Personal Portfolio Sprint 2

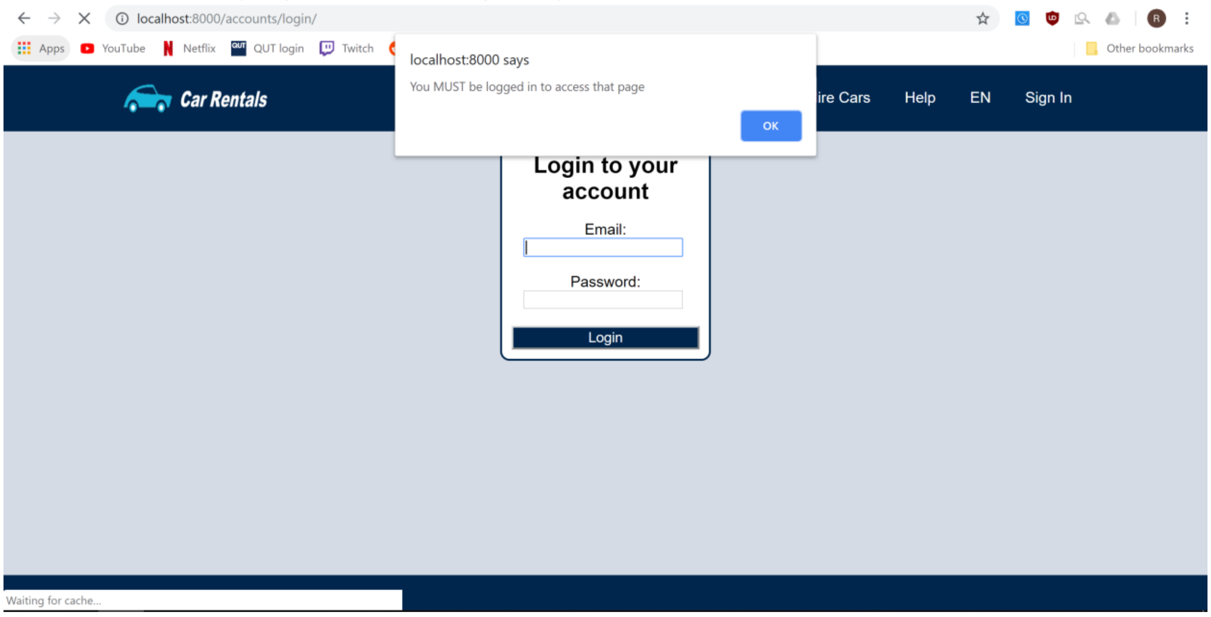
Group 70
Riley Duggan – n9700277
(IFB299-Group-70-S.K.R.A.M)

Artefact 1 – User Verification Class

This is artefact is a class designed to verify users and their authentication level. It has a variety of functions returning if the current user is logged and if there logged in as the correct level of user.

The artefact is used throughout the project as a system for determining user access levels. A function of the class is used whenever the website required user authentication of any kind.

