
53         self.save_trick.clicked.connect(self.save_button_clicked)
54         self.trick_name.textChanged.connect(self.validate_trick_name)
55         self.trick_description.textChanged.connect(self.validate_trick_descri
56         self.trick_obstacle.textChanged.connect(self.validate_trick_obstacle)
57         self.trick_tutorial.textChanged.connect(self.validate_trick_tutorial)
58
59     def keyPressEvent(self,event):
60         indexes=self.results_table.selectionModel().selectedRows()
61         for index in indexes:
62             Row=index.row()
63             if event.key()==Qt.Key_Delete:
64                 self.delete_trick_warning(Row)
65
66
67
68     def delete_trick_warning(self,Row):
69         self.delete_dialog=QDialog()
70         self.dialog_VBoxLayout=QVBoxLayout()
71         self.dialog_button_layout=QHBoxLayout()
72         if not hasattr(self,"delete_message"):
```

```
73         self.delete_message=QLabel("Are you sure  
74             you wish to delte this trick?")  
75         self.dialog_VBoxLayout.addWidget(self.delete_message)  
76     if not hasattr(self,"delete_cancel"):  
77         self.delete_cancel=QPushButton("Cancel")  
78         self.dialog_button_layout.addWidget(self.delete_cancel)  
79     if not hasattr(self,"delete_save"):  
80         self.delete_save=QPushButton("Save")  
81         self.dialog_button_layout.addWidget(self.delete_save)  
82     self.dialog_VBoxLayout.addLayout(self.dialog_button_layout)  
83     self.delete_dialog.setLayout(self.dialog_VBoxLayout)  
84     self.delete_dialog.show()  
85     self.delete_cancel.clicked.connect(self.delete_cancel_clicked)  
86     self.delete_save.clicked.connect(self.delete_save_clicked,Row)  
87  
88     def delete_cancel_clicked(self):  
89         self.delete_dialog.close()  
90         self.parent.StatusBar.showMessage("Trick Not  
91             Deleted",2000)  
92  
93     def delete_save_clicked(self,Row):  
94         self.delete_dialog.close()  
95         self.connection.delete_row(Row)  
96         query = self.connection.show_all_tricks()  
97         self.model.setQuery(query)  
98         self.parent.StatusBar.showMessage("Trick  
99             Successfully Deleted",2000)  
100  
101  
102     def validate_add_trick(self):  
103         TrickName=self.validate_trick_name()  
104         print(self.validate_trick_name())  
105         TrickDescription=self.validate_trick_description()  
106         TrickObstacle=self.validate_trick_obstacle()  
107         if self.trick_tutorial.text() == "":  
108             TrickTutorial=True  
109         else:  
110             TrickTutorial=self.validate_trick_tutorial()  
111  
112         if (TrickName==True) and  
113             (TrickDescription==True) and  
114             (TrickObstacle==True) and
```

```
    (TrickTutorial==True):
114        self.connection.add_trick_to_database(self.trick_difficulty.curr
115            ,self.trick_name.text(),self.trick_description.text(),self.tr
116            self.trick_tutorial.text())
117        self.parent.StatusBar.showMessage("Trick
118            Successfully Saved.",2000)
119        query = self.connection.show_all_tricks()
120        self.model.setQuery(query)
121        return True
122    else:
123        self.parent.StatusBar.showMessage("Not
124            all Fields are Valid.",2000)
125    return False
126
127
128
129
130
131
132
133
134
135
136
137
138    def validate_trick_name(self):
139        Text=self.trick_name.text()
140        TrickNameExpression=re.compile("^(?!\\s*$).+")
141        Match=TrickNameExpression.match(Text.upper())
142        if Match:
143            self.trick_name.setStyleSheet(self.GreenBorder)
144            return True
145        else:
146            self.trick_name.setStyleSheet(self.RedBorder)
147            return False
148
149    def validate_trick_description(self):
150        Text=self.trick_description.text()
151        TrickDescriptionExpression=re.compile("^(?!\\s*$).+")
152        Match=TrickDescriptionExpression.match(Text.upper())
153        if Match:
154            self.trick_description.setStyleSheet(self.GreenBorder)
```

```
155         return True
156     else:
157         self.trick_obstacle.setStyleSheet(self.RedBorder)
158         return False
159
160     def validate_trick_tutorial(self):
161         Text=self.trick_tutorial.text()
162         TrickTutorialExpression=re.compile("(?:.+?)?(?:\\/v\\/|watch\\/|\\?v=|\\&")
163         Match=TrickTutorialExpression.match(Text)
164         if Match:
165             self.trick_tutorial.setStyleSheet(self.GreenBorder)
166             return True
167         else:
168             self.trick_tutorial.setStyleSheet(self.RedBorder)
169             return False
170
171
172     def clear_trick_line_edit(self):
173         self.trick_name.clear()
174         self.trick_description.clear()
175         self.trick_obstacle.clear()
176         self.trick_tutorial.clear()
177         self.trick_difficulty.setCurrentIndex(0)
178
179
180     def display_results(self):
181         self.results_layout.addWidget(self.results_table)
182         self.results_widget.setLayout(self.results_layout)
183
184         self.stacked_layout.addWidget(self.results_widget)
185
186
187     def open_connection(self):
188         self.path="{0}{1}".format(os.getcwd(),"\skateboard_progress_tracker")
189         print(self.path)
190         self.connection=TricksSQLConnections(self.path)
191         self.connection.open_database()
192
193     def show_tricks_layout(self):
194
195         if self.connection != None:
196             self.query =
197                 self.connection.show_all_tricks()
198
199             self.show_results(self.query)
```

```
200         else:
201             print("A DB Connection must be opened")
202
203
204     def show_results(self,query):
205
206         self.model = QSqlQueryModel()
207         self.model.setQuery(query)
208         self.results_table.setModel(self.model)
209         self.results_table.show()
210
211
212
213
214
215
216     def display_tricks_layout(self):
217         self.stacked_layout.addWidget(self.add_trick_widget)
218
219         self.LayoutWidget.setLayout(self.stacked_layout)
220
221     def table_stacked(self):
222         self.stacked_layout.setCurrentIndex(0)
223
224     def add_trick_stacked(self):
225         self.stacked_layout.setCurrentIndex(1)
226
227
228
229
230
231     def display_tricks_toolbar_widget(self):
232         if not hasattr(self,"tricks_tool_bar"):
233             self.tricks_tool_bar=DisplayTricksToolbar(self)
234             self.ToolBarWidgetLayout.addWidget(self.tricks_tool_bar)
235             self.ToolBarWidgetLayout.addWidget(self.LayoutWidget)
236
237     def add_tricks(self):
238         self.add_trick_table.setModel(self.model)
239         if not hasattr(self,"trick_name"):
240             self.trick_name=QLineEdit()
241             self.trick_name.setPlaceholderText("Trick
242                                         name")
243
244             self.add_trick_VBoxLayout.addWidget(self.trick_name)
245         if not hasattr(self,"trick_description"):
```

```
245             self.trick_description=QLineEdit()
246             self.trick_description.setPlaceholderText("Trick
247                 Description")
248             self.add_trick_VBoxLayout.addWidget(self.trick_description)
249             if not hasattr(self,"trick_obstacle"):
250                 self.trick_obstacle=QLineEdit()
251                 self.trick_obstacle.setPlaceholderText("Trick
252                     Obstacle")
253                 self.add_trick_VBoxLayout.addWidget(self.trick_obstacle)
254             if not hasattr(self,"trick_image"):
255                 self.trick_image=QPushButton("Browse
256                     Trick Image")
257                 self.add_trick_VBoxLayout.addWidget(self.trick_image)
258             if not hasattr(self,"trick_tutorial"):
259                 self.trick_tutorial=QLineEdit()
260                 self.trick_tutorial.setPlaceholderText("Trick
261                     Tutorial Link")
262                 self.add_trick_VBoxLayout.addWidget(self.trick_tutorial)
263             if not hasattr(self,"trick_difficulty"):
264                 self.trick_difficulty=QComboBox()
265                 self.trick_difficulty.addItem("Easy")
266                 self.trick_difficulty.addItem("Medium")
267                 self.trick_difficulty.addItem("Hard")
268                 self.add_trick_VBoxLayout.addWidget(self.trick_difficulty)
269             if not hasattr(self,"cancel_trick"):
270                 self.cancel_trick=QPushButton("Cancel")
271                 self.add_trick_button_layout.addWidget(self.cancel_trick)
272             if not hasattr(self,"save_trick"):
273                 self.save_trick=QPushButton("Save")
274                 self.add_trick_button_layout.addWidget(self.save_trick)
275             self.add_trick_VBoxLayout.setLayout(self.add_trick_button_layout)
276             self.add_trick_HBoxLayout.setLayout(self.add_trick_VBoxLayout)
277             self.add_trick_HBoxLayout.addWidget(self.add_trick_table)
278             self.add_trick_widget.setLayout(self.add_trick_HBoxLayout)

279         def cancel_button_clicked(self):
280             print("Cancel")
281             self.table_stacked()
282             self.clear_trick_line_edit()

283         def save_button_clicked(self):
284             Valid=self.validate_add_trick()
285             if Valid:
286                 self.clear_trick_line_edit()
287             else:
```

```
287         print()
288
289
290
291
292
293
294     def image_button_clicked(self):
295         print("hi")
296         path=QFileDialog.getOpenFileName()
297         if path=="":
298             print("No picture Added")
299         else:
300
301             replace="\."
302             path=path.replace("/",replace[0])
303             print(path)
304             self.TrickFilePath=path
```

4.11.9 Tricks Toolbar

```
1 import sys
2 from PyQt4.QtGui import *
3 from PyQt4 import QtGui
4 from PyQt4.QtCore import *
5 from PyQt4.QtSql import *
6
7
8 class DisplayTricksToolbar(QToolBar):
9     def __init__(self, parent):
10         super().__init__()
11         self.parent=parent
12
13         self.add_trick=QAction("Add Trick",self)
14
15         self.addAction(self.add_trick)
16
17         #connections
18         self.add_trick.triggered.connect(self.add_trick_connection)
19
20
21     def add_trick_connection(self):
22         self.parent.add_trick_stacked()
```

4.11.10 Tricks SQL Connections

