

P.E.T CAR RENTAL SYSTEM

Raees Qaisir

THE ROYAL GRAMMAR SCHOOL

Car Rental Coursework

Contents

Analysis	2
Background	2
Why does this system need to be implemented?	2
What should the new system do (Success Criteria)?	2
Problems with the existing system (why is it solvable by computational methods?)	5
Why using the Platform is the best way to solve the problem?	6
The project can be broken down into 3 main sections.....	6
Interview with Stakeholders (Godwin)	6
Analysis of interview with Stakeholder.....	8
Analysis of interview with Customer (Richard).....	9
Researching Existing Companies.....	9
Hardware/ Software requirements (recommended)	11
Limitations- What cannot be included within the Programme	12
Design- Coursework	13
Breakdown of Solution:	13
Algorithms.....	17
The initial form ideas:	24
Different buttons the user can interact with, in the programme:.....	26
Different Inputs the user can interact with in the programme	27
Test Plan.....	29
What can I Test Before the Programme is Complete?	38
Code Development- Section 3	38
Evaluation	90
Comparing Forms Design	125
Test Plan:.....	131
Failed Tests.....	144
Maintenance and Development	146
Big O time complexity of the Programme:	146
Maintenance Issues (Portability of Programme)	147

Analysis

Background

- Premium Executive Travel (PET) are a relatively new company. They have been a listed company for 3 years now. However, in that time they have gained many employees who live in the UK. They have bases in London, Outer London, Manchester, Liverpool, Birmingham and Glasgow. The company has several cars to choose from, brands including Mercedes, BMW and Audi.
- The owners (stakeholders) of the company are
 - Godwin
 - Olek
 - Josh
- Godwin will be the stakeholder who I will be showing the developments in the programme to. He will be checking to see whether the success criteria are whether enough of the criteria has been met. The other stake holders (Olek and Josh) will also have input into the design of the programme, however they will not be present for most meetings. This is known as agile development, where I create a function and show the result to the stakeholder (Godwin), and he gives me feedback as to what I can improve. I then go back and make the improvements to each function. The reason for using this methodology is because I have used the OCR Computer Science textbook written by George Rouse, Jason Pitt and Sean O'Byrne, specifically chapters 19-21. There, the text book has given a breakdown of the solution and how I should present the different stages to Godwin and what is the best way to approach a complex solution.
- The programme I will be creating for Premium Executive Travel will be a finite solution. This is because there are only a limited amount of options that will be available within the programme. This means that the different options can be accessed within an amount of time.

Why does this system need to be implemented?

- The aim of creating this system is to help a car rental company (Premium Executive Travel) organise customer information and provide an easy and reliable system for a customer to make their rental booking.
- The end user is looking to rent a car for a period time from this company. They want the process of the renting application to be simple yet thorough as well as can easily navigate the way through the form and browse the different options available to them. They target group for the people who will be looking to rent a car from this service must be over the age of 21, gender does not affect the final price of the quote.

What should the new system do (Success Criteria)?

1. Branches will be available if the customer needs to go into a store
2. The branches are also there for customer to pick up and drop off the car
3. Customers as well as staff should be able to use this system effectively and with relative ease. This will be done by using multiple data entry forms (booking form, customer detail forms)
4. The programme should offer the customer the option to purchase the car after they have rented one

5. The programme should offer a discount on the car the customer has rented if they want to purchase it
 6. The programme should offer different cars that are available within the programme if the customer wants to purchase one
 7. The programme should be able to search for different cars depending on the search criteria the customer enters if they want to purchase a car
 8. The programme should display the contact details of the company
 9. The programme should display images of the cars that are on offer from the company
 10. The programme should allow a user to create a CV, so they can apply for a job at the company
- Login
 1. When the programme is run, the login screen should be displayed
 2. The login screen should take the inputs (username, password) from the user
 3. The inputs should be checked to see whether they match previous data from the database
 4. If they match the user should be shown a message box
 5. If they match the user should be shown to next screen (booking form)
 6. If they do not match, then the user will be prompted to re-enter details by a message box
 7. If the user does not have an account made, they should be able to make one
 - Sign Up
 1. The programme should be able to "tab" through the different text boxes
 2. The programme should take inputs from the user in the text boxes
 3. The form includes all details that may be needed by the programme
 4. The information entered by the user in the sign-up form should be stored to a database
 5. The information should be stored in the same order it is taken from the user
 6. The programme should auto increment a unique id for each new user
 7. The programme should be able to hash a password a user has entered when they sign up
 8. The programme should only allow the user to "sign up" if they have a username longer than 4 characters and a password longer than 5 characters
 9. The programme should not accept a username less than 4 characters
 10. The programme should not accept a password less than 5 characters
 11. The programme should not accept a username if it already exists in the database
 - Changing/Forgot Password
 1. The programme should allow the user to change password
 2. Each button displayed on the window should carry out a function similar to the description of the button
 3. The programme should prompt the user to enter their email
 4. The programme should send an email to the account they have entered
 5. The email should be from the PET company and have a professional format
 6. The programme should generate a random code
 7. The code should contain a mixture of letters and numbers
 8. The programme should then open a new window
 9. The code should then be copied back into the new window opened by the programme
 10. A new window should open

11. This window will allow the user to change their password to a new one
- Admin
 1. Depending on the type of account the user signs in with, they should have access to different options (Admin)
 2. If the admin details are correct, a new window should appear giving the admin different options dependent on what they want to view
 3. If the admin details are incorrect, the new window should not open, instead a message box alerting the user that they have entered incorrect details
 4. Allow only the administrator to make changes to user details
 5. The information should be updated in the database
 6. Allow only the admin to delete user accounts
 7. If the user signs in with the admin the programme will display graphs to show the most popular rented car
 8. When the user hovers over a “slice” the “slice” should enlarge slightly

User Booking

1. The user should be able to delete their booking by signing back into the programme
2. The user should be able to view how long they have hired a car for
3. The user should be able to change how long they have hired a car for
4. The programme should be able to give the customer a view of all the cars available
5. The programme should prevent the customer from making a booking if there aren't cars available
6. The programme should prevent a customer from making a booking if they are below the age of 21
7. The programme should calculate the age of the user when they input their DOB
8. The programme should save the details of the booking to the database
9. The programme should limit the number of decimal places calculations are done to
10. The programme should display prices dependent on the age of the customer
11. The programme needs to consider the amount of days the car has been rented for
12. The programme needs to consider the different prices the different cars will have
13. The programme should provide a discount for corporate customers
14. The programme should provide discounts for returning customers
15. The programme should display similar alternatives for the customer if they cannot make a booking
16. The programme should allow the user to filter through cars from a drop- down list
17. The programme should be able to provide a receipt of the transaction the customer may have completed
18. The programme should allow the customer to print off a receipt
19. The programme should price surge. This is when there are peak rental times, so the price of the car that is being rented will increase due to demand.

Contact us

1. The programme should display contact information for the company
2. The programme should can create a CV for a user
3. The programme can print this CV for the user

Cars

1. The programme should display the different cars on offer from the company

Validation

1. The programme will validate data that the user has inputted into the text boxes
2. The programme should present the user with enough buttons to easily navigate through the programme
3. The programme should have a unique ID for each customer
4. Each table within the database must have a primary key
5. All data should be stored in the correct format

Problems with the existing system (why is it solvable by computational methods?)

- There were a lot of hard copy files. This meant they were easily lost and damaged. If sensitive information was to be on the hard copy documents, then the company would be liable for breaking the law as they had not kept the information safe. Furthermore, the hard copy files were not an efficient use of space or time, this was because more money was spent on rent trying to store the information in storage facilities/offices. Searching and trying to make edits to the existing customer information was also extremely difficult. This is because the staff would have to manually search through all the existing files and then make an edit, then find a way to save the new copy. This was extremely inefficient use of time. The new system will mean that making edits to existing data will be faster as well as more secure as all the customer's details will be in one place so it will be better protected through the use of security keys
- Since a lot of calls came through on the telephone some customers were not always connected to a member of staff. This means the company would be losing out on money and potential clients, however with the new system it will be available 24 hours of the day. Any specific requests can be dealt with in the morning by staff in an organised manner.
- Using an online, real time system, will prevent and redundancies in data.
- 24-hour service was not available
- This system will be valuable to the stake holders involved because this way they can maximise their profits.
 - Extra money for storage won't have to be spent for the hard copy files.
 - An increased number of potential clients are available, as many more people have access to the internet.
 - Less time will be spent by staff trying to make edits or search for existing clients, so their time will be utilised properly.
- To fix this I will create a database which will include all relevant details
 - The database will include separate tables for customer and car
 - The following table will include columns such as (Make/Model, Type, Registration, Date the car was issued, Expected return date of the car,)
 - Another table for the customer details (First/ Last name, copy of official document, Address, Amount placed for Deposit, amount they have paid for the car)
 - These tables will be kept separately for security reasons as well as to keep the database organised.
 - Make a more user friendly GUI, so the customer can search and book easily and with minimal time as possible. It also means staff members can have information about the different cars to hand quickly improving the customers experience with the company.

- Use a more secure method of a customer making payment (PayPal)
- Allow the manager access to the different staff accounts (administrator)
- Since the processing will be done by the system, there won't be need for hard copy documents, unless the customer specifically asks for one. This means details of customers will be very secure and prevent the chance of other people who should not have access to it, having access to it.

Why using the Platform is the best way to solve the problem?

- I have chosen to use python as my platform to code the programme. This is because I believe python offers a lot of in built modules that would be useful (hashlib,random). Furthermore, using python means when I create my different forms, which will be the GUI that a user uses, it will be easier to integrate into my programme. This is because python is used widely for graphics design. Furthermore, coding is simpler in python than compared to other languages, as it closer to spoken English. For example, the syntax difference between python and java are quite large.
- I could have used other programming languages such as PHP, however, this would mean I would be limited to web development. Initially the programme would need to be created, tested and then improved. This would be easier to do in python. Once the programme is complete it could then be written in PHP, by that time the company should also be more popular so there will be more traffic on the company website making use of the programme. Furthermore, since I wasn't using other programmes such as word press to create a website, PHP did not make sense to use.

The project can be broken down into 3 main sections

- The database- this will include the information about the customer/ staff as well as information about the different cars the company offers. For this I, will be using SQL. The database is important as it will be storing customer information such as their booking details as well as personal details. The information stored in the database will also be used to login into the programme.
- Creating the GUI, for this I will use the programme PYQT. This is important as this is what the user will see when they are using the programme. This means they GUI should be aesthetically appealing as well as being able to provide easy navigation for the user.
- Python will also be used to code and provide the different functions that each form should be doing.

Interview with Stakeholders (Godwin)

- **What are the objectives of your organisation?**
 - *Maximise profits*
 - *Create an easy to use simplistic booking system for customers to choose between our range of cars on offer to hire.*
 - *Expand our company into different countries*
 - *Become the leading Car rental company in the UK*
 - *Provide a happy work place*
- **What are the current procedures of your organisation?**
 - *We have some implementation of computer systems within our procedures, however they are not currently being used effectively. For example, when a customer comes in and wants to rent a car we have to take their details. These details are not properly filed and kept safe. Scans should be taken and stored in a main folder, and then sub folders, from there onwards. However, this does not*

happen which means we could be liable to being sued if sensitive data was released to others.

- *Furthermore, we lose a lot of customers through our website as it is hard to navigate when trying to book a car online. Thus, customers turn to other car rental companies.*
- *We do however have excellent staff who are always willing to help and communicate with customer effectively when they need help or have any queries.*
- **What computer systems do you currently use?**
 - *It's very basic, and slow;*
 - *Windows XP*
 - *Microsoft Office 2003*
 - *AMD Processor*
 - *2GB RAM*
 - *250GB HDD*
- **Who are the biggest competitors and what worries you about them?**
 - *Our biggest competitors are companies such as Hertz and Enterprise. Both are well established companies who provide customers with an easy experience trying to book their hire car. This is mostly done using their website. They are easy to find and access as they have many branches around the country and abroad.*
- **How do you expect to make your company different so that customers choose you?**
 - *The majority of large companies struggle to employ good staff who actually take the time to go the extra mile and help our customers who have questions/queries. However, our company has just that, excellent staff which is why I think our company has the ability to expand and grow to twice the size of the market leaders.*
- **What would you like most from the implementation of the new system?**
- *The company now isn't maximising its profits. This is due to the lack of accessibility for the customers (internet). Being inaccessible means that customers go to other companies rather than with us, which means we are losing money. Furthermore, a lot of time from staff is being spent on trying to organise files from customers, such as their booking order and personal information. This means their time isn't being utilised properly.*



-
- **Interview with a previous customer (Richard)**
 - **What is a must for you when trying to hire a car?**
 - *Being able to book the car via the internet. This means that an easy to navigate, stress free website would be ideal as it can be done quickly with minimal issues.*

If problems to arise, then customer service representatives who are helpful and willing to help you would be good.

- **Would you refer our company to friends/colleagues?**
 - *At the moment, it would be difficult to do so. This is because your company doesn't have a large presence on the internet and isn't easy to find. For example, you could improve on your home page when customers visit your site, and increase your presence on social media.*
- **What companies do you refer to friends/ colleagues?**
 - *Companies such as Hertz, Rent-a-Car, Enterprise.*
- **If you could make an improvement to any car rental company now, what would it be?**
- *Previously when we have hired a car before from car rental companies, we have really enjoyed driving them. We have asked before weather we could purchase the car from the company (the one we have rented) but the answer has been no. an improvement could be to make a way for us to be able to have the option to purchase the car. This could be done through the website or by going in branch and asking the customer service. We would rather purchase the car from the car hire company s they are usually the brand-new models and have low mileage. It also means we know how the car drives and feels, which plays a big part when buying a car. Furthermore, car rental companies would have to keep good records of the history of the car.*



○

Analysis of interview with Stakeholder

- One thing that was made abundantly clear from the interview with the stakeholder was that profits had to be maximised for the company. He had already listed some ways in which this goal could be achieved through the implementation of the new system. Another problem which was clear form the interview was that the computing equipment the company was using was extremely outdated. This would mean that simple processes such as updating spreadsheets or having multiple processes running at the same time would be slow and strenuous, thus leading back to the problem that the company isn't maximising profit due to slow processing times. Lastly improvements can be made on the company's current procedures to process new requests from customers. For example, a lot of the information being processed is still being stored on paper and being filed away, however this could lead to a messy and un reliable system. Moving the information to a new database via the new system will mean it will be faster to access, this is because it is faster to search through information on a computer than it is manually as the computer has more processing power. Only staff who are allowed

access to the information are able to see it, furthermore making backups of this data will be easier, thus the data will be more secure overall.

Analysis of interview with Customer (Richard)

- From the interview, I can conclude that we need to be easily accessible to the outside world. This is mostly likely going to be achieved through the presence we hold on social media and on the internet. We need to improve on our website and make it easier for customers to make a booking, whilst at the same time still collecting all the required information. This can be done by creating a new GUI (which means new fonts, colours and pictures and information can be added). The customer also has mentioned that companies such as Hertz are our biggest competitors, this means when I am collecting research to bring back to the stakeholders, I should look at what Hertz is doing well and the possible downsides to their current system. Moreover, the customer also wanted a place where they could buy the car they have rented. This could be an advantage to our company if we can make it work as it would provide us with something niche. Not a lot of car rental companies are offering this.

Researching Existing Companies

- There are already some companies that offer the services that the end user is looking for, for example; Hertz
 - As soon as you click on their website link, the “welcome” page is loaded. On the “welcome” screen you are immediately shown their booking form asking for dates you require the vehicle from/to and the type of vehicle you want. This means the customer can enter details straight away which means there is a higher chance that the customer will be spending money with the company. This is because they don’t have to waste time and become frustrated by searching for different forms by clicking different tabs. It gives the customer more incentive to make a booking. To create something like this design I would have to use PYQT to create my GUI and have all the relevant information on the form the customer would be required to fill out.
 - The company offers the user to make a booking as a “guest”, however, in the programme I will be creating, you have to create an account to be able to make a booking with Premium Executive Travel



Book a Car View/Modify/Cancel a Reservation

1 Pick-up Location (City, State or Airport Code)

[Help me find a Hertz rental location](#)

☐ Return car to a different Hertz location

2 Pick-up Date and Time: 12:00

Return Date and Time: 12:00

3 Your age at time of pick-up

Your age at time of pick-up *

Hire Car Type:

Show Me All

☐ Use my Hertz Gold Plus Rewards Points

☐ Apply a discount code

Book as a Member Book as a Guest

- Furthermore, they have a simplistic and easy to navigate page. This has been achieved using hyperlinks and tabs. This means the customer doesn't become frustrated through the process of booking a car, which means they are more likely going to spend their money with this company. The tabs at the top of the page are also drop down lists. This gives the customer more information as to where they need to be searching on the website to find answers to queries. I would also include something similar to this in my programme, having separate pages or text that has been hyperlinked so it sends you to a different page depending on where you click. This would be down through the use of PYQT (creating the buttons) and Python (having the code so it sends you to the correct page)



- At the top of the page the customer has the option to login into the website. Hertz offer discount and loyalty rewards for becoming a "member" of their

company. This could entice some customers into becoming a member, which means they are also more likely to make bookings. I could implement something similar to this into my programme, a place for customers to login to see what awards they have achieved. Moreover, the staff could also use a similar set up to login in a be shown a different dashboard, one that would help them with completing bookings. This would be done through the use of PYQT (creating the buttons and having the writing in the correct place on the GUI) and using Python and SQL to compare the entered username and password to the database which contains all the username and passwords.



Hardware/ Software requirements (recommended)

- For the programme to run, the user will need
 - *Windows 7 or higher*- this is so that the system can be utilised fully, efficiently and to ensure maximum compatibility
 - *A copy of python installed, 2.7 or newer (Python 3.4 would be optimal)*- so that the programme can be read
 - *SQL lite installed*- this is so that the database can be read and written to
 - *PyQT (4.8.4)*- this is so that the forms can be loaded
 - *A web browser* (Chrome, Explorer, Firefox)- this is so that the programme could eventually be used on a browser
 - *An internet connection*- the website will be internet based eventually
 - *Communications* (outlook)- to be able to communicate to customers and staff)
- A computer with (recommended):
 - *4GB Ram*- this is so that the system can run as smoothly as possible without interrupts affecting the operating system, or other background processes
 - *Intel Core Duo/AMD Athlon 64X2 2GHz or higher*- this is so that processing can occur quickly and efficiently
 - *1 terabyte hard drive* - this is so that there will be plenty of space to store the required programmes that need to be downloaded for the programme to run. Furthermore, the database will most likely increase in size as the company grows
 - *Access to external hard drives* (preferably 1 terabyte or larger)- this is so that data and programmes can be stored to the hard drives as a form of backing up the data
 - *Access to a server* (nasbox is optional)- this is so that all staff in the organisation can access the booking system
 - *Web hosting server*- this is so that any internet traffic can be passed through an external company which a subscription must be paid to. This usually depends on how much traffic the company is expecting to have

- A display (1920x1080)- this is so that the GUI can be viewed. A higher resolution will mean that reading text from the screen will be easier, furthermore, any images on the programme will look clearer. This is also so that the programme will be displayed correctly on the monitor
 - Mouse and keyboard- this will be used to input data into the system
- Further meeting with Stakeholder (Godwin)

I have carried out research into existing companies with a similar product to the ones we are offering. Furthermore, I have also analysed our previous meeting in which I interviewed yourself and a customer. From this I have collated information, both good and bad which I will now present you both.

The good

I would suggest that we create a new user interface for the customers, one that will make booking a hire car much easier for them than it is now. One suggestion would be to have the home page with a miniature booking form on it. This form would only take up a quarter of the page but it starts the booking process straight away, thus making the customer more likely to continue booking with us. When the customer clicks next, it would take them to a different page which is more in depth and will ask them for their personal details. Another good idea which we should implement into our programme is having short cuts. This could be in the form of having text/button which is hyperlinked to a different page so the customer doesn't have to spend time searching for it can see the information right away. Furthermore, offering incentives such as money off bookings or deals would also entice customers to make a booking. This could be displayed alongside the booking form which would be on the home page, when the customer see's the deal, they are more likely to book with us.

Limitations- What cannot be included within the Programme

Unfortunately, we cannot offer everything that you have asked for. During the interview with the customer, Richard suggested that it would be a good idea for the company to offer the customer the option to purchase the car once it had been rented. This is because the customer (Richard) may have really enjoyed the experience of the car during the days he had rented it for. Instead of having the hassle of finding another car with a similar specification, a new dealer and negotiating on price, it would be easier for Richard to purchase the car directly from PET. Furthermore, another feature that would be good would be the ability for Richard to search through a list of different Premium cars with specific search criteria and the programme would return cars for sale with similar criteria. However, this is not something that I would be able to achieve within the time frame I have. This is because, the main objective is to get a programme that makes renting a car from PET easier and more organised. Purchasing a car is something else entirely as a new system would need to be implemented as well as more new data. Moreover, since the type of car PET offers are "premium" they would have a high price and they are not easy to find, compared to a "normal" car such as a

Vauxhall Astra. This is something to work towards in the future, once the rental system has been created and implemented.

Another limitation I have is surging the prices of the cars when they are being rented. This is cannot be done because the programme would constantly be having to check the most popular car that is being rented and also find out the time at which this is happening. Working with time in Programming is very difficult, furthermore, the programme would have to check what a suitable price would be to raise the current price to. This means it would have to find a way to be connected to the internet, so that the programme can find the current price of a similarly spec car.

Conclusion

The implementation of the new system will mean the company will be provided with a new GUI, a simplistic and easy to navigate design for the customer to make their booking, this will be achieved through using PYQT and coding specific functions in Python. A new organised database will be created for the company to store customer and staff and vehicular information. This will be created by using SQL. Everything that was included in the success criteria at the top of document is what we are aiming to include in the final programme.

Design- Coursework

Breakdown of Solution:

The programme being created is complex and has multiple parts to it. Therefore, I need to break down the programme into smaller, easier parts which will be more manageable. The 3 main parts I will break the programme down into will be creating the forms, creating a database and finally creating the programme. Within these 3 parts, I will further have to break down each part.

Initially I will need to create the different forms I will be using to collect the data from Bob. To do this I will need to choose a programme that allows you to create GUI interfaces. At the moment, I can choose between using Tkinter or PyQt. The forms will need to have a professional design as well as being easy to navigate for Bob. This is so that all the required information can be collected as Bob does not get confused or miss out details.

Once the forms have been created, I will then need to create the database for the programme. This is where the data that is collected from Bob will be stored. The database will need 4 tables. Data will be written to and read from the database. I will most likely choose SQL to create the database. Once this has been done I will also then be able to create the SQL queries that will allow me to write and read from the database within the programme. The database structure I will be using is dynamic. This is because there will be data that is inserted to, and deleted from the different tables in the database.

Once the database is created, I will then start to work on creating the programme and the different classes that will be included. There are many languages to choose from to create the programme, however I need to choose one that is easily integrated with PyQt and SQL. Furthermore, I need to choose a language that I am confident with to create a complex programme. Therefore, I will most likely choose Python. Python is where I will create the different functions and procedures to create

the programme, as well as include the different SQL queries so that data can be read from and written to the database. Furthermore, I will work on the programme each class at a time, creating each function within the class one step at a time. This is so that I can find any errors within the code and make the process more manageable.

Use Case Diagram (Login)

<u>User Action</u>	<u>What Should the Programme Do</u>	
Bob will enter their user name and password into the required fields.	The inputted data should be checked with existing data in the database. If they match correctly, then Bob will be allowed access to the next page. The password entered will be hidden by the use of asterisks. The password will also be hashed.	
Bob can now continue		

Use Case Diagram (Sign Up)

<u>User Action</u>	<u>What Should the Programme Do</u>	
Bob will click the "Sign-up" button	The programme will display a new window with additional fields that Bob needs to fill out.	
Bob will click the "Sign Up" button	The programme will store the entered information into the database.	
Bob will click "go back" button	The programme will display the previous page (home screen)	
Bob can now continue to log into the programme.		

Use Case Diagram (Cars)

--

Bob will click "cars" button	The programme will open a new window which displays the cars available to the customer which they can hire.	
Bob can click "go back" button	The programme will display the home window	
Bob can continue		

Use Case Diagram (Admin)

User Action	What Should the Programme Do	
Bob will enter "admin" into username box and "raees" into the password box.	The programme will display a new window. This window with additional push buttons for Bob to click.	
Bob will click the "Customer" button	The programme will open a new window which displays all customer information from the "Customer" table from the database.	
Bob will click a cell	The programme should display what cell is being edited	
Bob will enter new data into the cell and click the update button	The programme should save the new data entered into the cell into the database.	
Bob will click "go back" button	The programme will display the previous page (admin window)	
Bob can click the "graphs" button.	The programme will display a new window which displays the most popular car that is being rented at the current time. Other graphs which show monthly incomes and outgoings will also be displayed.	
Bob will click "go back" button	The programme will display the previous page (admin window)	
Bob will click the "Home" button	The programme will display the previous window (Home window)	
Bob can now continue		

Use Case Diagram (Forget Password)

User Action	What should the Programme Do
-------------	------------------------------

Bob will click the “forget password” button	The programme should open a new window	
Bob will enter an email into the text entry. Bob will then click next	The programme will send an email to the email address that has been entered	
Bob will copy the code that was in the email into the programme. Bob will then click the next button	The programme will open a new window	
Bob will then enter a new password into the text entry	The programme will save the new password and enter it into the database	
Bob will click the “home” button	The programme will return Bob to the home screen	
Bob can now continue		

Use Case Diagram (Booking)

<u>User Action</u>	<u>What Should the Programme Do</u>	
Bob will login to the programme	The programme should display a new window where Bob can make a booking if the details entered are correct	
Bob will select a car from a drop down list he wants to hire	The programme should save the car that Bob wants to hire and display it at the top of the drop-down list	
Bob will click the date he wants to hire the car till	The programme will then carry out a calculation dependent on the amount of days Bob wants the car for and dependent on the car he wants to hire. The programme will display the final price Bob has to pay, including VAT	
Bob will click the finish button	The programme will save the details that Bob clicked into the data base. A new window will open when the “finish” button has been clicked. Booking will only be saved if Bob is over the age of 21	

Use Case Diagram (Contact Us)

<u>User Action</u>	<u>What Should the Programme Do</u>
--------------------	-------------------------------------

Bob will click the “contact us” button	The programme will open a new window which displays contact information	
Bob will click the “cv” button	The programme will open a new window	
Bob will enter text into the relevant boxes	The programme will display the text being entered into each text entry box	
Bob will click the “show me cv” button	The programme will open a new window. The new window will display the text that was entered in the previous window on the new window in a similar format	
Bob will click the “print cv” button if he is happy with the information he has entered	The programme will display a new window where Bob can choose which printer he wants to send the document to. Bob can also change his preferences here	
Bob will click the “cancel” button	The programme will display the previous window	
Bob will click the “edit cv” button	The programme will open the orevious window where Bob had previously entered text	
Bob will click the “home” button	The programme will return Bob to the home screen	
Bob can continue		

Algorithms

Login Algorithm

BEGIN Login

INPUT username

INPUT password

FOR LENGTH OF CustomerDatabase

FOR EACH Customer

IF Self.uname = username THEN

```
templID <= Customer.ID OF Customer WHERE  
CustomerDatabase.uname = username
```

IF Customer self.Password = password THEN

OUTPUT "Correct username and password"

```
                CALL MainScreen
            ELSE
                OUTPUT "The password is incorrect. Please try again"
                CALL Login
            END IF
        END FOR
    END
```

Login Algorithm Overview

Customers and staff will have access to a login form. There they will be asked to input their username and password. This will then be checked against the data already added into the database, specifically the customer details table. If the username and password match, then he/she will have access to the next form (booking form)

New Login Algorithm

```
BEGIN CreateNewCustomer
    INPUT signup.uname
    IF LENGTH(signup.uname) > 14 THEN
        OUTPUT "Username is too long. Please enter a shorter username."
        CALL CreateNewCustomer
    END IF
    IF LENGTH(signup.uname) =< 2 THEN
        OUTPUT "Username is too short. Please enter a longer username."
        CALL CreateNewCustomer
    END IF
    INPUT signup.pword
    IF LENGTH(newPass) > 20 THEN
        OUTPUT "Password is too long. Please enter a shorter password."
        CALL CreateNewCustomer
    END IF
    IF LENGTH(signup.pword) =< 2 THEN
        OUTPUT "Password is too short. Please enter a longer password."
```

Raees Qaisir

```
CALL CreateNewCustomer
INPUT verifysignup.pword
IF verifysignup.pword=signup.pword THEN
    OUTPUT "Passwords match"
END IF
END CREATE
OUTPUT "You have successfully created an account"
Login
END
```

SignUp Overview

The customer will press the SignUp push button. Another form will then open where the customer will enter their username and password. They will then be prompted to re-enter the password and check if it matches the input from the first password input. A dialog box will then appear alerting Bob that they have created an account which now means they can sign into the programme. Their username and password will be written to the database, specifically the customer table.

Booking Algorithm

Begin Booking

Booking Algorithm

Begin Calculation

```
If user 25<age>21
    SELECT car_rental FROM RentalDB
    CALL start_date()
    CALL end_date()
    Num_days= end_date-start_dat
```

```
Total_cost=car_rental*Num_days*225

If user 65<age>25

    SELECT car_rental FROM RentalDB
    CALL start_date()
    CALL end_date()
    Num_days= end_date-start_dat
    Total_cost=car_rental*Num_days*125

If user age>65+

    SELECT car_rental FROM RentalDB
    CALL start_date()
    CALL end_date()
    Num_days= end_date-start_dat
    Total_cost=car_rental*Num_days*190

OUTPUT Total_cost

Else

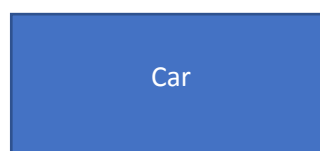
If user_age<21

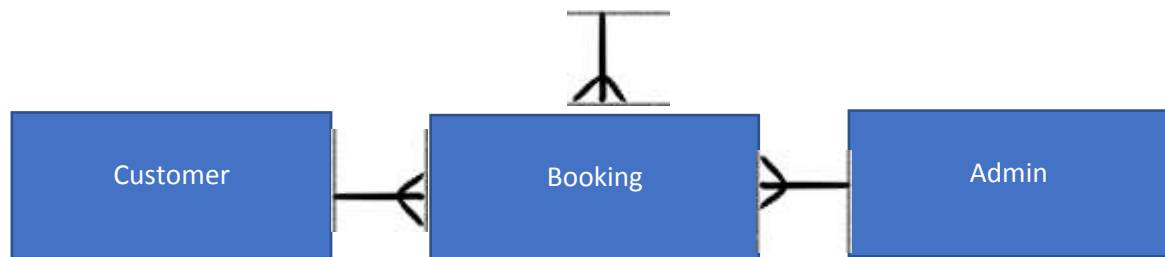
    OUTPUT Sorry, you are not of age to rent a car
```

Overview of Booking Algorithm

When a customer tries to make a booking, they will have to input their age. If they are older than 21 they will be allowed to continue with the booking. The booking will ask them for the date the car is required from and they date they want the car to. Next Bob will select the car they want. The total cost will then be calculated by multiplying the cost of the car with the daily rate. The cost of each car will be different.

