Skateboarding Progress Tracker

Ben Keppie

March 11, 2015

Contents

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.`

`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 taskscauses 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 advertisments. 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 compatability 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 ← 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" ← 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 ← 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 ← true
10:            ENDIF
11:        ELSE
12:            OUTPUT "Keep searching for a replacement part"
13:        ENDIF
14:    ENDIF
15: ENDWHILE
```

Data flow diagram

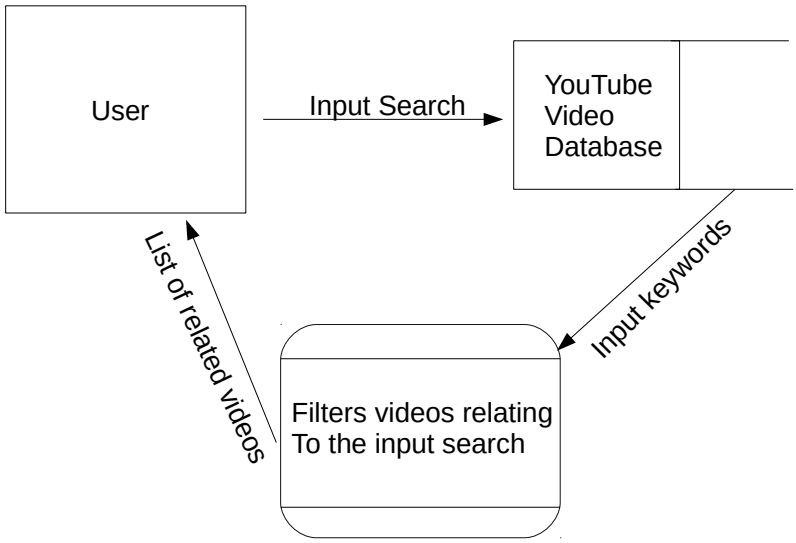


Figure 1.1: Data Flow Diagram of Searching for a YouTube Tutorial

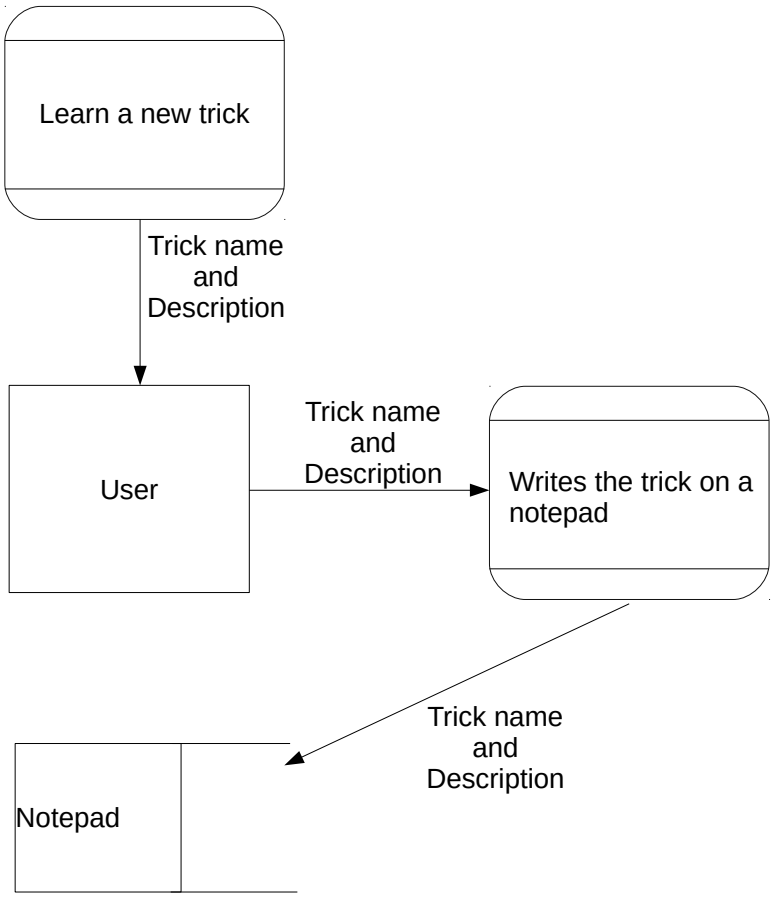


Figure 1.2: Data Flow Diagram of writing recently learnt tricks on a note pad

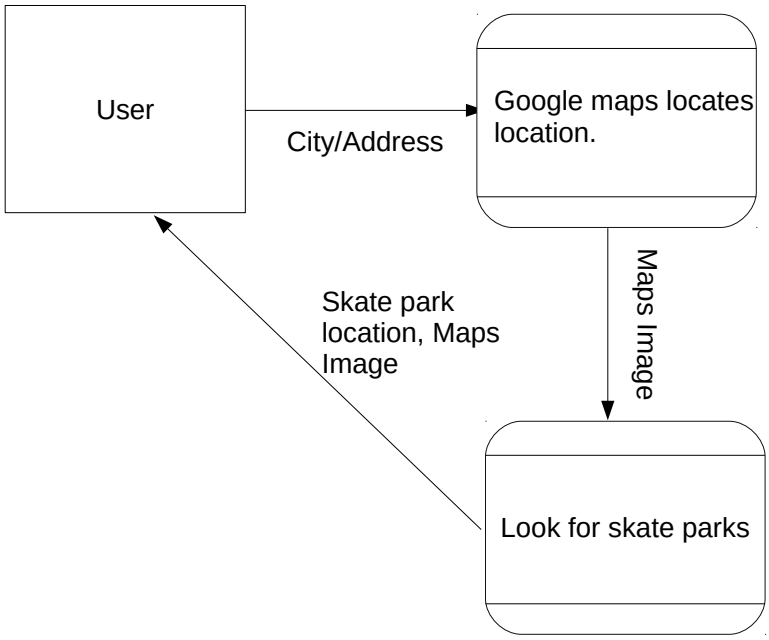


Figure 1.3: Data Flow Diagram of Searching for a skate park

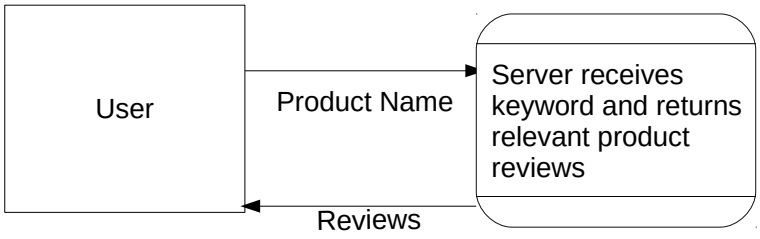


Figure 1.4: Data Flow Diagram of Searching for reviews of a product

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 obsitcles at the skate park, the tricks he learnt on that day and tricks that he saw and possibly wants to try and learn.

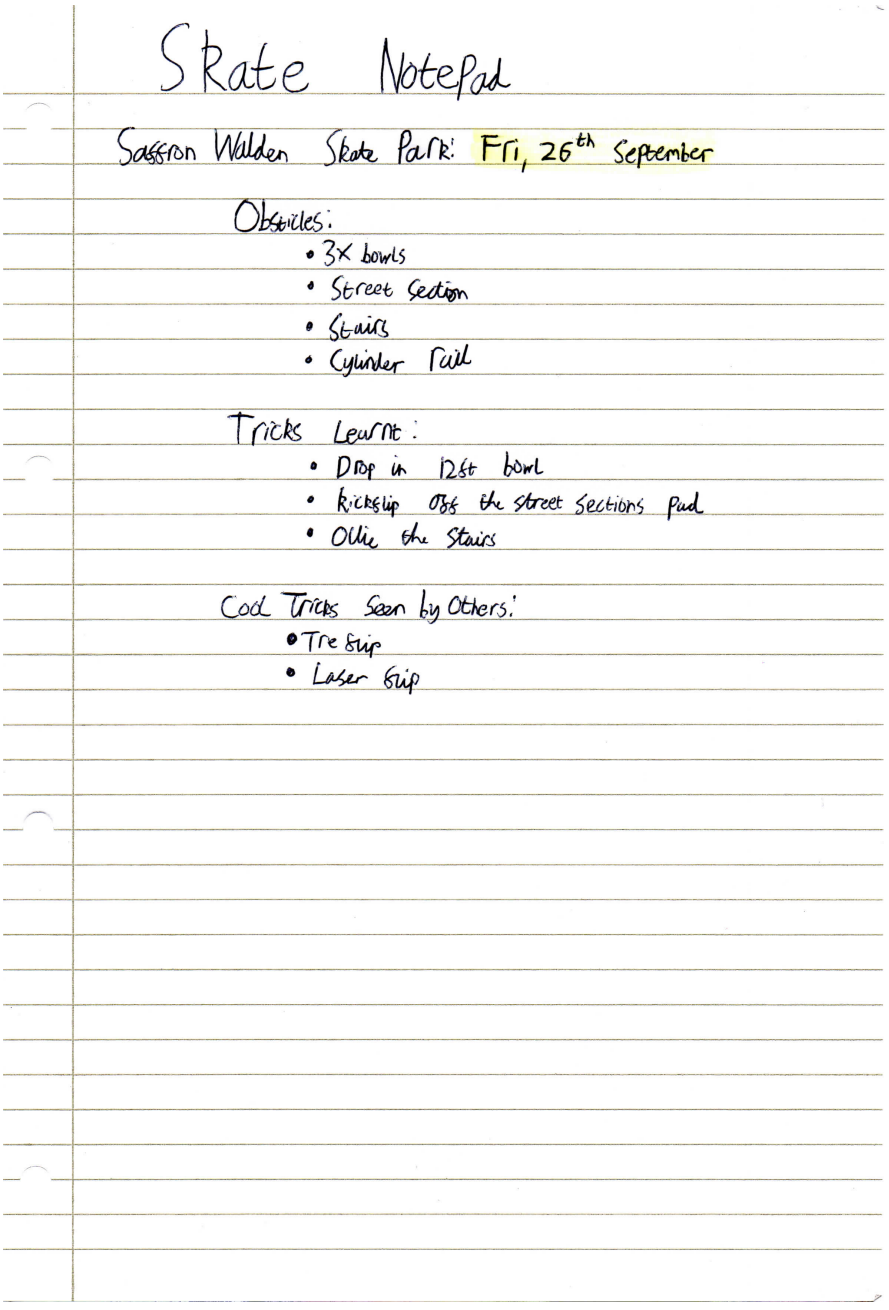


Figure 1.5: A page from Stuarts notepad

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

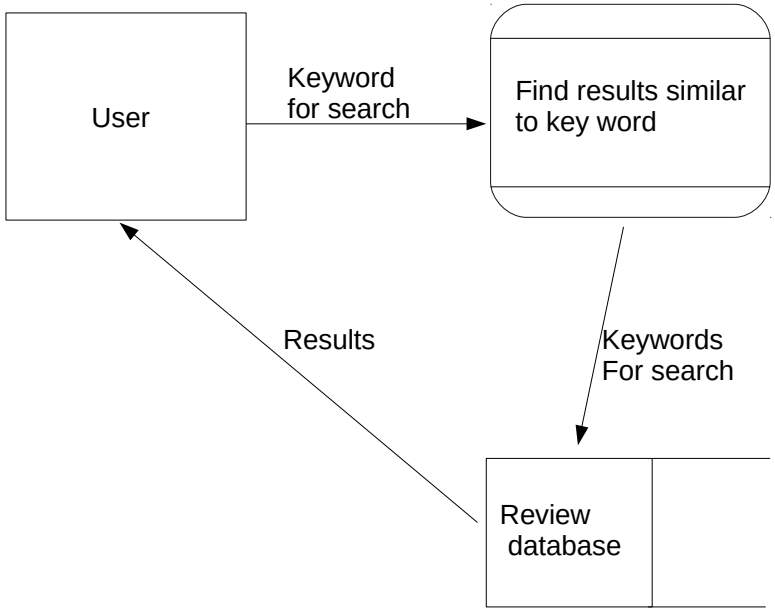


Figure 1.6: Data flow diagram for the new systems review search

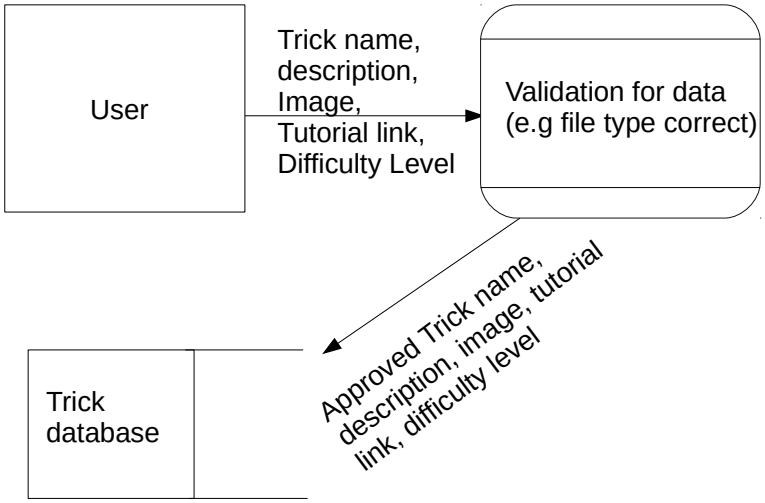


Figure 1.7: Data flow diagram for adding new tricks to the database

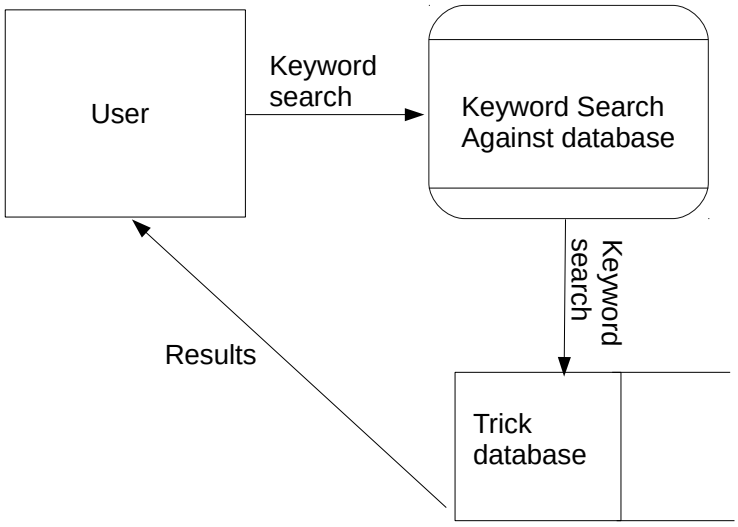


Figure 1.8: Data flow diagram for reading tricks from the database

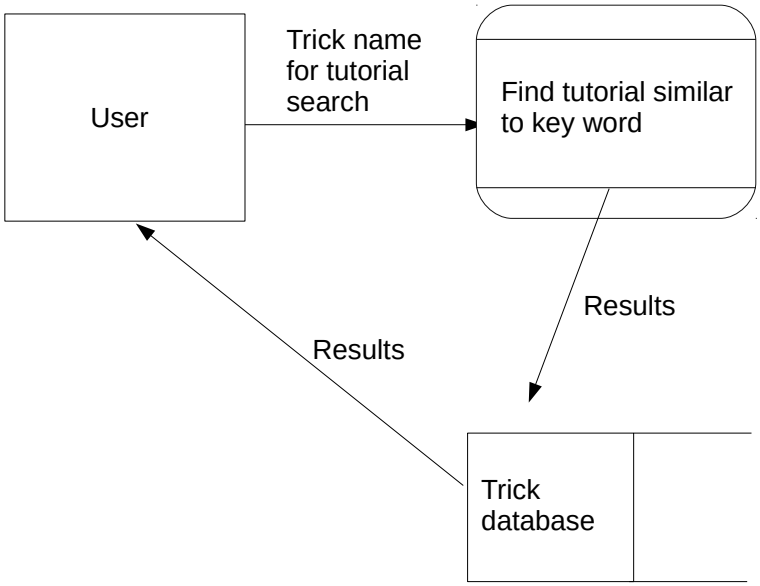


Figure 1.9: Data flow diagram for the new systems tutorial search

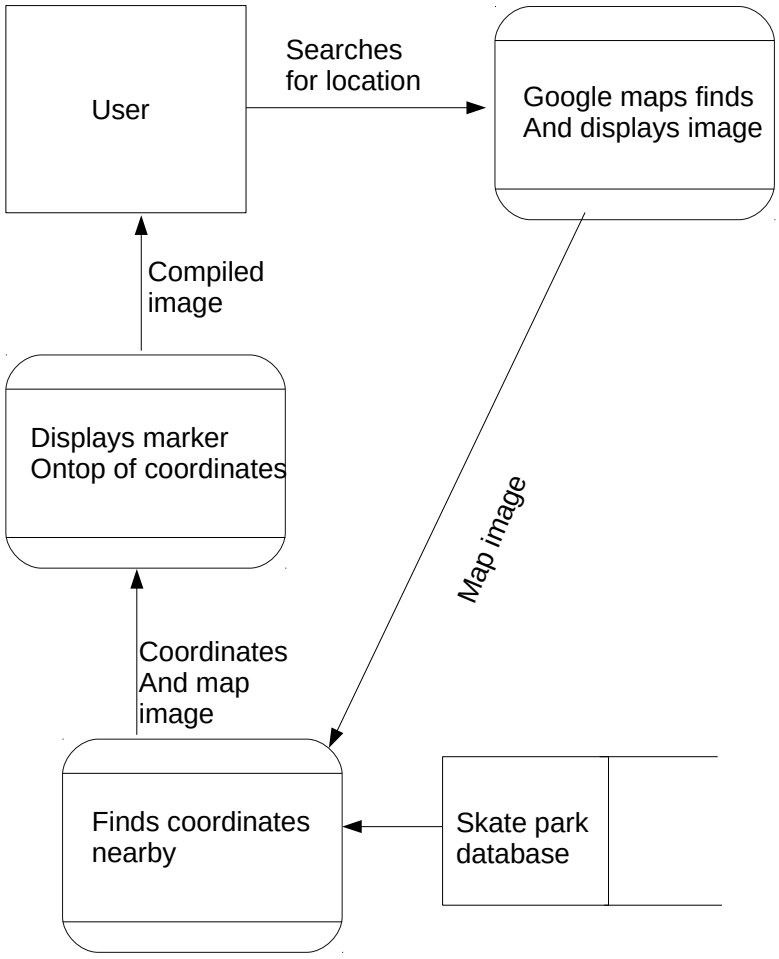


Figure 1.10: Data flow diagram for the new systems skate park search

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	interger	range 1-5	Non-biased	1	Moderated

Volumetrics

For the initial size of the propsed 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 doliphin 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 = 16859470Bytes

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 yout 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

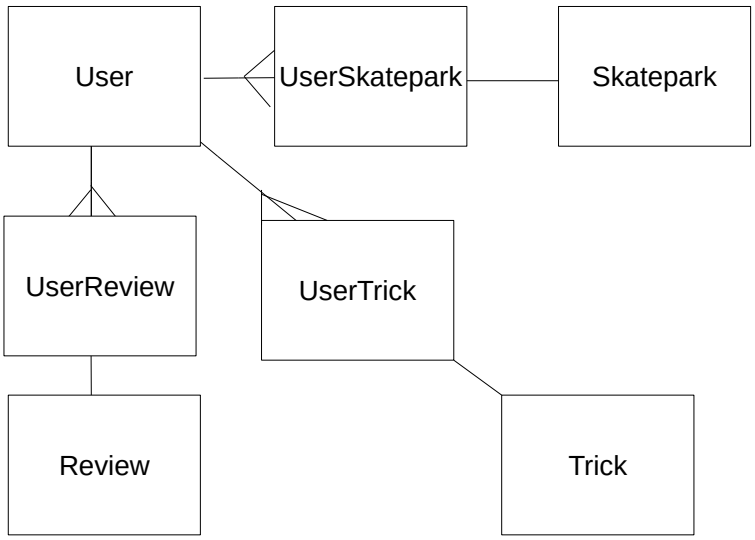


Figure 1.11: Entity-Relationship Diagram

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

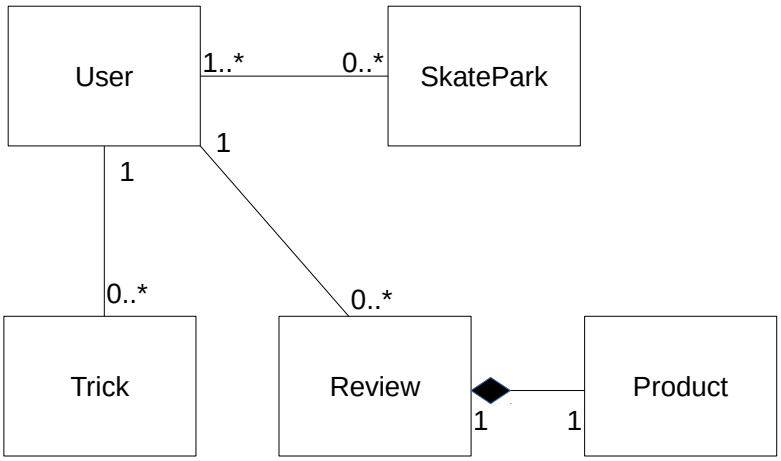


Figure 1.12: Relationship Diagram

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

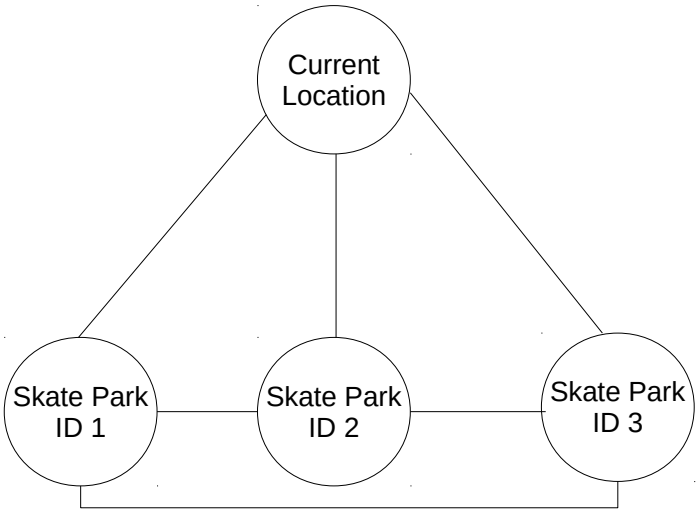


Figure 1.13: Graph to show how the user can map their current location to skateparks.

1.7 Constraints

1.7.1 Hardware

The new system will need to be able to run on Stuart's computer, the current specifcation 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 13h 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 recomend products.
- Phone app.

1.9 Solutions

1.9.1 Alternative solutions

Solution	**Advantages**	**Disadvantages**
Web-Based Application	• 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.)	• Lack of experience in web based programming. • More complex security for hiding locations. • Web Hosting costs money. • More extensive knowledge to fix problems.

• Making the current system more efficient	• No need for the client to learn anything new. • No expense.	• Problems with the current system will still exist. • The system won't be as efficient.