```
1 import sqlite3
2 from PyQt4.QtSql import *
3
4 class TricksSQLConnections:
5     """Handles the connection to the SQL database for
6     the tricks tab"""
7
8     def __init__(self, path):
9         self.path = path
10        self.db = None
11
12    def delete_row(self, Row):
13        values = (Row + 1,)
14        with
15            sqlite3.connect("skateboard_progress_tracker.db")
16                as db:
17                    cursor = db.cursor()
18                    sql = "DELETE FROM Trick WHERE TrickID=?"
19                    cursor.execute(sql, values)
20                    db.commit()
21                    print()
22
23    def open_database(self):
24        if self.db:
25            self.close_database()
26
27        self.db = QSqlDatabase.addDatabase("QSQLITE")
28        self.db.setDatabaseName(self.path)
29
30        opened_ok = self.db.open()
31        return opened_ok
32
33    def show_all_tricks(self):
34        query = QSqlQuery()
35        query.prepare(""" SELECT * FROM Trick""")
36        query.exec_()
37        return query
38
39    def add_trick_to_database(self, DifficultyID, TrickName, TrickDescription, TrickImage):
40        print("hi2")
```

```
40     values=(DifficultyID , "Ben Keppie",
41             TrickName , TrickDescription ,
42             TrickObstacle , TrickImage ,
43             TrickTutorialLink)
44     with
45         sqlite3.connect("skateboard_progress_tracker.db")
46             as db:
47                 cursor = db.cursor()
48                 sql="insert into Trick(DifficultyID ,
49                     TrickCreator,TrickName,TrickDescription ,TrickObstacle ,TrickIm
50                     values (?,?,?,?,?,?)"
51                 cursor.execute(sql,values)
52                 db.commit()
53                 print()
54                 print("Trick Successfully Created .")
55                 print()
```

4.11.11 Skateparks Widget

```
1  from PyQt4.QtWebKit import *
2
3  from skateparks_toolbar import *
4  from skateparks_sql_connections import *
5  from skatepark_view_only import *
6  import re
7
8
9
10
11 from google_maps_view import *
12 from google_maps_test import *
13
14 #GOOGLE MAPS API KEY:
15 #AIzaSyC5RcJ7vLSEYF32KqDusnuRcLJiHW8EbDg
16
17 class DisplaySkateparksWidget(QWidget):
18     """A class to display the Skatepark Widget"""
19
20     def __init__(self,parent):
21         super().__init__()
22         self.parent=parent
23
24         self.RedBorder="border: 1px solid red;"
```

```
25     self.GreenBorder="border: 1px solid green;"  
26     self.SkateparksSQLConnections=SkateparksSQLConnections()  
27     self.coordinate_change=None  
28  
29     self.add_skatepark_VBoxLayout=QVBoxLayout()  
30     self.add_skatepark_HBoxLayout=QHBoxLayout()  
31     self.add_skatepark_button_layout=QHBoxLayout()  
32     self.add_skatepark_widget=QWidget()  
33  
34  
35     self.LayoutWidget=QWidget()  
36     self.ToolBarWidgetLayout=QVBoxLayout()  
37     self.VBoxLayout=QVBoxLayout()  
38  
39     self.add_skatepark()  
40     self.display_skateparks_toolbar_widget()  
41     self.setLayout(self.ToolBarWidgetLayout)  
42  
43     self.cancel_skatepark.clicked.connect(self.cancel_button_clicked)  
44     self.save_skatepark.clicked.connect(self.save_button_clicked)  
45     self.skatepark_name.textChanged.connect(self.validate_skatepark_name)  
46     self.skatepark_description.textChanged.connect(self.validate_skatepark_description)  
47     self.skatepark_latitude.textChanged.connect(self.validate_latitude)  
48     self.skatepark_longitude.textChanged.connect(self.validate_longitude)  
49  
50     def validate_skatepark_description(self):  
51         Text=self.skatepark_description.text()  
52         SkateparkDescriptionExpression=re.compile("^(?!\\s*$).+")  
53         Match=SkateparkDescriptionExpression.match(Text.upper())  
54         if Match:  
55             self.skatepark_description.setStyleSheet(self.GreenBorder)  
56             return True  
57         else:  
58             self.skatepark_description.setStyleSheet(self.RedBorder)  
59             return False  
60  
61     def validate_skatepark_name(self):  
62         Text=self.skatepark_name.text()  
63         SkateparkNameExpression=re.compile("^(?!\\s*$).+")  
64         Match=SkateparkNameExpression.match(Text.upper())  
65         if Match:  
66             self.skatepark_name.setStyleSheet(self.GreenBorder)  
67             return True  
68         else:  
69             self.skatepark_name.setStyleSheet(self.RedBorder)  
70             return False
```

```
71     def validate_longitude(self):
72         Text=self.skatepark_longitude.text()
73         if Text=="":
74             self.skatepark_longitude.setStyleSheet(self.RedBorder)
75             return False
76         else:
77             self.skatepark_longitude.setStyleSheet(self.GreenBorder)
78             return True
79     def validate_latitude(self):
80         Text=self.skatepark_latitude.text()
81         print(Text)
82         if Text=="":
83             self.skatepark_latitude.setStyleSheet(self.RedBorder)
84             return False
85         else:
86             self.skatepark_latitude.setStyleSheet(self.GreenBorder)
87             return True
88
89
90
91     def validate_add_skatepark(self):
92         SkateparkName=self.validate_skatepark_name()
93         SkateparkDescription=self.validate_skatepark_description()
94         SkateparkLatitude=self.validate_latitude()
95         print(SkateparkLatitude)
96         SkateparkLongitude=self.validate_longitude()
97         if (SkateparkName==True) and
98             (SkateparkDescription==True) and
99             (SkateparkLatitude==True) and
100             (SkateparkLongitude==True):
101             self.parent.StatusBar.showMessage("Skatepark
102                 Successfully Saved.",2000)
103             return True
104         else:
105             self.parent.StatusBar.showMessage("Not
106                 all Fields are Valid.",2000)
107             return False
108
109     def cancel_button_clicked(self):
110         print("Cancel")
111         self.clear_skatepark_line_edit()
112         self.skatepark_name.hide()
113         self.skatepark_description.hide()
114         self.skatepark_longitude.hide()
115         self.skatepark_latitude.hide()
116         self.cancel_skatepark.hide()
```

```
112         self.save_skatepark.hide()
113         self.add_skatepark_map.delete_all_markers()
114         self.add_skatepark_map.get_marker_coordinates()
115
116     def save_button_clicked(self):
117         print("Save")
118         Valid=self.validate_add_skatepark()
119         if Valid:
120             self.SkateparksSQLConnections.add_skatepark(self.skatepark_name.)
121             self.clear_skatepark_line_edit()
122             self.add_skatepark_map.delete_all_markers()
123             self.add_skatepark_map.get_marker_coordinates()
124
125         else:
126             print()
127
128     def view_add_skatepark(self):
129         self.skatepark_name.show()
130         self.skatepark_description.show()
131         self.skatepark_longitude.show()
132         self.skatepark_latitude.show()
133         self.cancel_skatepark.show()
134         self.save_skatepark.show()
135
136
137     def clear_skatepark_line_edit(self):
138         self.skatepark_name.clear()
139         self.skatepark_description.clear()
140         self.skatepark_latitude.clear()
141         self.skatepark_longitude.clear()
142
143     def fill_line_edits(self,LastMarker):
144         self.latitude_coor=str(LastMarker[0])
145         self.longitude_coor=str(LastMarker[1])
146         self.skatepark_latitude.setText(self.latitude_coor)
147         self.skatepark_longitude.setText(self.longitude_coor)
148
149
150
151     def add_skatepark(self):
152         if not hasattr(self,"skatepark_name"):
153             self.skatepark_name=QLineEdit()
154             self.skatepark_name.setPlaceholderText("Skatepark
155                                         Name")
156             self.skatepark_name.hide()
```

```
157         self.add_skatepark_VBoxLayout.addWidget(self.skatepark_name)
158     if not hasattr(self, "skatepark_description"):
159         self.skatepark_description=QLineEdit()
160         self.skatepark_description.setPlaceholderText("Skatepark
161             Description")
162         self.skatepark_description.hide()
163         self.add_skatepark_VBoxLayout.addWidget(self.skatepark_descripti
164
165         if not hasattr(self, "skatepark_latitude"):
166             self.skatepark_latitude=QLineEdit()
167             self.skatepark_latitude.setPlaceholderText("Latitude")
168             self.skatepark_latitude.setReadOnly(True)
169             self.skatepark_latitude.hide()
170             self.add_skatepark_VBoxLayout.addWidget(self.skatepark_latitude)
171         if not hasattr(self, "skatepark_longitude"):
172             self.skatepark_longitude=QLineEdit()
173             self.skatepark_longitude.setPlaceholderText("Longitude")
174             self.skatepark_longitude.setReadOnly(True)
175             self.skatepark_longitude.hide()
176             self.add_skatepark_VBoxLayout.addWidget(self.skatepark_longitude)
177
178         if not hasattr(self, "cancel_skatepark"):
179             self.cancel_skatepark=QPushButton("Cancel")
180             self.cancel_skatepark.hide()
181             self.add_skatepark_button_layout.addWidget(self.cancel_skatepark)
182         if not hasattr(self, "save_skatepark"):
183             self.save_skatepark=QPushButton("Save")
184             self.save_skatepark.hide()
185             self.add_skatepark_button_layout.addWidget(self.save_skatepark)
186
187         if not hasattr(self, "add_skatepark_map"):
188             self.add_skatepark_map=ViewOnlyMap(self)
189
190
191
192         self.add_skatepark_VBoxLayout.addLayout(self.add_skatepark_button_la
193         self.add_skatepark_HBoxLayout.addLayout(self.add_skatepark_VBoxLayout)
194
195         self.add_skatepark_HBoxLayout.addWidget(self.add_skatepark_map)
196
197         self.LayoutWidget.setLayout(self.add_skatepark_HBoxLayout)
198
199
200
201
```

```
202
203     def display_skateparks_toolbar_widget(self):
204         if not hasattr(self, "skateparks_tool_bar"):
205             self.skateparks_tool_bar=DisplaySkateparksToolbar(self)
206             self.ToolBarWidgetLayout.addWidget(self.skateparks_tool_bar)
207             selfToolBarWidgetLayout.addWidget(self.LayoutWidget)
```

4.11.12 Skateparks Map

```
1  from PyQt4.QtWebKit import *
2  import sqlite3
3  from PyQt4.QtSql import *
4  import time
5
6  #      my API key =
7  #      "AIzaSyC5RcJ7vLSEYF32KqDusnuRcLJiHW8EbDg"
8
9  class CustomQWebPage(QWebPage):
10    """A class to act as the webpage for the google
11       maps module"""
12    def __init__(self):
13        super().__init__()
14        print("QWebPage constructor")
15
16    def javaScriptConsoleMessage(self,message,lineNumber,sourceID):
17        #An overridden method to display a javascript
18        #console message
19        print()
20        print(message,lineNumber,sourceID)
21        print("javascript console message^")
22
23
24    def __init__(self,parent):
25        super().__init__()
26        self.parent=parent
27
28    #Changing web settings to access certain
29    #functionality
```