Entity Relationship Diagram :





Bookings Table

Field Name	Type	Example	Description
BOOKING ID	VARCHAR	BI1234	Each booking a customer makes, they are given a unique Booking ID number (PK) which has all the information related to their booking (Car, Price, Date)
REGISTRATION	VARCHAR	RJ60RWB	Similar to a primary key. Each vehicle has its own unique number plate
RENT DATE	INT	30/11/12	The date which the customer has rented the car from
RETURN DATE	INT	01/12/12	The date the customer has to return the car
AMOUNT PAYABLE	INT	2000	Total amount the customer has to pay for the days of rental

Car Table

Field Name	Type	Example	Description
REGISTRATION	VARCHAR	RJ60RWB	Similar to a primary key. Each vehicle has its own unique number plate
Make	STRING	MERCEDES	The brand of the vehicle, allows the

			customer to decide which car to rent
MODEL	VARCHAR	E CLASS	Allows the customer to decide exactly which model they want to rent
Price	INT	2500	This will be the amount of money the car will cost to hire per day. This will be a figure used when calculating the total cost of the rental

Customer Table

Field Name	Type	Example	Description
CUSTOMER ID	VARCHAR	CI1234	Primary key for the customer table
Username	VARCHAR	userBob	Allows the customer to sign into the programme
F.NAME	STRING	BOB	Information needed for the customer's record
S.NAME	STRING	Smalls	Information needed for the customer's record
ADDRESS	VARCHAR	FERN VILLA 2 GAVIOTS ROAD	Information needed for the customer's record
POSTCODE	VARCHAR	SL9 8DP	Information needed for the customer's record
COUNTY	STRING	BUCKINGHAMSHIRE	Information needed for the customer's record
MOBILE NUMBER	INT	07821180416	Information needed for the customers record. Also, allows the customer to be contacted quickly regarding booking details
DOB	INT	24/01/1999	Information needed for the customers record. Also, means that insurance prices

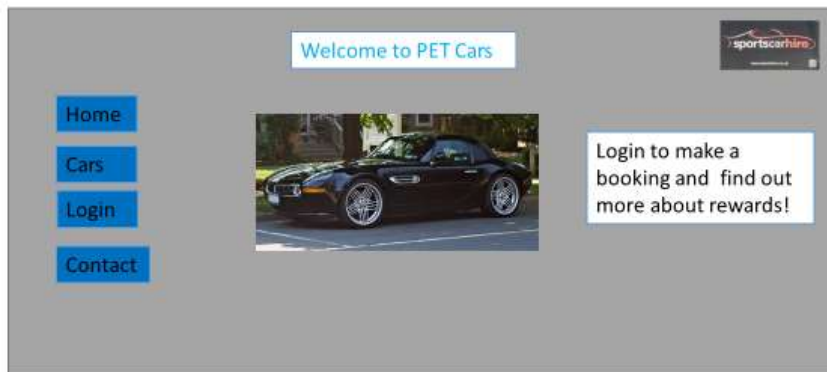
			can be calculated for the customer.
Pword_hash	STRING	2hbj2hb2jjiu2njnn4jn3kj	This will be the hash given to the input from the customer. This is to improve the security of the customers account
Email	STRING	BOB@gmail.com	Information needed for the customer reocrds

Admin Table

Field Name	Type	Example	Description
Username	STRING	Raees	A string used as part of verification of entering the admin rights part of the programme
Password	STRING	Administrator	Password related to the account of the admin user

The initial form ideas:

Initial idea for Home screen



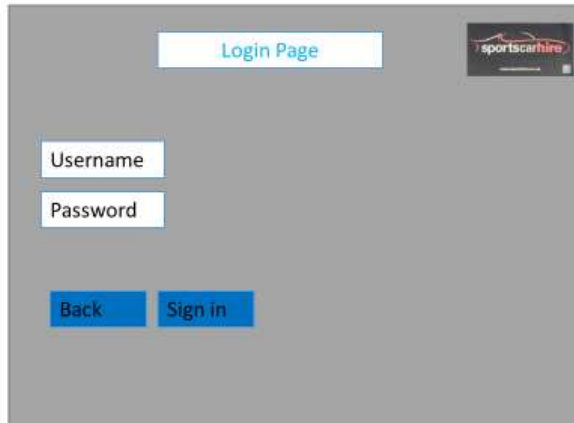
The home screen will display push buttons that will take you to the next window when clicked. When the Cars button is clicked, a new window will open where the user can view the different cars that are available. When the Login button is clicked, a new window will appear where the user can enter their email and password so they continue to make a booking. When the contact button is clicked, a new window will open where the user can view the contact details for the company.

Initial idea for Customer Details Form

A mockup of a customer details form. It has a grey background. At the top center, a white box contains the text 'Sign Up'. In the top right corner, there is a small logo for 'sportscarhire'. The form consists of several input fields arranged in two columns. The left column contains: 'F.Name', 'S.Name', 'DOB', 'Address', and 'Postcode'. The right column contains: 'Mobile No.', 'Copy of official ID document', 'Username', and 'Password'. At the bottom right, there are three blue buttons with white text: 'Sign Up', 'Next Page', and 'Previous Page'.

The Sign Up screen will ask the user to input the displayed information. All of this information will be saved to the "customer" table in the database. This means when they try to sign into the programme, the programme will check to see if they already have a username and password, which would have been stored in the database. This form will also act as collecting customer details.

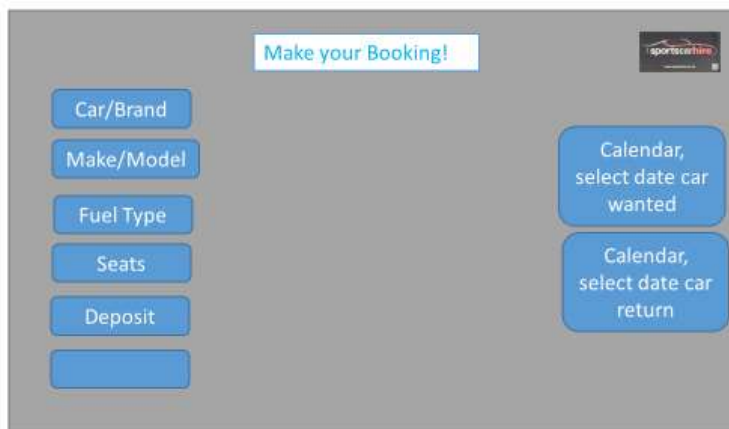
Initial idea for Loginscreen



The login screen has a grey background. At the top center is a white box with the text "Login Page" in blue. In the top right corner is a small logo for "sports car hire". Below the title, there are two white input boxes: "Username" and "Password". At the bottom left, there are two blue buttons: "Back" and "Sign in".

Once the user have entered their details on the previous page (Sign Up), they can then click the "sign up" button from the "home" screen. They will then be prompted to input their username and password for the programme and click the "login" button. The information entered will then be compared to the data saved from the customer table. If the information matches, then the user will be able to continue to the next page of the programme and they can make a booking. If the information does not match, then a message box will be shown "you have entered incorrect information, please try again"

Initial Idea for Booking Form



The booking form has a grey background. At the top center is a white box with the text "Make your Booking!" in blue. In the top right corner is a small logo for "sports car hire". On the left side, there are five blue buttons stacked vertically: "Car/Brand", "Make/Model", "Fuel Type", "Seats", and "Deposit". On the right side, there are two blue buttons stacked vertically: "Calendar, select date car wanted" and "Calendar, select date car return".

The booking form will allow the user to choose from the information displayed. Each box will be a drop down menu where the user can select from the different information.

The programme will then calculate the price of the rental depending on which car has been selected and how long the user wants the car for. The programme will display the total figure.

If they are happy with these details, they can continue on to finish their booking. However the user can also cancel their booking if they do not want to continue.

Initial idea for Contact Us Window

Here the user can view the contact details for the company if they require them. The "back" push button will return the user to the home page.

Different buttons the user can interact with, in the programme:

Button Name	What Action should it Complete
Sign up	When pressed, the push button should take you to the next page of the programme. This will allow the user to create an ID and then they will be able to sign into the programme.
Enter	When pressed, the push button should either take you to the next page of the programme. The contents of the input box will be checked against the database, if they match the user will be able to login into the programme.
Cars	When pressed, the push button should either take you to the next page of the programme. The page will display the different cars on offer by the company.
Contact	When pressed, the push button should either take you to the next page of the programme. The page will display any contact information the user may need (phone number, Email).
Back	When pressed, the push button should either take you to the previous page of the programme. This will allow for easy navigation through the programme.
Get CV	When pressed, the push button should either take you to the next page of the programme. A new window will open and display a CV form that the user can fill out.
Edit CV	When pressed, the push button should take you to the previous page of the window. This will allow the user to make changes to the programme.
Print	When pressed, the push button should display a new window where the user can select the printer they want to send the document to. The user can view a preview of what they want to print.

Different Inputs the user can interact with in the programme

Field (Input)	What should it do?	Stored Format	Example of What can't be entered
Login Box	The box should allow you to input text into the box. The box will be validated to prevent a certain amount of characters from being exceeded.	VARCHAR	U\$er/1
Password Box	The box should allow you to input text into the box. The box will be validated to prevent a certain amount of characters from being exceeded. The box will also hide any entered characters.	VARCHAR	"Password?#!"
Re- enter Password Box	The box should allow you to input text into the box. The box will be validated to prevent a certain amount of characters from being exceeded. The box will also hide any entered characters. The input of the box will be compared the input from the (password box). If the two match Bob will be able to use this password for his login, however, if they do not match, Bob will have to re-enter the passwords.		"pa55word?#!"
Name	The box should allow you to input text into the box. The box will be validated to prevent certain characters from being entered, as well as a limit on the amount of characters being entered.	TEXT	Bob123

Surname	The box should allow you to input text into the box. The box will be validated to prevent certain characters from being entered, as well as a limit on the amount of characters being entered.	TEXT	Rob123
Address	The box should allow you to input text into the box. The box will be.	VARCHAR	9 Gaviots road?!
County	The box should allow Bob to input the county they live in. this box is not a necessary requirement to fill out.	TEXT	Berk5hire
Postcode	The box should allow Bob to input the postcode they are living at.	VARCHAR	SL9 7YP?!
Email	The box should allow you to input text into the box. The box will be validated to check if the @ sign is in the address.	VARCHAR	Bobandrob.com
Phone Number	The box should allow you to input text into the box. The box will be validated to prevent a certain amount of characters from being exceeded (11). Also, only numbers will be allowed to be entered. Furthermore, only numbers starting with (07) can be used.	INTEGER	82190412
DOB	The box should allow you to enter your date of birth in the DD/MM/YY format. Validation will be used to prevent data that	INTEGER	34/13/27

	does not fit the parameters of DOB		
Experience	The box should allow you to input text into the box.	TEXT	
Previous employment	The box should allow you to input text into the box.	TEXT	
Hobbies	The box should allow you to input text into the box.	TEXT	
Qualifications	The box should allow you to input text into the box.	TEXT	
Calendar	When pressed, the calendar will register the date that Bob has pressed. This input will be needed to work out the cost of hiring a car for the days selected.	Text	A price being calculated for hiring a car before the current date.

Test Plan

The test plan I am creating is so that I can test all the different aspects of the programme after each class has been created. This is so that I can check that each function is working the way it supposed to and so that the programme does not crash. This plan will be used in the code development section of the documentation whereby I can test the different functions of the programme.

The first stage in testing is when I will be the one who enters the details and checks the results. This is called alpha testing

The second step in testing is when I show the finished classes to the stakeholder (Godwin) and he goes through and checks each class for usability. This is called beta testing.

Login:

Test Number	Test Action	When will this be Tested
1	Start the programme	Throughout development
2	Click each button on the login screen and see if the different windows open	End of Development
3	Click the enter button without entering data into the programme	End of Development
4	Click the enter button when only the username box has been filled out correctly	End of Development
6	Click the enter button when only the password box has been filled out with incorrect details	End of Development
8	Click the enter button when both the username and password box have been filled out with correct details	Throughout development
9	Click the enter button when both the username and password box have been filled out with incorrect details	End of Development
10	Click the enter button multiple times quickly to see if the programme crashes	End of Development
11	Click the sign up button	Throughout development
12	Click any message boxes that appear when I try and login to the programme	Throughout development
13	Enter extreme details into the username and password box	End of Development

Admin:

Test Number	Test Action	When will this be Tested
-------------	-------------	--------------------------

1	Enter the admin username and password into the programme	Throughout development
2	Enter only the admin username into the text entry	End of Development
3	Enter incorrect admin details into the username and password text entry	End of Development
4	Click "OK" on any message boxes that appear	Throughout development
5	Click "database" button on the admin window When the window opens scroll through the records within the table	Throughout development
6	Try to change data within the table	Throughout development
7	Enter extreme details into the cells	End of Development
8	Enter null data into the cells	End of Development
9	Press the update button	Throughout development
10	Check to see whether the changes have been made in the correct table in the database	Throughout development
11	Click a cell and check to see whether the programme displays which cell a user is currently choosing to edit	End of Development
12	Click multiple cells quickly	End of Development
13	Click the "back" button on the table view window	Throughout development
14	Hover over the chart to see if each "slice" enlarges so it can be viewed easier	End of Development
15	Click the "back" button on the graphs window	Throughout development

16	Click the “back” button quickly on the graphs window	End of Development
17	Click the “back” button on the admin window	Throughout development

Sign Up:

Test Number	Test Action	When will this be Tested
1	Click the sign up button	Throughout development
2	See if the “sign up” window opens correctly	Throughout development
3	Enter data into the text entries	Throughout development
4	Press the “sign up” button	Throughout development
5	Check to see whether the data has been saved to the correct table in the database Check to see whether the auto increment function is working properly	End of Development
6	Do not enter details into the text entries and click the “sign up” button	End of Development
7	Check to see what data has entered the database	End of Development
8	Enter some data into the entries and leave other blank	End of Development
9	Enter extreme details into the text entries	End of Development
10	Click the “back” button on the “sign up” window	Throughout development

Cars:

Test Number	Test Action	When will this be Tested
1	Click the “cars” button to check whether the window opens	End of Development
2	Check to see whether all the cars that the programme offers are on display	End of Development
3	Check to see if the “back” button works	End of Development

Contact us:

Test Number	Test Action	When will this be Tested
1	Click the “contact us” button	End of Development
2	Check whether the relevant details are being displayed on the contact window	End of Development
3	Click the “back” button	End of Development
4	Click the “cv” button	End of Development

Curriculum Vitae:

Test Number	Test Action	When will this be Tested
1	Click the “cv” button	Throughout development
2	Click the “home” button	Throughout development
3	Enter text into the text entries	Throughout development
4	Enter extreme data into the text entries	End of Development
5	Enter text in some text entries and leave other blank	End of Development
6	Press the “show me cv” button	Throughout development

Printing CV:

Test Number	Test Action	When will this be Tested
1	Check to see whether the correct details are being displayed on the window	End of Development
2	Check to see if the formatting of the data on the window is correct	Throughout development
3	Press the "edit" cv button	Throughout development
4	Press the "print" button	Throughout development
5	Check to see whether the printing window opens	Throughout development
6	Check to see whether I can click a printer I want to send the document to	End of Development
7	Click a printer which I know the document cannot be sent to	End of Development
8	Check to see whether I can change the preferences	End of Development
9	Check to see whether I can find more printers	End of Development
10	Click the "cancel" button	Throughout development
11	Click the "Print" button	End of Development
12	Check whether a preview is created of the document	End of Development
13		

Rental Calculation:

Test Number	Test Action	When will this be Tested
1	Check to see if the rental calculator window opens when the user enters the correct login details	End of Development
2	Check to see if the rental calculator window opens when incorrect details are entered	End of Development

3	Check to see if the drop down list displays each car	End of Development
4	Check to see if I can select each car from the drop-down list	End of Development
5	Check to see whether the calendar works when clicked	Throughout development
6	Check to see whether I can click a date before today's date	End of Development
7	Select a car from the list and click a date from the calendar	Throughout development
8	Click finish	Throughout development
9	Check to see whether the correct price is displayed	End of Development
10	Select a car from the drop down list and select a date before today's date	End of Development
11	Click finish button to see what happens	End of Development
12	Click the "cancel" button	Throughout development
13	Click the "finish" button to see whether the booking window will open	Throughout development
14	Click the "finish" button and check whether a message box appears	Throughout development
15	Click the "finish" button to check to see whether a user can make a booking if they are above the age of 21	End of Development
16	Click the "finish" button to make a booking when the user is below the age of 21	End of Development
17	Check to see if a message box appears when the user makes	End of Development

	a booking but they are below the age of 21	
18	Check to see if a message box opens when the user makes a booking and they are above the age of 21	End of Development
19	Check to see if the information entered by the user is saved to the database	End of Development
20	Check to see whether information from another table in the database, has been saved to the bookings table in the database	End of Development
21	Check to see whether data has been saved to the database if the user is underage and can't make a booking	End of Development

Booking:

Test Number	Test Action	When will this be Tested
1	Check to see if the data made from the booking is the same as the data shown on the final form	End of Development
2	Check to see if the "print" button	Throughout development
3	Check to see if the "home" button works	Throughout development

Forgot password:

Test Number	Test Action	When will this be Tested
1	Check to see if the new window opens when the "forgot password" button is clicked	Throughout development
2	Check to see if the "cancel" button works on the new window	Throughout development

3	Check to see if text can be entered the "enter email box"	Throughout development
4	Enter extreme data in	End of Development
5	Enter an invalid email	End of Development
6	Click the "next" button	Throughout development
7	Check and see if the new window opens	Throughout development
8	Enter a valid email in	Throughout development
9	Click the next button	Throughout development
10	Check and see if the email was received	End of Development
11	Check to see whether the code appears random	End of Development
12	Check to see if the code has letters and numbers in it	End of Development
13	Check to see whether the email sent has a subject and sender	End of Development
14	Check to see whether the message sent in the email has a professional looking format	End of Development
15	Resend an email to a valid email	End of Development
16	Check to see whether the code received is different from last time	End of Development
17	Copy the code into the programme	Throughout development
18	Click the "next" button	Throughout development
19	Check to see if a new window opens	Throughout development
20	Enter data into the "new password" box	Throughout development

21	Check and see if the new password has been updated into the database	End of Development
----	--	--------------------

What can I Test Before the Programme is Complete?

Before the programme is complete I will be able to test whether each form that should open, will open. This is to check whether the forms look professional or if I need to make edits to the design of the forms before they are presented to the Godwin. This is also to test the usability of the forms, whether the size of the text boxes is large enough and whether any words on the form can be read by a customer.

Code Development

Before I tried to import any modules or import any forms that had been created using PyQt, I decided to write the main part of the login class hardcoded to see if what I had in mind could be incorporated into the programme. Since this was only the begging of the programme and I was simply testing some code, I decided not to fetch data from the database, and instead hardcode any information I might need into the programme.

```
ask_admin=input("are you and Admin (Y/N):")
if ask_admin=="N":
    username=input("enter your username here:")
    print(a)
    password=input("enter your password here:")
else:
    ask_admin=="Y"|
    admin_uname=input("enter the admin username")
```

This screen shot shows the first bit of code I wrote for the programme. It is asking the user to state whether or not they are an admin. Depending on their answer, they will then have the option to enter the admin username and password or the customer username and password.

```
>>>
are you and Admin (Y/N):N
enter your username here:|
```

```
are you and Admin (Y/N):Y
enter the admin username:|
```

Once this had been done, the next part was to check to see if the user could login with the correct details and what would happen if they tried to login with the wrong details.

```
customer="bob"
customer_password="hello"
```

The first step was to define 2 variables, one for the user's name and the other for their password.

```
if username==customer:
    if password==customer_password:
        print("hello bob, welcome to your account")
else:
    print("wrong details")
```

The next part was to create an IF statement that would check to see whether the input from the user matched the details the programme already had. If they did match, the user would see a message saying "hello __, welcome to your account".

```
>>>
are you and Admin (Y/N):N
enter your username here:bob
enter your password here:hello
hello bob, welcome to your account
>>> |
```

If the details did not match then the user would see a message saying "wrong details")


```
>>>
are you and Admin (Y/N):N
enter your username here:notbob
enter your password here:bye
wrong details
>>> |
```

A similar procedure needed to be carried out for the admin as well

```
are you and Admin (Y/N):Y
enter the admin username:admin
enter the admin password:raees
welcome admin, what would you like to view?
>>> |
```

```
>>>
are you and Admin (Y/N):Y
enter the admin username:notadmin
enter the admin password:bye
wrong details
>>> |
```

Now that I had the basis of my login class working, I decided to move on and import the functions I would need for the programme. This is because I would now implement what I had just done, hardcode, with the forms I had created using PyQt and data from the database which had been created using SQL

Initialisation and importing Modules

Stage 1:

```
import sys
import os
from PyQt4 import QtCore, QtGui, uic
import sqlite3 as db
```

This code was to initialise the modules that I would need to be able to create the programme.

Within each window that has been created, I have placed “next”, “back” or “cancel” buttons on each window to provide easy navigation through the programme for any user.

```
def go_home(self):  
    forgot_pword_gui.hide()  
    login_gui.show()
```

This code shows 1 example of this has been achieved.

Login Screen

Stage 1:

The first step I took was to initialise the class I would be using for the login modules.

```
class login_class(QtGui.QMainWindow, login_window):  
    def __init__(self, parent=None):  
        QtGui.QMainWindow.__init__(self, parent)  
        self.setupUi(self)
```

```
login_gui = login_class(None)  
login_gui.show()
```

Once, this was done and window opened, I then started to create the different variables I would be using throughout the programme.

```
self.uname=self.enterstaffloginbox.text()  
self.password=self.enterpasswordbox.text()  
print(self.uname, self.password)
```

This code was checking that username that Bob had input into the textbox matched username BOB had previously created. Similarly, when Bob inputs his password, the programme is checking to see if they match with the one they have previously created.

```
QString=type("")
self.uname=str("")
self.password=""
self.signup_uname=str("")
self.signup_pword=""
```

Initially this code was creating variables that would save the data that Bob would enter. For example, when Bob would enter his username, it would be saved to the self.signup_uname variable name. This was because at this stage I had not yet implemented the GUI that would take the place of the variable names. However, since this was the beginning of the programme, it was ok to use so that I could continue with other aspects.

Stage 2:

When the user presses the "cars" button, a new window will open where the user can see the choice of cars the company has to offer. From here they can get an idea as to which one they may want to rent.

When the user presses the "contact" button, a new window will open where they can view the company's contact information (number, address)



User enters their username into the box if they already have one.

User enters their password into the box if they already have one.

When the user clicks enter, their input will be checked against the database to see if it matches the criteria in the table.

If the user does not have a username or password, they can click the "sign up" button. This will open a new window where they can enter their details.

```
Traceback (most recent call last):
  File "C:\Users\Raees Q\Documents\6 Form\Computing\Coursework\PycharmProjects\Coursework\PROGRAMME.py", line 146, in <module>
    login_gui = login_class(None)
  NameError: name 'login_class' is not defined
```

Initially when I tried to run the programme I ran into this error. I soon realised it was because the class had not yet been initialised and I hadn't said what window the programme needed to use for the class.

The next step I took was to create the window and initialise the class.

```
login_window = uic.loadUiType("Coursework GUI Login Page.ui")[0]

login_gui = login_class(None)
login_gui.show()

class login_class(QtGui.QMainWindow, login_window):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
```

This was the code to initialise the class.

This code is so that the window will be visible when the programme is loaded

The next step was to link the login window created in PyQt (an IDE tool, used to create different forms) to the programme in python. This is so that Bob would be able to fill out the necessary details in the programme to login (username box, password box).

I also needed to link the buttons which would be displayed on the login screen to carry out an instruction when clicked

```
self.enterbtn.clicked.connect(self.login)
```

This code here is so that when the "enter" button is clicked on the page, the "login" function is run. When the button is pressed and the information entered is correct, then the login window will be hidden and the sign up window will be shown.

```
def open_signup_page(self):
    self.hide()
    signup_gui.show()
```

The reason we must use "self" is for encapsulation. This is so that the variables can be called in different classes in order to make efficient use of memory in other functions. It is a good programming technique.

Stage 3:

```

CURRENT PROGRAMME.py - C:\Users\Raees Q\Documents\6 Form\Computing\Coursework\PycharmProjects\Coursework\CURR...
File Edit Format Run Options Window Help

def login(self):
    if self.enterstaffloginbox.text() == admin_username:
        if self.enterpasswordbox.text() == admin_password:
            self.open_admin()

    ##
    self.line = enterstaffloginbox()
    regex = QtCore.QRegExp("[a-z-A-Z_]+")
    validator = QtGui.QRegExpValidator(regex)
    self.line.setValidator(validator)

    else:
        self.uname = self.enterstaffloginbox.text()
        self.password = self.enterpasswordbox.text()
        print(self.uname, self.password)

        username_list = []
        contents = db.connect("Rental System.sqlite")
        cursor = contents.cursor()
        # cursor.execute("SELECT F_Name, Password FROM Staff WHERE F_Name=?", (self.uname,))
        cursor.execute("SELECT Username FROM Customer WHERE Username=?", (self.uname,))
        data = cursor.fetchall()
        # print("The data in cursor, which is all the usernames that match the entered username:")
        # data

        for each in data:
            username_list.append(each)
            print(username_list)

        username_found = False

        for each in username_list:
            print(str(each))
            if each[0] == self.uname:
                QtGui.QMessageBox.information(self, "Login Status", "Correct")


                username_found = True
                break



        if username_found == False:
            QtGui.QMessageBox.critical(self, "Login Status", "Incorrect, Try Again")

        if username_found == True:

```

This is the complete login function, where the programme will Bobs' inputs (username, password). If username and password match the ones previously stored in the database, Bob will be able to login and continue to the next screen. The programme is selecting all usernames from the database and checking if usernames match the one Bob had inputted. If usernames and passwords match, Bob will be displayed with a message box saying his "login status" is "correct".

What am I Testing	How is it tested	Was it successful
Does the login screen load when the programme is run	I will run the programme. The programme should load the login screen where Bob can input their details.	Successful- 

Does the message box appear when login is successful	I will run the programme and login in with the correct username and password	Successful- 
Does the message box appear when the login is unsuccessful	I will run the programme and login in with incorrect details	Successful- 
Does each button work when it is pressed	I will check this by clicking on each button on the window and check to see whether the correct function is carried out when this happens	Successful

Once I created the login module, I showed it to Godwin, one of the stakeholders from PremiumExecutiveCars. It covered the aspects they were looking for. However, after the meeting the feedback he gave me was that the login screen needed to look more professional and complete. This is because this would be the first screen Bob would see when he starts the programme, it also conveys a message to Bob about what the company is about. He suggested I should add some colour, images and better navigation (buttons) around the programme. Therefore, I decided to improve the visual aspects of the login screen.




This is now how the login screen looks like:



I showed Godwin the new and improved design of the login screen and he was happy. All aspects of the login screen still work (login box, password box) each button will now need to be linked to a new window. This will be the next stage of development. Furthermore, they also added that they would like to have control over the data being entered into the programme, which would not have been available to “normal” users of the programme (administrator privileges). An example that Godwin gave for needing this feature was because a user may need to change their data which they entered previously, either due to it being incorrect or a change in circumstance. They also wanted the programme to generate a graph of the most popular cars that are being rented through the programme. The administrator could login in through the same way “normal” users would. The additional buttons on the page also mean that I have created new windows for each button. I also

added a feature that would “hide” any password that would be typed in the (password_box). This is to add an extra layer of protection when Bob is entering his information.

Testing after new Login form has been created

What am I testing	How will I test this	Was it successful;
Does the programme login when the username is correct but you have an incorrect password	I will check this by entering a correct username that is already in the database but no password and check to see whether the programme will continue to the next form.	Failed-  <p>This should not have happened</p>
Does the programme accept a login if the user has entered a correct user name but incorrect password	I will check this by entering a correct username but incorrect password. The programme should not continue to the next form.	Failed-  <p>The programme should not have accepted the incorrect password</p>
Does the programme accept and incorrect username and no password/wrong password	I will check this by entering an incorrect username that isn't in the database already. The programme should not continue to the next form	Pass- 
Does the programme crash if the buttons are pressed to many times	Does the programme crash if the buttons are pressed to many times	Pass, the programme did not crash

From the further testing, I carried out on the new login window, I noticed some errors. The programme would allow a user to login if they did not have the correct password/no password, but they did have the correct username. However, if they did not have a correct username, then they would not be able to login into the programme. From the testing I can deduce that the programme, is checking to see whether the username entered, matches a username that is already in the database, instead of checking to see whether the passwords match.

This piece of code is where the error lies

```
for each in data:
    username_list.append(each)
print(username_list)

username_found = False

for each in username_list:
    print(str(each))
    if each[0]==self.uname:
        QtGui.QMessageBox.information(self, "Login Status", "Correct")

        username_found = True
        break
if username_found == False:
    QtGui.QMessageBox.critical(self, "Login Status", "Incorrect, Try Again")

if username_found== True:
```

Stage 4:

To fix this error I went back to the login class and changed some code.

The first step I took was to create a Boolean value for whether login was successful or not, after which I then created a variable which would save what Bob would enter into the username text box, similar to what I had done before. I also connected to the database.

```
login_successful=False
self.uname=self.enterstaffloginbox.text()
contents= db.connect("Rental System.sqlite")
cursor=contents.cursor()
```

Stage 5:

The next step I did was to write another SQL query which selected all the usernames from my customer table in the database.

```
cursor.execute("""SELECT Username FROM Customer""")
data=cursor.fetchall()
```

Stage 6:

The next step was to create a loop which would cycle through the different usernames from the "customer" table and checked if any matched the input Bob may have entered. If the usernames

matched, then the programme should create a new variable that would save the password that Bob would enter. However, the variable would be overwritten with the hash of that password. This value will later be used in the programme. Once this had been done a new SQL query was written to select all the hashed passwords from the customer table within the database, however the password that would be selected would be the one that matches the username that Bob has entered.

```
for name in data:
    if name[0]==self.uname:
        self.pword=self.enterpasswordbox.text()
        self.pword=hashing(self.pword)
        cursor.execute("""SELECT Pword_hash FROM Customer WHERE Username='%s'"""%self.uname)
        result=cursor.fetchall()
```

Stage 7:

```
for password in result:
    if password[0]==self.pword:
        login_successful= True
```

This piece of code is showing another loop which is going through the hashed passwords within the database. If the hashed passwords in the database, match the one entered by Bob, which the programme has then saved as a hashed result, the Boolean Value for the login will then be changed to "True" and the user can continue to the booking form.

