**Extension 3: Data-User-Interaction Analysis and Design**

**Usability on a mobile device such as a smartphone:**

A short survey of a few friends and housemates revealed what we had guessed; our CarShare site would not be the most mobile user-friendly site in the world. It does, however, have potential. A desktop site design will simply not work on a mobile device. Just a few issues: The layout will become unstable and unpredictable, button sizes will be much too small and print too hard to read, drop down and selections menus near impossible to manoeuvre. This overall decreases user experience, and as much research has shown, decreased user experience leads to less conversions (in our case, bookings/signups), and less profit and reliability for our CarShare company.

Ideally, a new separate domain or app would need to be created in order to cater for mobile users, and many sites currently do. We would create ‘m.CarShare.com’! Creating a specific mobile website would allow us to implement some of the many mobile UX widgets and ideas that have been created, specifically to positively influence our user’s experience on our website. For example, scrolling menus on smartphones and tablets rather than drop down menus, page scrolling on smartphones and tablets rather than side-bar scrolling.



In order to apply these changes, most of it would be in the html and css format, even just applying a new domain under m.servername.com, as opposed to www.etc, which is where the redirected mobile user interface will be. Most of the code and information would remain exactly the same, the only key change is in the style of the web pages. For some the use of apps is a much better method of creating a mobile interface, which often requires the use of java for the main coding component. Still, the SQL queries would remain the same as on the default desktop interface.

**On finding an available car close to current position:**

In terms of finding an available car close to the user’s current position, the system would be reliant upon the GPS of the device in use. This is most applicable for a mobile user who has access to the phone’s GPS, however even desktop users or devices which rely upon a Wi-Fi connection are able to gather the user’s current position based off IP address. By gathering the longitude and latitude of both the current user and all available cars or car bays with available cars in it at the time of booking, a simple algorithm or connection to a google maps widget can discern the closest available car.

If this were to be done manually by our own system the most direct method of ascertaining the closest available car would be to use Pythagoras’ theorem to find the direct distance and scaling the difference in longitude and latitude to km. It may also be able to set an alternative position and find the nearest car available, without using GPS, so if you wished to book a car near a position you will be in, as opposed to currently are in, you may.

With regards to which sections of code this would change, most of the changes would occur in the html portion of code, purely changing the access to widgets, applications, or the device’s other functions. For the sake of mobile devices, it is likely to require a java section within the python to provide mobile use, or as an app, which accesses google widgets or other applications.

In terms of SQL code changes, it would require a function which would find all available cars possible within a region, then compare those carbay locations by latitude and longitude to the current GPS, by the function: sqrt((current latitude - car latitude)^2 + (current longitude - car longitude)^2), which can then be scaled by multiplication of some empirical value to convert to km. The corresponding carbay with the minimum distance would then be shown to the user and possible booking periods shown to the user. This comparison could be provided by an embedded function as the member\_busy & car\_busy functions have been implemented in the make\_booking() function of the database.py, but could also be done purely beside the python and SQL queries.

**An efficient way to extend a currently active booking:**

Currently, in order to extend a booking, the user must make a new separate booking, which is time consuming and a hassle for the user to receive two invoices and two separate bills.

One would imagine most users would want to extend a particular booking *during* that booking, so it could also be assumed most users would be extending a booking on a mobile device, which is where mobile usability is important again. For desktop and mobile booking extensions however, it’s important that the user go through as little clicks as possible, and is able to extend their booking as fast as possible. Research has shown that if a transaction takes too much time and too many clicks, the number of conversions (in our case, completed bookings) drop steeply. We would design a booking extender as follows:

Scenario a)

It is 1pm. Mikaela is currently on her booking, driving Doris the Yaris. She would like to make a spontaneous trip to visit her Aunt Mildred out West, but her car booking ends at 2pm, and in order to visit her Aunt she would need the car until 4pm. She goes to the car share website, logs in and...  
She is greeted by a message over her home screen: “You are currently driving Doris the Yaris!”. Below the message, she is given two options: 1) Extend my current booking, or 2) Take me to home screen. She clicks “Extend my current booking”, and views a time-slot calendar which highlights times the Doris the Yaris is available today. She selects until ‘4pm’, then clicks the submit button “extend my booking”. Her card is automatically charged and she receives and updates invoice to her email. If Doris the Yaris isn’t available for a certain time, the time slot is greyed out and Mikaela is unable to click on that time slot, receiving a bubble “Doris is busy at that time”, if she tries to click on an already booked time slot. If Doris the Yaris is busy for the remainder of the day, the site suggests nearby available cars that Mikaela could book instead.

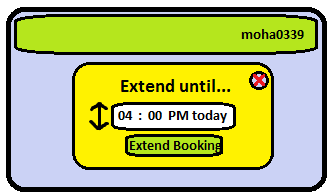
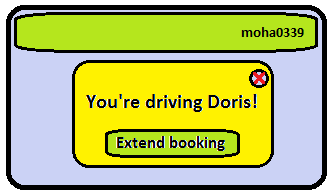
Because Mikaela is currently on her booking, and in a rush (as most people are), it was important that she was able to easily extend her booking without entering any new card details, and in as few clicks as possible.

Scenario b) Simon has booked a car for tomorrow from 9am-11am, but forgot he has a laser tag match at 12pm, and will need to extend his booking so he can drive to the laser tag arena! He logs onto the car share website, and is automatically taken to the home page, as normal. He then…

Goes to his ‘Bookings’ page, and clicks on his booking from tomorrow. On this page, there is an obvious “change/extend my booking times” button. He now goes through the same process as Mikaela.