```
30         self.settings().setAttribute(QWebSettings.JavascriptEnabled,
31             True)
32         self.settings().setAttribute(QWebSettings.JavascriptCanOpenWindows,
33             True)
34         self.settings().setAttribute(QWebSettings.JavascriptCanAccessClipboard,
35             True)
36         self.settings().setAttribute(QWebSettings.DeveloperExtrasEnabled,
37             True)
38
39
40
41         self.set_code()
42         print("Code set")
43
44
45     def mousePressEvent(self, event):
46         super().mousePressEvent(event)
47         if not
48             self.parent.skatepark_name.isReadOnly():
49                 self.get_last_marker()
50             else:
51                 print()
52                 print("In view only mode.")
53                 print()
54
55     def get_last_marker(self):
56         self.LastMarker=
57             self.CustomPage.mainFrame().evaluateJavaScript("GetMarkers()")
58         print(self.LastMarker)
59         self.parent.fill_line_edits(self.LastMarker)
60
61
62
63
64
65     def handle_load_finished(self,ok):
66         if ok:
67             self.get_marker_coordinates()
68         else:
69             print()
```

```
70
71
72
73
74
75     def get_marker_coordinates(self):
76         with
77             sqlite3.connect("skateboard_progress_tracker.db")
78                 as db:
79                     cursor=db.cursor()
80                     sql="select SkateparkLatitude ,
81                         SkateparkLongitude ,
82                         SkateparkDescription , SkateparkName
83                         from Skatepark"
84                     cursor.execute(sql)
85                     self.Coordinates=cursor.fetchall()
86                     for coordinate in self.Coordinates:
87                         Name=str(coordinate[3])
88                         Name=',{0}'.format(Name)
89
90
91
92
93
94
95
96
97     def set_code(self):
98
99
100            self.html='''<!DOCTYPE html>
101            <html>
102                <head>
103                    <meta name="viewport" content="initial-scale=1.0,
104                        user-scalable=no">
105                    <meta charset="utf-8">
106                    <title>Simple markers</title>
107                    <style>
108                        html , body , #map-canvas {
```

```
108         height: 100%;  
109         width: 100%  
110         margin: 0px;  
111         padding: 0px  
112     }  
113 </style>  
114 <script  
    src="https://maps.googleapis.com/maps/api/js?key=AIzaSyC5RcJ7vLSEYF32  
<script>  
116  
117     var map;  
118     var markers = [];  
119     var results = [];  
120     var coords = [];  
121     var highestLevel;  
122  
123  
124     function initialize() {  
125  
126  
127         var Centre = new  
            google.maps.LatLng(52.20255705185695,0.1373291015625);  
128         var mapOptions = {  
129             zoom: 8,  
130             minZoom: 3,  
131             center: Centre,  
132         }  
133         map = new  
            google.maps.Map(document.getElementById('map-canvas'),mapOptions)  
134  
135         google.maps.event.addListener(map, 'click',  
            function(event) {  
136             AddMarker(event.latLng);  
137         });  
138     }  
139  
140  
141     function  
        MarkersFromDatabase(SkateparkLat,SkateparkLng,SkateparkDescripti  
        SkateparkName) {  
142  
143         var Skatepark = new  
            google.maps.LatLng(SkateparkLat,SkateparkLng);  
144  
145
```

```
146     AddMarker(Skatepark ,SkateparkDescription ,
147                 SkateparkName); }
148
149
150
151     function AddMarker(Location ,Description ,
152                           SkateparkName) {
153
154         var marker = new google.maps.Marker({
155             title: 'Test',
156             position: Location ,
157             animation: google.maps.Animation.DROP ,
158             map: map
159
160         });
161         //markers.push(marker);
162         var lat = marker.getPosition().lat();
163         var lng = marker.getPosition().lng();
164         markers.push({"Object":marker , "Lat":lat , "Lng":lng ,
165                     "Desc":Description});
166
167         var contentString = ('<div id="content"><div
168             id="siteNotice"></div> <h1
169             id="firstHeading" class="firstHeading">' +
170             SkateparkName + '</h1> <div
171             id="bodyContent"><p>' + Description
172             +'</p></div></div>');
173
174         var infowindow = new google.maps.InfoWindow({
175             content: contentString
176         });
177
178         google.maps.event.addListener(marker ,
179             'rightclick' , function(event) {
180             marker.setMap(null);
181         });
182         google.maps.event.addListener(marker ,
183             'mouseover' , function(event) {
184             infowindow.open(map ,marker);
185         });
186         google.maps.event.addListener(marker , 'mouseout' ,
187             function(event){
188             infowindow.close(map ,marker)
189         });
190     }
```

```
181
182
183     }
184
185
186     function GetMarkers(){
187         var Longitude=markers[markers.length - 1]["Lng"]
188         var Latitude=markers[markers.length - 1]["Lat"]
189         var coors=[Latitude,Longitude]
190         return coors;
191     }
192
193
194     function DeleteMarkers(){
195         markers = [];
196         initialize();
197     }
198
199     google.maps.event.addDomListener(window, 'load',
200         initialize);
201         </script>
202     </head>
203     <body>
204         <div id="map-canvas"></div>
205     </body>
206 </html> '''
207         self.setHtml(self.html)
```

4.11.13 Skateparks Toolbar

```
1 import sys
2 from PyQt4.QtGui import *
3 from PyQt4 import QtGui
4 from PyQt4.QtCore import *
5 from PyQt4.QtSql import *
6
7
8 class DisplaySkateparksToolbar(QToolBar):
9     """A class to represent the skateparks tab
10        toolbar"""
11
12     def __init__(self,parent):
13         super().__init__()
14         self.parent=parent
```

```

14         self.add_skatepark=QAction("Add
15             Skatepark",self)
16
17         self.addAction(self.add_skatepark)
18
19         self.add_skatepark.triggered.connect(self.add_skatepark_connection)
20
21     def add_skatepark_connection(self):
22         print("Add Skatepark")
23         self.parent.view_add_skatepark()

```

4.11.14 Skateparks SQL Connections

```

1 import sqlite3
2
3 class SkateparksSQLConnections:
4     """Handles the connection to the SQL database for
5        the skateparks tab"""
6
7     def __init__(self):
8         print("Skatepark SQL Connection")
9
10    def
11        add_skatepark(self,SkateparkName,SkateparkDescription,Latitude,Longitude)
12        values=(SkateparkName,SkateparkDescription,Latitude,Longitude)
13        with
14            sqlite3.connect("skateboard_progress_tracker.db")
15            as db:
16                cursor = db.cursor()
17                sql="insert into
18                    Skatepark(SkateparkName,SkateparkDescription,
19                        SkateparkLatitude, SkateparkLongitude)
20                        values (?,?,?,?,?)"
21                cursor.execute(sql,values)
22                db.commit()
23                print()
24                print("Skatepark Successfully Created.")
25                print()

```

4.11.15 Reviews Widget

```

1 from reviews_toolbar import *

```

```
2 from reviews_sql_connections import *
3
4 import os
5 import sys
6 import re
7 class DisplayReviewsWidget(QWidget):
8     """A class to display the reviews widget"""
9
10    def __init__(self, parent):
11        super().__init__()
12        self.parent=parent
13
14
15        self.RedBorder="border: 1px solid red;"
16        self.GreenBorder="border: 1px solid green;"
17
18        self.main_stacked_layout=QStackedLayout()
19
20        self.add_review_VBoxLayout=QVBoxLayout()
21        self.add_review_button_layout=QHBoxLayout()
22        self.add_review_HBoxLayout=QHBoxLayout()
23        self.add_review_widget=QWidget()
24        self.add_review_table=QTableView()
25
26        self.stacked_layout=QStackedLayout()
27        self.results_table=QTableView()
28        self.results_layout=QVBoxLayout()
29        self.results_widget=QWidget()
29
30
31        self.LayoutWidget=QWidget()
32        selfToolBarWidgetLayout=QVBoxLayout()
33
34        self.open_connection()
35        self.display_results()
36        self.show_review_layout()
37
38
39
40        self.add_review()
41
42        self.display_review_layout()
43        self.display_review_toolbar_widget()
44        self.setLayout(self.ToolBarWidgetLayout)
45
46        self.cancel_review.clicked.connect(self.cancel_button_clicked)
47        self.save_review.clicked.connect(self.save_button_clicked)
```

```
48             self.review_type.currentIndexChanged.connect(self.update_review_boxe
49
50     def cancel_button_clicked(self):
51         self.clear_review_line_edit()
52         self.table_stacked()
53
54
55
56
57     def save_button_clicked(self):
58         Valid=self.validate_add_review()
59         if Valid:
60             self.clear_review_line_edit()
61         else:
62             print()
63     def validate_add_review(self):
64         ReviewType=self.validate_review_type()
65         ReviewSize=self.validate_review_size()
66         ReviewName=self.validate_review_name()
67         ReviewRating=self.validate_review_rating()
68         ReviewReview=self.validate_review_review()
69         if (ReviewType==True) and (ReviewSize==True)
70             and (ReviewName==True) and
71             (ReviewName==True) and
72             (ReviewRating==True) and
73             (ReviewReview==True):
74             self.connection.add_review_to_database(self.review_type.currentT
75                 self.review_brand.currentText(),self.review_name.text(),self.
76                 self.review_review.text())
77                 query=self.connection.show_all_reviews()
78                 self.model.setModel(query)
79                 return True
80             else:
81                 print("Not all fields valid")
82                 return False
83
84
85
86
87     def validate_review_type(self):
88         return True
89
90     def validate_review_size(self):
91         return True
92     def validate_review_name(self):
93         return True
94     def validate_review_brand(self):
95         return True
96     def validate_review_rating(self):
```