Command-Line Application	• Runs extremely fast. • Uses minimal system resources.	• 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	• Runs complex queries fast. • Could store all the information in a compressed form. • Information easy to access.	• 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)	• 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.	• 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 picutre, 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

One 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 functinality 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 permenantly delete that skatepark. Once the skatepark is deleted you will no longer be able to view the marker of 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

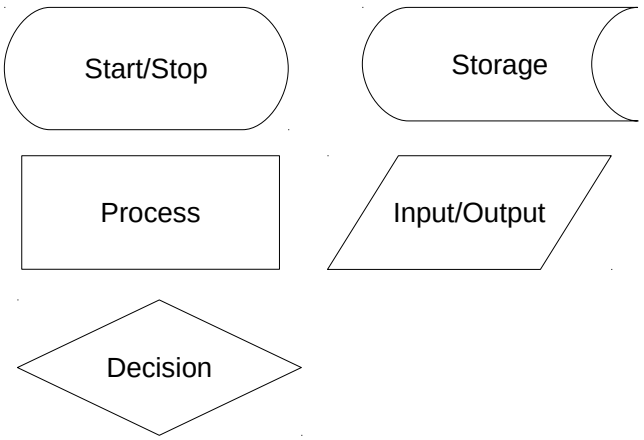


Figure 2.1: System Flowchart Key

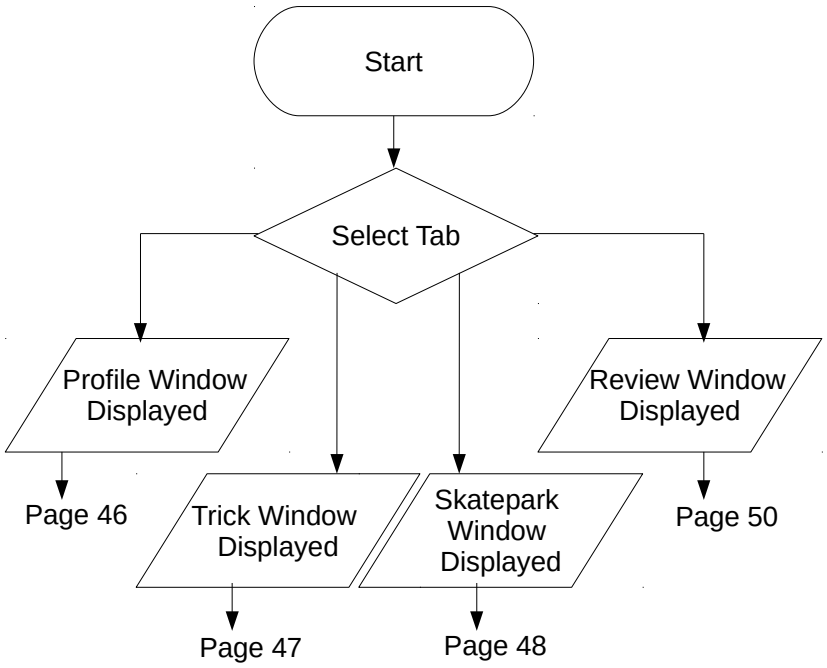


Figure 2.2: Profile Window Flowchart

The flowchart above shows the flow of operations between tabs.

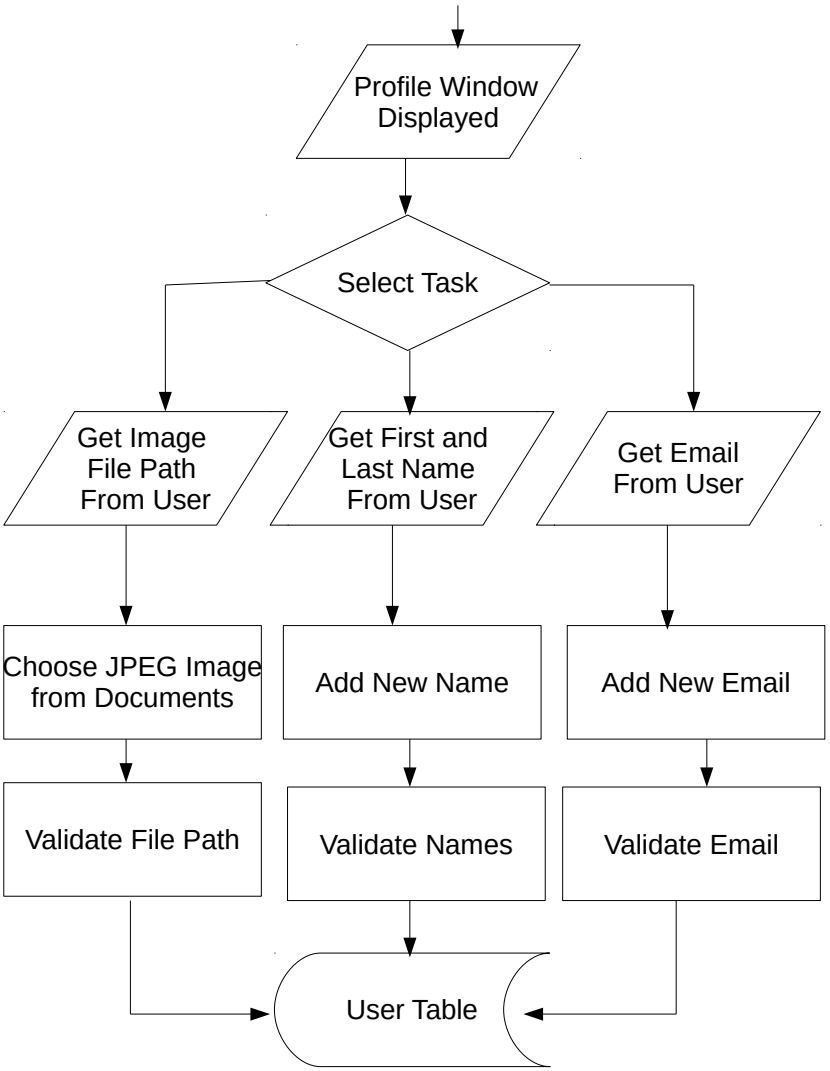


Figure 2.3: Profile Window Flowchart

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.

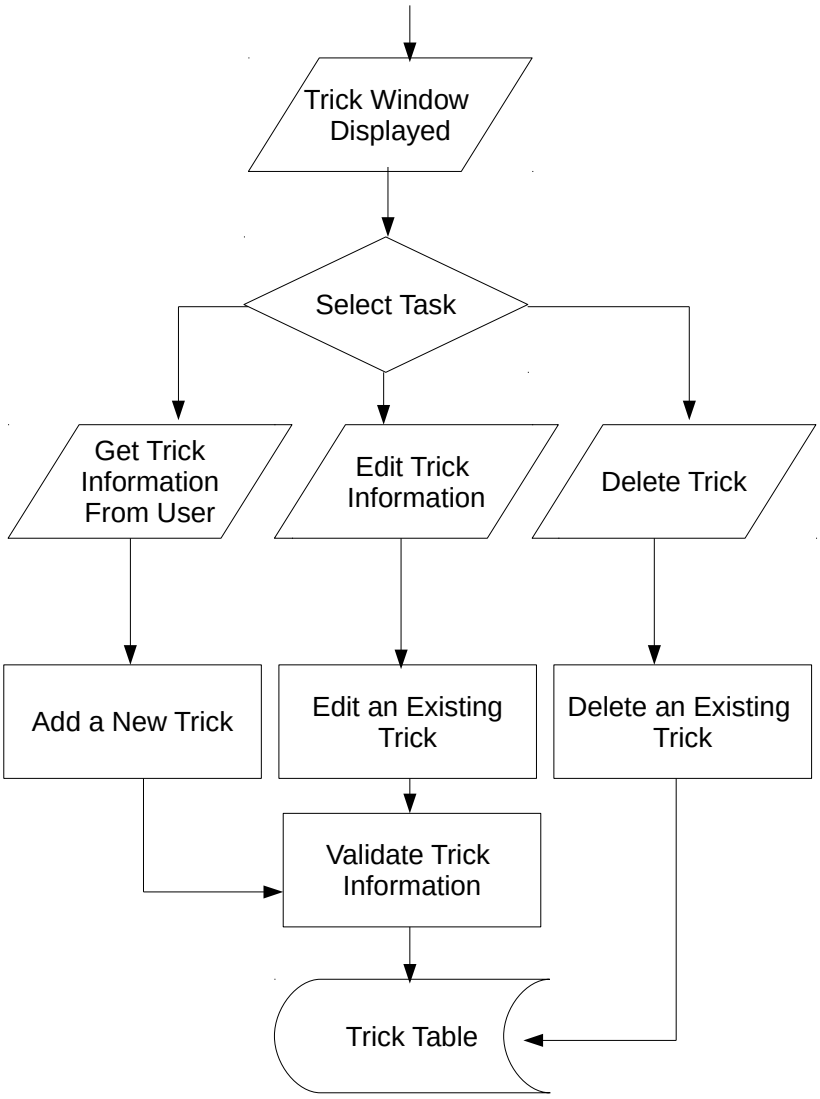


Figure 2.4: Trick Window Flowchart

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

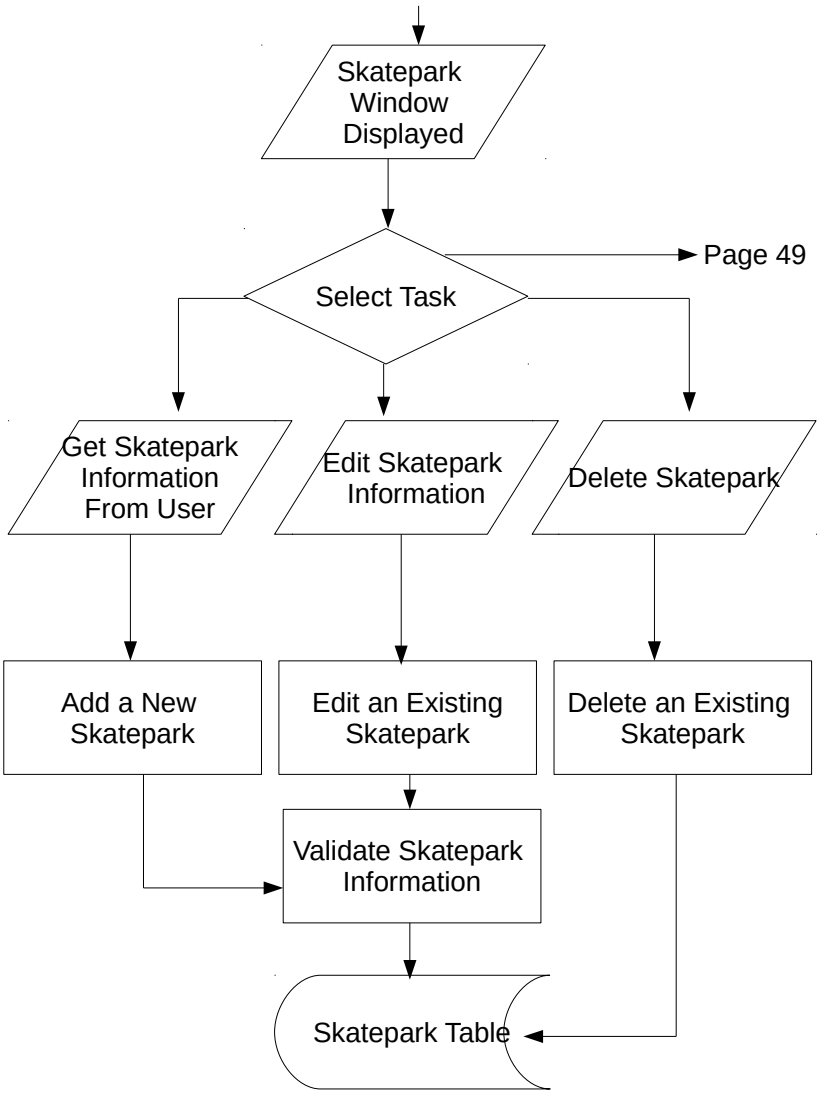


Figure 2.5: Skatepark Window Flowchart

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.

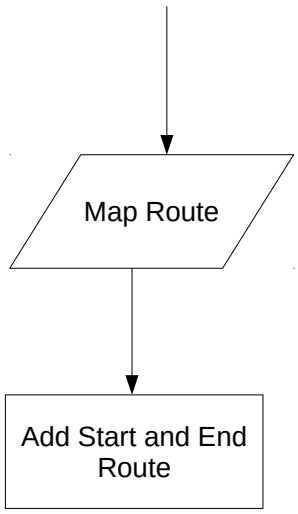


Figure 2.6: Skatepark Window Flowchart

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

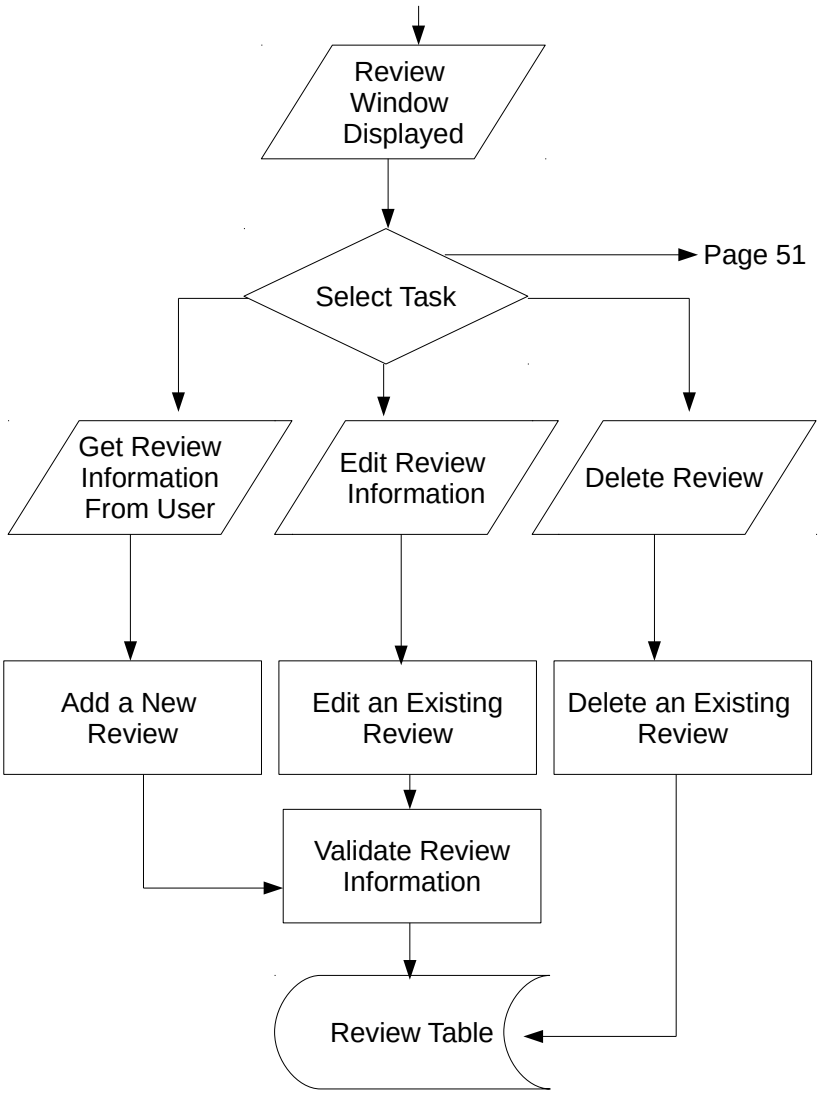


Figure 2.7: Review Window Flowchart

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

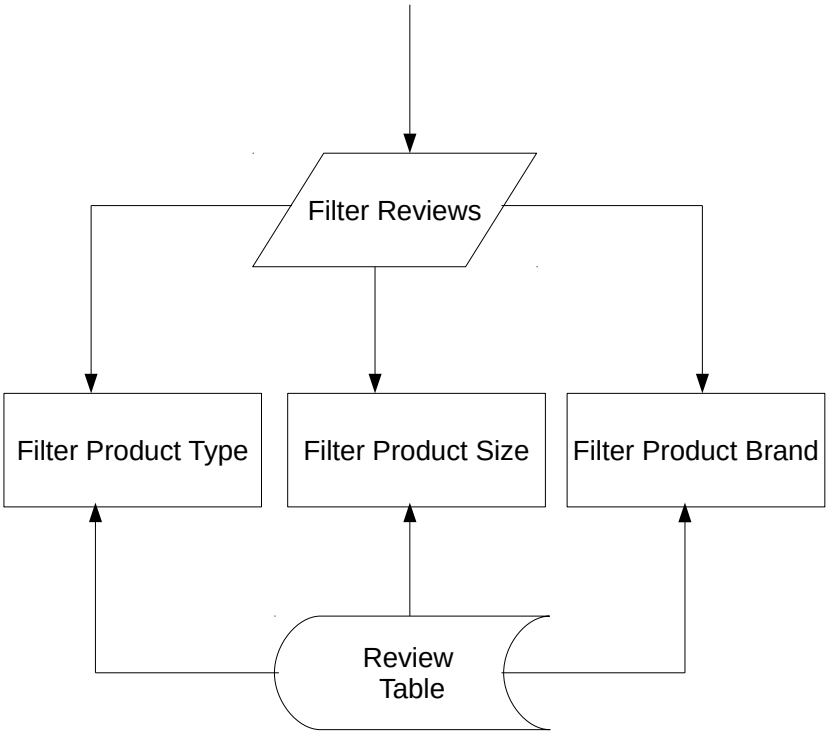


Figure 2.8: Review Window Flowchart

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.

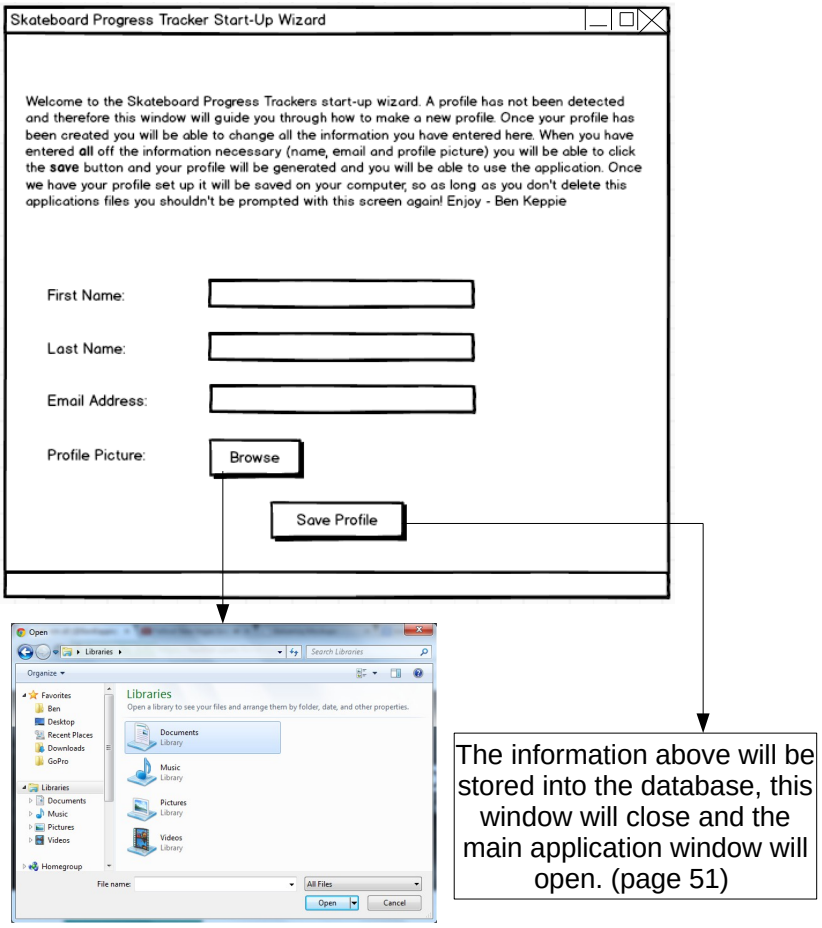


Figure 2.9: The User Interface for the Start-Up Wizard

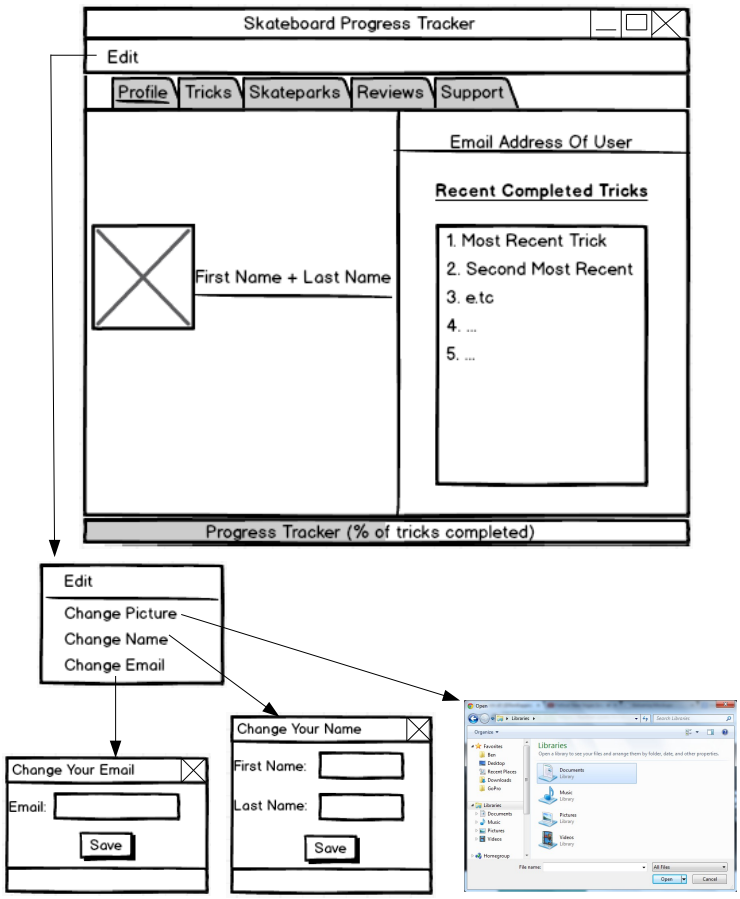


Figure 2.10: The User Interface for the profile section

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.

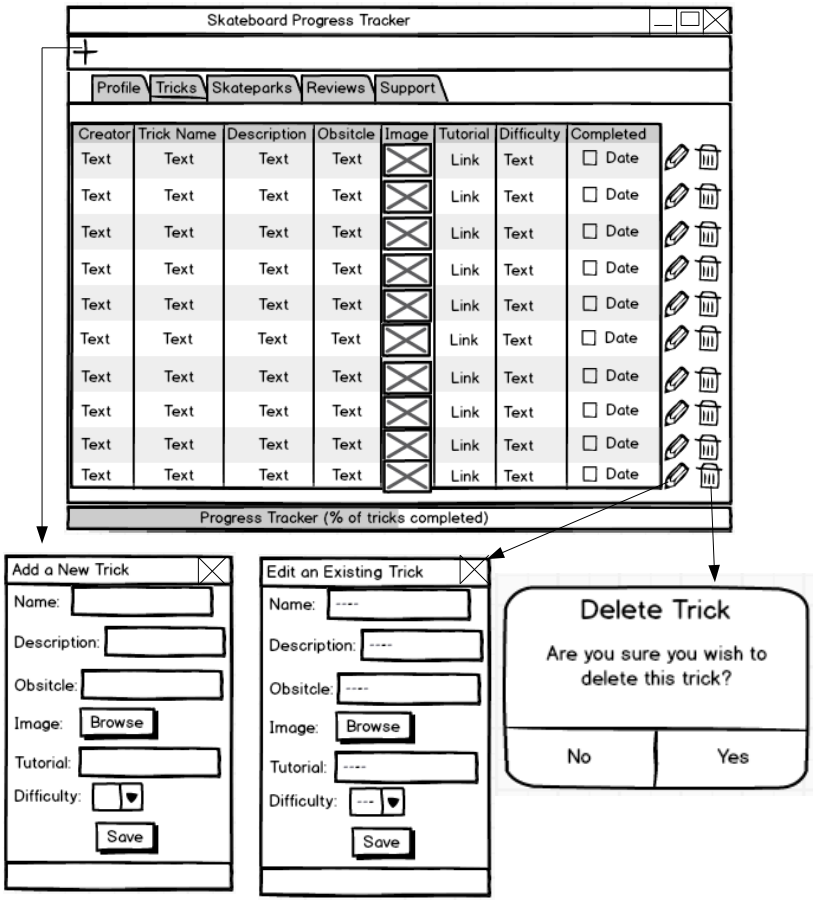


Figure 2.11: The User Interface for the trick section

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.

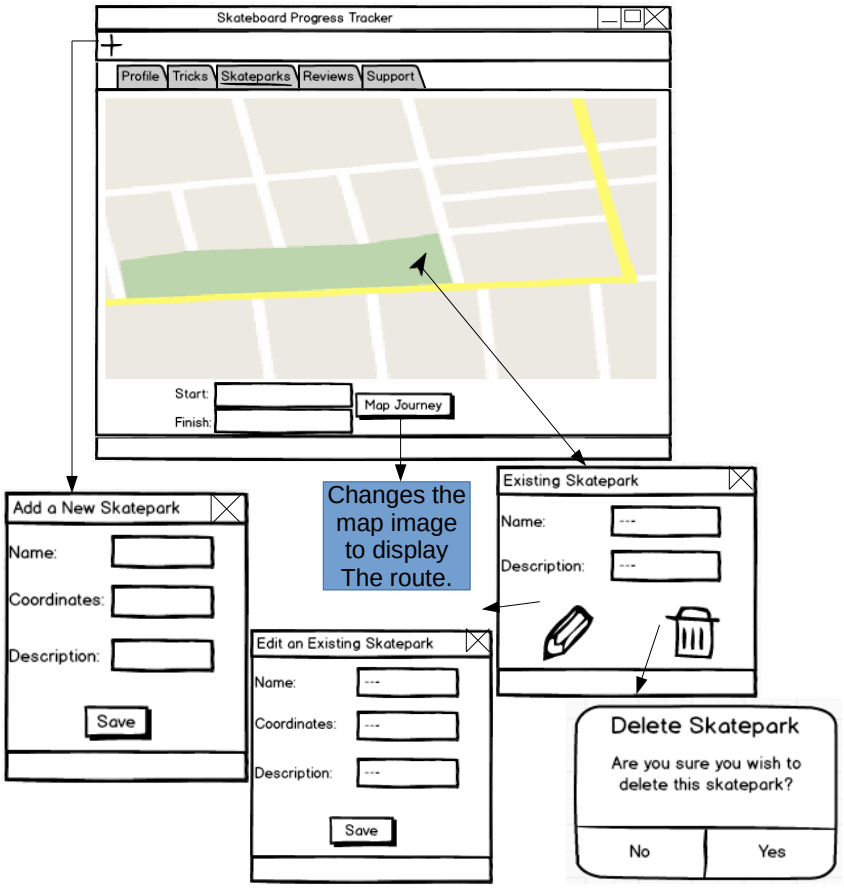


Figure 2.12: The User Interface for the skatepark section

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.

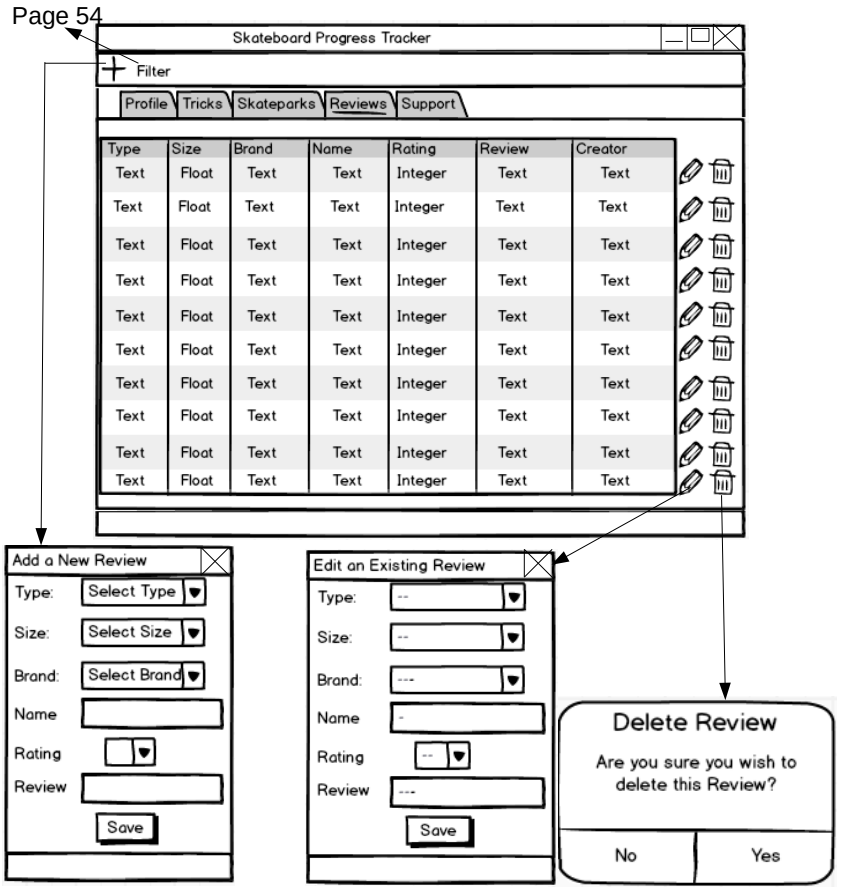


Figure 2.13: The User Interface for the review section

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.

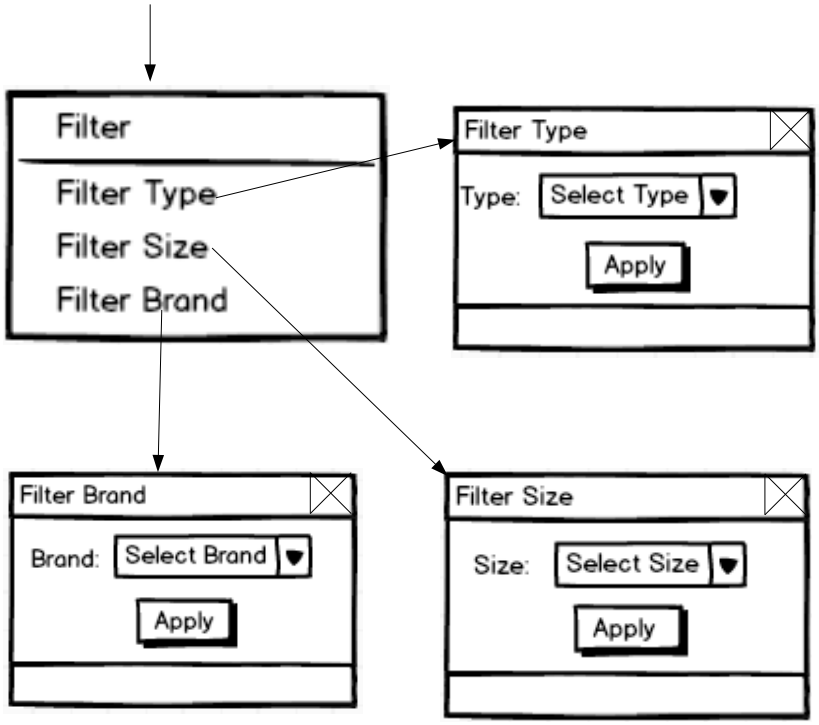


Figure 2.14: The continued User Interface for the review section

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.

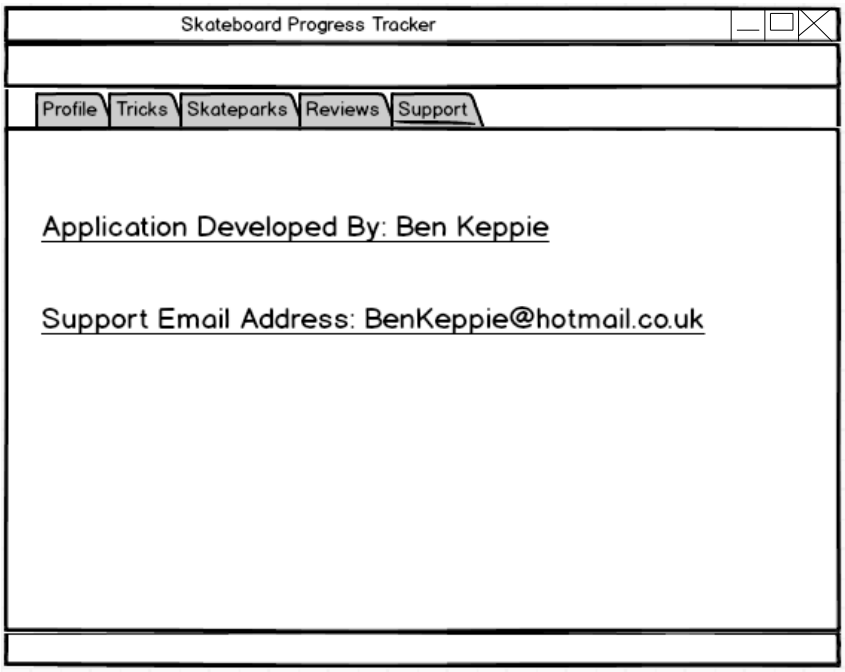


Figure 2.15: The User Interface for the support section

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 applcation 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 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

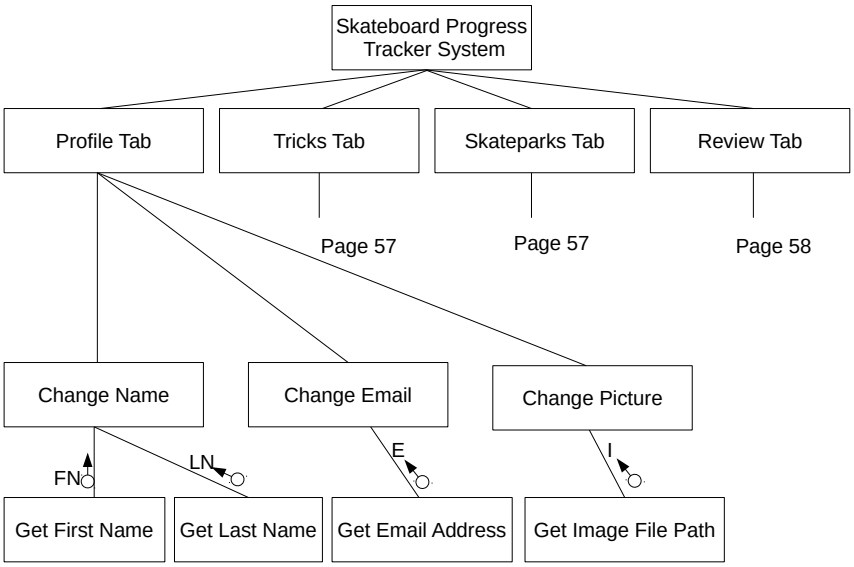


Figure 2.16: Profile Top-Down Design Chart

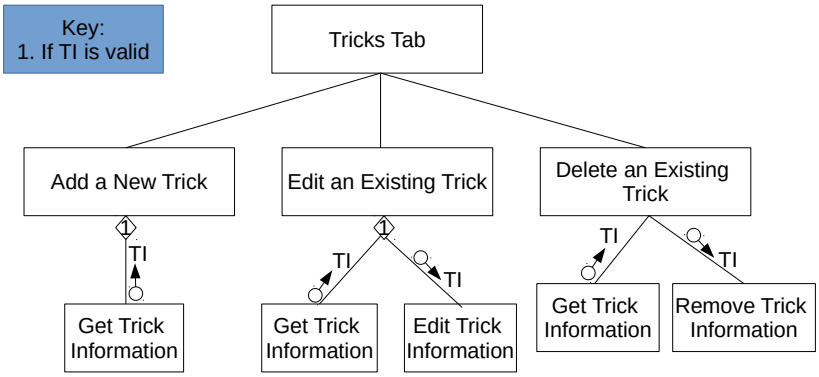


Figure 2.17: Tricks Top-Down Design Chart

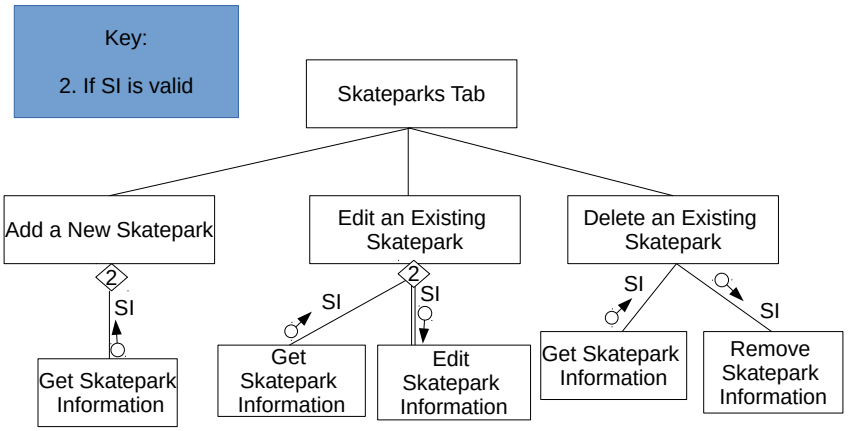


Figure 2.18: Skateparks Top-Down Design Chart

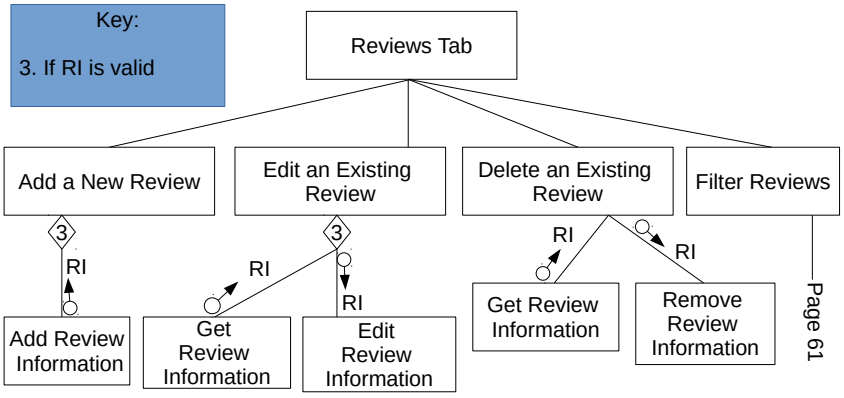


Figure 2.19: Reviews Top-Down Design Chart

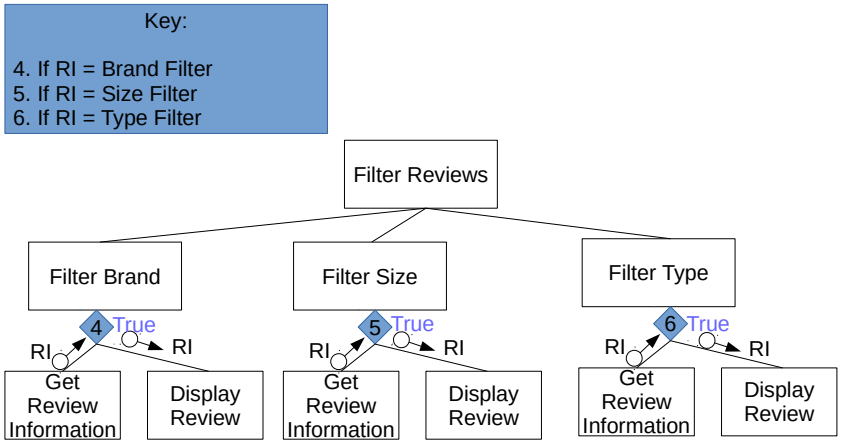


Figure 2.20: Review Filters Top-Down Design Chart

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

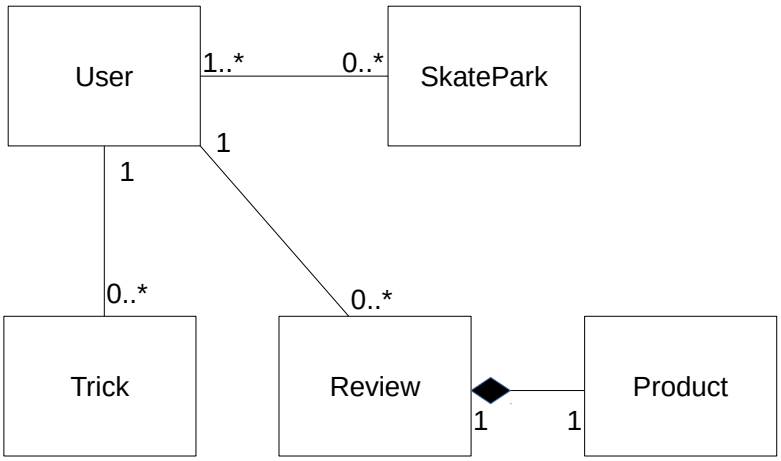


Figure 2.21: Relationship Diagram

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' seciton 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.

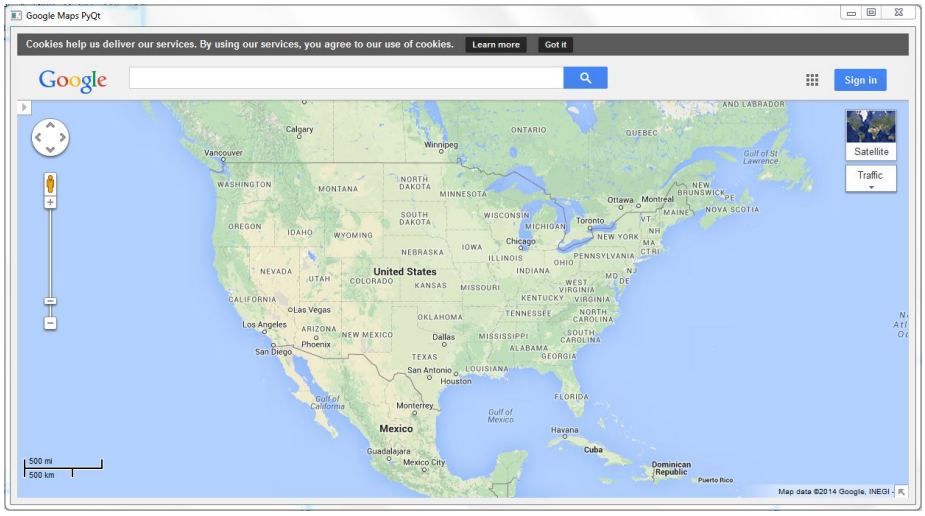


Figure 2.22: Google Maps in Python Application

My code for this is shown below.