Stage 8:

```
if login_successful=False:
    QtGui.QMessageBox.critical(self, "Status", "Incorrect Details")
```

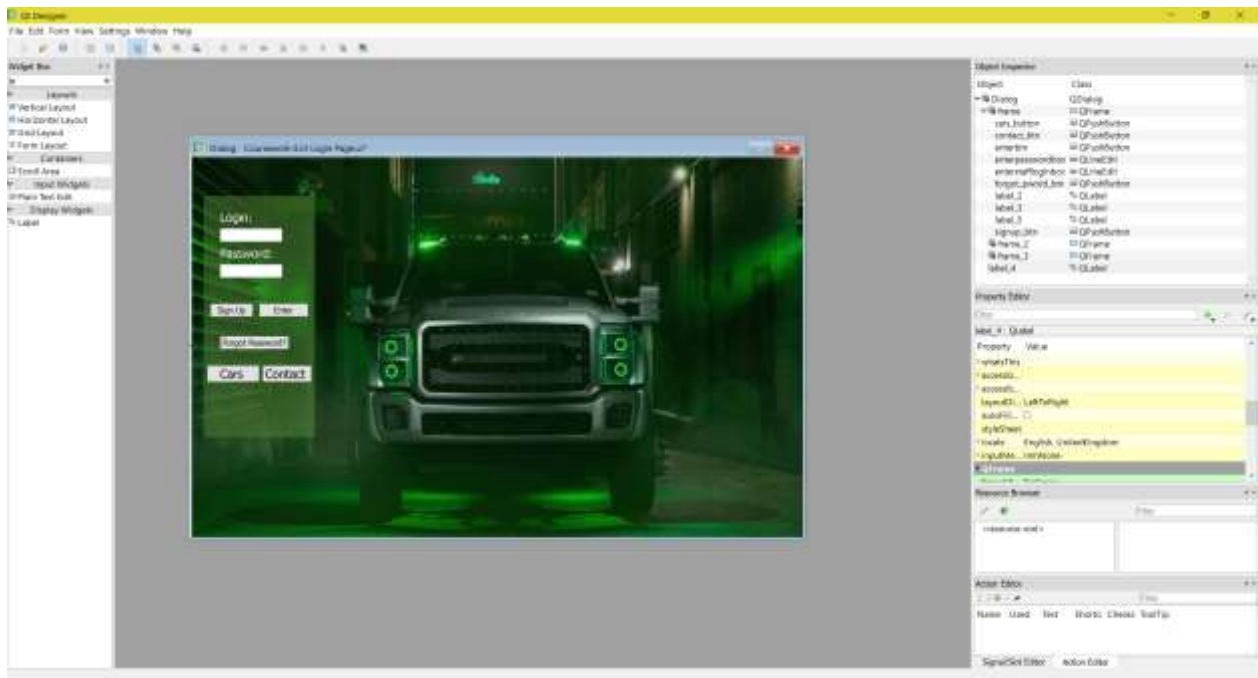
If Bob has entered the incorrect details and tries to sign in, then the user will be shown a message box that will display a message saying "incorrect details have been entered".

```
else:
    QtGui.QMessageBox.information(self, "Status", "Information entered correct")
    self.open_booking_form()
```

If the Bob has the correct information, then a message box will appear saying "information entered correctly" and he will be able to continue to the next form.

Stage 9:

This screen shot shows that I am using PyQt to create the forms that I will need to use to collect the data from Bob. I have implemented a label in which I have put a picture (PNJ file) to use as the background. After that I created a new label where I shifted the transparency to get a layering effect, therefore one box is lighter than the other. Once this had been finished, I then created the buttons that the Bob would click to navigate through the programme.



Administrator- creation and table view:

Stage1:

```
#####
#####
admin_username="admin"
admin_password="raees"
#####
```

To begin with, I declared the variables that would later be needed to login into the programme. It did not matter that it was hardcoded in as for the time being, it was just being used to check whether the function would work properly.

Stage 2:

The next step was to go back to the "login" function.

```
def login(self):
    if self.enterstaffloginbox.text()==admin_username:
        if self.enterpasswordbox.text()==admin_password:
            self.open_admin()
```

Example of Hungarian
Notation

This piece of code is showing how the programme is taking the data which has been entered into the textbox's on the login window (username_textbox,password_textbox) and checking to see if it matches the data which has been entered previously into the defined variable

(admin_username,admin_password). If the data which has entered matches, then the administrator can continue through to the next stage of the programme where they can edit data entered by the customer.

Furthermore, the screen shot refers to the text box as “enterstaffloginbox”. This is an example of Hungarian notation. I have used Hungarian notation to make my code easier to understand, if someone is reading the code, then they know that I am saving the data from the staffloginbox to a new variable. I have used Hungarian notation throughout my programme.

Stage 3:

The next step was to load the admin window, where the administrator could select which option they wanted to select.

```
admin_window= uic.loadUiType("Coursework GUI Admin Form.ui")[0]

class admin_class(QtGui.QMainWindow, admin_window):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
```

Once, this had been done, the next step was to create the functions that would be part of the administrator rights, (the ability to change the data inputted by Bob, being able to view a graph of the most popular cars rented).

```
def open_table_view(self):
    admin_gui.hide()
    table_view_gui.show()

def open_graphs(self):
    admin_gui.hide()
    graph_gui.show()
```

Stage 4:

To begin with I created the table_view_class which would give the administrator the ability to change the data which was in the “customer” table within the database. This also meant I would need to create the table view window GUI so Bob could select which data they wanted to edit.

```
class table_view_class(QtGui.QMainWindow, customer_table_view):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.data=[]
```

Stage 5:

The next step was to connect to the database and select where the programme should be getting the data from so it could be viewed in the table view.

```
def load_data(self):  
    con = db.connect('Rental System.sqlite') #opens a file  
    cur = con.cursor() #keeps track the "tape" position  
  
    customer_display="""SELECT * from Customer""" #selecting all data within the Customer table  
    cur.execute(customer_display)  
    #run the query  
    self.data = cur.fetchall() #query data comes back
```

Once this had been done, I would now be able to view the data which was in the table in the table view within the programme. However, I found it difficult to know which piece of I would be editing. Therefore, the next step was to create a function which would display to Bob which field they were editing. This is an example of polymorphism. I have previously used this code in the login class where the customer could login into the programme. I have used the same code but made some edits to it so that it will work in the admin class.

Stage 6:

```
def show_selected(self):  
    index = self.tableView.currentIndex() #returns currently selected cell  
    r=index.row() #returns the row of the currently selected cell  
    c=index.column() #returns the column of the currently selected cell  
    cell_contents=index.data() #returns the contents of the currently selected cell  
    #print("index,r,c",r,c,cell_contents)  
    self.lbl_data.setText(cell_contents)  
    self.lbl_row.setText(str(r) )  
    self.lbl_col.setText(str(c) )
```

This code is showing that when Bob selects the field they want to edit, it will be displayed on the window. Moreover, the programme will also display what Bob is editing within that cell. This will make the programme easier to use and reduce the chance of Bob editing information incorrectly.

Stage 7:

This stage was making use of the "UPDATE" query in SQL so that previous data could be changed from the programme.

```
def upd_record(self):
```



```

id=self.model.data(self.model.index(index.row(), 0))
print(id,self.model.data(self.model.index(index.row(), 1))
name=self.model.data(self.model.index(index.row(), 1))
test1=self.model.data(self.model.index(index.row(), 2))
test2=self.model.data(self.model.index(index.row(), 3))
test3=self.model.data(self.model.index(index.row(), 4))
test4=self.model.data(self.model.index(index.row(), 5))
test5=self.model.data(self.model.index(index.row(), 6))
test6=self.model.data(self.model.index(index.row(), 7))
test7=self.model.data(self.model.index(index.row(), 8))
test8=self.model.data(self.model.index(index.row(), 9))
test9=self.model.data(self.model.index(index.row(), 10))
test10=self.model.data(self.model.index(index.row(), 11))

```

This code is showing how all 11 rows within the “customer” table are saved to new variables so they can be edited within the programme.


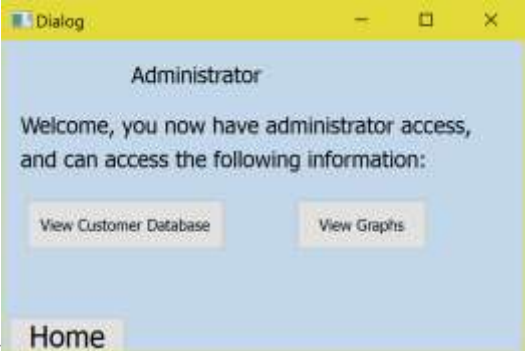
```

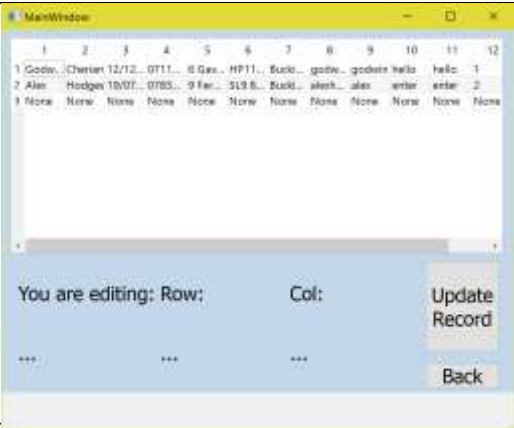
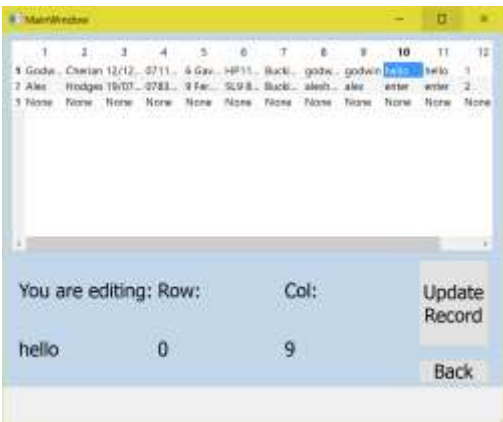
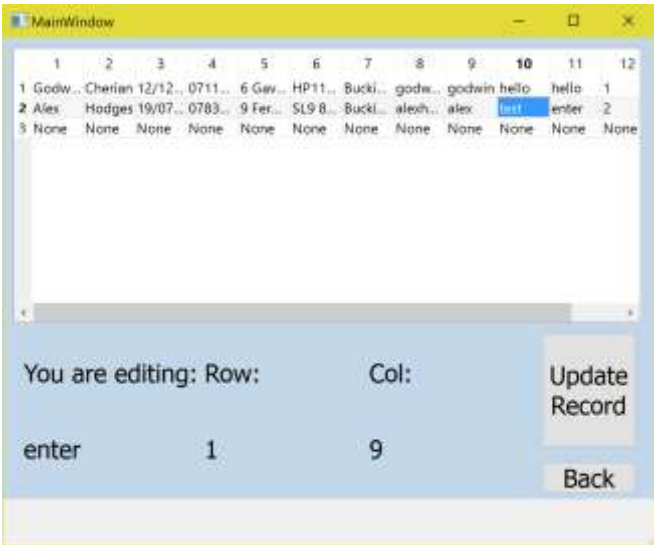
print(id,name,test1,test2, test3)
self.model.data(self.model.index(4,1))
con = db.connect('Rental System.sqlite')#opens a file
cur = con.cursor()#keeps track of the "tape" position
#parameter query - get criteria from Python
cur.execute("""UPDATE Customer SET S_Name='%s',DOB='%s',Mobile_No='%s',
Address='%s',Postcode='%s',County='%s',Email='%s',Username='%s',Password='%s',
V_Password='%s',CustomerID='%s'WHERE F_Name='%s'"""%(name,test1,test2,test3,test4,test5,test6,test7,test8,test9,test10,id))

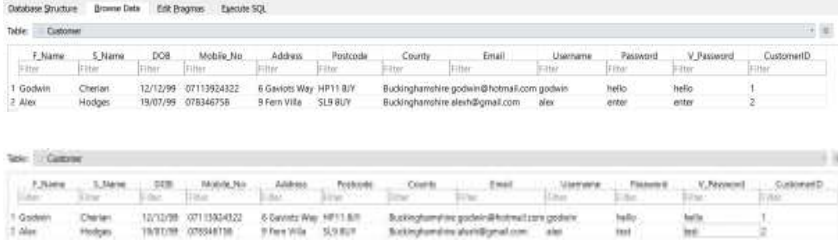
```

This piece of code is showing how the SQL query is implemented into the programme. Once the programme has been connected to the database, each column within the “customer” table can now be changed. To do this the “UPDATE” query was used.

Stage 8:

What am I Testing	How will I test it	Was is successful
Does the admin screen load	I will test this by loading the programme and then try to log into the programme with the admin username and password.	 <p>Successful-</p> 

Once the admin screen has loaded, does the customer table view load in the programme once the button has been clicked.	I will test this by pressing the "View Customer Database" and see whether the next window loads	 <p>Successful-</p>
When the table view window has loaded, will the data being edited be highlighted and displayed in the bottom left hand colour of Bob	I will test this by selecting a record from the table. If it is successful, then the information within the record will be displayed in the bottom left of the window.	 <p>Successful-</p>
Once the data has been selected, will the new data be displayed in the table view	I will test this by entering new data into the highlighted cell.	 <p>Successful-</p>
Has the data been updated in SQL	I will test this by opening SQL and checking	Successful- this screen shot shows the database before the update statement. Looking at the password column, you can see that the data entered is (hello, enter)

	whether the new data has been entered into the database	 <p>This new screen shot shows that the database ha been updated with the contents of the password cell having been changed to “test” Therefore, I can conclude that the update statement works as it should.</p>
Does each button displayed on the windows work when clicked	I will test this by selecting each button on the window. Depending on the button it should either open a new window or carry out a different function, such as completing an update request	Successful

Administrator- HTML graphs

Stage 1:

The final step to the admin function was to be able to display a HTML graph of the most popular cars being rented from the company. to do this I fist initialised the class I would be using to create the graph.

```
class graphs_class(QtGui.QMainWindow, graphs_window):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
```

Stage 2:

The next step was to create a window where the graph would be displayed. Once this had been done I needed to connect to the web view so Bob could actually see the graph.

```
self.webView.setHtml
```

After this had been done, I needed to initialise the HTML code, in a similar way the python modules needed to be initialised at the beginning of the programme. HTML is responsible for the creation of the graph. Once this was completed I could use the CSS code which would allow me to edit the visual aspects of the code, such as colour, size of text, size of graph and titles.

Stage 3:

```
</body>
<script>
Highcharts.chart('container', {
  chart: {
    plotBackgroundColor: null,
    plotBorderWidth: null,
    plotShadow: false,
    type: 'pie'
  },
  title: {
    text: 'Popularity of Cars Rented'
  }
});
```

This piece of code is the start of the graph creation. I have titled the graph “Popularity of Cars Rented”. At the same time, I decided that a pie chart would be the best way to display the information to the admin user as they can interpret the information easier than if the information was displayed on a line graph. Furthermore, I decided that it would be best to have a clean background, which is why I chose white or “null”.


```

        tooltip: {
            pointFormat: '{series.name}: <b>{point.percentage:.1f}%</b>'
        },
        plotOptions: {
            pie: {
                allowPointSelect: true,
                cursor: 'pointer',
                dataLabels: {
                    enabled: true,
                    format: '<b>{point.name}</b>: {point.percentage:.1f} %',
                    style: {
                        color: (Highcharts.theme && Highcharts.theme.contrastTextColor) || 'black'
                    }
                }
            }
        },
        series: [{
            name: 'Makes',
            colorByPoint: true,
            data: [{
                name: 'AUDI',
                y: 56.33
            }, {
                name: 'Ferrari',
                y: 24.03,
                sliced: true,
                selected: true
            }, {
                name: 'McLaren',
                y: 10.38
            }, {
                name: 'Mercedes',
                y: 4.77
            }, {
                name: 'Porsche',
                y: 0.91
            }, {
                name: 'Rolls Royce',
                y: 0.2
            }
        ]
    }
]);
</script>
</html>

===>

```

This part of the code is creating the different “section” or “slices” of the graph. For the time being there was no need for the graph to be displaying the information from the database as the booking function had not yet been completed. Therefore, to provide test data to show to Godwins, I decided to put several different cars into the chart and define how large each “slice” would look like.

Stage 4:

The last step was to go back and link the admin window to the graphs window, so that when a button is clicked, the graphs window will open.

```
def open_table_view(self):
    admin_gui.hide()
    table_view_gui.show()
```

What am I testing	How will I test it	Was it successful
Does the admin window open when it is loaded	I will enter the admin details (username,password) into Bobname and password box on the login window	Successful
Does the graphs window open when it is clicked	I will test this by clicking the "graphs" button on the admin window.	Successful
Once the window is open, is the graph displayed on the window	I will test this by loading the window. I should then see a graph displayed on the window	Successful

Once the admin function had been completed, I showed the class to Godwin. He was pleased that each window looked professional and complete. He was also pleased with how easy to navigate it was between each window. However, he did want the way to login to the admin function to change. This was because he said different admins may need to be assigned and so they would wish to store this information in a database. Therefore, I would need to go back and edit the code, so the admin information would be fetched from the database. Secondly, he said they would wish to see the information displayed on the graph coming from the bookings table, instead of how it was now (being written into the code). I did explain them that now this could not have been done as the bookings function hadn't been written. He was understanding.

Testing Admin Function (2)

After further testing of the programme, I noticed that I needed fix the way that a stake holder (Godwin) would login to the programme. This is because I had "hard coded" the login information into the programme. However, that would mean only I would know the correct username and password to access the admin features. Moreover, if anyone else needed to become an admin they wouldn't be able to. Therefore, I decided to go back to where I created the code to login as admin and change the way the programme read the information.

Stage 1:

```
admin=False
```

- Step 1 was to create a Boolean value for the Admin feature.

Stage 2:

```
cursor.execute("""SELECT Username, Password FROM Admin""")
admin_combo=cursor.fetchall()
```

I then created a new SQL statement which would select the username and password from the Admin table in the database. The results were then stored to a new variable.


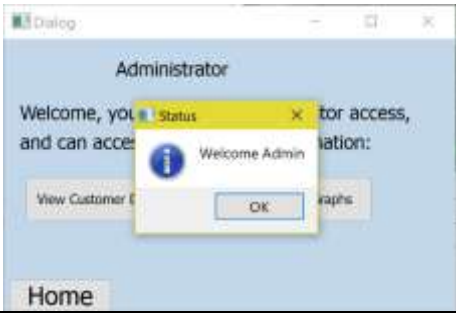
Stage 3:


```
combination = (self.uname, self.pword)
print(combination)
if admin_combo[0]==combination:
    admin=True
    self.open_admin()
```

This IF statement is making sure that the information entered by a stake holder (Godwin) matches the information already Admin table for the database. If the information entered by Godwin matches the one in the database, then the admin window will open for Godwin.

```
QtGui.QMessageBox.information(self, "Status", "Welcome Admin")
```

This message box will appear if Godwin enters the correct admin information. If he does not then the admin window will not open.

What am I Testing	How will I Test it	Was is Successful
Does the admin window open with the correct username and password	I will enter the correct username and password for the admin into the programme	Successful-  
Does the admin window open with	I will test this by entering incorrect	Successful-

the incorrect information entered	information into the programme	

Contact Us

From the research, I carried out on previous companies at the start of the project, every single company had a page where Bob could contact the company. Therefore, I decided that next step I should take would be to create a “Contact Us” window.

Stage 1:

The first step to do this was to initialise the window. Once this had been done the next step was to create the window. Since I had made many previous windows for the stake holders, and at each meeting they told me exactly what they were looking for, I had a good understanding of how it should look. However, I thought PremiumExectuiveTravel needed a way to differentiate itself from the rest of their competitors. Therefore, I decided it would be a good idea to implement a “CV” function, where a user could click a button and be sent to a new window where they could fill out the required details to apply for a job at the company. This would have been a useful feature as many people often look at an employer’s website when they are considering to fill out an application. This way, if a user wanted to join the company, since it was expanding at a fast rate, they could find what they need directly form the website.

```
class contact_us_class(QtGui.QMainWindow, contact_us_window):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.back_btn.clicked.connect(self.go_back)
        self.cv_btn.clicked.connect(self.open_cv)

    def go_back(self):
        contact_gui.hide()
        login_gui.show()

    def open_cv(self):
        contact_gui.hide()
        cv_gui.show()
```

All the necessary information that a user may need to contact the company can be found on this window.

Stage 2:

This step was creating the window where Bob could fill out the fields would be asked on the “CV form”. To do this I created a new class and a new window.

```
class cv_class(QtGui.QMainWindow, cv_window):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)

        self.cv_btn.clicked.connect(self.openCV)
        self.home_btn.clicked.connect(self.go_back)
    def openCV(self):
        self.name=self.lineEdit.text()
        self.pexperience=self.lineEdit_5.text()
        self.pemployment=self.lineEdit_4.text()
        self.hobbies=self.lineEdit_3.text()
        self.qualifications=self.lineEdit_2.text()
        self.hide()
        cv_print_gui.show()

    def go_back(self):
        cv_gui.hide()
        login_gui.show()
```

In this piece of code, I am creating new variable names in which the data that has been entered by Bob into the line edits, will be stored. This is so that the data can be called at a later point. At the same time I made sure the buttons on the window made for easy navigation through the different windows.

These are also more examples of using Hungarian notation throughout my code. Here is am naming buttons that explain what each button will show when they are clicked.

Stage 3:

Once the “CV form” was completed I realised Bob would need to be able to print the form off. This is so that they could, if they wanted, keep a hard copy for their own personal documents. Alternatively, they may decide they want to post the CV to the company or hand it in personally.

To do this I created a new class. This meant I would have to initialise the class and create a new window. On this window Bob could decide whether to go back to “edit” their cv in case they wanted to add something, or they could just print it.

```
class cv_print_class(QtGui.QMainWindow, cv_window_print):  
    def __init__(self, parent=None):  
        QtGui.QMainWindow.__init__(self, parent)  
        self.setupUi(self)
```

Stage 4:

The next step was to get the data which Bob had entered the previous window (CV form). This was so that the programme knew which data needed to be printed, and in what format. This is why I decided to save the data into different variable names so it would be easier to call if I needed it elsewhere. The reason “self.” Was used, was because the information needed is found in a different class. This is an example of inheritance used within my programme. I am using properties and data from the previous class and it is being inherited into my new class. The parent class is the CV class, whereas the CV_Print_class is the child class as it is inheriting properties from the other class.

```
def replace(self):  
    self.name_address.setText(cv_gui.name)  
    self.experience.setText(cv_gui.pexperience)  
    self.p_employment.setText(cv_gui.pemployment)  
    self.hobbies.setText(cv_gui.hobbies1)  
    self.qualifications.setText(cv_gui.qualifications1)
```

This piece of code shows how I have used Hungarian notation to name each text box. This will make the code easier to understand.

Stage 5:

This stage was creating the actual print function, so the data could be sent to a printer

```
def goPrinter(self):  
    printer=QtGui.QPrinter()  
    dialog = QtGui.QPrintDialog(printer, self)  
    if(dialog.exec_() != QtGui.QDialog.Accepted):  
        return
```

This piece of code is allowing Bob choose which printer they want to send their document to. Once they have selected a printer, they will be able to view a preview of what the document will look like before they click print.

```
p=QtGui.QPixmap.grabWidget(self.frame)  
printLabel = QtGui.QLabel()  
printLabel.setPixmap(p)  
painter = QtGui.QPainter(printer)  
printLabel.render(painter)  
painter.end()
```


Sign Up:

Stage 1:

```
class signup(QtGui.QMainWindow, signup_window):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.signup_uname=""
        self.signup_pword=""
```

The first step was to initialise the sign-up class and declare the variables I would be using within the function.

```
def sign_up(self):
    signup_gui.hide()
    login_gui.show()
    contents = db.connect("Rental System.sqlite")
    cursor=contents.cursor()
    self.signup_uname=self.signup_uname_box.text()
    self.signup_pword=self.signup_pword_box.text()
```

The next step was to create new variables that would save the data that Bob had inputted into the programme. This is so that I can later call these variables in the programme so that the data which has been inputted can be saved to the database. These variables will be the ones used in the login function so that the user can continue through the programme. The code also shows the programme connecting to the database so that data can be read and written to.

This is also an example of using Hungarian notation.

Stage 2:

```
signup_window = uic.loadUiType("Coursework GUI Sign up Page.ui")[0]
```

The next stage was to load the sign-up window so that Bob had somewhere to enter their details.

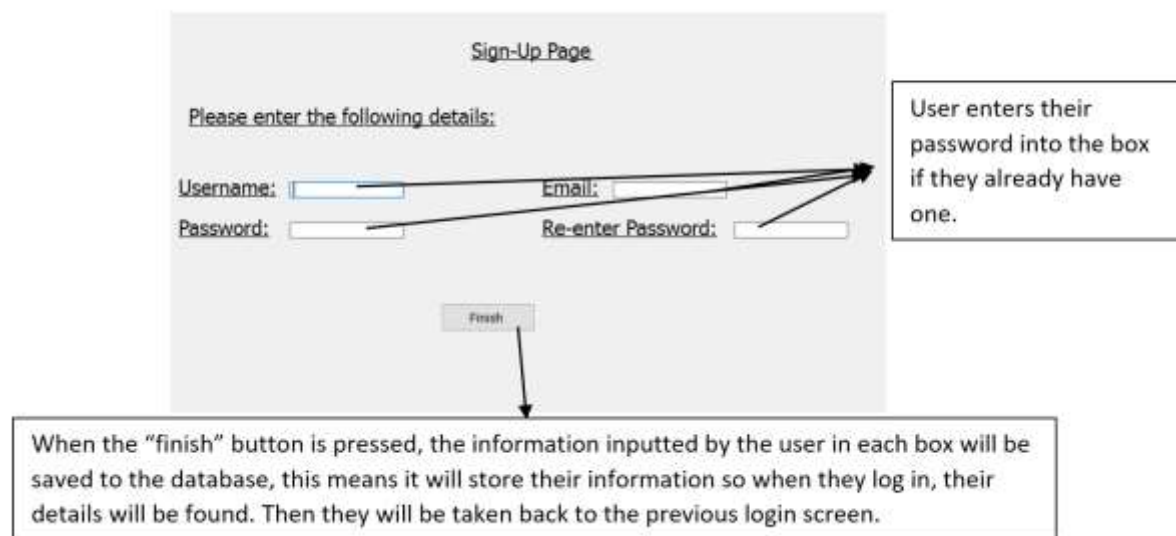
Stage 3:


The last step was to save the data that Bob would input into database

```
contents.commit()
```

```
contents.close()
```

This piece of code shows that information is being saved to the DB and then the DB is being "closed" to prevent data redundancy.



What am I Testing	How is it tested	Was it successful												
Does the sign-up window load when the programme is run	I will run the programme. The programme should load the login screen. From here I should be able to click the Sign-up window so that it will appear in the programme.	Successful- 												
When Bob fills out the details on the sign-up form, are their details saved in the database	I will run the programme and fill out the necessary fields on the form. I will then check to see whether the database has been updated.	Successful- <table><tr><th>Email</th><th>Username</th><th>Password</th><th>V Password</th></tr><tr><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td></tr><tr><td>godwin@hotmail.com</td><td>godwin</td><td>hello</td><td>hello</td></tr></table>	Email	Username	Password	V Password	Filter	Filter	Filter	Filter	godwin@hotmail.com	godwin	hello	hello
Email	Username	Password	V Password											
Filter	Filter	Filter	Filter											
godwin@hotmail.com	godwin	hello	hello											

Once I had completed my testing of the sign-up function I showed the results to the stake holders of PremiumExecutiveTravel. They were pleased with the results as the programme took the inputs of Bob and saved them to the database which was the main feature of this page. However, they again said the window needed to look more aesthetically appealing as well as the fact that more

information would need to be collected from Bob. Furthermore, Godwins pointed out I would need an auto increment system for any new customers who may join the system. Knowing this I went away and re designed the sign up form.

This was the new window, which Godwin was pleased with. The form collected more data from Bob as well as giving a more professional look to the system.



As you can see there have been more fields added so more information can be collected. This meant the code had to change slightly.

```
cursor.execute("""INSERT INTO Customer (F_Name, L_Name, DOB, Mobile_No, Address, Postcode, County, Email, Username, Password, V_Password, CustomerID) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)""",
                (self.first_name.text(),
                 str(self.surname.text()),
                 str(self.dob.text()),
                 str(self.number.text()),
                 str(self.address.text()),
                 str(self.postcode.text()),
                 str(self.country.text()),
                 str(self.email_box.text()),
                 str(self.signup_username_box.text()),
                 str(self.signup_password_box.text()),
                 str(self.signup_verify_password_box.text()),
                 new_customer_id))
```

Secondly:

This piece of code shows how I developed the auto increment system asked for by Godwin.

```
auto_increment="""SELECT CustomerID FROM Customer ORDER BY CustomerID DESC LIMIT 1"""
cursor.execute(auto_increment)
response=0
response=cursor.fetchone()
print(response)
res = int(response[0])
new_customer_id=response[0] +1
print(new_customer_id)
```

When a new user presses the “sign up” button, they will be asked to enter their details. From there their details will be saved to a database, specifically the Customer table. Within this table is a column called “customer ID”. This column is the primary key of the table. However, when each new customer signs up they will need a “customer ID” that is unique to them. This piece of code is implementing a way for the programme to assign a new, unique “customer ID” to each new user. It does this by selecting the last entry in the table, and adding 1 to the previous “customer ID”

Raees Qaisir

This screen shot shows a new customer who is signing up for an account so they can access the programme.

Please enter the following details:

First Name:	Address:	Username:
Alex	9 Fern Villa	alex
Surname:	Postcode:	Password:
Hodges	SL9 8UY	enter
DOB:	County:	Re-enter Password:
19/07/99	Buckinghamshire	enter
Mobile Number:	Email:	
078346758	alexh@gmail.com	

Go Back Sign Up

The data has been saved to the database.

F_Name	S_Name	DOB	Mobile_No	Address	Postcode	County	Email	Username	Password	V_Password	CustomerID
Godwin	Cherian	12/12/99	07113924322	6 Gavlots Way	HP11 8UY	Buckinghamshire	godwin@hotmail.com	godwin	hello	hello	1

This screen shot shows a stakeholder who has already created an account with the company. From the screen shot you can see that the “customer ID” is “1”.

F_Name	S_Name	DOB	Mobile_No	Address	Postcode	County	Email	Username	Password	V_Password	CustomerID
Godwin	Cherian	12/12/99	07113924322	6 Gavlots Way	HP11 8UY	Buckinghamshire	godwin@hotmail.com	godwin	hello	hello	1
Alex	Hodges	19/07/99	078346758	9 Fern Villa	SL9 8UY	Buckinghamshire	alexh@gmail.com	alex	enter	enter	2

As you can see the new user has been added to the database. However, this time instead of the “customer ID” being the same as the previous user, they have a new unique “ID”. The “ID” has autoincremented.

Lastly;

I realised that it wouldn't be very secure if I was to store the passwords that each user had created into the Customer table within the database. This is because anyone with access to the database could easily find a password for any account they wanted, meaning that the customers private information could be changed easily. Therefore, I decided that each password that would be created by a new user who is signing up, would need to be hashed. This is to improve data security. Once a

password has been hashed, you cannot go back and find the original password, ensuring that the customers password is now safe.

To begin with I went to the beginning of my “Sign Up” class where I had previously defined the variables which would be taking the input of Bobs password.

```
import hashlib

def sign_up(self):
    signup_gui.hide()
    login_gui.show()
    contents = db.connect("Rental System.sqlite")
    cursor=contents.cursor()
    self.signup_uname=self.signup_uname_box.text()
    self.signup_pword=self.signup_pword_box.text()
    self.signup_pword_hash=hashlib.sha256(self.signup_pword.encode()).hexdigest()
```

The final line of code shows how I would hash each password that was entered by Bob. It also shows I have used Hungarian notation to name each textbox.

The next step was to save the hashed password to the customer table in the database. This meant I had to re edit my previously written SQL statement.

```
print(str(self.signup_uname),str(self.signup_pword))
cursor.execute("""INSERT INTO Customer (f_name, s_name, DOB, Mobile_No, Address, Postcode, County, Email, Username, Pword_hash, CustomerID) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)""",
    (self.firm_name.text(),
    str(self.surname.text()),
    str(self.dob.text()),
    str(self.number.text()),
    str(self.address.text()),
    str(self.postcode.text()),
    str(self.county.text()),
    str(self.email_box.text()),
    str(self.signup_uname_box.text()),
    str(self.signup_pword_box.text()),
    str(self.signup_pword_hash),
    new_customer_id))
```

The code above is showing how I have changed what used to be previously a line edit that was saving the data written into it by a user, to a variable name which was defined earlier which stored the hash value.