```
88         return True
89     def validate_review_review(self):
90         return True
91
92     def clear_review_line_edit(self):
93         self.review_type.setCurrentIndex(0)
94         self.review_size.setCurrentIndex(0)
95         self.review_brand.setCurrentIndex(0)
96         self.review_name.clear()
97         self.review_rating.setCurrentIndex(0)
98         self.review_review.clear()
99         self.review_review.setText("Review")
100
101
102
103
104
105     def open_connection(self):
106         self.path = "{0}{1}".format(os.getcwd(), "\skateboard_progress_tracker")
107         self.connection = ReviewsSQLConnections(self.path)
108         self.connection.open_database()
109
110     def display_results(self):
111         self.results_layout.addWidget(self.results_table)
112         self.results_widget.setLayout(self.results_layout)
113
114         self.stacked_layout.addWidget(self.results_widget)
115
116
117
118
119     def show_review_layout(self):
120         if self.connection != None:
121             self.query =
122                 self.connection.show_all_reviews()
123
124             self.show_results(self.query)
125         else:
126             print("A DB Connection must be opened")
127
128     def show_results(self, query):
129         self.model = QSqlQueryModel()
130         self.model.setQuery(query)
131         self.results_table.setModel(self.model)
132         self.results_table.show()
```

```
133     def update_review_boxes(self):
134         self.review_size.clear()
135         self.review_brand.clear()
136         self.review_size.addItem("-Select a Size-")
137         self.review_brand.addItem("-Select a Brand-")
138
139
140         SizeOptions=self.size_options_check(self.review_type.currentText())
141         for Size in SizeOptions:
142             self.review_size.addItem(Size)
143
144     def type_options_check(self):
145         self.type_options=self.connection.get_all_product_type()
146         #self.type_options=["Deck","Trucks","Wheels","Bearings","GRIPTAPE","BOLTS"]
147         return self.type_options
148
149     def size_options_check(self,Type):
150         pass
151     ##        if Type=="Deck":
152     ##            self.size_options=
153     ##        elif Type=="Trucks":
154     ##            self.size_options=
155     ##        elif Type=="Wheels":
156     ##            self.size_options=
157     ##        elif Type=="Bearings":
158     ##            self.size_options=
159     ##        elif Type=="GRIPTAPE":
160     ##            self.size_options=
161     ##        elif Type=="BOLTS":
162     ##            self.size_options=
163     ##        else:
164     ##            self.size_options=[]
165     ##        return self.size_options
166
167     def brand_options_check(self,):
168         self.brand_options=self.connection.get_all_product_brand()
169
170         return self.brand_options
171
172
173
174     def add_review(self):
175         self.add_review_table.setModel(self.model)
176
177
178         if not hasattr(self,"review_type"):
```

```
179         self.review_type=QComboBox()
180         self.review_type.addItem("-Select a
181             Type-")
182         TypeOptions=self.type_options_check()
183         self.review_type.setModel(TypeOptions)
184
185         self.add_review_VBoxLayout.addWidget(self.review_type)
186     if not hasattr(self,"review_size"):
187         self.review_size=QComboBox()
188         self.review_size.addItem("-Select a
189             Size-")
190
191         self.add_review_VBoxLayout.addWidget(self.review_size)
192     if not hasattr(self,"review_brand"):
193         self.review_brand=QComboBox()
194         self.review_brand.addItem("-Select a
195             Brand-")
196
197         self.add_review_VBoxLayout.addWidget(self.review_brand)
198     if not hasattr(self,"review_name"):
199         self.review_name=QLineEdit()
200         self.review_name.setPlaceholderText("Product
201             Name")
202
203         self.add_review_VBoxLayout.addWidget(self.review_name)
204     if not hasattr(self,"review_rating"):
205         self.review_rating=QComboBox()
206         self.review_rating.addItem("-Select a
207             Rating-")
208
209         for count in range(1,6):
210             self.review_rating.addItem(str(count))
211         self.add_review_VBoxLayout.addWidget(self.review_rating)
212     if not hasattr(self,"review_review"):
213         self.review_review=QTextEdit("Review")
214
215         self.add_review_VBoxLayout.addWidget(self.review_review)
216
217
218         self.add_review_VBoxLayout.setLayout(self.add_review_button_layout)
219         self.add_review_HBoxLayout.setLayout(self.add_review_VBoxLayout)
```

```
220         self.add_review_HBoxLayout.addWidget(self.add_review_table)
221         self.add_review_widget.setLayout(self.add_review_HBoxLayout)
222
223     def display_review_layout(self):
224         self.stacked_layout.addWidget(self.add_review_widget)
225
226         self.LayoutWidget.setLayout(self.stacked_layout)
227
228
229
230     def display_review_toolbar_widget(self):
231         if not hasattr(self, "reviews_tool_bar"):
232             self.reviews_tool_bar=DisplayReviewsToolbar(self)
233             selfToolBarWidgetLayout.addWidget(self.reviews_tool_bar)
234             selfToolBarWidgetLayout.addWidget(self.LayoutWidget)
235
236     def table_stacked(self):
237         self.stacked_layout.setCurrentIndex(0)
238
239     def add_review_stacked(self):
240         self.stacked_layout.setCurrentIndex(1)
```

4.11.16 Reviews Toolbar

```
1 import sys
2 from PyQt4.QtGui import *
3 from PyQt4 import QtGui
4 from PyQt4.QtCore import *
5 from PyQt4.QtSql import *
6
7
8 class DisplayReviewsToolbar(QToolBar):
9     """A class to create the toolbar for the review
10    tab"""
11
12    def __init__(self, parent):
13        super().__init__()
14        self.parent=parent
15
16        self.add_review=QAction("Add Review",self)
17
18        self.addAction(self.add_review)
19
20        #connections
21        self.add_review.triggered.connect(self.add_review_connection)
```

```
20
21
22     def add_review_connection(self):
23         self.parent.add_review_stacked()
24         print()
25         print("Add Review")
26         print()
```

4.11.17 Reviews SQL Connections

```
1 import sqlite3
2 from PyQt4.QtSql import *
3
4 class ReviewsSQLConnections:
5     """Handles the connection to the SQL database for
6        the reviews tab"""
7
8     def __init__(self, path):
9         self.path = path
10        self.db = None
11
12    def open_database(self):
13        if self.db:
14            self.close_database()
15
16        self.db = QSqlDatabase.addDatabase("QSQLITE")
17        self.db.setDatabaseName(self.path)
18
19        opened_ok = self.db.open()
20        return opened_ok
21
22    def show_all_reviews(self):
23        query = QSqlQuery()
24        query.prepare(""" SELECT
25            ProductType, ProductName, ReviewDescription, ReviewRating, ProductBrand
26            FROM review, product,
27            productbrand, producttype, productsize""")
28        query.exec_()
29        return query
30
31    def get_all_product_type(self):
32        query = QSqlQuery()
33        query.prepare("""SELECT ProductType FROM
34            ProductType""")
```

```
30         query.exec_()
31     model= QSqlQueryModel().setQuery(query)
32     return model
33
34
35
36
37     def get_deck_sizes(self):
38         pass
39
40     def get_trucks_sizes(self):
41         pass
42
43     def get_wheels_sizes(self):
44         pass
45
46     def get_bearings_sizes(self):
47         pass
48
49     def get_griptape_sizes(self):
50         pass
51
52     def get_bolts_sizes(self):
53         pass
54
55     def get_all_product_brand(self):
56         pass
57
58     def
59         add_review_to_database(self,ReviewType,ReviewSize,ReviewBrand,ReviewN
```

4.11.18 Support Widget

```
1 import smtplib
2
3 from PyQt4.QtGui import *
4 from PyQt4 import QtGui
5 from PyQt4.QtCore import *
6
7 from email.mime.text import MIMEText
8
9 class DisplaySupportWidget(QWidget):
```

```
10     """A class to display the model and represent a
11         view on the support tab"""
12
13     def __init__(self, parent):
14         super().__init__()
15         self.parent=parent
16         self.VBoxLayout=QVBoxLayout()
17         self.form_layout=QGridLayout()
18         self.form_widget=QWidget()
19
20         self.display_support_layout()
21
22         self.setLayout(self.VBoxLayout)
23
24         self.submit_button.pressed.connect(self.send_email)
25
26     def send_email(self):
27         msg=MIMEMultipart(self.form_email_line_edit.text()+
28                           self.form_message_line_edit.toPlainText())
29         print(msg)
30         msg["Subject"]="Skateboard Progress Tracker
31                         Support"
32         msg["From"]="SkateboardProgressTracker@gmail.com"
33         msg["To"]="BenKeppie@hotmail.co.uk"
34
35         Send=smtplib.SMTP("smtp.gmail.com")
36         Send.sendmail(msg["From"],msg["To"],msg)
37         Send.quit
38
39     def display_support_layout(self):
40         if not hasattr(self, "developer"):
41             self.developer=QLabel("Application
42                               Developed By: Ben Keppie")
43             self.VBoxLayout.addWidget(self.developer)
44
45         if not hasattr(self, "form_name"):
46             self.form_name=QLabel("Name: ")
47             self.form_layout.addWidget(self.form_name,0,0)
48         if not hasattr(self,"form_name_line_edit"):
49             self.form_name_line_edit=QLineEdit()
50             self.form_layout.addWidget(self.form_name_line_edit,0,1)
51
52         if not hasattr(self,"form_email"):
53             self.form_email=QLabel("Email Address: ")
54             self.form_layout.addWidget(self.form_email,1,0)
55         if not hasattr(self,"form_email_line_edit"):
```

```

52         self.form_email_line_edit=QLineEdit()
53         self.form_layout.addWidget(self.form_email_line_edit,1,1)
54
55     if not hasattr(self,"form_message"):
56         self.form_message=QLabel("Message: ")
57         self.form_layout.addWidget(self.form_message,2,0)
58     if not hasattr(self,"form_message_line_edit"):
59         self.form_message_line_edit=QTextEdit()
60         #print(self.form_message_line_edit.toPlainText())
61         self.form_layout.addWidget(self.form_message_line_edit,2,1)
62
63     self.form_widget.setLayout(self.form_layout)
64     self.VBoxLayout.addWidget(self.form_widget)
65
66     if not hasattr(self,"submit_button"):
67         self.submit_button=QPushButton("Submit")
68         self.VBoxLayout.addWidget(self.submit_button)

```

4.11.19 CLI Menu

```

1  from database import *
2  from database_table_menu import *
3  from get_menu_option import *
4
5
6  def display_menu():
7      print()
8      print("Skateboard Progress Tracker Database
          Management")
9      print()
10     print("1. (Re)Create Database")
11     print("2. Edit Profile Table")
12     print("3. Edit Trick Table")
13     print("4. Edit Skatepark Table")
14     print("5. Edit Review Table")
15     print("0. Exit")
16
17
18 def main():
19     Finished=False
20     while not Finished:
21         display_menu()
22         Choice=get_menu_option()
23         if Choice==0:

```

```
24             Finished=True
25             print()
26         elif Choice==1:
27             Finished=database_creator()
28
29
30         elif Choice==2:
31             Finished=profile_table()
32
33         elif Choice==3:
34             Finished=trick_table()
35
36         elif Choice==4:
37             Finished=skatepark_table()
38
39         elif Choice==5:
40             Finished=review_table()
41     else:
42         print()
43     print("Menu Terminated")
44
45 if __name__=="__main__":
46     main()
```

4.11.20 CLI Get Menu Option

```
1 def get_menu_option():
2     Option=int(input("Please select an option: "))
3     return Option
4 if __name__=="main":
5     pass
```

4.11.21 CLI Database Table Menu

```
1 from profile_edit_options import *
2 from trick_edit_options import *
3 from skatepark_edit_options import *
4 from review_edit_options import *
5 from get_menu_option import *
6 from database import *
7 from menu import *
```

```
9  def database_creator_menu():
10     print()
11     print("Database Table Management")
12     print()
13     print("1. (Re)Create All Tables")
14     print("2. (Re)Create User Table")
15     print("3. (Re)Create Difficulty Table")
16     print("4. (Re)Create Trick Table")
17     print("5. (Re)Create Review Table")
18     print("6. (Re)Create Product Brand Table")
19     print("7. (Re)Create Product Type Table")
20     print("8. (Re)Create Product Size Table")
21     print("9. (Re)Create Skatepark Table")
22     print("10. (Re)Create User Trick Table")
23     print("11. (Re)Create User Review Table")
24     print("12. (Re)Create User Skatepark Table")
25     print("13. (Re)Create Review Table")
26     print("0. Exit")
27
28 def database_creator():
29     Finished=False
30     while not Finished:
31         database_creator_menu()
32         Choice=get_menu_option()
33         if Choice==0:
34             return False
35         elif Choice==1:
36             create_user_table()
37             create_difficulty_table()
38             create_trick_table()
39             create_review_table()
40             create_product_brand_table()
41             create_product_type_table()
42             create_product_size_table()
43             create_skatepark_table()
44             create_user_trick_table()
45             create_user_review_table()
46             create_user_skatepark_table()
47             Finished=create_product_table()
48
49         elif Choice == 2:
50             create_user_table()
51         elif Choice == 3:
52             create_difficulty_table()
53         elif Choice == 4:
54             create_trick_table()
```

```
55     elif Choice ==5:
56         create_review_table()
57     elif Choice==6:
58         create_product_brand_table()
59     elif Choice==7:
60         create_product_type_table()
61     elif Choice==8:
62         create_product_size_table()
63     elif Choice==9:
64         create_skatepark_table()
65     elif Choice==10:
66         create_user_trick_table()
67     elif Choice==11:
68         create_user_review_table()
69     elif Choice==12:
70         create_user_skatepark_table()
71     elif Choice==13:
72         create_product_table()
73
74
75
76
77
78
79 def profile_table_menu():
80     print()
81     print("Profile Table Management")
82     print()
83     print("1. Add Profile")
84     print("2. Change Name")
85     print("3. Change Email")
86     print("4. Change Picture")
87     print("5. Delete Profile")
88     print("0. Exit")
89
90 def profile_table():
91     Finished=False
92     while not Finished:
93         profile_table_menu()
94         Choice=get_menu_option()
95         if Choice==0:
96             return False
97         elif Choice==1:
98             Finished=add_profile()
99
100        elif Choice == 2:
```

```
101         Finished=change_name()
102     elif Choice == 3:
103         Finished=change_email()
104     elif Choice == 4:
105         Finished=change_picture()
106     elif Choice ==5:
107         Finished=delete_profile()
108
109
110 def trick_table_menu():
111     print()
112     print("Trick Table Management")
113     print()
114     print("1. Add a New Trick")
115     print("2. Edit an Existing Trick")
116     print("3. Delete an Existing Trick")
117     print("0. Exit")
118
119 def trick_table():
120     Finished=False
121     while not Finished:
122         trick_table_menu()
123         Choice=get_menu_option()
124         if Choice==0:
125             return False
126
127         elif Choice == 1:
128             Finished=add_trick()
129         elif Choice == 2:
130             Finished=edit_trick()
131         elif Choice == 3:
132             Finished=delete_trick()
133
134
135 def skatepark_table_menu():
136     print()
137     print("Skatepark Table Management")
138     print()
139     print("1. Add a New Skatepark")
140     print("2. Edit an Existing Skatepark")
141     print("3. Delete an Existing Skatepark")
142     print("0. Exit")
143
144 def skatepark_table():
145     Finished=False
146     while not Finished:
```

```
147     skatepark_table_menu()
148     Choice=get_menu_option()
149     if Choice==0:
150         return False
151     elif Choice == 1:
152         Finsihed=add_skatepark()
153     elif Choice == 2:
154         Finished=edit_skatepark()
155     elif Choice == 3:
156         Finished=delete_skatepark()
157
158
159 def review_table_menu():
160     print()
161     print("Skatepark Table Management")
162     print()
163     print("1. Add a New Review")
164     print("2. Edit an Existing Review")
165     print("3. Delete an Existing Review")
166     print("4. Filter Brand")
167     print("5. Filter Type")
168     print("6. Filter Size")
169     print("0. Exit")
170
171 def review_table():
172     Finished=False
173     while not Finished:
174         review_table_menu()
175         Choice=get_menu_option()
176         if Choice==0:
177             return False
178         elif Choice == 1:
179             Finished=add_review()
180         elif Choice == 2:
181             Finishededit_review()
182         elif Choice == 3:
183             Finished=delete_review()
184         elif Choice==4:
185             Finished=filter_brand()
186         elif Choice ==5():
187             Finished=filter_type()
188         elif Choice==6():
189             Finished=filter_size()
190
191
192 if __name__=="__main__":
```

193 pass

4.11.22 CLI Create Database

```
1
2 import sqlite3
3
4 def create_table(db_name, table_name, sql):
5     with sqlite3.connect(db_name) as db:
6         cursor = db.cursor()
7         cursor.execute("select name from
8             sqlite_master where name=?", (table_name,))
9         result=cursor.fetchall()
10
11         keep_table=True
12         if len(result)==1:
13             response=input("The table {0} already
14                 exists, do you wish to recreate it?
15                 (y/n) ".format(table_name))
16             if response == "y":
17                 keep_table=False
18                 print("The {0} table will be
19                     recreated - all existing data will
20                     be lost".format(table_name))
21             cursor.execute("drop table if exists
22                 {0}".format(table_name))
23             db.commit()
24             else:
25                 print("The existing table was kept")
26             else:
27                 keep_table=False
28             if not keep_table:
29
30                 cursor.execute(sql)
31                 db.commit()
32
33             db.commit()
34
35         def create_user_table():
36             db_name="skateboard_progress_tracker.db"
37             sql= """create table User (UserID integer,
38                 FirstName text, LastName text, UserPicture
39                 image, UserEmail text, Primary Key(UserID))"""
40             create_table(db_name,"User",sql)
```

```
33     print("Blank User Table Created.")
34
35 def create_user_trick_table():
36     db_name="skateboard_progress_tracker.db"
37     sql= """create table UserTrick (UserTrickID
38         integer, UserID integer, TrickID integer,
39         Primary Key(UserTrickID), Foreign Key(UserID)
40         references User(UserID), Foreign Key(TrickID)
41         references Trick(TrickID))"""
42     create_table(db_name,"UserTrick", sql)
43     print("Blank UserTrick Table Created.")
44
45
46 def create_trick_table():
47     db_name="skateboard_progress_tracker.db"
48     sql= """create table Trick (TrickID integer,
49         DifficultyID integer, TrickCreator text,
50         TrickName text, TrickDescription text,
51         TrickObstacle text, TrickImage image,
52         TrickTutorialLink text, TrickCompleted boolean,
53         TrickCompletedDate text, Primary Key(TrickID),
54         Foreign Key(DifficultyID) references
55         Difficulty(DifficultyID))"""
56     create_table(db_name,"Trick", sql)
57     print("Blank Trick Table Created.")
58
59
60 def create_difficulty_table():
61     db_name="skateboard_progress_tracker.db"
62     sql= """create table Difficulty (DifficultyID
63         integer, TrickDifficulty text,
64         DifficultyDescription text, Primary
65         Key(DifficultyID))"""
66     create_table(db_name,"Difficulty", sql)
67     print("Blank Difficulty Table Created.")
68
69
70 def create_user_review_table():
71     db_name="skateboard_progress_tracker.db"
72     sql= """create table UserReview (UserReviewID
73         integer, UserID integer, ReviewID integer,
74         Primary Key(UserReviewID), Foreign Key(UserID)
75         references User(UserID), Foreign Key(ReviewID)
76         references Review(ReviewID))"""
77     create_table(db_name,"UserReview", sql)
78     print("Blank UserReview Table Created.")
79
80
81
```

```
62 def create_review_table():
63     db_name="skateboard_progress_tracker.db"
64     sql= """create table Review (ReviewID integer,
65         ProductID integer, ReviewCreator text,
66         ReviewDescription text, ReviewRating integer ,
67         Primary Key(ReviewID), Foreign Key(ProductID)
68             references Product(ProductID))"""
69     create_table(db_name,"Review", sql)
70     print("Blank Review Table Created.")
71
72 def create_product_table():
73     db_name="skateboard_progress_tracker.db"
74     sql= """create table Product(ProductID integer,
75         ProductBrandID integer, ProductTypeID integer ,
76         ProductSizeID integer, ProductName text ,
77         Primary Key(ProductID), Foreign
78             Key(ProductBrandID) references
79                 ProductBrand(ProductBrandID),Foreign
80                 Key(ProductSizeID) references
81                 ProductSize(ProductSizeID),Foreign
82                 Key(ProductTypeID) references
83                 ProductType(ProductTypeID))"""
84     create_table(db_name,"Product", sql)
85     print("Blank Product Table Created.")
86
87 def create_product_brand_table():
88     db_name="skateboard_progress_tracker.db"
89     sql= """create table ProductBrand (ProductBrandID
90         integer, ProductBrand text , Primary
91         Key(ProductBrandID))"""
92     create_table(db_name,"ProductBrand", sql)
93     print("Blank ProductBrand Table Created.")
94
95 def create_product_type_table():
96     db_name="skateboard_progress_tracker.db"
97     sql= """create table ProductType (ProductTypeID
98         integer, ProductType text , Primary
99         Key(ProductTypeID))"""
100    create_table(db_name,"ProductType", sql)
101    print("Blank ProductType Table Created.")
102
103 def create_product_size_table():
104     db_name="skateboard_progress_tracker.db"
105     sql= """create table ProductSize (ProductSizeID
106         integer, ProductSize text , Primary
```

