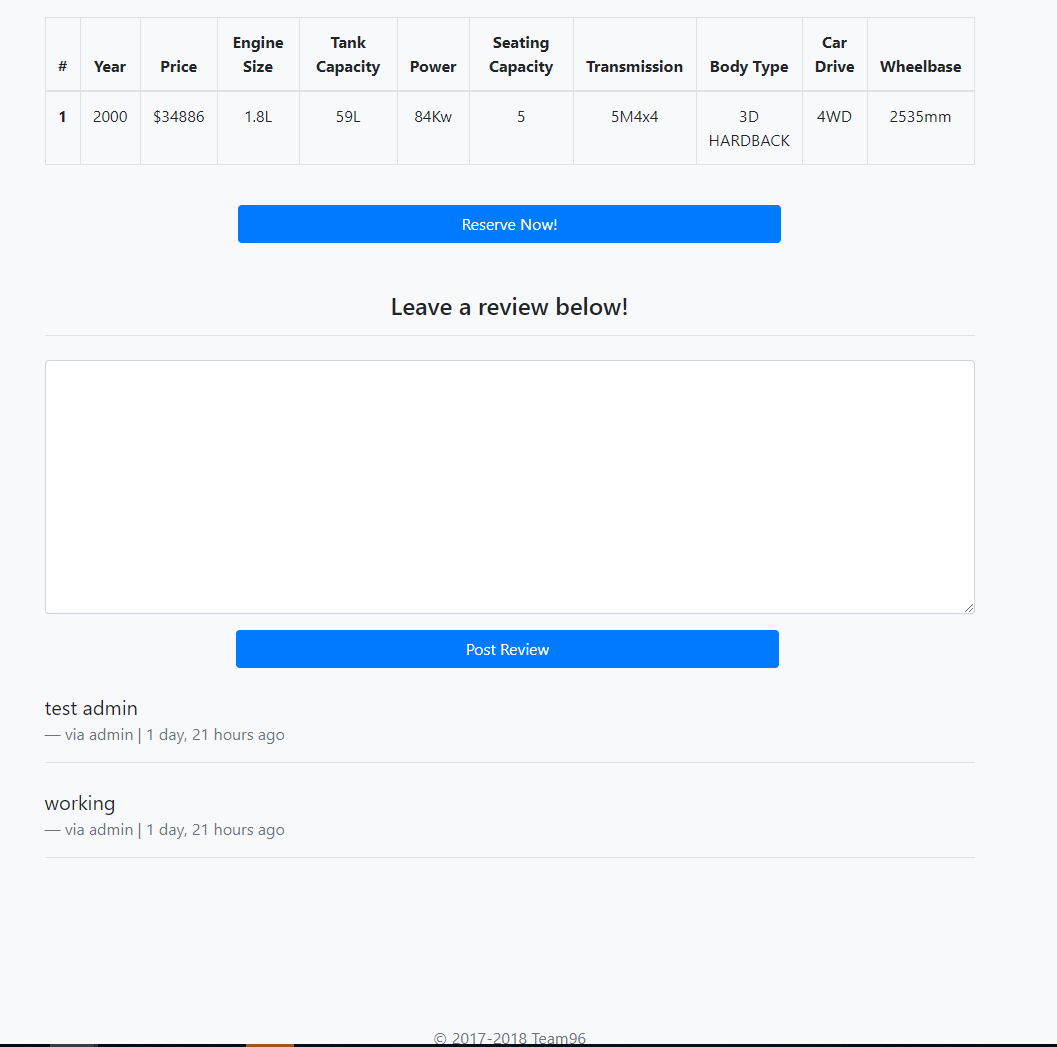
**Personal Portfolio Template**

(Group 96)

(Hugh Duong-Tran-Tien n9747362)

(Github - https://github.com/hughdtt/IFB299-Team96/)