Shows someone trying to access a page they are not allowed to access



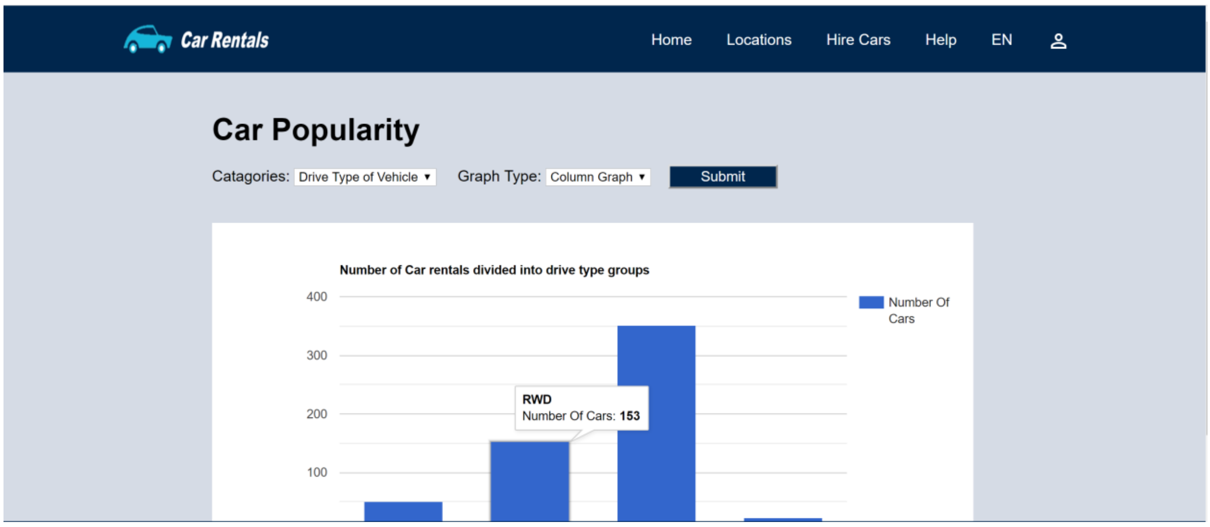
localhost:8000 says

You MUST be logged in to access that page

Artefact 2 – Car Popularity Function

This function is used to create a series of results used to showcase the most popular cars. The function takes the GET inputs from the http request and uses them to determine the grouping of the cars and the type of graph for the results.

This function is used on the car popularity page for all the backend functionality related to sorting the data.



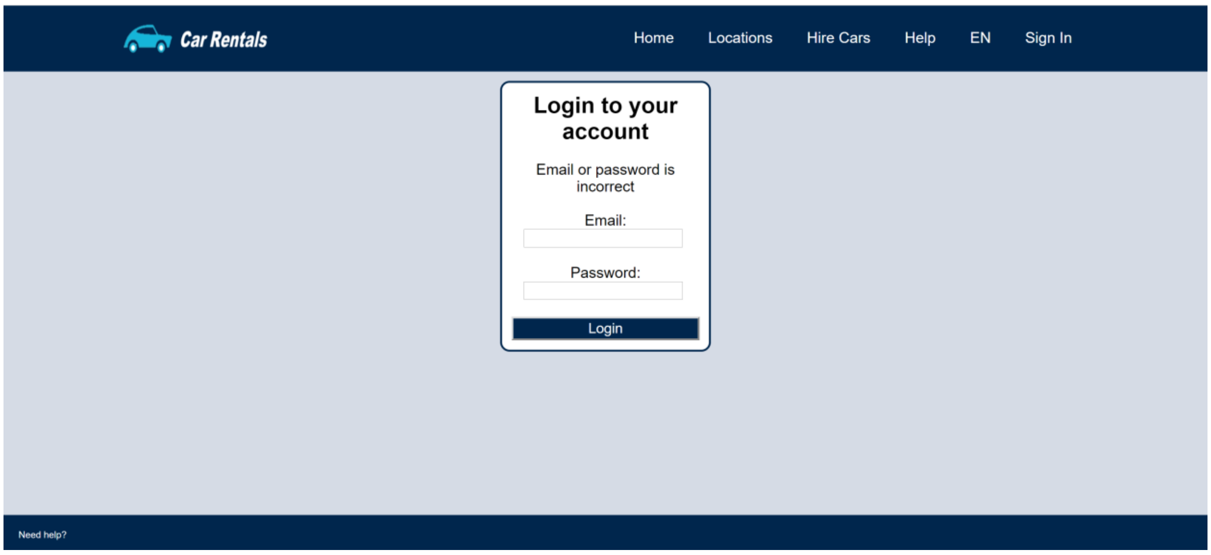
Car Popularity

Catagories: Drive Type of Vehicle ▾ Graph Type: Column Graph ▾ Submit

Artefact 3 – Login page design and implementation

The user interface design of the login page, making sure that the page is easy for a user to understand. An import element of this that is not initially evident is the displaying of error messages which was also designed. This artefact also covers the creation and implementation of this page into the website

The login page is used in the website and it regularly visited page for both customer and staff as there is only one login page for both groups.



Login to your account

Email or password is incorrect

Email:

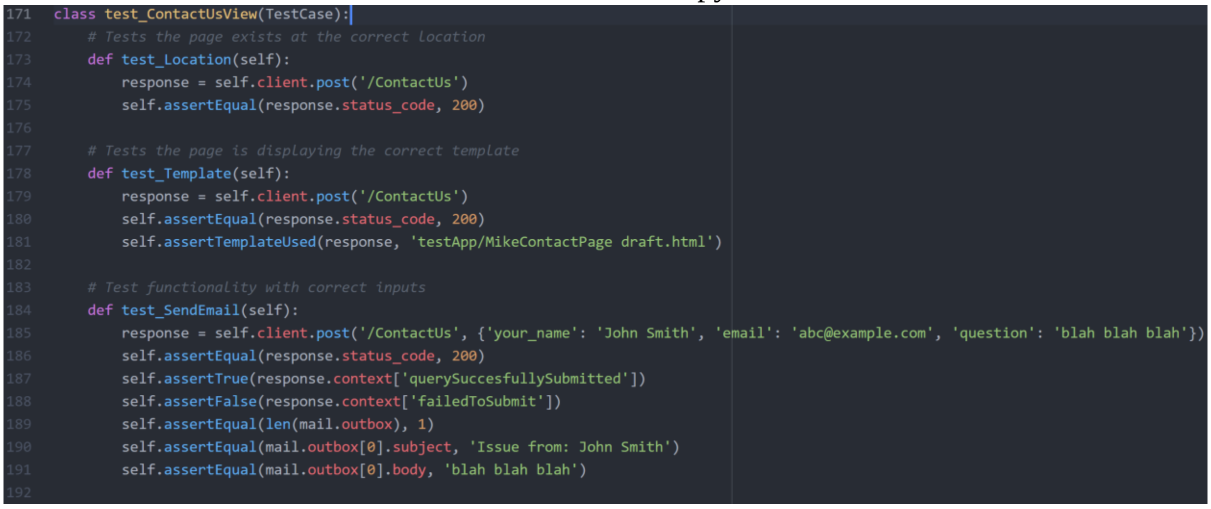
Password:

Artefact 4 – Contact Us view unit test

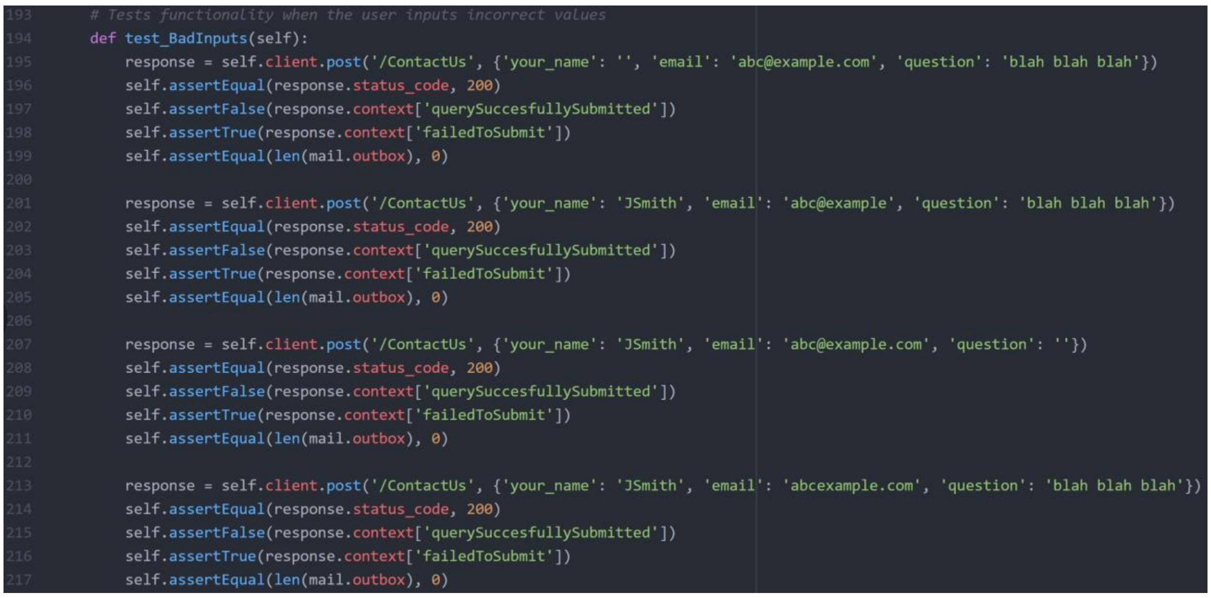
Testing the functionality of the Contact Us page of the website. The tests confirm that all the functionality of the page is working as expected, even for potential edge cases, such as strange inputs from the user.

This testing is used to confirm that the page is working as expected when changes are still being made to the code and to show to the client that the page is working when we are showcasing the website.

The code can be found in lines 171-217 of test_views.py

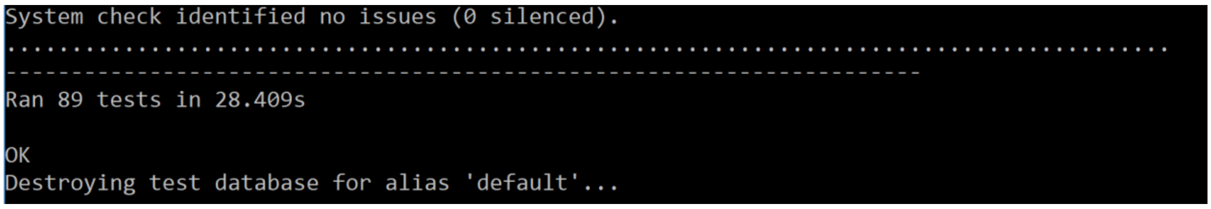


```
171 class test_ContactUsView(TestCase):
172     # Tests the page exists at the correct location
173     def test_Location(self):
174         response = self.client.post('/ContactUs')
175         self.assertEqual(response.status_code, 200)
176
177     # Tests the page is displaying the correct template
178     def test_Template(self):
179         response = self.client.post('/ContactUs')
180         self.assertEqual(response.status_code, 200)
181         self.assertTemplateUsed(response, 'testApp/MikeContactPage draft.html')
182
183     # Test functionality with correct inputs
184     def test_SendEmail(self):
185         response = self.client.post('/ContactUs', {'your_name': 'John Smith', 'email': 'abc@example.com', 'question': 'blah blah blah'})
186         self.assertEqual(response.status_code, 200)
187         self.assertTrue(response.context['querySuccesfullySubmitted'])
188         self.assertFalse(response.context['failedToSubmit'])
189         self.assertEqual(len(mail.outbox), 1)
190         self.assertEqual(mail.outbox[0].subject, 'Issue from: John Smith')
191         self.assertEqual(mail.outbox[0].body, 'blah blah blah')
192
```



```
193     # Tests functionality when the user inputs incorrect values
194     def test_BadInputs(self):
195         response = self.client.post('/ContactUs', {'your_name': '', 'email': 'abc@example.com', 'question': 'blah blah blah'})
196         self.assertEqual(response.status_code, 200)
197         self.assertFalse(response.context['querySuccesfullySubmitted'])
198         self.assertTrue(response.context['failedToSubmit'])
199         self.assertEqual(len(mail.outbox), 0)
200
201         response = self.client.post('/ContactUs', {'your_name': 'JSmith', 'email': 'abc@example', 'question': 'blah blah blah'})
202         self.assertEqual(response.status_code, 200)
203         self.assertFalse(response.context['querySuccesfullySubmitted'])
204         self.assertTrue(response.context['failedToSubmit'])
205         self.assertEqual(len(mail.outbox), 0)
206
207         response = self.client.post('/ContactUs', {'your_name': 'JSmith', 'email': 'abc@example.com', 'question': ''})
208         self.assertEqual(response.status_code, 200)
209         self.assertFalse(response.context['querySuccesfullySubmitted'])
210         self.assertTrue(response.context['failedToSubmit'])
211         self.assertEqual(len(mail.outbox), 0)
212
213         response = self.client.post('/ContactUs', {'your_name': 'JSmith', 'email': 'abcexample.com', 'question': 'blah blah blah'})
214         self.assertEqual(response.status_code, 200)
215         self.assertFalse(response.context['querySuccesfullySubmitted'])
216         self.assertTrue(response.context['failedToSubmit'])
217         self.assertEqual(len(mail.outbox), 0)
```

Showcasing that the tests do not fail



```
System check identified no issues (0 silenced).
.........................................................................................
----------------------------------------------------------------------
Ran 89 tests in 28.409s

OK
Destroying test database for alias 'default'...
```