```

    Key(ProductSizeID))"""
91     create_table(db_name, "ProductSize", sql)
92     print("Blank ProductSize Table Created.")
93
94 def create_user_skatepark_table():
95     db_name="skateboard_progress_tracker.db"
96     sql= """create table UserSkatepark
97         (UserSkateparkID integer, UserID integer,
98          SkateparkID integer, Primary
99            Key(UserSkateparkID), Foreign Key(UserID)
100           references User(userID), Foreign
101             Key(SkateparkID) references
102               Skatepark(SkateparkID))"""
103     create_table(db_name, "UserSkatepark", sql)
104     print("Blank UserTrick Table Created.")
105
106
107 if __name__=="__main__":
108     create_user_table()
109     create_difficulty_table()
110     create_trick_table()
111     create_review_table()
112     create_product_brand_table()
113     create_product_type_table()
114     create_product_size_table()
115     create_skatepark_table()
116     create_user_trick_table()
117     create_user_review_table()
118     create_user_skatepark_table()
119     create_product_table()

```

4.11.23 CLI Profile Edit Options

```
1 import sqlite3
```

```
2
3 def test_profile():
4     with
5         sqlite3.connect("skateboard_progress_tracker.db")
6             as db:
7                 cursor=db.cursor()
8                 cursor.execute("select * from User")
9                 User=cursor.fetchall()
10                User=len(User)
11                return User
12
13
14 def add_profile():
15     existing_user=test_profile()
16     if existing_user==0:
17         FirstName=get_first_name()
18         LastName=get_last_name()
19         Email=get_email()
20         FilePath=get_file_path()
21         values=(FirstName,LastName,Email,FilePath)
22         with
23             sqlite3.connect("skateboard_progress_tracker.db")
24                 as db:
25                     cursor = db.cursor()
26                     sql="insert into
27                         User(FirstName,LastName,UserEmail,UserPicture)
28                         values (?,?,?,?,?)"
29                     cursor.execute(sql,values)
30                     db.commit()
31                     print()
32                     print("Profile Successfully Created.")
33                     print()
34
35     else:
36         print()
37         print("Profile already exists - To create a
38             new profile, please delete the existing
39             profile.")
40         print()
41
42 def delete_profile():
43     Finished=False
44     while not Finished:
45         Delete=input("Are you sure you wish to delete
46             your profile? (Y/N) ")
47         Delete=Delete.upper()
48
```

```
39         if Delete=="Y":
40
41             data=(1,)
42             with
43                 sqlite3.connect("skateboard_progress_tracker.db")
44                     as db:
45                         cursor=db.cursor()
46                         sql="delete from User where UserID=?"
47                         cursor.execute(sql,data)
48                         db.commit()
49                         print()
50                         print("Profile Successfully Deleted.")
51                         print()
52                         Finished=True
53
54             elif Delete == "N":
55                 print()
56                 print("Delete Aborted")
57                 print()
58                 Finished=True
59
60
61
62
63
64     def get_first_name():
65         print()
66         FirstName=input("Please Enter Your First Name: ")
67         print()
68         return FirstName
69
70
71     def get_last_name():
72         print()
73         LastName=input("Please Enter Your Last Name: ")
74         print()
75         return LastName
76
77     def change_name():
78         FirstName=get_first_name()
79         LastName=get_last_name()
80         values=(FirstName,LastName, 1)
81         with
82             sqlite3.connect("skateboard_progress_tracker.db")
```

```
        as db:
82         cursor = db.cursor()
83         sql="update User set FirstName=?, LastName=?
84             where UserID=?"
85         cursor.execute(sql,values)
86         db.commit()

87
88     def get_email():
89         print()
90         Email=input("Please Enter Your Email Address: ")
91         print()
92         return Email

93
94     def change_email():
95         Email=get_email()
96         values=(Email,1)
97         with
98             sqlite3.connect("skateboard_progress_tracker.db")
99                 as db:
100                     cursor = db.cursor()
101                     sql="update User set UserEmail=? where
102                         UserID=?"
103                     cursor.execute(sql,values)
104                     db.commit()

105
106     def get_file_path():
107         print()
108         FilePath=input("Please Enter The File Path For
109             The JPEG Image: ")
110         print()
111         return FilePath

112
113     def change_picture():
114         FilePath=get_file_path()
115         values=(FilePath,1)
116         with
117             sqlite3.connect("skateboard_progress_tracker.db")
118                 as db:
119                     cursor = db.cursor()
120                     sql="update User set UserPicture=? where
121                         UserID=?"
122                     cursor.execute(sql,values)
123                     db.commit()

124
125 if __name__=="__main__":
126
```

```
119     test_profile()
```

4.11.24 CLI Trick Edit Options

```
1 import sqlite3
2
3 def get_difficulty_id():
4     Finished=False
5     while not Finished:
6         Difficulty=input("Please enter a difficulty:
7             (Easy/Medium/Hard) ")
8         Difficulty=Difficulty.lower()
9         if Difficulty=="easy":
10            Difficulty=1
11            Finished=True
12        elif Difficulty == "medium":
13            Difficulty=2
14            Finished=True
15        elif Difficulty=="hard":
16            Difficulty=3
17            Finished=True
18        else:
19            print()
20            print("Please select a correct
21                 difficulty.")
22            print()
23    return Difficulty
24
25
26
27
28
29 def get_trick_creator():
30     User="Name"
31     return User
32
33
34 def get_trick_name():
35     TrickName=input("Please enter the trick name: ")
36     return TrickName
37
38
39 def get_trick_description():
40     TrickDescription=input("Please enter the trick
41                     description: ")
42     return TrickDescription
43
44
45 def get_trick_obstacle():
```

```
38     TrickObstacle=input("Please enter the trick
39         obstacle: ")
40     return TrickObstacle
41
42 def get_trick_image():
43     TrickImage = input("Please enter a JPEG images
44         file path: ")
45     return TrickImage
46
47 def get_trick_tutorial_link():
48     TrickTutorialLink=input("Please enter a trick
49         tutorial link: ")
50     return TrickTutorialLink
51
52 def get_trick_completed():
53     pass
54
55 def add_trick():
56     DifficultyID=get_difficulty_id()
57     TrickCreator=get_trick_creator()
58     TrickName=get_trick_name()
59     TrickDescription=get_trick_description()
60     TrickObstacle=get_trick_obstacle()
61     TrickImage=get_trick_image()
62     TrickTutorialLink=get_trick_tutorial_link()
63     TrickCompleted=get_trick_completed()
64     TrickCompletedDate=get_trick_completed_date()
65
66     values=(DifficultyID,TrickCreator,TrickName,TrickDescription,TrickObstacle)
67     with
68         sqlite3.connect("skateboard_progress_tracker.db")
69             as db:
70                 cursor = db.cursor()
71                 sql="insert into Trick(DifficultyID,
72                     TrickCreator,TrickName,TrickDescription,TrickObstacle,TrickImage,
73                     values (?,?,?,?,?,?)"
74                 cursor.execute(sql,values)
75                 db.commit()
76                 print()
77                 print("Trick Successfully Created.")
78                 print()
```

```
77     TrickID=int(input("Please enter the TrickID of
78         the trick you wish to edit: "))
79     DifficultyID=get_difficulty_id()
80     TrickCreator=get_trick_creator()
81     TrickName=get_trick_name()
82     TrickDescription=get_trick_description()
83     TrickObstacle=get_trick_obstacle()
84     TrickImage=get_trick_image()
85     TrickTutorialLink=get_trick_tutorial_link()
86     TrickCompleted=get_trick_completed()
87     TrickCompletedDate=get_trick_completed_date()
88
89     values=(DifficultyID,TrickCreator,TrickName,TrickDescription,TrickObstic
90             TrickID)
91     with
92         sqlite3.connect("skateboard_progress_tracker.db")
93             as db:
94                 cursor = db.cursor()
95                 sql="update Trick set DifficultyID=?,
96                     TrickCreator=? , TrickName=? ,
97                     TrickDescription=? ,
98                     TrickObstacle=? , TrickImage=? ,
99                     TrickTutorialLink=? , TrickCompleted=? ,
100                     TrickCompletedDate=? where TrickID=?"
101                 cursor.execute(sql,values)
102                 db.commit()
103
104     def delete_trick():
105         data=int(input("Please enter the TrickID of the
106             trick you wish to delete: "))
107         data=(data,)
108         with
109             sqlite3.connect("skateboard_progress_tracker.db")
110                 as db:
111                     cursor=db.cursor()
112                     sql="delete from Trick where TrickID=?"
113                     cursor.execute(sql,data)
114                     db.commit()
115                     print()
116                     print("Trick Successfully Deleted.")
117                     print()
118
119     if __name__=="__main__":
120         get_trick_creator()
```

4.11.25 CLI Skatepark Edit Options

```
1 import sqlite3
2
3 def get_skatepark_name():
4     Name=input("Please enter the skateparks name: ")
5     return Name
6
7 def get_skatepark_description():
8     Description=input("Please enter the description
9         for the skatepark: ")
10    return Description
11
12 def get_skatepark_longitude():
13     Longitude = input("Please enter the longitude for
14         the skatepark: ")
15     return Longitude
16
17 def get_skatepark_latitude():
18     Latitude=input("Please enter the latitude for the
19         skatepark: ")
20     return Latitude
21
22 def add_skatepark():
23     SkateparkName=get_skatepark_name()
24     SkateparkDescription=get_skatepark_description()
25     SkateparkLongitude=get_skatepark_longitude()
26     SkateparkLatitude=get_skatepark_latitude()
27     values=(SkateparkName,SkateparkLongitude,SkateparkLatitude,SkateparkDescription)
28     with
29         sqlite3.connect("skateboard_progress_tracker.db")
30             as db:
31                 cursor = db.cursor()
32                 sql="insert into
33                     Skatepark(SkateparkName,SkateparkLongitude,SkateparkLatitude,SkateparkDescription)
34                     values(?, ?, ?, ?)"
35                 cursor.execute(sql,values)
36                 db.commit()
37                 print()
38                 print("Skatepark Successfully Created.")
39                 print()
```

```

38     SkateparkID=int(input("Please enter the
39         SkateparkID of the skatepark you wish to edit:
40             "))
41     SkateparkName=get_skatepark_name()
42     SkateparkDescription=get_skatepark_description()
43     SkateparkLongitude=get_skatepark_longitude()
44     SkateparkLatitude=get_skatepark_latitude()
45     values=(
46         SkateparkName,SkateparkLongitude,SkateparkLatitude,SkateparkDescripti
47     with
48         sqlite3.connect("skateboard_progress_tracker.db")
49             as db:
50                 cursor = db.cursor()
51                 sql="update Skatepark set SkateparkName=?,
52                     SkateparkLongitude=? , SkateparkLatitude=? ,
53                     SkateparkDescription=? where SkateparkID=?"
54                 cursor.execute(sql,values)
55                 db.commit()
56
57     def delete_skatepark():
58         data=int(input("Please enter the SkateparkID of
59             the skatepark you wish to delete: "))
60         data=(data,)
61         with
62             sqlite3.connect("skateboard_progress_tracker.db")
63                 as db:
64                     cursor=db.cursor()
65                     sql="delete from Skatepark where
66                         SkateparkID=?"
67                     cursor.execute(sql,data)
68                     db.commit()
69                     print()
70                     print("Skatepark Successfully Deleted.")
71                     print()

```

4.11.26 CLI Review Edit Options

```

1 import sqlite3
2
3 def get_product_id():
4     value=int(input("Please enter the product id of
5         the product you wish to review: "))
6     with
7         sqlite3.connect("skateboard_progress_tracker.db")

```

```
    as db:
6        cursor=db.cursor()
7        cursor.execute("select ProductID from Product
8            where ProductID=?",(value,))
9        ProductID=cursor.fetchone()
10       return productID
11
12   def get_review_creator():
13       with
14           sqlite3.connect("skateboard_progress_tracker.db")
15           as db:
16               cursor=db.cursor()
17               cursor.execute("select FirstName,LastName
18                   from User where UserID=?",(1,))
19               User=cursor.fetchone()
20               User= "{0} {1}".format(User[0], User[1])
21               return User
22
23   def get_review_description():
24       ReviewDescription=input("Please enter a
25           description for your review: ")
26       return ReviewDescription
27
28   def get_review_rating():
29       Finished=False
30       while not Finished:
31           ReviewRating=int(input("Please rate the
32               product: (1-5) "))
33           if (ReviewRating>0) and (ReviewRating<=5):
34               Finished=True
35               print()
36           else:
37               print("Invalid entry")
38               print()
39
40   def add_review():
41       ProductID= get_product_id
42       ReviewCreator=get_review_creator()
43       ReviewDescription=get_review_description()
44       ReviewRating=get_review_rating()
45
46       values=(ProductID,ReviewCreator,ReviewDescription,ReviewRating)
```

```
44     with
45         sqlite3.connect("skateboard_progress_tracker.db")
46             as db:
47                 cursor = db.cursor()
48                 sql="insert into Review(ProductID,
49                     ReviewCreator,ReviewDescription,ReviewRating)
50                         values (?,?,?,?,?)"
51                         cursor.execute(sql,values)
52                         db.commit()
53                         print()
54                         print("Review Successfully Created.")
55                         print()
56
57
58
59
60
61
62
63
64
65
66
67
68     def edit_review():
69         TrickID=int(input("Please enter the TrickID of
70             the trick you wish to edit: "))
71         ProductID=get_product_id()
72         ReviewCreator=get_review_creator()
73         ReviewDescription=get_review_description()
74         ReviewRating=get_review_rating()
75
76         values=(ProductID,ReviewCreator,ReviewDescription,ReviewRating,TrickID)
77         with
78             sqlite3.connect("skateboard_progress_tracker.db")
79                 as db:
80                     cursor = db.cursor()
81                     sql="update Review set ProductID=?,
82                         ReviewCreator=?, ReviewDescription=?,
83                             ReviewRating=? where TrickID=?"
84                     cursor.execute(sql,values)
85                     db.commit()
86
87
88     def delete_review():
89         data=int(input("Please enter the ReviewID of the
90             review you wish to delete: "))
91         data=(data,)
92         with
93             sqlite3.connect("skateboard_progress_tracker.db")
94                 as db:
95                     cursor=db.cursor()
96                     sql="delete from Review where ReviewID=?"
97                     cursor.execute(sql,data)
98                     db.commit()
99                     print()
100                     print("Review Successfully Deleted.")
```

```
78         print()
79
80     def filter_size():
81         pass
82
83     def filter_brand():
84         pass
85
86     def filter_size():
87         pass
```

4.11.27 CLI Make New Difficulty

```
1 import sqlite3
2
3 def get_difficulty():
4     difficulty=input("Please enter a difficulty: ")
5     return difficulty
6
7 def get_description():
8     description=input("Please enter a description: ")
9     return description
10
11
12 def create_difficulties():
13     TrickDifficulty=get_difficulty()
14     DifficultyDescription=get_description()
15     values=(TrickDifficulty,DifficultyDescription)
16     with
17         sqlite3.connect("skateboard_progress_tracker.db")
18         as db:
19             cursor = db.cursor()
20             sql="insert into
21                 Difficulty(TrickDifficulty,DifficultyDescription)
22                 values (?,?)"
23             cursor.execute(sql,values)
24             db.commit()
25             print()
26             print("Difficulty Successfully Created.")
27             print()
```

4.11.28 CLI Make New Product

```
1 import sqlite3
2
3 def get_product_brand():
4     Brand=input("Please enter a product brand: ")
5     return Brand
6
7 def get_product_size():
8     Size=input("Please enter a product size: ")
9     return Size
10
11 def get_product_type():
12     Type=input("Please enter a product type: ")
13     return Type
14
15
16 def create_product_brand():
17     ProductBrand=get_product_brand()
18
19     values=(ProductBrand,)
20     with
21         sqlite3.connect("skateboard_progress_tracker.db")
22             as db:
23                 cursor = db.cursor()
24                 sql="insert into ProductBrand(ProductBrand)
25                     values (?)"
26                 cursor.execute(sql,values)
27                 db.commit()
28                 print()
29                 print("Product Brand Successfully Created.")
30                 print()
31
32     def create_product_size():
33         ProductSize=get_product_size()
34         values=(ProductSize,)
35         with
36             sqlite3.connect("skateboard_progress_tracker.db")
37                 as db:
38                     cursor = db.cursor()
39                     sql="insert into ProductSize(ProductSize)
40                         values (?)"
41                     cursor.execute(sql,values)
42                     db.commit()
43                     print()
44                     print("Product Size Successfully Created.")
```

```
39         print()
40
41     def create_product_type():
42         ProductType=get_product_type()
43         values=(ProductType,)
44         with
45             sqlite3.connect("skateboard_progress_tracker.db")
46             as db:
47                 cursor = db.cursor()
48                 sql="insert into ProductType(ProductType)
49                     values (?)"
50                 cursor.execute(sql,values)
51                 db.commit()
52                 print()
53                 print("Product Type Successfully Created.")
54                 print()
55
56     if __name__ == "__main__":
57         while 1==1:
58             #create_product_brand()
59             #create_product_size()
60             create_product_type()
```

Chapter 5

User Manual

5.1 Introduction

The purpose of my program is to act as a skateboarding progress tracker and personal assistant. The program was built to aid skateboarding progress, maximise physical performance and enhance muscle memory by reminding the user of newly learnt tricks. The skateboard progress tracker also caters for the users skateboarding buying needs as it includes its very own skateboard part review hub. The integrated Google maps keeps a store of all of the skateparks and skate spots in the world, which skaters using the program can add to.

The intended audience of my program is currently purely for my client; however in the future, once a multi-user platform has been integrated the intended audience will be for the whole skating community.

As the skateboard progress tracker is still an uncompleted program, some of the functionality is not currently available in a graphical user interface format and not all of the validation is fully functional.

Currently, the version of the skateboard progress tracker includes a personalised profile tab where you can change the profile picture, email address and name to your own personal information. The tricks tab contains a table with a list of all the tricks in the database. You may delete tricks from your database by selecting the desired row and pressing delete. Another function on the tricks tab is the ability to add a trick to the database. With an easy to fill, side form that appears when 'add trick' is selected. This tab is used to keep a record of all the tricks that the user can do. The skatepark tab contains a customisable Google maps object which allows you to add and remove skateparks. This tab is used so that the user can plan on visiting new skateparks that they have never been to before. The review tab contains a table which displays all the reviews in the database. Functionality with the graphical user interface will be integrated in

the next version of the program. The support tab allows the user to report any problems with the program, and request any additional functionality that they wish me to implement.

5.2 Installation

System Requirements

Before installing, your computer must meet the system requirements. As discussed in the analysis and design section (Subsections 1.7.1 and 2.3 respectively) I am developing this program for my client, therefore the system is built to run on:

- 15.6" HD 1366x768 Screen
- i5-2450M Dual Core Processor (Sandy Bridge) 2.5GHz (overclocked to 3.1GHz) 3MB Cache
- 8GB DDR3 RAM
- 500GB HDD Memory
- Intel HD3000 Graphics Card
- Windows 7 Operating System

My program has been tested, and works on Windows 7 and Vista. Although the program hasn't been tested on Mac computers or windows 8, but the program should still work fine as long as the required programs are installed correctly as discussed below.

5.2.1 Prerequisite Installation

Installing Python 3.4

Installing PyQt

5.2.2 System Installation

To install the system go to my personal github page and find the Comp4 repository (<http://www.github.com/BenKeppie>). Click on the download zip button and then save the file in an appropriate place on your hard drive.

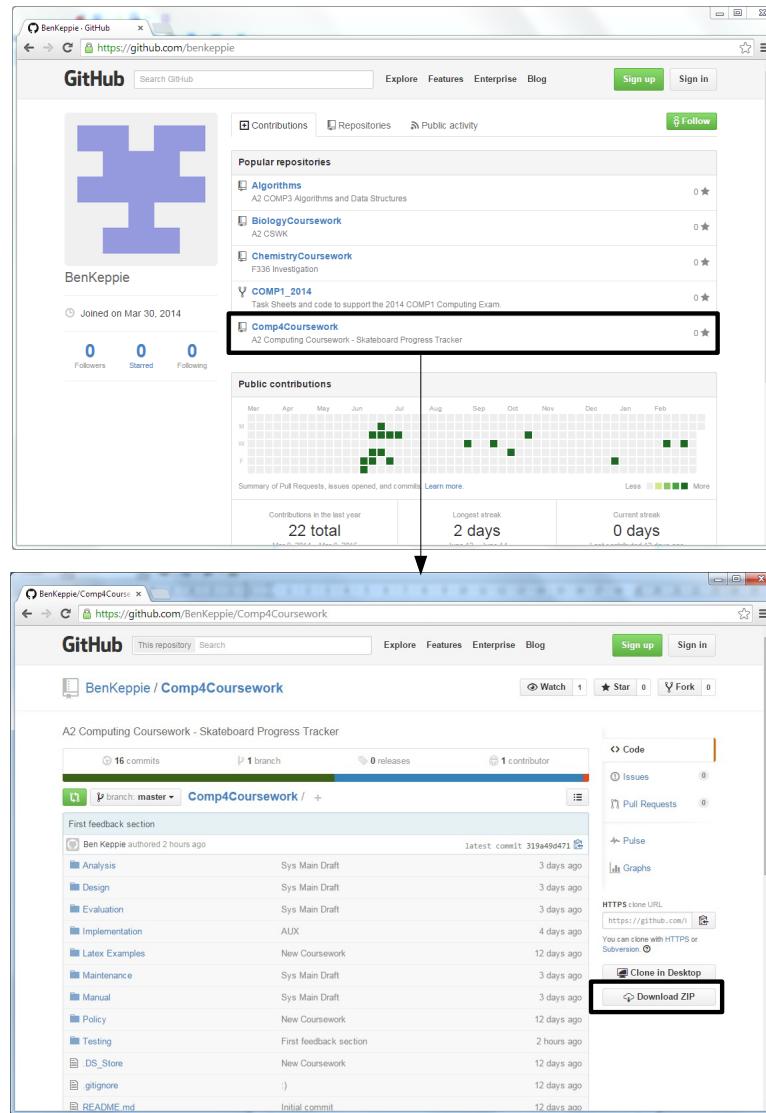


Figure 5.1: Downloading Skateboarding Progress Tracker Application

To extract the files to a usable form you will need to extract the zip file. To do this you need to find the downloaded zip file, right click on it and save the extracted folder in an appropriate destination.

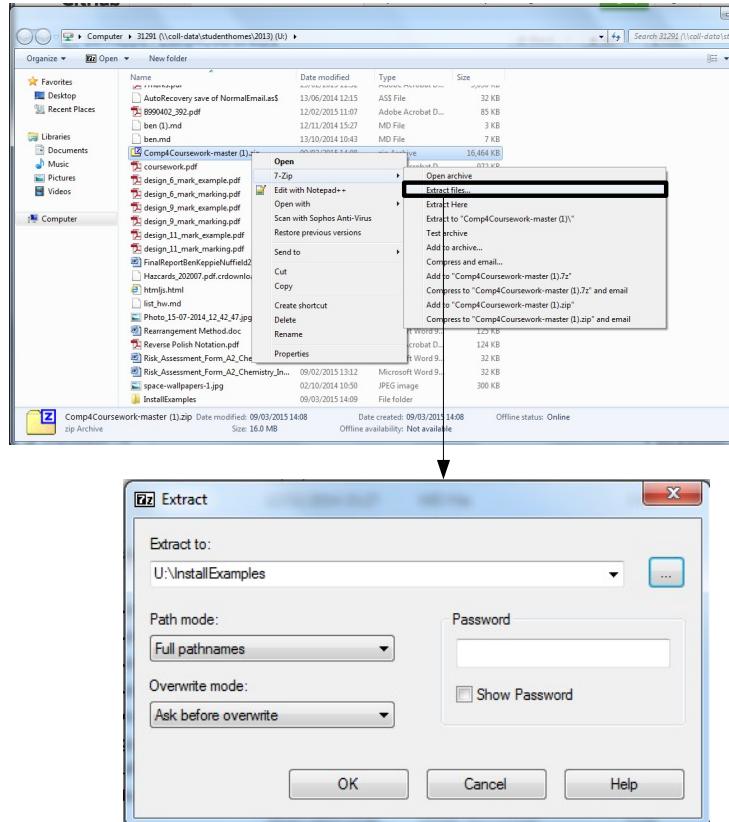


Figure 5.2: Extracting the zip File

The system is now available to run as long as the appropriate programs discussed above are installed.

5.2.3 Running the System

To run the system, open the file location that you extracted the zip file to previously and navigate into the folder 'Implementation' and then click on the file 'main_window.pyw', the program will now be loading, as shown by the start up of the splash screen.

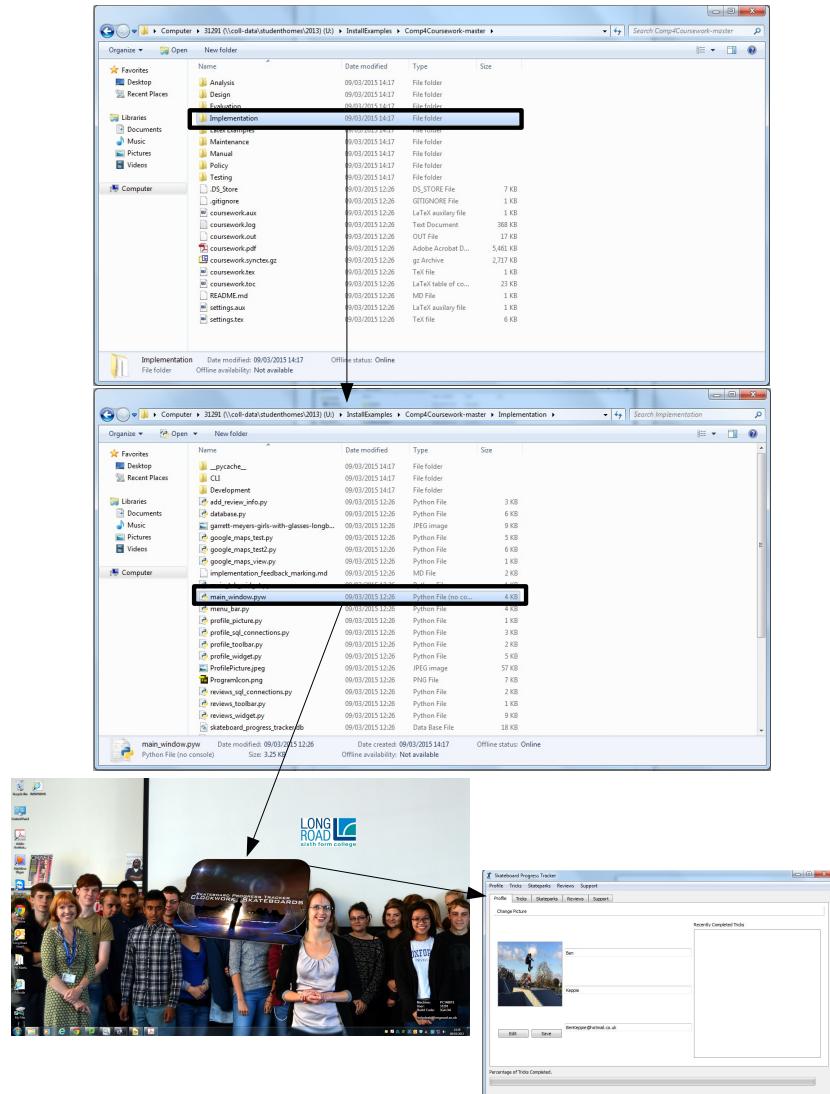


Figure 5.3: Starting up the program

5.3 Tutorial

5.3.1 Introduction

5.3.2 Assumptions

5.3.3 Tutorial Questions

Question 1

Question 2

5.3.4 Saving

My program saves information automatically due to the underlying SQL framework.

5.3.5 Limitations

As discussed in the introduction some sections are not fully implemented into a graphical user interface.

5.4 Error Recovery

5.4.1 Error 1

5.4.2 Error 2

5.5 System Recovery

5.5.1 Backing-up Data

Save to an external hard drive/USB stick

5.5.2 Restoring Data

Copy the database file back.

Chapter 6

Evaluation

6.1 Customer Requirements

Below is a list of all my general and specific objectives that I set myself in the analysis section. In this section I will determine whether I have met all of these objectives and the reasoning behind it. The subsections with *NEW* in the title are objectives that I did not identify in my analysis section; however during the course of my implementation, I attempted to meet the objectives.

6.1.1 Aesthetically pleasing, easy to navigate GUI.

6.1.2 Videos organised and filtering capabilities.

This objective has not been met.

6.1.3 Correct and accurate mapping to the skate parks/spots.

This objective has not been met.

6.1.4 Correct directions from current location to skate park/ spot on the map.

This objective has not been met.

6.1.5 Non-biased reviews.

This objective has not been met.

6.1.6 Clear database with a list of tricks in.**6.1.7 Easy to filter through tricks known.**

This objective has not been met.

6.1.8 Display status bar messages at appropriate times to inform the user of changes *NEW*

This objective has been met.

6.1.9 Allow for the user to contact the developer *NEW*

This objective has been met.

6.1.10 Ensure that the profile picture can be changed easily *NEW*

This objective has been met.

6.1.11 Ensure that the profile name can be edited easily *NEW*

This objective has been met.

6.1.12 Ensure that the profile email can be edited easily *NEW*

This objective has been met.

**6.1.13 Ensure that videos can be filtered by categories.
e.g easy, medium, hard tricks.**

This objective hasn't been met.

6.1.14 Ensure that videos load correctly and are linked to the right video.

This objective hasn't been met.

6.1.15 Ensure that videos are displayed at the correct size/resolution that the monitor of the computer is.

This objective hasn't been met.

6.1.16 Ensure the database can add, edit and remove trick data (Name, description, image, completed status and tutorial link).

This objective hasn't been met. However part of the objective has.

6.1.17 Ensure that the database is displayed correctly inside the application at all resolutions.

This objective has been met.

6.1.18 Ensure that the tricks are marked by how hard they are by a three way scale of: Easy, Medium or Hard.

This objective has been met.

6.1.19 Ensure a checkbox is by the side of a trick to represent whether the user has completed that trick or not.

This objective hasn't been met.

6.1.20 Ensure there is a search bar for a specific trick name.

This objective hasn't been met.

6.1.21 Ensure there are filters for tricks e.g Switch trick filters.

This objective hasn't been met.

Ben Keppie

Candidate No. 4609

Centre No. 22151

- 6.1.22 Ensure that the map is accurate to current roads.
- 6.1.23 Ensure location of the user is not revealed to anyone else.
- 6.1.24 Ensure that the current location marker is accurate.
- 6.1.25 Ensure that when giving directions to skate parks from your current location that the mapping route is correct and on viable roads.
- 6.1.26 Ensure that the program can mark skate park locations.
- 6.1.27 Ensure no biased reviews are posted to the app and that they're moderated before they are universally posted.
- 6.1.28 Ensure the program runs fast without lag when navigating between areas of the application.

6.2 Effectiveness

6.2.1 Objective Evaluation

6.3 Learnability

6.4 Usability

6.5 Maintainability

6.6 Suggestions for Improvement

6.7 End User Evidence

6.7.1 Questionnaires

6.7.2 Graphs

6.7.3 Written Statements 293