```
import sys
from PyQt4.QtGui import *
from PyQt4.QtCore import *
from PyQt4.QtWebKit import *

class MainWindow(QMainWindow):
    """The main window for my application"""
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Google Maps PyQt")
        self.create_layout()

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

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

        self.widget=QWidget()
        self.widget.setLayout(self.layout)
        self.setCentralWidget(self.widget)

if __name__ == "__main__":
    application=QApplication(sys.argv)
    window=MainWindow()
    window.show()
    window.raise_()
    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.

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

```
        &q=long+road+sixth+form+college
        &attribution_source=Google+Maps+Embed+API
        &attribution_web_url=http://www.butchartgardens.com/
        &attribution_ios_deep_link_id=comGooglemaps://?daddr=long+road+sixth
        +form+college"> </iframe>''')
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.

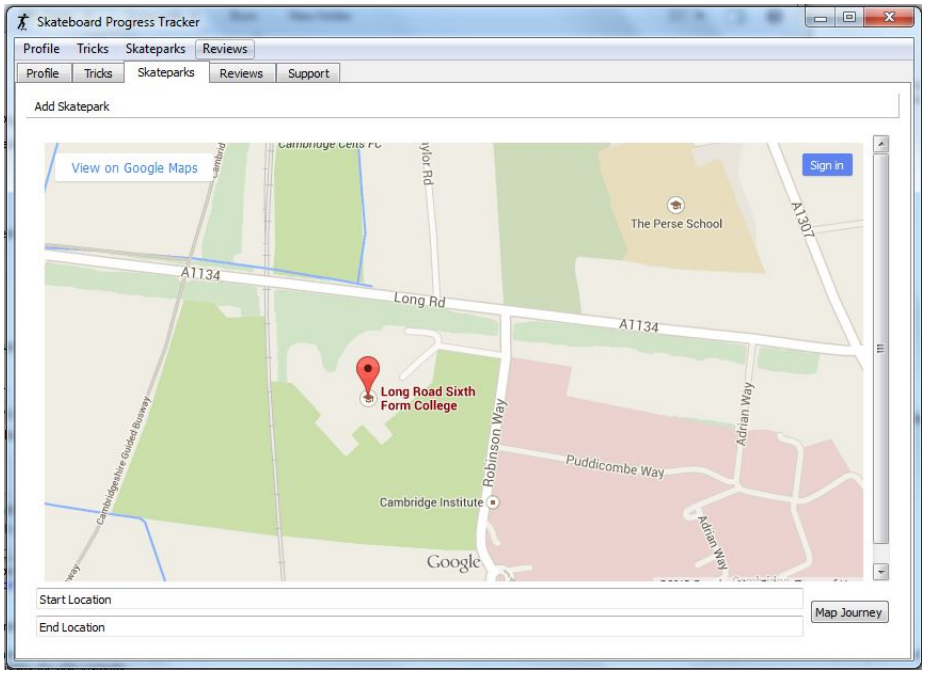


Figure 2.23: Pin Marker on Embedded Google Maps

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:

```
import sys
from PyQt4.QtGui import *
from PyQt4.QtCore import *

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Tabbed Interface")
        self.create_tabs()

    def create_tabs(self):

        self.tabs=QTabWidget()

        #Create Widgets
        self.profile_tab=QWidget()
        self.tricks_tab=QWidget()
        self.skateparks_tab=QWidget()
        self.reviews_tab=QWidget()
        self.support_tab=QWidget()

        #Add Tabs
        self.tabs.addTab(self.profile_tab, "Profile")
        self.tabs.addTab(self.tricks_tab, "Tricks")
        self.tabs.addTab(self.skateparks_tab,
            "Skateparks")
        self.tabs.addTab(self.reviews_tab, "Reviews")
        self.tabs.addTab(self.support_tab, "Support")

        self.setCentralWidget(self.tabs)

if __name__=="__main__":
    application=QApplication(sys.argv)
    window=MainWindow()
    window.show()
    window.raise_()
    application.exec_()
```

Figure 2.24: Tab Navigation Code

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

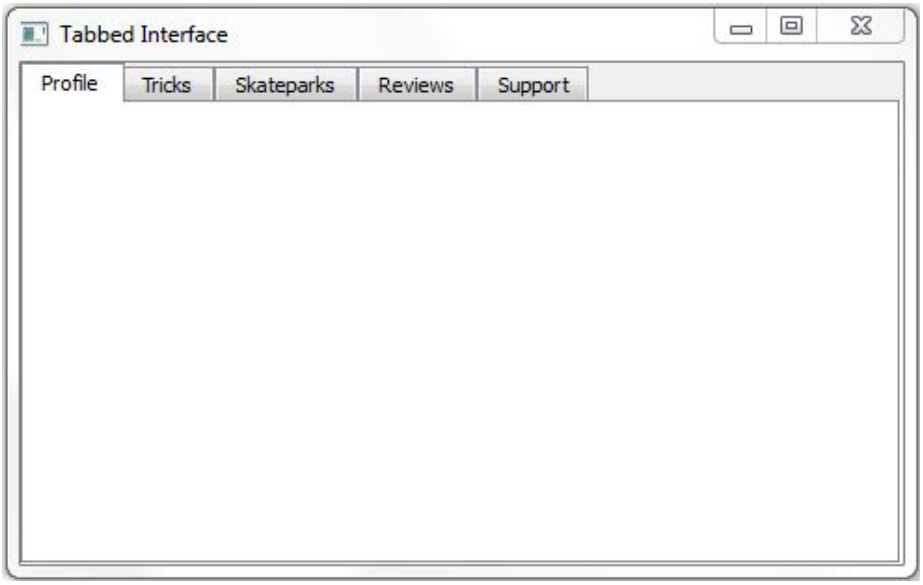


Figure 2.25: Tabbed Navigation in Python Application

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=("{0}{1}".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
TrickObsitcle	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 the 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

Name	Data Type	Length	Validation	Example Data	Comment
UserID	Interger	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.uk	None
TrickCreator	String	41 Characters	Adds First and last name	Ben Keppie	None
TrickID	Interger	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 acessible 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

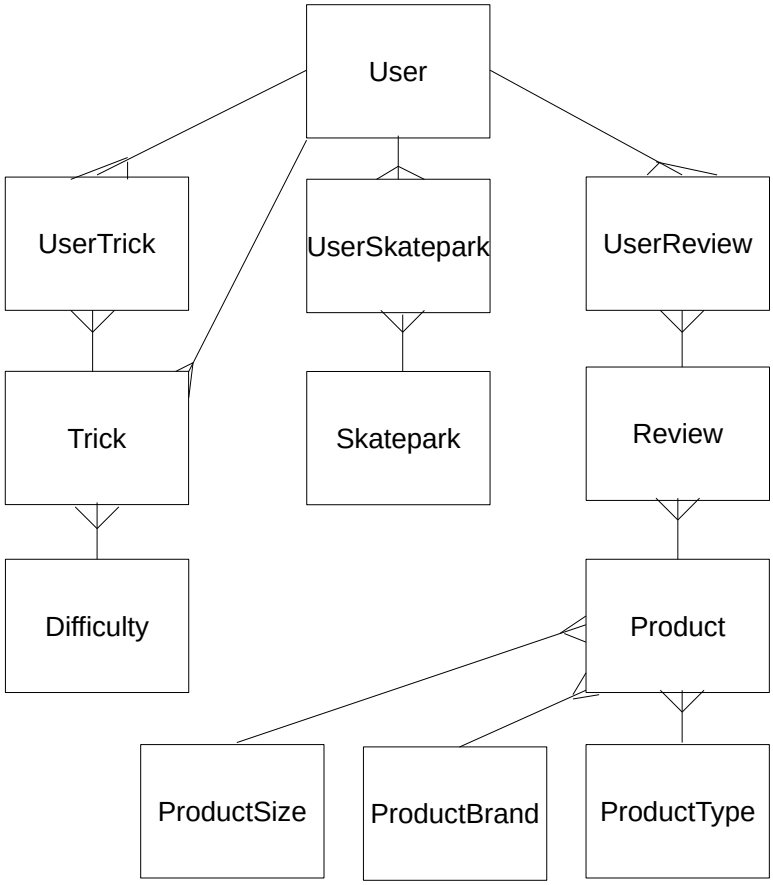


Figure 2.26: Entity-Relationship Diagram

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

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

ReviewID *ProductID* ReviewDescription ReviewRating ReviewCreator (UserID)
ProductBrandID ProductBrand
ProductTypeID ProductType
ProductSizeID 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.

```
SELECT *
FROM Review, Product
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.

```
SELECT *
FROM Review, Product
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.

```
SELECT *
FROM Review, Product
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.

```
SELECT TrickID
FROM Trick
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.