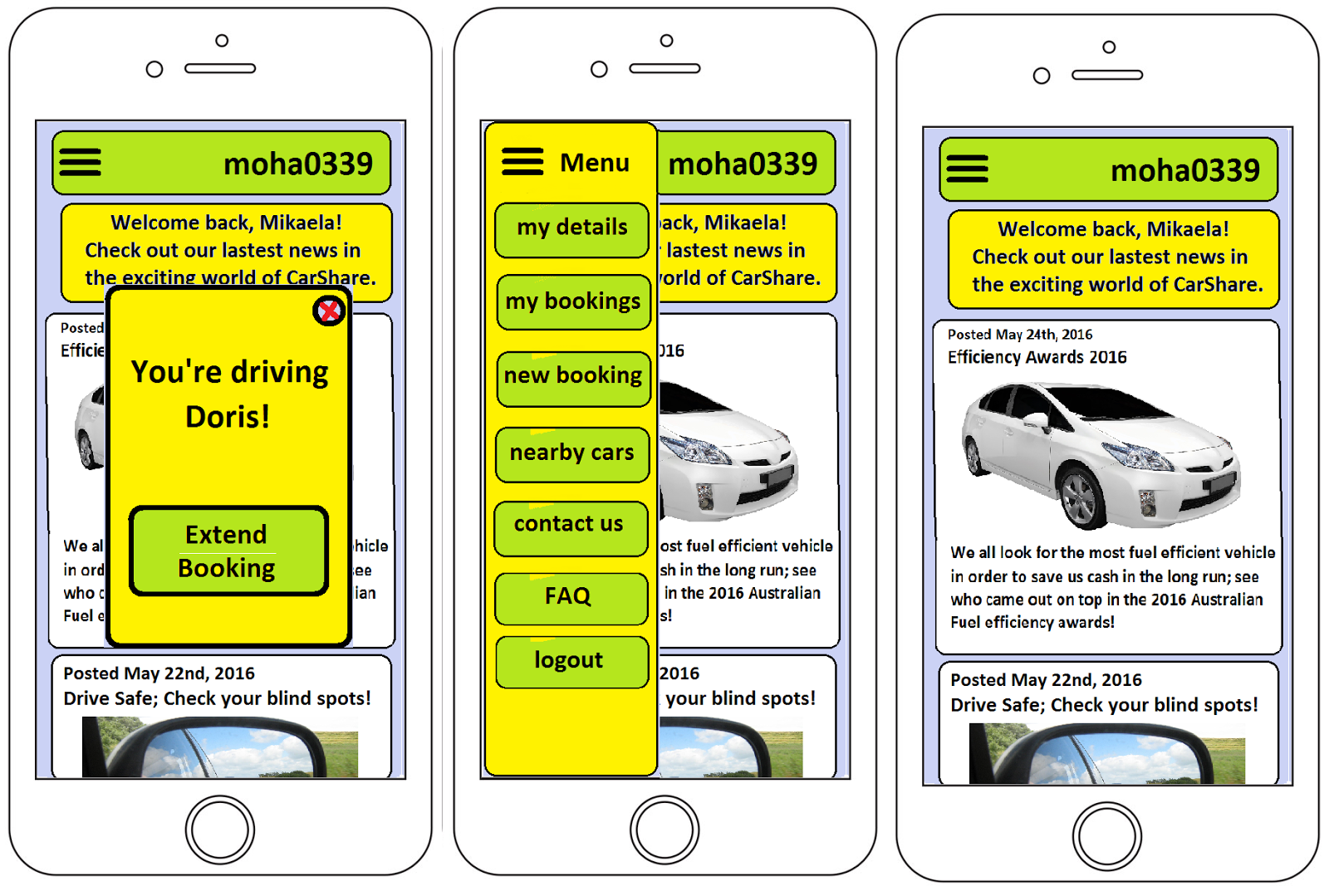
Because Simon wasn’t already on his booking, it’s best that he doesn’t receive any pop ups. It would be very annoying to have a window overlay come up every time you go onto the car share website. It does mean he has a few more clicks than Mikaela, but since he wasn’t in a rush it’s not quite as important.

Below is a very basic idea of the clicks Mikaela would go through.



To implement the above, it would be required that there be a new HTML field for user entry and accompanying CSS, in the form of a drop down menu. In terms of SQL requirements, a call to the bookings that the current member has would then be compared to the current time and the time of earliest booking, which in this case would be their booking currently in use. The name, and current end point of booking would be required, and the duration of the booking would be updated to the new extension provided by HTML input. The start time of booking and duration of requested extended booking would be compared to all current bookings on the car. In the above example, if Doris the Yaris had been booked 1 hour after the current booking was to end, and an extension of two hours was requested, the python would return a false result and the HTML would then tell the user that such a booking is not allowed due to a pre-existing booking. This additional feature would be a standalone function in the database.py, and as such would not require any modifications to current code, aside from links on user account page or main page to the new HTML address.

Below is a simple mock up design, with a side menu for mobile use and a few interesting links to articles. The simplistic design and easy to read articles grab the reader’s attention and give users another relevant reason to visit our website, and therefore think of us when they need to book their next car. We have also included some more basic features such as, FAQ, contact us etc., which are very basic but necessary parts of any website.



And also a wireframe of the two main desktop webpages:

