Personal Portfolio - Sprint 2

Group Number: Team 92 – Blink 192
Name: Taylor Bindon
Student Number: n9296506
GitHub: taylorbindon

Artefact 1 – Linking Pages In Django

Description

After Kirsten Moylan committed 'Every Webpage', I was able to take the static files and link them in Django so that we could run the server and navigate the website. In this iteration, a Model View Function (MVF) structure was implemented as the pages were simply navigating from one to another at this point, there was no login or token restrictions preventing access to certain parts of the website at this point.

How It Contributed to the Project

Having all of the pages connected together enabled real development in Django to begin. This was because the initial HTML layouts were finished, meaning that the backend work could be added in to create the functional components of the website and replace the placeholders. This further aided the establishment of the development environment as the static files folder was implemented for the storage of the HTML pages and their respective images and reports.

Where It Is Used

This structure has been used throughout the project as it enables the website to actually be used and operated. Without this structure in place the project could not begin. As I was the only member in the group intent on learning Django, this task was designated to me, along with all subsequent Django utilities that were implemented, such as forms.py and middleware.py.

GitHub Commit – taylorbindon/IFB299/Linking Pages Together

Artefact 2 – Database Queries and Search Results (Partially Shared)

Description

Kirsten Moylan initially attempted a query in her commits 'adding search functionality' and 'search query'. These attempts contained functions that were correct, but the query itself did not work due to incorrect variables being passed into the queries. In the 'Employee screen – Customer Search Working Commit' I implemented our first query. This produces results for an employee searching for a customer. With this initial query established, Kirsten and Wenona implemented employee vehicle and customer vehicle queries in Kirsten's 'search functions' commit, there were some minor errors as her database connection was not working correctly, however these were corrected in my commit 'Search Function Fixes – Working'.

The queries also required the implementation of the Model View Class (MVC) structure, as opposed to MVF structure, to be able to use the forms easier and display the results on the HTML page. I implemented the MVC calls in the Views.py file for the views that were utilising Django forms and they were called through urls.py as views.

How It Contributed to the Project

By having the database queries working, we were able to complete many of our user stories that required data to be displayed to a user. Furthermore, I had to implement the forms.py module and update the models.py module. The update to models.py included migrations that needed to be made to correct discrepancies between the models and the database, and to allow easier setup of the Django Admin page. Forms.py allowed me to create the forms and inputs for the queries, including regular text inputs and drop down selections.

Furthermore, along with the creation of the queries, I implemented Ginja script that displayed the results of the query to the relative search page.

Where It Is Used

These queries are used in all search functions, as shown below in the 'Store Search' example:

Github Commit – taylorbindon/IFB299/Employee screen – Customer Search Working

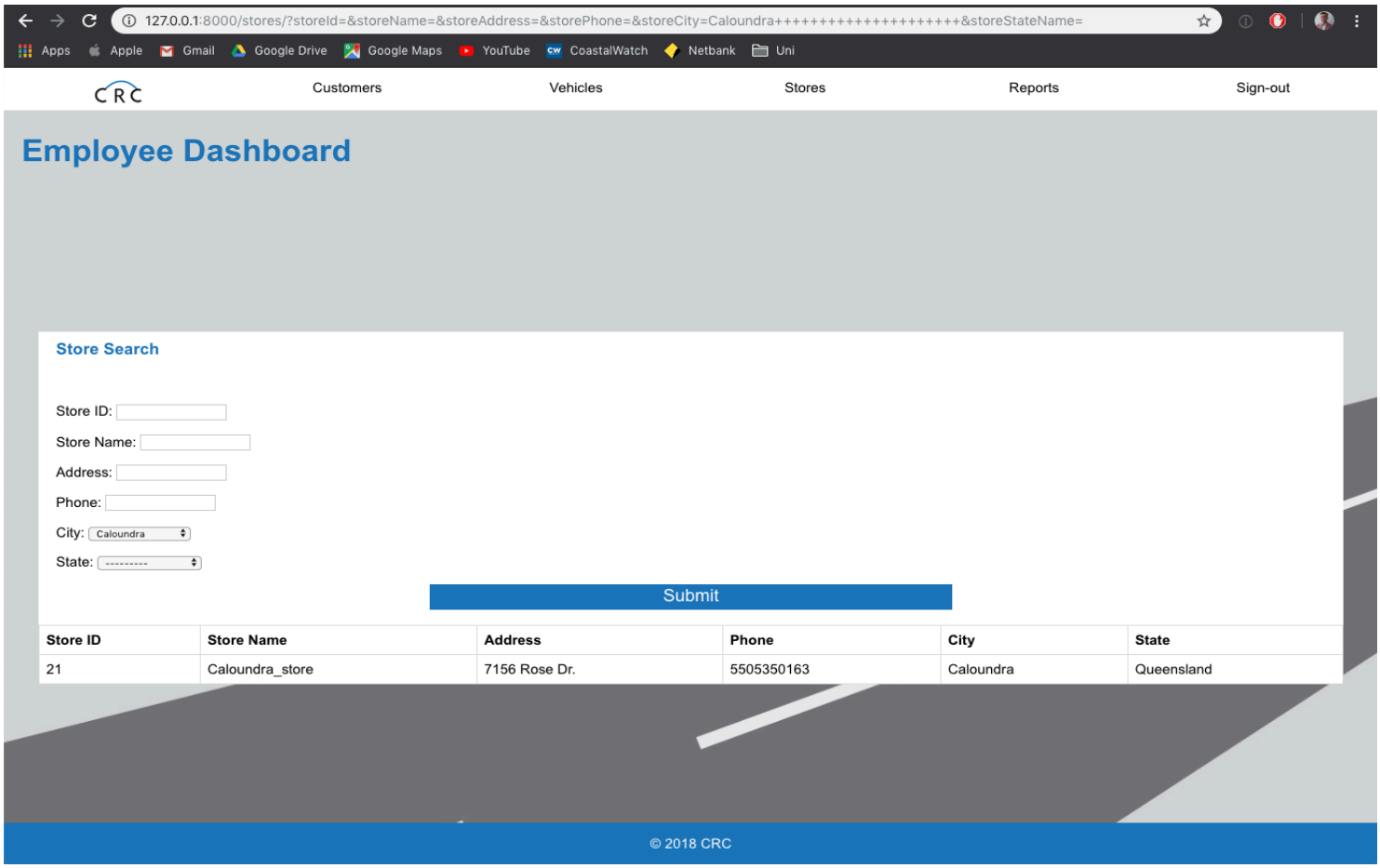
Github Commit – taylorbindon/IFB299/Search function fixes – Working

Github Commit – taylorbindon/IFB299/Implemented All Searches – Working

Github Commit – taylorbindon/IFB299/Admin Page with models

- Referencing this for the model work that allowed the database to work correctly.

P.T.O for example.



Views.py:

```
class EmployeeStores(FormView):
    def get(self, request):
        form = StoreSearch(self.request.GET or None)
        context = {'form': form}

        if form.is_valid():
            storeId = self.request.GET.get('storeId')
            storeName = self.request.GET.get('storeName')
            storeAddress = self.request.GET.get('storeAddress')
            storePhone = self.request.GET.get('storePhone')
            storeCity = self.request.GET.get('storeCity')
            storeStateName = self.request.GET.get('storeStateName')



            results = Stores.objects.filter(Q(store_id__icontains = storeId) & Q(store_name__icontains = storeName) & Q(store_address__icontains = storeAddress) & Q(stor
            storeIds = Stores.objects.filter(Q(store_id__icontains = storeId) & Q(store_name__icontains = storeName) & Q(store_address__icontains = storeAddress) & Q(sto

            context = {'form': form, 'results': results}
            return render (request, 'CRCApplication/stores.html', context)
        else:
            return render(request, 'CRCApplication/stores.html', context)
```

Forms.py:

```
class StoreSearch(forms.Form):
    storeId = forms.CharField(required = False, label = 'Store ID')
    storeName = forms.CharField(required = False, label = 'Store Name')
    storeAddress = forms.CharField(required = False, label = 'Address')
    storePhone = forms.CharField(required = False, label = 'Phone')
    storeCity = forms.ModelChoiceField(required = False, label = 'City', queryset = Stores.objects.values_list('store_city', flat = True).distinct(), to_field_name = 's
    storeStateName = forms.ModelChoiceField(required = False, label = 'State', queryset = Stores.objects.values_list('store_state_name', flat = True).distinct(), to_fie
```

Artefact 3 – Katalon Automated Testing

Description

Upon completion of the minimum viable product that our group achieved, accurate testing could be conducted for the whole application. This was achieved through the use of a Selenium based automated web testing program, Katalon. The software allows the website to be tested as a user regularly would, through click throughs and data entry, and Katalon captures the sequence of events and saves it as a test case. Customer facing and employee facing test cases were implemented in order to test the various components of the website.