```
SELECT TrickID
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.

```
SELECT *
FROM Trick
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, periodicly 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 obsitcle 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

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 habe 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 corrected	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 in the existing skatepark pop-up) functions properly	This should load a pop-up to edit a skatepark	Click the pencil in the existing skatpark 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 skatpark 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 sould 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.JPEG 2.Picture.PNG 3.Picture.txt	1.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@hotmail.co.uk 2.BenKeppieEmail.com	3.Jim290.co.uk	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.JPEG 2.Picture.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/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)		
2.08	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.co.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 databse	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 databse	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 displated 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

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 corrected	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 in the existing skatepark pop-up) functions properly	This should load a pop-up to edit a skatepark	Click the pencil in the existing skatpark 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 skatpark 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 sould 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.JPEG 2.Picture.PNG 3.Picture.txt	1.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@hotmail.co.uk 2.BenKeppieEmail.com 3.Ji1290.co.uk	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 Trick.JPGs	1.Picture.JPEG 2.Picture.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/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)		
2.08	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.co.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 databse	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 databse	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

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.JPEG 2.Picture.PNG 3.Picture.txt	1.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.JPEG 2.Picture.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)		
2.08	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.co.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 databse	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 databse	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

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@hotmail.com 2.BenKeppieEmail.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 Series	**Purpose of Test**	**Test Description**	**Test Data**	**Test Data Type (Normal/ Erroneous/ Boundary)**	**Expected Result**	**Actual Result**	**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/YYY	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

- 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	Figure 3.2 on page 153.
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 3.3 on page 154
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 3.4 on page 155, Figure 3.5 on page 156
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 3.6 on page 157
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 3.7 on page 159 and Figure3.8 on page 160

2.01	Verify an appropriate picture is selected in the 'Change Picture' pop-out	Should only accept Trick.JPGs	1.Picture.JPEG 2.Picture.PNG 3.Picture.txt	1.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@hotmail.co.uk 2.BenKeppieEmail.com 3.Ji1290.co.uk	1.Normal 2. 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.Error(Presence)	1.Passed 2.Passed	Figure 3.9 on page 161, Figure 3.10 on page 162
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)	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.Error(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.JPEG 2.Picture.PNG 3.Picture.txt	1.Normal 2.Erroneous 3.Erroneous	1.Accept 2.Error (File Type) 3.Error (File Type)	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 3.11 on page 163, Figure 3.12 on page 164, Figure 3.13 on page 165
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	Figure 3.14 on page 166 and Figure 3.15 on page 167.
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 3.16 on page 168

3.01	Verify the profile picture is inputted into the database	A Trick.JPG should be added to the database	ProfilePicture.JPG	Normal	Accept	File path was added to the database	Figure 3.17 on page 169
3.02	Verify an email is inputted into the database	An email should be added to the database	BenKeppie@hotmail.co.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 3.18 on page 170
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.JPG	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 databse	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 3.19 on page 171
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 databse	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 3.20 on page 172, Figure 3.21 on page 173

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 3.22 on page 174).

3.3.2 Evidence

Test 1.00 Evidence

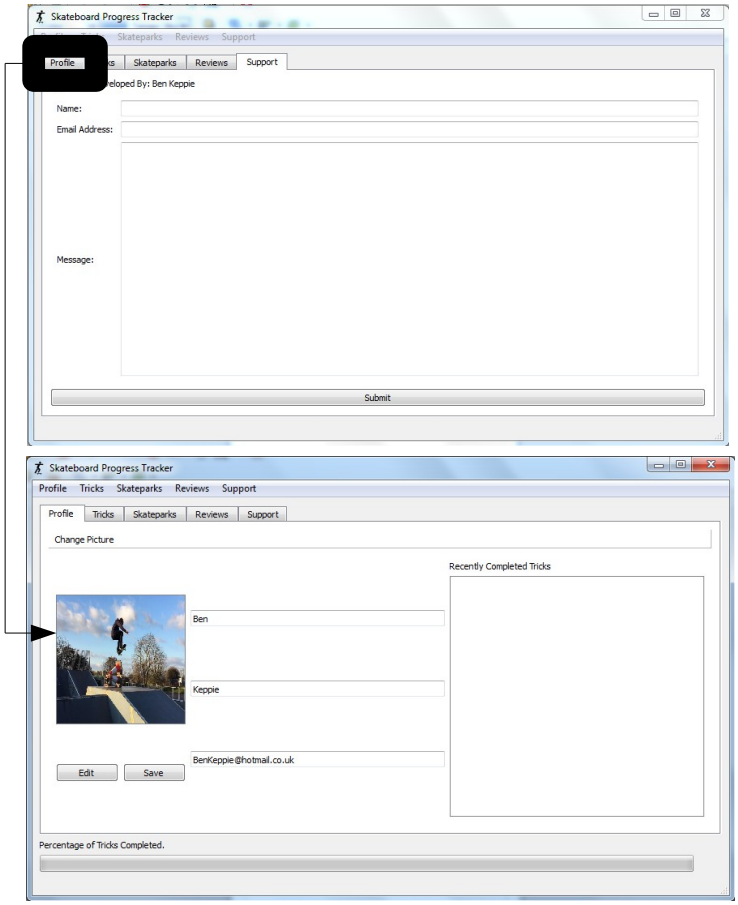


Figure 3.1: Evidence for Test 1.00

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

Test 1.03 Evidence

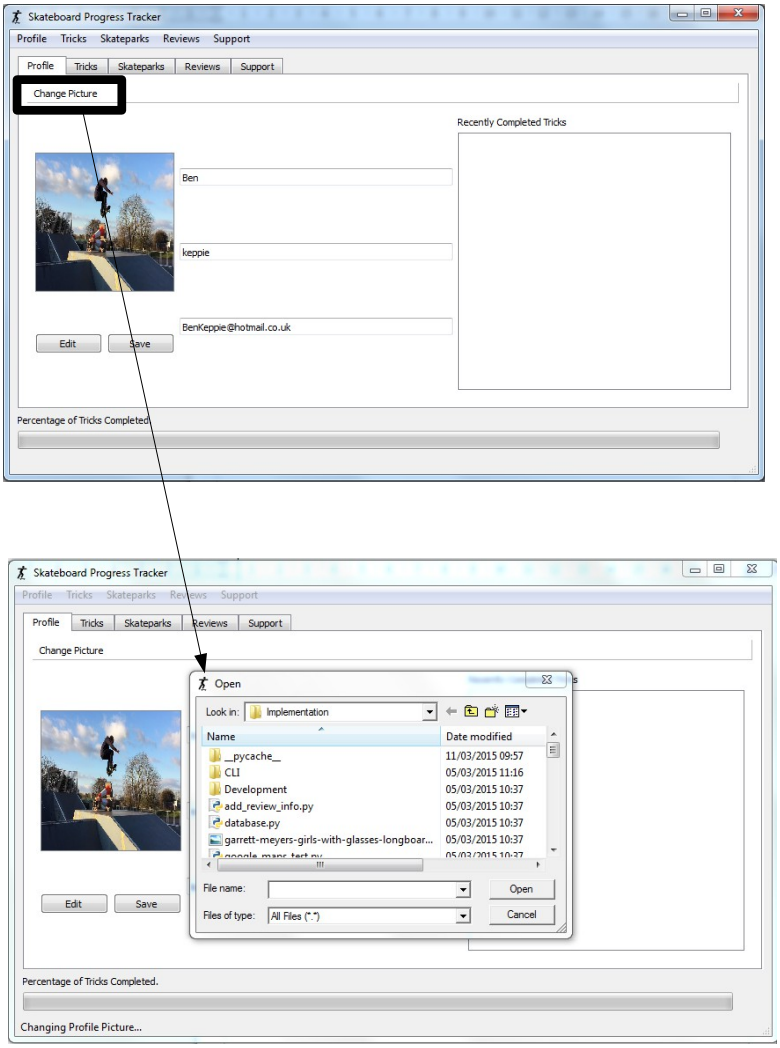


Figure 3.2: Evidence for Test 1.03

This test shows that in the profile tab, when the 'change picture' button on the toolbar is pressed, a QFileDialog appears. This dialog allows you to pick a file to use as your profile picture. This test was successful.

Test 1.05 Evidence

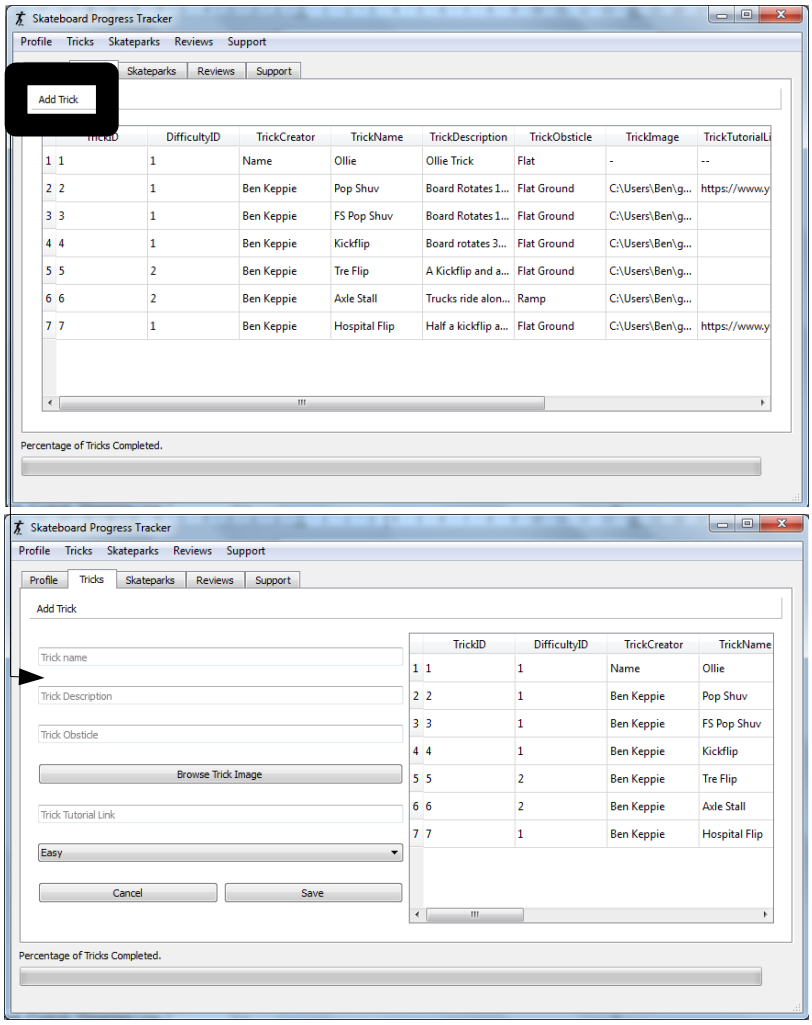


Figure 3.3: Evidence for Test 1.05

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

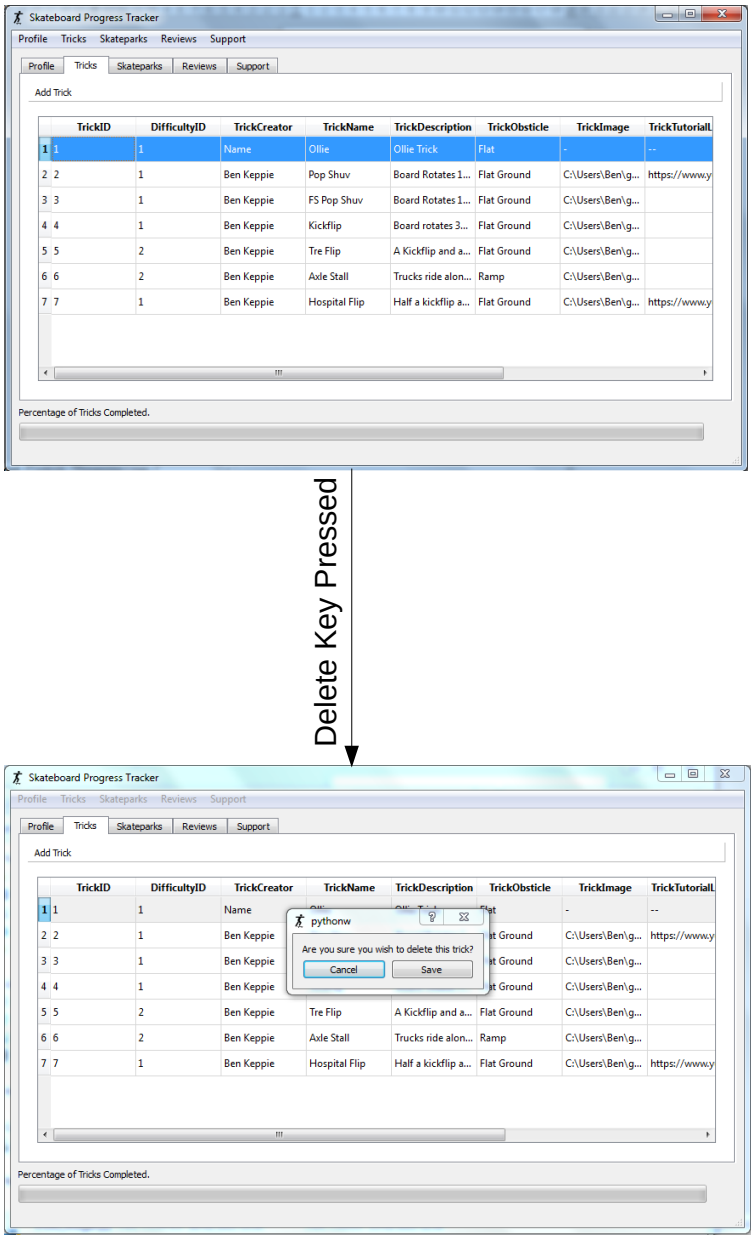


Figure 3.4: Evidence for Test 1.07

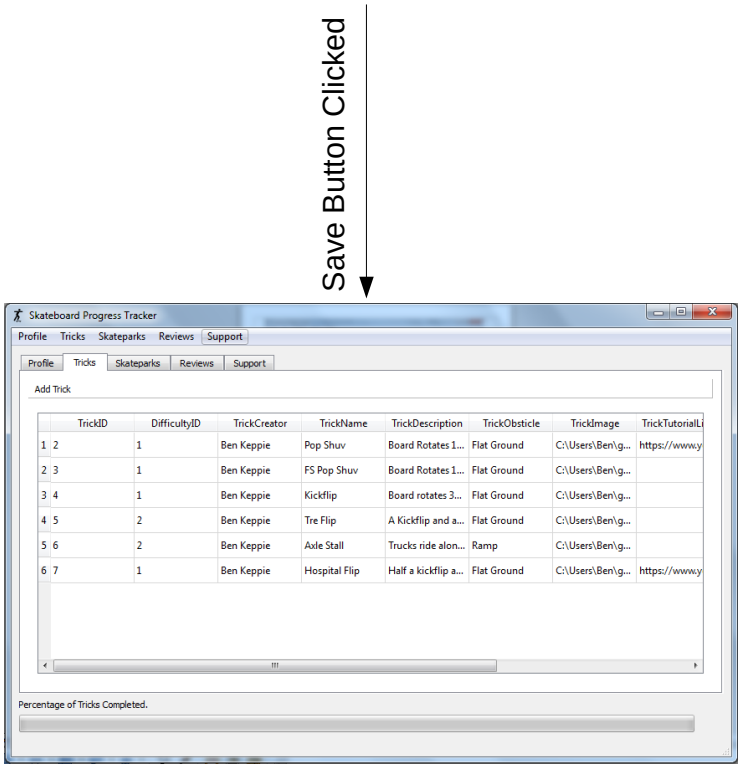


Figure 3.5: Evidence for Test 1.07 Part 2

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 is '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

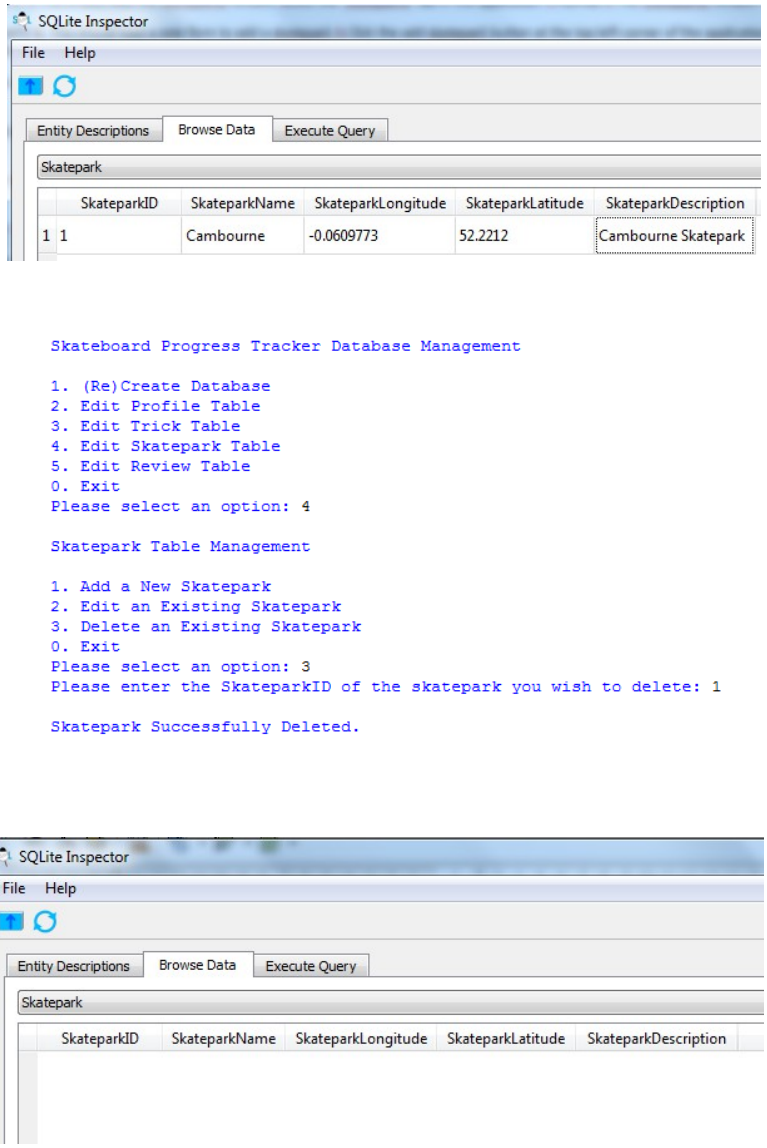


Figure 3.6: Evidence for Test 1.12

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

Test 2.00 Evidence

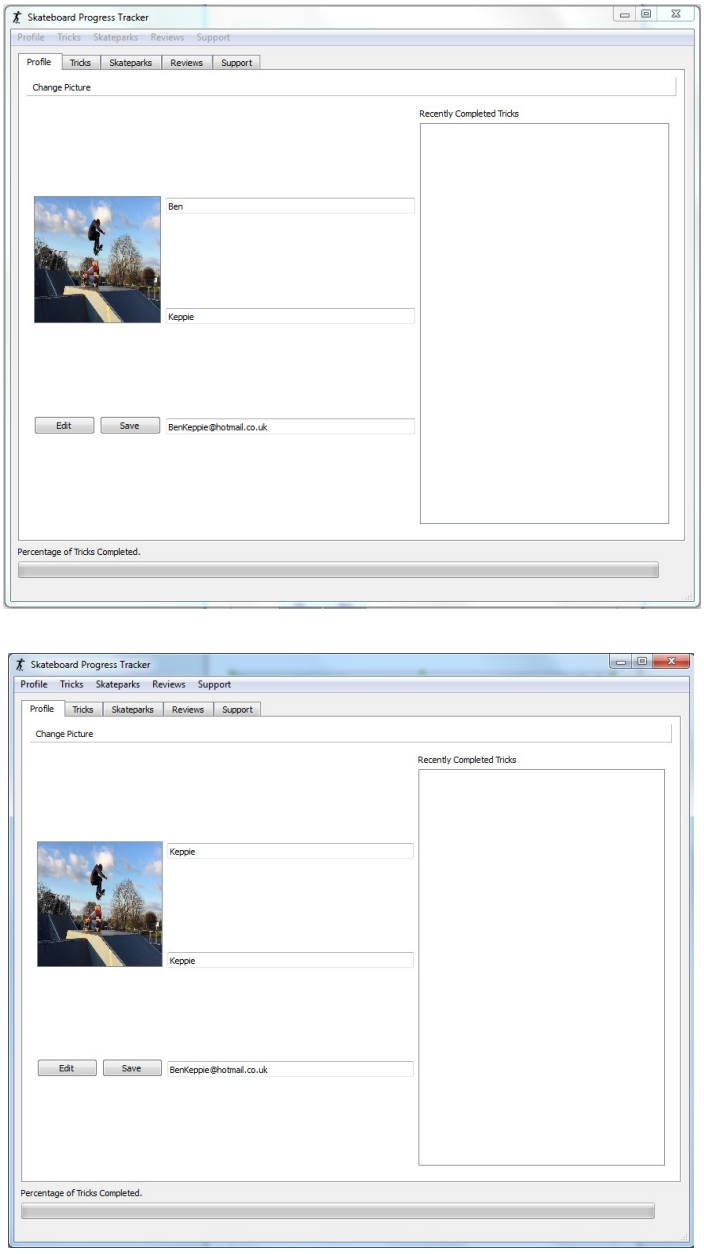


Figure 3.7: Evidence for Test 2.00 Part 1

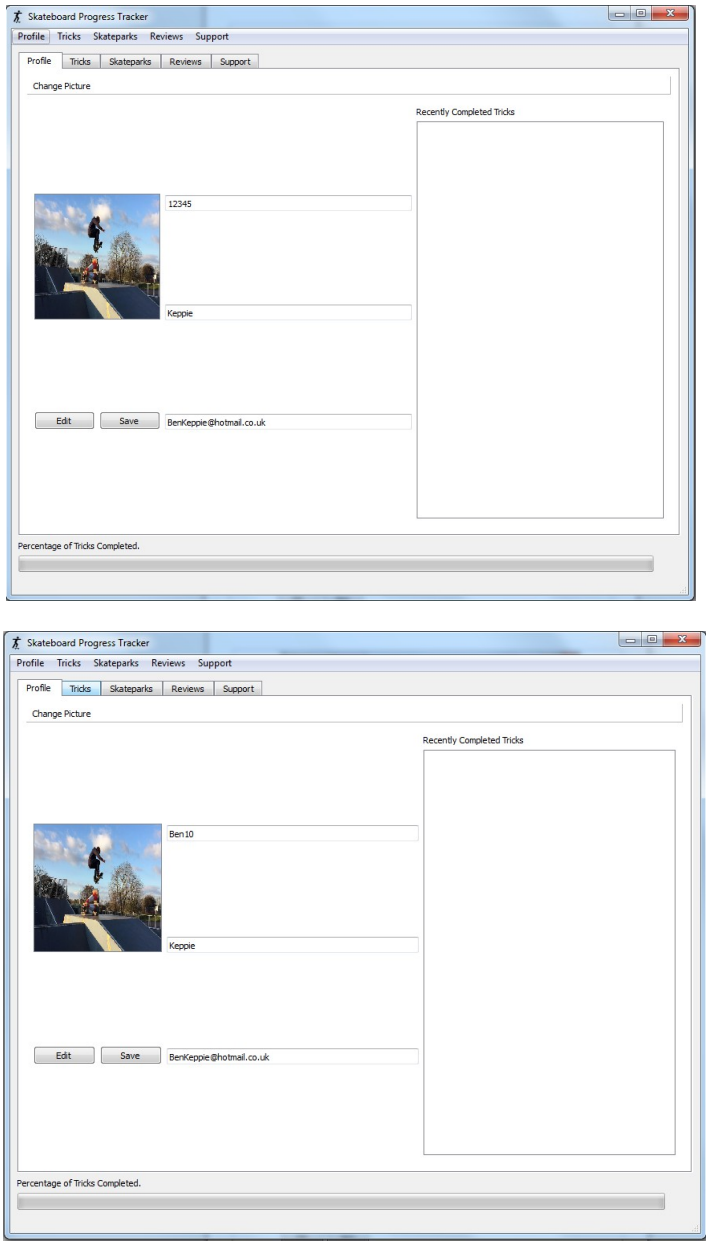


Figure 3.8: Evidence for Test 2.00 Part 2

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

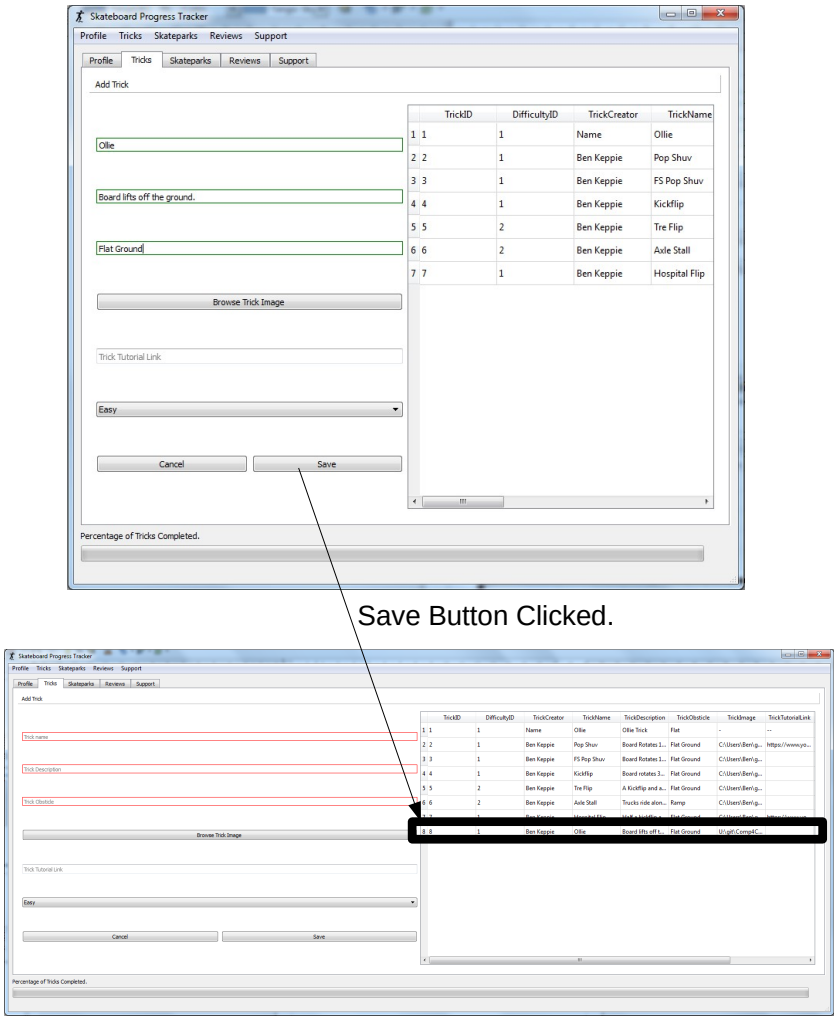


Figure 3.9: Evidence for Test 2.03 Part 1

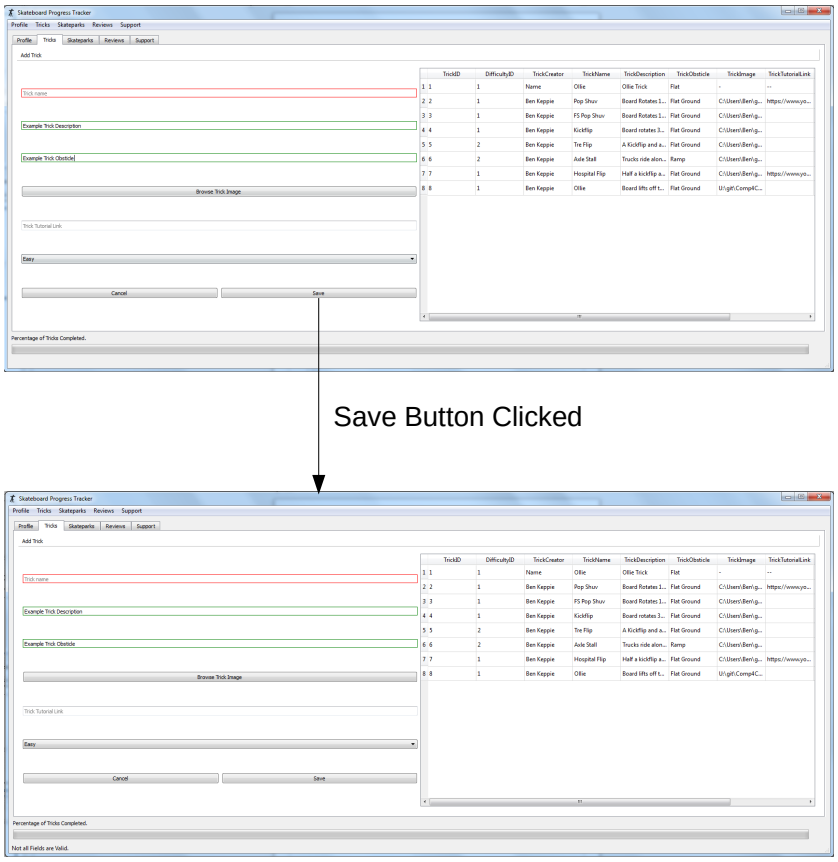


Figure 3.10: Evidence for Test 2.03 Part 2

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

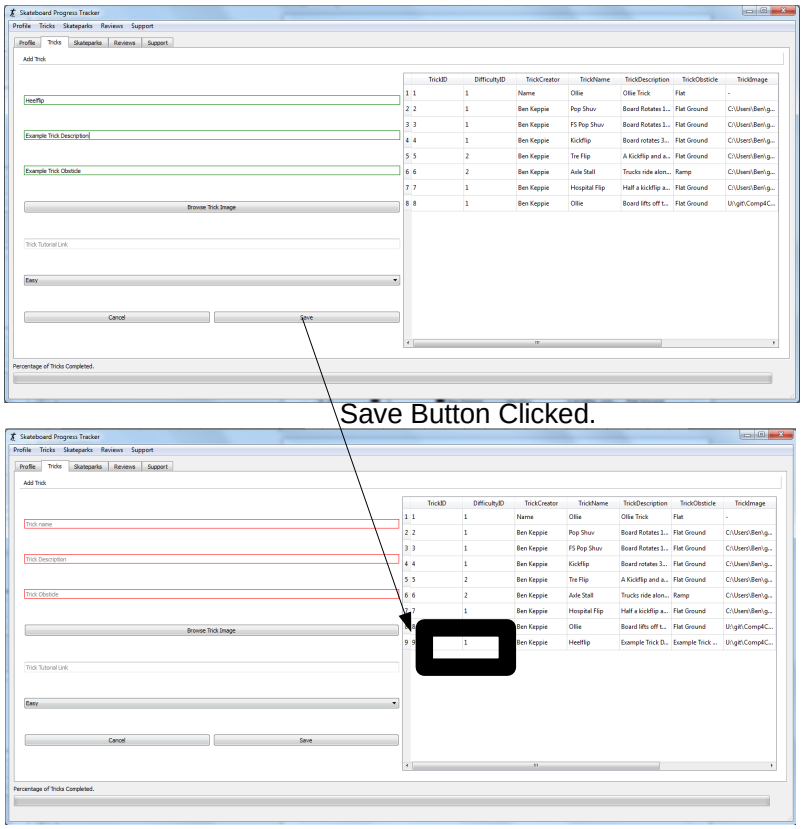


Figure 3.11: Evidence for Test 2.06 Part 1

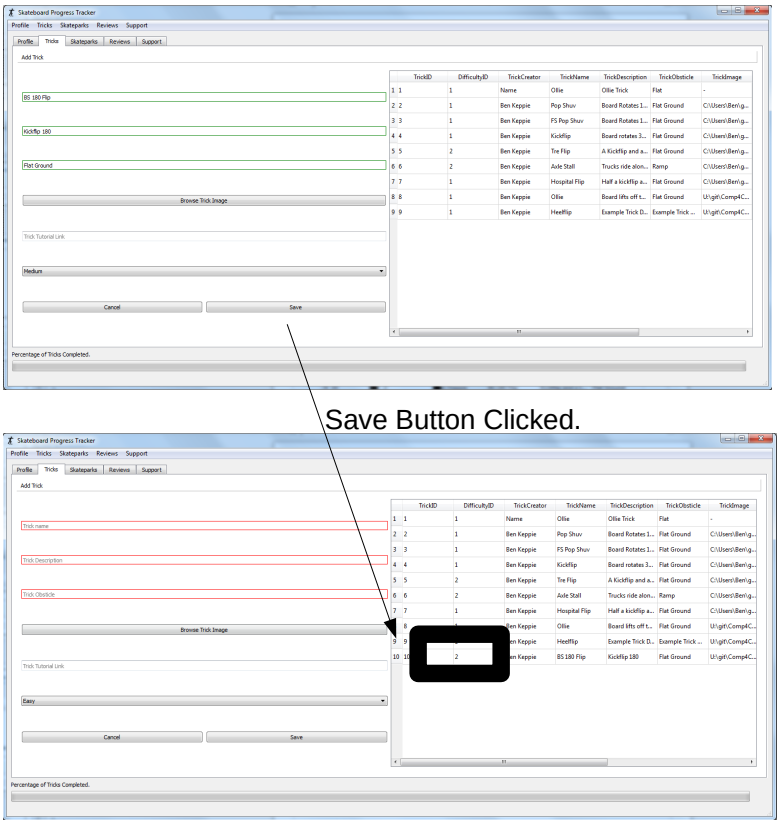


Figure 3.12: Evidence for Test 2.06 Part 2

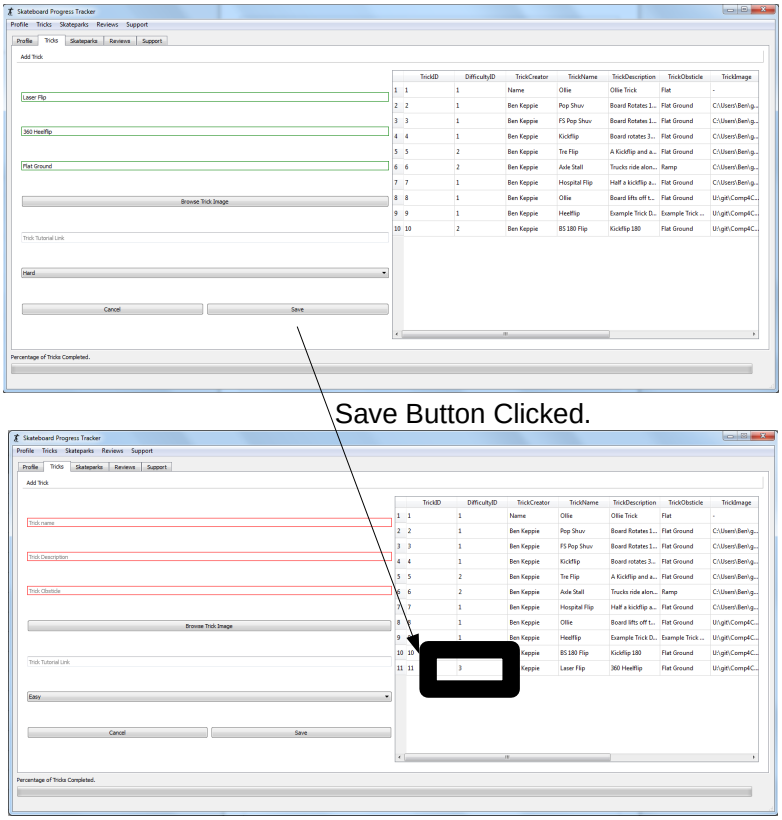


Figure 3.13: Evidence for Test 2.06 Part 3

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 2.08 Evidence

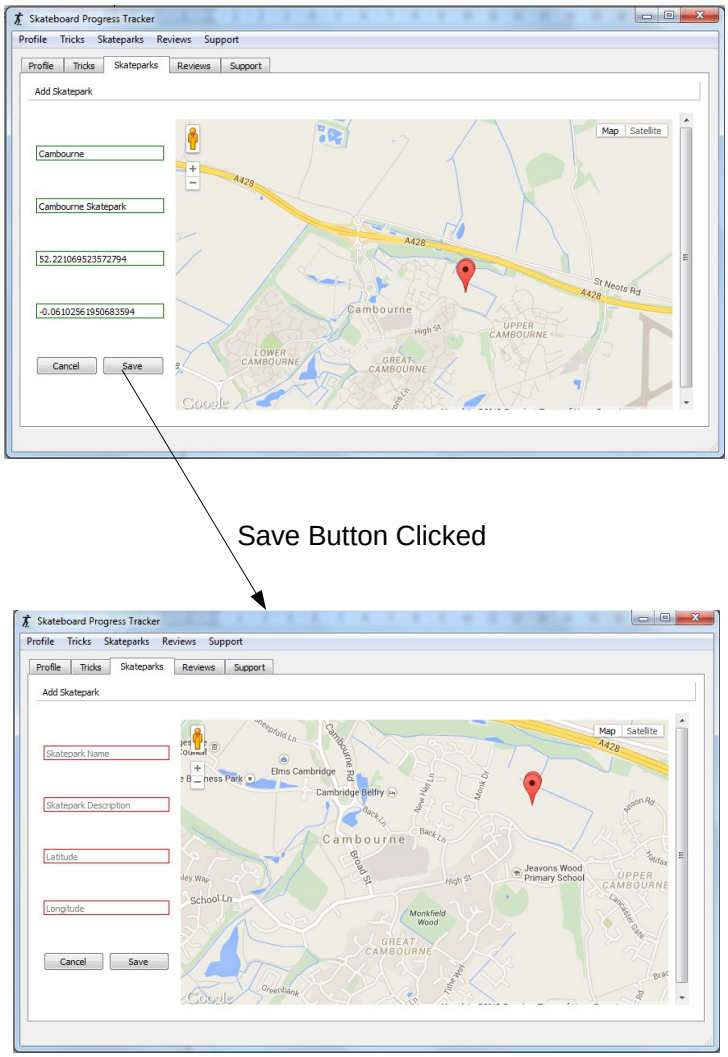


Figure 3.14: Evidence for Test 2.08 Part 1

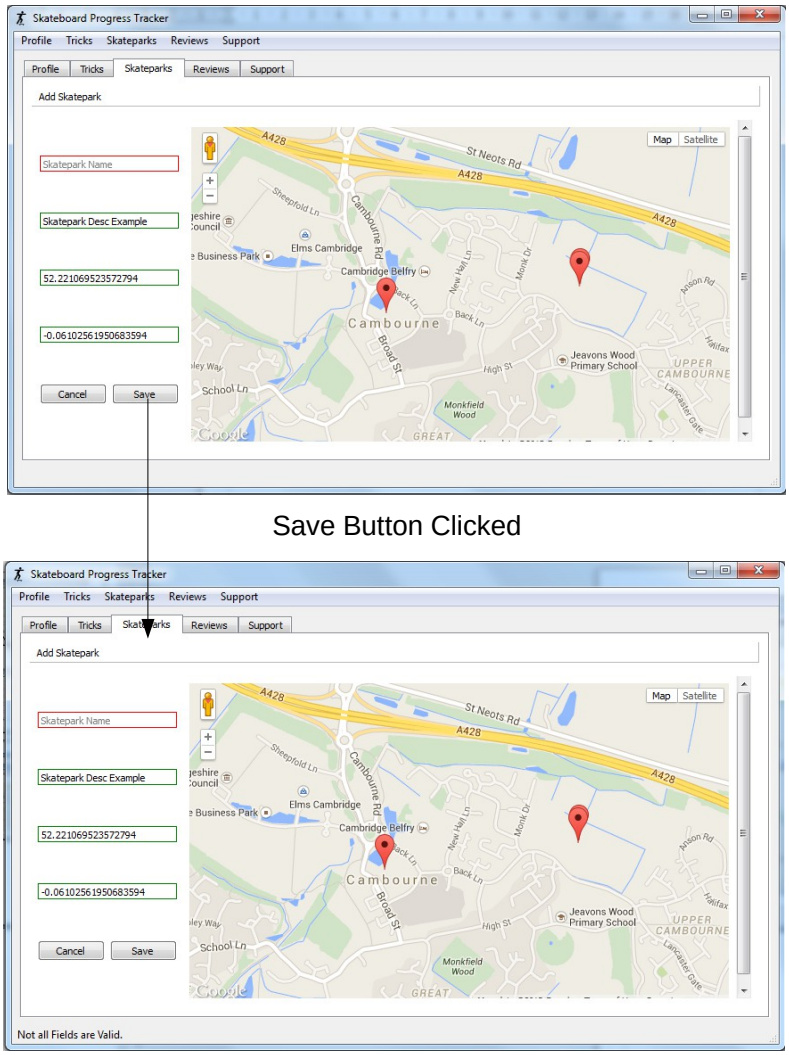
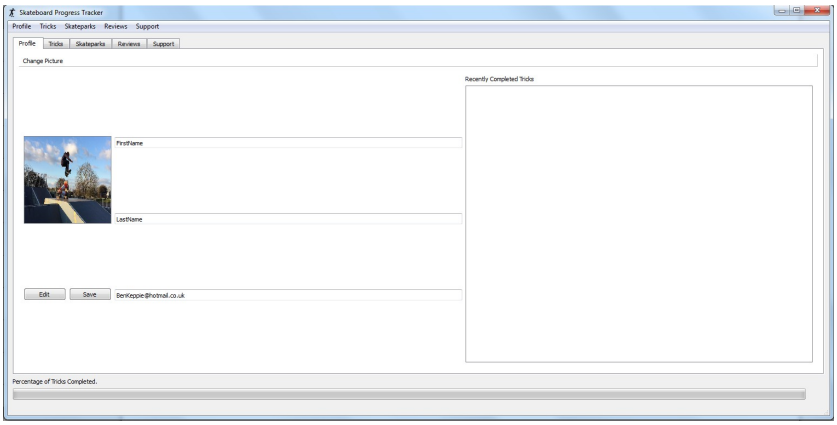


Figure 3.15: Evidence for Test 2.08 Part 2

The screen shots above show that in the 'add skatepark' functionality accepts the process of adding a skatepark when the skatepark name field is filled in; however if it is left empty the status bar displays an error message saying not all fields are valid, and also a red border is around the field, showing the user that the field is not valid. This test was successful.

Test 3.00 Evidence



User

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

Figure 3.16: 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.

3.01 Evidence

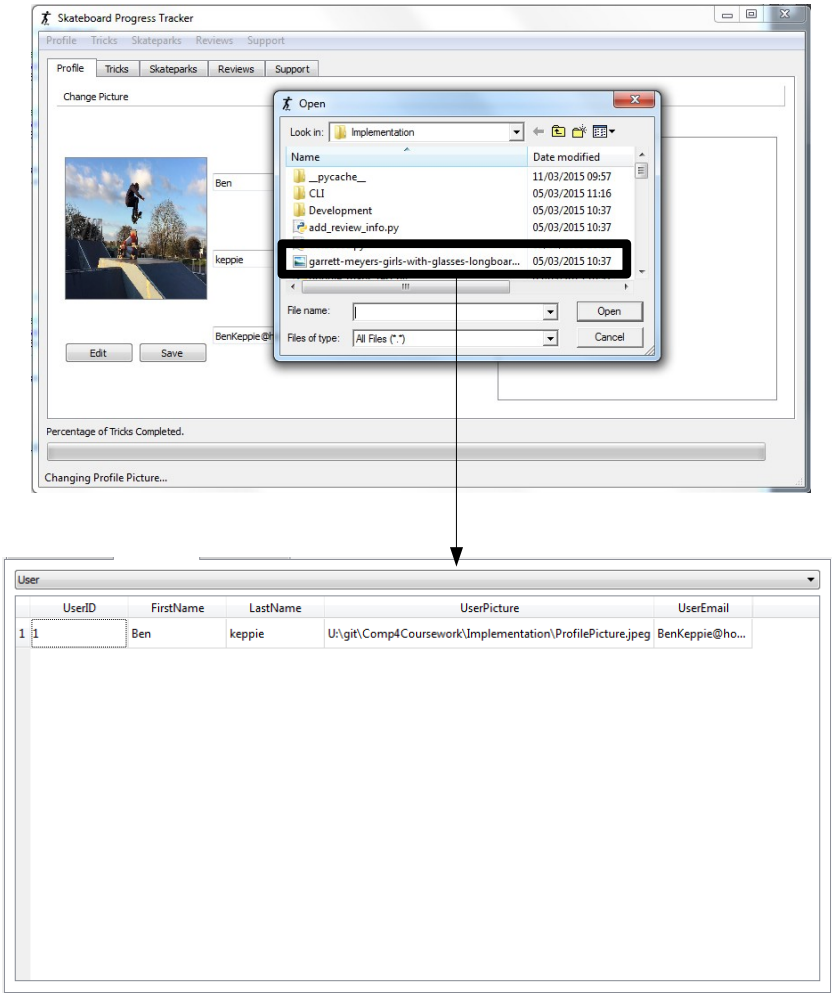


Figure 3.17: Evidence for Test 3.01

The screen shot above shows that when an image file is chosen to change the profile picture it gets copied and stored into the database and 'ProfilePicture.jpg'. I chose to do this in case the user deleted the original image, therefore the profile of the program would still have an image to use. This test was successful.

Test 3.03 Evidence

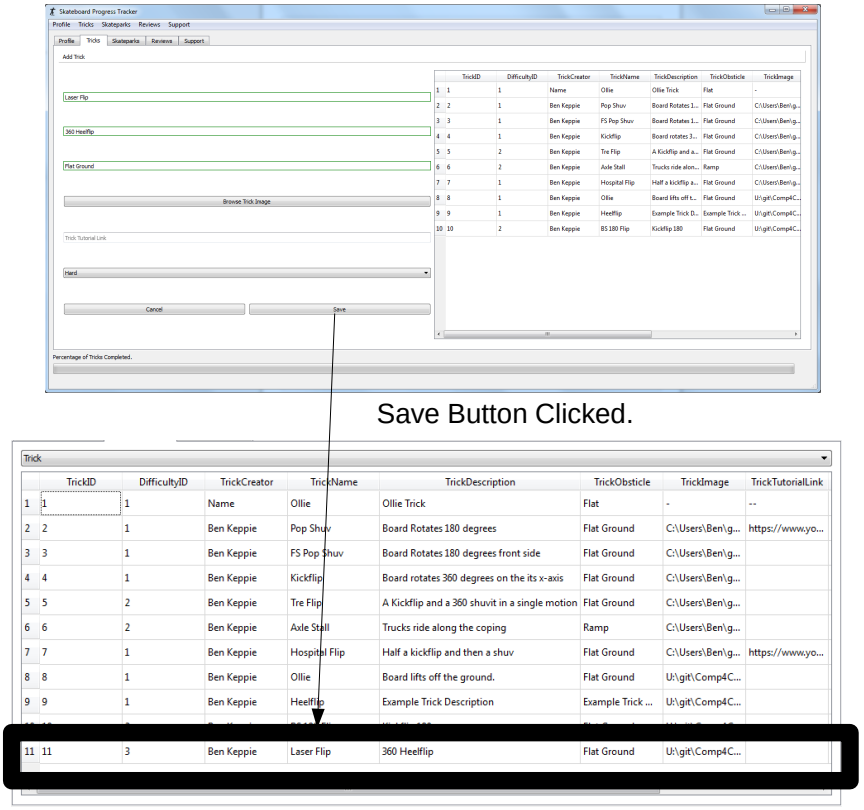


Figure 3.18: Evidence for Test 3.03

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

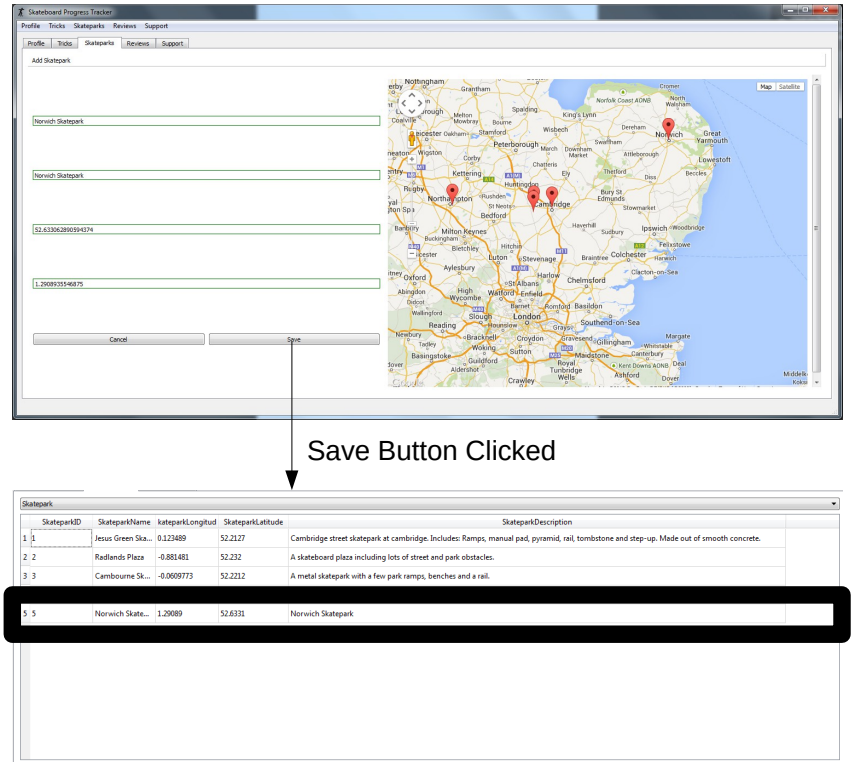


Figure 3.19: Evidence for Test 3.09

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

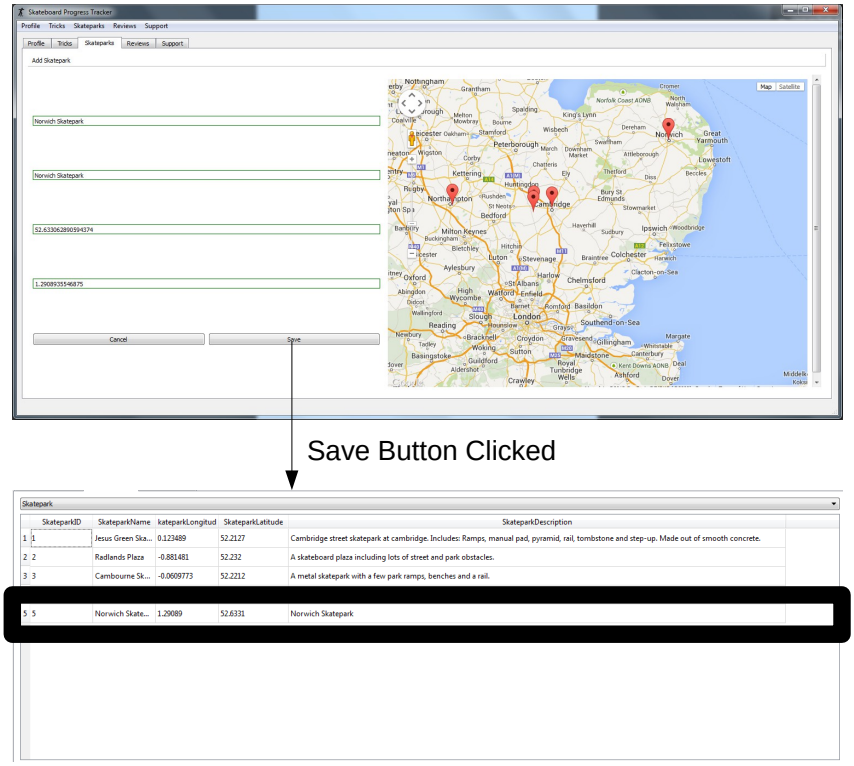


Figure 3.20: Evidence for Test 4.05 Part 1

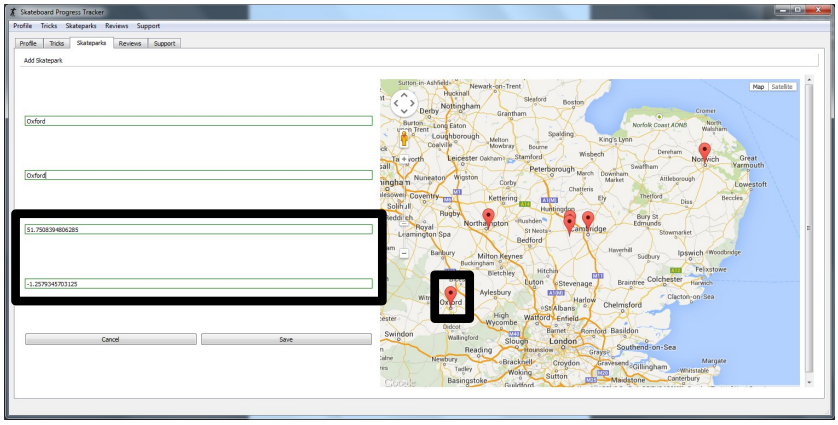
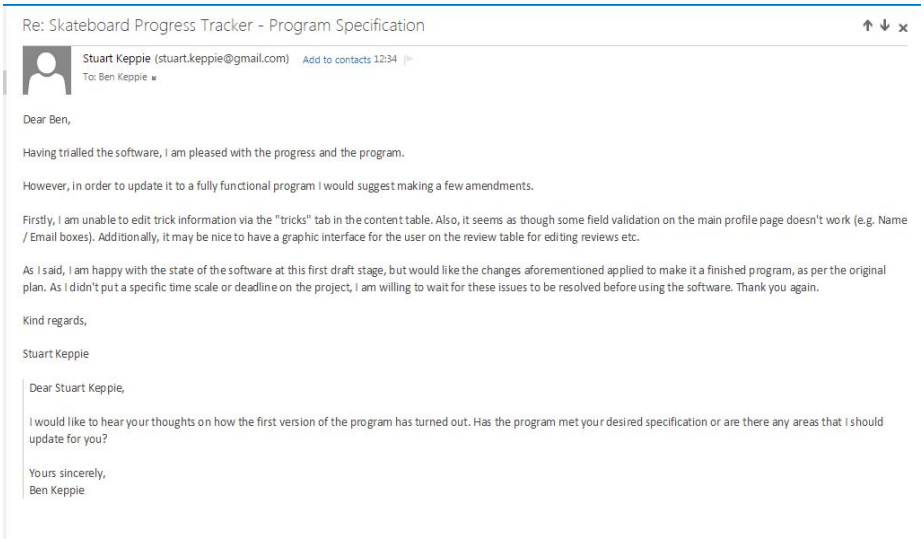


Figure 3.21: Evidence for Test 4.05 Part 2

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 3.7 on page 159 and Figure 3.8 on page 160). On the other hand most of my tests passed e.g saving tricks (Figure 3.18 on page 170). This was also brought to my attention by my client as shown by an email conversation below.


Re: Skateboard Progress Tracker - Program Specification

Stuart Keppie (stuart.keppie@gmail.com) Add to contacts 12:34
To: Ben Keppie

Dear Ben,

Having trialled the software, I am pleased with the progress and the program.

However, in order to update it to a fully functional program I would suggest making a few amendments.

Firstly, I am unable to edit trick information via the "tricks" tab in the content table. Also, it seems as though some field validation on the main profile page doesn't work (e.g. Name / Email boxes). Additionally, it may be nice to have a graphic interface for the user on the review table for editing reviews etc.

As I said, I am happy with the state of the software at this first draft stage, but would like the changes aforementioned applied to make it a finished program, as per the original plan. As I didn't put a specific time scale or deadline on the project, I am willing to wait for these issues to be resolved before using the software. Thank you again.

Kind regards,

Stuart Keppie

> Dear Stuart Keppie,
>
> I would like to hear your thoughts on how the first version of the program has turned out. Has the program met your desired specification or are there any areas that I should update for you?
>
> Yours sincerely,
> Ben Keppie

Figure 3.22: 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 its the only way to ensure that the fundamental programming asepcts 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, for example I had 17 tests for my first test series (shown in my test table 3.5 on page 151. The use of multiple different testing types (Top-down, bottom-up,white box, black box and acceptance testing) 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 Pyton, 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 QLineEdits showing the users first name, last name and email address, to the right of these QLineEdits 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 QLineEdits containing the first name, last name and email address of the user become available to edit. Once the QLineEdits 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 QLineEdits, 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 thorugh 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 QLineEdits 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 QLineEdits 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.

```
class MainWindow(QMainWindow):
    """Class for the main window of my program"""
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Skateboard Progress
            Tracker")
        self.app = QtGui.QApplication([])
        self.set_icon()
        self.main_VBoxLayout=QVBoxLayout()
        self.central_widget=QWidget()
        self.ProgressBar=QProgressBar()
        self.ProgressBarLabel=QLabel("Percentage of
            Tricks Completed.")


        #create tabs
        self.create_tabs()

        #Create Menu Bar
        self.Menu=Menu(self)
        self.setMenuBar(self.Menu.MenuBar)

        #Create Statusbar
        self.StatusBar=QStatusBar()
        self.setStatusBar(self.StatusBar)

        self.tabs.currentChanged.connect(self.progress_bar_hide)


    def progress_bar_hide(self):
        if (self.tabs.currentIndex() >=2):
            self.ProgressBarLabel.hide()
            self.ProgressBar.hide()
        else:
            self.ProgressBar.show()
            self.ProgressBarLabel.show()


```

```
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
49      def create_tabs(self):
50
51
52          self.tabs=QTabWidget()
53
54          #Create Widgets
55          self.profile_tab=DisplayProfileWidget(self)
56
57          self.tricks_tab=DisplayTricksWidget(self)
58          self.skateparks_tab=DisplaySkateparksWidget(self)
59          self.reviews_tab=DisplayReviewsWidget(self)
60          self.support_tab=DisplaySupportWidget(self)
61
62
63          #Add Tabs
64          self.tabs.addTab(self.profile_tab, "Profile")
65          self.tabs.addTab(self.tricks_tab, "Tricks")
66          self.tabs.addTab(self.skateparks_tab,
                "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.

```
class CustomQTabWidget(QTabWidget):
    """A class for my custom QTabWidget"""
    def __init__(self,parent):
        super().__init__()
        self.parent=parent

    def currentChanged(self):
        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.

```
class DisplayTricksToolbar(QToolBar):
    def __init__(self, parent):
        super().__init__()
        self.parent=parent

        self.add_trick=QAction("Add Trick",self)

        self.addAction(self.add_trick)

            #connections
        self.add_trick.triggered.connect(self.add_trick_connection)


```

```
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.

```
def main():
    application=QApplication(sys.argv)
    window=MainWindow()
    #splash_screen()


    window.show()
    window.raise_()
    application.exec_()
    print()



if __name__=="__main__":
    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
TrickObsticleExpression	Holds the compiled regular expression for the trick obsticle	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.

4.5 System Evidence

4.5.1 User Interface

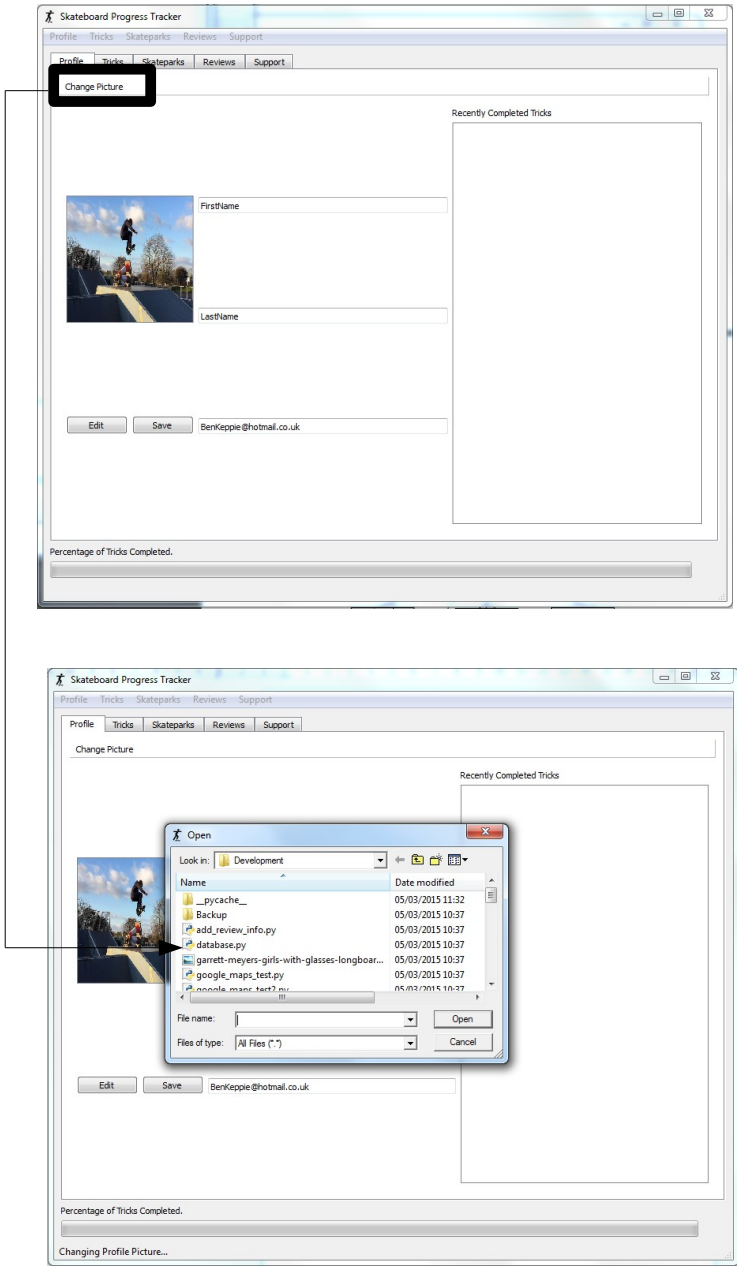


Figure 4.1: User interface, changing profile picture

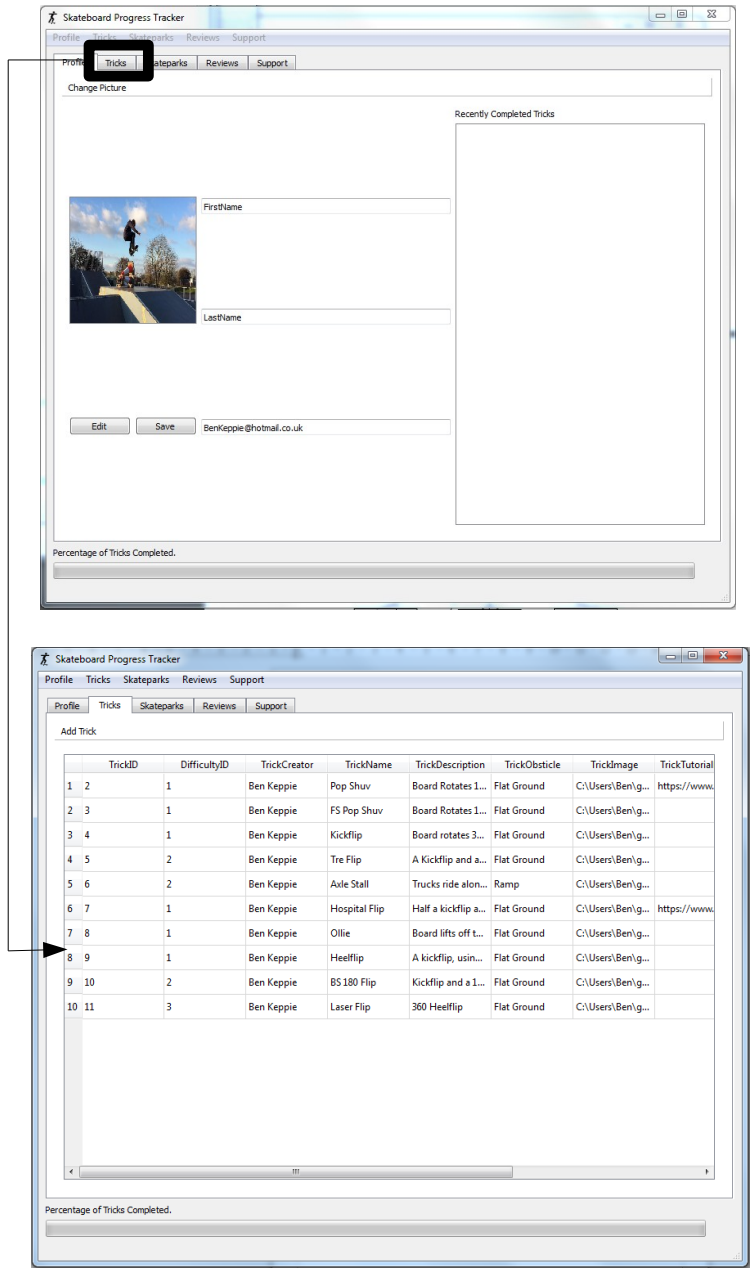


Figure 4.2: User interface, switching to the tricks tab

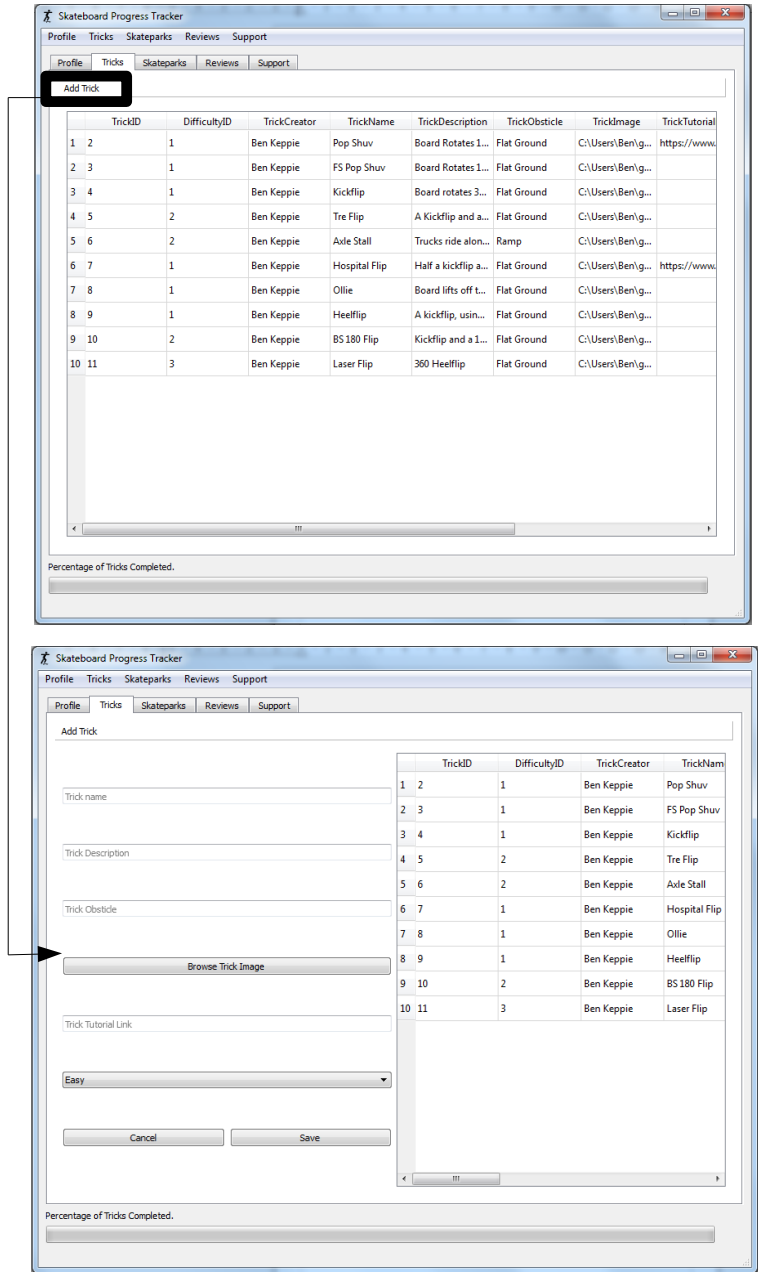


Figure 4.3: User interface, adding a trick

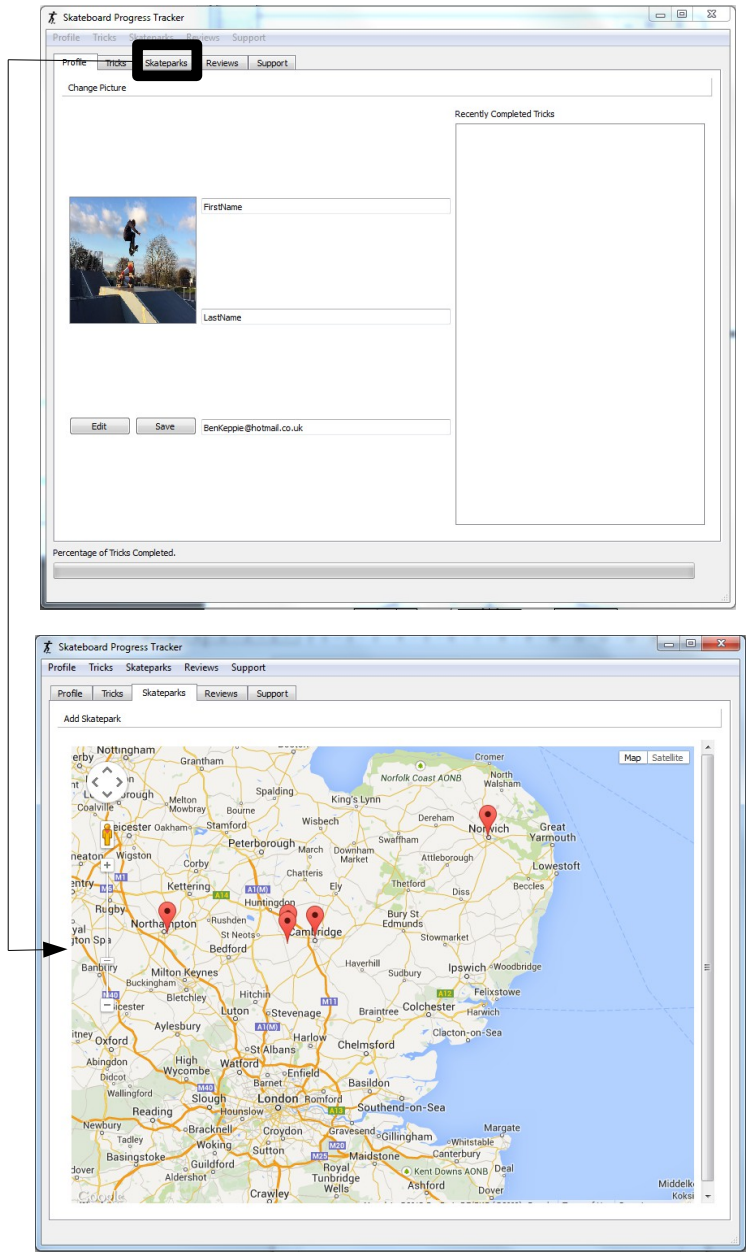


Figure 4.4: User interface, switching to the skateparks tab

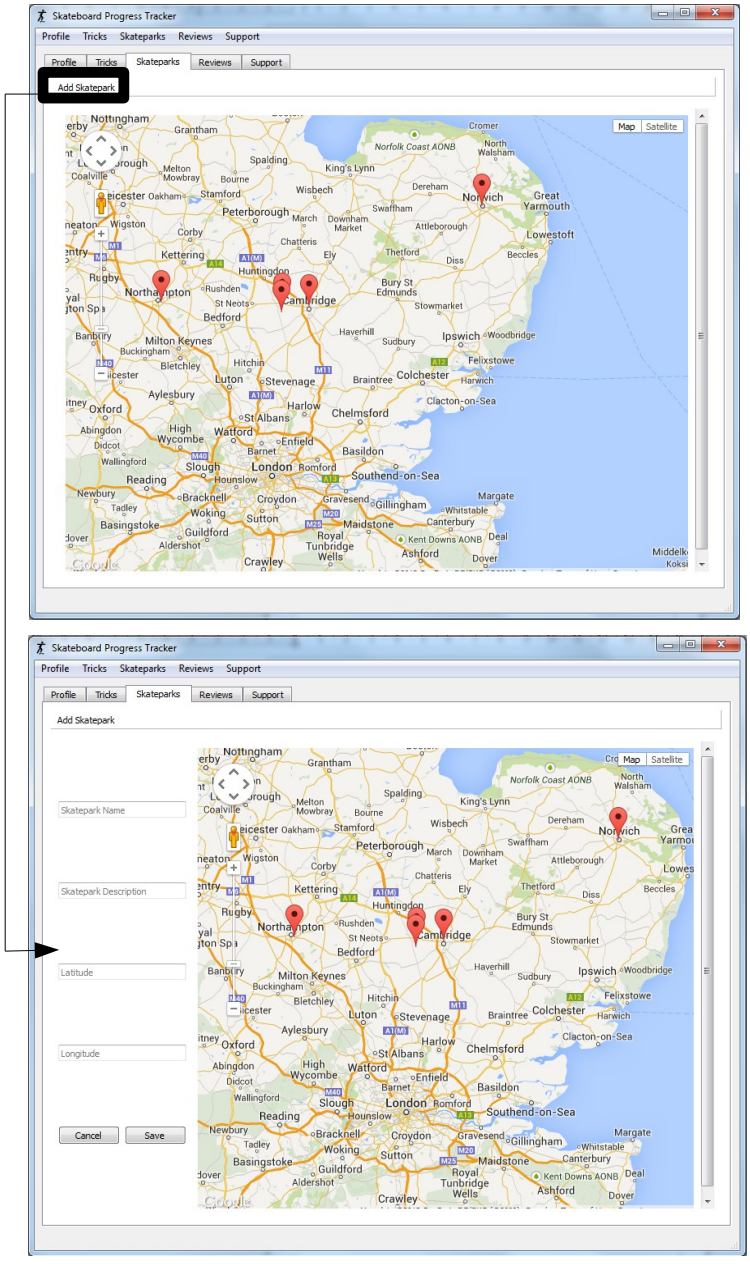


Figure 4.5: User interface, adding a skatepark

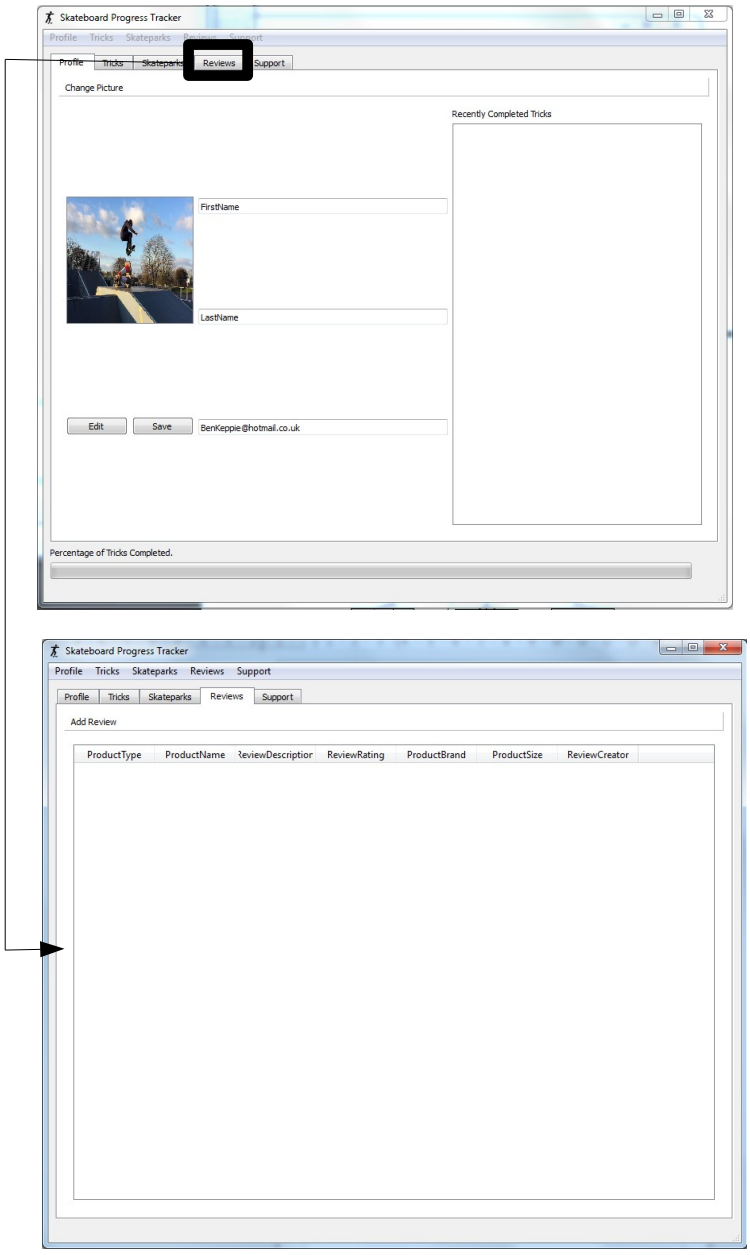


Figure 4.6: User interface, switching to the reviews tab

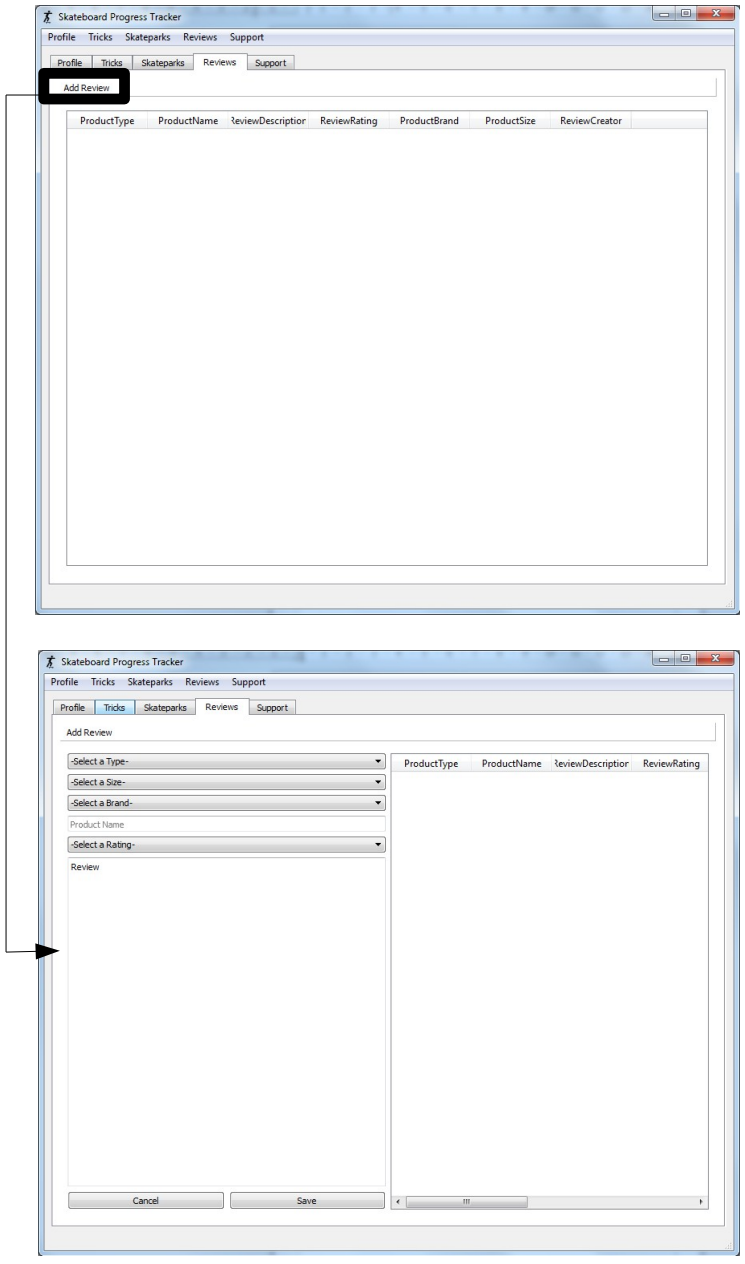


Figure 4.7: User interface, adding a review

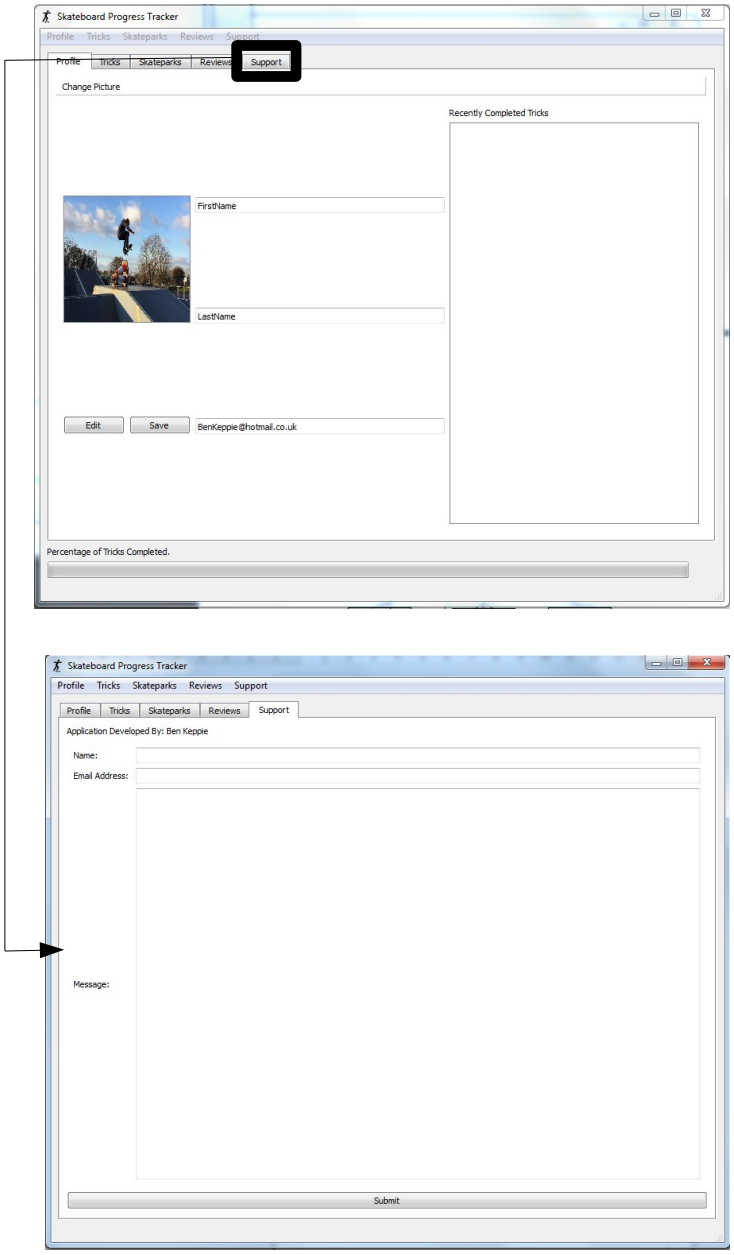


Figure 4.8: User interface, switching to the support tab

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

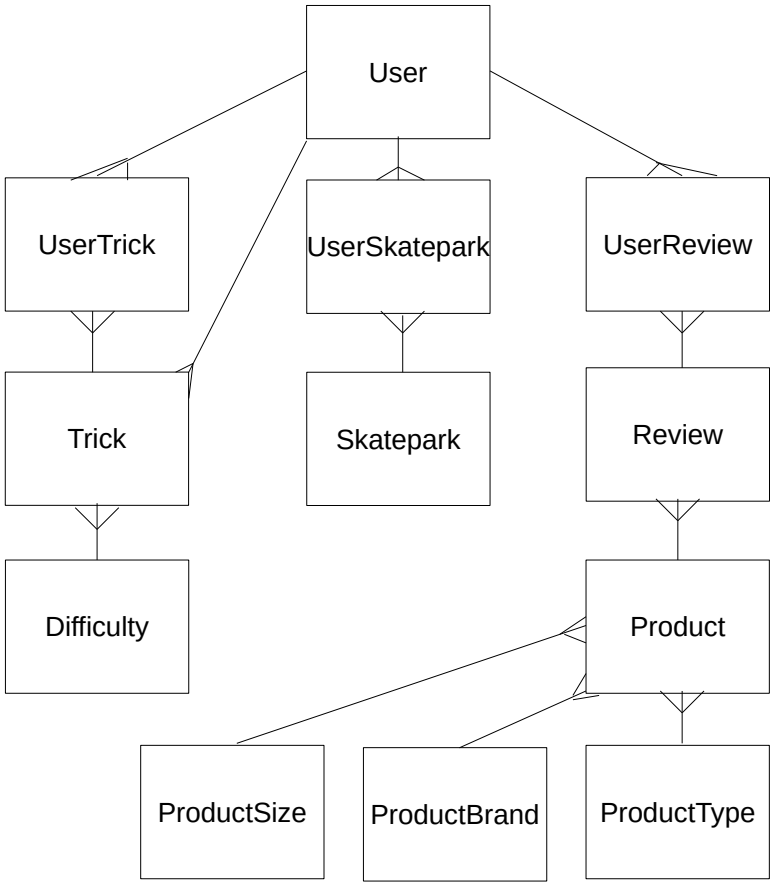


Figure 4.9: Entity-Relationship Diagram

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.

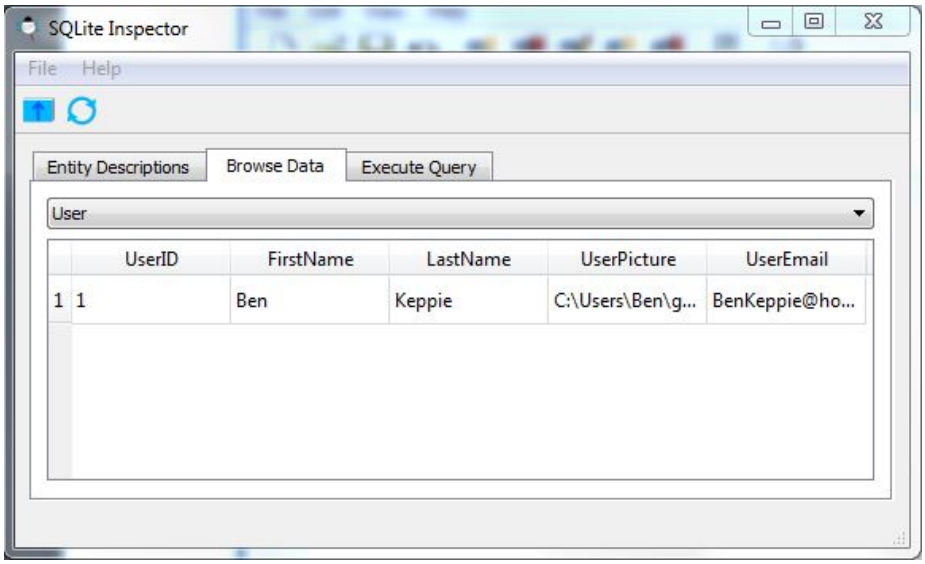


Figure 4.10: The user table of the database

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

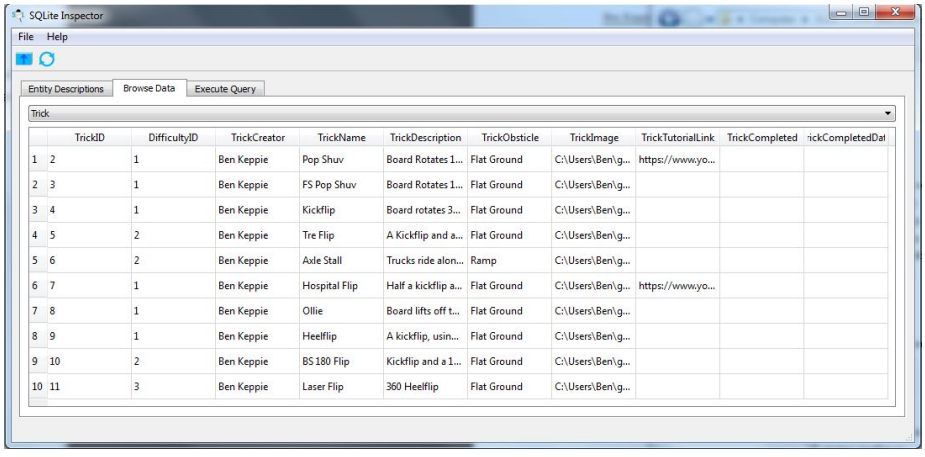


Figure 4.11: The trick table of the database

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

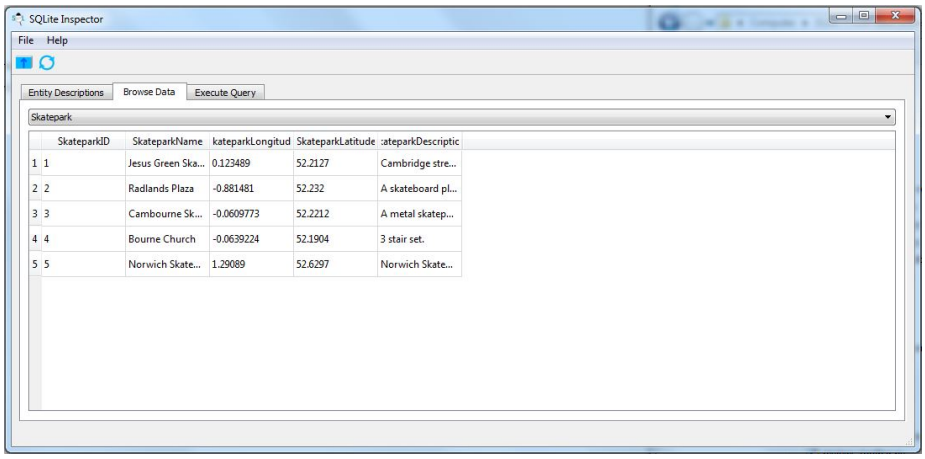


Figure 4.12: The skatepark table of the database

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

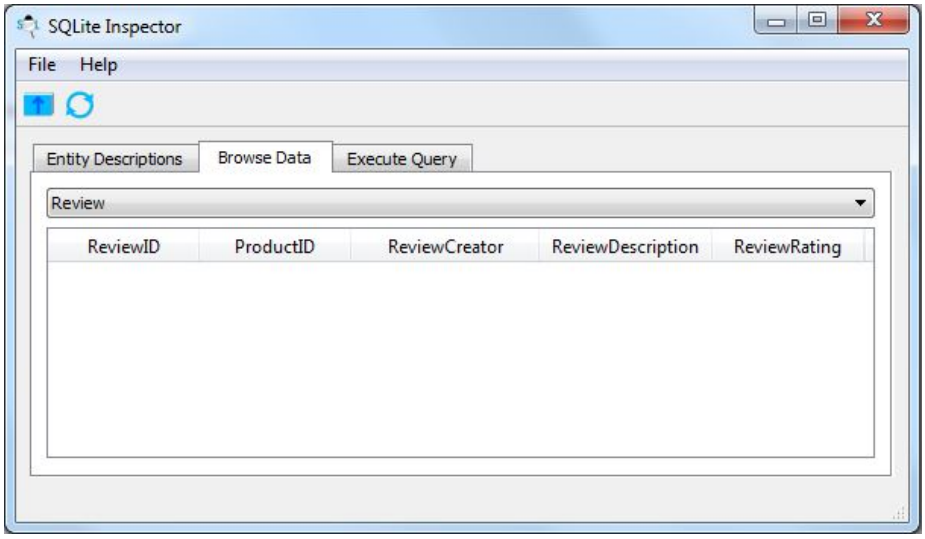


Figure 4.13: The review table of the database

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

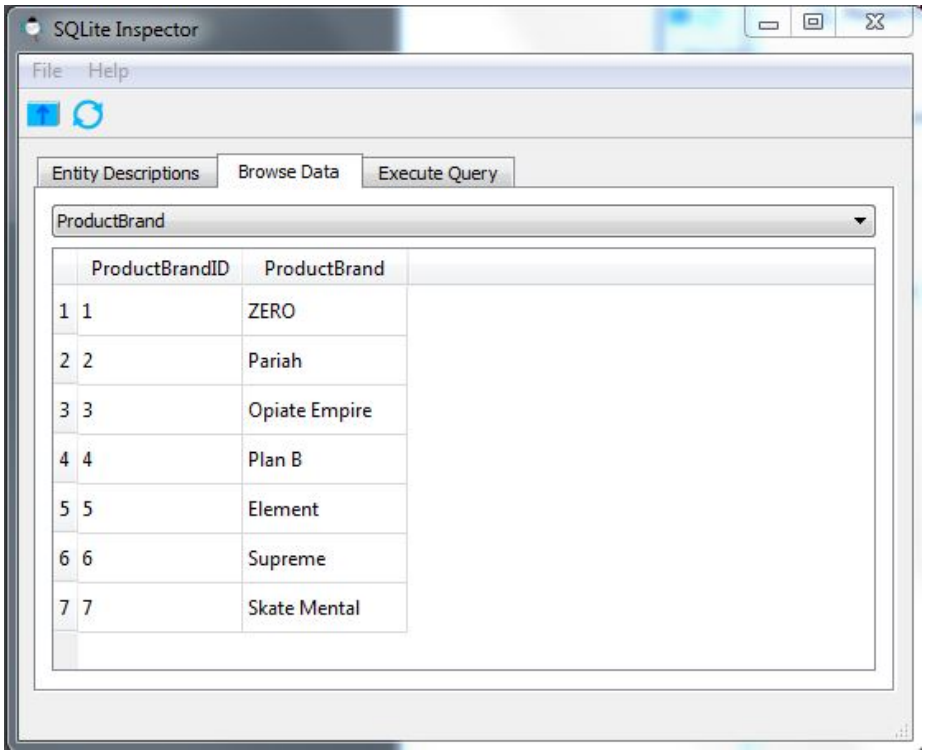


Figure 4.14: The product brand table of the database

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

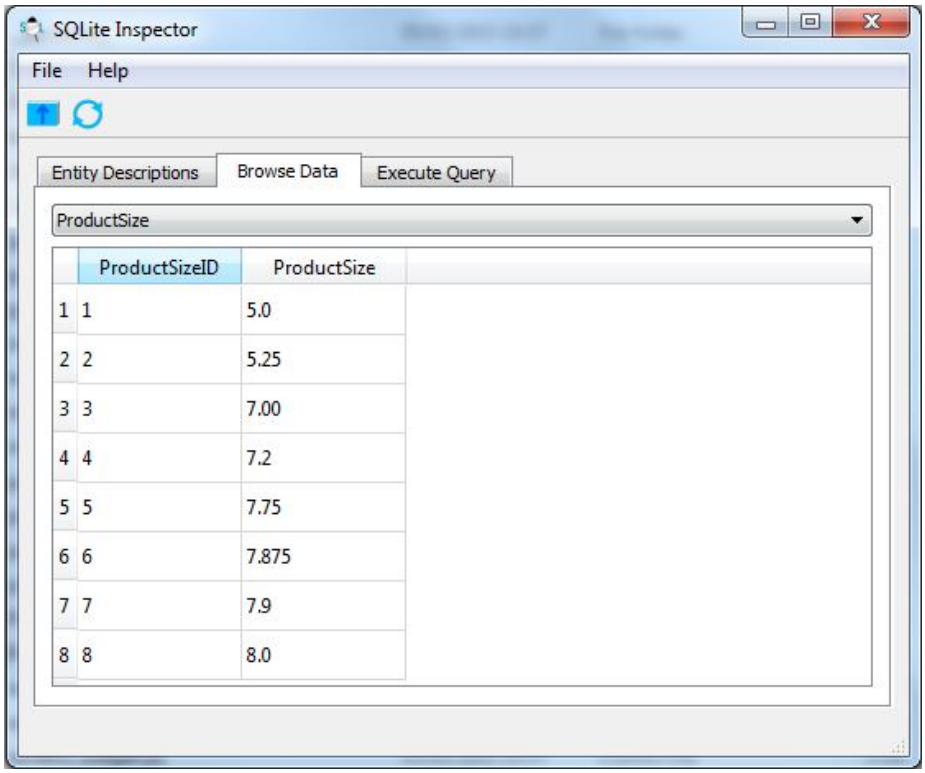


Figure 4.15: The product size table of the database

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

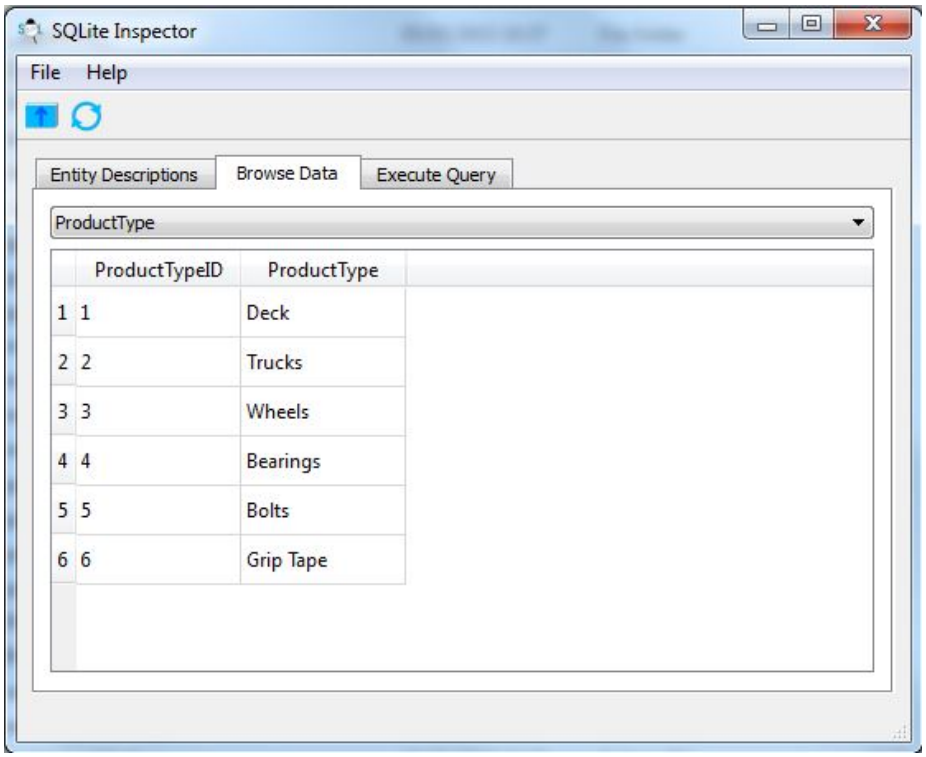


Figure 4.16: The product type table of the database

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.

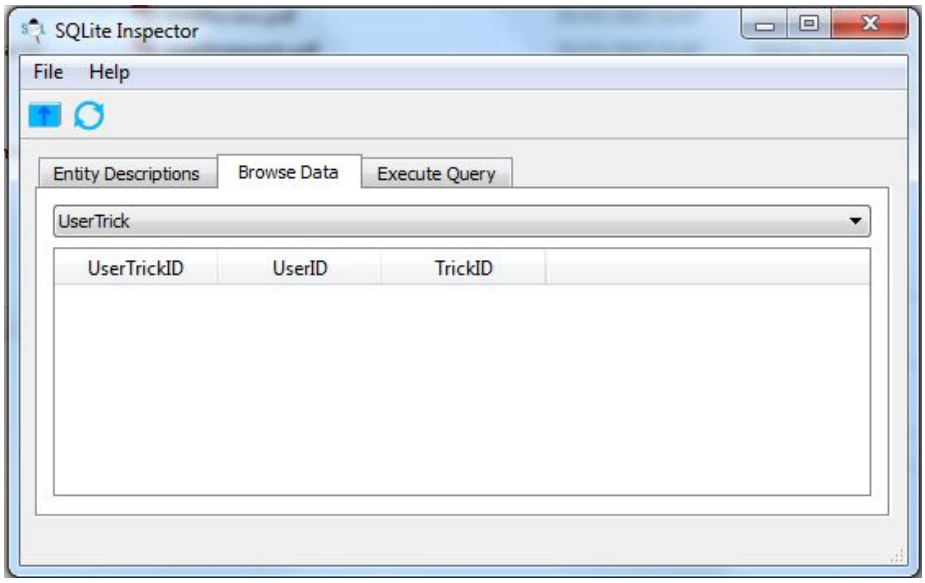


Figure 4.17: The user trick table of the database

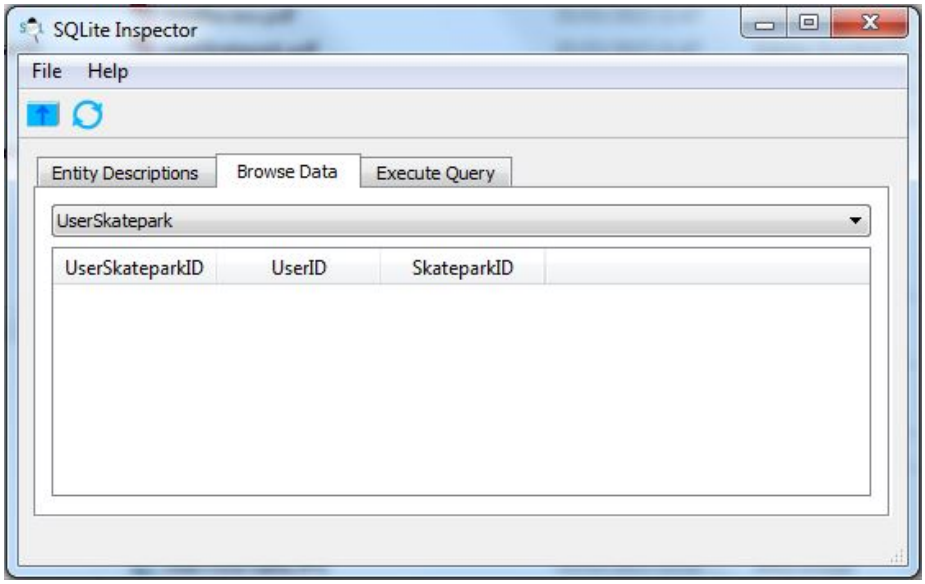


Figure 4.18: The user skatepark table of the database

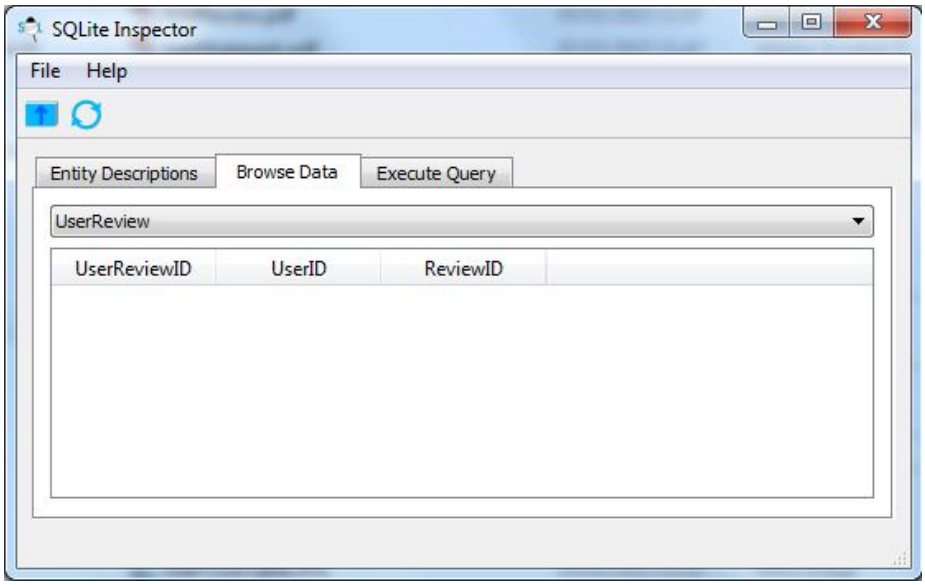


Figure 4.19: The user review table of the database

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

```
14
15 def create_trick_table():
16     db_name="skateboard_progress_tracker.db"
17     sql= """create table Trick (TrickID integer,
           DifficultyID integer, TrickCreator text,
           TrickName text, TrickDescription text,
           TrickObsticle text, TrickImage image,
           TrickTutorialLink text, TrickCompleted boolean,
18     TrickCompletedDate text, Primary Key(TrickID),
           Foreign Key(DifficultyID) references
           Difficulty(DifficultyID))"""
19     create_table(db_name,"Trick", sql)
20     print("Blank Trick Table Created.")
21
22 def create_difficulty_table():
23     db_name="skateboard_progress_tracker.db"
24     sql= """create table Difficulty (DifficultyID
           integer, TrickDifficulty text,
           DifficultyDescription text, Primary
           Key(DifficultyID))"""
25     create_table(db_name,"Difficulty", sql)
26     print("Blank Difficulty Table Created.")
27
28
29 def create_user_review_table():
30     db_name="skateboard_progress_tracker.db"
31     sql= """create table UserReview (UserReviewID
           integer, UserID integer, ReviewID integer,
           Primary Key(UserReviewID), Foreign Key(UserID)
           references User(UserID), Foreign Key(ReviewID)
           references Review(ReviewID))"""
32     create_table(db_name,"UserReview", sql)
33     print("Blank UserReview Table Created.")
34
35 def create_review_table():
36     db_name="skateboard_progress_tracker.db"
37     sql= """create table Review (ReviewID integer,
           ProductID integer, ReviewCreator text,
           ReviewDescription text, ReviewRating integer ,
           Primary Key(ReviewID), Foreign Key(ProductID)
           references Product(ProductID))"""
38     create_table(db_name,"Review", sql)
39     print("Blank Review Table Created.")
40
41 def create_product_table():
42     db_name="skateboard_progress_tracker.db"
```

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

```
        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
            integer, SkateparkName text,
            SkateparkLongitude integer, SkateparkLatitude
            integer, SkateparkDescription text,  Primary
            Key(SkateparkID))"""
76      create_table(db_name,"Skatepark", sql)
77      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
                    sqlite3.connect("skateboard_progress_tracker.db")
                    as db:
9                   cursor = db.cursor()
10                  sql="update User set FirstName=?,
                        LastName=? where UserID=?"
11                  cursor.execute(sql,values)
12                  db.commit()
```

4.6 Testing

Change Email SQL Query

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

```
        if (Email==""):
            print("Email Not Changed")
        else:
            values=(Email,1)
            with
                sqlite3.connect("skateboard_progress_tracker.db")
                as db:
                cursor = db.cursor()
                sql="update User set UserEmail=?
                    where UserID=?"
                cursor.execute(sql,values)
                db.commit()
```

Change Picture SQL Query

```
    def change_picture(self,FilePath):
        FilePath=FilePath

        values=(FilePath,1)
        with
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
            cursor = db.cursor()
            sql="update User set UserPicture=? where
                UserID=?"
            cursor.execute(sql,values)
            db.commit()
            print("Picture Changed")
```

Select First Name SQL Query

```
    def get_first_name(self):
        with
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
                cursor=db.cursor()
                cursor.execute("select FirstName from
                    User where UserID=?",(1,))
                FirstName=cursor.fetchone()
                print(FirstName)
                return FirstName
```

Select Last Name SQL Query

```
    def get_last_name(self):
        with
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
                cursor=db.cursor()
                cursor.execute("select LastName from
                    User where UserID=?",(1,))
                LastName=cursor.fetchone()
                return LastName
```

Delete Trick Row SQL Query

```
    def delete_row(self,Row):
        values=(Row+1,)
        with
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
             cursor = db.cursor()
             sql="DELETE FROM Trick WHERE TrickID=?"
             cursor.execute(sql,values)
             db.commit()
```

Select All Tricks From the Trick Table SQL Query

```
    def show_all_tricks(self):
        query = QSqlQuery()
        query.prepare(""" SELECT * FROM Trick""")
        query.exec_()
        return query
```

Add Trick to Database SQL Query

```
    def
        add_trick_to_database(self,DifficultyID,TrickName,TrickDescription,Tr
        print("hi2")

        values=(DifficultyID, "Ben Keppie",
            TrickName, TrickDescription,
            TrickObsticle, TrickImage,
            TrickTutorialLink)
```

```
        with
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
             cursor = db.cursor()
             sql="insert into Trick(DifficultyID,
                TrickCreator,TrickName,TrickDescription,TrickObsticle,TrickIm
                values (?,?,?,?,?,?,?)"
             cursor.execute(sql,values)
             db.commit()
             print()
             print("Trick Successfully Created.")
             print()
```

Add Skatepark to Database SQL Query

```
    def
        add_skatepark(self,SkateparkName,SkateparkDescription,Latitude,Longit
         values=(SkateparkName,SkateparkDescription,Latitude,Longitude)
         with
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
             cursor = db.cursor()
             sql="insert into
                Skatepark(SkateparkName,SkateparkDescription,
                SkateparkLatitude, SkateparkLongitude)
                values (?,?,?,?)"
             cursor.execute(sql,values)
             db.commit()
             print()
             print("Skatepark Successfully Created.")
             print()
```

Select all Reviews Fom the Reviews Table SQL Query

```
    def show_all_reviews(self):
        query = QSqlQuery()
        query.prepare(""" SELECT
            ProductType,ProductName,ReviewDescription,ReviewRating,ProductBra
            FROM review, product,
            productbrand,producttype,productsize""")
        query.exec_()
        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

```
class CustomQWebPage(QWebPage):
    """A class to act as the webpage for the google
       maps module"""
    def __init__(self):
        super().__init__()
        print("QWebPage constructor")

    def
       javaScriptConsoleMessage(self,message,lineNumber,sourceID):
        #An overridden method to display a javascript
           console message
        print()
        print(message,lineNumber,sourceID)
        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

```
        self.html='''<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="initial-scale=1.0,
       user-scalable=no">
    <meta charset="utf-8">
    <title>Simple markers</title>
    <style>
      html, body, #map-canvas {
        height: 100%;
        width: 100%
        margin: 0px;
        padding: 0px
      }
    </style>
```

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

```
52          function AddMarker(Location,Description,
                SkateparkName) {
53
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          });
62          //markers.push(marker);
63          var lat = marker.getPosition().lat();
64          var lng = marker.getPosition().lng();
65          markers.push({"Object":marker,"Lat":lat,"Lng":lng,
                "Desc":Description});
66
67          var contentString = ('<div id="content"><div
                id="siteNotice"></div> <h1
                id="firstHeading" class="firstHeading">'+
                SkateparkName + '</h1> <div
                id="bodyContent"><p>' + Description
                +'</p></div></div>');
68
69        var infowindow = new google.maps.InfoWindow({
70        content: contentString
71    });
72
73        google.maps.event.addListener(marker,
              'rightclick', function(event) {
74          marker.setMap(null);
75          });
76      google.maps.event.addListener(marker,
            'mouseover', function(event) {
77      infowindow.open(map,marker);
78    });
79    google.maps.event.addListener(marker,'mouseout',
          function(event){
80    infowindow.close(map,marker)
81    });
82
83
84          }
85
86
87  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',
        initialize);
101      </script>
102    </head>
103    <body>
104      <div id="map-canvas"></div>
105    </body>
106 </html> '''
107         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=
               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
                self.parent.skatepark_name.isReadOnly():
4               self.get_last_marker()
5           else:
6               print()
7               print("In view only mode.")
8               print()
9
10      def get_last_marker(self):
11          self.LastMarker=
                self.CustomPage.mainFrame().evaluateJavaScript("GetMarkers()")
12          print(self.LastMarker)
13          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"):
```

```
            self.delete_message=QLabel("Are you sure
                you wish to delte this trick?")
            self.dialog_VBoxLayout.addWidget(self.delete_message)
        if not hasattr(self,"delete_cancel"):
            self.delete_cancel=QPushButton("Cancel")
            self.dialog_button_layout.addWidget(self.delete_cancel)
        if not hasattr(self,"delete_save"):
            self.delete_save=QPushButton("Save")
            self.dialog_button_layout.addWidget(self.delete_save)
        self.dialog_VBoxLayout.addLayout(self.dialog_button_layout)
        self.delete_dialog.setLayout(self.dialog_VBoxLayout)
        self.delete_dialog.show()
        self.delete_cancel.clicked.connect(self.delete_cancel_clicked)
        self.delete_save.clicked.connect(self.delete_save_clicked,Row)

    def delete_cancel_clicked(self):
        self.delete_dialog.close()
        self.parent.StatusBar.showMessage("Trick Not
            Deleted",2000)

    def delete_save_clicked(self,Row):
        self.delete_dialog.close()
        self.connection.delete_row(Row)
        query = self.connection.show_all_tricks()
        self.model.setQuery(query)
        self.parent.StatusBar.showMessage("Trick
            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.

```
    def delete_row(self,Row):
        values=(Row+1,)
        with
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
            cursor = db.cursor()
            sql="DELETE FROM Trick WHERE TrickID=?"
            cursor.execute(sql,values)
            db.commit()
            print()
```

Sending Emails

```
    def send_email(self):
        msg=MIMEText(self.form_email_line_edit.text()+
            self.form_message_line_edit.toPlainText())
        print(msg)
        msg["Subject"]="Skateboard Progress Tracker
            Support"
        msg["From"]="SkateboardProgressTracker@gmail.com"
        msg["To"]= "BenKeppie@hotmail.co.uk"

        Send=smtplib.SMTP("smtp.gmail.com")
        Send.sendmail(msg["From"],msg["To"],msg)
        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-pyq` 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
12 from menu_bar import *
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
               Tracker")
34         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
42          #create tabs
43          self.create_tabs()
44
45          #Create Menu Bar
46          self.Menu=Menu(self)
47          self.setMenuBar(self.Menu.MenuBar)
48
49          #Create Statusbar
50          self.StatusBar=QStatusBar()
51          self.setStatusBar(self.StatusBar)
52
53          self.tabs.currentChanged.connect(self.progress_bar_hide)
54
55
56      def progress_bar_hide(self):
57          if (self.tabs.currentIndex() >=2):
58              self.ProgressBarLabel.hide()
59              self.ProgressBar.hide()
60          else:
61              self.ProgressBar.show()
62              self.ProgressBarLabel.show()
63
64
65
66
67
68      def set_icon(self):
69          self.app_icon=QtGui.QIcon()
70          self.app_icon.addFile("ProgramIcon.png",QSize(16,16))
71          self.app.setWindowIcon(self.app_icon)
72
73
74
75
76
77      def create_tabs(self):
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,
                "Skateparks")
95          self.tabs.addTab(self.reviews_tab, "Reviews")
96          self.tabs.addTab(self.support_tab, "Support")
97
98          self.main_VBoxLayout.addWidget(self.tabs)
99          self.main_VBoxLayout.addWidget(self.ProgressBarLabel)
100         self.main_VBoxLayout.addWidget(self.ProgressBar)
101         self.central_widget.setLayout(self.main_VBoxLayout)
102
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,
            Qt.WindowStaysOnTopHint)
117     splash.setMask(splash_pix.mask())
118     splash.show()
119     time.sleep(2)
120     splash.finish(splash)
121
122 def main():
123     application=QApplication(sys.argv)
```

```
124     window=MainWindow()
125     #splash_screen()
126 
127 
128     window.show()
129     window.raise_()
130     application.exec_()
131     print()
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"""
```

```
    def __init__(self,parent):
        super().__init__()
        self.parent=parent
        self.MenuBar=QMenuBar()
        #create actions

        self.change_picture=QAction("Change
            Picture",self)

        self.add_trick=QAction("Add Trick", self)

        self.add_skatepark=QAction("Add
            Skatepark",self)

        self.add_review=QAction("Add Review",self)

        self.contact_support=QAction("Contact
            Support",self)



        #create options
        self.profile_menu=self.MenuBar.addMenu("Profile")
        self.tricks_menu=self.MenuBar.addMenu("Tricks")
        self.skateparks_menu=self.MenuBar.addMenu("Skateparks")
        self.reviews_menu=self.MenuBar.addMenu("Reviews")
        self.support_menu=self.MenuBar.addMenu("Support")


        #add actions to menu

        self.profile_menu.addAction(self.change_picture)

        self.tricks_menu.addAction(self.add_trick)

        self.skateparks_menu.addAction(self.add_skatepark)

        self.reviews_menu.addAction(self.add_review)

        self.support_menu.addAction(self.contact_support)

        #connections

        self.change_picture.triggered.connect(self.change_picture_connection
        self.add_trick.triggered.connect(self.add_trick_connection)
        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
               Profile Picture...")
70          self.parent.tabs.setCurrentIndex(0)
71          print("Find Picture")
72          path=QFileDialog.getOpenFileName()
73
74          if path=="":
75              print("Picture not changed.")
76              self.parent.StatusBar.showMessage("Profile
                   Picture Not Changed.",2000)
77          else:
78              replace="\."
79              path=path.replace("/",replace[0])
80              destination=("{0}{1}{2}".format(os.getcwd(),replace[0],"ProfileP
81              print(destination)
82              print(path)
83              shutil.copy2(path,destination)
84              self.connection=ProfileSQLConnections()
85              self.connection.change_picture(destination)
86              print(path)
87              self.parent.StatusBar.showMessage("Profile
                   Picture Successfully Changed.",2000)
88
89      def add_trick_connection(self):
90          self.parent.tabs.setCurrentIndex(1)
91          self.parent.tricks_tab.add_trick_stacked()
92          print("add Trick")
93
94      def add_skatepark_connection(self):
95          self.parent.tabs.setCurrentIndex(2)
96          self.parent.skateparks_tab.view_add_skatepark()
```

4.11.4 Profile Widget

```
import sys
from PyQt4.QtGui import *
from PyQt4 import QtGui
from PyQt4.QtCore import *
from PyQt4.QtSql import *

from main_window import *
from profile_toolbar import *
from profile_picture import *
from profile_sql_connections import *


class DisplayProfileWidget(QWidget):
    """A class to display the model and represent a
       view on the profile tab"""

    def __init__(self,parent):
        super().__init__()
        self.parent=parent


        self.ProfileSQLConnections=ProfileSQLConnections()
        self.ToolBarWidgetLayout=QVBoxLayout()
        self.LayoutWidget=QWidget()

        self.HBoxLayout=QHBoxLayout()
        self.LeftVBoxLayout=QVBoxLayout()
        self.LeftHBoxLayout=QHBoxLayout()
        self.MiddleVBoxLayout=QVBoxLayout()
        self.RightVBoxLayout=QVBoxLayout()


        self.display_profile_layout()
        self.display_profile_toolbar_widget()
        self.setLayout(self.ToolBarWidgetLayout)
        self.model=None

        self.edit_button.clicked.connect(self.edit_button_clicked)
        self.save_button.clicked.connect(self.save_button_clicked)

```

```
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
                    Completed Tricks")
88              self.RightVBoxLayout.addWidget(self.recent_tricks)
89          if not hasattr(self,"recent_tricks_list"):
90              self.recent_tricks_list=QListWidget()
91              #self.recent_tricks_list.addItem("Ollie")
92              self.RightVBoxLayout.addWidget(self.recent_tricks_list)
93
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_pictur
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
                Picture Successfully Changed.", 2000)
110
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 provies a grpahics view for the
           profile picture"""
12     def __init__(self):
13         super().__init__()
14         self.PictureSQLConnection=ProfileSQLConnections()
15         self.scene=QGraphicsScene()
16
17         self.picture()
18
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

```
import sys
from PyQt4.QtGui import *
from PyQt4 import QtGui
from PyQt4.QtCore import *
from PyQt4.QtSql import *

from profile_widget import *
from profile_sql_connections import *
from main_window import *


class DisplayProfileToolbar(QToolBar):
    """A class to create the profile tabs' toolbar"""
    changedPicture=pyqtSignal()
    def __init__(self,parent):
        super().__init__()
        self.parent=parent

        self.change_picture=QAction("Change
           Picture",self)

        self.addAction(self.change_picture)


        self.change_picture.triggered.connect(self.change_picture_connection

    def change_picture_connection(self):
        self.parent.parent.StatusBar.showMessage("Changing
           Profile Picture...")

        print("Find Picture")
        path=QFileDialog.getOpenFileName()
        if path=="":
            print("Picture not changed.")
            self.parent.parent.StatusBar.showMessage("Profile
               Picture Not Changed.", 2000)
        else:
            replace="\."
            path=path.replace("/",replace[0])
            destination=("{0}{1}{2}".format(os.getcwd(),replace[0],"ProfileP
            print(destination)
            print(path)
            shutil.copy2(path,destination)
            self.connection=ProfileSQLConnections()
            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
           the profile tab"""
12
13     def __init__(self):
14         print("Profile SQL Connection")
15
16     def change_name(self,FirstName,LastName):
17         FirstName=FirstName
18         LastName=LastName
19         if (FirstName=="") or (LastName==""):
20             print("Name Not Changed")
21         else:
22             values=(FirstName,LastName, 1)
23             with
                   sqlite3.connect("skateboard_progress_tracker.db")
                   as db:
24                  cursor = db.cursor()
25                  sql="update User set FirstName=?,
                        LastName=? where UserID=?"
26                  cursor.execute(sql,values)
27                  db.commit()
28
29
30
31     def change_email(self,Email):
32         Email=Email
33         if (Email==""):
34             print("Email Not Changed")
35         else:
```

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

```
70                  LastName=cursor.fetchone()
71                  return LastName
72      def get_email(self):
73          with
                sqlite3.connect("skateboard_progress_tracker.db")
                as db:
74                  cursor=db.cursor()
75                  cursor.execute("select UserEmail from
                        User where UserID=?",(1,))
76                  UserEmail=cursor.fetchone()
77                  return UserEmail
78      def get_picture(self):
79          FilePath=("{0}{1}".format(os.getcwd(),"\ProfilePicture.jpeg"))
80          print(FilePath)
81          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          self.ToolBarWidgetLayout=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_descr
56          self.trick_obsticle.textChanged.connect(self.validate_trick_obsticle
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
                    you wish to delte this trick?")
74              self.dialog_VBoxLayout.addWidget(self.delete_message)
75          if not hasattr(self,"delete_cancel"):
76              self.delete_cancel=QPushButton("Cancel")
77              self.dialog_button_layout.addWidget(self.delete_cancel)
78          if not hasattr(self,"delete_save"):
79              self.delete_save=QPushButton("Save")
80              self.dialog_button_layout.addWidget(self.delete_save)
81          self.dialog_VBoxLayout.addLayout(self.dialog_button_layout)
82          self.delete_dialog.setLayout(self.dialog_VBoxLayout)
83          self.delete_dialog.show()
84          self.delete_cancel.clicked.connect(self.delete_cancel_clicked)
85          self.delete_save.clicked.connect(self.delete_save_clicked,Row)
86
87      def delete_cancel_clicked(self):
88          self.delete_dialog.close()
89          self.parent.StatusBar.showMessage("Trick Not
                Deleted",2000)
90
91      def delete_save_clicked(self,Row):
92          self.delete_dialog.close()
93          self.connection.delete_row(Row)
94          query = self.connection.show_all_tricks()
95          self.model.setQuery(query)
96          self.parent.StatusBar.showMessage("Trick
                Successfully Deleted",2000)
97
98
99
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         TrickObsticle=self.validate_trick_obsticle()
107         if self.trick_tutorial.text()=="":
108             TrickTutorial=True
109         else:
110             TrickTutorial=self.validate_trick_tutorial()
111
112
113         if (TrickName==True) and
                (TrickDescription==True) and
                (TrickObsticle==True) and
```

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

```
155             return True
156         else:
157             self.trick_obsticle.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 
173     def clear_trick_line_edit(self):
174         self.trick_name.clear()
175         self.trick_description.clear()
176         self.trick_obsticle.clear()
177         self.trick_tutorial.clear()
178         self.trick_difficulty.setCurrentIndex(0)
179 
180 
181     def display_results(self):
182         self.results_layout.addWidget(self.results_table)
183         self.results_widget.setLayout(self.results_layout)
184 
185         self.stacked_layout.addWidget(self.results_widget)
186 
187 
188     def open_connection(self):
189         self.path=("{0}{1}".format(os.getcwd(),"\skateboard_progress_tracker
190         print(self.path)
191         self.connection=TricksSQLConnections(self.path)
192         self.connection.open_database()
193 
194     def show_tricks_layout(self):
195 
196         if self.connection != None:
197             self.query =
                    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
                        name")
242 
243                 self.add_trick_VBoxLayout.addWidget(self.trick_name)
244             if not hasattr(self,"trick_description"):
```

```
245             self.trick_description=QLineEdit()
246             self.trick_description.setPlaceholderText("Trick
                    Description")
247             self.add_trick_VBoxLayout.addWidget(self.trick_description)
248         if not hasattr(self,"trick_obsticle"):
249             self.trick_obsticle=QLineEdit()
250             self.trick_obsticle.setPlaceholderText("Trick
                    Obsticle")
251             self.add_trick_VBoxLayout.addWidget(self.trick_obsticle)
252         if not hasattr(self,"trick_image"):
253             self.trick_image=QPushButton("Browse
                    Trick Image")
254             self.add_trick_VBoxLayout.addWidget(self.trick_image)
255         if not hasattr(self,"trick_tutorial"):
256             self.trick_tutorial=QLineEdit()
257             self.trick_tutorial.setPlaceholderText("Trick
                    Tutorial Link")
258             self.add_trick_VBoxLayout.addWidget(self.trick_tutorial)
259         if not hasattr(self,"trick_difficulty"):
260             self.trick_difficulty=QComboBox()
261             self.trick_difficulty.addItem("Easy")
262             self.trick_difficulty.addItem("Medium")
263             self.trick_difficulty.addItem("Hard")
264             self.add_trick_VBoxLayout.addWidget(self.trick_difficulty)
265 
266         if not hasattr(self,"cancel_trick"):
267             self.cancel_trick=QPushButton("Cancel")
268             self.add_trick_button_layout.addWidget(self.cancel_trick)
269         if not hasattr(self,"save_trick"):
270             self.save_trick=QPushButton("Save")
271             self.add_trick_button_layout.addWidget(self.save_trick)
272         self.add_trick_VBoxLayout.addLayout(self.add_trick_button_layout)
273         self.add_trick_HBoxLayout.addLayout(self.add_trick_VBoxLayout)
274         self.add_trick_HBoxLayout.addWidget(self.add_trick_table)
275         self.add_trick_widget.setLayout(self.add_trick_HBoxLayout)
276 
277     def cancel_button_clicked(self):
278         print("Cancel")
279         self.table_stacked()
280         self.clear_trick_line_edit()
281 
282     def save_button_clicked(self):
283         Valid=self.validate_add_trick()
284         if Valid:
285             self.clear_trick_line_edit()
286         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
302             replace="\."
303             path=path.replace("/",replace[0])
304             print(path)
305             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

```
import sqlite3
from PyQt4.QtSql import *

class TricksSQLConnections:
    """Handles the connection to the SQL database for
       the tricks tab"""

    def __init__(self,path):
        self.path=path
        self.db=None

    def delete_row(self,Row):
        values=(Row+1,)
        with
           sqlite3.connect("skateboard_progress_tracker.db")
           as db:
            cursor = db.cursor()
            sql="DELETE FROM Trick WHERE TrickID=?"
            cursor.execute(sql,values)
            db.commit()
            print()


    def open_database(self):
        if self.db:
            self.close_database()

        self.db = QSqlDatabase.addDatabase("QSQLITE")
        self.db.setDatabaseName(self.path)

        opened_ok=self.db.open()
        return opened_ok

    def show_all_tricks(self):
        query = QSqlQuery()
        query.prepare(""" SELECT * FROM Trick""")
        query.exec_()
        return query

    def
       add_trick_to_database(self,DifficultyID,TrickName,TrickDescription,Tr
        print("hi2")

```

```
40          values=(DifficultyID, "Ben Keppie",
                TrickName, TrickDescription,
                TrickObsticle, TrickImage,
                TrickTutorialLink)
41          with
                sqlite3.connect("skateboard_progress_tracker.db")
                as db:
42              cursor = db.cursor()
43              sql="insert into Trick(DifficultyID,
                    TrickCreator,TrickName,TrickDescription,TrickObsticle,TrickIm
                    values (?,?,?,?,?,?,?)"
44              cursor.execute(sql,values)
45              db.commit()
46              print()
47              print("Trick Successfully Created.")
48              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:
       AIzaSyC5RcJ7vLSEYF32KqDusnuRcLJiHW8EbDg
15
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_skatepa
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
               (SkateparkDescription==True) and
               (SkateparkLatitude==True) and
               (SkateparkLongitude==True):
98              self.parent.StatusBar.showMessage("Skatepark
                   Successfully Saved.",2000)
99              return True
100         else:
101             self.parent.StatusBar.showMessage("Not
                   all Fields are Valid.",2000)
102             return False
103
104     def cancel_button_clicked(self):
105         print("Cancel")
106         self.clear_skatepark_line_edit()
107         self.skatepark_name.hide()
108         self.skatepark_description.hide()
109         self.skatepark_longitude.hide()
110         self.skatepark_latitude.hide()
111         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 
117         def save_button_clicked(self):
118             print("Save")
119             Valid=self.validate_add_skatepark()
120             if Valid:
121                 self.SkateparksSQLConnections.add_skatepark(self.skatepark_name.
122                 self.clear_skatepark_line_edit()
123                 self.add_skatepark_map.delete_all_markers()
124                 self.add_skatepark_map.get_marker_coordinates()
125 
126             else:
127                 print()
128 
129         def view_add_skatepark(self):
130             self.skatepark_name.show()
131             self.skatepark_description.show()
132             self.skatepark_longitude.show()
133             self.skatepark_latitude.show()
134             self.cancel_skatepark.show()
135             self.save_skatepark.show()
136 
137 
138         def clear_skatepark_line_edit(self):
139             self.skatepark_name.clear()
140             self.skatepark_description.clear()
141             self.skatepark_latitude.clear()
142             self.skatepark_longitude.clear()
143 
144         def fill_line_edits(self,LastMarker):
145             self.latitude_coor=str(LastMarker[0])
146             self.longitude_coor=str(LastMarker[1])
147             self.skatepark_latitude.setText(self.latitude_coor)
148             self.skatepark_longitude.setText(self.longitude_coor)
149 
150 
151 
152         def add_skatepark(self):
153             if not hasattr(self,"skatepark_name"):
154                 self.skatepark_name=QLineEdit()
155                 self.skatepark_name.setPlaceholderText("Skatepark
                        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
                    Description")
161             self.skatepark_description.hide()
162             self.add_skatepark_VBoxLayout.addWidget(self.skatepark_descripti
163 
164         if not hasattr(self,"skatepark_latitude"):
165             self.skatepark_latitude=QLineEdit()
166             self.skatepark_latitude.setPlaceholderText("Latitude")
167             self.skatepark_latitude.setReadOnly(True)
168             self.skatepark_latitude.hide()
169             self.add_skatepark_VBoxLayout.addWidget(self.skatepark_latitude)
170         if not hasattr(self,"skatepark_longitude"):
171             self.skatepark_longitude=QLineEdit()
172             self.skatepark_longitude.setPlaceholderText("Longitude")
173             self.skatepark_longitude.setReadOnly(True)
174             self.skatepark_longitude.hide()
175             self.add_skatepark_VBoxLayout.addWidget(self.skatepark_longitude
176 
177         if not hasattr(self,"cancel_skatepark"):
178             self.cancel_skatepark=QPushButton("Cancel")
179             self.cancel_skatepark.hide()
180             self.add_skatepark_button_layout.addWidget(self.cancel_skatepark
181         if not hasattr(self,"save_skatepark"):
182             self.save_skatepark=QPushButton("Save")
183             self.save_skatepark.hide()
184             self.add_skatepark_button_layout.addWidget(self.save_skatepark)
185 
186         if not hasattr(self,"add_skatepark_map"):
187             self.add_skatepark_map=ViewOnlyMap(self)
188 
189 
190 
191 
192         self.add_skatepark_VBoxLayout.addLayout(self.add_skatepark_button_la
193         self.add_skatepark_HBoxLayout.addLayout(self.add_skatepark_VBoxLayou
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         self.ToolBarWidgetLayout.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 =
       "AIzaSyC5RcJ7vLSEYF32KqDusnuRcLJiHW8EbDg"
7
8  class CustomQWebPage(QWebPage):
9      """A class to act as the webpage for the google
           maps module"""
10     def __init__(self):
11         super().__init__()
12         print("QWebPage constructor")
13
14     def
           javaScriptConsoleMessage(self,message,lineNumber,sourceID):
15         #An overridden method to display a javascript
               console message
16         print()
17         print(message,lineNumber,sourceID)
18         print("javascript console message^")
19
20
21 class ViewOnlyMap(QWebView):
22     """A class to create the google maps window"""
23
24
25     def __init__(self,parent):
26         super().__init__()
27         self.parent=parent
28
29         #Changing web settings to access certain
               functionality
```

```
30          self.settings().setAttribute(QWebSettings.JavascriptEnabled,
                True)
31          self.settings().setAttribute(QWebSettings.JavascriptCanOpenWindows,
                True)
32          self.settings().setAttribute(QWebSettings.JavascriptCanAccessClipboar
                True)
33          self.settings().setAttribute(QWebSettings.DeveloperExtrasEnabled,
                True)
34
35
36          self.CustomPage=CustomQWebPage()
37          self.Coordinates=None
38          self.setPage(self.CustomPage)
39          self.loadFinished.connect(self.handle_load_finished)
40
41
42          self.set_code()
43          print("Code set")
44
45      def mousePressEvent(self,event):
46          super().mousePressEvent(event)
47          if not
                self.parent.skatepark_name.isReadOnly():
48              self.get_last_marker()
49          else:
50              print()
51              print("In view only mode.")
52              print()
53
54      def get_last_marker(self):
55          self.LastMarker=
                self.CustomPage.mainFrame().evaluateJavaScript("GetMarkers()")
56          print(self.LastMarker)
57          self.parent.fill_line_edits(self.LastMarker)
58
59      def delete_all_markers(self):
60          self.CustomPage.mainFrame().evaluateJavaScript("DeleteMarkers()")
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
                sqlite3.connect("skateboard_progress_tracker.db")
                as db:
77              cursor=db.cursor()
78              sql="select SkateparkLatitude,
                    SkateparkLongitude,
                    SkateparkDescription, SkateparkName
                    from Skatepark"
79              cursor.execute(sql)
80              self.Coordinates=cursor.fetchall()
81          for coordinate in self.Coordinates:
82              Name=str(coordinate[3])
83              Name="'{0}'".format(Name)
84
85              Description=str(coordinate[2])
86              Description="'{0}'".format(Description)
87
88              self.CustomPage.mainFrame().evaluateJavaScript("MarkersFromDatab
                    {1}, {2}, {3})".format(coordinate[0],
                    coordinate[1], Description, 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,
          user-scalable=no">
104     <meta charset="utf-8">
105     <title>Simple markers</title>
106     <style>
107       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
115     <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
142          function
                MarkersFromDatabase(SkateparkLat,SkateparkLng,SkateparkDescripti
                SkateparkName) {
143
144         var Skatepark = new
               google.maps.LatLng(SkateparkLat,SkateparkLng);
145
```

```
146         AddMarker(Skatepark,SkateparkDescription,
                SkateparkName); }
147
148
149
150
151         function AddMarker(Location,Description,
                SkateparkName) {
152
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,
                "Desc":Description});
165
166         var contentString = ('<div id="content"><div
                id="siteNotice"></div> <h1
                id="firstHeading" class="firstHeading">'+
                SkateparkName + '</h1> <div
                id="bodyContent"><p>' + Description
                +'</p></div></div>');
167
168       var infowindow = new google.maps.InfoWindow({
169       content: contentString
170   });
171
172       google.maps.event.addListener(marker,
              'rightclick', function(event) {
173         marker.setMap(null);
174         });
175     google.maps.event.addListener(marker,
          'mouseover', function(event) {
176     infowindow.open(map,marker);
177   });
178   google.maps.event.addListener(marker,'mouseout',
        function(event){
179   infowindow.close(map,marker)
180   });
```

```
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',
        initialize);
200     </script>
201   </head>
202   <body>
203     <div id="map-canvas"></div>
204   </body>
205 </html> '''
206         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
           toolbar"""
10     def __init__(self,parent):
11         super().__init__()
12         self.parent=parent
13
```

```
14          self.add_skatepark=QAction("Add
                Skatepark",self)
15
16          self.addAction(self.add_skatepark)
17
18          self.add_skatepark.triggered.connect(self.add_skatepark_connection)
19
20      def add_skatepark_connection(self):
21          print("Add Skatepark")
22          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
           the skateparks tab"""
5
6      def __init__(self):
7          print("Skatepark SQL Connection")
8
9      def
           add_skatepark(self,SkateparkName,SkateparkDescription,Latitude,Longit
10         values=(SkateparkName,SkateparkDescription,Latitude,Longitude)
11         with
               sqlite3.connect("skateboard_progress_tracker.db")
               as db:
12             cursor = db.cursor()
13             sql="insert into
                   Skatepark(SkateparkName,SkateparkDescription,
                   SkateparkLatitude, SkateparkLongitude)
                   values (?,?,?,?)"
14             cursor.execute(sql,values)
15             db.commit()
16             print()
17             print("Skatepark Successfully Created.")
18             print()
```

