ClickCar Project
Project Report

One Term Individual Project
CM2303
Sam Douthwaite-Robinson
C1127442

Project Supervisor: David W Walker
Project Moderator: Alun D Preece

Abstract

The aim of this project was to create an system which would improve transport. The main platform for the project would be online in order to make the system social. Based on the merits of a social network the system aims to link drivers and potential passengers together through a search engine which links queries to lifts available on a database.

The system works by allowing people to register as a passenger, free of charge, and allow them to join as a driver if they have a car and the relevant documents such as a driving license, insurance, MOT, and tax. Once a user has become a "driver" they can create journeys, which they wish to carry out. To avoid abuse of the system all passengers are required to review each journey they have been on. In turn this filters trustworthy drivers from the not so trustworthy drivers.

Each user has their own profile which holds the basis of a social networking platform such as information about each user, a profile photo, messaging system and a user rating system. The higher a rating, the more trustworthy a user becomes.

This project has been based on similar services which already exist. While one particular service dominates the scares market I found a gap in the market when it came to simplicity & reliability.

While I didn't have enough time to include all the features that one would have desired I did concentrate on the key areas that would make my project differ from services which currently exist.

Table of Contents

1.0 - Introduction

1.1 - Aims

The aim of the system was to create a service which would run as self sufficiently as possible. It was to connect potential passengers with drivers that wouldn't mind sharing their car seats for a contribution towards the travel costs.

There would be two major areas in this project. The first would be the basic registration, or passenger registration which would take some basic details, email address, preferences and some information about regular journeys a passenger makes in order to improve the quality of the service.

The second would be aimed towards drivers. Information here would include the name and model of a car, car registration, as well as if the car was petrol or diesel in order to give accurate calculations when working out the cost of any given journey distance.

The system would also need a method of interaction between various users and so a messaging system was needed.

Security & privacy is vital in such a project and so no sensitive information is to be shared between users. Privacy details would need to be made clear to the user.

1.2 – Target Audience

I expect my target audience to mainly be young people. For security reasons there is an age limit of 18 years old, which is verified when a user registered by providing their date of birth. Students fall into this category, as they tend to need cheaper travel that is convenient for them. Although it is thought that the system will attract mostly young people the system could attract a wider audience with a higher level of security.

1.3 - Scope

Before I started researching my project I already knew that I would be using a mixture of html, PHP, JavaScript and MySQL. Having already done some similar projects based on these languages in the past I thought it would be a wiser option to use technologies which I was already comfortable in.

I knew the project would be online based and so I decided to place it on my server. This provided a constant backup online as well as copies on GitHub & my own hard drive.

I did have to research into some factors such as policies on hosting such a service & the development of a small mobile app.

1.4 – Approach

Having known that I was going to produce a web based service I decided that the waterfall model would be most appropriate. I needed solid deadlines for each stage so that I could allow myself enough time for the final parts of the programming stage for things like debugging, & fixing tedious errors.

I spent the first two weeks researching some of my major concerns such as an option of handling the billing between passengers & drivers. I also wanted to know more about the possibilities of tracking a users position, as one of my ideas was to track the journeys live progress. Because of my limited time scale I had to rationally think of the major areas I wanted to concentrate on. I wanted a system which allows users to share journeys to potential passengers. However I also wanted strong security features around the idea.

I concentrated on the requirements & design until I felt ready to make some progress. Starting out with the users registration I ran through the system bit by bit as though I was using it myself bringing up any concerns along the way such as the data I was going to collect & how I was going to use it to create a useful system. By running through the system step by step I was able to add in features as I went along making adjustments to future parts. This did cause me some back tracking when I realized I was missing features which needed to be added in at an earlier stage.

An example of this is when I decided I would use post code matching for when a user searches for a journey. Original it was based on place names but I found those details weren't accurate enough for me to provide places close to where the user wanted to travel from or to, & so I needed to add that feature in.

By planning weekly milestones, as well as daily tasks I was able to stay on top of the majority of the work. I was able to asses if a particular feature was necessary to demonstrate what I'm trying to do. What I did find is that I have created a

system which allows me to program ideas which I didn't have time for now, in the future such as text messaging verification.

2.0 - Background

2.1 – Issues

2.1.1 - Safety

As already mentioned there are already systems out there which offer lift sharing services. Although there is proof that the system works there are problems that arise such a driver not picking up a passenger when they said they would. While this isn't the responsibility of the service I feel that more could be done to stop people trolling the service. An example would be a ban from becoming a driver if the user receives a certain amount of poor reviews.

2.1.1 - Payment

Many of the current systems do not deal with payment methods. Originally I wanted to handle payments via PayPal & ClickCar (*my developed system*), however after researching some of the laws this was not feasible, as my service would come under a similar business as a taxi or bus service. Because I want to keep any running costs down, this wasn't the type of approach I wanted. I still felt it would be achievable to handle some form of the payment calculation & have methods in place, which send receipts ones journeys were complete.

2.2 – Planning

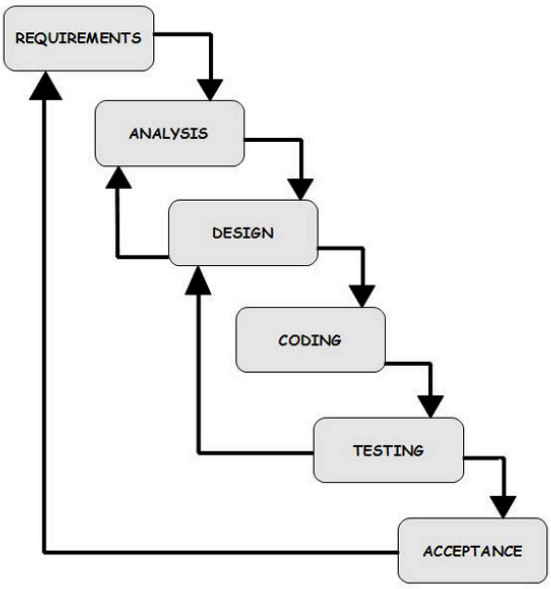
Some other issues I had was finding a database containing a cars make and model. This was needed in order to keep data consistent when users uploaded their car details. I found this was a struggle as most companies that offered this data charged. Luckily I found a GitHub repo, which had been generated by a very generous person [1]. While this database does not update itself automatically it does allow me to update it quite easily as it fits into my MySQL database structure. I had a similar problem when trying to find a database of post code regions for every town in the UK [2]. These databases contain thousands of rows which wouldn't be possible to create manually.

The Waterfall Model

The waterfall model is a widely used technique for developing software. It allows me to work through step by step while allowing for fallbacks and time to develop on each section. It also allows for time to evaluate each section as I work through the project.
The only downfall of using the waterfall model is that it doesn't allow much time for major changes in the system after the design stage. Due to my short development period it would be best to stick my original designs rather than re-design half the system after weeks of work.

One other downside is that the waterfall model cannot always turn designs into successful products [3] and so it's important for me to design within my means

As I have used the waterfall model in the past & it has been successfully tried & tested by many I feel it will be the most useful method of developing my project.



A references to this illustration of the waterfall model can be found at point [4] in the Bibliography.

3.0 – Research

3.1 – Locations & Directions

When thinking about how I would allow drivers to plan their routes I knew that I had to use a developed system. The most obvious choice for me was Google Maps with navigation. This API was not only free but also came with plenty of documentation. [5] By registering and using your API key in your code you are allowed to access the service 2500 times which fits my needs fine.

Another problem faced was trying to keep place names that users typed in consistent. When developing my system I realized that it was very case sensitive and so I needed a more intelligent system in place.

Auto correct allows search queries to be consistent and minimize a poor experience. Once again an adequate system was needed and Google Map's API came in handy again. As my system uses HTML & JavaScript, Google's system fitted in nicely.

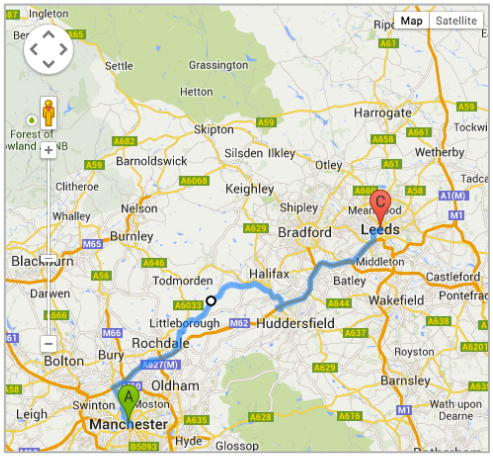
The below code demonstrates how my system uses user input in order to locate the best known driving route. This code was taken from examples on the Google Maps developer's site, & adapted to fit my needs.

```
var request = {
      origin:start,
      destination:end,
      waypoints:[{location: dest1}],
      travelMode: google.maps.TravelMode.DRIVING,
      unitSystem: google.maps.UnitSystem.IMPERIAL
```

The *dragability* features to change a route in Google Maps worked well in order for a user to work out their exact route.

It was possible to pull the information generated by this map in order to work out distance from the start point and the destination, including any particular routes the user may have chosen to take.

Useful information includes data like the distance of the route which allowed me to calculate the overall cost of the journey based on what type of fuel the car uses.



An example of the Google Maps Directions feature.

3.2 – Calculating Journey Prices

In order to generate the overall estimate cost of each journey I needed to know what factors make up a journey. I developed an algorithm which takes into account the price of fuel, (petrol or diesel), the average mpg, and the distance. By using these it was possible to generate an estimate for each journey. This price could then be divided by the number of seats the driver is willing to give up.

On current services like *BlaBlaCar* I noticed that the pricing for each passenger was quite low. Having worked out that this cost was based on the maximum number of passengers the driver is willing to have I decided that it would only be fair but to offer two prices. The first price, and most clear price would be the cheapest possible price based on the number of seats that the driver is willing to give away. The second price would be based on only two people travelling. One passenger & one driver. This is a fairer pricing system as each lift is only likely to get one passenger. One could only imagine that a driver would not want to be making many pickups prior to driving his/her actual journey.

3.3 – GeoLocation

In order to improve safety & trip assurance it would be desirable to have a system in place which tracks the drivers journey in real time. I found that this would be the only way of assuring that journeys are completed. The system could also be used in order to provide a more accurate & detailed user rating allowing potential passengers to make the correct choices.

Because I want it to work as a website & a mobile app I feel I have researched into *geo-locating* using android, & which method is better to use.

This area will focus on tracking mobile phone devices. GPS is a good solution in doing so however it is well known for draining a phones battery. I feel that this would discourage users as I hope to make it mandatory for drivers to have the mobile app in order to track their journey progress.

GPS

GPS or Global Positioning System uses a system of satellite systems to locate a device on the map. Mobile phone devices can also use a cellular network to determine the location of a device by using the nearest transmitters.
If the devise can calculate signals from 3 or more antennas then the signal can be calculated accurately [1].

Although accurate it does consume battery power quickly and is only available outdoors. This should work fine in a vehicle due to the large amount of windows and the fact that satellite navigation systems work in the same manner.

Android Network Location Provider

This system uses Wi-Fi signals and the cell tower. It also consumes less battery & responds faster than GPS according to Androids developer site. On an android system you are able to use GPS or the Network location provider [2].

Either system can contain flaws & can be inaccurate at times. When contemplating my system in regards to this the system doesn't need to be over accurate but should have a rough location reading within 500m diameters. This should be sufficient enough to determine if a journey has been completed or not.

Android recommends renewing the user location as users moves every so often. In my case I think it will be relevant to perform location update every 5 minutes or so. Performing an update every few seconds would not be worth the hassle as I expect typical journeys to be long distance and so they may last around 30 minutes to 3 hours, (give or take, depending on the journey).

Android location detector.

```
String locationProvider = LocationManager.NETWORK_PROVIDER;
// Or, use GPS location data:
// String locationProvider = LocationManager.GPS_PROVIDER;

locationManager.requestLocationUpdates(locationProvider, 0, 0,
locationListener);
```

The above code has been given by the Android developer pages [2] & can be used to get the location of a driver, or indeed a passenger.

Passenger & The Mobile App

The system may be more accurate if the passengers too have a log of their location. This could provide added security when it comes to the driver and passenger travelling together.

To encourage the use of this feature a passenger could gain a higher score rating by using the app.

The benefits of this are that you can tell if a passenger has taken the journey they have paid for. I intend for payments to be made through the site and so this could prove to be helpful and make sure that both passengers and drivers are treated fairly.

App Development

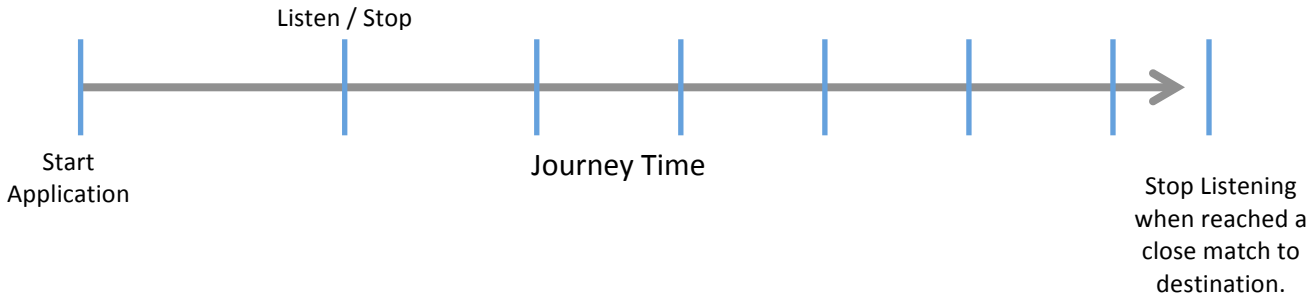
Due to the large amounts of technical support offered by Android I have decided to concentrate on developing the App through Android. During development if there is enough time I may try and develop an iOS app which will work with the website.

It is possible to stop "listening" for a location with the following line of code.

```
// Remove the listener you previously added
locationManager.removeUpdates(locationListener);
```

The shorter the "listening" window the less the phone uses the network location service. Obviously I want to preserve as much battery life as possible and so there won't be much need to keep using the facility or listening for a long period. One reading every 10 minutes could be sufficient.

Below is a diagram, which will illustrate the mobile application listening for the users location in the background.



Application In Action

When the driver starts the journey on his smartphone the app will detect the users location. Once it has a location it will stop listening. It could then listening for a location again after 30 minutes & then update the location on the server.

The final goal will be for the journey to end when either the driver clicks "End Journey" or the phone has detected a geographic match to the destination.

Mobile Website

Another possible method would be to develop a mobile website based on a smartphone application. This could be a relevant solution to my short time constraint. It would be possible to base a system on the main web service. The smaller website would work in a similar manner to the system describe previously.

3.4 – Car Data

One of the main early issues was having a database of car make and models. Most companies charged for such databases, finding a free source wasn't easy.

There were multiple options. I could either use JSON to query a website with a registration plate number which could return details about a particular vehicle. This is a gray area as again most providers wanted to charge. There was the option of finding a "hack" by using page element names to send a post request to these sites each time and simply scrape the data. Many services do not allow this, & either forbid it in their terms of use, or have anti-robot *captcha* systems, such as the DVLA and other third party sites.

Although not as accurate as a live data warehouse, I decided to go in search of a database. Again there aren't many free services out there. Google returned a few sites which all charge, the only free data was what people had shared on *GitHub*. These consisted of up to date data about cars, however once copied the data would remain static & records need to be manually updated.

Because there are thousands of different car make & models I needed to find a database which a reasonable amount of data. After a few attempts it was an SQL dataset was found. It was fairly simple to query these tables as they had being laid out with the type of system I had in mind. A user is able to select a make, a second query would then filter the models down depending on the vehicle make.

Credit is given to this GitHub repo for their free access to a large dataset of car makes and models

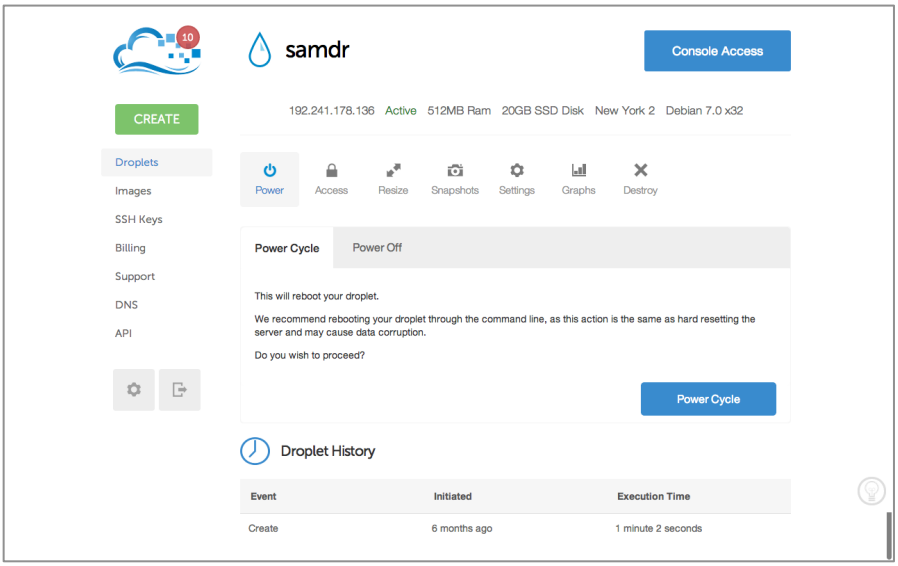
```
https://github.com/VinceG/Auto-Cars-Makes-And-Models/blob/master/dump.sql
```

3.5 – Data Protection Act

The Data Protection Act is something that is taken very seriously in the UK, unlike countries such as America. My servers are based in New York, United States & so as far as I am aware I do not need to worry too much about this. Neither am I currently releasing this to the open public. It is on my development server and isn't enabled for Google's crawler bots to index it.

I will not be releasing the app onto Google's Play store, or Apple's App Store, & so I won't have to worry about declaring what information I have used.

I have included a feature which would be useful if the website was to go live. This is a page that shows all the details stored on a particular user. It shows what details are shown publicly & what details are only down to individual users. An example would be only showing car data to passenger that gave been accepted onto a journey.



A screenshot of my hosting account & the server location. (NY2).

*My cloud server is provided by Digital Ocean.
http://www.digitalocean.com

3.6 – Law

When looking at other services like *blablaCar*, they clearly state in their terms of use that drivers cannot carry more than 7 passengers. After this they are required to have special insurance.[11] For the sake of assuming everyone offering a lift will be using a car, & not a minibus I will be limiting the number of seats to 4. Ideally most journeys will only be offering one person a seat. I feel that this will be better as the last thing a driver would want to do is spend the first part of their journey driving around picking people up. This will not only cost more in fuel for the driver, but it will also be inconvenient and go against one of my desires for the service by making journey sharing easy, & hassle free.

Background Checks

I would also like to state in the websites terms of service that the website does not do background checks on individual people. I have researched into background checks and found that most services charged, and limited the number of requests. It would also be very time consuming making sure the system would deny access to people with criminal records, etc. It could be a case of denying the wrong people access, which in turn would again limit the use for the system. This could be something that could be worked on in the future.

4.0 – Specification

4.1 – Desired Features

Having researched into various other services that currently exist I have a short lifted desired features. They have been grouped into features, which are mandatory for my system to work.

Mandatory features will be required for my system to work to a basic standard, and complete the solution to the problem.

Additional features will be features that will enhance the basic system & create a better end user experience. These features are the "*if we have time*" features.

Future features are features which will be optional *add-ons*. They are not required to be completed by the end of the project, however they could expand the system in the future.

4.1.1 – Mandatory Features

Member system:

- Basic Registration process;
  - Username, password, email, first name, surname, address, telephone, gender, age, bio, profile photo.

- Registration for drivers;
  - Car make & model, tax, mot, insurance, driving license, colour of car, car comfort.

    This registration will happen after the basic registration process & if the user wishes to carry out shared journeys.

- Personal preferences about members;
  - Hobbies, interest.
  - Regular journeys made, date & time
  - Profiles

Search for available journeys;

- Search to & from on a particular date, near a convenient time.
- Search for return journeys.
- Reserve a seat for the selected journey.
- Select number of seats required.

Message drivers

- Passengers are able to send messages to drivers.
- Drivers are able to reply to these messages.

Configure journeys;

- Driving to & from.
- Date & time of journey.
- Maximum detour in miles.
- Number of spaces in car.

Mobile Application

- This must be downloaded by drivers to track a journey.
- Prompt driver when journey is about to start.
- Deliver receipt to passenger when journey is complete.

Verification;

- Verify email address.
- Verify when a journey has begun and ended, simple press of a button on a mobile device.

Integration with Maps;

- Preferably Google Maps as it's free to use.
- Similar services have also used Google Maps.
- Accurate plotting of location on maps.

Rate Journeys;

- Passenger can rate journey & drivers.
  - Rate comfort, punctuality, driving skills, car, friendliness, comment, "Would share a car with them again".
- Drivers can rate passengers.
  - Rate friendliness, likeability, punctuality, luggage space, comment, "Would share a car with them again".

4.1.2 – Additional Features

Vehicle system:

- Car photo, Car Registration.
- Enter car registration to find out vehicle details automatically.

Search for available journeys;

- Place request for journey & match new journeys.
  - Alert user about new journeys

Friend System;

- Add & remove friends.
- Edit Friends list.
- Find friends based on Facebook (*or any other social media*) friends.

Configure journeys;

Suggest price based on miles, ability to edit/ tweak price.
Number of spaces in car.
Suggested price per person based on number of seats & number of confirmed passengers.
Maximum luggage space – Rough description & quantity, e.g. rucksack, hold-all bag, box.

Mobile Application

Track passenger journey progress alongside driver journey progress.

Create journeys via mobile App

Perform features which will only be originally available on the desktop site.
Find journeys based on current GPS location.

Verification;

Verify phone number.

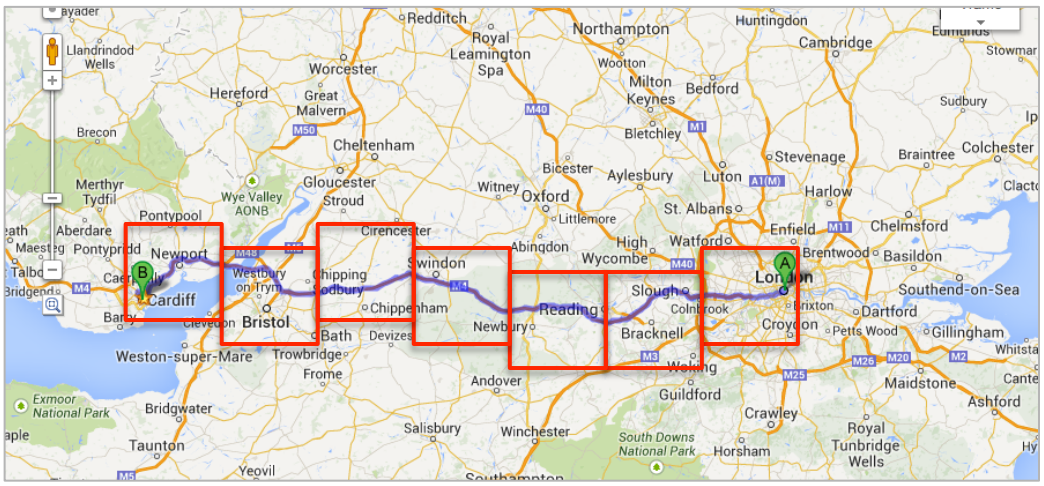
Help & Support

Provide a contact email address for help & support.
Online support pages.

4.1.3 – Future Features

Future features are features I would like to include in the future, or perhaps if I am ahead of schedule. Some useful features could be a more sophisticated search engine automatically picking up towns *en-route* of each journey.

Google Maps allows you to draw polygons on top of the map. Using these it would be possible to draw a radius for each part of the route on the map. Using the coordinates it could pick all the towns within the range.



The diagram above is a screenshot taken form Google Maps with polygons around a route form London to Cardiff marked on there. All the towns within these squares could be used to perform a loop through a list of towns matching journeys. Therefore if a user wished to be dropped off half way through the journey it could calculate the if your desired pickup or drop-off point is on the route.

The search algorithm would need to perform far more searches in the database that could put a strain on my basic MySQL database. A more robust database would be needed, & preferably something that could be scaled well.

Reoccurring Journeys

Reoccurring journeys could be useful for people are who regular commuters along a particular route. They could set the journey up, & allow it to automatically renew on selected dates & times. This could save the driver time, instead of having to manually set up each journey, which is how the current system will be designed. I feel that this is not essential for my first development of the project.

4.0 – Approach

4.1 – Requirements

I have developed a list of requirement based on the weaknesses of current services and what my desired features are. The final product must and will reflect each of these points.

Basic User (All Users)

1) Registration System – Allow new members to register.
2) Driver Registration System – Allow members to join as drivers.
3) Search Journeys – To & From Location, Desired Date of travel. Optional time of travel. Prompt number of seats, edit route via Google Maps.
4) Display Price of Journey, number of seats.
5) Allow users to enquire about journeys.
6) Edit personal details.
7) Edit profile photo.
8) Secure Login
9) Logout
10) Allow passengers who are linked to completed journeys to review those journeys.
11) Send automated emails out to alert users about changes in journeys, or changes in profile configurations.
12) Display journeys that may be of interest to the user.
13) View privacy details

Journey

14) Display details of each journey in detail.
15) Display route that the user wishes to take.
16) Display 2 prices. One based on minimum price & one based on likely price. (Overall price divided by 2)

Driver

17) Allow driver to create new journeys. Add a to & from location, time & date.
18) Allow driver to tweak price of journey.
19) Allow driver to edit number of seats, baggage allowance.
20) Edit car photo.
21) Accept journey requests from potential passengers.
22) Display driver statistics based on reviews.
23) Cancel journeys.
24) Edit journey through the map.
25) Get a calculation of the distance.

Platform

26) The website must display & run on all major up to date browsers.
27) There must be a mobile application to track a journeys progress.
28) It must run correctly on all major operating systems.

I aim to meet these requirements to the best of my abilities, as they are what I consider the core grounds on what my project needs to meet in order to be successful.

4.2 – Algorithms

4.2.1 – Price Calculation Algorithm

I have designed an algorithm in order to calculate a particular journey price using parameters, which have been generated by Google Maps.

The only parameters required by Google Maps is the distance of the journey. Google Maps is rather accurate at doing this. I have tested it on various journeys of which I know the millage and the result is almost accurate down to the final mile.

This distance is then used in a PHP algorithm. By using the average price current price of petrol & diesel, which needs to be updated manually. This is then checked against whether the driver's vehicle is petrol or diesel. Using this it is then possible to use a third parameter which are the miles per gallon.

All of these are used to calculate the price of each journey based on then price of fuel, consumption and the distance.

The algorithm below should explain how the end price is calculated.

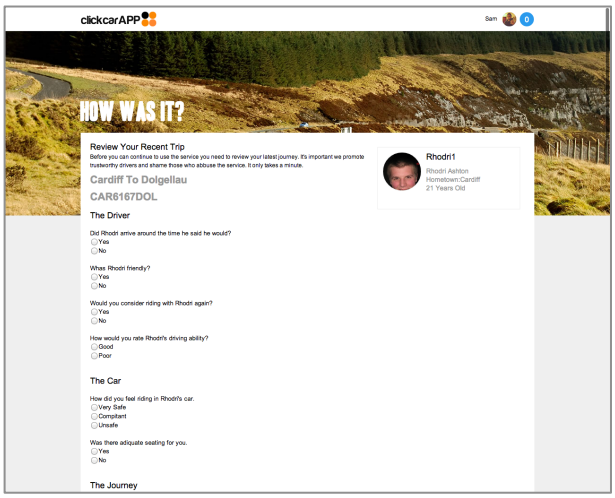
```
if(fuel == "petrol"){
    $averageMPG = 37;
    $averagePrice = 1.30; // average price of petrol
}
else if(fuel == "diesel"){
    $averageMPG = 47;
    $averagePrice = 1.35; // average price of diesel
}

// calculate price
$averagePrice = 1.33;

$travelGallons = $distance / $averageMPG;
$travelLitres = $travelGallons * 4.54; // 4.54 liters in a gallon
$travelCost = $travelLitres * $averagePrice;
$travelCost = ceil($travelCost);
```

** The average price of petrol & diesel was collected from http://www.petrolprices.com/

4.2.2 – Review Algorithm

The Review algorithm is used when a user reviews a driver's competence, and journey. This will take form of a questionnaire automatically when the journey has been confirmed as completed. Once the passengers fill out the questionnaire the data will be used in a scoring system based on various factors, such as the journey rating, car rating, driver rating, and over all experience. The ratings will appear on the drivers profile for other users to see. These will be shown as a general average rating based on all ratings from all passengers the driver has ever shared journeys with.

Below is an algorithm defining the inputs, and the process of converting the form data into ratings that can be displayed on the users profile.



Here is a screenshot of the form that will be used in order to collect passenger feedback.

It has a clean simple design that asks simple questions and requires simple consistent answers. It's possible for the reviewer (*passenger*) to leave an individual personal comment at the bottom of the page.

Below is a list of questions that will be asked.

Review Questions

1) Did the driver arrive the time they said they would?
   Yes
   No

2) Was the driver friendly?
   Yes
   No

3) Would you consider driving with the driver again?
   Yes
   No

4) How would you rate the drivers driving ability?
   Good
   Poor

5) How did you feel in the drivers' car?
    Very Safe
    Competent
    Unsafe

6) Was their adequate seating for you?
    Yes
    No

7) Was the lift quicker / more convenient than public transport?
    Yes
    No

8) Was the lift cheaper than public transport?
    Yes
    No

9) Would you use ClickCar again?
    Yes
    No

10) User leaves a general comment about the journey.

Review Summary

The above review questionnaire could be optimized depending on how well the results display when combined together in the trial run. I may find that there is a need to address other issues.

Review Algorithm

The algorithm follows a point system. Based on the most important factors in my personal opinion I have rated each positive & negative response for each question. Each section will have a maximum addition result of 10. The closer the rating is to ten the better.

Algorithm - Driver Rating

```
// y = yes
// n =no
// arrival time, late or on time
    if(arrival == "y"){
        driverRating = 2;
    } else {driverRating = 1;}

    // friendlyness of driver
    if(friendly == "y"){
        driverRating = driverRating + 2;
    } else {}

    // safety of drivers driving ability
    if(safety == "y"){
        driverRating = driverRating + 3;
    } else {}

    // would the passenger ride with this driver again
     if(ride == "y"){
        driverRating = driverRating + 3;
    } else {driverRating = driverRating + 2;}
```

Algorithm – Punctuality Rating

```
// punctuality Rating
if(arrival == "y"){
        punctuality = "10";
    } else {punctuality = "2";}
```

Algorithm – "Use ClickCar again?"

```
// the maximum value for this question is 1, as there is no
rating displayed for this question. It will simply be for
research reasons.
// would you use clickcar again
    if(again == "y"){
        $clickcar='1';
    }else {clickcar='0';}
```

Algorithm – Journey Rating

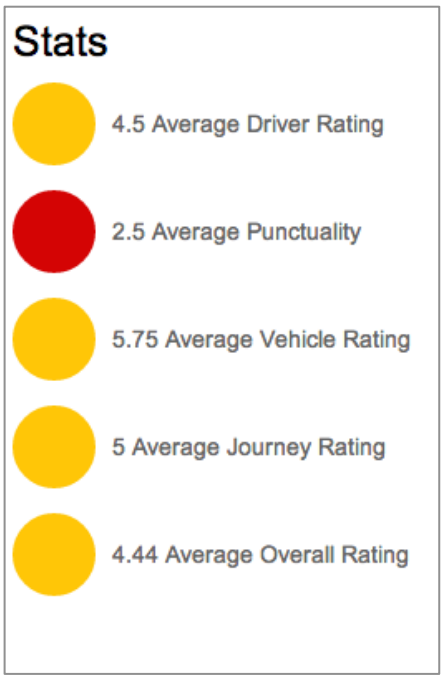
```
// calculate journey rating
    if(quicker == "y"){
        journeyRating = journeyRating + 3;
    } else {journeyRating = journeyRating + 1;}

// was it cheaper than public transport
    if(cheaper == "y"){
        journeyRating = journeyRating + 3;
    } else {journeyRating = journeyRating + 1;}

// did they arrive on time
    if($arrival == "y"){
        journeyRating = journeyRating + 4;
    } else {}
```

Algorithm – Journey Rating

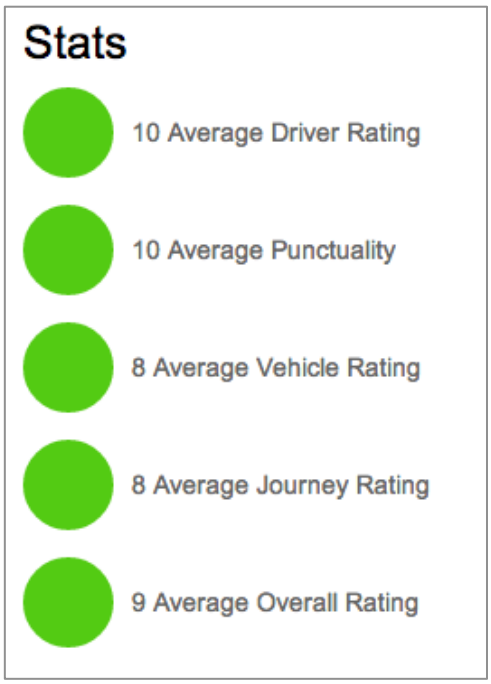
```
//calculate rating for the car

    // how cheap is it to run the car
    if(cheaper == "y"){
        carRating = carRating + 3;
    } else {carRating = carRating + 2;}

    // car comfort
    if(carComfort == "1"){
        carRating = carRating + 3;
    }else if(carComfort == "2"){
         carRating = carRating + 2;
    }else {}

    // would you ride again with this driver in this car
    if(ride == "y"){
        carRating = carRating + 1;
    } else {}

    // was the seating adequate
    if(seating == "y"){
        carRating = carRating + 3;
    } else { carRating = carRating + 1;}
```

Algorithm – Displaying Results

Displaying the results will combine the 4 factors in order to display a colour coded rating system. It will use a traffic light system. Red is the worst category, amber is mid range, and green is in the highest category.



A display of the colour coded system for a particular driver.



A good review based on user ratings for a particular user.

Below is an algorithm for showing the colour-coded user rating system.
It's based on one particular feature. The other ratings are in the same format.

```
if($driver < 3){$driverC="#d40404;";}else if(($driver >2) &&
($driver < 7)){$driverC="#ffc607;";}else if(($driver >6) &&
($driver < 11)){$driverC="#53cb13;";}
```

4.2.3 – Inserting Into Database Method

Every time a new row is inserted into the database something similar to the following code will execute.

```
$sqlInsertUser = mysql_query("INSERT INTO user
(username, email, password, phone, phoneCode,
emailCode) VALUES('$username', '$email', '$password',
'$phone', '$phoneVerification', '$randomEmailCode')")
or die (mysql_error("Could not insert data"));
```

4.2.4 – Searching Algorithm

The searching algorithm will perform with 3 to 4 parameters. These parameters include the destination and the starting point of the journey, as well as a date and a time. The time parameter will be optional and will only be used if an exact journey matching the users criteria can be found.

Several queries will be made around the date. The algorithm will search for journeys that are scheduled on a particular date, one day after, & one day before. This will give the user a wider chance of finding a journey that will appeal to them. Thus the aim is that this feature will make the service more useful to the user.

The searching algorithm performs it's search in two primary ways. It first looks for a match by location name. For example it will use "*Cardiff*" as a search term. The other method is to look up the postcode for a particular town or city. In this case it would search "*CF*" rather than "*Cardiff*". The idea of this is that a post code has more than one town in it's region. The postcode area "*CF*" covers not only Cardiff city centre itself, but also areas of the valleys, Barry & Penarth. This makes the search wider and more useful to the user.

4.2.5 – Display Useful Journeys Algorithm

The useful journeys section will be used in order to show the user potential journeys they can take before performing any search results. This should entice the user to use the service just by logging on. It will create a personalised homepage that is something other services don't offer.

In order to work it takes the information filled out when the user registered such as journeys the user may be interested in. It uses these arrays of data in order to perform live searches every time the page is loaded and reports back to the user with results.

The second method of searching useful journeys is to use the user history of journeys they have sent enquiries for in the past. The enquiries about a journey

don't have to be accepted by the driver for them to be added into the users history. This data is then used to perform searches and collect journeys that the user may be interested in. It can also search for town/ cities that are close to where the user has shown interest in before. This can once again assure that the journeys that users' see are of the correct area for them.

Below is the algorithm of how the "*Useful Journeys*" are displayed.

4.2.6 – Offering A Journey Algorithm

This algorithm is used when a driver wishes to create a journey. In order to do this the system first checks if the user logged in is registered as a driver. It checks the driver database.
If the user is found in the drivers database it shows the create journey page, or else it shows a driver registration page.

Google Maps

After the user has entered all the relevant data for a journey it calculated the distance using Google Maps. Google Maps has a built in navigation tool that is used in order to calculate the distance between two locations via a particular route.

```
var rendererOptions = {
  draggable: true
};
var directionsDisplay = new
google.maps.DirectionsRenderer(rendererOptions);
var directionsService = new google.maps.DirectionsService();
var map;

function initialize() {
  //directionsDisplay = new google.maps.DirectionsRenderer();
  var uk = new google.maps.LatLng(55.279115, -1.098633);
  var mapOptions = {
    zoom:5,
    center: uk
  }
  map = new google.maps.Map(document.getElementById('map-
canvas'), mapOptions);
  directionsDisplay.setMap(map);


   google.maps.event.addListener(directionsDisplay,
'directions_changed', function() {
    computeTotalDistance(directionsDisplay.directions);
  });

  calcRoute();
}
```

```
function calcRoute() {
  var start = document.getElementById('departure').value;
  var end = document.getElementById('destination').value;
  var dest1 = document.getElementById("passingPlaces1").value;

  if(dest1.length < 2){dest1 = end;}

  var request = {
      origin:start,
      destination:end,
      waypoints:[{location: dest1}],
      travelMode: google.maps.TravelMode.DRIVING,
      unitSystem: google.maps.UnitSystem.IMPERIAL
  };


  directionsService.route(request, function(response, status) {
    if (status == google.maps.DirectionsStatus.OK) {
      directionsDisplay.setDirections(response);
    }
  });
}

// calculate distance
function computeTotalDistance(result) {
  var total = 0;
  var myroute = result.routes[0];
  for (i = 0; i < myroute.legs.length; i++) {
    total += myroute.legs[i].distance.value;
  }

// convert from KM to Miles
  total = total / 1000;
  total = total/ 1.6;
  total = Math.round(total);
  // display total miles
  document.getElementById("total").innerHTML = total + "
miles";

 // set distance hidden field value
 var distance = document.getElementById("distance");
 distance.value = total;
```

This code was adapted from .....WRTIE HERE

Because the code produces the results in KM there is a need to convert it into miles to meet the UK market. The total is divided by 1000 & then divided again by 1.6. This is because there are 1600m in a mile.

This distance is then used in order to calculate the price. See point 4.2.1.

4.3 – Design

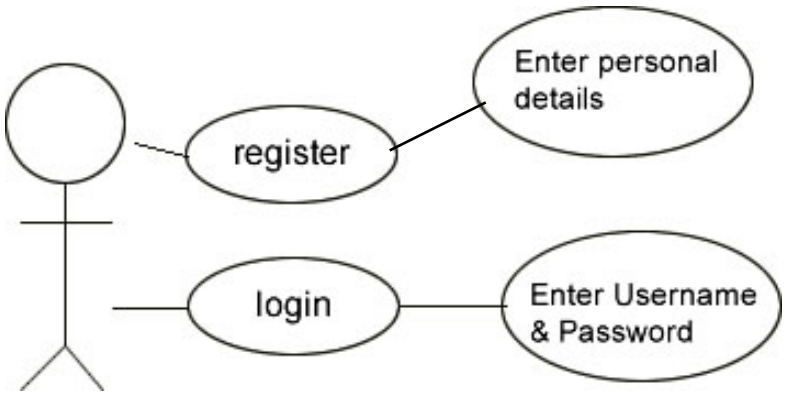
4.3.1 – Use Case Diagrams

The following use case diagrams focus on how each user has different permission levels & different abilities on the site.

4.3.1.1 Basic Users

This area focuses on the features that every user can access. In order to access the system the user must first become a member & register. Random visitors to the site will have very limited access to many of the pages such as viewing profiles, messaging users, creating journeys, searching for journeys & enquiring about journeys.

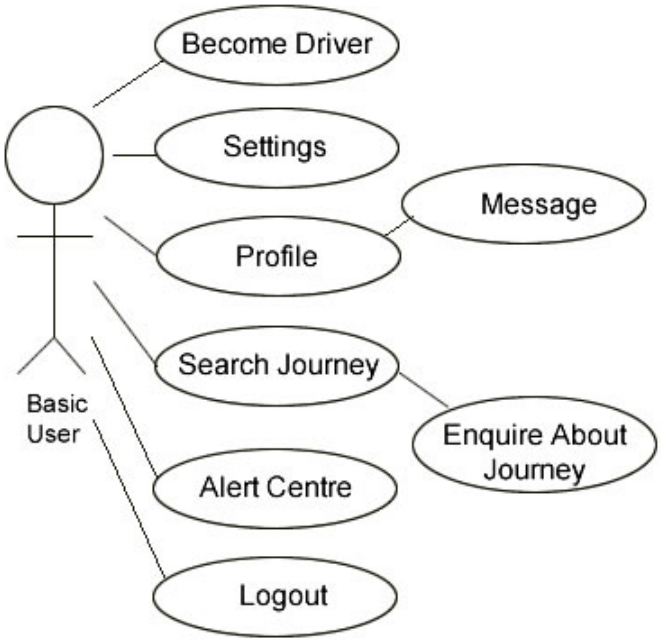
New User



A new user refers to a random user first visiting the site.

They are given two options. Register or Login.

Basic User – Logged In

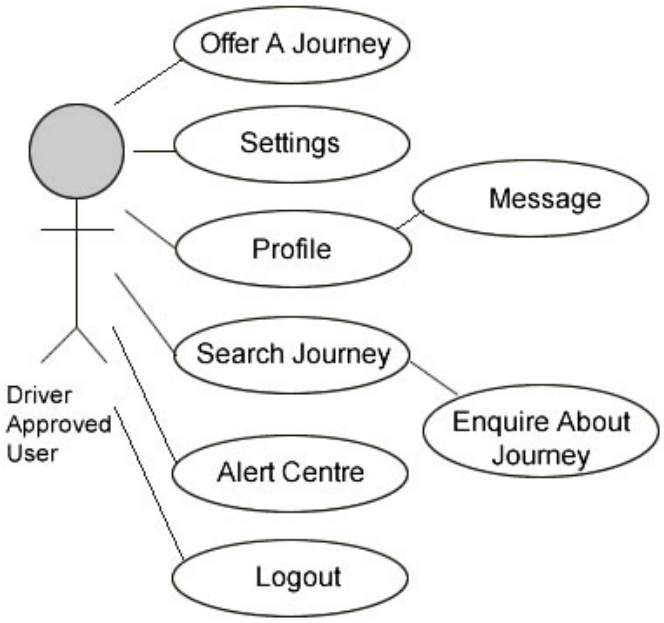


The basic user is able to do everything except for create a journey.

In order to do this they must apply to become a driver. This involves entering details about their vehicle.

The settings page will reveal more options.

Driver Approved User – Logged In

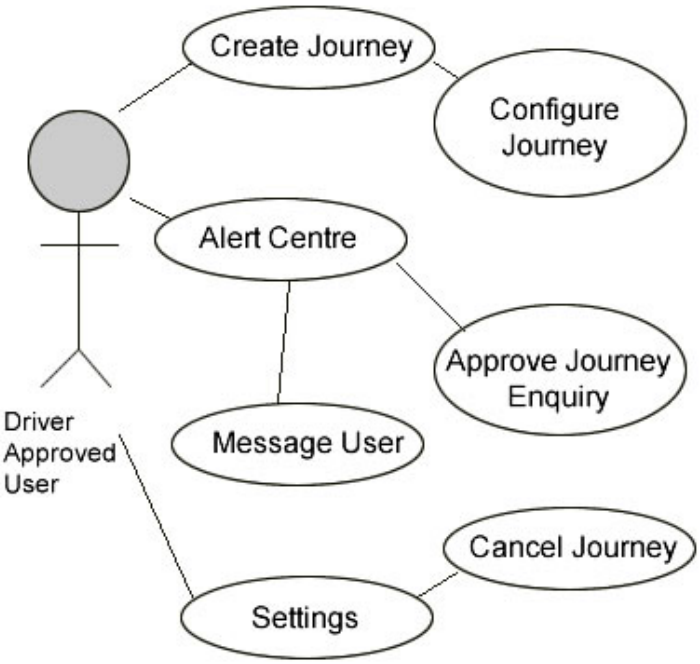


A “Driver Approved User” refers to users who have added a vehicle to their account.

This user is able to create journeys, unlike a user without a drivers account.

All other features remain the same.

Driver Approved User – Create Journey



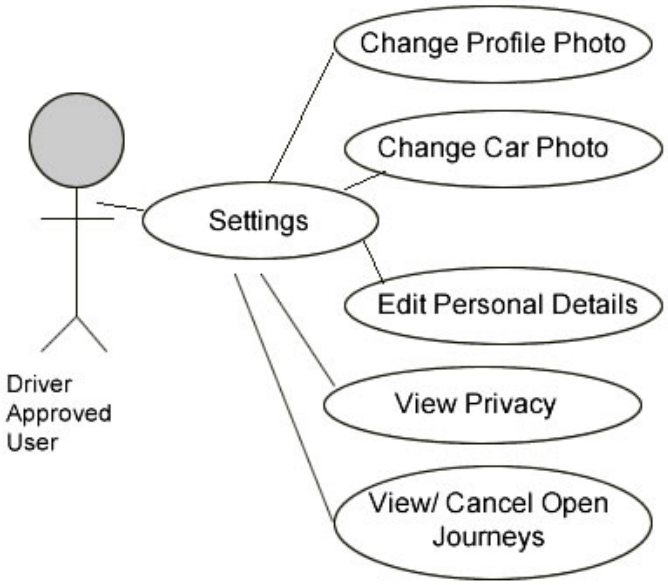
A driver is able to create a journey, and configure the price they want to charge.

They are also able to approve other users who want to ride on their journey.

They can also message users with questions, & general conversations. These are all private.

In settings, they are able to cancel journeys. An email will be sent to all passengers to tell them a journey has been canceled.

Driver Approved User – Settings

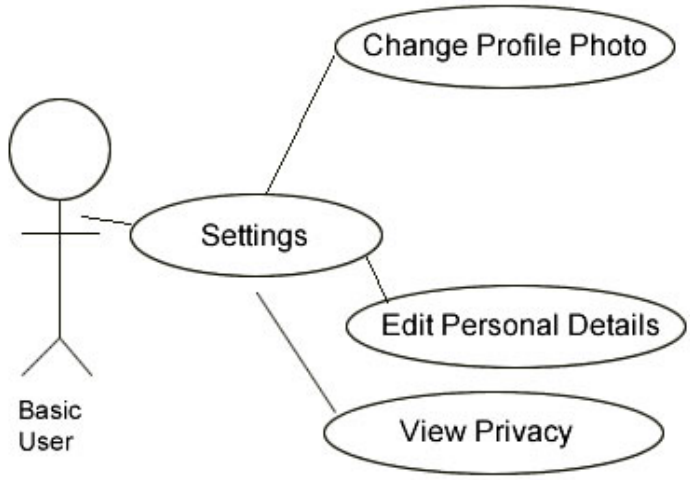


A driver-approved user is able to change their profile photo, car photo & edit their basic user details such as address & name.

They are able to view what details are shared about them & which details are kept private & only used in order to improve the service the end user receives.

A user is able to cancel any open journeys up to 48 hours before they are due. It is not possible after that.

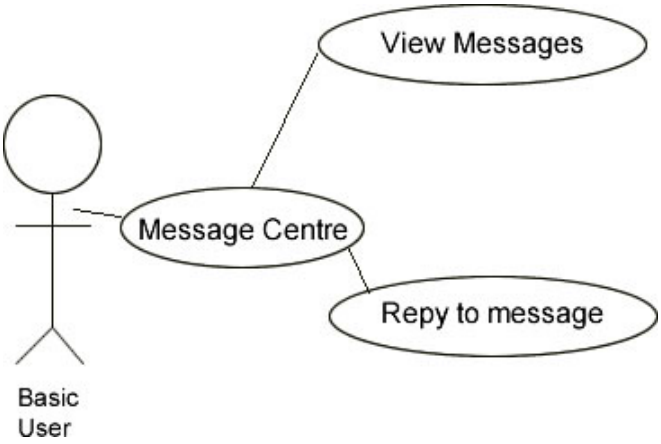
Basic User – Settings



A basic user is a user that does not have a vehicle attached to their account.

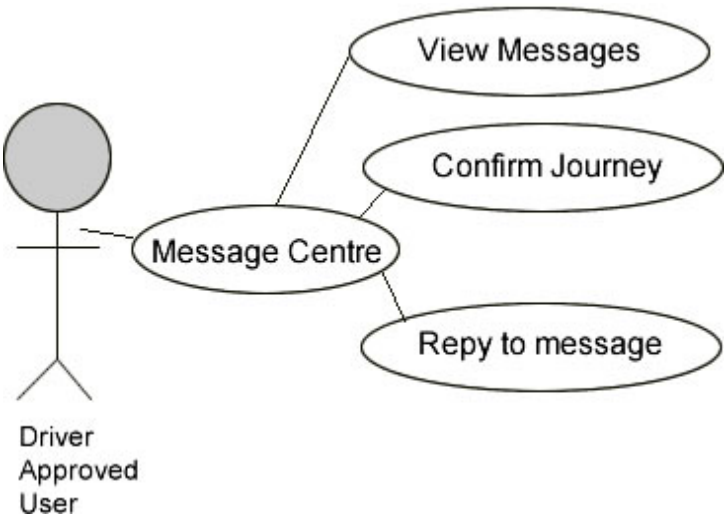
They have a more limited settings page. Unlike a driver they cannot edit any car details, add a car photo or cancel any journeys.

Basic User – Message Centre



A basic user is able to use all the basic features of the message centre. They can send & receive messages from all users.

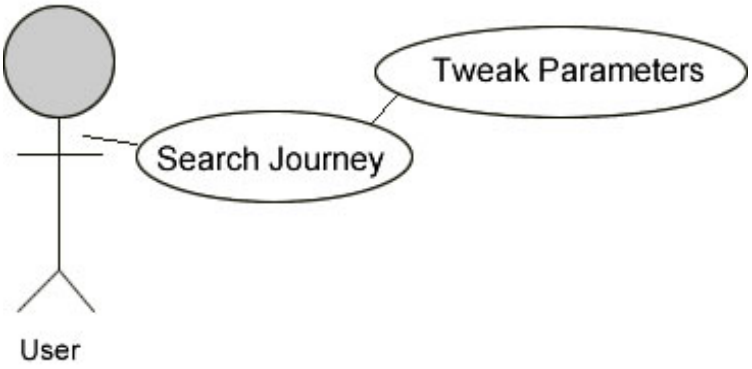
Driver Approved – Message Centre



A driver can access all the features of a basic user such as send & receive messages.

However they can also accept journey requests. They will have the ability to check the profile of the passenger requesting to join the journey before they accept.

All Users – Search journey



All users can search and tweak their search preferences such as start & end point of their journey, as well as the date and time.

These UML diagrams are guides only. The way I have structure the development of my project does allow me to tweak each of the major interactions if I feel I can adjust my schedule in order to include new functions & features.

4.4 – Screen Layout Design

Screen layouts designs are good ways to invasion how the general flow of the system will look. As well as performing well I feel that the visual design is important.

The design face was a case of studying what the end product needed to do & drawing up a design. Using a whiteboard and marker pen I drew out each design, tweaking it around until it looked satisfactory. I then used *Balsamiq* in order to adapt & perfect my screen designs.

My design is very clean. It contains no unnecessary additional content and each page serves a purpose. The general layout & design is consistent throughout the website linking all pages. Even the mobile app will follow a similar, stripped down minimal design, again, serving a purpose without necessary additional content.

Why *Balsamiq*

Developers use *Balsamiq* widely for it's easy to use method. It allows fast graphical prototyping and is self-explanatory without having to learn any additional languages or skills.

Every main process or action taken by the user will be drawn out.

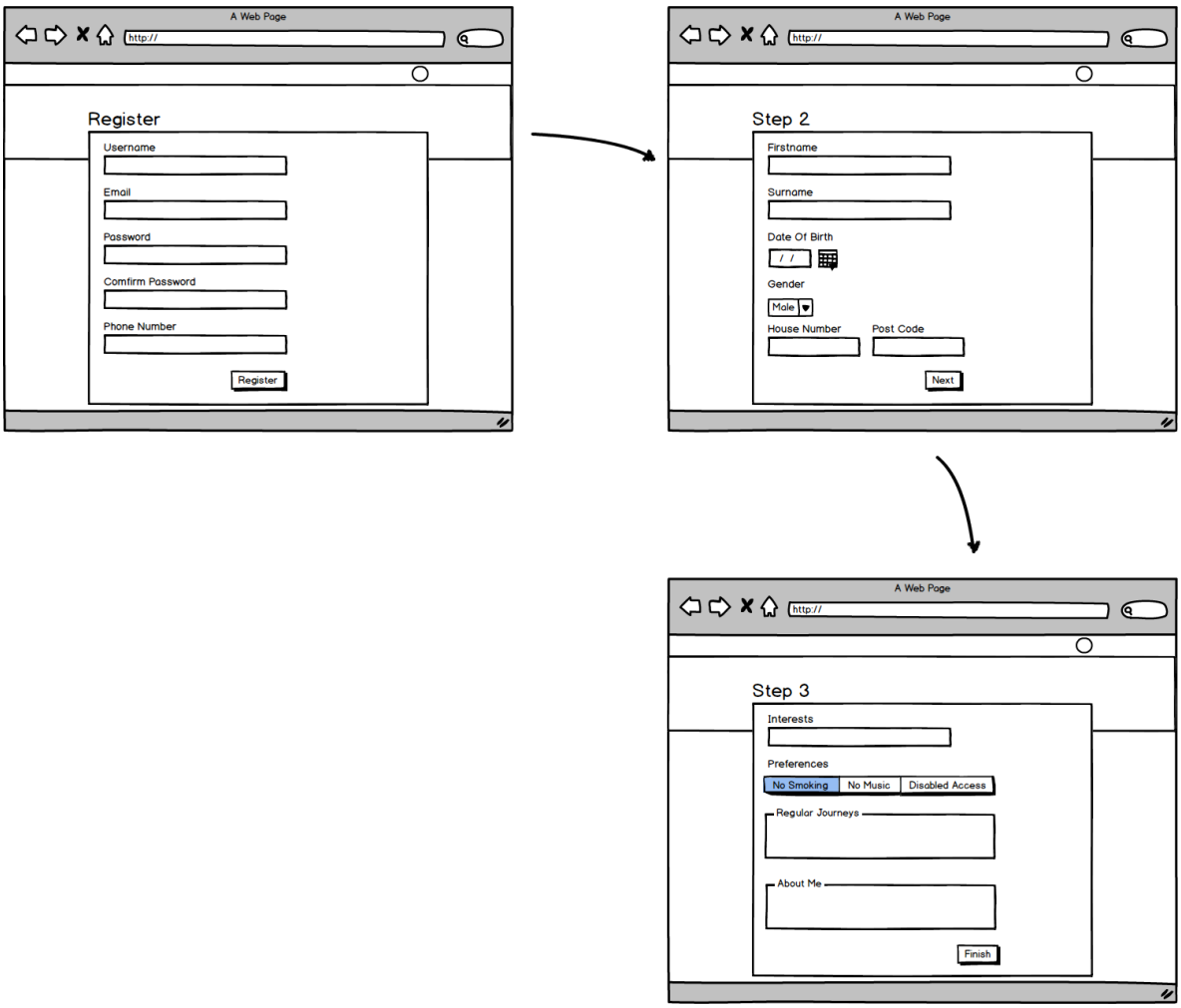
I will be using *Balsamiq* in order to draw out what these pages will look like. The good thing about this program is that it has a lot of present features such as browser windows & mobile phone screen layouts that make the credibility of any given design more foreseeable & realistic. It allows you to see how something will look without having to write any code at all.

Illustrations

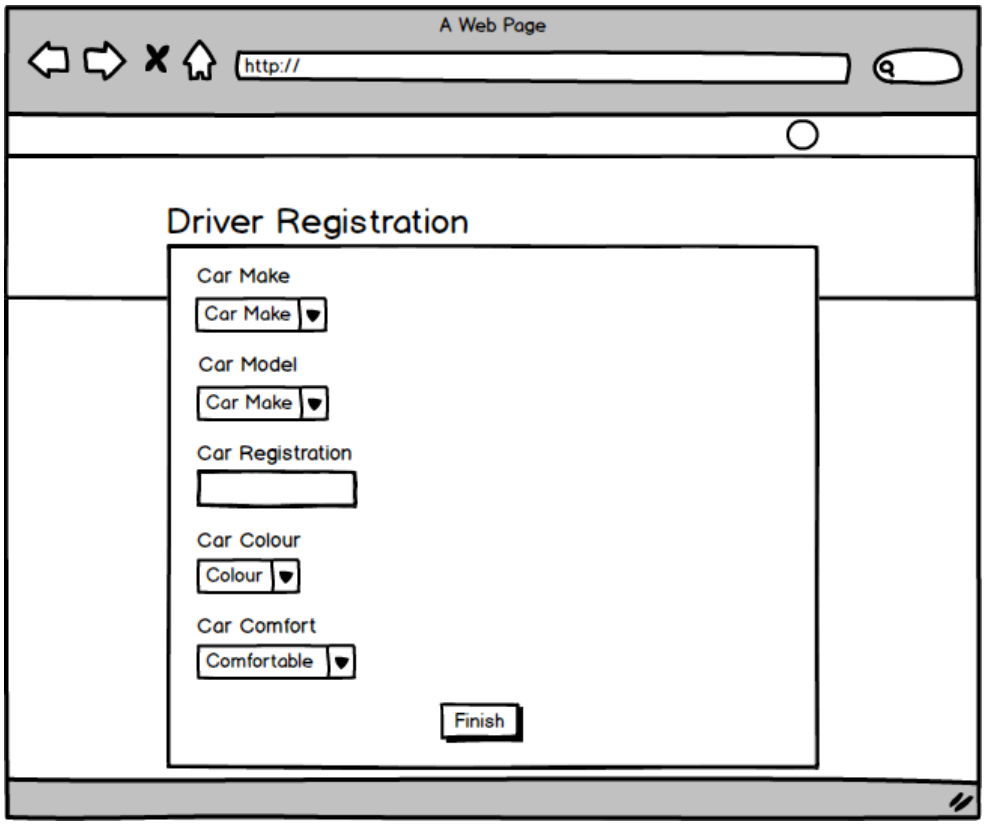
The next section will focus on the designs drawn for my project.
They will not cover every possible page as there are many but it will cover the main features.

It will be possible for me to adapt some of the pages where some of the features are similar to others.

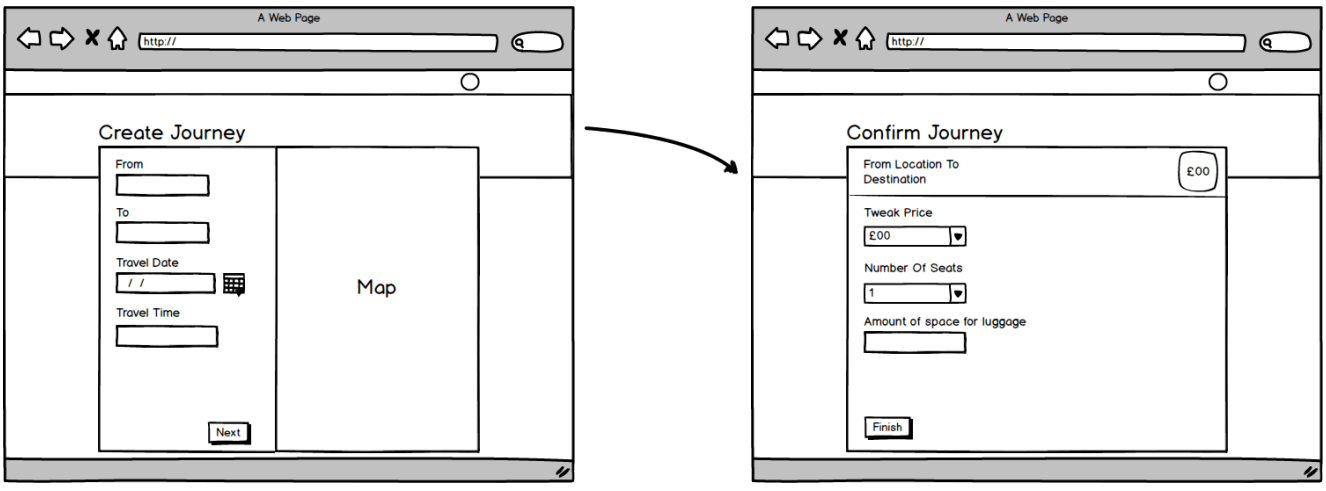
Registration Page



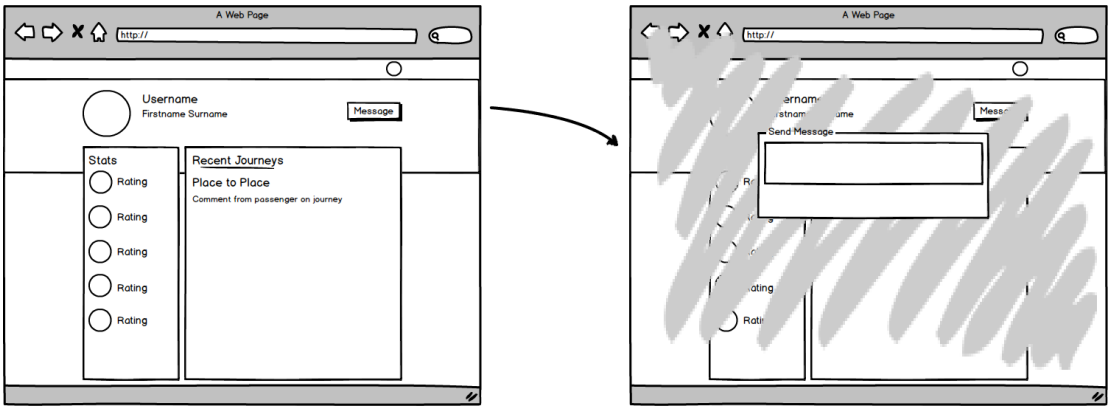
Driver Registration Page



Create Journey



Profile + Message User

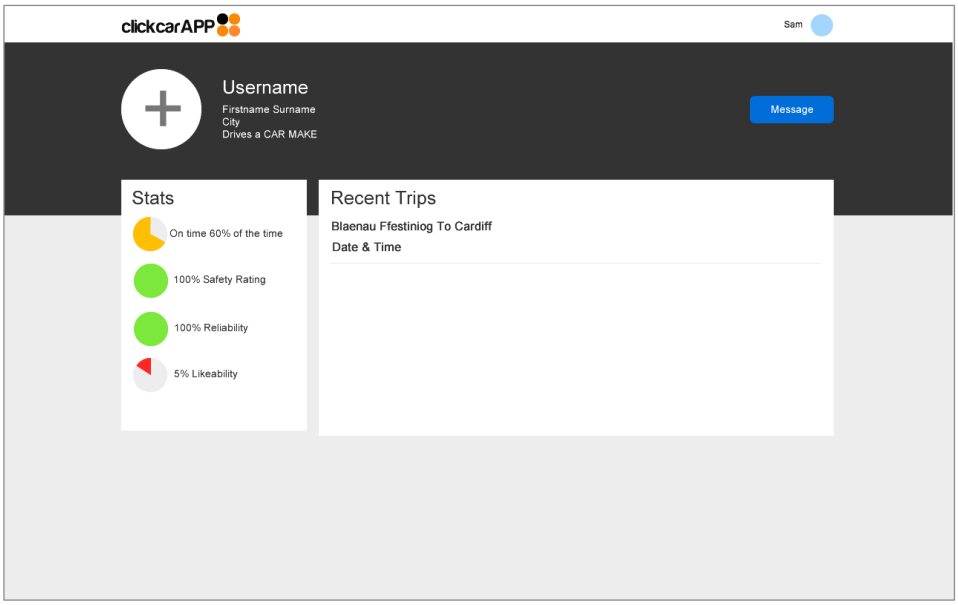


** The message button will appear when the user is anybody else's profile except for their own. When they are on their own profile they will see an option to edit their profile. This button will link them through to the settings page.

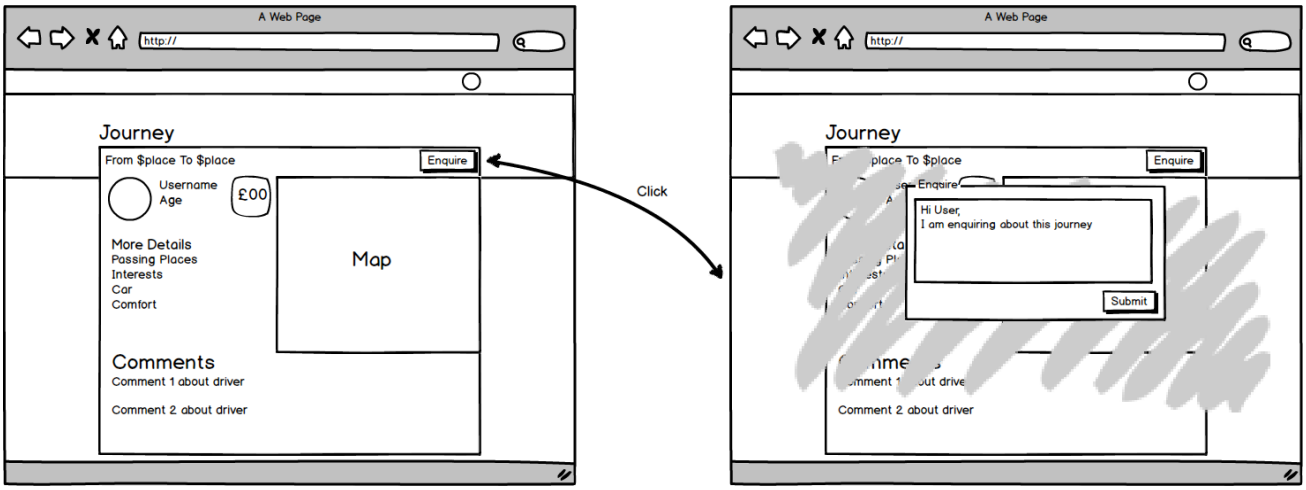
This should follow an important principle of web designing, where a user should be able to get to their desired location in around 2 clicks or less.

The message button will display a customized window that will allow the user to input their message & send it directly to the profile owner's inbox.

In addition to some sketch style drawings I have also done more detailed drawings in Photoshop in order to show a more in depth design. These type of designs allow to visualize a direct copy of what the page will look like. I could simply follow this design directly & have a page that looks exactly like this.



Display Journey

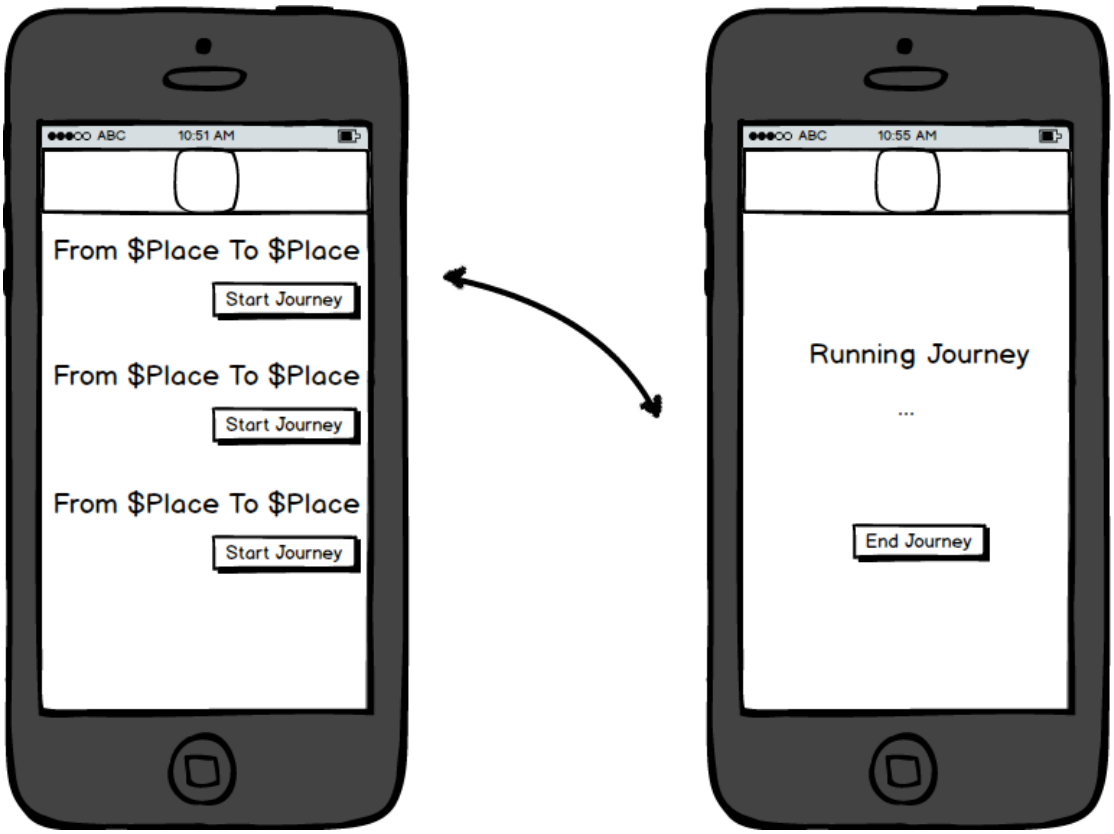


This page will display any journey that has been created by the driver. It will show all details of the journey & allow any interested registered passengers to enquire about the journey.

The page will also show the reviews from the last 5 journeys. This should give any potential passengers an overview of the driver.

Mobile Application

Originally the mobile app will only used by drivers. It's aim will be to track any given journeys progress where passengers are travelling on the journey.



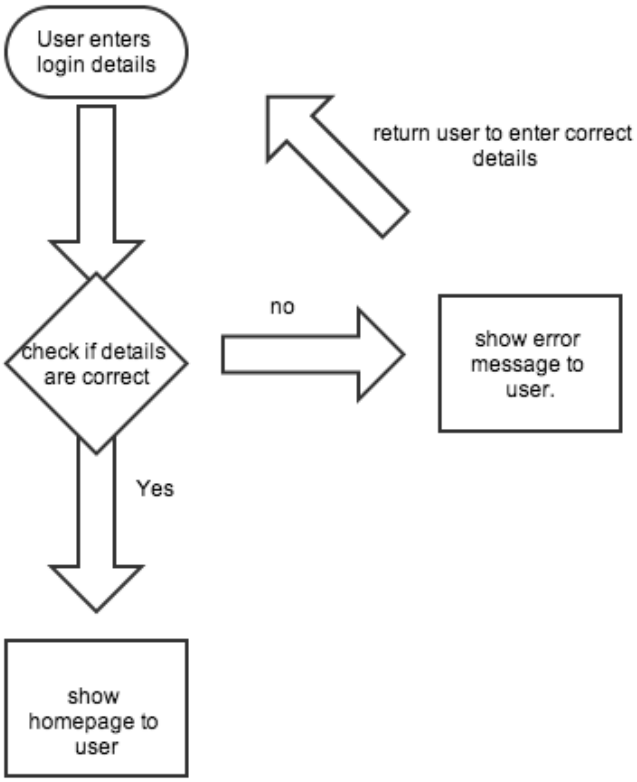
The App will not be available on the App Store, or the Google Play store, however there will be a link on how to download the app, & the source packages attached in the attachments.

4.5 – Flowcharts

Flowcharts are an effective way of explaining how specific algorithms run. They can often be more useful than words. Most of my code is based on the following three flowcharts. They simply use different variables but essentially use the same login time after time.

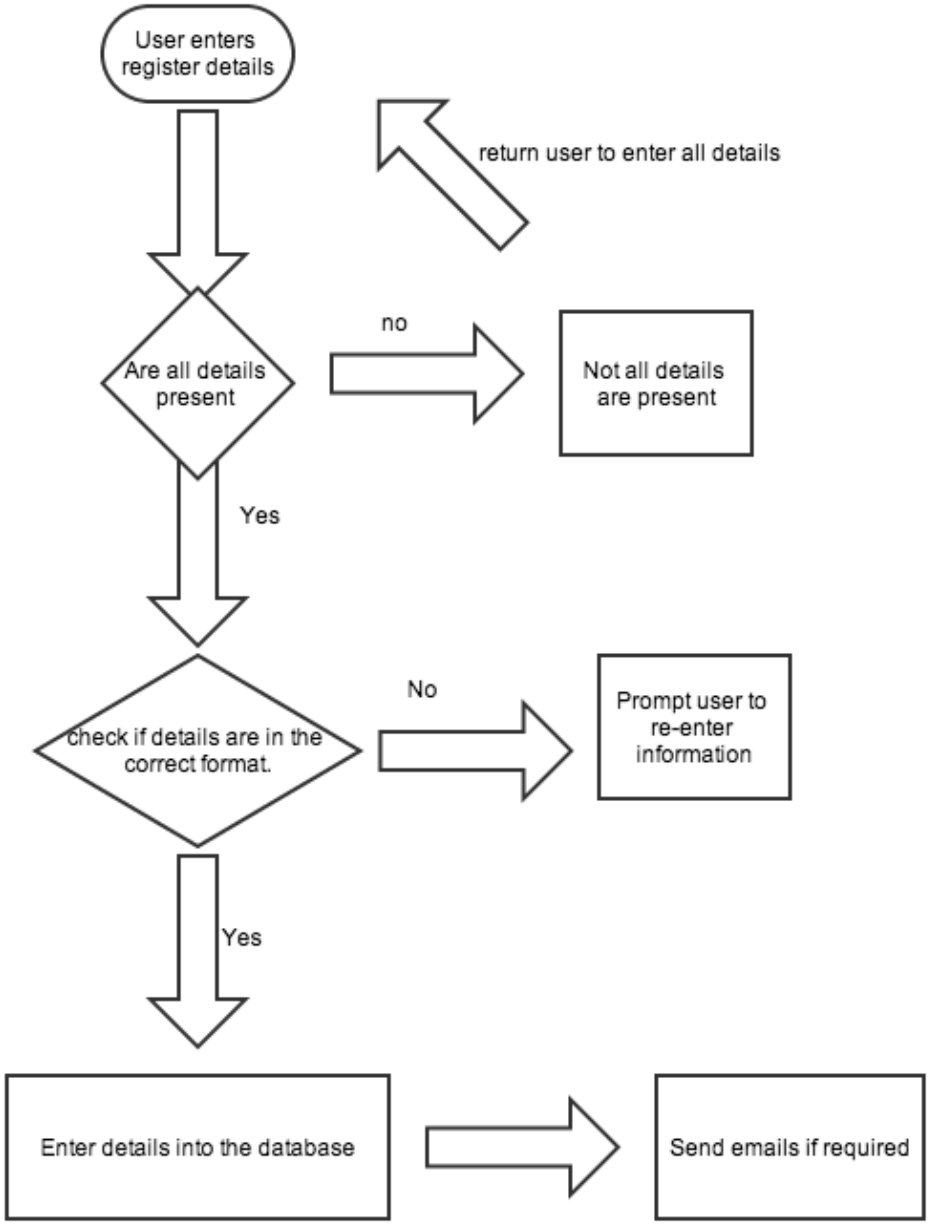
4.5.1 Login Flowchart

This flowchart is rather simple where that the user enters data, that data is put through some security filters. It is then checked against the records on the database. If any values are returned then the user will be logged in. If not then it will display an error message.



4.5.2 Register Flowchart

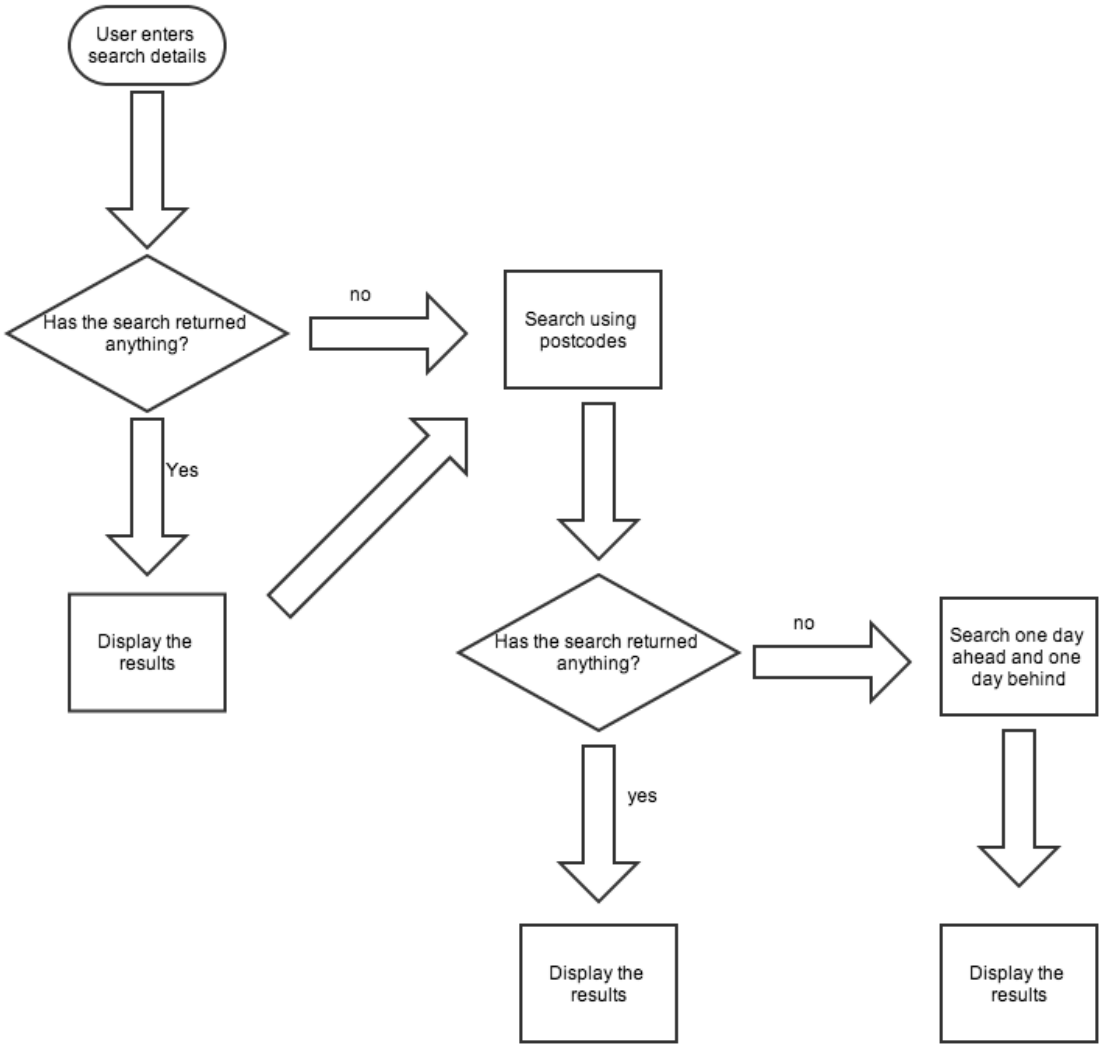
The register flowchart shows how data is not only checked if is present but also if it is in the correct format. This happens with many other pages on the site, not only the register page. I have only included one example as it's rather repetitive. It usually checks if the format of a post code, car registration, phone number or email address is correct. There are various short algorithms & functions that do this.



4.5.2 Search Flowchart

The search feature is something that I want to keep perfecting. I have many future improvement that I want to carry out that are stated later on in my project report.

This search feature will keep trying the next best solution until it either returns a search result or runs out of options to try. If it returns nothing in the end it will simply return a message saying “No Results Found”.



4.5 – Database Design

The database in this particular system is what drives the application. The application reads from the database on every page giving relevant information to the user. Whether it's control over different user access levels or collecting journey information, the database is involved.

The system is designed to use a regular MySQL database that is running on my personal server. I have used MySQL in the past & so it is the most obvious choice for me.

Pros of using MySQL

- I have a strong skillset in MySQL.
- There is wide support for it.
- It fits all my needs.

Cons of using MySQL

- MySQL doesn't scale as well as other database systems [6]
- It's features are rather basic compared to some other database systems.

In order to demonstrate how my relational database system will work I have drawn up an entity relationship diagram. The key purpose of this is for me to have a better understanding of the structure of the database system. As mentioned before I see the database as the heart of a system like this. Without it the system will not work. Therefore it is vital that the core database system is accurate and meets all my needs now & in the future.

What Will I Need?

Based on the requirements I need the following tables in my database.

These are the primary tables within my database.

User - This will hold user information when they first register.
Driver – This will hold all car information.
userJourney – Contains all journeys created
Reviews – Contains all reviews

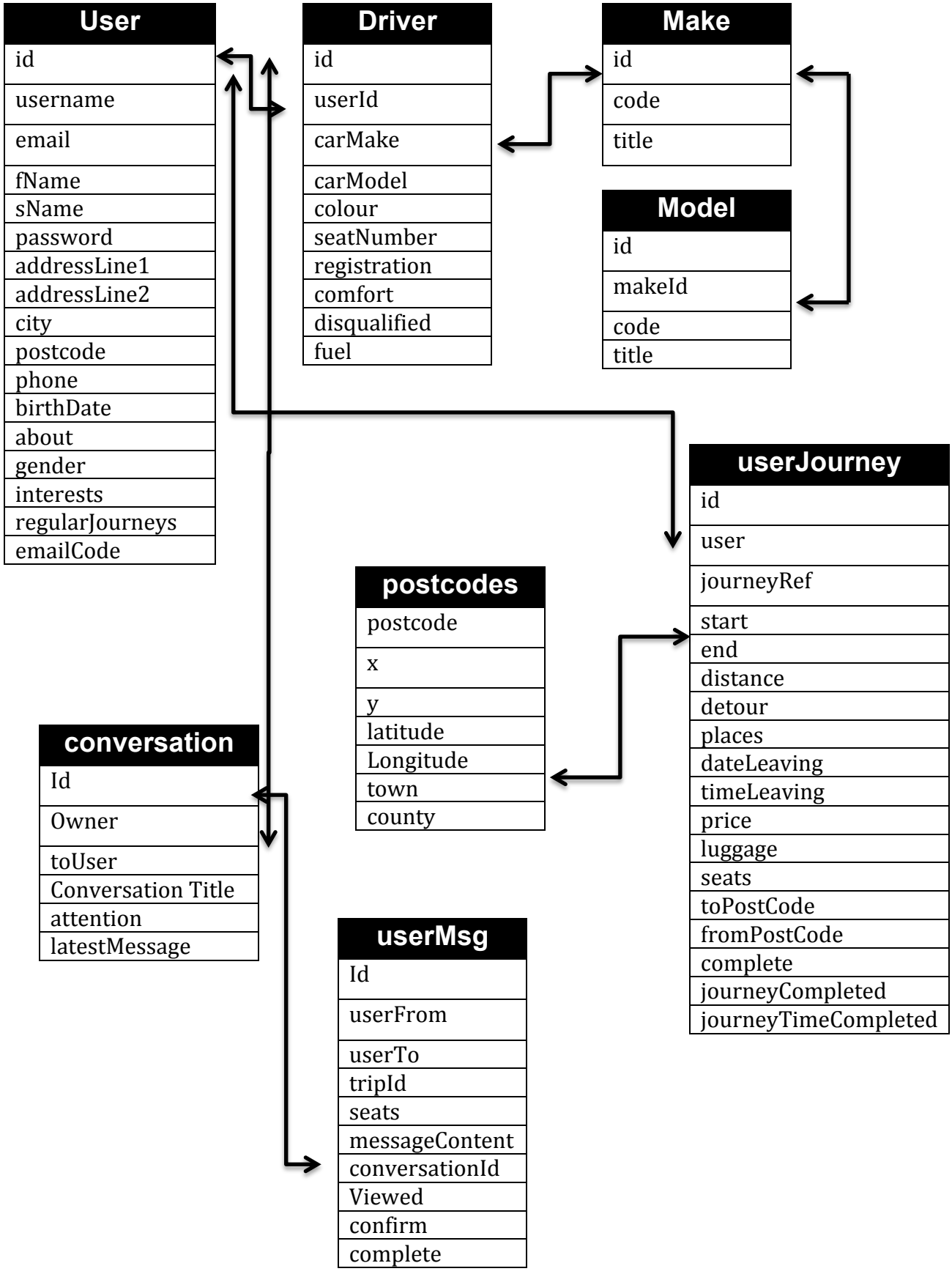
These are additional tables for certain algorithms to run.

Conversation – This holds all subjects of the message centre.
userMsg – This contains the raw messages relating to each subject.

The following are downloaded from open sources

Make – Collection of vehicle makes.
Model – Collection of vehicle models.
Postcodes – Collection of all towns & postcode region in the UK

4.5.1 Relational Database Diagram



4.0 – Implementation

4.1 – Tools

In order to code my project I have been using software called *Brackets.* [7] Brackets is a text editor tool which has been written in html. It's very lightweight and saves any working state on screen when you close the application. I find it practically useful for it's automated suggestions when coding & also it's built colour picker which generates the hexadecimal number for my colour scheme.

I shall be coding the majority of the program manually except for the parts it is either not relevant or of a benefit to write myself or the segment that I need has already been written for me. All credit will be given to borrowed code segments above the code in my software. I have made the relevant checks that I am allowed to use the author's code.

Some segments of code are not meant for commercial use & so some research & editing would needed to be done before my project would go into the mainstream market.

Why not BootStrap?

Bootstrap is becoming the new friendly face of the web. It is used my various web designer's & companies today. Any possible widgets & CSS styled components are ready to be used.
So why not use Bootstrap?

I find that my design skills are very high and so I will be able to customize my software to my needs by doing all of my own CSS. I am able to turn Photoshop illustrations into fully working web pages.

Many web pages today have transitions on them. Requests can be sent to the server without having to refresh the users desktop site. As we have such technology today, I am going to use this to my advantage to have speedier search.

JQuery

JQuery is a JavaScript library that enables you to write JavaScript code in a sort of short hand. What this means is that once I include the JavaScript library I can use jQuery animations. JQuery also offers a method of using a simpler version of AJAX. AJAX usually deals with passing data over form an HTML front end to a backend page that is written in a backend language such as PHP.

The great thing about JQuery is that there are plenty of additional libraries. The JQuery UI library has premade features such as calendars, progress bars, & interactive features like drag & drop. It makes web pages feel far more like a desktop application.

JQuery UI

JQuery UI had some useful features which saved me a large amount of time. I used a *date-picker* from the JQuery UI library. This provides a working well designed solution with minimal effort.

Google Maps API

Google Maps has an API that is available for multiple platforms. As well as dedicated libraries for mobile & tablet devices it also has support for web browsers. It allows you to embed a script to the Google Maps API, that in turn renders a map on the web page.

The API comes with various built in features such as navigation & distance calculation. You can decide whether you want to drive, walk, or ride a bike. Of course for this application I am using the driving route navigation system. You can also choose between metric & imperial.

There are other features available such as drawing on the map. This would be useful for a more sophisticated search system. I could draw shapes around a given route, finding the names of towns and addresses within those parameters. Unfortunately I wasn't given enough time to develop this system therefor I have chosen a simpler route by downloading a database of all towns in the UK, & their postcodes.

GitHub

GitHub is a useful tool for developers. It allows you to save a backup of your work, but it also allows you to store various stages of your work. This enables you to back track if any major errors are made, making it easy to return to an earlier stage of your software and make the relevant changes without having to redo the entire project.

I have 3 copies backed up of my work. The first is on my local hard drive; the second is on my server, which hosts the web app. The third is a copy on GitHub. This way I am not able to loose any of my work if the inevitable happens.

PHP MyAdmin

Of course I could use a text based terminal in order to manage my database. This would be no problem. The only issue I found with this is it doesn't allow me to concentrate on seeing a clear view of the data. Especially while building my project I needed to keep a close eye on my test data. Making sure data was entered correctly, & making sure data linked up correctly. I found that the easiest way to do this was to install PHP MyAdmin.

PHP MyAdmin is something I have used many times, & so my familiarity with this software has come in useful to allow me to concentrate on a sophisticated system.

PHP MyAdmin also has useful features in order to import data. This has been especially useful when I needed to import outsourced databases. With research I was able to find the relevant databases on repo's from GitHub via helpful links on Stack Overflow. After downloading these large files it was easy to upload them and implement them into my database. I didn't have to build around them as they just fitted in with my already developed software. Most of the imported database tables had around 7000 rows. Copy & pasting these one by one would have been unreasonable. I have given credit to these sources in the acknowledgements as well as in the source code itself.

FileZilla FTP

My choice for using FileZilla is simply it's because it was already linked up to my cloud server. FileZilla offers a simple file transfer system that works effectively. All the html & front end design was done on a local MAMP server, however as I wanted to build on a platform that would be accessible from any location I chose to upload all my work onto the server & have a live developers version of the site running. It also allows me to play around with download speeds. If I have any large graphics that are slow at loading it would be hard to tell on a local server. Therefor FileZilla comes in useful as a gateway to my server.

4.2 – Tools, Mobile Application.

I intended to write my mobile application in java for Android. The main reason for this is because my mobile phone is an android device. This would have enabled me to test my mobile app through & through.

Issue

Due to my short development period I was unable to develop a new application which would work along my website. As it was an important part of my project that the mobile application worked along side the website I found another method that would allow me to use existing code & convert it into an application.

Solution

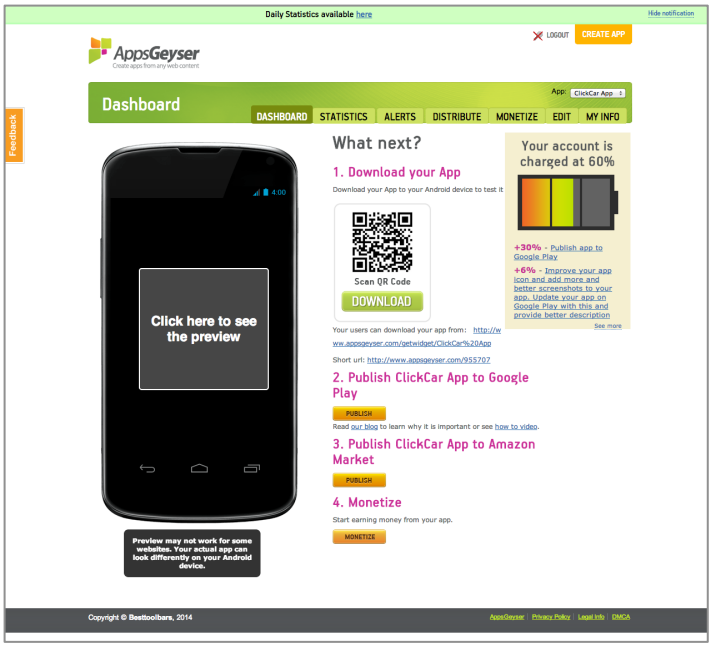
The mobile app was needed during the time when a trip is running. The driver needs to begin the journey, & the app will track the journeys progress. Once the journey is complete it runs a script, which emails a receipt to the passenger. This email basically informs the user that the trip is complete. For any reason the passenger was stood up, then it will be possible for the passenger to give the driver a bad rating, thus reflecting badly on their personal profile.

I decided that I would develop a smaller, scaled down website that would be perfect for touch screen mobile phones. It only tracks the status of any given

journey by the driver. The application first needs to have a username & password which will log in as normal, using the same script found on the main website. Next it will display any journeys that are yet to be made. Once selected it runs & tracks the drivers town location.

As it would take too long to develop an app & a website I did have to research into other methods. I found a simple tool online that would make building an app much easier.

App Geyser allows you to choose a type of app, in my case it would display a website [8]. I then had to simply just apply the URL I wanted the app to direct to and the app worked.



A screenshot taken from App Geyser.

4.3 – Method

Database

Originally I intended to write the entire project using object oriented PHP. This would possibly be the most reasonable solution as it is a rather large project. Due to a short time period available for developing my project I chose not to take this approach. It would have been more professional to write classes and objects, and may have made some features in my code work smoother. There are some parts of repetition that are unavoidable, such as error trapping data that is being uploaded into the database.

Making sure that the system wasn't easily *hackable* was an important step as some of the data is very sensitive.

For every data input I always put the data through a type of filter. It makes sure no characters are entered that could modify the code as it's run.

```
$input = mysql_real_escape_string($input);
$input = stripslashes($input);
$input = strip_tags($input);
```

"Stripslashes" & "strip_tags" makes sure no slashes or tags are entered into the MySQL query. Without these it would be possible for potential hackers to MySQL inject the system, entering their own query either downloading all personal data of each user, or simply deleting all the data in the database. Something like this would be catastrophic for my system.

"mysql_real_escape_string" also removes some potentially dangerous characters which can be used to MySQL inject a database. It removes the following characters. [9]

```
\x00, \n, \r, \, ', " and \x1a.
```

Connecting to the Database

Every page that connects to the MySQL database will connect to the database every time a functions needs to connect.

```
$db_host = "localhost";
$db_username = "root"; // database username
$db_pass = "erevveiogsdb"; // database password
$db_name = "samdr"; // database name

// Run the connection here
@mysql_connect("$db_host","$db_username","$db_pass") or die
('Could not connect port 4');
@mysql_select_db("$db_name") or die ("500 Internal Error, no
database could be located.");
```

$_SESSION

For each time I require to see if the user has logged in I must call a session. When the user first logs in a session is created. I have called this session, "tee". (`$_SESSION['tee'];`)

The reason behind calling this `$_SESSION['tee'];` & not "id" is because I originally ran into a problem where the session was not saving. I put this down to being on a shared server & it may have been over written by another site running on another directory. Having not being able to solve this problem I decided to call the logged in session "tee". It serves no meaning, however I knew that it would not have been used before.

This session can be changed in future development in order for better developer maintenance. This could potentially affect many different features, which is why I did not want to disturb this vital function late on in the project.

Driver Check

For each feature that can only be accessed by accepted drivers the following code must run. It first checks the database with the saved user id session, (`$_SESSION['tee'];`)

When the driver joins a new row is created in the "driver" table. This contains a user id, (userId), which refers back to the user table.

The following code is run every time we need to check if the active user is a driver.

```
$driverCheck = mysql_query("SELECT userId FROM driver WHERE
userId='$userId' LIMIT 1");
        $countDriver = mysql_num_rows($driverCheck);

        if($countDriver < 1){
            echo'<div id="eMessage">You need to apply to be a
driver. Apply <a href="./driver">here</a></div>';
        }
```

IF Statements

The majority of my web app will follow if statements. It will reason through various conditions narrowing each subsection down. The result in this will be the data being either entered into the database or data will be pulled from the database.

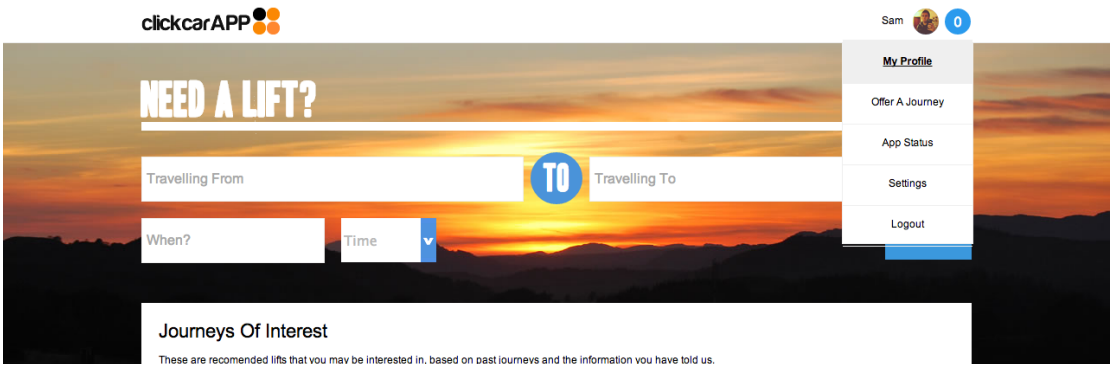
```
If($variable == $condition){
      //Do something
}
else
{
      //Do something else
}
```

PHP Include

I have used the PHP include function in order to include frequently used pages. The website header is used on every page and so I will use the following code in order to display the header on each page.

```
<header>
    <?php
        include_once"./addon/header.php";
    ?>
</header>
```

Issues –
I did have some issues with the header. The main issue was developing a header which displays logo's & includes links to other pages. As the header is used all over the website it is used in different directories. Sometimes when a header was in a lower directory it would change the URL of the links.



Page Structure

Each page structure will be formed in the same way. A typical page will include a header, footer, jQuery library, main style sheet, local style sheet, as well as a consistent page layout. By designing every page like this it makes the process of developing the website far simpler.

I have included a typical code snippet for a typical page, they all follow a similar layout that has been designed based on the key functions that each page needs to perform.

```
<?php
session_start();
include_once'./algorithm/connection.php';

if(!$_SESSION['tee']){
     header( 'Location: ./login' ) ;
}

?>
<!DOCTYPE html>
<html>
<head>
<title>ClickCar App / Offer A
Journey</title>

<link rel="stylesheet" type="text/css"
href="./addon/header.css">
<link rel="stylesheet" type="text/css"
href="./styles/journey.css">
<link rel="stylesheet" type="text/css"
href="./styles/home.css">
<script
src="//ajax.googleapis.com/ajax/libs/jquer
y/1.11.0/jquery.min.js"></script>

</head>
<header>
    <?php
        include_once"./addon/header.php";
    ?>
</header>
<body>
    <div id="banner">
        <div id="eMessageHolder"></div>
        <div id="bannerInner">
            <h2>Offer A Journey /
Price</h2>
        </div>
        <div id="pageHolder">

           PAGE CONTENT GOES HERE

        </div>
<?php include_once"./addon/footer.php"; ?>
</body>
<script type="text/javascript">
        $('#priceForm').submit(function()
{
            var priceTweak = $(
"#priceTweak" ).val();
            var luggage = $( "#luggage"
).val();
            var seats = $( "#seats"
).val();


$.post("algorithm/journey.php?r=2",
{priceTweak: priceTweak, luggage: luggage,
seats: seats})
                    .done(function(data) {

$('#eMessageHolder').empty().append(data)
if($("#eMessageHolder:contains('...')").le
ngth)
```

Check that the user is logged in

HTML document header

Include CSS style sheets

jQuery library

Website header

Website banner

Page Content

Include Footer

JavaScript form interaction with PHP backend.

Functions

PostCode Checking

The follow bit of code is used to in order to check that a post code has been entered in the correct format. If it hasn't then it will return a false statement otherwise it will be true. It simply checks that numbers and letter are in the correct order and format.

```
$postCode = strtoupper(str_replace(' ','',$postCode));
        if(preg_match("/^[A-Z]{1,2}[0-9]{2,3}[A-
Z]{2}$/",$postCode) || preg_match("/^[A-Z]{1,2}[0-9]{1}[A-
Z]{1}[0-9]{1}[A-Z]{2}$/",$postCode) || preg_match("/^GIR0[A-
Z]{2}$/",$postCode))
        {
            $postCodeGood = "true";
        }
        else
        {
           $postCodeGood = "false";
        }
```

Sending An Email

Because I want emails to be reliable I have used a third party script for sending emails. In the past I have had problems where emails have been ending up in the junk folder & not where I want them to be. Therefore I have decide that I would research and use a PHP class which sent all the correct added information to avoid my email from ending up in visitors trash. The reason why I wanted to avoid this is because some emails have activation codes in them. Without them the user would not be able to activate their accounts, thus locking them out.

On the next page is an example of an email script I used in order to send emails to users.

This particular email script was used in order to send emails for new users to activate their accounts.

```
// Send email for activation
 // script is open source from
https://github.com/PHPMailer/PHPMailer
        require 'email/PHPMailerAutoload.php';
        $mail = new PHPMailer;

        $mail->isSMTP();                                      //
Set mailer to use SMTP
        /*$mail->Host = 'smtp1.example.com;smtp2.example.com';  //
Specify main and backup server
        $mail->SMTPAuth = true;                               //
Enable SMTP authentication
        $mail->Username = 'jswan';                            //
SMTP username
        $mail->Password = 'secret';                           //
SMTP password
        $mail->SMTPSecure = 'tls';                            //
Enable encryption, 'ssl' also accepted*/

        $mail->From = 'clickcar@samdr.co.uk';
        $mail->FromName = 'Click Car';
        $mail->addAddress('' . $email . '', '' . $username . '');
// Add a recipient

        $mail->WordWrap = 50;                                 //
Set word wrap to 50 characters
        $mail->isHTML(true);                                  //
Set email format to HTML

        $mail->Subject = 'Activate Your Account';
        $mail->Body    = '
        <h4>Hi ' . $username . '<h4><br>
        <span>First of all, welcome to ClickCar. You may be still
in the middle of creating your account. We have sent you this email
because we need you to activate your account. <br>
        <h4>Your Code</h4><br>
        <span><b>' . $randomEmailCode . '</b></span>
        <br>
        Just copy the code into the activation page.<br><br>
        Glad to see you on board,<br>
        Kind Regards,<br>
        ClickCar</span>
        ';
        $mail->AltBody = 'Hi ' . $username . ', this is just a
quick email to let you know you need to activate. Copy and paste
the url or just click on it. <a
href="http://samdr.co.uk/clickcar/active.php?u=' . $UserId .
'&code=' . $randomEmailCode . '">
        http://samdr.co.uk/clickcar/active.php?u=' . $UserId .
'&code=' . $randomEmailCode . '</a>. Glad to see you on board, Kind
Regards, ClickCar.';

        if(!$mail->send()) {
           echo 'Message could not be sent.';
           echo 'Mailer Error: ' . $mail->ErrorInfo;
           exit;
        }
        ////////////////////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
```

4.4 – Dependency Issues

Some features depend on external services in order to work. While this currently creates a richer & more useful web application, it could cause problems in the future. The service will not be as self-sufficient when it relies on service like Google.

The greatest dependency is on the use of Google Maps. My API key is restriction to 2000 requests per day. After that the service will cost money. There is also the unlikely chance that the service will become obsolete. This could be down to Google needing to charge for the user of their service, or the links & libraries used becoming expired. One must appreciate like all programs they will need to be updated throughout their lifetime.

4.4.1 XML Dependencies

My mobile application also relies on an xml file where an argument is passed to it via the URL. [10]

```
$ip =  $_SERVER['REMOTE_ADDR'];
$journeyId =  $_GET['id'];

$xml=simplexml_load_file("http://www.geoplugin.net/xml.gp?ip
=$ip");

    $city =  $xml->geoplugin_city;
    $region =  $xml->geoplugin_region;
```

The code above links to the website geoplugin.net. This is the only free service I could find that would provide geological data with the use of the users IP address. Being free it may not be 100% reliable, & therefor there are many room for improvements. The service seems reliable after reading comments about the service.

```
$city =  $xml->geoplugin_city;
```

The above line is used to pull the data from inside the tag “geoplugin_city”

When passing a users IP address through the service it produces the following xml document.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<geoPlugin>
<geoplugin_request>81.106.0.215</geoplugin_request>
<geoplugin_status>200</geoplugin_status>
<geoplugin_credit>
Some of the returned data includes GeoLite data created by
MaxMind, available from <a
href=\'http://www.maxmind.com\'>http://www.maxmind.com</a>.
</geoplugin_credit>
<geoplugin_city>Cardiff</geoplugin_city>
<geoplugin_region>Cardiff</geoplugin_region>
<geoplugin_areaCode>0</geoplugin_areaCode>
<geoplugin_dmaCode>0</geoplugin_dmaCode>
<geoplugin_countryCode>GB</geoplugin_countryCode>
<geoplugin_countryName>United Kingdom</geoplugin_countryName>
<geoplugin_continentCode>EU</geoplugin_continentCode>
<geoplugin_latitude>51.483601</geoplugin_latitude>
<geoplugin_longitude>-3.1477</geoplugin_longitude>
<geoplugin_regionCode>X5</geoplugin_regionCode>
<geoplugin_regionName>Cardiff</geoplugin_regionName>
<geoplugin_currencyCode>GBP</geoplugin_currencyCode>
<geoplugin_currencySymbol>&#163;</geoplugin_currencySymbol>
<geoplugin_currencySymbol_UTF8>£</geoplugin_currencySymbol_UTF8>
<geoplugin_currencyConverter>0.5923</geoplugin_currencyConverter>
</geoPlugin>
```

Taken from geoplugin.net using my IP address.

4.4.2 Date Management

It is vital that each journey is not displayed in the search results after the planned journey date has passed. It is therefore important that I perform a check when dealing with any journey management. Journey management consists of search results, enquiring for journeys, & generally viewing a journey.

I have put measures in place that should prevent this from happening.

Dating In Search Results

When a user searches for a journey using the parameters I want it so that it doesn't display old search results. This would not be helpful for the user. The only positive outcome is that the user could message the driver if they plan on making any similar journeys in the near future.

I have the following code that should stop any search results from displaying "out of date" journeys.
Below is the code that is used in order to filter out of date search results out of the system.

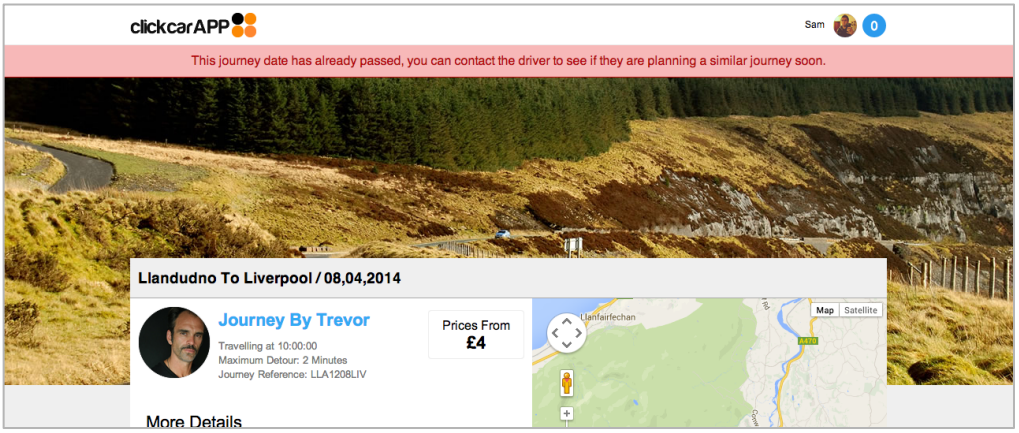
```
// check dates
            $dateTimestamp = strtotime('' . $jDate . '');
            if($dateTimestamp < strtotime('now')) {
               // show nothing for this search return
                echo'date gone';
            }
            else
            {

            $jCount ++;
// display journey onto html page
            echo '
              <div class="journeyDisplay" style="' .
$tripAlert . '">
                <a href="trip?trip=' . $JId . '">
                    <h4>' . $start . ' To ' . $end . '</h4>
                    <h6>@' . $jTime . ' On ' . $jDate .
'</h6>
                    <div class="journeyPrice">
                        <h5>&pound;' . $jPrice . '</h5>
                    </div>
                </a>
                </div>
            ';
            }
```

It runs by taking todays date & comparing it to the planned journey date. Both dates must be put into the same format in order to compare them. This is done with the PHP function:

`strtotime`

Dating On The Journey Page

I am not going to stop displaying a particular journey when it expires. This is because I feel it would be useful to keep backdated journeys on show if a passenger wishes to query a driver if the journey will be happening again. Although it will not be feasible to enquire about this journey they can still contact the driver through their profile.



The red banner across the top of the journey page demonstrates that the journey has expired. It cannot have any more enquiries.

```
$tripAlert='';
    $dateTimestamp = strtotime('' . $jDate . '');
    if($dateTimestamp < strtotime('now')) {
    $tripAlert='<div id="alreadyBeen">
   <span>This journey date has already passed, you can
contact the driver to see if they are planning a similar
journey soon.</span>
      </div>
           ';
           }
```

Dating On The Enquiry Button

The enquiry button will not display if any of the following conditions are true.

- The date of the journey has already passed.
- The user is not logged in
- The user has already enquired about the journey
- The user is on their own journey page.

** “The user” refers to the active using session on the website.

4.4.3 Google Maps Dependencies

The Google Maps API has a direct link to my application. When developing the software I found that the parameters were very sensitive. If Google were to

change their parameters required to return a route using the Maps API it could change the way my web pages behave. Especially those directly related to displaying information about a particular journey.

My database structure is not large or sophisticated enough to deal with spatial database mapping. Therefor it is necessary to use third party software such as Google Maps.

Documentation

There is plenty of documentation to help any future developers, or indeed myself, to maintain the mapping part of the system. The parameters are quite abstract and don't have to be specific. Google has a clever way of knowing what you want even if you only type in a vague search term. E.G "*Newport*". Judging by data collected on you it will choose the closest Newport to you.

Overcoming Issues

There were some issues with my search function. It had a tendency not to perform a journey search with some place names such as Barry. Barry is a name of a town, however it would pick out street names or shop names rather than the town. Instead I decided to use Google Maps suggestion box. What this does is suggest a place name when you start typing. It automatically chooses the best match, giving a full address.

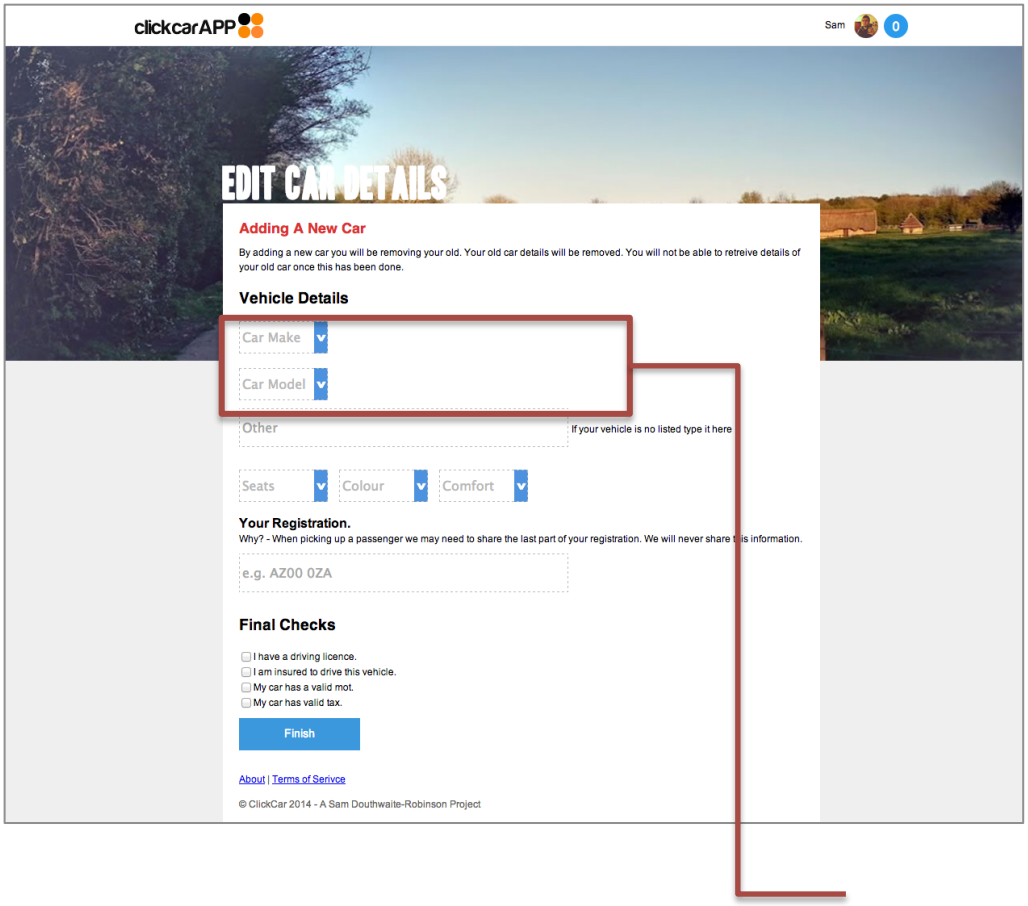
When processing this data the PHP page makes an array using each comma (,) as a parameter. It then uses the first array block in order to perform a satisfying search from my database. However when mapping on Google Maps it would use the full search term.

```
// construct start and end string
    $startArray = explode(",",$start);
    $endArray = explode(",",$end);

    $start = $startArray[0];
    $end = $endArray[0];

…..
…..
…..
$getPostCodeStart= mysql_query("SELECT postcode FROM
postcodes WHERE town='$start' LIMIT 1");
              $countGetPostCodeStart =
mysql_num_rows($getPostCodeStart);
                while($row =
mysql_fetch_array($getPostCodeStart))
                {
                    $startPostCode= $row["postcode"];
….
….
….
```

4.4.4 Car Database Dependencies

As mentioned in earlier parts the Car Database containing the make & model of most of the cars ever registered was created by a third party. Because I did not want to pay the charges of car database services I tried to find a free way of downloading a details of every car make & model ever registered. This table contains 72 car manufacturers & 1330 car makes. Even when I first downloaded this information I noticed some cars missing such as Vauxhall Corsa & Astra. Because I cannot be 100% guaranteed that this database contains every possible car I have included a third field for the driver application page. This field is stated as “Other”. Which allows the user to freely enter car information.

This should cover all possibilities.
Bellow is a screenshot of the page, new drivers will have to fill out before they can create journeys.



These dropdown menus' will load car details from the database.

```
<select id="carMake" name="carMake" class="signUpInputDropDownLong">

                <option value="" disabled="disabled"
selected="selected">Car Make</option>
            <!-- car make -->
            <?php
                // select car models from database
                $getModel = mysql_query("SELECT id, title FROM make");
                while($row = mysql_fetch_array($getModel)){
                $code = $row["id"];
                $title = $row["title"];
                   echo'<option value="' . $code . '">' . $title .
'</option>';
                }
            ?>
</select>
```

The above code selects all the car makes from the database and prints them out in a HTML dropdown menu, by looping through the list. It assigns the **id** to each one, and performs an AJAX function to return the model for each option.

The code below is the JavaScript / JQuery which sends the id of the make chosen through to a PHP page. This then loads up all the models under that make and displays it in a new dropdown menu.

```
$('#carMake').on('change', function() {
       $.post("testCar2.php", { html: this.value})
     .done(function(data) {

$('#modelResult').empty().append(data)
                              });
                  return false;
```

4.5 Validation

I have plenty of validation in my software that avoids incorrect or inconsistent data from being input into the system. Some of these are simple measures such as drop down menu's rather than allowing the user to freely input information. This can be particularly useful when I need to work with the data that has been input.

An example of this is when a user enters their data of birth during the initial registration. While this is seen as something simple I wanted to keep the format of every date of birth entered the same. This is because I do not want to display personal details like data of birth on the website. Instead I would rather display an age.

Another example is the "gender" category. I don't want people to be entering various types of gender. In the future I intend to match people depending on gender preferences & so allowing a free input outside of male or female, although not always seen as politically incorrect, may cause problems to arise in my system.

Dropdown menus & radio buttons are a great way of collecting consistent data patterns, & can be useful for research & maintenance on my system.

5.0 – Testing

This part of the report will focus on the testing stages taken when making sure my software is fit for purpose. The testing stage will focus on two parts. One will use the test cases to make sure that the software works correctly, the other will use the requirements to help me with the evaluation section as to how many of the requirements my software meets.

First of all I am going declare all of my test cases. Usually these would appear in the design section however I feel they are easily accessible if they are here.

I shall be visiting each page with major important functionally while designing my test cases.

5.1 Black Box Testing

As an addition to the testing I did while developing the software this should be a helpful analysis of everything as a finished product. I have already tried the software out on friends & the website appears to run fine.

Black box testing took place when I did some unofficial tests. It was a great opportunity for me to discover how people reacted to the software & how they felt using it.

Results

Although nothing official was documented of these unofficial tests, I did have a successful outcome where both candidates managed to register without any help from me.

Both candidates found it easy to edit their profile & upload their own content such as a cover photo, profile photo & car photo. They particularly like the feedback given from the software to confirm when a specific action took place.

I checked if the candidates could add vehicles to their accounts, & if they could create new journeys. One interesting addition found was that even some of the smallest towns in the UK have been included in the Post Code database. This came as a shock to me as I did not expect that level of detail from a GitHub repo. Therefor I am very pleased with the result. Not only that, but the chosen locations had the correct post-codes attached & so they worked when locating nearby towns en route.

I was not able to test the app on both candidates as the current app only works on android devices. Both candidates were Apple users. This clearly is a limitation that can be fixed in the short term, given more time.

Overall I found these unofficial tests to be successful as they worked when the website link was given to them. The website proved to be self-explanatory & so I count this to be a success.

5.1 Test Candidates

I have invented some test candidates that will allow me to test my software. I will make 2 different taste candidates. One will have a driver's account, & the other will have a normal basic account. This should give enough scope to cover both sides of the program.

First Candidate

Username: Jamsie
Name: James Harris
Email: jamesHarrisClickar@gmail.com
Password: treetop5
Date of birth: 23/4/1990
Address: 6 Salisbury Road, Cardiff, CF245HU
Interests: Snooker, Reading, Movies
Regular Journeys: Cardiff to Swansea, Swansea to Cardiff, Merthyr Tufil to Cardiff, Cardiff to Merthyr Tudfil
Gender: Male
Bio: Accountant by day, Film buff by night. I like to socialize and meet new people
Driver: Yes
Drives a Nissan Juke
Car Registration: CU09 TYZ

Taken From
http://blogs.voices.com/voxdaily/2009/04/hidden_features_add_a_client_list_and_testimonials_to_your_profile.html

Second Candidate

Username: Lucy
Name: Lucy Jones
Email: sm20007@msn.com
Password: underhill32
Date of birth: 07/12/1992
Address: 78 Treharris Street, Cardiff, CF242HG,
Interests: Television, my dog, reading, going to the gym
Regular Journeys: Cardiff to Birmingham, Birmingham to Cardiff
Gender: Female
Bio: Full time student wishing to make new friends & have a quick way of getting home.
Driver: No

Taken From *:*
http://thatshortblondegirl.blogspot.co.uk/2011_08_01_archive.html

** The above photos were taken from random locations on the internet. They do not intend to be used when the software if released & are simply for demonstration purposes only. Their real names, or personal information are not used.

5.2 Test Table

The test cases are a checklist of tests that need to be carried out with specific information. The following table covers all the possible areas that need to be tested.

Title	Input	Expected Outcome	Actual Outcome	Pass/ Fail
Check registration page validation.	Input test candidate 1 into the field. Leave some fields blank	The page will display an error message, saying all fields need to filled out	An error message displays.	Pass
Registration Page 2, validation.	Try to continue without entering any inputs	The page will display an error message	The page displayed an error message	Pass
Registration Page 3, validation.	Try to continue without entering any inputs	The page will display an error message	The page displayed an error message	Pass
Check all data has been entered correctly	All data collected through the registration process.	Correct data will be in the “user” table.	The data has been correctly copied into the database.	Pass
Activate Account	Email is sent to the relevant email address	An email with an activation code will be sent to the email address.	An email with the code was sent.	Pass
Failed Login	Check wrong username & password	The user will see a message saying they couldn’t be logged in.	The user wasn’t logged in.	Pass
Login	Check correct Username &	The user is logged in.	The user was logged	Pass

	password		in & taken to the homepage	
Login	Check Email address & password	The email address is used instead of a username to login.	The matching email address logged the user in.	Pass
Become A Driver	Enter missing details in.	The user will be shown a missing details message.	The user is shown a missing details message	Pass
Become A Driver	User selects car make	User selects car make & sees car model.	The matching car models in shown in a dropdown menu.	Pass
Become A Driver	Details are correctly copied into the database.	All the user's details are copied into the database.	The drivers details were correctly copied into the database.	Pass
Become A Driver	Send email to driver.	An email is sent to the driver confirming it with them.	An email was successfully sent to the driver.	Pass
Offer A Journey	Offer a Journey without being a driver.	Offering a journey without being a driver shows a driver application page.	The non-driver account is directed to apply for a drivers account.	Pass
Connect App	After the user has become a driver they must connect the app. It will show a download link	The user is shown a link to download the app to their phone.	The app link was shown.	Pass
Offer A Journey As a Driver	Offer a journey after becoming a driver.	The create journey page should show	The create journey page is displayed.	Pass

Create A Journey	Test if the validation for the page works.	Do not enter all the relevant information. An error message will show.	An error message is displayed to the user.	Pass
Calculate journey distance	Test if the Google Maps journey calculation tool works.	The Google Maps Journey Calculation should show a distance between the start point & end point.	A distance of 152 miles is shown between Cardiff & London.	Pass
Calculate Price	Calculate the price of journey.	An automatic price generation algorithm will calculate the price between the 2 locations.	A Price of £30 for the journey has been calculated. That appears about right for a 152 mile journey.	Pass
Confirm Journey	Click Finish to confirm journey	Confirm journey will update the database value from 0 to 1.	The database value is changed & a confirmation message is shown.	Pass
Display confirmation Page	Journey confirmation page is shown. It shows a breakdown of all the details shared.	Once the finished button is clicked it shows a confirmation message.	A confirmation message is shown.	Pass
Settings	Change user profile image	Profile photo uploaded, message displayed	Profile photo is changed	Pass
Settings	Change user cover photo	Cover photo uploaded. Message displayed	Cover photo is changed	Pass

Settings	Change car photo	Car photo uploaded, Message displayed	Car photo is changed	Pass
Settings	User uploads photo that is too large. (over 200mb)	User uploads a photo over 200mb. An error message is shown and photo is not uploaded.	User photo is not uploaded. Error message is shown.	Pass
Settings	Upload file that is not a photo.	User tries to upload a file that is not a photo. An error message should be shown.	An error message is shown.	Pass
Edit Car Details	Take user to car page.	Add vehicle page is shown.	The user is shown the add vehicle page.	Pass
	Fill out form as normal.	Remove old car and insert new car	Old car is removed & new car is inserted. Confirmation email is sent.	Pass
Edit User Details	User changes details	Details are updated in the database.	Details are successfully updated in the database.	Pass
Edit User Details	Leave field empty.	Error message shown appear.	Error message appears. “You must fill in all fields”	Pass
Settings, Privacy	Display privacy page.	Display all details stored by ClickCar & state what is shared.	A list of details are shown as well as what is shared.	Pass
Cancel open Journeys	Open Journeys are displayed. The user has the right to cancel up to 48 hours in advance	The open journeys are displayed. Once the Cancel button is clicked	The journey is still saved in the database, however	Pass

		the journey is canceled. Any passengers are notified by email.	marked as inactive. The attending passenger is notified by email.	
Alert & Message Centre	Messages are displayed.	A list of subjects is displayed to the user.	A list of messages is pulled from the database and displayed to the user.	Pass
Reply to a Message	Message is displayed & reader has the option to add to the conversation.	User clicks on a subject, reads the message and can reply by typing a message and clicking "Reply". Message is displayed in conversation.	Message is successfully added & is stored in the database, & displayed in the conversation	Pass
Logout	User logs out.	User clicks header icon, and selects log out. User is logged out & shown the homepage.	Dropdown header menu is shown, user clicks logout, & is logged out. Session Destroyed!	Pass
Display own profile	Display users own profile.	User clicks on profile link & their own profile is shown.	Users own profile is shown.	Pass
Display friends profile.	Display friends profile.	User passes their friends **username** through the URL. Their profile is displayed.	If profile is successfully found, that particular profile is displayed.	Pass
Display 404 on Profile	Display a 404 page on profile with an invalid username.	A username is not found & so it displays a 404 page. Example uername = "Ted123"	404 page is displayed to the user as Ted123's profile was not found.	Pass

Display Reviews on Profile	Reviews from various users are displayed on profile.	The system combines all reviews and generates colour-coded charts in order to show visitors the ratings of each user.	The user ratings are displayed. An average is given based on various reviews.	Pass
Display Comments	Display comments from past journey reviews.	Collect comments from the last few reviews left about this user. State whether the user was a driver or passenger. Link to their profile.	Comments are pulled from the review database, stating who left them and on which journey they took.	Pass
Message User	Message User from profile.	Visitor of profile messages the profile owner by clicking on message. A dialog appears & the user can message the profile owner. The message is then received in the notification center.	The user managed to message the profile owner successfully, A confirmation message was left. The message was then readable by the recipient in the notification center.	Pass
Search Journey	User searches for journey.	User enters journey from Cardiff to Manchester. If journey is found it should display.	Journey is displayed with price per passenger and option to see more details.	Pass
Show journeys you may be interest in.	Show a list of journeys the user may be interested in based on past enquiries & interest form.	A list of journeys that may interest the user will be displayed based on history & interested journeys. These journeys must be in date.	No journeys were displayed as they were any in the system. On a more active older account, some journeys are shown to the user.	Pass
Direct to trip page	User clicks trip and is directed to relevant page.	When the user clicks the trip want to see more of they are directed to that page. If the journey is old an out of date message is displayed.	An out of date message is displayed on old journeys. Current journeys are displayed with an “enquire” button & more details on the journey.	Pass
Enquire about trip	User enquires about a trip.	The user enquires about a trip & a enquiry box displays	The driver receives the request, and is able to accept the	Pass

		allowing the user to request seat. The driver can then accept the seat request.	journey request if they wish to.	
Login With App	User logs in via mobile app with incorrect details	User logs in, receives an error message if no connection is made.	Username or password not correct message is displayed.	Pass
	User logs into app successfully	User logs in with correct login details & is shown a list of open journeys waiting to be made.	User is shown open journeys & can click on them to start them.	Pass
Begin Journey	User begins journey after clicking on it.	User starts journey and location is detected. Once journey is complete user clicks “end” & email receipt is sent to passengers.	The user completed the journey & an email is sent to the passenger.	Pass

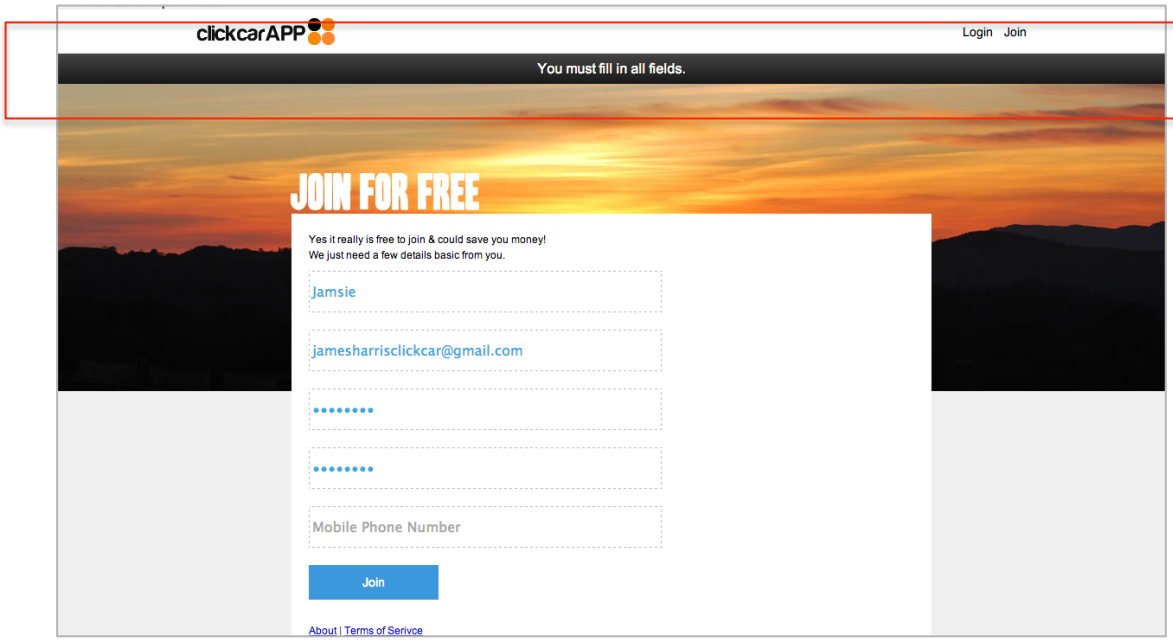
5.3 White Box Testing

White box testing will involve me enter the mock-user’s information into the system. It will allow me to prove that my system works correctly & possibly pick up on any bugs that are found. By using made up candidates it allows me to check whether the information stored on system matches what I intend to enter, & that no data is changed without me knowing.

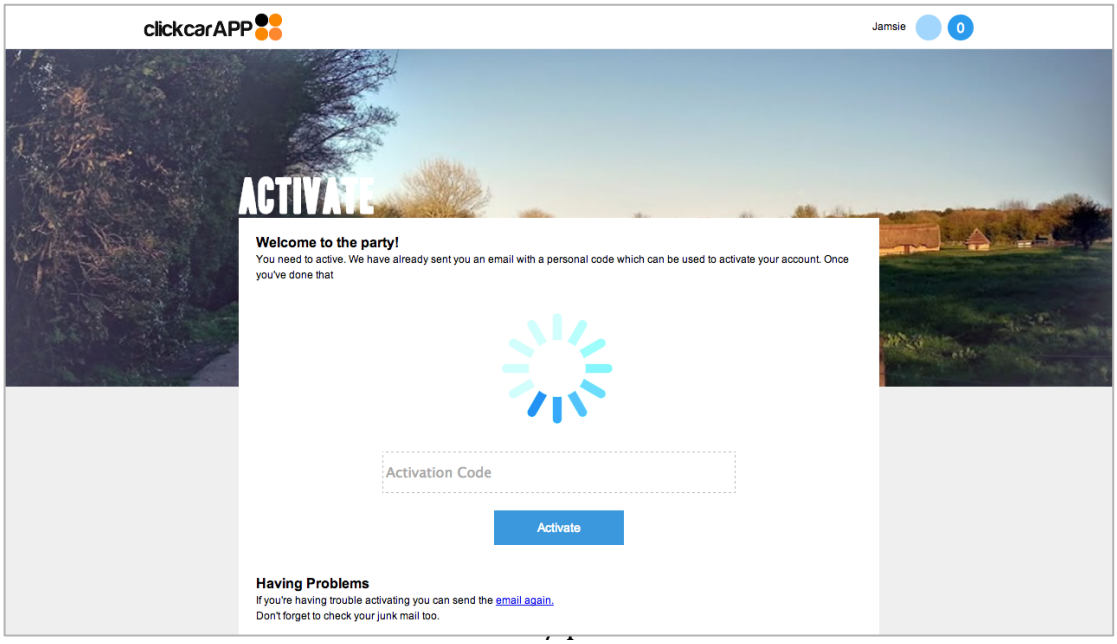
The following section will show some screenshot evidence, of the results of some of the testing.

5.2.1 Registration

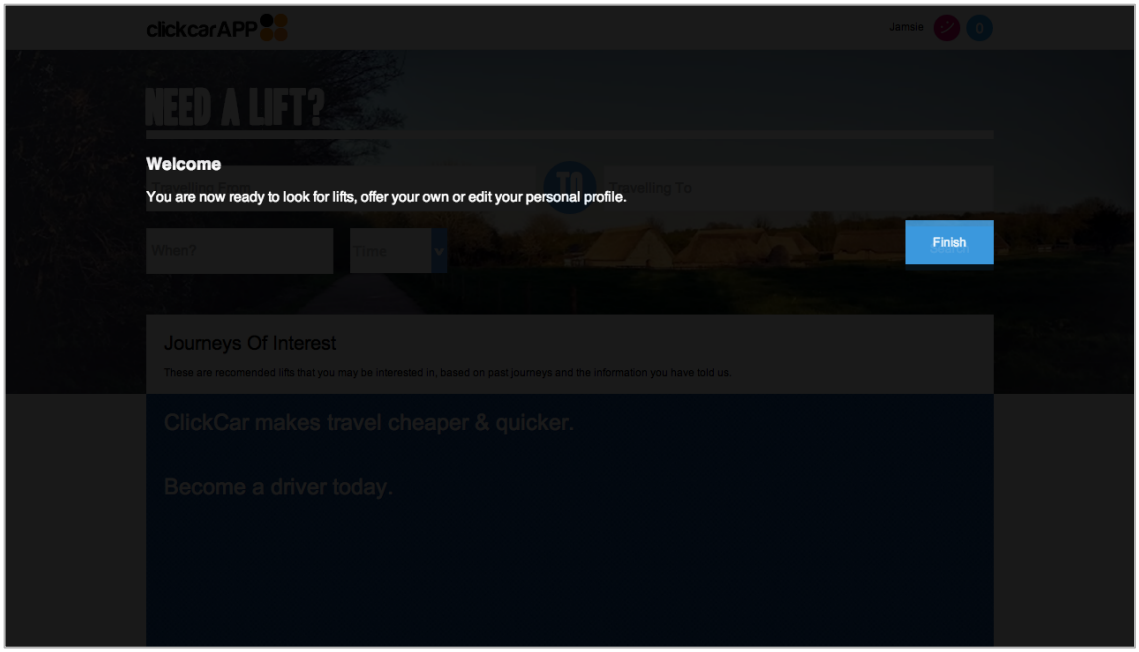
The registration page is used for when new member wish to join. They are required to enter some basic details that will help them log into their account, as well as some personal details so that they can identified by other members.



When the user hasn't entered all the required details error messages are shown.

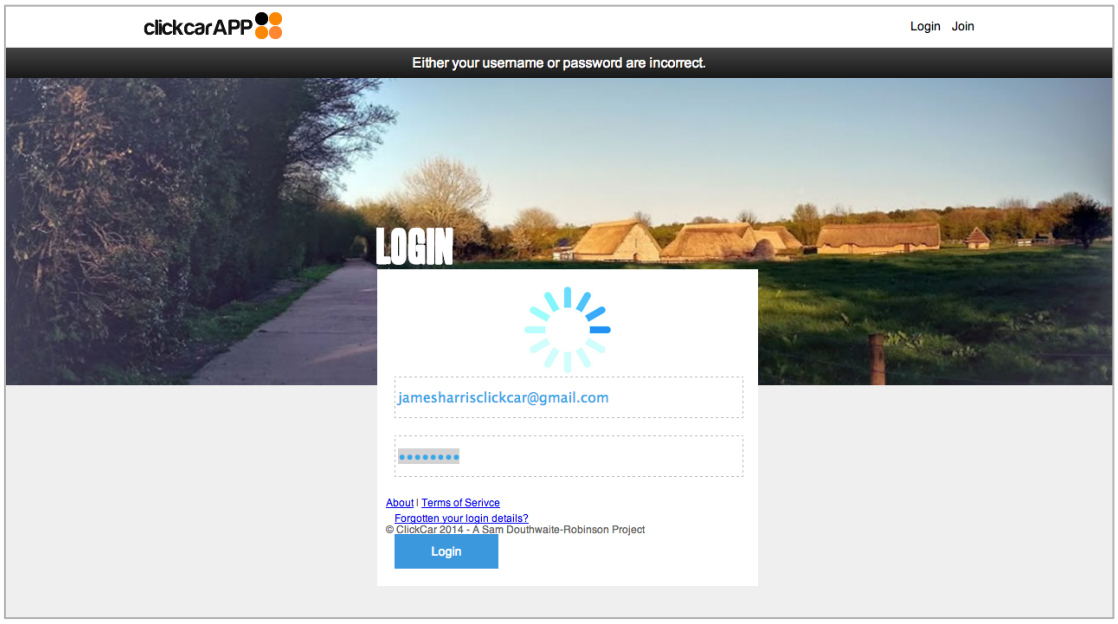


The activation page works by requesting a code that has been randomly generated and sent to the user. This code can then be entered and it will redirect the user to the welcome page, as seen below.



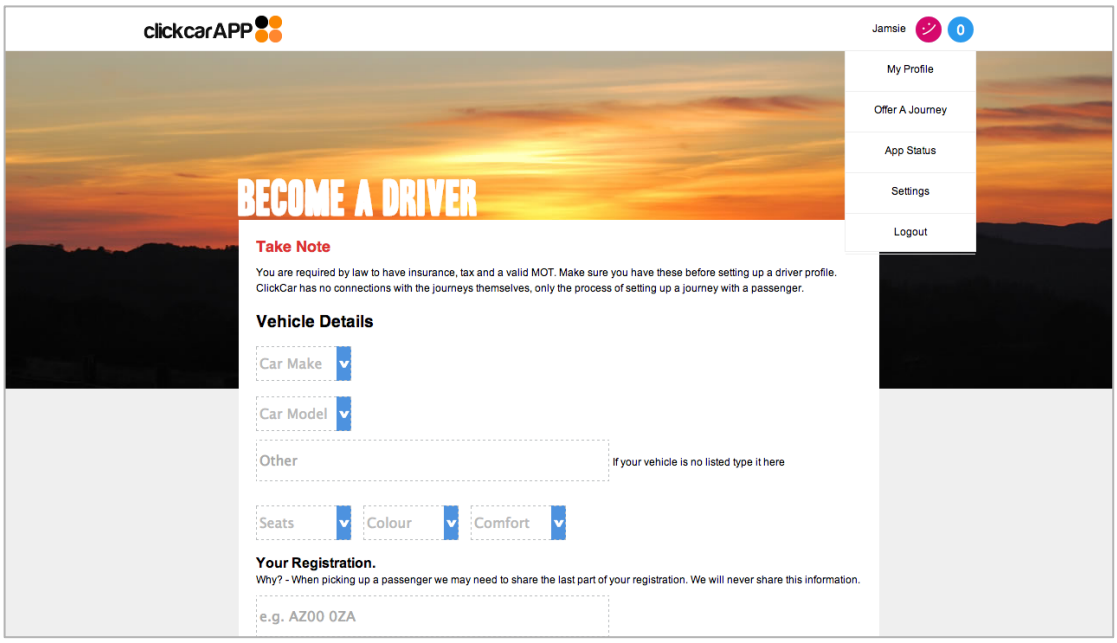
5.2.2 Login

The login page works by the user simply entering their login details. If the details are incorrect an error message displays. It is also possible for the user to log in with their email address instead of their username.



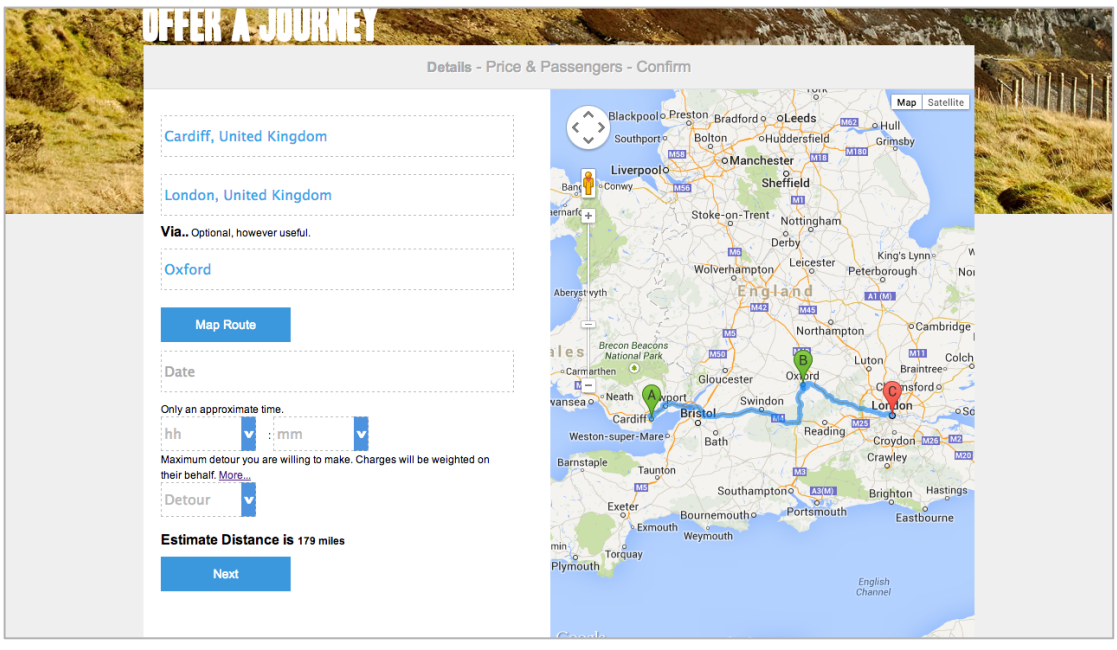
5.2.3 Become A Driver

Before a user can create any journeys they must first become a driver. This is done by simply adding in your vehicle details. One these details are entered the user can then create some journeys. This page also has validation to stop the swerve the user away from entering bogus information.

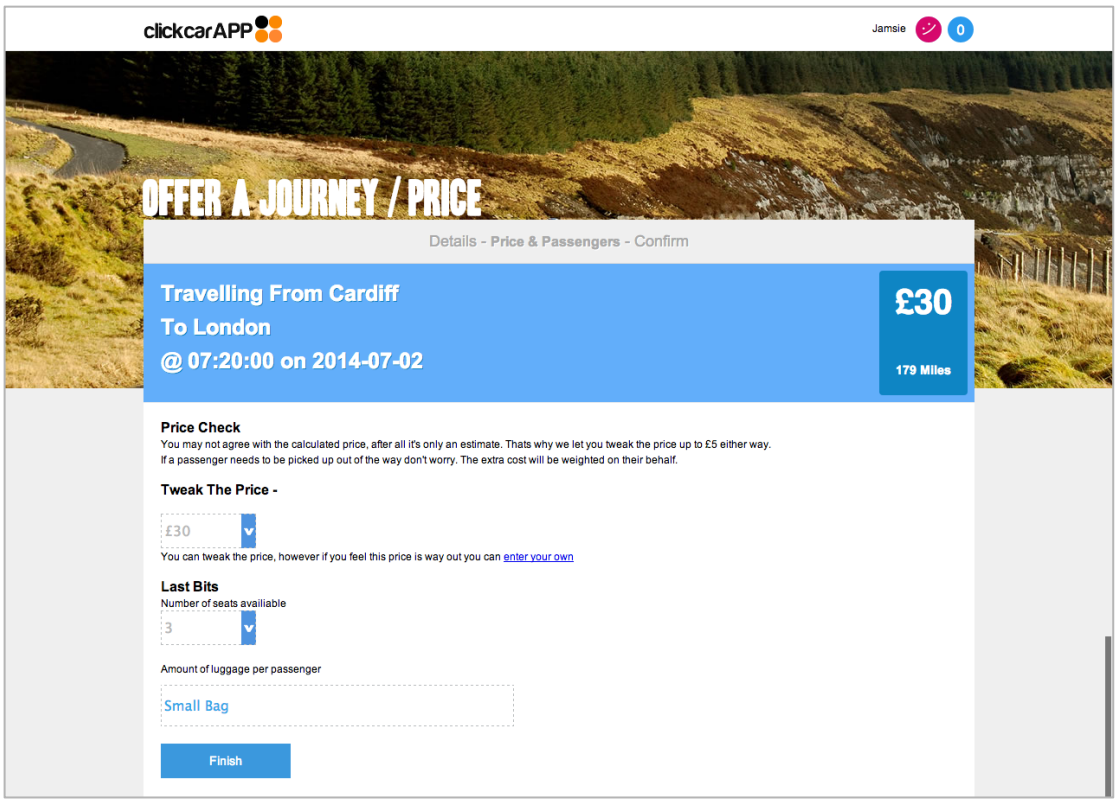


5.2.4 Create a Journey

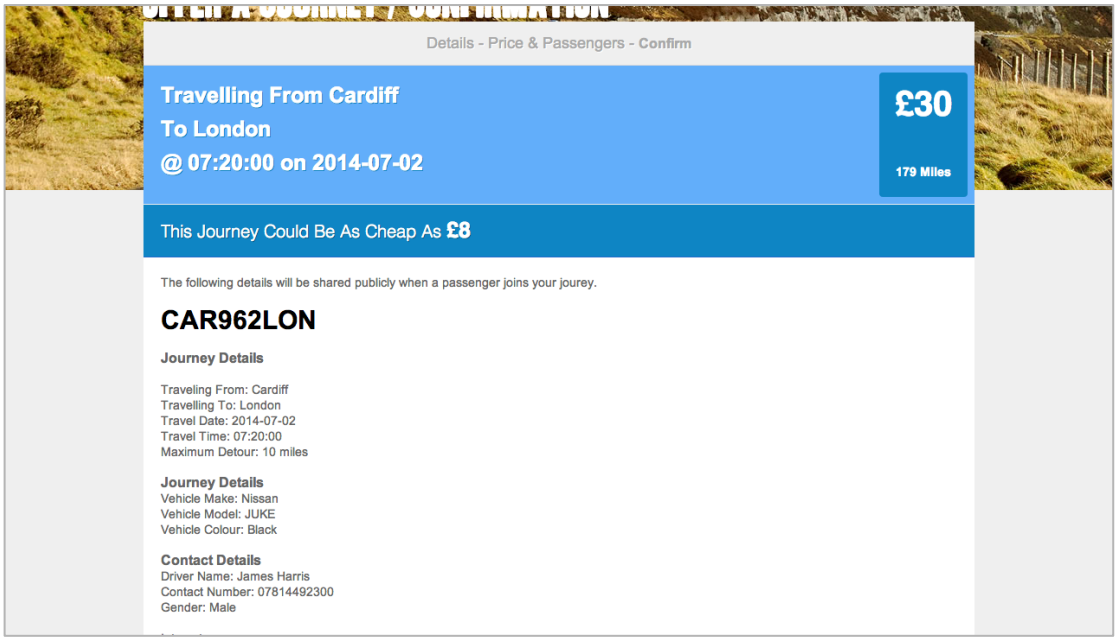
The journey creation is done in 3 steps. The first requires you to plan your journey. The second step requires you to tweak the price & the number of seats you are willing to give up. The third is a confirmation page explaining all the information that will be shared.



Step 2


clickcarAPP
Jamsie
0
OFFER A JOURNEY / PRICE
Details - Price & Passengers - Confirm
Travelling From Cardiff
To London
@ 07:20:00 on 2014-07-02
£30
179 Miles
Price Check
You may not agree with the calculated price, after all it's only an estimate. Thats why we let you tweak the price up to £5 either way.
If a passenger needs to be picked up out of the way don't worry. The extra cost will be weighted on their behalf.
Tweak The Price -
£30
You can tweak the price, however if you feel this price is way out you can enter your own
Last Bits
Number of seats availiable
3
Amount of luggage per passenger
Small Bag
Finish

Step 3


Details - Price & Passengers - Confirm
Travelling From Cardiff
To London
@ 07:20:00 on 2014-07-02
£30
179 Miles
This Journey Could Be As Cheap As £8
The following details will be shared publicly when a passenger joins your jourey.
CAR962LON
Journey Details
Traveling From: Cardiff
Travelling To: London
Travel Date: 2014-07-02
Travel Time: 07:20:00
Maximum Detour: 10 miles
Journey Details
Vehicle Make: Nissan
Vehicle Model: JUKE
Vehicle Colour: Black
Contact Details
Driver Name: James Harris
Contact Number: 07814492300
Gender: Male

6.0 – Future Work

As this project has gradually developed I have come up with many additional ideas. I did have time to implement some of these new ideas, however not all of them were achievable in the time frame given. Perhaps if the module was ran over 8 months rather than 4 it would have been possible to do much more.

I did however try to stick to my timetable as strictly as possible to give me enough time to complete the project. I also wanted enough time to test thoroughly test the application, and allow for any errors, & additional researched required.

6.1 – Additional Added Features

These features are features that have been added in, even though they were not part of my original requirements. These include elements such as making the profiles more like social network profiles with cover photo's, car photo's & creating an all round more user run platform.

Cover Photo

Cover photo's were not an initial thought. They do not do much for the core functionality of the system; however I feel that they are a nice addition to allow people to edit the look & feel of their profile. Features like this have been successful with other social network such as Twitter & Facebook, & so why not ClickCar. The overall coding for this feature was not much extra work.

Coding this feature was simply a case of adding an upload feature for a cover photo. It requires some validation on the uploaded image such as making sure it's below a certain size. It was then just a case of copying the file into the unique user folder.

```
if ($_FILES['newCoverPhoto']['tmp_name'] != "") {
            $maxfilesize = 307200; // 307200 bytes equals 300kb
            if($_FILES['newCoverPhoto']['size'] > $maxfilesize ) {
                    echo ($_FILES['newCoverPhoto']['size']);
                        $errorMsg='<div class="errorMessage">Your image
was way too large.</div>';
            unlink($_FILES['newCoverPhoto']['tmp_name']); // remove photo
from temp folder

            } else if (!preg_match("/\.(gif|jpg|png)$/i",
$_FILES['newCoverPhoto']['name'] ) ) {
                $errorMsg='<div class="errorMessage">Please upload a PNG,
Jpeg or GIF image.</div>';
                        unlink($_FILES['newCoverPhoto']['tmp_name']);

            } else {
                        $newname = "cover.jpg"; //  new name of the image
                        $place_file = move_uploaded_file(
$_FILES['newCoverPhoto']['tmp_name'], "./u/$myId/".$newname); // personal
user file
                // redirect page
                 header( 'Location: ./settings?msg=imgDone' ) ;
            }
    }
```

6.2 – Future Additions

Future additions are additional features I would like to add in the future. Some of these include elements like people matching. If the service was popular some of these features could work very well.

Profile Matching

In a world where intelligent algorithms try to guess what we want before even we know exactly what we want I believe an algorithm could be developed to match passengers and drivers based on personality & interests. This was something I would have liked to integrate further into the current version, however unfortunately that was capped at only searching through journeys that the user has already show interest in.

Such an algorithm could look like the following.

```
Get interests from database,
Get gender from database.

// collect any journeys the user has been interested in from
the messages table.

Loop through all relevant journeys matching these results.
Get driver personal details.
Get driverInterests
Get driverGender
Get driver Age

// match these results to show more personalised results.
```

The above shorthand *pseudocode* looks at various elements about a user already stored in the database and attempts to make matches based on this. This could have been achievable in my time-scale however; in my case I feel that it may not be suitable as for testing purposes I would need to have a very large data-set to test with.

Full Mobile Application

Mobile Applications are become ever more popular. According to *Mobile Marketing* in 2013, 7 in 10 people own a smartphone. [10]. This fact is enough to think about a fully functional set of mobile application. It would have all the features that are used on the main desktop website currently.

Moving towards a mobile application is inevitably going to reach out to more users, and make the system more desirable.

MiddleMan Payments

Currently the system does not handle any payments between users. This is because research shows it would count as a form of taxi service. Being such a grey area, the service does not operate like a taxi. It simply helps people find lifts that they wouldn't be able to usually find.

A system could be put in place where the passenger pays a deposit. As long as both parties are satisfied that journey was carried out in the end then the payment would go through to the driver, & the passengers could be given a receipt. This could easily improve the reliability of the service & create a safer environment because both passengers & drivers would be guaranteed safe payment.

If the payment was to be handled by ClickCar it could make sure that no one is miss treat. Any Internet "trolls" would be deterred from enquiring about journeys they have no intention of using.

Taking these steps would require more research into how the current system in place works. No doubt the service would need to pass certain regulations. It would also need a secure payment system.

7.0 – Conclusion & Reflection

This project has been a challenge in many areas. I have had to do a lot of research into many new areas, applying the skills I already have with newfound knowledge.

7.1 Met the Requirements?

My end product has met most of the requirements. The only part that isn't as I imagined is the mobile application. As the deadline for this project came ever so closer I found that I had to prioritize, and so I had think of a quicker solution to developing the application side. In the future I would expect to develop both Android & iPhone apps that have full site functionally. This would simply make it easier and more appealing for new users to try the service out.

7.2 Advantages

ClickCar has many advantages & areas that could make it tower over it's current competitors. Other services have been around for a few years now & have not thought of mobile applications, or tracking the journey. I feel that I would be able to handle elements of the service such as payment, which is something other services don't currently do.

7.3 Disadvantages

The main disadvantages of my project are that there are still areas of it that need some development. For instance the user messaging system does not currently allow you to block or report users. It does not allow you to delete messages either. These are elements I would like to implement; however I did not feel that they were necessary at this time.

7.4 Reflection

What have I learnt?

While developing this project I have learnt many things. I original intended to stick to my weekly plan. However I found this to be difficult as there was so much work that needed to be doing. I spent the first 2 weeks working on the research. In hindsight I shouldn't have spent this long on research as the programming took up most of my time. The design was done mostly just before the implementation & often I had to adapt the designs as I changed my mind about how certain features should work.

With another opportunity I would plan everything one week early. This way I would be able to leave enough time for things like the report, & final brush ups on the project, which haven't plaid a role as large as I would have liked.

Perhaps I did spend a lot of time of the visual design of the project. This could have been avoided with the use of a CSS style sheet bundle such as Bootstrap; however it may not have had much of a unique feel to the website. In the past I have also fallen out with the idea of Bootstrap as I find it quite hard to break. Everything is so well packed in together it's very hard to modify the look of something like the way a button looks.

Using something like a weekly milestone planner could have been useful. I did have something similar in place, which was a white board with a checklist of features that needed to be completed every week. I did stick to this, however I didn't look into the future, say 3 weeks in advance to know when I should be completing major parts and hitting major milestones.

7.4 Conclusion

I feel that my project has meat most the requirements, & has turned out to be a very credible piece of online software. Mainly due it's professional look, the feel, and the consistency between each of the pages.

Although the project has been a challenge it has also been enjoyable learning about the new aspects. This is something that I would wish to improve & later release it to the general public.

8.0 – Appendices

If you would like to try out the website you can visit it on my server.

The link is:

http://www.samdr.co.uk/clickcar

The website will be live until the end of 2014 at least. After that it will be a case of running the source code manually.

Also included in the appendices are:

- A zip file copy of the project.

- A backup download of the MySQL database, also in a Zip folder.

Bibliography

I have referenced to the links of where I sourced some of my research information from. Some credits have been given during the report to particular online communities where their knowledge & open source work has been particularly useful to me.

[1] https://www.facebook.com/notes/micro-lost-mobile-tracking-system-lmts/how-does-the-gps-tracking-system-in-your-mobile-phone-work/170495272997776

[2] http://developer.android.com/guide/topics/location/strategies.html

[3] http://www.techrepublic.com/article/understanding-the-pros-and-cons-of-the-waterfall-model-of-software-development/#.

[4] http://www.buzzle.com/articles/waterfall-model-diagram.html

[5] https://developers.google.com/maps/web/

[6] http://blog.smartbear.com/open-source/5-reasons-its-time-to-ditch-mysql/

[7] http://brackets.io/

[8] http://www.appsgeyser.com/

[9] http://stackoverflow.com/questions/6327679/what-does-mysql-real-escape-string-really-do

[10] http://mobilemarketingmagazine.com/7-10-people-uk-now-own-smartphone/

[11] http://www.blablacar.com/trust-safety-insurance