**Artefact 1** – (Code for Reviews) 

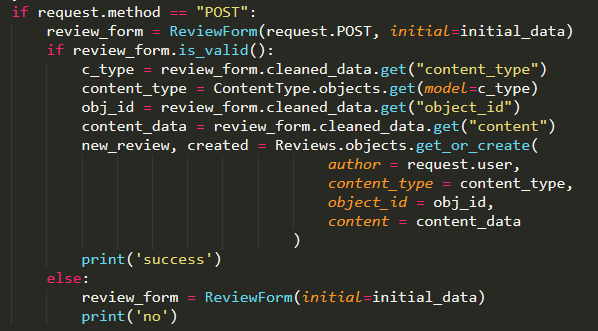
(view page at http://localhost:8000/reservation/car\_id\*/)

\*car\_id can be replaced with any valid car id in the database ie.”14811”

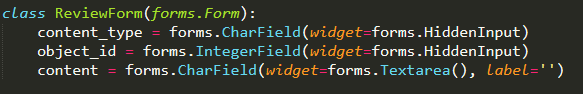
The code below creates the final car\_details.html (as shown above). It works by first, validating anything submitted into the Form, saving the Form data and then displaying the Form data below it (alongside author and timestamp). The code is written using Django’s ‘contenttypes’ framework - providing a high-level, generic interface when working with more than 2 models. This allows us to link both the Reviews model with any another model. For this iteration, Cars and Reviews were linked; each car has its own set of reviews. It also provides scalability as any of the models can be linked; not just Cars and Reviews.

From both management and user perspectives, it is important that users are able to leave reviews/ feedback on rental cars for management to see. Not only this, it potentially opens up pathways for user-user relationships as users can review/ read reviews from previous users.

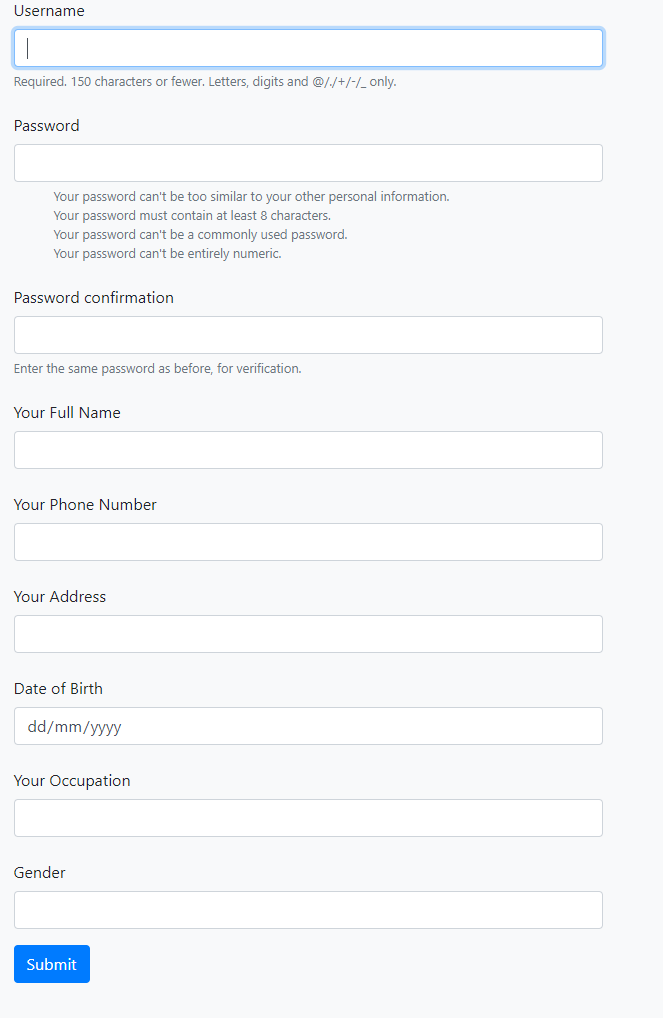
In terms of how it fitted with the team work, it’s necessary to understand how to save Form data and display it since most of this project used many Forms; account creation forms, order forms etc.



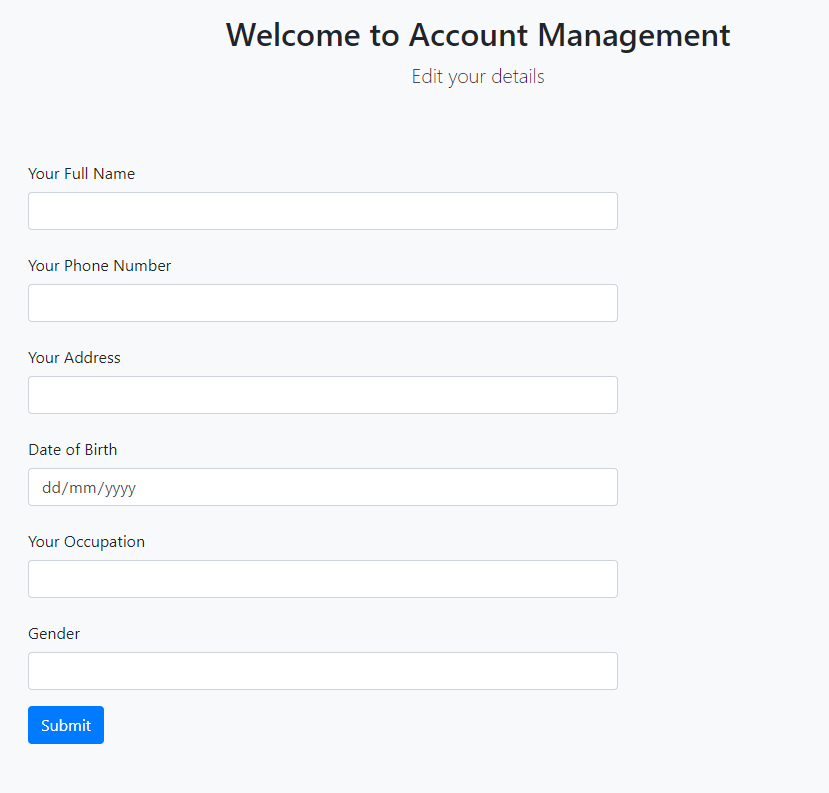
(view full code at IFB299-Team96/Project/Reservation/view.py)



(view full code at IFB299-Team96/Project/Reservation/forms.py)

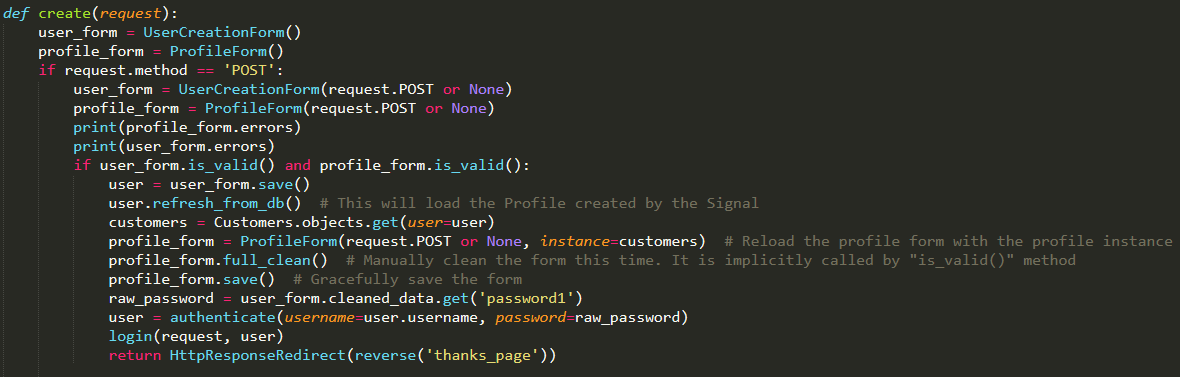
**Artefact 2** – (Code for Account Edit and Registration)  


( view page at <http://localhost:8000/account/register/>)



( view page at <http://localhost:8000/account/edit/>)

As discussed with Clients, it was an important aspect of the Project that users were able to register their own accounts and edit their account information. It was important because it enhances the user experiences, thus creating a better quality final product. This was done with the code shown below.



(view full code at IFB299-Team96/Project/Account/views.py)

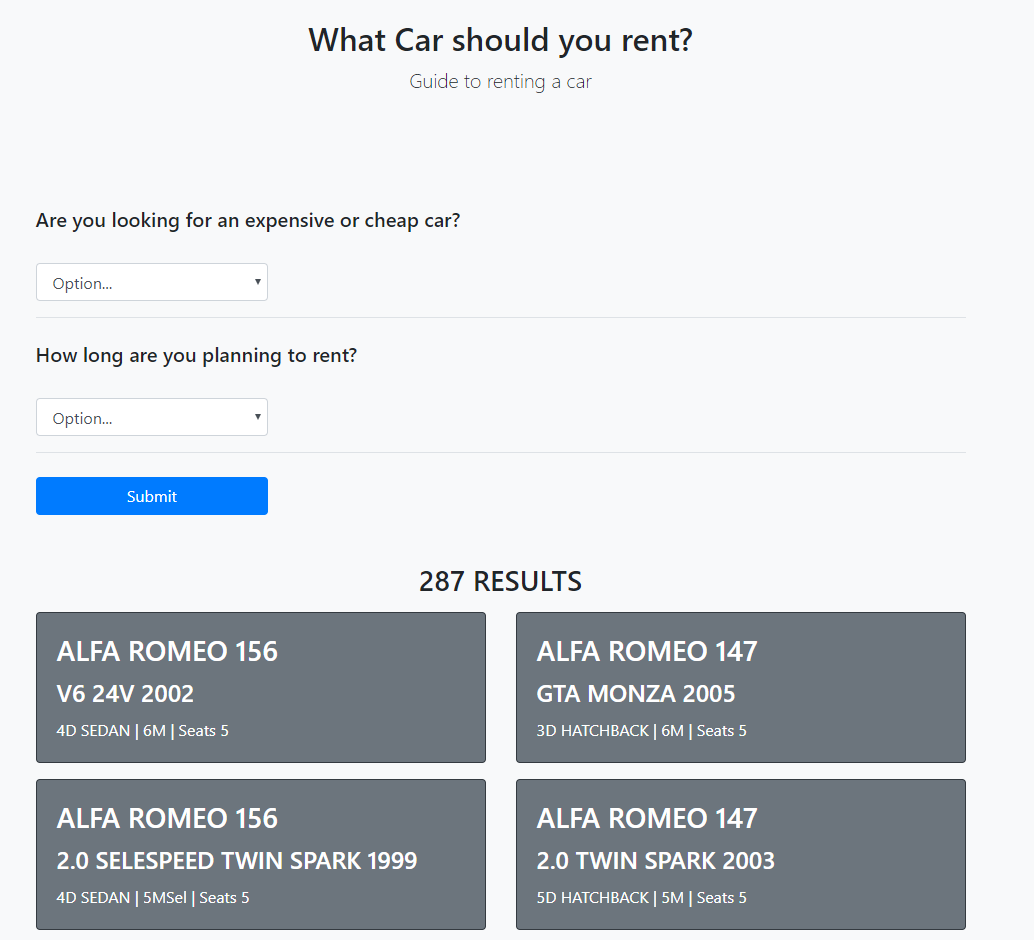


(view full code at IFB299-Team96/Project/Account/views.py)

To successfully create an account, users must correctly fill out a rather generic account registration form. It is filled with the typical fields – usernames, passwords, names, address, contact details, D.O.B etc. The code uses Django’s built in UserCreationForm and our own custom Customer model. The forms works by combining these two forms into a one-step creation form. It first creates the User and then adds the details of that User into a linked Customer model. Then, it automatically logs you in and returns you to a ‘success’ page. This is an upgrade to the code found in Sprint 1 where only the UserCreationForm was implemented.

The Edit page works similarly; it accesses your Customer Model information and allows you to save any new information.

**Artefact 3** – (Code for Recommendations)



(view page at <http://localhost:8000/searchpage/recommendation/>)

It was discussed that some users may not have a clear understanding of what type of car they wanted to rent. To aid them in this experience, a car recommendation system was implemented. The code provided below creates the html view above.



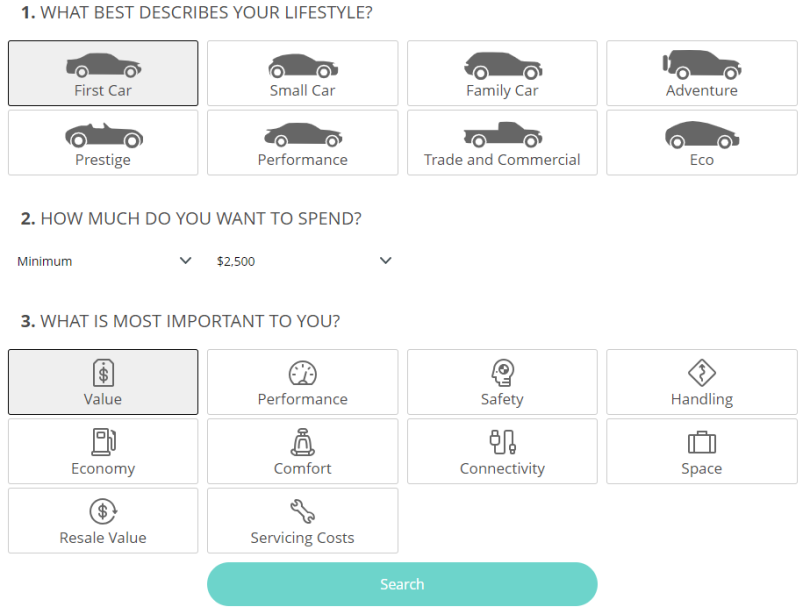
(view full code at IFB299-Team96/Project/Searchpage/views.py)

This code helps filter the various types of cars based on two criteria – cost to rent as well as time rented. Some independent research was done to enhance the quality of this filter. As this process was time-consuming it was discussed with the team to not implement a substantial number of different options but rather keep it simple while still retaining functionality. This is the reason the designs of the recommendations changed and thus have 2 iterations.

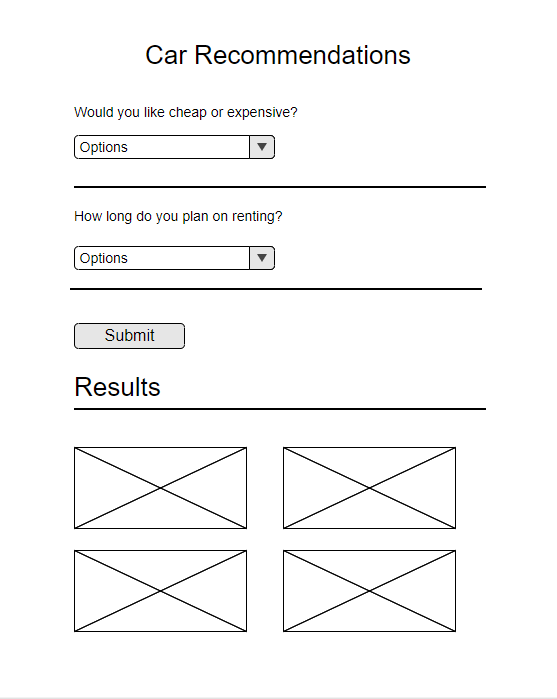
**Artefact 4** – (Mockup for Recommendations )

The mockups seen below were used when designing the interface for the Car Recommendations html (seen in Artefact 3). It turned out that the original mockup was too difficult and time-consuming after a failed implementation as well as discussion with team members.

At the request of other developers, another iteration of the mockup was designed. The second iteration of the mockup was simpler but still withheld the high quality functionality as seen in the Final Release.

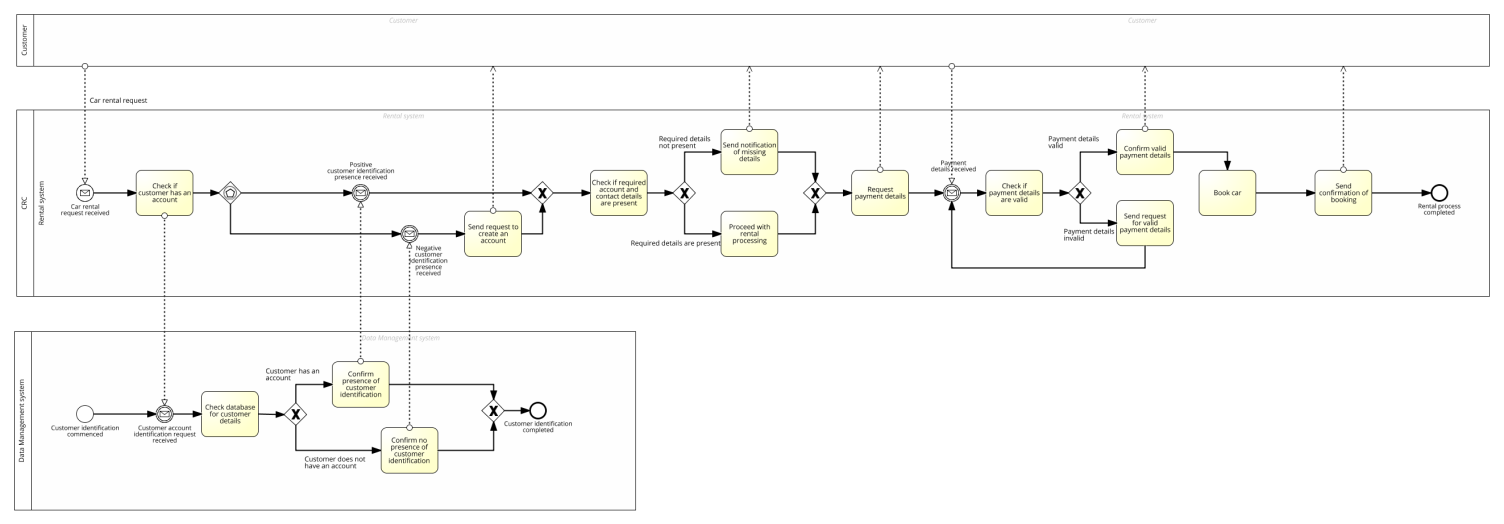


(original mockup – can be found IFB299-Team96/Portfolio Artifacts/Hugh/Sprint2/MockupRecommends1.PNG)



(Second iteration mockup – can be found IFB299-Team96/Portfolio Artifacts/Hugh/Sprint2/MockupRecommends2.PNG)

**Artefact 5** – (BPMN w/ Zac)



A BPMN diagram (as seen above) is a graphical representation of the business model with the current or proposed processes. It was noted early on in the project that Clients may not fully understand all of the technical jargon used by developers. As a result, there needs to be a method for both developers and clients to understand how the product will impact the business structure. From the discussion with the developers, it was decided that a BPMN diagram was the most suitable to help illustrate the business process with the developed data management system. This way, there is clear communication between the two parties and different functionalities can be added/ removed depending on the needs/ feedback from the client. This clear and direct communication resulted in a smooth translation from ideas/ wants into a functional management system.