Whilst editing the SQL statement, I realised that it would be irrelevant to have a hashed password in the database if I still had the original password written next to it. Therefore, I decided to remove the “original” passwords that were being saved to the table, with only the hashed value.

The final step was to go to the code which checked to see if the correct passwords matched, (and if they did, Bob could login in), and edit the code so that instead, only the hashed version of the password entered would be checked to see if it was matched one another.

However, before I did this I decided to create a hashing function. This is because if I decided to use the hash somewhere else, in the programme, I would simply just have to call the function instead of re writing code. It also made the login more simple when checking the hashed passwords against one another.

```
def hashing(string):
    hashed_string=hashlib.sha256(string.encode()).hexdigest()
    return hashed_string
```

I created the function outside of the class so that the function could be called from anywhere in the programme. This function is taking the input of any string and turning it into a hashed “password”.

The check for the password match has been done in the Login class at the beginning of the documentation.

Lastly, I validated the username and password boxes that the user would enter data into. This is to ensure that passwords and usernames are long enough to provide enough security. Furthermore, the validation also prevents a user from creating the same username as another customer who is already in the database.

```
cursor.execute("""SELECT Username FROM Customer""")
data=cursor.fetchone()
```

Firstly I created a SQL query which would save the usernames from the database to a new variable called "data".

Once this had been done I created an if statement which would prevent the user from using the same username as somebody else who was already in the database.

```
if self.signup_uname_box.text()!=(data):
    QtGui.QMessageBox.critical(self, "Status", "Username already taken")
```



Furthermore, if the user tried to enter a username or password that was below the length required for the programme, another message box would appear preventing them from creating an account.



The code to create this is as follows

```
class:
    def __init__(self, signup_username_box, signup_password_box):
        self.signup_username_box = signup_username_box
        self.signup_password_box = signup_password_box

        self.signup_password_hash = hashlib.sha256(self.signup_password.encode()).hexdigest()
        print(self.signup_password_hash)

        auto_increment = """SELECT CustomerID FROM Customers ORDER BY CustomerID DESC LIMIT 1"""
        cursor.execute(auto_increment)
        response = cursor.fetchone()
        print(response)
        new_customer_id = response[0] + 1
        print(new_customer_id)

        #print(str(self.signup_username), str(self.signup_password))
        cursor.execute("""INSERT INTO Customer (First_name, Surname, DOB, Mobile_No, Address, Postcode, County, Email, Username, Password_hash, CustomerID) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)""")
        cursor.execute("""INSERT INTO Customer (First_name, Surname, DOB, Mobile_No, Address, Postcode, County, Email, Username, Password_hash, CustomerID) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)""")

        #QtGui.QMessageBox.critical(self, "Signup Status", "You have successfully created an account")

        signup_gui.hide()
        login_gui.show()

        contents.commit()

    def close(self):
        QtGui.QMessageBox.critical(self, "Status", "Your username needs to be longer than 4 character, password needs to be longer than 5 character")
```

However, if Bob created a new username that wasn't already in the database, and was longer than 4 characters and the password was longer than 5 characters, then Bob would be able to make his account.



The screenshot shows a SQLite Database Browser window. The title bar indicates the file path: C:\Users\Raees Q\Documents\9 Form\Computing\Coursework\PythonProject\Coursework\Rent. The window has tabs for 'New Database', 'Open Database', 'Write Changes', and 'Revert Changes'. Below the tabs are buttons for 'Database Structure', 'Browse Data', 'Edit Pragma', and 'Execute SQL'. The 'Database Structure' tab is active, showing a table with 11 columns: i_Name, i_Name, DOB, email, Address, ostead, County, Email, sernam, jondhe, and stoner. The table contains 4 rows of data.

	i_Name	i_Name	DOB	email	Address	ostead	County	Email	sernam	jondhe	stoner
1	Alex	Cherian	2006/...	0706...	7 Gav...	godw...	Bucks	bob...	kiran	2cf24...	1
2	Godw...	Cherian	1995/...	0707...	8 Gav...	SL7 8...	Bucks	email...	godwin	2cf24...	2
3	Bob	Smith	97/04...	0702...	4 Ma...	SL6 ...	Buckl...	asmit...	alex...	e00d...	3
4	Paul	Martin	93/06...	0702...	8 Mas...	SL7 8...	Berks...	pMar...	george	5e88...	4

Booking:

Since the login window and sign up window had now been done and the programme was starting to take form, I thought now would be a good time to start the booking function. However, the booking function requires many different parts to work properly, so I decided to start with the rental calculator part of the programme which will produce the price of the car dependent on the model of car rented and the amount of days.

Stage 1:

Once the rental class had been initialised and the window had been created, I started to define which cars would be in the programme that Bob could select. I wrote each car into the programme at this point as the "cars" table had not yet been created. If I tried to fetch the cars from the "cars" table within the database, I would get this error.

Car Rental Booking Form

Logout

☐ Driver Over 21

Car:

Make/Model:

Type:

Registration:

Fuel Type:

Condition of Car:

:

Select the Time/Date for the car leaving:

00:00:00

December 2016

Sun	Mon	Tue	Wed	Thu
27	28	29	30	1
4	5	6	7	8
11	12	13	14	15

Select the Time/Date the car should be returned:

00:00:00

December 2016

Sun	Mon	Tue	Wed	Thu
27	28	29	30	1
4	5	6	7	8
11	12	13	14	15

This is the booking form. From here the customer will be able to enter the following details so they can hire the car.

sqlite3.OperationalError: no such table: Cars

So, to get around this I wrote each car into the programme. Furthermore, at this point, I just needed to see whether I could get the function working how it should be so I had something to show Godwin.

```
self.cars={"Audi RS6":150,"Ferrari 488 GTB":140,
           "McLaren 675 LT":250,"Mercedes C63 AMG":190,
           "Mercedes SL65 AMG":200,"Porsche 911 R":180,
           "Porsche Turbo S":145,"Rolls Royce Dawn":220,"VW Golf R":180}
```

These are the cars I decided to use within the programme. The number after each car depicts how much each one will cost per day the car is rented. Once this had been done I needed to define some variables.

Stage 2:

```
self.cost=0
self.car=""
self.rental_days=0
self.today=QtCore.QDate.currentDate()
select_car=self.comboBox.currentText()
```

This piece of code shows how I am stating what each value the variable should start with. At the same time, I am calling a built-in function, to help with the calculation for the rental. At the same time, I am also creating a new variable, where the contents of the drop down list which will contain the different cars Bob can select, will be stored so it can be called at a later point.

Stage 3:

```
for each in self.cars.keys():
    item = QtGui.QListWidgetItem(each)
    self.list1.addItem(item)
```

This piece of code is showing a loop function, where each item is being iterated through the list "cars". Another variable "item" is being defines which is using an inbuilt function to help with the final calculation. The procedure is then being called at the end of the loop.

Stage 4:

```
def set_date(self, date):
    self.rental_days=self.today.daysTo(date)
    daily_rate=self.cars[self.car]
```

This piece of code is showing how I make use of inbuilt functions. I am creating a new variable "rental_days" which has the value of getting todays date and the date to which the customer wants to rent the car. The daily_rate variable is how much the different cars cost per day

```
def on_item_changed(self, curr, prev):
    self.car=curr.text()
    self.calc_quote()
```

Stage 5:

This was the final stage of creating the calculation function. At this point I needed to multiply the amount of days the car is rented, by the value the car had been given earlier on.

```
def calc_quote(self):  
    self.cost=self.cars[self.car]*self.rental_days  
    print(self.cost)  
    self.lbl1.setText("{}:5}  {}".format(self.cost))
```

A am also defining what format I want to see the price printed.

This price would then be shown on the calculation window.

The class I had created worked. It displayed a price the customer would have to pay based on the car they have selected and the amount of days they want to rent the car for. However, the class was still missing key details such as saving the data selected by Bob into a database, the cars Bob could select were written into the programme, where as they should have been fetched from the database, the calculation did not consider current VAT rates, nor did the programme check to see whether Bob was of age to make a booking (over 21). Therefore, before I showed Godwin the finished class, I decided to go back and add these key elements to the code.

Booking (2)

Stage 1:

The first step I took to improving the calculation class was to include an auto increment system into the function. This is so that each booking made by a customer would have its own unique ID. This means it could act as the primary key for the booking table.

The code used was similar to the code I used for my Sign Up” class

```
auto_increment_booking="""SELECT Booking_ID FROM Booking ORDER BY Booking_ID DESC LIMIT 1"""  
cursor.execute(auto_increment_booking)  
response=0  
response=cursor.fetchone()  
print(response)  
res = int(response[0])  
self.new_booking_id=response[0] +1  
print(self.new_booking_id)
```

For this piece of code to work, I needed to change the table where the programme was getting the information from. Instead of being from the “customer” table, it needed to be from the “bookings” table. The rest of the code takes looks at the previous booking ID and adds 1 to the ID. This will be the new booking ID for any booking made by a user (Bob).

Stage 2:

After this had been completed, the next problem I began to work on was making sure the programme would retrieve the different cars, along with the prices, registration and any other details from the database, specifically the “cars” table.

```
cursor.execute("""SELECT Model FROM Car""")  
car_models =cursor.fetchall()
```

This piece of code was the first step in making sure the programme would retrieve the cars from the database. The SQL query is selecting the different models from the "car" table and then saving the results to a variable called car_models.

Once this was done, I needed to display the different cars available from the company to Bob.

```
self.comboBox.clear()  
for model in car_models:  
    self.comboBox.addItem(model[0])
```

This piece of code is displaying all the cars saved to the variable car_models in the combo box. I have implemented a for loop so that all the cars within the list are displayed.

Stage 3:

Once the cars were being displayed in the combo box, the next step was to get the price of each car from the database so the programme could do the calculation for Bob.

First, I created a new variable which saved Bob's choice of car. This is so that the variable could be called at a later point in the calculation

```
self.current_car = self.comboBox.currentText()
```

Once this had been done I connected to the database

```
contents = db.connect("Rental System.sqlite")  
cursor = contents.cursor()
```

After I did this, the next step was to write an SQL query which selected the price of each car depending on which model Bob had selected.

```
cursor.execute("""SELECT Price FROM Car WHERE Model='%s'"""%self.current_car)  
car_price = cursor.fetchone()  
self.car_price = int(car_price[0])
```

The model that Bob had selected was saved to the variable current_car. The price of the car was then saved to the variable car_price. I needed to make the value being fetched from the database into an integer value so that the calculation in the programme could be carried out properly.

Stage 4:

Once this had been completed, I decided to work on a function that would save the current day and the day that the car would be returned by Bob. This is because this data would need to be stored in the database as part of the information about the booking.

Since I was using a calendar within my GUI, I could use the internal functions associated with the calendar, however, I saved each value to new variables so it would be easier for me to understand what the code was doing when I was reading through it.

```
self.today = QtCore.QDate.currentDate()
self.end_date = self.cal.selectedDate()
```

To find the amount of days the car would be rented, I needed to find the difference between the current date and the end date. Again, I could use an inbuilt module to do this.

```
self.days_rented = self.today.daysTo(self.end_date)
```

Once this was complete, I needed to save the values into a new format. This is because when the values were stored in the database, it looked like this

Table: Booking

Booking	Registration	Rent_Date	Return_Date	Amount_Payable
Filter	Filter	Filter	Filter	Filter
1				
2	('RB16YUX')	PyQt4.QtCore.QDate(2017, 3, 31)	PyQt4.QtCore.QDate(2017, 3, 31)	0.0
3	('RB16YUX')	PyQt4.QtCore.QDate(2017, 3, 31)	PyQt4.QtCore.QDate(2017, 3, 31)	0.0
4	('RB16YUX')	PyQt4.QtCore.QDate(2017, 3, 31)	PyQt4.QtCore.QDate(2017, 3, 31)	0.0
5	('RB16YUX')	PyQt4.QtCore.QDate(2017, 3, 31)	PyQt4.QtCore.QDate(2017, 3, 31)	0.0
6	('RB16YUX')	PyQt4.QtCore.QDate(2017, 3, 31)	PyQt4.QtCore.QDate(2017, 3, 31)	0.0
7	('RB16YUX')	PyQt4.QtCore.QDate(2017, 3, 31)	PyQt4.QtCore.QDate(2017, 3, 31)	0.0
8	('RB16YUX')	PyQt4.QtCore.QDate(2017, 3, 31)	PyQt4.QtCore.QDate(2017, 3, 31)	0.0
8	('RB16YUX')	PyQt4.QtCore.QDate(2017, 3, 31)	PyQt4.QtCore.QDate(2017, 4, 7)	183.75
8	('RB15UOP')	PyQt4.QtCore.QDate(2017, 3, 31)	PyQt4.QtCore.QDate(2017, 4, 7)	244.99999999999997
8	('RB15UOP')	PyQt4.QtCore.QDate(2017, 3, 31)	PyQt4.QtCore.QDate(2017, 4, 7)	244.99999999999997
8	('RB15UOP')	PyQt4.QtCore.QDate(2017, 3, 31)	PyQt4.QtCore.QDate(2017, 4, 7)	244.99999999999997
8	('RB15UOP')	PyQt4.QtCore.QDate(2017, 3, 31)	PyQt4.QtCore.QDate(2017, 4, 7)	244.99999999999997

However, I did not want it in this format

Therefore I decided to change the format of how the date was stored in the database

To do this I overwrote the variables I had previously created, and specified the format I wanted the dates to be stored in the database

```
self.today=self.today.toString('yyyy/MM/dd')
self.end_date=self.end_date.toString('yyyy/MM/dd')
```

This code is showing that the dates should be saved to each variable in the format of year, month and then day.

This now meant that the dates within the database would be stored like this

28	16	GY11NUY	2017/03/31	2017/04/07	232.75
29	17	RB16YUX	2017/03/31	2017/04/08	210.0
30	18	AP15TGB	2017/03/31	2017/04/08	252.0

Stage 5:

The next stage I decided to work on was the actual calculation that would be displayed to Bob. However, Bob could only make a booking if he was over the age of 21, if not, he could not complete the booking. Therefore the first step I took was to select the age of Bob from the database. This information was available as he would have entered it when he created his account to login to the programme.

To do this I created an SQL statement.

```
contents = db.connect("Rental System.sqlite")
cursor = contents.cursor()
cursor.execute("""SELECT DOB FROM Customer WHERE username='%s'"""%username)
get_DOB=cursor.fetchone()
```

This statement is telling the programme to get the DOB of the user depending on what their user name is, so when the Bob logs into the programme, only his DOB will be fetched. The value for Bob's DOB was then saved to a new variable called get_DOB.

Once this was done, I needed to change the format of the way the programme interpreted the date. This is because, for the programme to calculate whether Bob was old enough to make a booking, all of the dates used needed to be in the same format as one another (Year, Month, Date).

```
self.today = datetime.strptime(self.today, date_format)
self.dob = datetime.strptime(get_DOB[0], date_format)
```

This piece of code is showing how the programme gets the current date. However, I am saving the current date in a new format and saving it to a new variable.

```
today = QtCore.QDate.currentDate()
self.today=today.toString('yyyy/MM/dd')
```

Here I am specifying the format I want the date to be stored in

```
date_format = "%Y/%m/%d"
```

This piece of code is re writing the variables to store the date in the format stated above.

```
self.today = datetime.strptime(self.today, date_format)
self.dob = datetime.strptime(get_DOB[0], date_format)
```

Since the variable get_DOB stored the data within it as a tuple, I needed to place [0] around it so that only the date was selected.

Since all the dates I needed to use within the programme were now stored in the same format as one another, I could now carry out a calculation where the programme calculated the difference in days between the current date and the date that Bob was born.

```
delta = self.today - self.dob
```

This piece of code is showing how the data stored to the variables I had previously created are being called and saved to a new variable called delta.

Once the difference in days had been calculated, I could now write an IF statement where the user had to be over 21 to make a booking.

```
if int(delta.days) < 21*365:  
    QtGui.QMessageBox.critical(self, "Status", "Sorry you are not of age to make a booking")
```

This piece of code is how the programme is calculating whether Bob can make a booking. I did this by calculating the amount of days someone has to have lived within 21 years. If Bob has lived less days than 21*365, then the programme knows the Bob is younger than the age of 21 and prevents him from making a booking. A message box appears when Bob clicks the "finish" button saying that is not of age to make a booking. Furthermore, calculating the amount of days Bob is, is a good way to improve on validation of the programme. For example, if I had calculated by year or month, then Bob could technically fool the programme by making it think that he was 21 when his birthday was in fact in August. However, since the programme thinks that every January is a new year, the programme would assume Bob is 21, when in fact he is still 8 months away from being 21. Therefore, doing the calculation based on days was the most valid method.

If Bob was over the age of 21, then the programme would let Bob continue and make a booking. First of all, I called the 2 variables, which had been previously defined, needed to do the calculation for the price Bob would need to pay.

```
else:  
    self.get_car_price()  
    self.get_days_rented()
```

Once this had been done I created a new variable (self.cost) where I multiplied the cost of the car by the amount of days Bob wanted the rental for.

```
self.cost = self.car_price * self.days_rented
```

However, this calculation did not take into account the current rate of VAT. Therefore I created a new variable where I took the result of (self.cost) and multiplied it by the current VAT rate.

```
self.cost = self.car_price * self.days_rented
self.cost_VAT=round(self.cost*0.2,2)
#VAT= 20% at the moment. To get the total cost (+VAT)
# I multiply the final cost by the current VAT rate.
#This will be the actual cost the customer will pay.
```

This would be the final cost that Bob would have to pay. Furthermore, I have also set the price shown to the customer to 2 decimal places. This is because all monetary transactions are done using the value to 2 decimal places.

However, this would have to be displayed to bob within the GUI, otherwise he wouldn't know how much he would owe.

```
self.lbl1.setText(str(self.cost_VAT))
```

This piece of code is showing how the price would be displayed on the rental_calc form.

```
self.cal.clicked.connect(self.calculate_price)
```

This piece of code is showing how the price will displayed, whenever Bob has selected a car and pressed a date on the calendar. As soon as he has pressed the date on the calendar, the price he needs to pay will be displayed in the QLabel because the calculate_price function is being run.

Stage 6:

The final stage of the rental_calculatoin class was to save the data Bob had selected to the database, specifically the bookings table. However, I needed to know which car was being rented. Instead of choosing to save the model, I decided to save the registration plate to the table as this was a primary key, as no two registrations can be the same.

To do this I needed to create a new SQL query .

```
contents = db.connect("Rental System.sqlite")
cursor = contents.cursor()

cursor.execute("""SELECT Registration FROM Car WHERE Model='%s'"""%(self.current_car))
fetch_reg=cursor.fetchone()
```

This statement is telling the programme to select the registration of the car that Bob has selected from the combo box. The registration is then being saved to a new variable called fetch_reg.

Once this was done, I could then write the details of the booking to the "Booking" table within the database.

This meant a new SQL query was needed.

```
cursor.execute("""INSERT into BOOKING VALUES(?, ?, ?, ?, ?) """,
               [str(self.new_booking_id),
                 str(fetch_reg[0]),
                 str(self.today),
                 str(self.end_date),
                 str(self.cost_VAT)])
```

This statement is showing how each variable is being entered into the "Booking" table in the database.

This is also another example of polymorphism. I have previously used similar code to this in my Sign-up Class, where the customer would enter their details and it would be saved to the database. Now I have edited the code so that it now saves the entered details from the booking form.

```
contents.commit()  
contents.close()
```

Since the programme is writing to the database, the database needs to be saved once the data has been written and closed. This is to ensure the integrity of the data in the database.

Once Bob had finished his booking, he can click the “finish” button where the rental window will close and the booking window will open. This window will confirm the details of his booking, as well as giving him the opportunity to print off a document with details of his booking on there.

The final step was to write a function that would open the booking form and close the rental window.

```
def open_booking_form(self):  
    wl.hide()  
    booking_gui.show()  
    QtGui.QMessageBox.information(self, "Status", "Booking Successful")
```

This piece of code shows the function that will close the window and open a new one. The function will also display a message box that congratulates Bob on completing his booking.

This function will be run when the “finish” button is clicked.

```
self.finish_btn.clicked.connect(self.open_booking_form)
```

Stage 7:

The last stage was to add some more validation to the rental calculation class.

```
if self.end_date<self.today:  
    QtGui.QMessageBox.critical(self, "Status", "You have entered an invalid date")
```

This piece of code is there so that if the user tries to click a date from the calendar before the current date (today's date), a message box will appear with a message alerting them that they cannot do this. The booking will not be saved to the database.

This is another example of polymorphism, as I have used similar methods to create these message boxes to appear, but I have made changes to the message box that is viewed and the messages within them.

```

else:
    if int(delta.days)<21*365:
        QtGui.QMessageBox.critical(self, "Status", "Sorry you are not of age to make a booking")

    else:
        self.get_car_price()
        self.get_days_rented()

        self.cost = self.car_price * self.days_rented
        self.cost_VAT=round(self.cost*0.2,2)
        #VAT= 20% at the moment. To get the total cost (+VAT)
        # I multiply the final cost by the current VAT rate.
        #This will be the actual cost the customer will pay.

        print(self.cost_VAT)
        print(self.days_rented)
        self.lbl1.setText(str(self.cost_VAT))
        self.lbl2.setText(str(self.days_rented))

        #all data needs to be saved to booking table

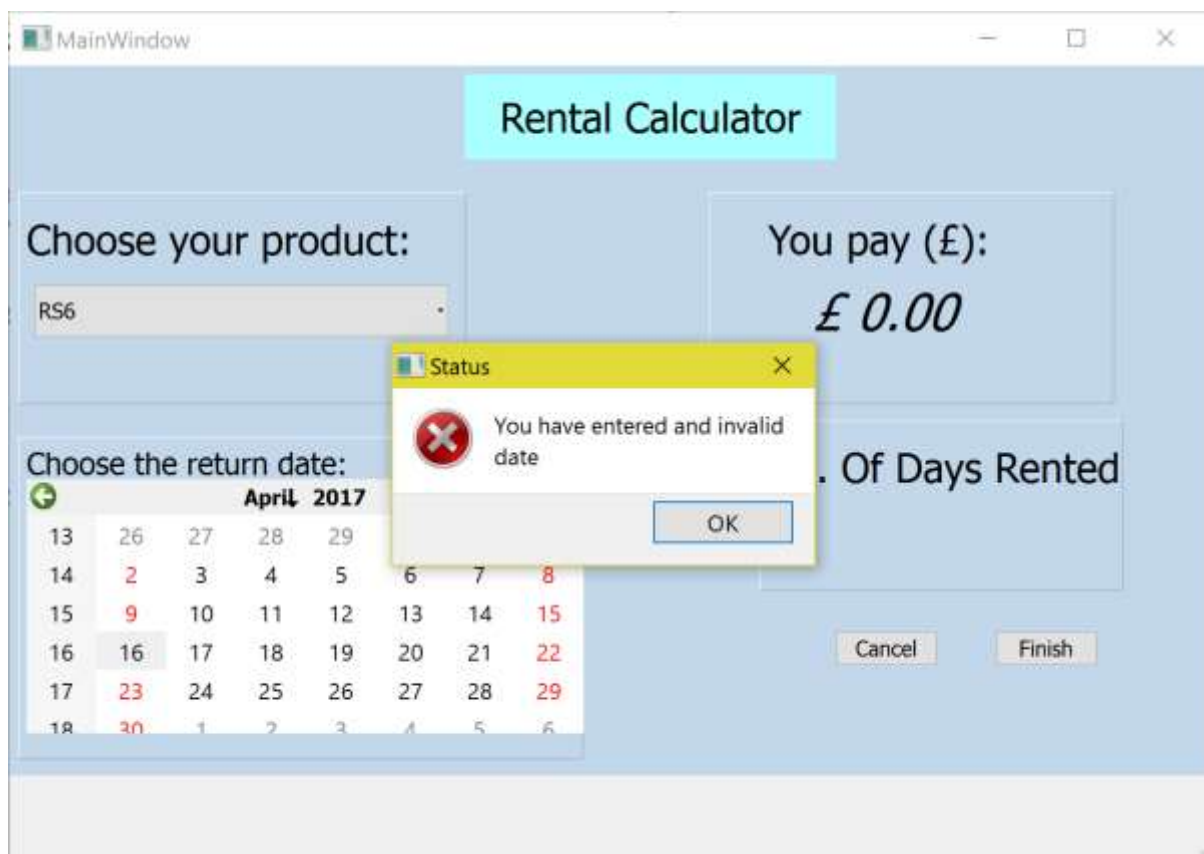
        contents = db.connect("Rental System.sqlite")
        cursor = contents.cursor()

        cursor.execute("""SELECT Registration FROM Car WHERE Model='%'""%(self.current_car))
        fetch_reg=cursor.fetchone()

        cursor.execute("""INSERT into BOOKING VALUES(?, ?, ?, ?, ?)""",
            [str(self.new_booking_id),
            str(fetch_reg[0]),
            str(self.today),
            str(self.end_date),
            str(self.cost_VAT)])

        contents.commit()
        contents.close()

```



Receipt Window

Stage 1:

I initialised the class

```
class booking_class(QtGui.QMainWindow, booking_window):  
    def __init__(self, parent=None):  
        QtGui.QMainWindow.__init__(self, parent)  
        self.setupUi(self)
```

Stage 2:

The next step was to make sure that Bob would be able to print off his receipt, in case he wanted to keep it for his own records.

```
def goPrinter(self): #from https://riverbankcomputing.com/pipermail/pyqt/2010-February/026944.html  
    printer=QtGui.QPrinter()  
    dialog = QtGui.QPrintDialog(printer, self)  
    if(dialog.exec_() != QtGui.QDialog.Accepted):  
        return  
  
    p=QtGui.QPixmap.grabWidget(self.frame)  
    printLabel = QtGui.QLabel()  
    printLabel.setPixmap(p)  
    painter = QtGui.QPainter(printer)  
    printLabel.render(painter)  
    painter.end()
```

I found the code on the internet to help me to create this function. Within the code I have put the URL as to where I found this information.

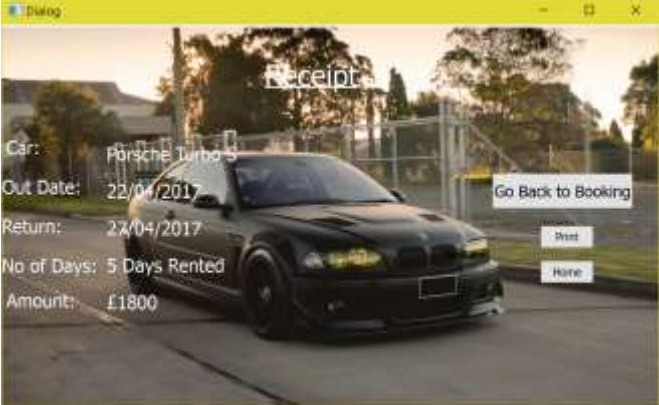
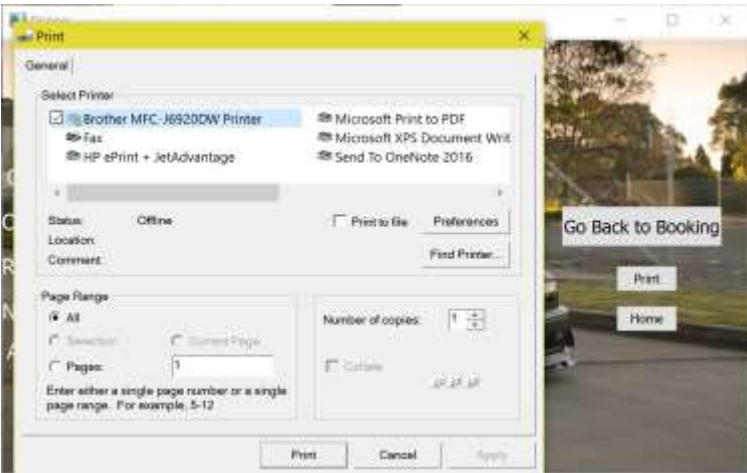
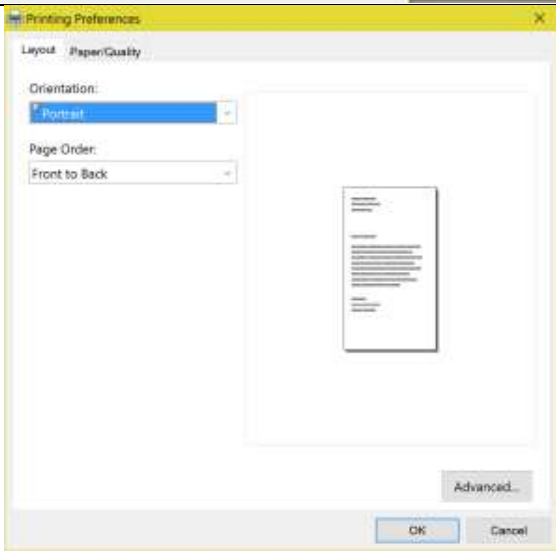
Stage 3:


Finally, the last stage was to create functions for the different buttons that would appear on the window

```
self.booking_back_btn.clicked.connect(self.open_calc)  
self.print_btn.clicked.connect(self.goPrinter)  
self.finish_btn.clicked.connect(self.go_back)
```

This code shows examples of Hungarian notation, where each button has been named specifically for the task it will carry out within the programme.

What am I Testing	How is it tested	Was it successful
Does the receipt window open	I will run the programme and create a	Successful-

<p>when the booking has finished</p>	<p>booking. I will click the “finish” button.</p>	
<p>Does the print window open when Bob clicks the “print” button</p>	<p>I will click the “print” button on the window</p>	<p>Successful-</p> 
<p>Does the programme allow Bob to make changes to preferences</p>	<p>I will click the preferences button on the printing window</p>	<p>Successful-</p> 

Does the programme return Bob to the login screen when he clicks the "Home" button	I will click the "home" button	Successful- 

Once testing was complete I showed the final version of the receipt window to Godwin. He was happy with the outcome, as it showed Bob what car he was renting, which day from, the day to, how many days he would rent the car for, and the total cost of the rental. He also liked the feature that allowed Bob to print off this receipt. Bob could change the amount of copies he would like of the receipt, change any preferences and change the printer he wants to send the document to.

Contact Us:

Stage 1

The first step was to initialise the class I would be using

```
class contact_us_class(QtGui.QMainWindow, contact_us_window):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
```

Once this was done, I then created the functions of what the different buttons would do that were displayed on the window.

```
def go_back(self):  
    contact_gui.hide()  
    login_gui.show()
```

```
def open_cv(self):  
    contact_gui.hide()  
    cv_gui.show()
```

Lastly I needed to link the buttons on the window, to the functions that had been created

```
self.back_btn.clicked.connect(self.go_back)  
self.cv_btn.clicked.connect(self.open_cv)
```

Stage 2:

The next stage was to create another class where Bob would be able to make a CV which he could then use to apply to the company.

```
class cv_class(QtGui.QMainWindow, cv_window):  
    def __init__(self, parent=None):  
        QtGui.QMainWindow.__init__(self, parent)  
        self.setupUi(self)
```

I then needed to save the data that had been entered by Bob into new variable names. This is so that when Bob came to printing his CV form, he would need the data that he had entered to be in the same format as when it is displayed.

```
def openCV(self):  
    self.name=self.lineEdit.text()  
    self.pexperience=self.lineEdit_5.text()  
    self.pemployment=self.lineEdit_4.text()  
    self.hobbies1=self.lineEdit_3.text()  
    self.qualifications1=self.lineEdit_2.text()  
    self.hide()  
    cv_print_gui.show()
```

Within this code, I am also closing the current window and opening a new window

Stage 3:

Lastly, I needed to link the buttons on the window to different functions so that Bob could navigate easily.

```
def go_back(self):  
    cv_gui.hide()  
    login_gui.show()  
  
self.cv_btn.clicked.connect(self.openCV)  
self.home_btn.clicked.connect(self.go_back)
```

More examples of Hungarian notation.

Stage 4:

The last stage was to display the text entered by Bob on a new window, similar to a print preview, so that Bob could either print the CV, or edit it if he wanted to.

To do this, I used the previous variables that I had saved the data to.

```
def replace(self):  
    self.name_address.setText(cv_gui.name)  
    self.experience.setText(cv_gui.pexperience)  
    self.p_employment.setText(cv_gui.pemployment)  
    self.hobbies.setText(cv_gui.hobbies1)  
    self.qualifications.setText(cv_gui.qualifications1)
```

Once this was done, I also used the same code I used previously for the Booking class so that Bob could print the CV


```
def goPrinter(self):
    printer=QtGui.QPrinter()
    dialog = QtGui.QPrintDialog(printer, self)
    if(dialog.exec_() != QtGui.QDialog.Accepted):
        return

    p=QtGui.QPixmap.grabWidget(self.frame)
    printLabel = QtGui.QLabel()
    printLabel.setPixmap(p)
    painter = QtGui.QPainter(printer)
    printLabel.render(painter)
    painter.end()
```

Stage 5:

The final step was to link the buttons on the window to their respective functions

```
def go_back_to_cv(self):
    cv_print_gui.hide()
    cv_gui.show()

cv_gui.cv_btn.clicked.connect(self.replace)
self.print_cv.clicked.connect(self.goPrinter)
self.edit_cv.clicked.connect(self.go_back_to_cv)
```

Forgot Password:

Stage 1:

Initialising the class, defining variables and linking functions to buttons

```
class forgot_pword_class(QtGui.QMainWindow, username_window):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.a = None
```

```
def go_home(self):  
    forgot_pword_gui.hide()  
    login_gui.show()  
  
def email_code(self):  
    forgot_pword_gui.hide()  
    enter_code_gui.show()  
  
self.cancel_btn.clicked.connect(self.go_home)  
self.nxt_btn.clicked.connect(self.email_code)
```

Stage 2:

Once that had been done, I created a new function. This function would create a random string mixed with letters and numbers

```
def random_id(length):  
    number = '0123456789'  
    alpha = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'  
    id = ''  
    for i in range(0, length, 2):  
        id += random.choice(number)  
        id += random.choice(alpha)  
    return id
```

To do this I created a for loop which went through each defined variable separately (number, alpha) and then saved the results to new variables (id). I also made use of an inbuilt python function (random)

I then needed the result to be saved to a new variable. However the variable needed to be a string and also only 8 characters long.

```
global a  
a=(str(random_id(8)))
```

Global was needed because the code didn't run, an error message would appear. This is because the variable would later be called from a different function. If Global wasn't used the variable would stay as a Local variable.


```
Traceback (most recent call last):
  File "C:\Users\Raees Q\Documents\6 Form\Computing\Coursework\PycharmProjects\Coursework\CURRENT PROGRAMME.py", line 221, in next_btn
    self.reset_pword()
  File "C:\Users\Raees Q\Documents\6 Form\Computing\Coursework\PycharmProjects\Coursework\CURRENT PROGRAMME.py", line 245, in reset_pword
    Code:*** + (a)
NameError: name 'a' is not defined
```

This is the error message that would appear.

```
def next_btn(self):
    global email
    email=self.enter_email.text()
    self.reset_pword()
```

This piece of code is so that the programme saves the email entered by Bob to a variable name.

Stage 3:

This step was to create the message that would be sent to the email that Bob had entered. To do this I first created a new function and then found some code on the internet.

```
def reset_pword(self):

    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login("executivepremiumtravel@gmail.com", "premiumexecutivetravel")

    msg = """Subject: Password Reset:\n
Hello, you have asked to reset your password, below you will find a code.\n
Please enter the code into the programme to create a new password:\n
Code:*** + (a)

    server.sendmail("executivepremiumtravel@gmail.com", email,msg)
    server.quit()
```

The code which was used from the internet was this:

```
server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()

server.login("executivepremiumtravel@gmail.com", "premiumexecutivetravel")

server.sendmail("executivepremiumtravel@gmail.com", email,msg)
server.quit()
```

These pieces of code are to connect the programme to the server that will be sending the email. I also have to input the email address which the email will be sent from, and the password to that account.

After that has been done, I then need to define which address (receiver) the email needs to be sent to. This is where the variable that I had saved the input from Bob is used. At the same time, I also need to tell the programme what needs to be sent to the recipient.

This "msg" is a variable name that has data saved to it.

Server. Quit is for when the email has been sent, otherwise, the data will not be sent. Similar to contents.close, contents.commit

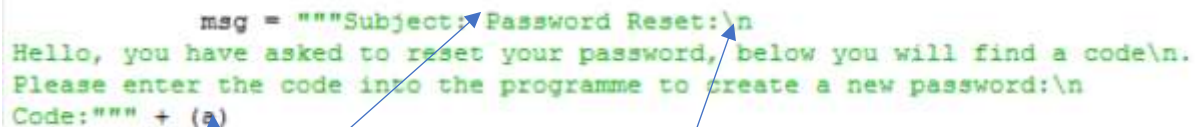
```
server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()
server.login("executivepremiumtravel@gmail.com", "premiumexecutivetravel")

msg = """Hello, you have asked to reset your password, below you will find a code\n.
Please enter the code into the programme to create a new password
Code: """""

server.sendmail("executivepremiumtravel@gmail.com", "raeesqaisir@gmail.com", msg)
server.quit()
print("email sent")
```

The message initialized looked like this. However, I did not feel as if it gave a professional image for the company, so I decided to change it.

```
msg = """Subject: Password Reset:\n
Hello, you have asked to reset your password, below you will find a code\n.
Please enter the code into the programme to create a new password:\n
Code: """"" + (a)
```



This is the message that will be received by the recipient (the email that Bob had previously entered).

The email needed to have a professional format, so that the email would look like it came from the company so that the user wouldn't automatically think it was spam. Therefore, I decided to create a subject heading for the email. I then used the line break to separate the heading from the main body of text. Once this had been done, I typed up the message.

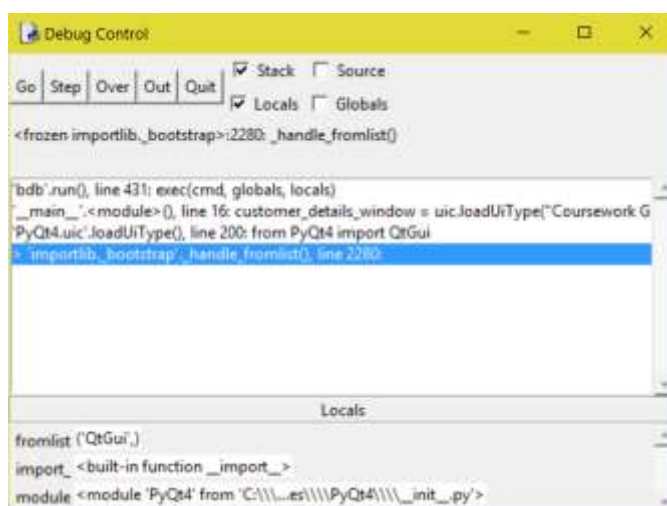
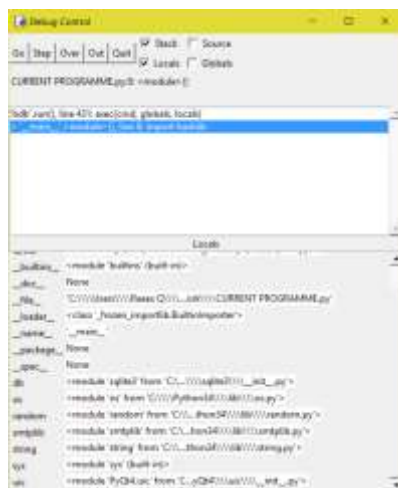
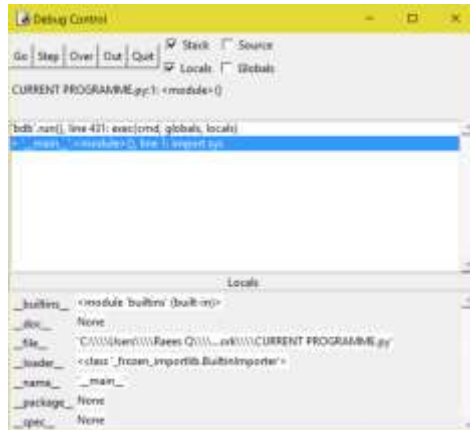
Finally, I then made use of the variable that I made previously that had stored the contents of the randomised string to the end of the email. This code is what the recipient would use to change their password at a later stage.

The reason Global was needed, was because the variable had been defined in a previous function, it was local, however it is being called from a different function, so the programme needs to be able to find it, therefore, Global was used.

Use of Debugger:

Throughout my programme, I made use of the debugger in IDLE.

```
>>> ===== RESTART =====  
[DEBUG ON]
```



The debugger shows all the different variables I have used throughout the class.

This meant that I could “step” through the programme, line by line if any errors occurred.




The debugger shows the procedure/function that I am currently working on. It also displays the current variables that are being used with the value that I held in it.

The debugger was useful feature to use in my programme as it meant that fixing errors within my code became far more manageable, and I could find the mistake easily.





Evaluation


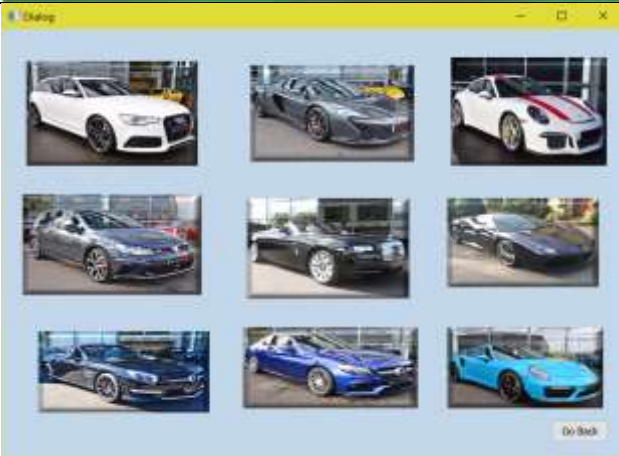

Now that the programme has been completed, each stage needed to be retested to see if the programme worked as it should.

Login:



Test Number (L)	What should it do (success criteria)?	How will I Test it?	Screen Shot of Testing	Successful
1	When the programme is run, the login screen should be displayed	I will run the programme		Pass
2	The login screen should take the inputs (username, password) from the user	I will enter correct login details into the text boxes		Pass
3	The inputs should be checked to see whether they match previous data from the database	I will press the "enter" button		Pass


4	If they match the user should be shown a message box	I will press the "enter" button		Pass
5	If they match the user should be shown to next screen (booking form)	I will press the "ok" button on the message box		Pass
6	If they do not match, then the user will be prompted to re-enter details by a message box	I will enter incorrect details into the text boxes and click the "enter" button		Pass
7	If the user does not have an account made, they should be able to make one	I will click the "sign up" button		Pass

8	If the user doesn't enter any details into the text entries	I will leave the text entries blank, and click the enter button		Pass
9	If the user enters a correct username, but a blank password	Enter a username in the database, but a blank password		Pass
10	If the user enters a correct password, but no username	I will enter a password, but no username		Pass
11	If the enter button is pressed multiples times quickly, will the program crash	I will click the enter button multiples time quickly		Pass





12	If the user enters in extreme details into the text entries, will the program stop this	I will enter in extreme values into the text boxes		Fail
13	Does the cars widow open	I will click the cars window		Pass
14	Does the cars window return to the home page when the back button is pressed	I will click the back button on the cars window		Pass


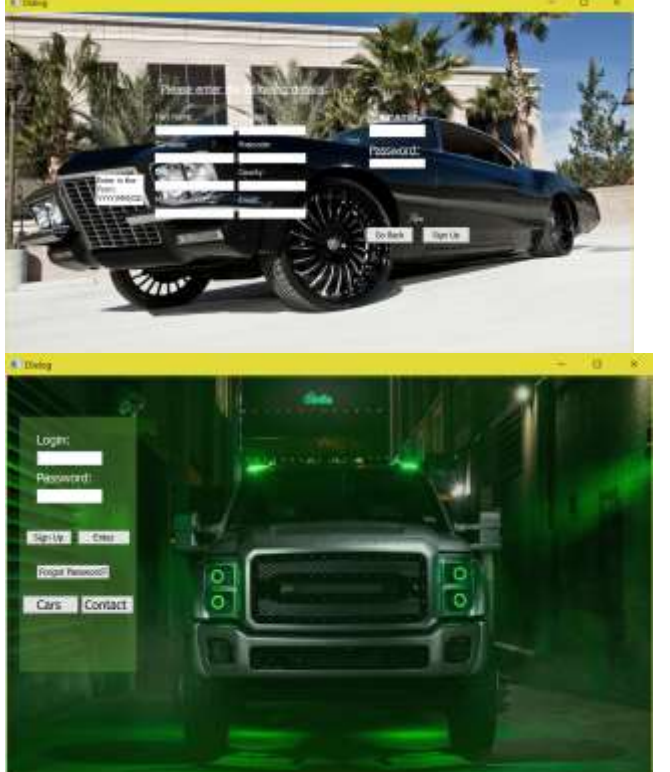
Sign Up:

Test Number (S)	What should it do (success criteria)?	How will I Test it?	Screen Shot of Testing	Successful
1	The program me should be able to "tab" through the different text boxes	I will click the "tab" button on the keyboard and check if the cursor moves through the different text boxes		Pass
2	The program me should take inputs from the user in the text boxes	I will enter data into the text boxes		Pass

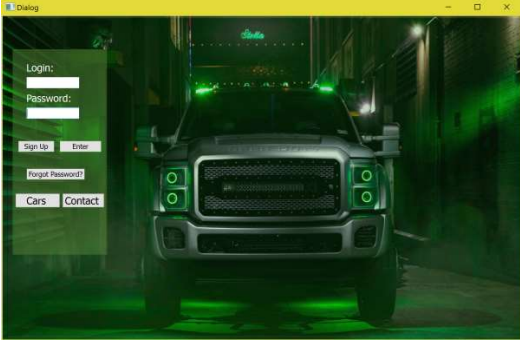
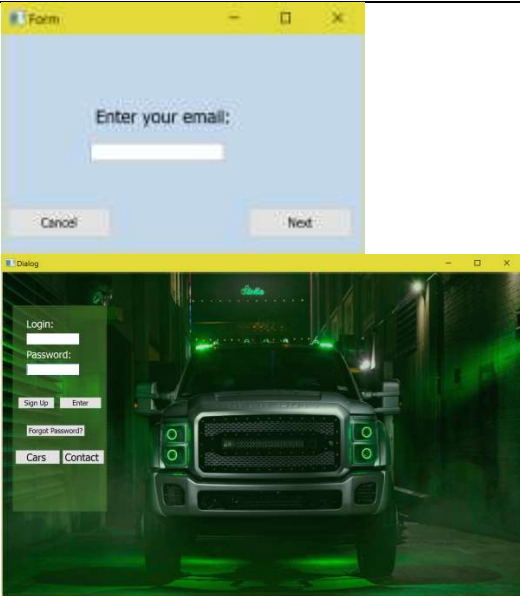
3	The form includes all details that may be needed by the programme	I will check this by filling out the relevant information		Pass																																																						
4	The information entered by the user in the sign-up form should be stored to a database	I will check the database to see if all the data entered on the form appears in the table when the “sign up” button is pressed	<table><thead><tr><th>F_Name</th><th>S_Name</th><th>DOB</th><th>Mobile No.</th><th>Address</th><th>Postcode</th><th>County</th></tr></thead><tbody><tr><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td></tr><tr><td>Alex</td><td>Cherian</td><td>06/02/12</td><td>07862...</td><td>7 Gaviots Way</td><td>SL3 8HY</td><td>Buckinghamshire</td></tr><tr><td>Godwin</td><td>Cherian</td><td>95/03/12</td><td>07872...</td><td>8 Gaviots Way</td><td>SL7 6YT</td><td>Buckinghamshire</td></tr><tr><td>Bob</td><td>Smith</td><td>97/04/09</td><td>07821...</td><td>4 Mumfords Road</td><td>SL6 8HG</td><td>Buckinghamshire</td></tr><tr><td>Paul</td><td>Martin</td><td>93/06/21</td><td>07823...</td><td>8 Nashdom Way</td><td>SL2 6HY</td><td>Berkshire</td></tr></tbody></table> <table><thead><tr><th>Username</th><th>Pword_hash</th></tr></thead><tbody><tr><td>Filter</td><td>Filter</td></tr><tr><td>kiran</td><td>2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824</td></tr><tr><td>godwin</td><td>2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824</td></tr><tr><td>alexsmith</td><td>e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c</td></tr><tr><td>george</td><td>5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8</td></tr></tbody></table>	F_Name	S_Name	DOB	Mobile No.	Address	Postcode	County	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Alex	Cherian	06/02/12	07862...	7 Gaviots Way	SL3 8HY	Buckinghamshire	Godwin	Cherian	95/03/12	07872...	8 Gaviots Way	SL7 6YT	Buckinghamshire	Bob	Smith	97/04/09	07821...	4 Mumfords Road	SL6 8HG	Buckinghamshire	Paul	Martin	93/06/21	07823...	8 Nashdom Way	SL2 6HY	Berkshire	Username	Pword_hash	Filter	Filter	kiran	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824	godwin	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824	alexsmith	e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c	george	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8	Pass
F_Name	S_Name	DOB	Mobile No.	Address	Postcode	County																																																				
Filter	Filter	Filter	Filter	Filter	Filter	Filter																																																				
Alex	Cherian	06/02/12	07862...	7 Gaviots Way	SL3 8HY	Buckinghamshire																																																				
Godwin	Cherian	95/03/12	07872...	8 Gaviots Way	SL7 6YT	Buckinghamshire																																																				
Bob	Smith	97/04/09	07821...	4 Mumfords Road	SL6 8HG	Buckinghamshire																																																				
Paul	Martin	93/06/21	07823...	8 Nashdom Way	SL2 6HY	Berkshire																																																				
Username	Pword_hash																																																									
Filter	Filter																																																									
kiran	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824																																																									
godwin	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824																																																									
alexsmith	e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c																																																									
george	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8																																																									
5	The information should be stored in the same order it is taken from the user	I will check the database to see if the information entered is in the same order as when it is entered in the programme when the sign up button is pressed	<table><thead><tr><th>F_Name</th><th>S_Name</th><th>DOB</th><th>Mobile No.</th><th>Address</th><th>Postcode</th><th>County</th></tr></thead><tbody><tr><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td></tr><tr><td>Alex</td><td>Cherian</td><td>06/02/12</td><td>07862...</td><td>7 Gaviots Way</td><td>SL3 8HY</td><td>Buckinghamshire</td></tr><tr><td>Godwin</td><td>Cherian</td><td>95/03/12</td><td>07872...</td><td>8 Gaviots Way</td><td>SL7 6YT</td><td>Buckinghamshire</td></tr><tr><td>Bob</td><td>Smith</td><td>97/04/09</td><td>07821...</td><td>4 Mumfords Road</td><td>SL6 8HG</td><td>Buckinghamshire</td></tr><tr><td>Paul</td><td>Martin</td><td>93/06/21</td><td>07823...</td><td>8 Nashdom Way</td><td>SL2 6HY</td><td>Berkshire</td></tr></tbody></table> <table><thead><tr><th>Username</th><th>Pword_hash</th></tr></thead><tbody><tr><td>Filter</td><td>Filter</td></tr><tr><td>kiran</td><td>2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824</td></tr><tr><td>godwin</td><td>2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824</td></tr><tr><td>alexsmith</td><td>e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c</td></tr><tr><td>george</td><td>5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8</td></tr></tbody></table>	F_Name	S_Name	DOB	Mobile No.	Address	Postcode	County	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Alex	Cherian	06/02/12	07862...	7 Gaviots Way	SL3 8HY	Buckinghamshire	Godwin	Cherian	95/03/12	07872...	8 Gaviots Way	SL7 6YT	Buckinghamshire	Bob	Smith	97/04/09	07821...	4 Mumfords Road	SL6 8HG	Buckinghamshire	Paul	Martin	93/06/21	07823...	8 Nashdom Way	SL2 6HY	Berkshire	Username	Pword_hash	Filter	Filter	kiran	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824	godwin	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824	alexsmith	e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c	george	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8	Pass
F_Name	S_Name	DOB	Mobile No.	Address	Postcode	County																																																				
Filter	Filter	Filter	Filter	Filter	Filter	Filter																																																				
Alex	Cherian	06/02/12	07862...	7 Gaviots Way	SL3 8HY	Buckinghamshire																																																				
Godwin	Cherian	95/03/12	07872...	8 Gaviots Way	SL7 6YT	Buckinghamshire																																																				
Bob	Smith	97/04/09	07821...	4 Mumfords Road	SL6 8HG	Buckinghamshire																																																				
Paul	Martin	93/06/21	07823...	8 Nashdom Way	SL2 6HY	Berkshire																																																				
Username	Pword_hash																																																									
Filter	Filter																																																									
kiran	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824																																																									
godwin	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824																																																									
alexsmith	e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c																																																									
george	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8																																																									

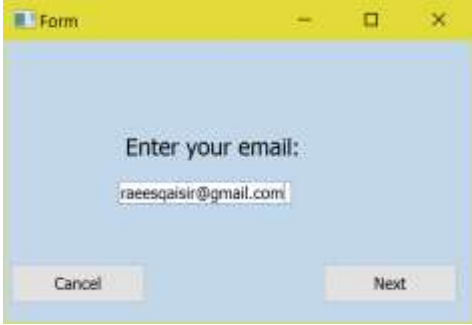

6	The program me should auto increment a unique id for each new user	I will check the database to see if Alex has a unique id when the sign up button is pressed	<table><tr><th>F_Name</th><th>S_Name</th><th>DOB</th><th>obile No</th><th>Address</th><th>Postcode</th><th>County</th><th>Email</th></tr><tr><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td></tr><tr><td>Alex</td><td>Cherian</td><td>06/02/12</td><td>07862...</td><td>7 Gaviots Way</td><td>SL3 8HY</td><td>Buckinghamshire</td><td>bob@hotmail.com</td></tr><tr><td>Godwin</td><td>Cherian</td><td>95/03/12</td><td>07872...</td><td>8 Gaviots Way</td><td>SL7 6YT</td><td>Buckinghamshire</td><td>email@email.com</td></tr><tr><td>Bob</td><td>Smith</td><td>97/04/09</td><td>07821...</td><td>4 Mumfords Road</td><td>SL6 &HG</td><td>Buckinghamshire</td><td>aSmith@gmail.com</td></tr><tr><td>Paul</td><td>Martin</td><td>93/06/21</td><td>07823...</td><td>8 Nashdom Way</td><td>SL2 6HY</td><td>Berkshire</td><td>pMartin@gmail.co</td></tr></table> <table><tr><th>Username</th><th>Pword_hash</th><th>CustomerID</th></tr><tr><td>Filter</td><td>Filter</td><td>Filter</td></tr><tr><td>kiran</td><td>2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824</td><td>1</td></tr><tr><td>godwin</td><td>2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824</td><td>2</td></tr><tr><td>alexsmith</td><td>e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c</td><td>3</td></tr><tr><td>george</td><td>5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8</td><td>4</td></tr></table>	F_Name	S_Name	DOB	obile No	Address	Postcode	County	Email	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Alex	Cherian	06/02/12	07862...	7 Gaviots Way	SL3 8HY	Buckinghamshire	bob@hotmail.com	Godwin	Cherian	95/03/12	07872...	8 Gaviots Way	SL7 6YT	Buckinghamshire	email@email.com	Bob	Smith	97/04/09	07821...	4 Mumfords Road	SL6 &HG	Buckinghamshire	aSmith@gmail.com	Paul	Martin	93/06/21	07823...	8 Nashdom Way	SL2 6HY	Berkshire	pMartin@gmail.co	Username	Pword_hash	CustomerID	Filter	Filter	Filter	kiran	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824	1	godwin	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824	2	alexsmith	e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c	3	george	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8	4	Pass
F_Name	S_Name	DOB	obile No	Address	Postcode	County	Email																																																															
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter																																																															
Alex	Cherian	06/02/12	07862...	7 Gaviots Way	SL3 8HY	Buckinghamshire	bob@hotmail.com																																																															
Godwin	Cherian	95/03/12	07872...	8 Gaviots Way	SL7 6YT	Buckinghamshire	email@email.com																																																															
Bob	Smith	97/04/09	07821...	4 Mumfords Road	SL6 &HG	Buckinghamshire	aSmith@gmail.com																																																															
Paul	Martin	93/06/21	07823...	8 Nashdom Way	SL2 6HY	Berkshire	pMartin@gmail.co																																																															
Username	Pword_hash	CustomerID																																																																				
Filter	Filter	Filter																																																																				
kiran	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824	1																																																																				
godwin	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824	2																																																																				
alexsmith	e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c	3																																																																				
george	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8	4																																																																				
7	The program me should be able to hash a password a user has entered when they sign up	I will check the database to see if the hash of the password has been stored in the table when the “sign up” button is pressed	<table><tr><th>F_Name</th><th>S_Name</th><th>DOB</th><th>obile No</th><th>Address</th><th>Postcode</th><th>County</th></tr><tr><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td></tr><tr><td>Alex</td><td>Cherian</td><td>06/02/12</td><td>07862...</td><td>7 Gaviots Way</td><td>SL3 8HY</td><td>Buckinghamshire</td></tr><tr><td>Godwin</td><td>Cherian</td><td>95/03/12</td><td>07872...</td><td>8 Gaviots Way</td><td>SL7 6YT</td><td>Buckinghamshire</td></tr><tr><td>Bob</td><td>Smith</td><td>97/04/09</td><td>07821...</td><td>4 Mumfords Road</td><td>SL6 &HG</td><td>Buckinghamshire</td></tr><tr><td>Paul</td><td>Martin</td><td>93/06/21</td><td>07823...</td><td>8 Nashdom Way</td><td>SL2 6HY</td><td>Berkshire</td></tr></table> <table><tr><th>Username</th><th>Pword_hash</th></tr><tr><td>Filter</td><td>Filter</td></tr><tr><td>kiran</td><td>2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824</td></tr><tr><td>godwin</td><td>2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824</td></tr><tr><td>alexsmith</td><td>e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c</td></tr><tr><td>george</td><td>5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8</td></tr></table>	F_Name	S_Name	DOB	obile No	Address	Postcode	County	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Alex	Cherian	06/02/12	07862...	7 Gaviots Way	SL3 8HY	Buckinghamshire	Godwin	Cherian	95/03/12	07872...	8 Gaviots Way	SL7 6YT	Buckinghamshire	Bob	Smith	97/04/09	07821...	4 Mumfords Road	SL6 &HG	Buckinghamshire	Paul	Martin	93/06/21	07823...	8 Nashdom Way	SL2 6HY	Berkshire	Username	Pword_hash	Filter	Filter	kiran	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824	godwin	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824	alexsmith	e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c	george	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8	Pass												
F_Name	S_Name	DOB	obile No	Address	Postcode	County																																																																
Filter	Filter	Filter	Filter	Filter	Filter	Filter																																																																
Alex	Cherian	06/02/12	07862...	7 Gaviots Way	SL3 8HY	Buckinghamshire																																																																
Godwin	Cherian	95/03/12	07872...	8 Gaviots Way	SL7 6YT	Buckinghamshire																																																																
Bob	Smith	97/04/09	07821...	4 Mumfords Road	SL6 &HG	Buckinghamshire																																																																
Paul	Martin	93/06/21	07823...	8 Nashdom Way	SL2 6HY	Berkshire																																																																
Username	Pword_hash																																																																					
Filter	Filter																																																																					
kiran	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824																																																																					
godwin	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824																																																																					
alexsmith	e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c																																																																					
george	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8																																																																					
8	The program me should only allow the user to “sign up” if they have a username longer than 4 characters and a password longer than 5 characters	I will enter text into the username box larger than 4 characters and text into the password box longer than 5 characters	<table><tr><th>F_Name</th><th>S_Name</th><th>DOB</th><th>obile No</th><th>Address</th><th>Postcode</th><th>County</th><th>Email</th></tr><tr><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td></tr><tr><td>Alex</td><td>Cherian</td><td>06/02/12</td><td>07862...</td><td>7 Gaviots Way</td><td>SL3 8HY</td><td>Buckinghamshire</td><td>bob@hotmail.com</td></tr><tr><td>Godwin</td><td>Cherian</td><td>95/03/12</td><td>07872...</td><td>8 Gaviots Way</td><td>SL7 6YT</td><td>Buckonghamshire</td><td>email@email.com</td></tr><tr><td>Bob</td><td>Smith</td><td>97/04/09</td><td>07821...</td><td>4 Mumfords Road</td><td>SL6 &HG</td><td>Buckinghamshire</td><td>aSmith@gmail.com</td></tr><tr><td>Paul</td><td>Martin</td><td>93/06/21</td><td>07823...</td><td>8 Nashdom Way</td><td>SL2 6HY</td><td>Berkshire</td><td>pMartin@gmail.com</td></tr></table> <table><tr><th>Username</th><th>Pword_hash</th></tr><tr><td>Filter</td><td>Filter</td></tr><tr><td>kiran</td><td>2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824</td></tr><tr><td>godwin</td><td>2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824</td></tr><tr><td>alexsmith</td><td>e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c</td></tr><tr><td>george</td><td>5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d</td></tr></table>	F_Name	S_Name	DOB	obile No	Address	Postcode	County	Email	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Alex	Cherian	06/02/12	07862...	7 Gaviots Way	SL3 8HY	Buckinghamshire	bob@hotmail.com	Godwin	Cherian	95/03/12	07872...	8 Gaviots Way	SL7 6YT	Buckonghamshire	email@email.com	Bob	Smith	97/04/09	07821...	4 Mumfords Road	SL6 &HG	Buckinghamshire	aSmith@gmail.com	Paul	Martin	93/06/21	07823...	8 Nashdom Way	SL2 6HY	Berkshire	pMartin@gmail.com	Username	Pword_hash	Filter	Filter	kiran	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824	godwin	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824	alexsmith	e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c	george	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d	Pass						
F_Name	S_Name	DOB	obile No	Address	Postcode	County	Email																																																															
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter																																																															
Alex	Cherian	06/02/12	07862...	7 Gaviots Way	SL3 8HY	Buckinghamshire	bob@hotmail.com																																																															
Godwin	Cherian	95/03/12	07872...	8 Gaviots Way	SL7 6YT	Buckonghamshire	email@email.com																																																															
Bob	Smith	97/04/09	07821...	4 Mumfords Road	SL6 &HG	Buckinghamshire	aSmith@gmail.com																																																															
Paul	Martin	93/06/21	07823...	8 Nashdom Way	SL2 6HY	Berkshire	pMartin@gmail.com																																																															
Username	Pword_hash																																																																					
Filter	Filter																																																																					
kiran	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824																																																																					
godwin	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824																																																																					
alexsmith	e08d706b3e4ce964b632746cf568913cb93f1ed36476fbb0494b80ed17c5975c																																																																					
george	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d																																																																					



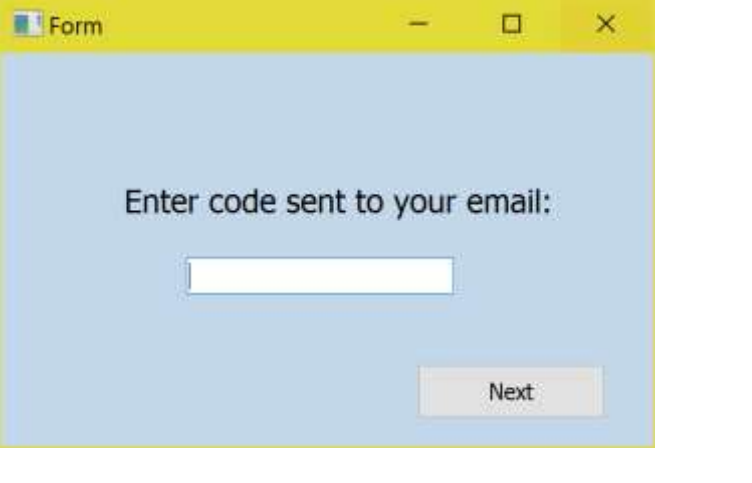
9	The program should not accept a username less than 4 characters	I will enter a username less than 4 characters in the box and press sign up		Pass
10	The program should not accept a password less than 5 characters	I will enter a password less than 5 characters and press sign up		Pass
11	The program should not accept a username if it already exists in the database	I will enter a username that already exists in the database		Pass
12	The program shouldn't allow a user to sign up if they haven't filled out the required fields	I will click the "sign up" button without entering any data into the text entries		Partial Pass

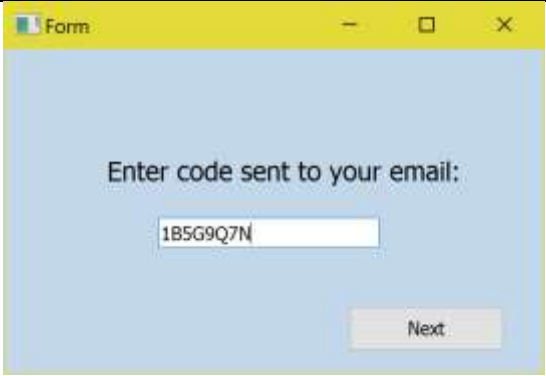


13	If the user enters extreme data into the program me, will it save to the database	I will enter extreme values into the text entries and click the “sign up” button		Pass
14	Return the user to the previous window when the back button is pressed	I will click the “back” button		Pass

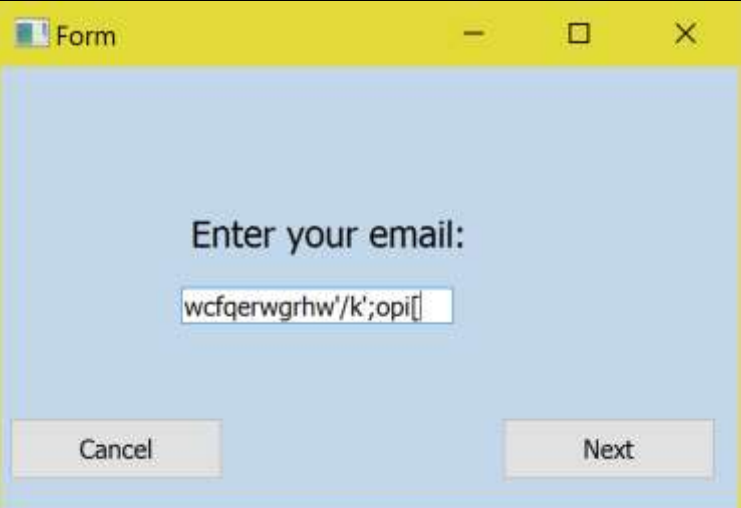
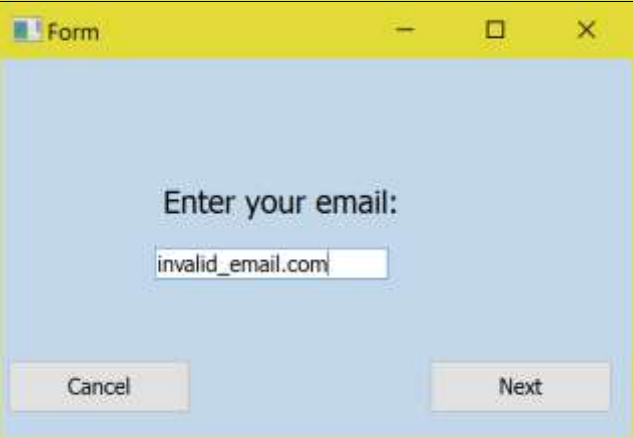

Changing/Forgot Password

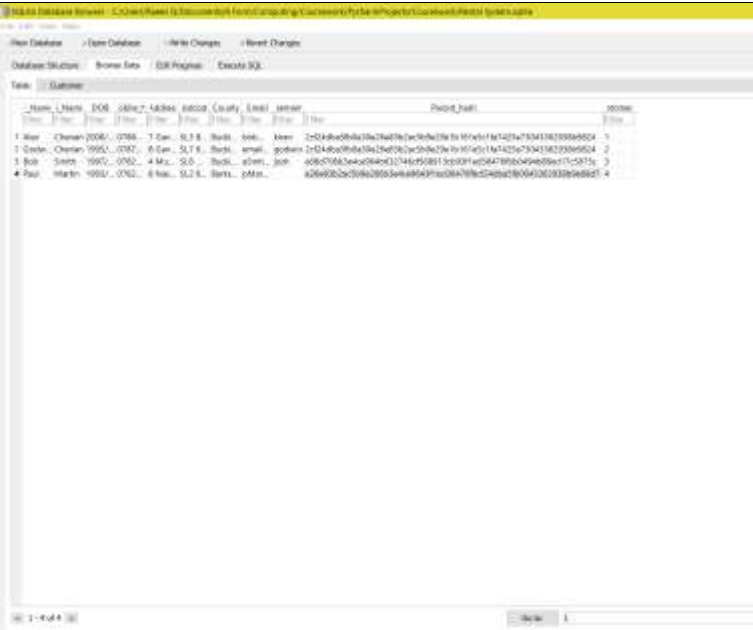
Test Number (C)	What should it do (success criteria)?	How will I Test it?	Screen Shot of Testing	Successful
1	The programme should allow the user to change password	I will click the "forgot password" button on the login window. A new window should then open.		Pass
2	Each button displayed on the window should carry out a function similar to the description of the button.	I will press the "cancel" button on the window.		Pass

3	The programme should prompt the user to enter their email	I will enter a valid email into the text box		Pass
4	The programme should send an email to the account they have entered	I will click the "next" button and then login into the email address entered to see if there is an email from PET there		Pass
5	The email should be from the PET company and have a professional format	I will click the email and see if there is a subject and a professional format		Pass


6	The programme should generate a random code	I will compare 2 emails and check to see if the code is different in each		Pass
7	The code should contain a mixture of letters and numbers	I will look at a code that has been sent via email and check if it contains letters and numbers		Pass
8	The programme should then open a new window	I will click the "next" button and check to see if a new window has opened		Pass

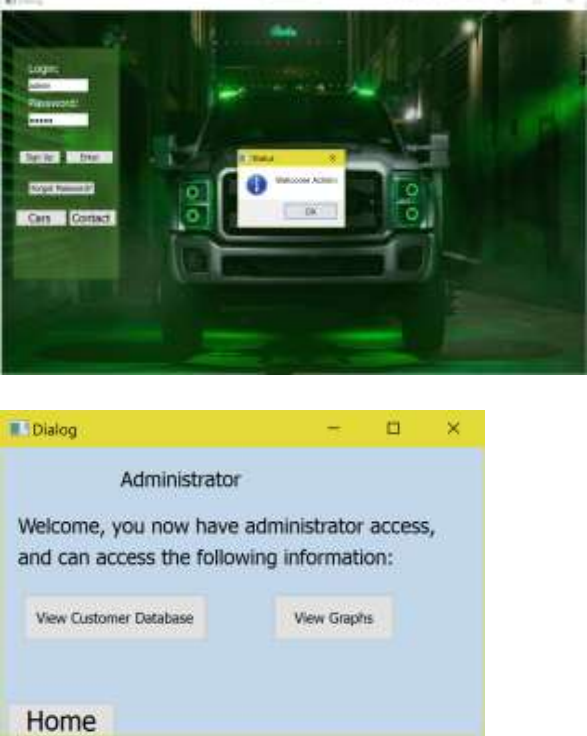

9	The code should then be copied back into the new window opened by the programme	I will copy the code that has been sent via email into the box		Pass
10	A new window should open	I will click the "next" button		Pass
11	The user should be able to return to the previous window when the cancel button is clicked	I will click the cancel button		Pass

12	The user should not be able to enter extreme data into the data entry	I will try and enter extreme data into the text entry		Fail
13	The user should not be able to enter an invalid email	I will try and enter an invalid email address into the text entry		Fail
14	Allow the user to enter a new password	I will enter a new password into the text entry		Pass

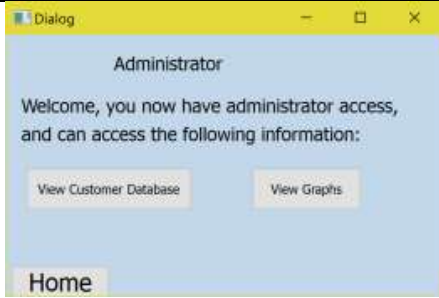
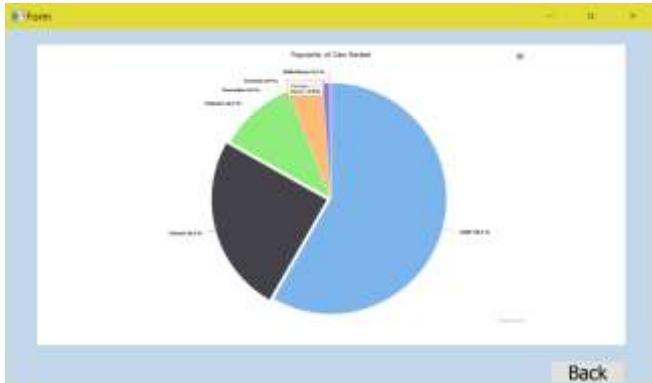
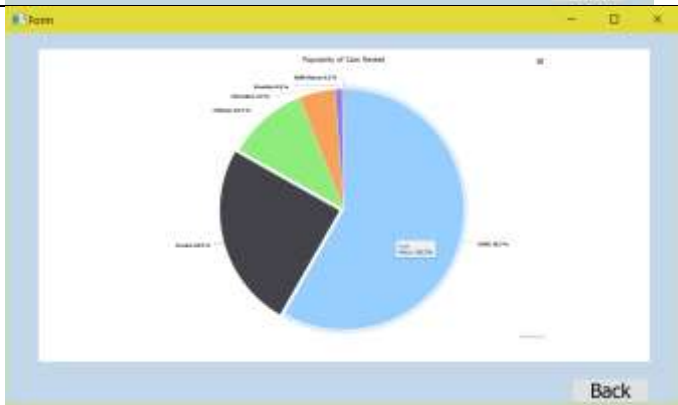

				

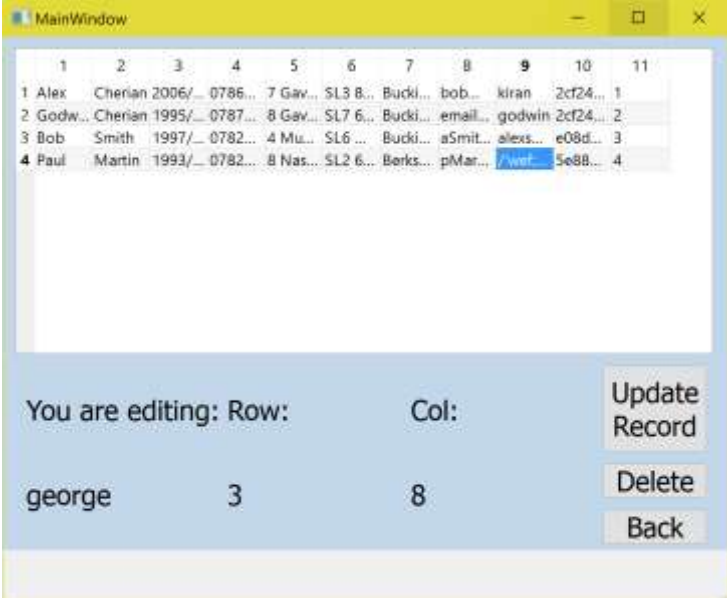
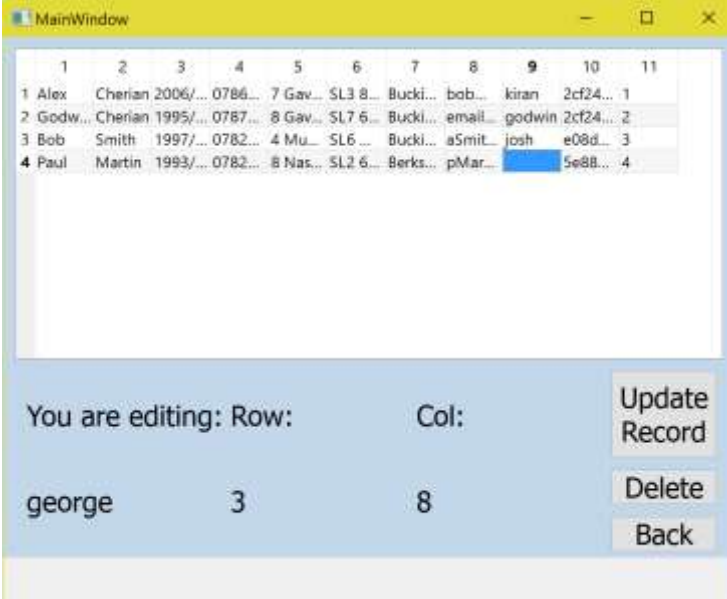
Admin:

Test Number (A)	What should it do (success criteria) ?	How will I Test it?	Screen Shot of Testing	Successful
1	Depending on the type of account the user signs in with, they should have access to different options (Admin)	I will open the login window and enter the admin username and password into the text boxes		Pass

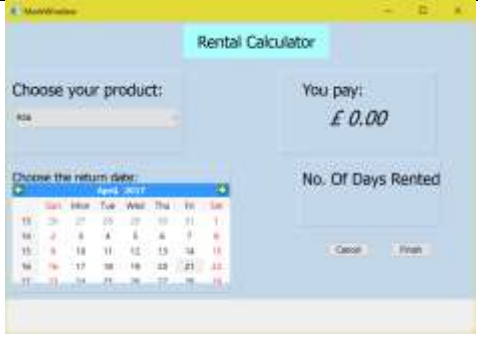
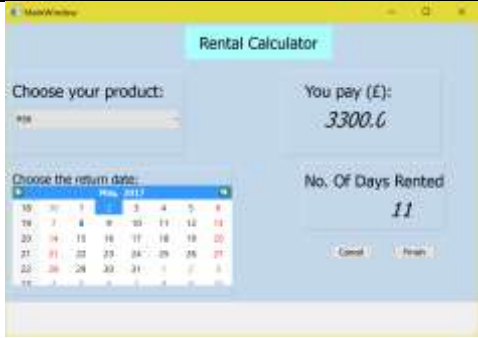

2	A new window should appear giving the admin different options dependent on what they want to view	I will click the enter button		Pass
3	If the admin details are incorrect, the new window should not open, instead a message box alerting the user that they have entered incorrect details	I will enter incorrect admin details into the text boxes on the login window		Pass

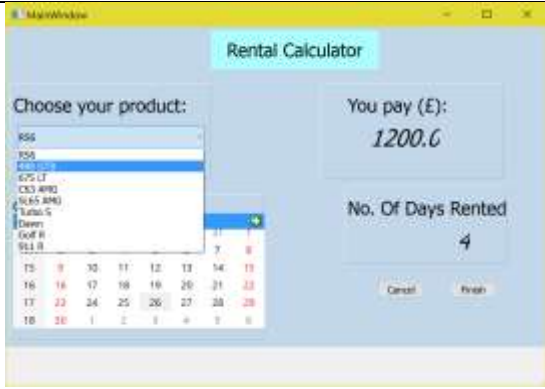
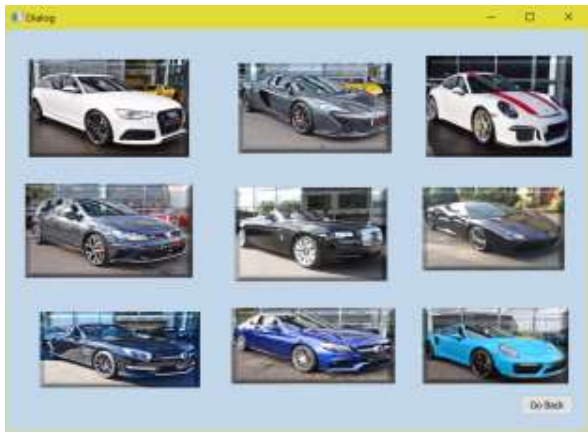
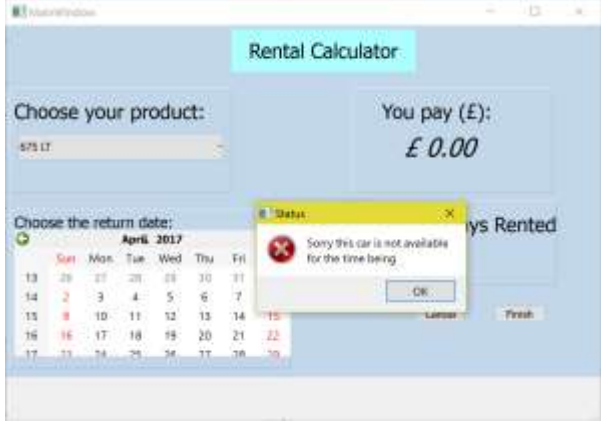
4	Allow only the administrator to make changes to user details	I will click the “customer database” button . A new window should open. I will then click a cell and change the contents of it	<div><div>Dialog</div><div>Administrator</div><div>Welcome, you now have administrator access, and can access the following information:</div><div><div>View Customer Database</div><div>View Graphs</div></div><div>Home</div></div> <div><div>MainWindow</div><div>MainWindow</div><table><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td></tr><tr><td>1</td><td>Alex</td><td>Cherian</td><td>2006/.. 0786..</td><td>7</td><td>Gav..</td><td>godw..</td><td>Bucki</td><td>bob..</td><td>kiran</td><td>2cf24.. 1</td></tr><tr><td>2</td><td>Godw..</td><td>Cherian</td><td>1995/.. 0787..</td><td>8</td><td>Gav..</td><td>SL7 6..</td><td>Bucki</td><td>email..</td><td>godwin</td><td>2cf24.. 2</td></tr><tr><td>3</td><td>Bob</td><td>Smith</td><td>97/04.. 0782..</td><td>4</td><td>Mu..</td><td>SL6 ..</td><td>Bucki</td><td>aSmit..</td><td>josh</td><td>e08d.. 3</td></tr></table><div><div>You are editing: Row:</div><div>Col:</div><div>Update Record</div><div>Delete</div><div>Back</div></div><div>kiran08</div></div>	1	2	3	4	5	6	7	8	9	10	11	1	Alex	Cherian	2006/.. 0786..	7	Gav..	godw..	Bucki	bob..	kiran	2cf24.. 1	2	Godw..	Cherian	1995/.. 0787..	8	Gav..	SL7 6..	Bucki	email..	godwin	2cf24.. 2	3	Bob	Smith	97/04.. 0782..	4	Mu..	SL6 ..	Bucki	aSmit..	josh	e08d.. 3	Pass
1	2	3	4	5	6	7	8	9	10	11																																						
1	Alex	Cherian	2006/.. 0786..	7	Gav..	godw..	Bucki	bob..	kiran	2cf24.. 1																																						
2	Godw..	Cherian	1995/.. 0787..	8	Gav..	SL7 6..	Bucki	email..	godwin	2cf24.. 2																																						
3	Bob	Smith	97/04.. 0782..	4	Mu..	SL6 ..	Bucki	aSmit..	josh	e08d.. 3																																						
5	The information should be updated in the database	I will open the “customer” table in the database and check if the information has been updated	<table><tr><td>ostcod</td><td>County</td><td>Email</td><td>sernal</td></tr><tr><td>Filter</td><td>Filter</td><td>Filter</td><td>Filter</td></tr><tr><td>SL3 8...</td><td>Bucki...</td><td>bob...</td><td>kiran</td></tr><tr><td>SL7 6...</td><td>Bucki...</td><td>email...</td><td>godwi</td></tr><tr><td>SL6 ...</td><td>Bucki...</td><td>aSmit...</td><td>josh</td></tr></table>	ostcod	County	Email	sernal	Filter	Filter	Filter	Filter	SL3 8...	Bucki...	bob...	kiran	SL7 6...	Bucki...	email...	godwi	SL6 ...	Bucki...	aSmit...	josh	Pass																								
ostcod	County	Email	sernal																																													
Filter	Filter	Filter	Filter																																													
SL3 8...	Bucki...	bob...	kiran																																													
SL7 6...	Bucki...	email...	godwi																																													
SL6 ...	Bucki...	aSmit...	josh																																													
6	Allow only the admin to delete user accounts	I will try to delete an account		Fail																																												



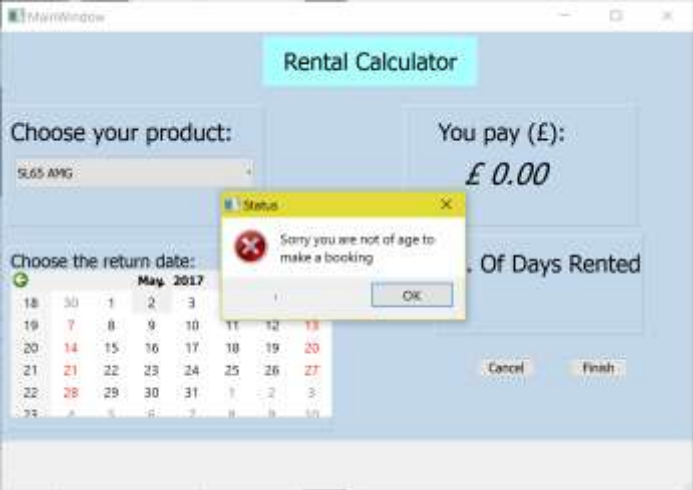
7	If the user signs in with the admin the programme will display graphs to show the most popular rented car	I will click the "graphs" button on the admin window	 	Pass
8	When the user hovers over a "slice" the "slice" should enlarge slightly	I will hover the mouse over the slice		Pass
9	Will the programme allow the admin to login with only the username correct	I will only enter the correct username for the admin, without the correct password		Pass

10	If admin enters in extreme data	I will enter in extreme data into the cell		Fail
11	If the admin enters null value into the cell	I will click a cell and not enter data into it		Fail

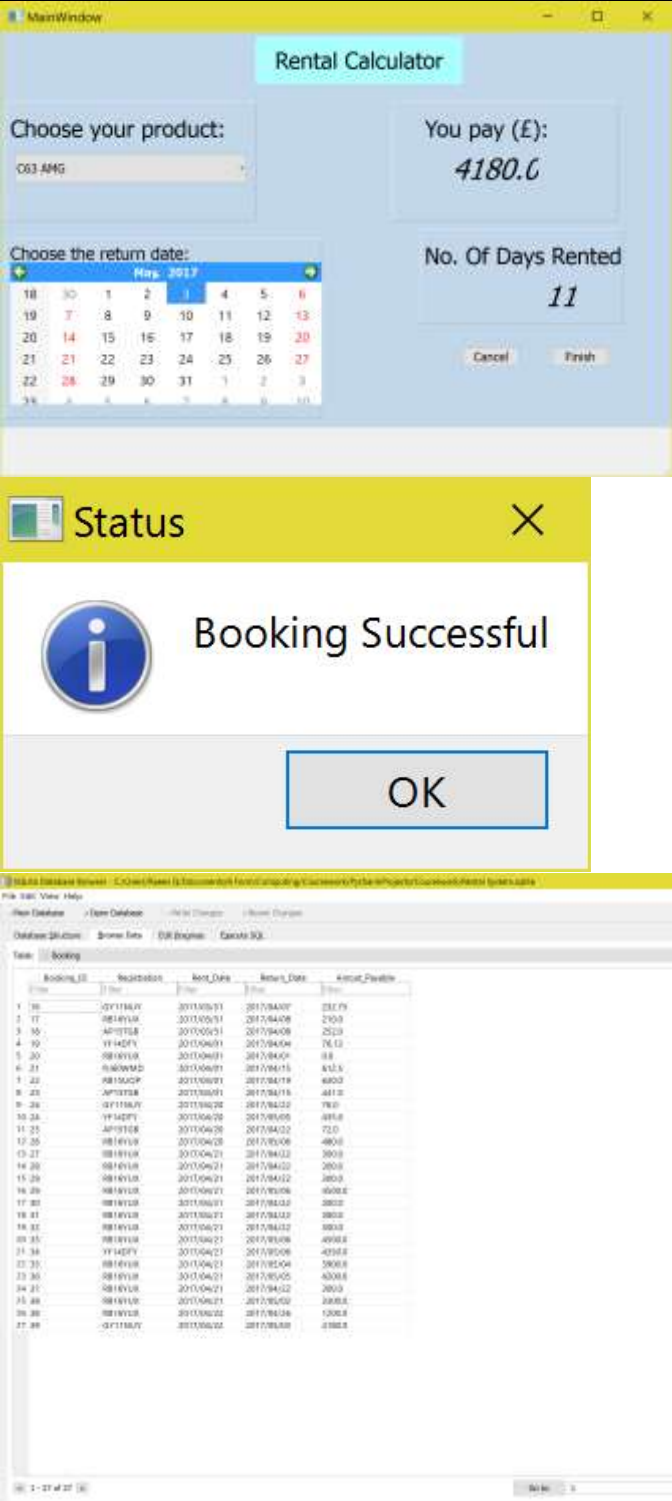
Booking:

Test Numb er (B)	What should it do (succe ss criteri a)?	How will I Test it?	Screen Shot of Testing	Succe ssful
1	The user should be able to delete their booking by signing back into the programme	I will try and delete a booking made as a customer from the programme		Fail
2	The user should be able to view how long they have hired a car for	I will try and view how long I have hired a car for by selecting a car and entering a date from the calendar		Pass
3	The user should be able to change how long they have	I will try and edit the amount of days I have hired the car for		Pass

	hired a car for			
4	The programme should be able to give the customer a view of all the cars available	I will click the drop-down list I will also open the cars window which displays a picture of each car PET offers	 	Pass
5	The programme should prevent the customer from making a booking if there aren't cars available	I will select a car that has already been hired by another customer and select a date they have hired it till		Pass

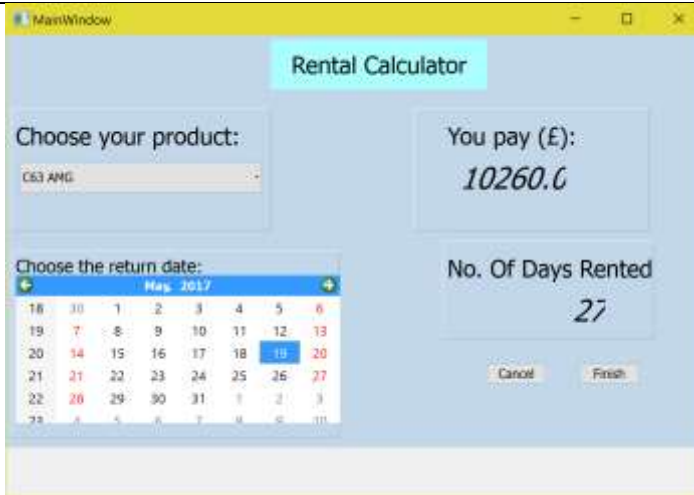
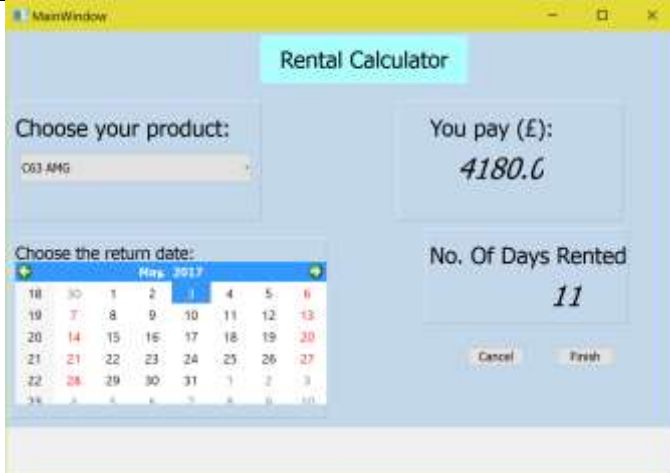
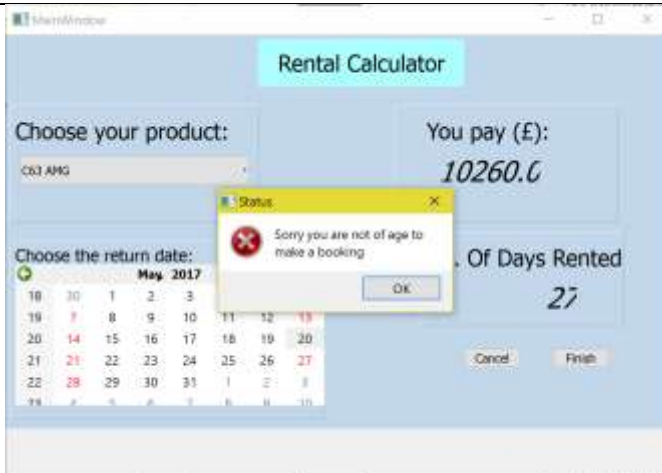
6	<p>The programme should prevent a customer from making a booking if they are below the age of 21</p>	<p>I will login into the programme with a user who is below the age of 21 and try to hire a car</p>	<div><p>The screenshot shows a login window titled 'Login' with fields for 'Login:' (username 'user') and 'Password:' (masked with asterisks). It includes 'Sign Up' and 'Enter' buttons, a 'Forgot Password?' link, and 'Cars' and 'Contact' buttons at the bottom. The background is a dark image of a car with green lights.</p></div> <div><p>This screenshot shows the same login window, but with a 'Status' dialog box overlaid. The dialog box contains an information icon and the text 'Information entered correctly' with an 'OK' button.</p></div> <div><p>The screenshot shows the 'MainWindow' titled 'Rental Calculator'. It has a 'Choose your product:' dropdown set to 'SL65 AMG', a 'Choose the return date:' calendar for May 2017, and a 'You pay (£): £ 0.00' display. A 'Status' dialog box with a red error icon is overlaid, stating 'Sorry you are not of age to make a booking'. There are 'Cancel' and 'Finish' buttons at the bottom right.</p></div>	Pass
---	--	---	--	------

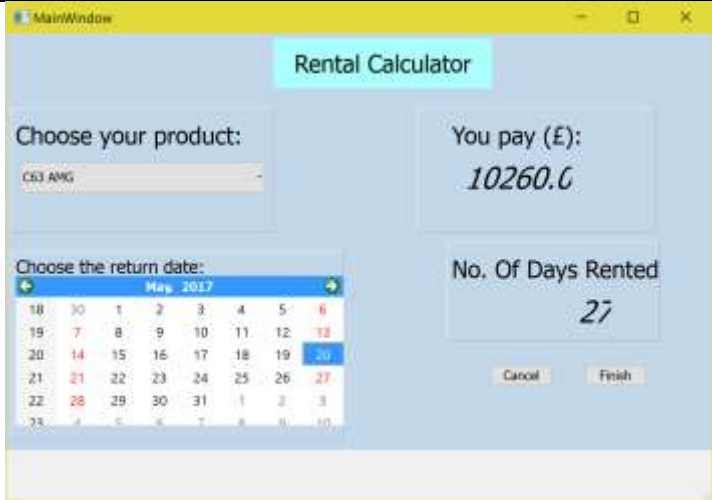
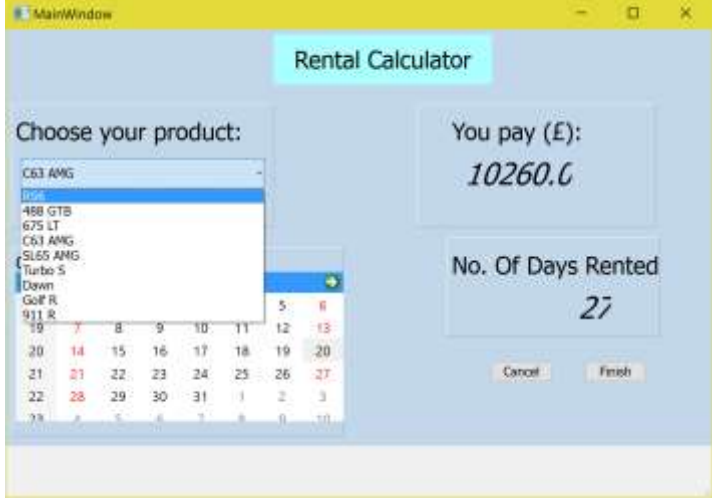
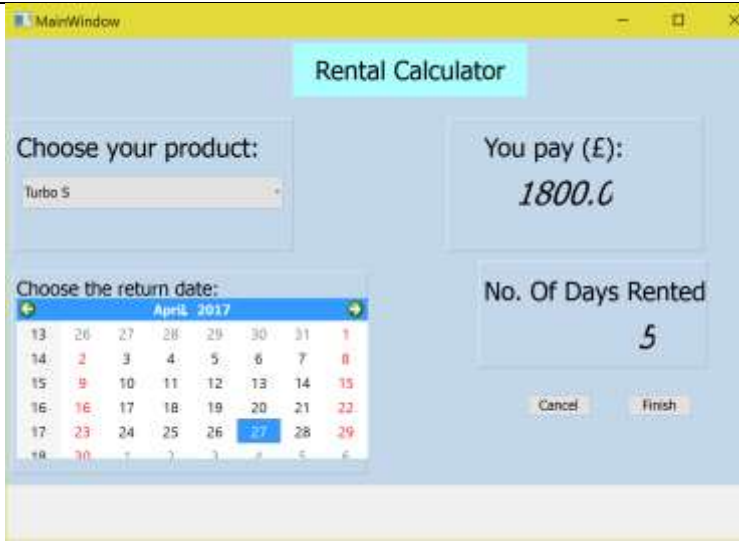
7	The programme should calculate the age of the user when they input their DOB	I will check this by trying to make a booking under the age of 21, and a booking over the age of 21	<div data-bbox="555 197 1257 683"> <p>Rental Calculator</p> <p>Choose your product: SL65 AMG</p> <p>Choose the return date: May, 2017</p> <p>You pay (£): £ 0.00</p> <p>No. Of Days Rented</p> <p>Cancel Finish</p> <p>Status: Sorry you are not of age to make a booking</p> </div> <div data-bbox="555 757 1235 1227"> <p>Rental Calculator</p> <p>Choose your product: C63 AMG</p> <p>Choose the return date: May, 2017</p> <p>You pay (£): 4180.0</p> <p>No. Of Days Rented 11</p> <p>Cancel Finish</p> </div> <div data-bbox="555 1265 1203 1774"> <p>Status</p> <p>Booking Successful</p> <p>OK</p> <p>39 GY11NUY 2017/04/22</p> <p>2017/05/03 4180.0</p> </div>	Pass
---	--	---	--	------


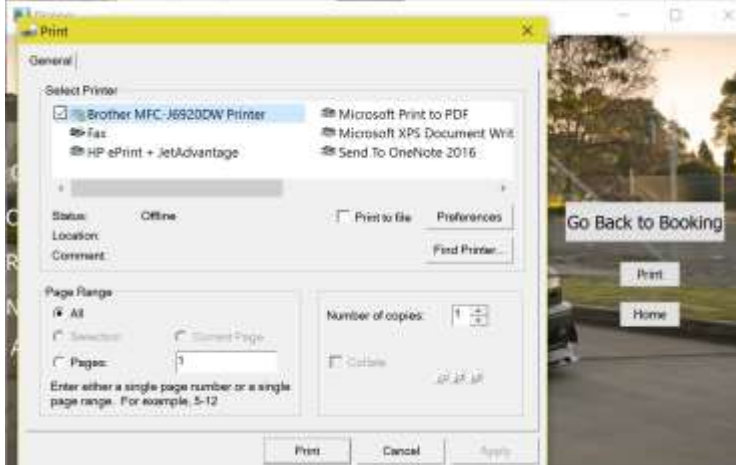
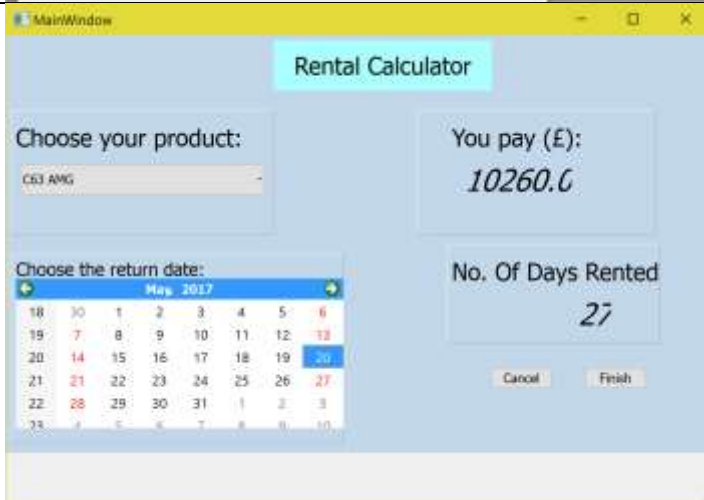
8	The programme should save the details of the booking to the database	I will make a booking as a customer, and then check to see whether the same details have been stored in the database	 <p>The screenshot shows the 'Rental Calculator' application interface. It includes a 'Choose your product:' dropdown menu with 'C63 AMG' selected, a 'Choose the return date:' calendar for May 2017, a 'You pay (£): 4180.6' display, and a 'No. Of Days Rented: 11' display. A 'Status' dialog box is overlaid, showing 'Booking Successful' with an 'OK' button. Below the dialog, a screenshot of a database table named 'Booking' is shown, containing 27 rows of booking data with columns: Booking_ID, Registration, Rent_Date, Return_Date, and Amount_Payable.</p> <table border="1"> <thead> <tr> <th>Booking_ID</th> <th>Registration</th> <th>Rent_Date</th> <th>Return_Date</th> <th>Amount_Payable</th> </tr> </thead> <tbody> <tr><td>1</td><td>Q7118L/J</td><td>2017/05/01</td><td>2017/05/07</td><td>232.79</td></tr> <tr><td>2</td><td>SB14V/L</td><td>2017/05/01</td><td>2017/05/08</td><td>210.9</td></tr> <tr><td>3</td><td>AP19T/L</td><td>2017/05/01</td><td>2017/05/08</td><td>252.9</td></tr> <tr><td>4</td><td>YF14Z/P</td><td>2017/05/01</td><td>2017/05/04</td><td>74.12</td></tr> <tr><td>5</td><td>SB14V/L</td><td>2017/05/01</td><td>2017/05/04</td><td>8.8</td></tr> <tr><td>6</td><td>R60W/M</td><td>2017/05/01</td><td>2017/05/15</td><td>612.5</td></tr> <tr><td>7</td><td>SB15G/P</td><td>2017/05/01</td><td>2017/05/19</td><td>600.9</td></tr> <tr><td>8</td><td>AP19T/L</td><td>2017/05/01</td><td>2017/05/19</td><td>641.9</td></tr> <tr><td>9</td><td>Q7118L/J</td><td>2017/05/04</td><td>2017/05/22</td><td>78.9</td></tr> <tr><td>10</td><td>YF14Z/P</td><td>2017/05/08</td><td>2017/05/08</td><td>88.8</td></tr> <tr><td>11</td><td>AP19T/L</td><td>2017/05/08</td><td>2017/05/22</td><td>72.0</td></tr> <tr><td>12</td><td>SB14V/L</td><td>2017/05/08</td><td>2017/05/26</td><td>480.8</td></tr> <tr><td>13</td><td>SB14V/L</td><td>2017/05/21</td><td>2017/05/22</td><td>380.9</td></tr> <tr><td>14</td><td>SB14V/L</td><td>2017/05/21</td><td>2017/05/22</td><td>280.9</td></tr> <tr><td>15</td><td>SB14V/L</td><td>2017/05/21</td><td>2017/05/22</td><td>280.9</td></tr> <tr><td>16</td><td>SB14V/L</td><td>2017/05/21</td><td>2017/05/26</td><td>480.8</td></tr> <tr><td>17</td><td>SB14V/L</td><td>2017/05/21</td><td>2017/05/22</td><td>380.9</td></tr> <tr><td>18</td><td>SB14V/L</td><td>2017/05/21</td><td>2017/05/22</td><td>380.9</td></tr> <tr><td>19</td><td>SB14V/L</td><td>2017/05/21</td><td>2017/05/22</td><td>380.9</td></tr> <tr><td>20</td><td>SB14V/L</td><td>2017/05/21</td><td>2017/05/26</td><td>480.8</td></tr> <tr><td>21</td><td>SB14V/L</td><td>2017/05/21</td><td>2017/05/26</td><td>480.8</td></tr> <tr><td>22</td><td>SB14V/L</td><td>2017/05/21</td><td>2017/05/26</td><td>480.8</td></tr> <tr><td>23</td><td>SB14V/L</td><td>2017/05/21</td><td>2017/05/26</td><td>480.8</td></tr> <tr><td>24</td><td>SB14V/L</td><td>2017/05/21</td><td>2017/05/22</td><td>280.9</td></tr> <tr><td>25</td><td>SB14V/L</td><td>2017/05/21</td><td>2017/05/22</td><td>280.9</td></tr> <tr><td>26</td><td>SB14V/L</td><td>2017/05/22</td><td>2017/05/26</td><td>1200.8</td></tr> <tr><td>27</td><td>Q7118L/J</td><td>2017/05/22</td><td>2017/05/26</td><td>1188.8</td></tr> </tbody> </table>	Booking_ID	Registration	Rent_Date	Return_Date	Amount_Payable	1	Q7118L/J	2017/05/01	2017/05/07	232.79	2	SB14V/L	2017/05/01	2017/05/08	210.9	3	AP19T/L	2017/05/01	2017/05/08	252.9	4	YF14Z/P	2017/05/01	2017/05/04	74.12	5	SB14V/L	2017/05/01	2017/05/04	8.8	6	R60W/M	2017/05/01	2017/05/15	612.5	7	SB15G/P	2017/05/01	2017/05/19	600.9	8	AP19T/L	2017/05/01	2017/05/19	641.9	9	Q7118L/J	2017/05/04	2017/05/22	78.9	10	YF14Z/P	2017/05/08	2017/05/08	88.8	11	AP19T/L	2017/05/08	2017/05/22	72.0	12	SB14V/L	2017/05/08	2017/05/26	480.8	13	SB14V/L	2017/05/21	2017/05/22	380.9	14	SB14V/L	2017/05/21	2017/05/22	280.9	15	SB14V/L	2017/05/21	2017/05/22	280.9	16	SB14V/L	2017/05/21	2017/05/26	480.8	17	SB14V/L	2017/05/21	2017/05/22	380.9	18	SB14V/L	2017/05/21	2017/05/22	380.9	19	SB14V/L	2017/05/21	2017/05/22	380.9	20	SB14V/L	2017/05/21	2017/05/26	480.8	21	SB14V/L	2017/05/21	2017/05/26	480.8	22	SB14V/L	2017/05/21	2017/05/26	480.8	23	SB14V/L	2017/05/21	2017/05/26	480.8	24	SB14V/L	2017/05/21	2017/05/22	280.9	25	SB14V/L	2017/05/21	2017/05/22	280.9	26	SB14V/L	2017/05/22	2017/05/26	1200.8	27	Q7118L/J	2017/05/22	2017/05/26	1188.8	Pass
Booking_ID	Registration	Rent_Date	Return_Date	Amount_Payable																																																																																																																																												
1	Q7118L/J	2017/05/01	2017/05/07	232.79																																																																																																																																												
2	SB14V/L	2017/05/01	2017/05/08	210.9																																																																																																																																												
3	AP19T/L	2017/05/01	2017/05/08	252.9																																																																																																																																												
4	YF14Z/P	2017/05/01	2017/05/04	74.12																																																																																																																																												
5	SB14V/L	2017/05/01	2017/05/04	8.8																																																																																																																																												
6	R60W/M	2017/05/01	2017/05/15	612.5																																																																																																																																												
7	SB15G/P	2017/05/01	2017/05/19	600.9																																																																																																																																												
8	AP19T/L	2017/05/01	2017/05/19	641.9																																																																																																																																												
9	Q7118L/J	2017/05/04	2017/05/22	78.9																																																																																																																																												
10	YF14Z/P	2017/05/08	2017/05/08	88.8																																																																																																																																												
11	AP19T/L	2017/05/08	2017/05/22	72.0																																																																																																																																												
12	SB14V/L	2017/05/08	2017/05/26	480.8																																																																																																																																												
13	SB14V/L	2017/05/21	2017/05/22	380.9																																																																																																																																												
14	SB14V/L	2017/05/21	2017/05/22	280.9																																																																																																																																												
15	SB14V/L	2017/05/21	2017/05/22	280.9																																																																																																																																												
16	SB14V/L	2017/05/21	2017/05/26	480.8																																																																																																																																												
17	SB14V/L	2017/05/21	2017/05/22	380.9																																																																																																																																												
18	SB14V/L	2017/05/21	2017/05/22	380.9																																																																																																																																												
19	SB14V/L	2017/05/21	2017/05/22	380.9																																																																																																																																												
20	SB14V/L	2017/05/21	2017/05/26	480.8																																																																																																																																												
21	SB14V/L	2017/05/21	2017/05/26	480.8																																																																																																																																												
22	SB14V/L	2017/05/21	2017/05/26	480.8																																																																																																																																												
23	SB14V/L	2017/05/21	2017/05/26	480.8																																																																																																																																												
24	SB14V/L	2017/05/21	2017/05/22	280.9																																																																																																																																												
25	SB14V/L	2017/05/21	2017/05/22	280.9																																																																																																																																												
26	SB14V/L	2017/05/22	2017/05/26	1200.8																																																																																																																																												
27	Q7118L/J	2017/05/22	2017/05/26	1188.8																																																																																																																																												

9	The programme should limit the number of decimal places calculations are done to	I will make a booking and check to see if the number of decimal places are to 2dp Also, shown in code development		Pass
10	The programme should display prices dependent on the age of the customer	I will make a booking with a user over the age of 21, but with different ages		Fail

11	The programme needs to consider the amount of days the car has been rented for	I will make a booking of the same car with different amount of days selected	<div></div> <div></div>	Pass																																																																													
12	The programme needs to consider the different prices the different cars will have	I will check to see if each car has a different value depending on the value of the car	<table><thead><tr><th>Registration</th><th>Make</th><th>Model</th><th>availat</th><th>Avai</th><th>nditi</th><th>Price</th></tr><tr><th>Filter</th><th>Filter</th><th>Filter</th><th>Filter</th><th>Filter</th><th>Filter</th><th>Filter</th></tr></thead><tbody><tr><td>OY17BNV</td><td>Ferrari</td><td>488 GTB</td><td></td><td></td><td></td><td>1400</td></tr><tr><td>YF14DTY</td><td>Rolls Royce</td><td>Dawn</td><td></td><td></td><td></td><td>1450</td></tr><tr><td>RB16YUX</td><td>Audi</td><td>RS6</td><td></td><td></td><td></td><td>1500</td></tr><tr><td>OY13KLI</td><td>Porche</td><td>Turbo S</td><td></td><td></td><td></td><td>1800</td></tr><tr><td>AP15TGB</td><td>Porsche</td><td>911 R</td><td></td><td></td><td></td><td>1800</td></tr><tr><td>GY11NUY</td><td>Mercedes</td><td>C63 AMG</td><td></td><td></td><td></td><td>1900</td></tr><tr><td>RB15UOP</td><td>Mercedes</td><td>SL65 AMG</td><td></td><td></td><td></td><td>2000</td></tr><tr><td>GG12POL</td><td>VW</td><td>Golf R</td><td></td><td></td><td></td><td>2200</td></tr><tr><td>RJ60WMD</td><td>Mclaren</td><td>675 LT</td><td></td><td></td><td></td><td>2500</td></tr></tbody></table>	Registration	Make	Model	availat	Avai	nditi	Price	Filter	Filter	Filter	Filter	Filter	Filter	Filter	OY17BNV	Ferrari	488 GTB				1400	YF14DTY	Rolls Royce	Dawn				1450	RB16YUX	Audi	RS6				1500	OY13KLI	Porche	Turbo S				1800	AP15TGB	Porsche	911 R				1800	GY11NUY	Mercedes	C63 AMG				1900	RB15UOP	Mercedes	SL65 AMG				2000	GG12POL	VW	Golf R				2200	RJ60WMD	Mclaren	675 LT				2500	Pass
Registration	Make	Model	availat	Avai	nditi	Price																																																																											
Filter	Filter	Filter	Filter	Filter	Filter	Filter																																																																											
OY17BNV	Ferrari	488 GTB				1400																																																																											
YF14DTY	Rolls Royce	Dawn				1450																																																																											
RB16YUX	Audi	RS6				1500																																																																											
OY13KLI	Porche	Turbo S				1800																																																																											
AP15TGB	Porsche	911 R				1800																																																																											
GY11NUY	Mercedes	C63 AMG				1900																																																																											
RB15UOP	Mercedes	SL65 AMG				2000																																																																											
GG12POL	VW	Golf R				2200																																																																											
RJ60WMD	Mclaren	675 LT				2500																																																																											

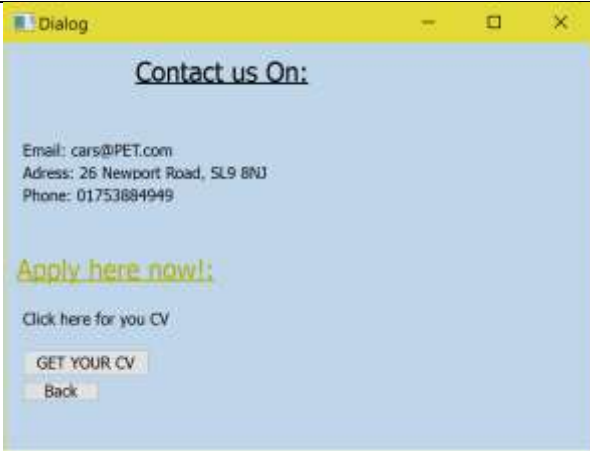
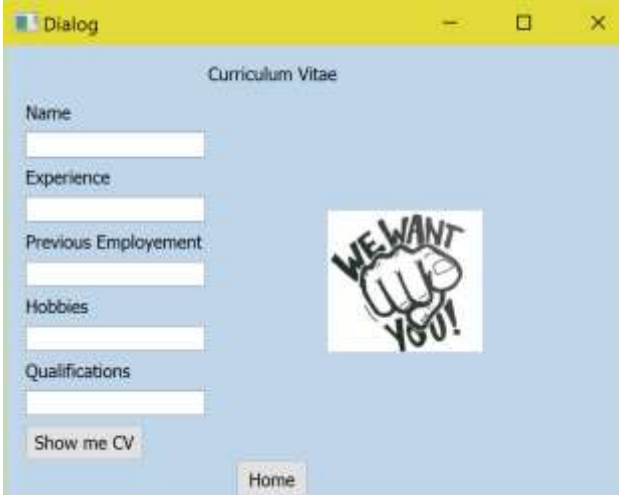

13	The programme should provide a discount for corporate customers	I will login with an account holder who is also a business owner and make a booking	 <p>The screenshot shows the 'Rental Calculator' window. Under 'Choose your product:', 'C63 AMG' is selected. Under 'Choose the return date:', a calendar for May 2017 shows the start date as 18 and the end date as 19. The calculated 'You pay (£):' is 10260.6 and 'No. Of Days Rented' is 27. 'Cancel' and 'Finish' buttons are at the bottom right.</p>	Fail
14	The programme should provide discounts for returning customers	I will make a booking with a customer account who has made a booking previously	 <p>The screenshot shows the 'Rental Calculator' window. Under 'Choose your product:', 'C63 AMG' is selected. Under 'Choose the return date:', a calendar for May 2017 shows the start date as 18 and the end date as 19. The calculated 'You pay (£):' is 4180.6 and 'No. Of Days Rented' is 11. 'Cancel' and 'Finish' buttons are at the bottom right.</p>	Fail
15	The programme should display similar alternatives for the customer if they cannot make a booking	I will make a booking that will fail	 <p>The screenshot shows the 'Rental Calculator' window with the same booking details as the previous tests. However, a 'Status' dialog box is overlaid in the center, displaying a red error icon and the message: 'Sorry you are not of age to make a booking'. The 'OK' button is visible on the dialog box.</p>	Fail


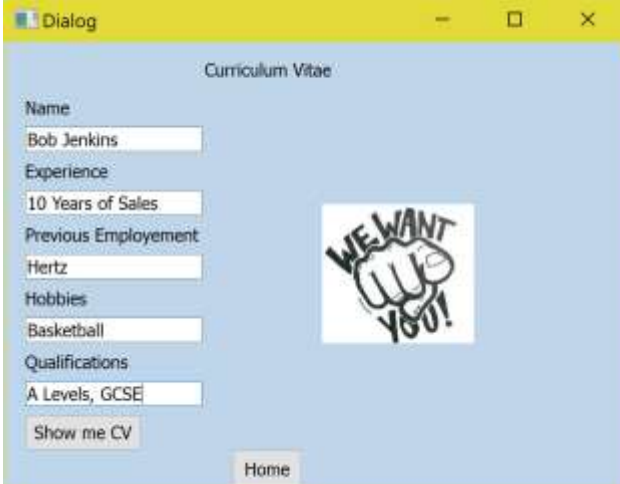
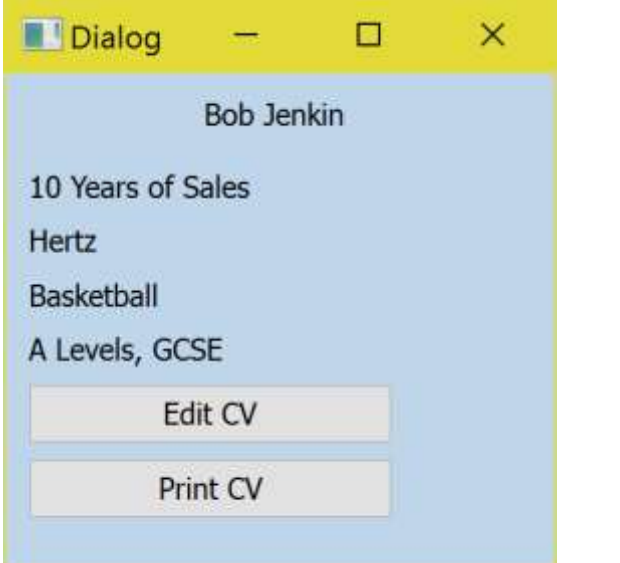
				
16	The programme should allow the user to filter through cars from a drop down list	I will make a booking and choose from a selection of cars from a drop down list		Pass
17	The programme should be able to provide a receipt of the transaction the customer may have completed	I will make a booking. The next window should display the content of my booking		Pass

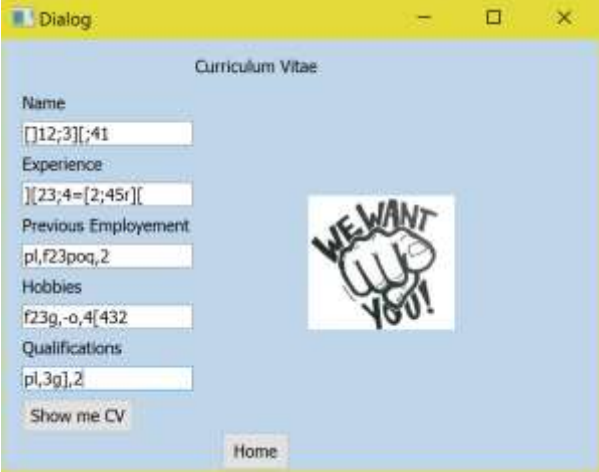
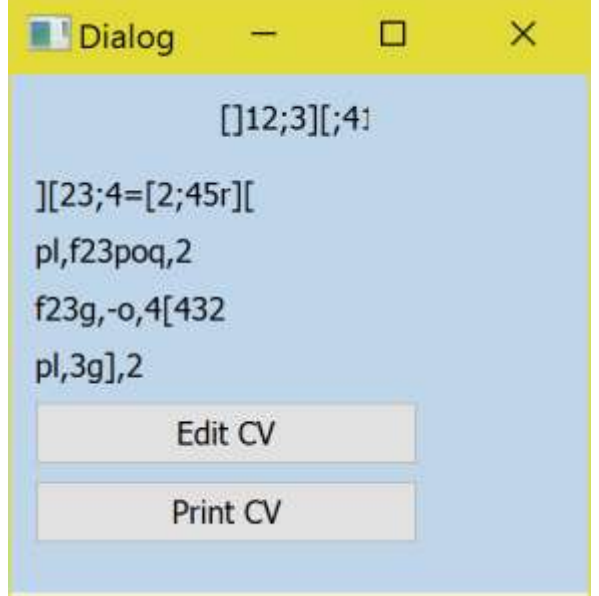

				
18	The programme should allow the customer to print off a receipt	I will print the receipt for the booking I have completed by clicking the "print" button		Pass
19	The programme should price surge. This is when there are peak rental times, so the price of the car that is being rented will increase due to demand	I will click the most popular car being rented and see if the price for the rental is more		Fail

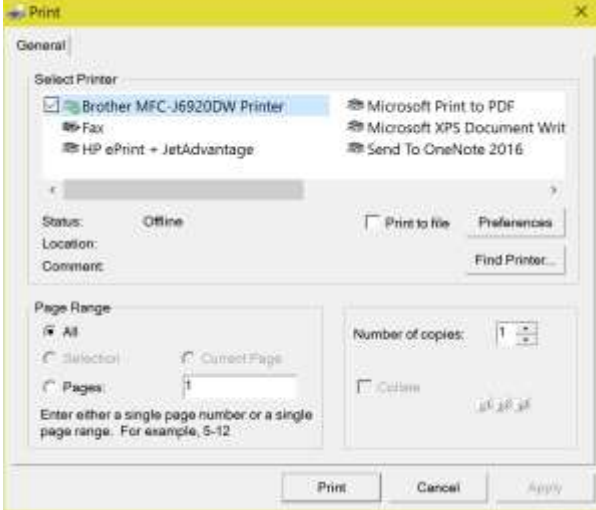
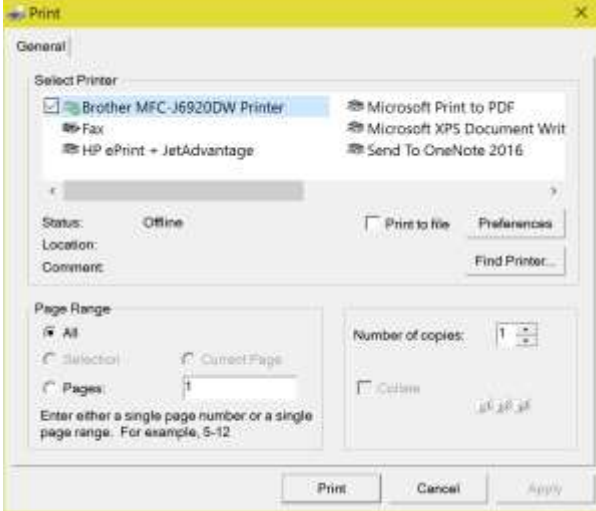
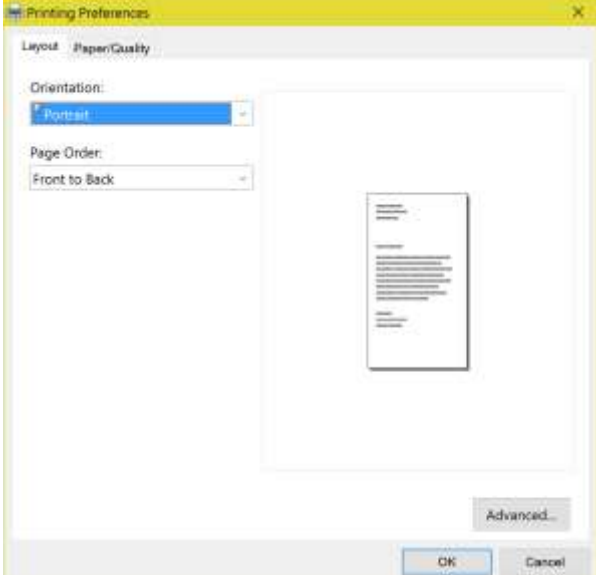
<p>The programme should n't allow a booking to be made if the user clicks a date before the current date</p>	<p>I will click a date on the calendar before today's date</p>			
<p>20</p>	<p>If the user wants to cancel booking midway through</p>	<p>I will click the cancel button</p>		<p>Pass</p>

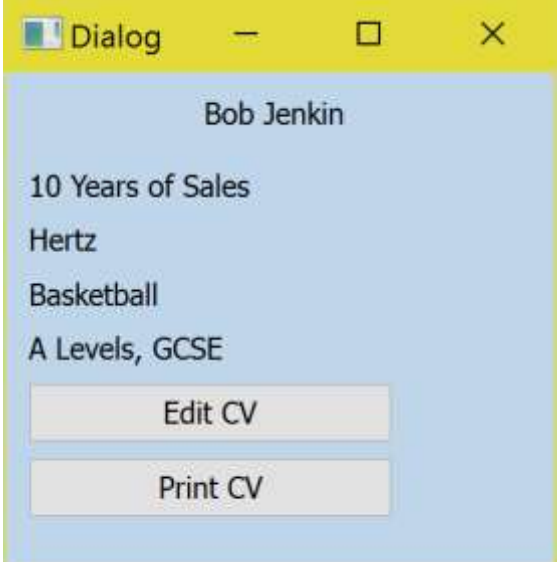
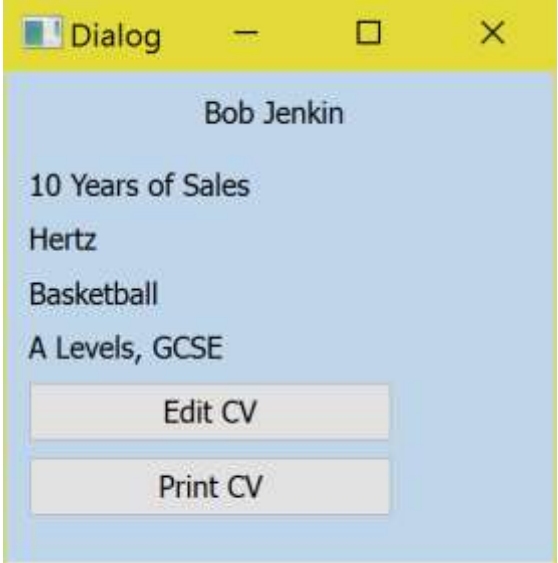
Contacts us

Test Number (G)	What should it do (success criteria) (success criteria)?	How will I Test it?	Screen Shot of Testing	Successful
1	The contact window should open when the button is clicked on the home window	I will click the contact us button		Yes
2	The cv window should open when the button is clicked	I will click the cv button		Pass
3	The cv window should return to the home window when the "back" button is clicked	I will click the back button		Pass

4	The user should be returned to the home window when they click the home button	I will click the home button		Pass
5	The user should be able to enter text into the different text entries	I will enter text into the text entries		Pass
6	The user should press the show me CV button and a new window will open, displaying the contents of what the user had entered	I will press the button		Pass

7	The programme should not take extreme values	I will enter extreme values into the text entries	 	Fail
8	The user should be able to change the details they have entered into the programme	I will click the edit cv button		Pass

9	The programme should allow the user to print their CV	I will click the Print CV button		Pass
10	The user should be able to select a printer they want to send the document to	I will click a printer		Pass
11	The user should be able to edit their preferences on the printer window	I will click the preferences button		Pass

12	The user should be able to cancel the printing action	I will click the cancel button		Pass
13	The user should be able to print preview	I will click the print button on the print window		Fail

Now that the programme was complete, Godwin was happy with the outcome. This is because the programme covered the majority of aspects it had asked for. The programme would allow a user to login to the programme, only make a booking if they were above the age of 21, prevent the user from making a double booking and prevent the user from entering an invalid date into the programme and make a complete booking. The programme also had validation when the user would sign up to the programme. He also liked the fact that the user was able to make a print of their final booking information. Godwin also liked the fact that the programme had a professional GUI for the user to navigate through the programme. This would give the customer the professional image the company was after. Godwin was also pleased with the small file size of the programme and the usability of the programme, this would mean the programme could easily be loaded onto any computer and anybody could use it with little training needed.

There were some aspects that the programme did not completely cover, such as the programme not accepting extreme values (test G7) and the fact that the admin is unable to delete a user (test A6),

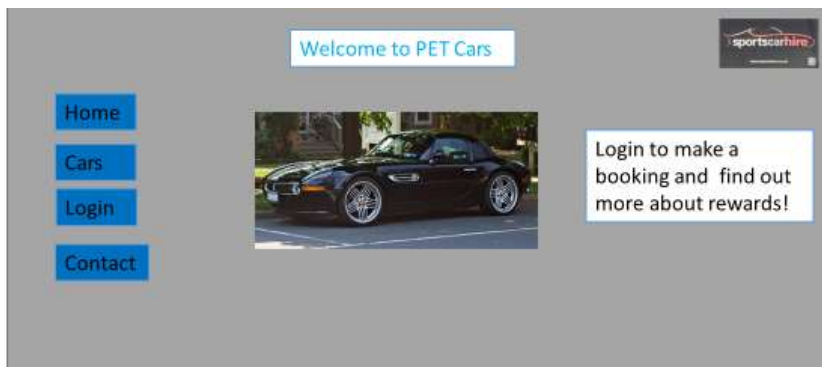
they can update a user. This was something I could continue to develop given more time, however, I was unable to do this for the time being. This did not affect the usability of the programme.

Comparing Forms Design

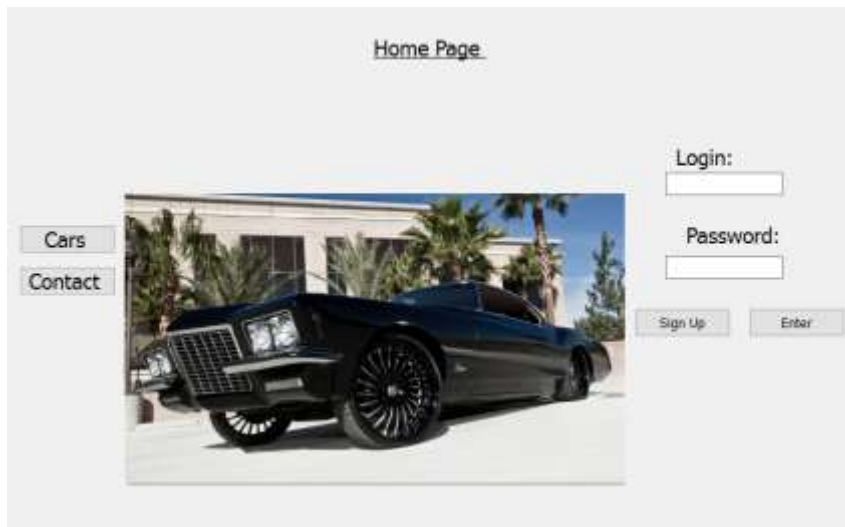
Login:

Initially the idea I had in mind for my forms I would be using were extremely basic.

Login screen:



This was how I initially thought I would design my home/login screen. Buttons would be placed on the side which would lead to new windows when clicked by Bob. A message box on the right-hand side would alert Bob of the rewards he could gain if he made an account with the company. In the top right hand corner, would be the logo/image for the company, and in the middle, an image of a car.



As the programme progressed, so did my forms. This was the next major change from the original design for my home/login screen. At this stage I had decided to use PyQt to create the different forms that Bob would see, and eventually become the GUI for the programme. I had integrated the login capability of the programme into the home page. I had also removed some of the buttons that used to appear on the left side of the window. Furthermore, I changed the picture I would use, to a higher resolution. Instead of grouping the buttons together, I spaced the buttons on either side of the screen so that Bob would easily be able to navigate his way around the window.



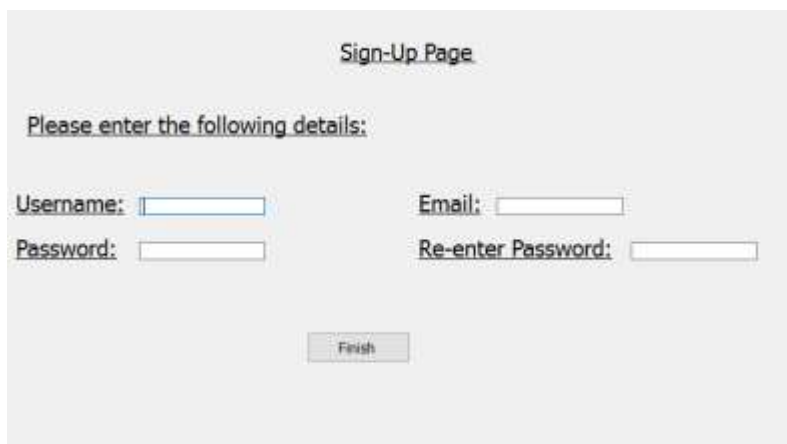
This is now the final design for my home/login screen. I have kept some features from the previous design, such as integrating the login with the home screen. However, I have now grouped the button onto one side of the window. Furthermore, I have used a layering effect where one box appears

more transparent than the background. Furthermore, I have replaced the picture with an aggressive car on the front to catch the customer's attention. The final design for the login screen looks professional and clean. The user can easily navigate around the page as it isn't cluttered.

Sign up:



This is how my sign-up form started out. It consisted of multiple text boxes asking for the user to input the required details. The back ground was plain. Furthermore, text was hard to read within the buttons (black text on dark blue buttons). The buttons were not aligned correctly, nor did the size match up. There was however a logo in the top right of the window, consistent with the logo on the previous window.



My sign-up window moved onto this design. The majority of the data entry boxes had been removed, and the window had been simplified, where the user was only asked to fill in 4 different boxes. Again, there wasn't an image for the background and it looked clinical.

Please enter the following details:

First Name: Last Name:

Surname: Postcode: Password:

Country: Email:

DOB: Enter in the Form: YY/MM/DD

This was the final design for my sign-up window. Compared to the previous design, I returned the other text entries as it was important for the company to collect as much necessary details from the customer as was possible. However, it still had a good format with all boxes aligned neatly, which conveyed the message that Bob was using a professional programme. A background was also placed on the window to give it a nicer feel. Simply 2 push buttons were placed neatly on the window, large enough for Bob to click easily, but not enough to draw unnecessary attention to. Furthermore, a small message was placed on the side of the DOB text box, to alert Bob to the format needed for the DOB to be entered so that calculations could run correctly.

Booking Form:

Make your Booking!

sportscarhire

Car/Brand

Make/Model

Fuel Type

Seats

Deposit

Calendar, select date car wanted

Calendar, select date car return

Initially I thought I would either use radio buttons or drop down lists, alongside the calendars for my booking form. This would prevent the need for having many text entries placed on the window, making the window feel cramped and disorganised. Instead having buttons would mean that Bob could easily navigate the window. The background was plain so that Bob wouldn't become distracted because there was too much happening within the window. Again the logo was in the top right corner.

Car Rental Booking Form

Logout

☐ Driver Over 21

Car:

Make/Model:

Type:

Registration:

Fuel Type:

Condition of Car:

:

Cancel Booking

Select the Time/Date for the car leaving:

00:00:00

Select the Time/Date the car should be returned:

00:00:00

December 2016

Sun	Mon	Tue	Wed	Thu
27	28	29	30	1
4	5	6	7	8
11	12	13	14	15

December 2016

Sun	Mon	Tue	Wed	Thu
27	28	29	30	1
4	5	6	7	8
11	12	13	14	15

Print

Finsh

My booking form then morphed into something very similar to the initial mock-up of the form. There was a mix of text entries, drop down lists, push buttons and calendars. I also added the feature of allowing the customer to print their booking form off. However, the form looked very clinical, there wasn't any colour nor was there a quick simple way to find a car and select a date.

Rental Calculator

Choose your product:

R56

488 GTB

675 LT

C63 AMG

SL65 AMG

Turbo S

Dawn

Golf R

911 R

You pay (£):

£ 0.00

No. Of Days Rented

Cancel

Finish

December 2016

Sun	Mon	Tue	Wed	Thu
13	14	15	16	17
2	3	4	5	6
9	10	11	12	13
16	17	18	19	20
23	24	25	26	27

December 2016

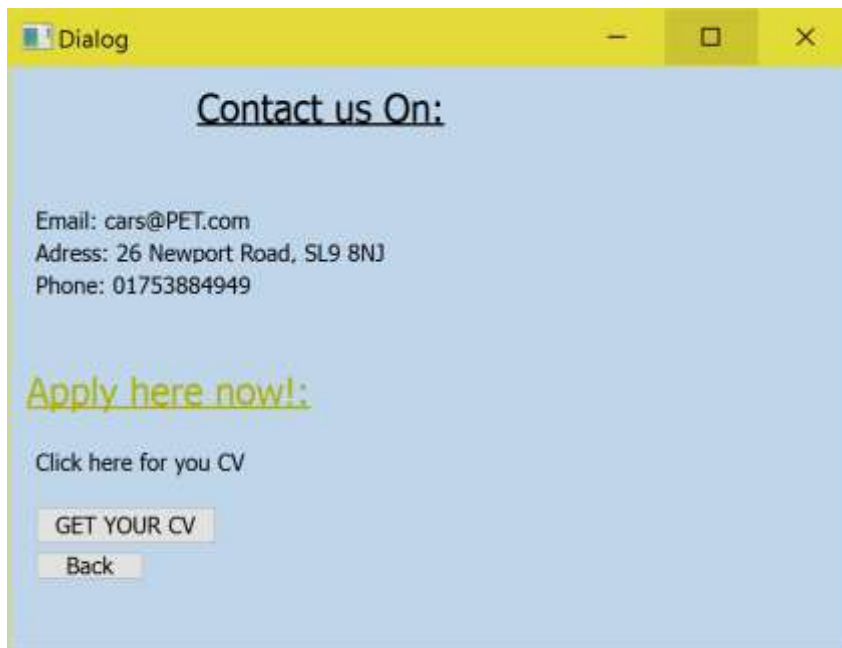
Fri	Sat
31	1
7	8
14	15
21	22
28	29

This is how the booking form looks like now. The first noticeable detail is that there is some colour. I chose a simple blue colour, as I didn't want the window to look over crowded with a picture of a car in the background. Furthermore, it gave the window a more "finished" look. The form has changed a lot from the previous design. I removed all text entries, instead only using a dropdown list where

Bob could select the car her wanted, 1 calendar which would be the date the user entered when they wanted to return the car, and the sue of labels to display the price and the amount fo days Bob would rent the car for. Moreover, I removed the “print” button from the form as I decided to create a receipt of the booking form as it would look more professional. This design makes it a lot easier and faster to make a booking, as there is minimal input from Bob.

A screenshot of a basic contact form window. At the top, there is a 'Contact' button and a small logo for 'sportscentre'. Below these, there are three input fields labeled 'Address', 'Email', and 'Telephone number'. At the bottom left, there is a blue 'Back' button.

This was my initial idea for the contact window. It was very basic, it only consisted of an address, email address and telephone number for the company. There was one button on the window to return to the home page. There wasn't any colour on the window.

A screenshot of an updated contact window. The window has a yellow title bar with the text 'Dialog' and standard window controls. The main area has a light blue background. It features the text 'Contact us On:' followed by contact details: 'Email: cars@PET.com', 'Adress: 26 Newport Road, SL9 8NJ', and 'Phone: 01753884949'. Below this, there is a green link 'Apply here now!:' and the text 'Click here for you CV'. At the bottom, there are two buttons: 'GET YOUR CV' and 'Back'.

This is how the contact window looks now. Again, it has a blue background, one matching that of the booking window which gives the window a more finished, compete look. I included the address, email address and telephone number of the company, like it was displayed on the original form, however I also added another button which Bob could click so he could create a CV so he could apply for a job at the company. This was because the company was fast expanding and it would be easier for someone looking for a job to visit the programme and get a CV easily.

Given more time, I would go back and make each form implement the use of radio buttons and drop down lists better. This would also make it easier for the programme to be ported to a mobile device at a later stage, as the user would find it easier to use the buttons and lists instead of entering text into the boxes. I would also make the windows large sizes. This is so that the information displayed on the windows do not look squashed in some circumstances. Furthermore, I would replace some images I used for the back grounds with images of cars that the company offers instead. I would also add a new window with customer feedback and reviews, for any new potential customers to see.

Test Plan:

This is the test plan which had been previously created. The objective of this plan is to go through each test number and check whether the programme is able to do this objective.

In the evidence box, there will be a letter followed by a number. This letter refers the to the table which the evidence for the test is done in. these tables are at the top of the evaluation document. The number next to it refers to the test number that was allocated to it.

Login:

Test Number	Test Action	When will this be Tested	Evidence	Customers Results
1	Start the programme	Throughout development	L1	Programme starts when executed
2	Click each button on the login screen and see if the different windows open	End of Development	L2-L4 and C1	Each button open to a new window when clicked
3	Click the enter button without entering data into the programme	End of Development	L8	Programme does not allow the user to login
4	Click the enter button when only the username box has been filled out correctly	End of Development	L9	Programme does not allow the user to login
6	Click the enter button when only the password box has been filled out	End of Development	L10	Programme does not allow the user to login

	with incorrect details			
8	Click the enter button when both the username and password box have been filled out with correct details	Throughout development	L4-L5	Programme allows the user to login
9	Click the enter button when both the username and password box have been filled out with incorrect details	End of Development	L9	Programme does not allow the user to login
10	Click the enter button multiple times quickly to see if the programme crashes	End of Development	L11	Programme does not crash; a message box appears each time
11	Click the sign up button	Throughout development	L7	The sign-up window opens
12	Click any message boxes that appear when I try and login to the programme	Throughout development	L6&8,9,10	Each message box, when clicked, disappears
13	Enter extreme details into the username and password box	End of Development	L12	The programme does not prevent the data from being entered, however it does limit the amount of characters that can be entered to 10. The programme does not continue with the login.

Test Number	Test Action	When will this be Tested	Evidence	Customers Results
1	Enter the admin username and password into the programme	Throughout development	A1&A2	The programme allows the admin to login and presents them with a different window
2	Enter only the admin username into the text entry	End of Development	A9	The programme doesn't allow the admin to login
3	Enter incorrect admin details into the username and password text entry	End of Development	A3	The programme does not allow the admin to sign in
4	Click "OK" on any message boxes that appear	Throughout development	A2&A3	The message boxes disappear when the message boxes are clicked
5	Click "database" button on the admin window When the window opens scroll through the records within the table	Throughout development	A4	The new window opens when the button is clicked
6	Try to change data within the table	Throughout development	A4	The data changes when the user clicks a cell and edits the contents
7	Enter extreme details into the cells	End of Development	A10	The programme allows extreme data to be entered, this should not happen

8	Enter null data into the cells	End of Development	A11	The programme allows null values to be entered into the cell, this should not happen
9	Press the update button	Throughout development	A4&10,11	The update buttons changes the contents of the cell to the new contents
10	Check to see whether the changes have been made in the correct table in the database	Throughout development	A5	The changes have been made in the database
11	Click a cell and check to see whether the programme displays which cell a user is currently choosing to edit	End of Development	A4	The programme displays the contents of the cell and tells the user which cell is being edited
12	Click multiple cells quickly	End of Development	A4	The programme does not crash
13	Click the "back" button on the table view window	Throughout development	A4	The button returns the user to the previous window
14	Hover over the chart to see if each "slice" enlarges so it can be viewed easier	End of Development	A8	The "slice" enlarges
15	Click the "back" button on the graphs window	Throughout development	A8	The button returns the user to the previous window
16	Click the "back" button quickly on the graphs window	End of Development	A8	The programme

				does not crash
17	Click the “back” button on the admin window	Throughout development	A2	The button returns the user to the home page

Sign Up:

Test Number	Test Action	When will this be Tested	Evidence	Customers Results
1	Click the sign up button	Throughout development	L2	The window opens
2	See if the “sign up” window opens correctly	Throughout development	S1	Everything is displayed as it should be
3	Enter data into the text entries	Throughout development	S2	The text entries allow data to be entered
4	Press the “sign up” button	Throughout development	S3	The sign up button works when clicked (carry’s out function)
5	Check to see whether the data has been saved to the correct table in the database Check to see whether the auto increment function is working properly	End of Development	S5	The data from the text entries is saved to the database
6	Do not enter details into the text entries and click the “sign up” button	End of Development	S12	The programme prevents the null values to be saved to the database
7	Check to see what data has entered the database	End of Development	S4-S8	The information entered is saved to the database in

				the correct order
9	Enter extreme details into the text entries	End of Development	S13	Partial success, the programme allows the data to be entered into the programme, however the data isn't saved to the database
10	Click the "back" button on the "sign up" window	Throughout development	S14	The button returns the user to the previous window

Cars:

Test Number	Test Action	When will this be Tested	Evidence	Customers Results
1	Click the "cars" button to check whether the window opens	End of Development	L13	The cars window opens when the button is selected
2	Check to see whether all the cars that the programme offers are on display	End of Development	L13	All the cars the company offers are on display there
3	Check to see if the "back" button works	End of Development	L14	The user is returned to the home window when the back button is pressed

Contact us:

Test Number	Test Action	When will this be Tested	Evidence	Customers Results
1	Click the "contact us" button	End of Development	G1	The window opens
2	Check whether the relevant details are being displayed on the contact window	End of Development	G1	All details are displayed on the window correctly
3	Click the "cv" button	End of Development	G2	The button opens a new window when clicked
4	Click the "back" button	End of Development	G3	The back button returns the user to the previous window when clicked

Curriculum Vitae:

Test Number	Test Action	When will this be Tested	Evidence	Customers Results
1	Click the "cv" button	Throughout development	G2	The button opens a new window when clicked
2	Click the "home" button	Throughout development	G3	The buttons return the user to the home window when clicked
3	Enter text into the text entries	Throughout development	G5	The text entries allow the user to enter data
4	Enter extreme data into the text entries	End of Development	G7	The text entries accept extreme data. I did not think I needed to add validation to the text

				entries, as it was not an integral part of the programme
5	Press the “show me cv” button	Throughout development	G6	The buttons display exactly what the user had entered from the previous window

Printing CV:

Test Number	Test Action	When will this be Tested	Evidence	Customers Results
1	Check to see whether the correct details are being displayed on the window	End of Development	G6	The correct details were displayed
2	Check to see if the formatting of the data on the window is correct	Throughout development	G6	The formatting is correct
3	Press the “edit” cv button	Throughout development	G8	The buttons return the user to the pervious window where they can make their changes
4	Press the “print” button	Throughout development	G9	The print button opens a new window when clicked
5	Check to see whether the printing window opens	Throughout development	G9&10	The window opens
6	Check to see whether I can click a printer I want to send the document to	End of Development	G10	I can click the printers available to me

8	Check to see whether I can change the preferences	End of Development	G11	I can change the printing preferences
9	Check to see whether I can find more printers	End of Development	G10	I can find more printers when the button is selected
10	Click the "cancel" button	Throughout development	G12	The printing action is cancelled when the button is pressed
12	Check whether a preview is created of the document	End of Development	G13	This failed to create print preview. However, I create a preview of what the user is about to print, so I don't think a print preview was needed again

Rental Calculation:

Test Number	Test Action	When will this be Tested	Evidence	Customers Results
1	Check to see if the rental calculator window opens when the user enters the correct login details	End of Development	L5	The window opens when the correct details are entered
2	Check to see if the rental calculator window opens when incorrect details are entered	End of Development	L6	The window does not open

3	Check to see if the drop down list displays each car	End of Development	B4	The drop down list displays all the cars
4	Check to see if I can select each car from the drop-down list	End of Development	B4	Each car can be selected from the drop-down list
5	Check to see whether the calendar works when clicked	Throughout development	B3	The calendar does change date when clicked
6	Check to see whether I can click a date before today's date	End of Development	B19	This can be done but a warning message appears
7	Select a car from the list and click a date from the calendar	Throughout development	B8	The calculation is carried out
8	Click finish	Throughout development	B8	The booking is saved
9	Check to see whether the correct price is displayed	End of Development	B12	The calculation is done correctly
10	Select a car from the drop down list and select a date before today's date	End of Development	B19	This can be done however a message box appears when this is done
11	Click the "cancel" button	Throughout development	B20	A warning box appears when a user try is to cancel a booking
12	Click the "finish" button to see whether the booking window will open	Throughout development	B17	The new window does open
13	Click the "finish" button and check	Throughout development	B17	A message box appears

	whether a message box appears			alerting the user of their booking
14	Click the “finish” button to check to see whether a user can make a booking if they are above the age of 21	End of Development	B7	This can be done
15	Check to see if a message box appears when the user makes a booking but they are below the age of 21	End of Development	B7	This is done
16	Check to see if a message box opens when the user makes a booking and they are above the age of 21	End of Development	B7	This is done
17	Check to see if the information entered by the user is saved to the database	End of Development	B8	The information is saved to the database
18	Check to see whether information from another table in the database, has been saved to the bookings table in the database	End of Development	B8	This is also saved to the database, information from the cars table is now also saved in the booking table
19	Check to see whether data has been saved to the database if the user is underage and can't make a booking	End of Development	B7&8	The information is not stored
20	Check to see if the user can make a double booking	End of Development	B5	A message box appears telling the user they cannot make

				a booking since the car is in use
--	--	--	--	-----------------------------------

Booking:

Test Number	Test Action	When will this be Tested	Evidence	Customers Results
1	Check to see if the data made from the booking is the same as the data shown on the final form	End of Development	B17	The new window opens
2	Check to see if the "print" button	Throughout development	B18	The printing window opens
3	Check to see if the "home" button works	Throughout development	B18	The user is returned to the home window when the button is pressed

Forgot password:

Test Number	Test Action	When will this be Tested	Evidence	Customers Results
1	Check to see if the new window opens when the "forgot password" button is clicked	Throughout development	C2	The window opens when the forgot password button is clicked
2	Check to see if the "cancel" button works on the new window	Throughout development	C11	The user is returned to the previous window when the cancel button is clicked
3	Check to see if text can be entered the "enter email box"	Throughout development	C3	Text can be entered into the text box
4	Enter extreme data in	End of Development	C12	The text entry allowed extreme data

				to be entered. I did not think of validating this entry because without a valid email address, the email wouldn't be sent anyway.
5	Enter an invalid email	End of Development	C13	The user can enter an invalid email address. I did not add validation because the email would not be sent anyway
6	Click the "next" button	Throughout development	C8	When pressed, a new window opens
7	Check and see if the new window opens	Throughout development	C8	Window opens
8	Enter a valid email in	Throughout development	C3	The email is sent to this address
9	Click the next button	Throughout development	C3	The function is carried out
10	Check and see if the email was received	End of Development	C4	The email is received
11	Check to see whether the code appears random	End of Development	C5	The email contains a code that appears to be random, with a mixture of letters and numbers
12	Check to see if the code has letters and numbers in it	End of Development	C5	The code contains uppercase letters and numbers

13	Check to see whether the email sent has a subject and sender	End of Development	C5	The email has a subject
14	Check to see whether the message sent in the email has a professional looking format	End of Development	C5	The email does look professional
15	Resend an email to a valid email	End of Development	C6	The email is resent
16	Check to see whether the code received is different from last time	End of Development	C7	The code is different from the previous emails that have been sent
17	Copy the code into the programme	Throughout development	C9	The code can be copied into the programme
18	Click the "next" button	Throughout development	C13	The button opens a new window when clicked
19	Check to see if a new window opens	Throughout development	C14	The window opens
20	Enter data into the "new password" box	Throughout development	C14	The user is able to enter a new password
21	Check and see if the new password has been updated into the database	End of Development	C14	The new password has been updated in the database

Failed Tests

The test that had failed were due to not enough validation or the fact that the customer found it hard to use the programme.

For example, after testing the customer said that they would like to see more textboxes appearing when they had carried out an action or an alert of some kind, maybe a text box changing colour. This

is so that they knew the action they wanted to do had been completed. At the moment, the programme may not always show a text box, so they were unsure if the action was completed.

Furthermore, Godwin asked for the whole programme to have validation across it. I however, opted to not put validation in some case (C11-13) because it would not have an effect on the system. I only put validation in for processes that would directly affect the programme, for example the calculations (B19&15-17). Given more time I would go through and put validation in at every point in the programme.

Customer testing Programme

Now that the programme had been completed, I asked Bert to test the programme, this would mean he would need to create an account, login to the programme, create a booking and use the different buttons on each GUI to navigate through the programme. He would then need to give me feedback on the different aspects of the programme.

- Initially, Bert saw the home/login screen of the programme. He was impressed with the how aggressive the login screen looked. The dark green colours made the GUI look sharp, whilst the image of the car gave Bert a good understanding of what the company was offering. Furthermore, he said that the grouping of the text boxes and buttons meant that the interface looked clean.
- Bert then created an account. After he had filled out the required fields, and clicked the sign-up button, he said that the text input boxes needed to be larger, but he thought that the buttons had been positioned in a good place. He liked the fact that the sign-up window had a background picture of the car. He also said that the text displayed on the window needed to be larger as at some points he found it difficult to read. Furthermore, he said that the title of the window was obscure due to some letters interfering with the background (white text with white background). Bert also said that he would prefer the text box to change colour if the data entered was incorrect. This is so that it is easier for Bert to find which text entry has the incorrect information.
- Bert logged into the programme. he liked that a message box appeared when he entered the correct username and password. He also liked that each window would “close” when a new window is opened. He found the rental form easy to use. This is because the form didn’t require any characters to be inputted by Bert, instead he could use the drop down list and the calendar that has been provided. He liked the fact that the calendar was large and the drop down list was large. He also liked the fact that the background was a simplistic blue colour as it didn’t distract Bert from the other details that were on display on the window.
- Bert then moved to the receipt window. He liked the fact that the text was large. The white text meant that the background didn’t defer from the text being displayed. Moreover, the fact that the window didn’t have a lot of information, it made the receipt window easy to read.
- Bert then viewed the different cars on offer by the company. He liked that the programme displayed pictures of the cars on offer, this is because not everybody is “car savvy”, so they may not know which car they are about to hire. However, he said that the formatting of each picture on display could improve. He also said the car should be hyperlinked, so when a car is selected, a new window opens displaying the car selected on the rental form.
- Bert then viewed the contact window and the CV forms. He was pleased with the ease of navigation through the different windows by using the push buttons provided.

Maintenance and Development

- As PET continues to grow, there will be more staff and more customers who will be joining the company. All of this personal data entered by both staff and customer will have to be stored into PET records and kept secure.
Eventually this could present a problem as there is only so much data that can be stored with the physical space available (servers take up a lot of room). A way in which the company could overcome this problem is to implement a “cloud Back-up” system within the programme. This is so that PET always has a log of which staff they hire, and which customers join.
Moreover, saving to a cloud back up means that PET can free up space on their servers for new customers/staff, or any other important information they may need to store. An alternative way in which PET could overcome this problem would be to store the data in an off-site server farm. This way the data would be very secure and less vulnerable to being hacked, unlike the cloud storage.
- I would also implement a “car buying” option, where the customer is able to purchase the car they have rented, or a car of their choice, directly from the company. This would mean that the company would need to have access to more than one car each car they have in stock. Alternatively, the company could continually change the cars they have on offer, however, returning customers may not like this. This option would mean that the programme would have to allow the customer to search for a specific car, or a car that is similar to their criteria of their choice, dependent on their search criteria they have entered.
- The programme could also start making use of basic AI. For example, the programme could display Customers Results of cars on offer by the company for a returning customer, dependent on their previous searches. This means that the programme would reduce the time taken for the customer to make a booking, so the customer would be pleased, and it would reduce the load on the company server which would make room for more traffic instead.
- Furthermore, the programme could also make use of API's. For example, many customers may already have accounts made with services such as (Google, Facebook, Youtube) and they may want to use those accounts to sign into the programme, instead of having to make another account for a separate service.

Big O time complexity of the Programme:

- The time complexity for my programme would be $O(n)$. This is a linear time complexity. This is because, as the data sets increases (more customer and cars) then the time taken for searching these different criteria to be carried out is directly proportional to the amount of data there is.
- However, if we were to change the algorithm, the programme uses to search for items within the database, we could potentially have a $O(\log(n))$ time complexity. This is because instead of the algorithm searching for a specific value within the database (assuming the company has grown considerably and there are more customers and data), the algorithm can use a binary search that would half the data set each time, which decreases the size of the data set

Maintenance Issues (Portability of Programme)

- This programme was created for a display of resolution (1920x1080p). This means that if the programme is run on a monitor of different resolution, the programme will not be displayed correctly and Bob will find it extremely difficult to read the text on the screen or even use the buttons and text entries properly.
- A similar issue would occur if the programme were to be used on a mobile device. The information on the programme would not be displayed accurately, nor would the programme be able to interface with SQL or PyQt properly. The programme would also need to make use of radio buttons and drop down lists more, instead of text entries, as they are easier to use on mobile devices.

Given more time I would add more validation to the programme. I would also like to add more features alerting users of the actions they were carrying out, whether that be a text entry changing colour when an action is executed, or more text boxes appearing. This could not be achieved due to time constraints on the project.

For the partially unmet criteria I would go back through the project and add more features that the customer might like to use, for example the addition of being able to purchase a car after the customer had rented one.