4.11.15 Reviews Widget

```
1  from reviews_toolbar import *
```

```
from reviews_sql_connections import *

import os
import sys
import re
class DisplayReviewsWidget(QWidget):
    """A class to display the reviews widget"""

    def __init__(self,parent):
        super().__init__()
        self.parent=parent


        self.RedBorder="border: 1px solid red;"
        self.GreenBorder="border: 1px solid green;"

        self.main_stacked_layout=QStackedLayout()

        self.add_review_VBoxLayout=QVBoxLayout()
        self.add_review_button_layout=QHBoxLayout()
        self.add_review_HBoxLayout=QHBoxLayout()
        self.add_review_widget=QWidget()
        self.add_review_table=QTableView()

        self.stacked_layout=QStackedLayout()
        self.results_table=QTableView()
        self.results_layout=QVBoxLayout()
        self.results_widget=QWidget()

        self.LayoutWidget=QWidget()
        self.ToolBarWidgetLayout=QVBoxLayout()

        self.open_connection()
        self.display_results()
        self.show_review_layout()



        self.add_review()

        self.display_review_layout()
        self.display_review_toolbar_widget()
        self.setLayout(self.ToolBarWidgetLayout)

        self.cancel_review.clicked.connect(self.cancel_button_clicked)
        self.save_review.clicked.connect(self.save_button_clicked)
```

```
48          self.review_type.currentIndexChanged.connect(self.update_review_boxes
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)
                and (ReviewName==True) and
                (ReviewName==True) and
                (ReviewRating==True) and
                (ReviewReview==True):
70              self.connection.add_review_to_database(self.review_type.currentT
                    self.review_brand.currentText(),self.review_name.text(),self.
                    self.review_review.text())
71              query=self.connection.show_all_reviews()
72              self.model.setModel(query)
73              return True
74          else:
75              print("Not all fields valid")
76              return False
77
78      def validate_review_type(self):
79          return True
80
81      def validate_review_size(self):
82          return True
83      def validate_review_name(self):
84          return True
85      def validate_review_brand(self):
86          return True
87      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 =
                    self.connection.show_all_reviews()
122
123             self.show_results(self.query)
124         else:
125             print("A DB Connection must be opened")
126
127     def show_results(self,query):
128         self.model = QSqlQueryModel()
129         self.model.setQuery(query)
130         self.results_table.setModel(self.model)
131         self.results_table.show()
132
```

```
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","
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 
175     def add_review(self):
176         self.add_review_table.setModel(self.model)
177 
178         if not hasattr(self,"review_type"):
```

```
179             self.review_type=QComboBox()
180             self.review_type.addItem("-Select a
                    Type-")
181             TypeOptions=self.type_options_check()
182             self.review_type.setModel(TypeOptions)
183 
184             self.add_review_VBoxLayout.addWidget(self.review_type)
185         if not hasattr(self,"review_size"):
186             self.review_size=QComboBox()
187             self.review_size.addItem("-Select a
                    Size-")
188 
189             self.add_review_VBoxLayout.addWidget(self.review_size)
190         if not hasattr(self,"review_brand"):
191             self.review_brand=QComboBox()
192             self.review_brand.addItem("-Select a
                    Brand-")
193             self.add_review_VBoxLayout.addWidget(self.review_brand)
194         if not hasattr(self,"review_name"):
195             self.review_name=QLineEdit()
196             self.review_name.setPlaceholderText("Product
                    Name")
197             self.add_review_VBoxLayout.addWidget(self.review_name)
198         if not hasattr(self,"review_rating"):
199             self.review_rating=QComboBox()
200             self.review_rating.addItem("-Select a
                    Rating-")
201             for count in range(1,6):
202                 self.review_rating.addItem(str(count))
203             self.add_review_VBoxLayout.addWidget(self.review_rating)
204         if not hasattr(self,"review_review"):
205             self.review_review=QTextEdit("Review")
206             self.add_review_VBoxLayout.addWidget(self.review_review)
207 
208 
209 
210 
211         if not hasattr(self,"cancel_review"):
212             self.cancel_review=QPushButton("Cancel")
213             self.add_review_button_layout.addWidget(self.cancel_review)
214         if not hasattr(self,"save_review"):
215             self.save_review=QPushButton("Save")
216             self.add_review_button_layout.addWidget(self.save_review)
217 
218         self.add_review_VBoxLayout.addLayout(self.add_review_button_layout)
219         self.add_review_HBoxLayout.addLayout(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             self.ToolBarWidgetLayout.addWidget(self.reviews_tool_bar)
234         self.ToolBarWidgetLayout.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
           tab"""
10     def __init__(self, parent):
11         super().__init__()
12         self.parent=parent
13 
14         self.add_review=QAction("Add Review",self)
15 
16         self.addAction(self.add_review)
17 
18             #connections
19         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
           the reviews tab"""
6
7      def __init__(self,path):
8          self.path=path
9          self.db=None
10
11     def open_database(self):
12         if self.db:
13             self.close_database()
14
15         self.db = QSqlDatabase.addDatabase("QSQLITE")
16         self.db.setDatabaseName(self.path)
17
18         opened_ok=self.db.open()
19         return opened_ok
20
21     def show_all_reviews(self):
22         query = QSqlQuery()
23         query.prepare(""" SELECT
               ProductType,ProductName,ReviewDescription,ReviewRating,ProductBra
               FROM review, product,
               productbrand,producttype,productsize""")
24         query.exec_()
25         return query
26
27     def get_all_product_type(self):
28         query=QSqlQuery()
29         query.prepare("""SELECT ProductType FROM
               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
            add_review_to_database(self,ReviewType,ReviewSize,ReviewBrand,ReviewN
59          pass
```

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
            view on the support tab"""
11
12      def __init__(self,parent):
13          super().__init__()
14          self.parent=parent
15          self.VBoxLayout=QVBoxLayout()
16          self.form_layout=QGridLayout()
17          self.form_widget=QWidget()
18
19          self.display_support_layout()
20
21          self.setLayout(self.VBoxLayout)
22
23          self.submit_button.pressed.connect(self.send_email)
24
25      def send_email(self):
26          msg=MIMEText(self.form_email_line_edit.text()+
                self.form_message_line_edit.toPlainText())
27          print(msg)
28          msg["Subject"]="Skateboard Progress Tracker
                Support"
29          msg["From"]="SkateboardProgressTracker@gmail.com"
30          msg["To"]= "BenKeppie@hotmail.co.uk"
31
32          Send=smtplib.SMTP("smtp.gmail.com")
33          Send.sendmail(msg["From"],msg["To"],msg)
34          Send.quit
35
36      def display_support_layout(self):
37          if not hasattr(self, "developer"):
38              self.developer=QLabel("Application
                    Developed By: Ben Keppie")
39              self.VBoxLayout.addWidget(self.developer)
40
41          if not hasattr(self, "form_name"):
42              self.form_name=QLabel("Name: ")
43              self.form_layout.addWidget(self.form_name,0,0)
44          if not hasattr(self,"form_name_line_edit"):
45              self.form_name_line_edit=QLineEdit()
46              self.form_layout.addWidget(self.form_name_line_edit,0,1)
47
48          if not hasattr(self,"form_email"):
49              self.form_email=QLabel("Email Address: ")
50              self.form_layout.addWidget(self.form_email,1,0)
51          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 *
8
```

```
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 
180             Finished=add_review()
181         elif Choice == 2:
182             Finishededit_review()
183         elif Choice == 3:
184             Finished=delete_review()
185         elif Choice==4:
186             Finished=filter_brand()
187         elif Choice ==5():
188             Finished=filter_type()
189         elif Choice==6():
190             Finished=filter_size()
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
                sqlite_master where name=?", (table_name,))
8           result=cursor.fetchall()
9
10          keep_table=True
11          if len(result)==1:
12              response=input("The table {0} already
                    exists, do you wish to recreate it?
                    (y/n) ".format(table_name))
13              if response == "y":
14                  keep_table=False
15                  print("The {0} table will be
                        recreated - all existing data will
                        be lost".format(table_name))
16                  cursor.execute("drop table if exists
                        {0}".format(table_name))
17                  db.commit()
18              else:
19                  print("The existing table was kept")
20          else:
21              keep_table=False
22          if not keep_table:
23
24              cursor.execute(sql)
25              db.commit()
26
27          db.commit()
28
29  def create_user_table():
30      db_name="skateboard_progress_tracker.db"
31      sql= """create table User (UserID integer,
            FirstName text, LastName text, UserPicture
            image, UserEmail text, Primary Key(UserID))"""
32      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
            integer, UserID integer, TrickID integer,
            Primary Key(UserTrickID), Foreign Key(UserID)
            references User(UserID), Foreign Key(TrickID)
            references Trick(TrickID))"""
38      create_table(db_name,"UserTrick", sql)
39      print("Blank UserTrick Table Created.")
40
41
42  def create_trick_table():
43      db_name="skateboard_progress_tracker.db"
44      sql= """create table Trick (TrickID integer,
            DifficultyID integer, TrickCreator text,
            TrickName text, TrickDescription text,
            TrickObsticle text, TrickImage image,
            TrickTutorialLink text, TrickCompleted boolean,
45      TrickCompletedDate text, Primary Key(TrickID),
            Foreign Key(DifficultyID) references
            Difficulty(DifficultyID))"""
46      create_table(db_name,"Trick", sql)
47      print("Blank Trick Table Created.")
48
49  def create_difficulty_table():
50      db_name="skateboard_progress_tracker.db"
51      sql= """create table Difficulty (DifficultyID
            integer, TrickDifficulty text,
            DifficultyDescription text, Primary
            Key(DifficultyID))"""
52      create_table(db_name,"Difficulty", sql)
53      print("Blank Difficulty Table Created.")
54
55
56  def create_user_review_table():
57      db_name="skateboard_progress_tracker.db"
58      sql= """create table UserReview (UserReviewID
            integer, UserID integer, ReviewID integer,
            Primary Key(UserReviewID), Foreign Key(UserID)
            references User(UserID), Foreign Key(ReviewID)
            references Review(ReviewID))"""
59      create_table(db_name,"UserReview", sql)
60      print("Blank UserReview Table Created.")
61
```

```
62 def create_review_table():
63     db_name="skateboard_progress_tracker.db"
64     sql= """create table Review (ReviewID integer,
          ProductID integer, ReviewCreator text,
          ReviewDescription text, ReviewRating integer ,
          Primary Key(ReviewID), Foreign Key(ProductID)
          references Product(ProductID))"""
65     create_table(db_name,"Review", sql)
66     print("Blank Review Table Created.")
67
68 def create_product_table():
69     db_name="skateboard_progress_tracker.db"
70     sql= """create table Product(ProductID integer,
          ProductBrandID integer, ProductTypeID integer,
          ProductSizeID integer, ProductName text ,
71     Primary Key(ProductID), Foreign
          Key(ProductBrandID) references
          ProductBrand(ProductBrandID),Foreign
          Key(ProductSizeID) references
          ProductSize(ProductSizeID),Foreign
          Key(ProductTypeID) references
          ProductType(ProductTypeID))"""
72     create_table(db_name,"Product", sql)
73     print("Blank Product Table Created.")
74
75
76 def create_product_brand_table():
77     db_name="skateboard_progress_tracker.db"
78     sql= """create table ProductBrand (ProductBrandID
          integer, ProductBrand text , Primary
          Key(ProductBrandID))"""
79     create_table(db_name,"ProductBrand", sql)
80     print("Blank ProductBrand Table Created.")
81
82 def create_product_type_table():
83     db_name="skateboard_progress_tracker.db"
84     sql= """create table ProductType (ProductTypeID
          integer, ProductType text , Primary
          Key(ProductTypeID))"""
85     create_table(db_name,"ProductType", sql)
86     print("Blank ProductType Table Created.")
87
88 def create_product_size_table():
89     db_name="skateboard_progress_tracker.db"
90     sql= """create table ProductSize (ProductSizeID
          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
            (UserSkateparkID integer, UserID integer,
            SkateparkID integer, Primary
            Key(UserSkateparkID), Foreign Key(UserID)
            references User(UserID), Foreign
            Key(SkateparkID) references
            Skatepark(SkateparkID))"""
97      create_table(db_name,"UserSkatepark", sql)
98      print("Blank UserTrick Table Created.")
99
100 def create_skatepark_table():
101     db_name="skateboard_progress_tracker.db"
102     sql= """create table Skatepark (SkateparkID
            integer, SkateparkName text,
            SkateparkLongitude integer, SkateparkLatitude
            integer, SkateparkDescription text,  Primary
            Key(SkateparkID))"""
103     create_table(db_name,"Skatepark", sql)
104     print("Blank ProductType 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
           sqlite3.connect("skateboard_progress_tracker.db")
           as db:
5          cursor=db.cursor()
6          cursor.execute("select * from User")
7          User=cursor.fetchall()
8          User=len(User)
9          return User
10
11
12 def add_profile():
13     existing_user=test_profile()
14     if existing_user==0:
15         FirstName=get_first_name()
16         LastName=get_last_name()
17         Email=get_email()
18         FilePath=get_file_path()
19         values=(FirstName,LastName,Email,FilePath)
20         with
               sqlite3.connect("skateboard_progress_tracker.db")
               as db:
21             cursor = db.cursor()
22             sql="insert into
                   User(FirstName,LastName,UserEmail,UserPicture)
                   values (?,?,?,?)"
23             cursor.execute(sql,values)
24             db.commit()
25             print()
26             print("Profile Successfully Created.")
27             print()
28     else:
29         print()
30         print("Profile already exists - To create a
               new profile, please delete the existing
               profile.")
31         print()
32
33 def delete_profile():
34     Finished=False
35     while not Finished:
36         Delete=input("Are you sure you wish to delete
               your profile? (Y/N) ")
37         Delete=Delete.upper()
38
```

```
39          if Delete=="Y":
40
41              data=(1,)
42              with
                    sqlite3.connect("skateboard_progress_tracker.db")
                    as db:
43                  cursor=db.cursor()
44                  sql="delete from User where UserID=?"
45                  cursor.execute(sql,data)
46                  db.commit()
47                  print()
48                  print("Profile Successfully Deleted.")
49                  print()
50                  Finished=True
51          elif Delete =="N":
52              print()
53              print("Delete Aborted")
54              print()
55              Finished=True
56          else:
57              print()
58              print("Incorrect input")
59              print()
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
            sqlite3.connect("skateboard_progress_tracker.db")
```

```
        as db:
82          cursor = db.cursor()
83          sql="update User set FirstName=?, LastName=?
                where UserID=?"
84          cursor.execute(sql,values)
85          db.commit()
86
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
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
98          cursor = db.cursor()
99          sql="update User set UserEmail=? where
                UserID=?"
100         cursor.execute(sql,values)
101         db.commit()
102
103 def get_file_path():
104     print()
105     FilePath=input("Please Enter The File Path For
            The JPEG Image: ")
106     print()
107     return FilePath
108
109 def change_picture():
110     FilePath=get_file_path()
111     values=(FilePath,1)
112     with
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
113         cursor = db.cursor()
114         sql="update User set UserPicture=? where
                UserID=?"
115         cursor.execute(sql,values)
116         db.commit()
117
118 if __name__=="__main__":
```

```
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:
               (Easy/Medium/Hard) ")
7          Difficulty=Difficulty.lower()
8          if Difficulty=="easy":
9              Difficulty=1
10             Finished=True
11         elif Difficulty =="medium":
12             Difficulty=2
13             Finished=True
14         elif Difficulty=="hard":
15             Difficulty=3
16             Finished=True
17         else:
18             print()
19             print("Please select a correct
                   difficulty.")
20             print()
21     return Difficulty
22
23 def get_trick_creator():
24     User="Name"
25     return User
26
27
28
29 def get_trick_name():
30     TrickName=input("Please enter the trick name: ")
31     return TrickName
32
33 def get_trick_description():
34     TrickDescription=input("Please enter the trick
           description: ")
35     return TrickDescription
36
37 def get_trick_obsticle():
```

```
38      TrickObsticle=input("Please enter the trick
            obsticle: ")
39      return TrickObsticle
40
41  def get_trick_image():
42      TrickImage = input("Please enter a JPEG images
            file path: ")
43      return TrickImage
44
45  def get_trick_tutorial_link():
46      TrickTutorialLink=input("Please enter a trick
            tutorial link: ")
47      return TrickTutorialLink
48
49  def get_trick_completed():
50      pass
51
52  def get_trick_completed_date():
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      TrickObsticle=get_trick_obsticle()
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,TrickObstic
67      with
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
68          cursor = db.cursor()
69          sql="insert into Trick(DifficultyID,
                TrickCreator,TrickName,TrickDescription,TrickObsticle,TrickImage,
                values (?,?,?,?,?,?,?,?,?)"
70          cursor.execute(sql,values)
71          db.commit()
72          print()
73          print("Trick Successfully Created.")
74          print()
75
76  def edit_trick():
```

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

4.11.25 CLI Skatepark Edit Options

```
import sqlite3

def get_skatepark_name():
    Name=input("Please enter the skateparks name: ")
    return Name

def get_skatepark_description():
    Description=input("Please enter the description
        for the skatepark: ")
    return Description

def get_skatepark_longitude():
    Longitude = input("Please enter the longitude for
        the skatepark: ")
    return Longitude

def get_skatepark_latitude():
    Latitude=input("Please enter the latitude for the
        skatepark: ")
    return Latitude




def add_skatepark():
    SkateparkName=get_skatepark_name()
    SkateparkDescription=get_skatepark_description()
    SkateparkLongitude=get_skatepark_longitude()
    SkateparkLatitude=get_skatepark_latitude()
    values=(SkateparkName,SkateparkLongitude,SkateparkLatitude,SkateparkDesc
    with
        sqlite3.connect("skateboard_progress_tracker.db")
        as db:
         cursor = db.cursor()
         sql="insert into
            Skatepark(SkateparkName,SkateparkLongitude,SkateparkLatitude,Skat
            values(?,?,?,?)"
         cursor.execute(sql,values)
         db.commit()
         print()
         print("Skatepark Successfully Created.")
         print()

def edit_skatepark():
```

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

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

```
44      with
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
45          cursor = db.cursor()
46          sql="insert into Review(ProductID,
                ReviewCreator,ReviewDescription,ReviewRating)
                values (?,?,?,?)"
47          cursor.execute(sql,values)
48          db.commit()
49          print()
50          print("Review Successfully Created.")
51          print()
52
53  def edit_review():
54      TrickID=int(input("Please enter the TrickID of
            the trick you wish to edit: "))
55      ProductID=get_product_id()
56      ReviewCreator=get_review_creator()
57      ReviewDescription=get_review_description()
58      ReviewRating=get_review_rating()
59
60      values=(ProductID,ReviewCreator,ReviewDescription,ReviewRating,TrickID)
61      with
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
62          cursor = db.cursor()
63          sql="update Review set ProductID=?,
                ReviewCreator=?, ReviewDescription=?,
                ReviewRating=?   where TrickID=?"
64          cursor.execute(sql,values)
65          db.commit()
66
67
68  def delete_review():
69      data=int(input("Please enter the ReviewID of the
            review you wish to delete: "))
70      data=(data,)
71      with
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
72              cursor=db.cursor()
73              sql="delete from Review where ReviewID=?"
74              cursor.execute(sql,data)
75              db.commit()
76              print()
77              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
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
17          cursor = db.cursor()
18          sql="insert into
                Difficulty(TrickDifficulty,DifficultyDescription)
                values (?,?)"
19          cursor.execute(sql,values)
20          db.commit()
21          print()
22          print("Difficulty Successfully Created.")
23          print()
24
25
26  if __name__ == "__main__":
27      create_difficulties()
```

4.11.28 CLI Make New Product

```
import sqlite3

def get_product_brand():
    Brand=input("Please enter a product brand: ")
    return Brand

def get_product_size():
    Size=input("Please enter a product size: ")
    return Size

def get_product_type():
    Type=input("Please enter a product type: ")
    return Type


def create_product_brand():
    ProductBrand=get_product_brand()

    values=(ProductBrand,)
    with
        sqlite3.connect("skateboard_progress_tracker.db")
        as db:
         cursor = db.cursor()
         sql="insert into ProductBrand(ProductBrand)
            values (?)"
         cursor.execute(sql,values)
         db.commit()
         print()
         print("Product Brand Successfully Created.")
         print()

def create_product_size():
    ProductSize=get_product_size()
    values=(ProductSize,)
    with
        sqlite3.connect("skateboard_progress_tracker.db")
        as db:
         cursor = db.cursor()
         sql="insert into ProductSize(ProductSize)
            values (?)"
         cursor.execute(sql,values)
         db.commit()
         print()
         print("Product Size Successfully Created.")
```

```
39          print()
40
41  def create_product_type():
42      ProductType=get_product_type()
43      values=(ProductType,)
44      with
            sqlite3.connect("skateboard_progress_tracker.db")
            as db:
45          cursor = db.cursor()
46          sql="insert into ProductType(ProductType)
                values (?)"
47          cursor.execute(sql,values)
48          db.commit()
49          print()
50          print("Product Type Successfully Created.")
51          print()
52
53
54
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.

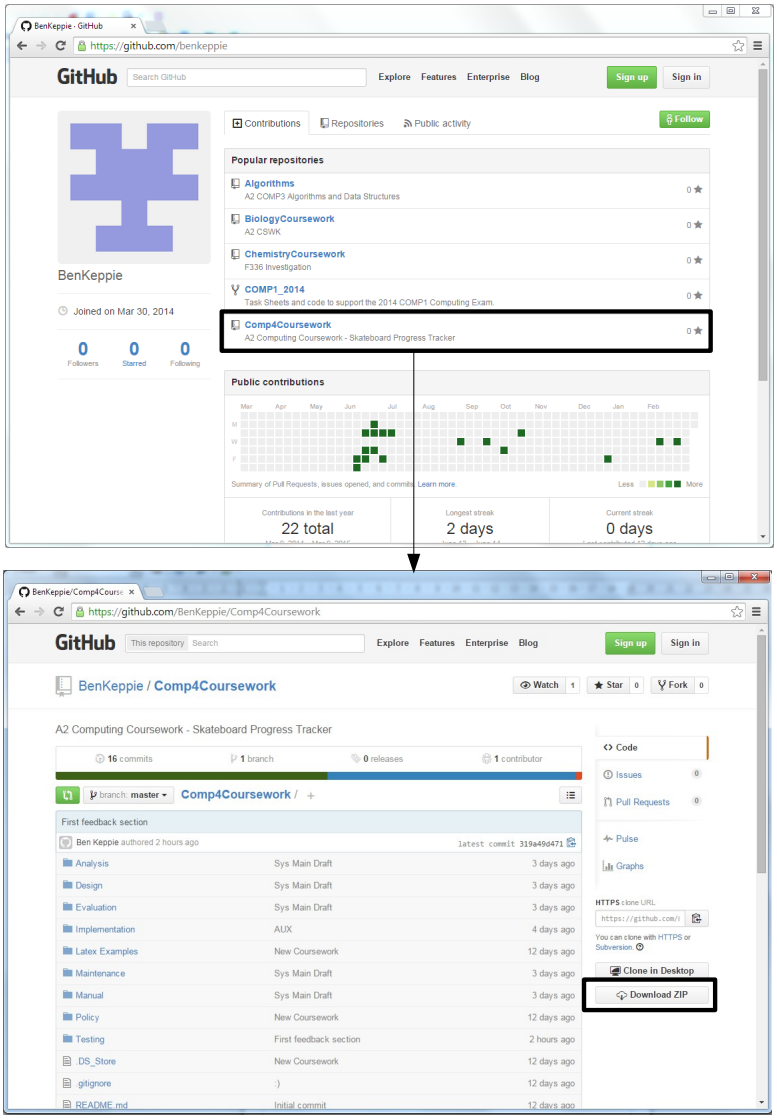


Figure 5.1: Downloading Skateboarding Progress Tracker Application

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.

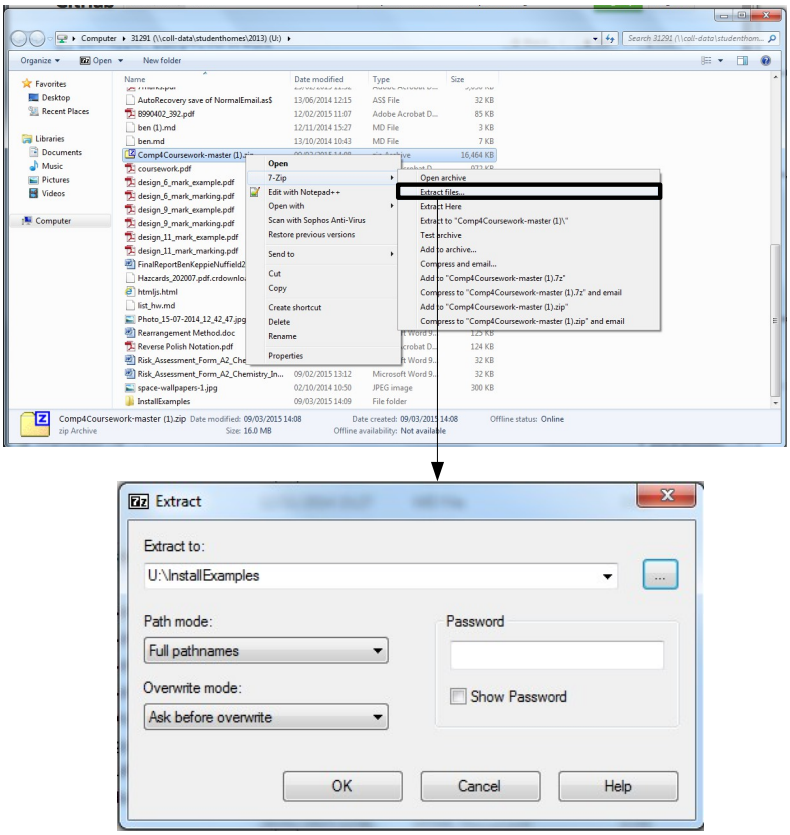


Figure 5.2: Extracting the zip File

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.

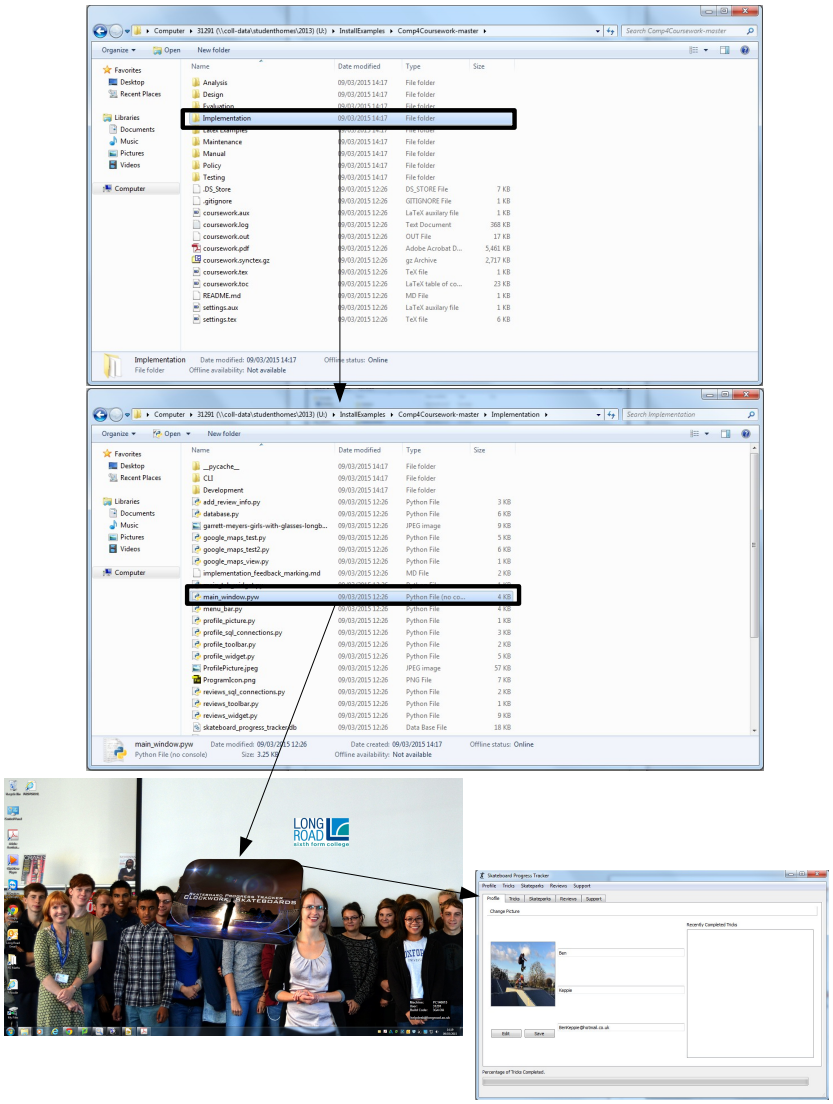


Figure 5.3: Starting up the program

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.

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