The only drawback to this software was when testing the session restrictions, i.e. anonymous users cannot access certain features. Because an anonymous user is redirected to the sign in page for any page the requires a session token to access, the software was attempting to request a page it did not have access to, and thus could not test the page. Although this component is not included in the test cases, there are very few pages that are restricted and these were manually tested instead.

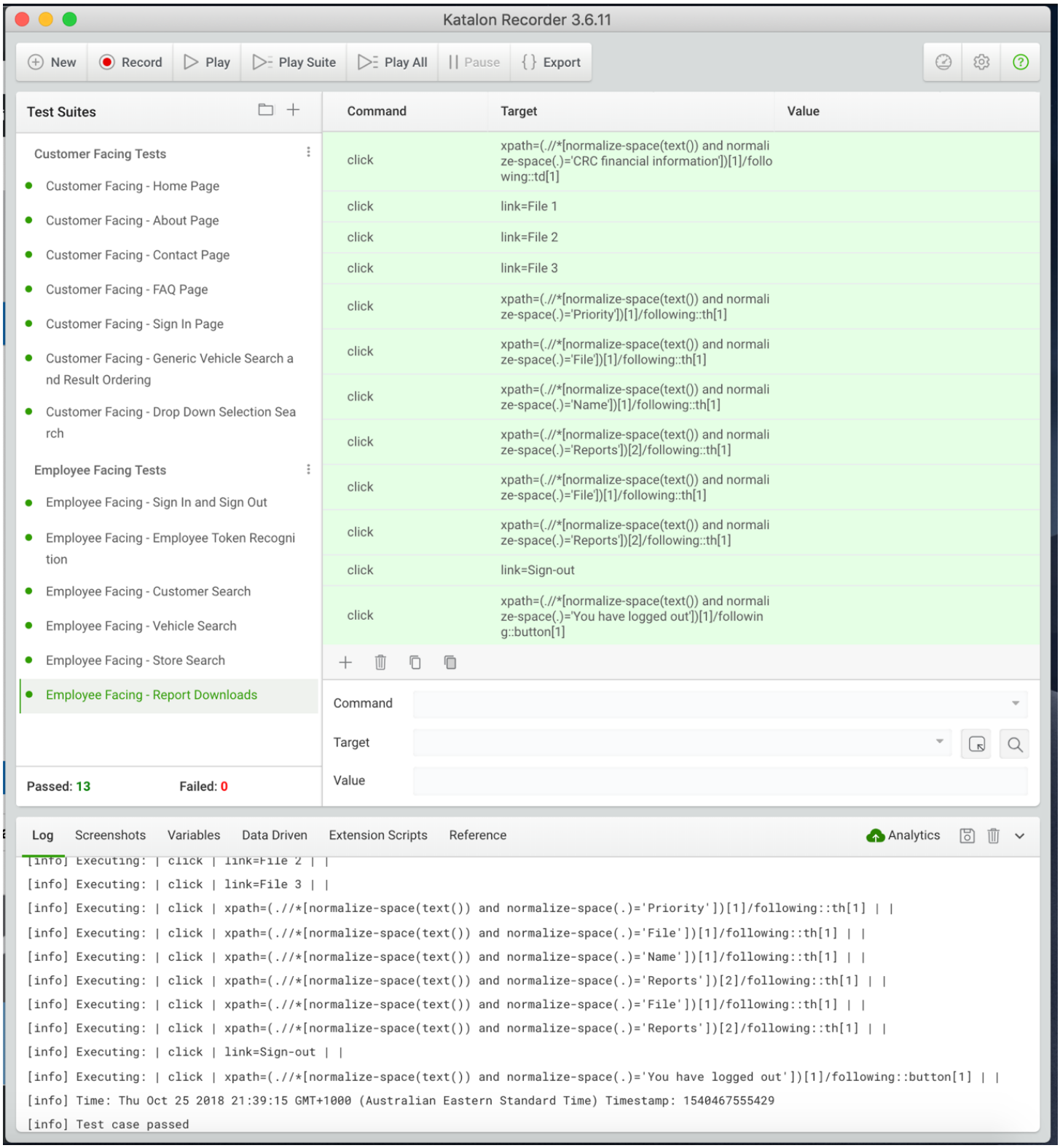
How It Contributed to the Project

This testing allows a single script to be run to test all components of the web application. This contributes to the project as it allows the team to see how their changes affect the test cases, and creates a simple testing method to provide accountability to the client.

Where It Is Used

Github Commit – taylorbindon/IFB299/Katalon Automated Testing

P.T.O for Katalon Chrome Extension UI and passed test cases.


Katalon Recorder 3.6.11
New
Record
Play
Play Suite
Play All
Pause
Export
Test Suites
Customer Facing Tests
Customer Facing - Home Page
Customer Facing - About Page
Customer Facing - Contact Page
Customer Facing - FAQ Page
Customer Facing - Sign In Page
Customer Facing - Generic Vehicle Search and Result Ordering
Customer Facing - Drop Down Selection Search
Employee Facing Tests
Employee Facing - Sign In and Sign Out
Employee Facing - Employee Token Recognition
Employee Facing - Customer Search
Employee Facing - Vehicle Search
Employee Facing - Store Search
Employee Facing - Report Downloads
Passed: 13
Failed: 0
Command
Target
Value
click
xpath=(.//*[normalize-space(text()) and normalize-space(.)='CRC financial information'])[1]/following::td[1]
click
link=File 1
click
link=File 2
click
link=File 3
click
xpath=(.//*[normalize-space(text()) and normalize-space(.)='Priority'])[1]/following::th[1]
click
xpath=(.//*[normalize-space(text()) and normalize-space(.)='File'])[1]/following::th[1]
click
xpath=(.//*[normalize-space(text()) and normalize-space(.)='Name'])[1]/following::th[1]
click
xpath=(.//*[normalize-space(text()) and normalize-space(.)='Reports'])[2]/following::th[1]
click
xpath=(.//*[normalize-space(text()) and normalize-space(.)='File'])[1]/following::th[1]
click
xpath=(.//*[normalize-space(text()) and normalize-space(.)='Reports'])[2]/following::th[1]
click
link=Sign-out
click
xpath=(.//*[normalize-space(text()) and normalize-space(.)='You have logged out'])[1]/following::button[1]
Log
Screenshots
Variables
Data Driven
Extension Scripts
Reference
Analytics
[info] Executing: | click | link=File 2 | |
[info] Executing: | click | link=File 3 | |
[info] Executing: | click | xpath=(.//*[normalize-space(text()) and normalize-space(.)='Priority'])[1]/following::th[1] | |
[info] Executing: | click | xpath=(.//*[normalize-space(text()) and normalize-space(.)='File'])[1]/following::th[1] | |
[info] Executing: | click | xpath=(.//*[normalize-space(text()) and normalize-space(.)='Name'])[1]/following::th[1] | |
[info] Executing: | click | xpath=(.//*[normalize-space(text()) and normalize-space(.)='Reports'])[2]/following::th[1] | |
[info] Executing: | click | xpath=(.//*[normalize-space(text()) and normalize-space(.)='File'])[1]/following::th[1] | |
[info] Executing: | click | xpath=(.//*[normalize-space(text()) and normalize-space(.)='Reports'])[2]/following::th[1] | |
[info] Executing: | click | link=Sign-out | |
[info] Executing: | click | xpath=(.//*[normalize-space(text()) and normalize-space(.)='You have logged out'])[1]/following::button[1] | |
[info] Time: Thu Oct 25 2018 21:39:15 GMT+1000 (Australian Eastern Standard Time) Timestamp: 1540467555429
[info] Test case passed

Artefact 4 – Login Form and Admin Page

Description

The employee login and admin pages were created in order to allow employees and directors to access and edit information pertaining to the Car Rental Company. This Sprint, I implemented the employee login so that employees could see the necessary information they need in order to carry out their employee duties. This login system utilises the Django authorisation process and has several default fields and values included. The login was completed in this manner so that in future releases, when directors need access to specific information, an enumerable field will be able to be added to the database and models that specifies an employee's role at the Car Rental Company. This field will dictate what pages they are able to view, similar to the session info system that I also implemented for the project.

The default Django administration page was utilised as this is the easiest way to add, edit, and delete fields throughout the database, without having to run SQL queries. It is intended that directors would be able to use the admin page to create employees and add more data to the database.

How It Contributed to the Project

This artefact contributed greatly to the project as it enabled the group to complete multiple user stories. The artefact provides directors with some initial control over the data in the database upon this release and employees can complete searches after logging in.

This artefact also required some consolidation of data between the database and the models.py file, as without the migrations being run, the admin page would not operate. This was due to how the database was originally created and lacked primary keys.

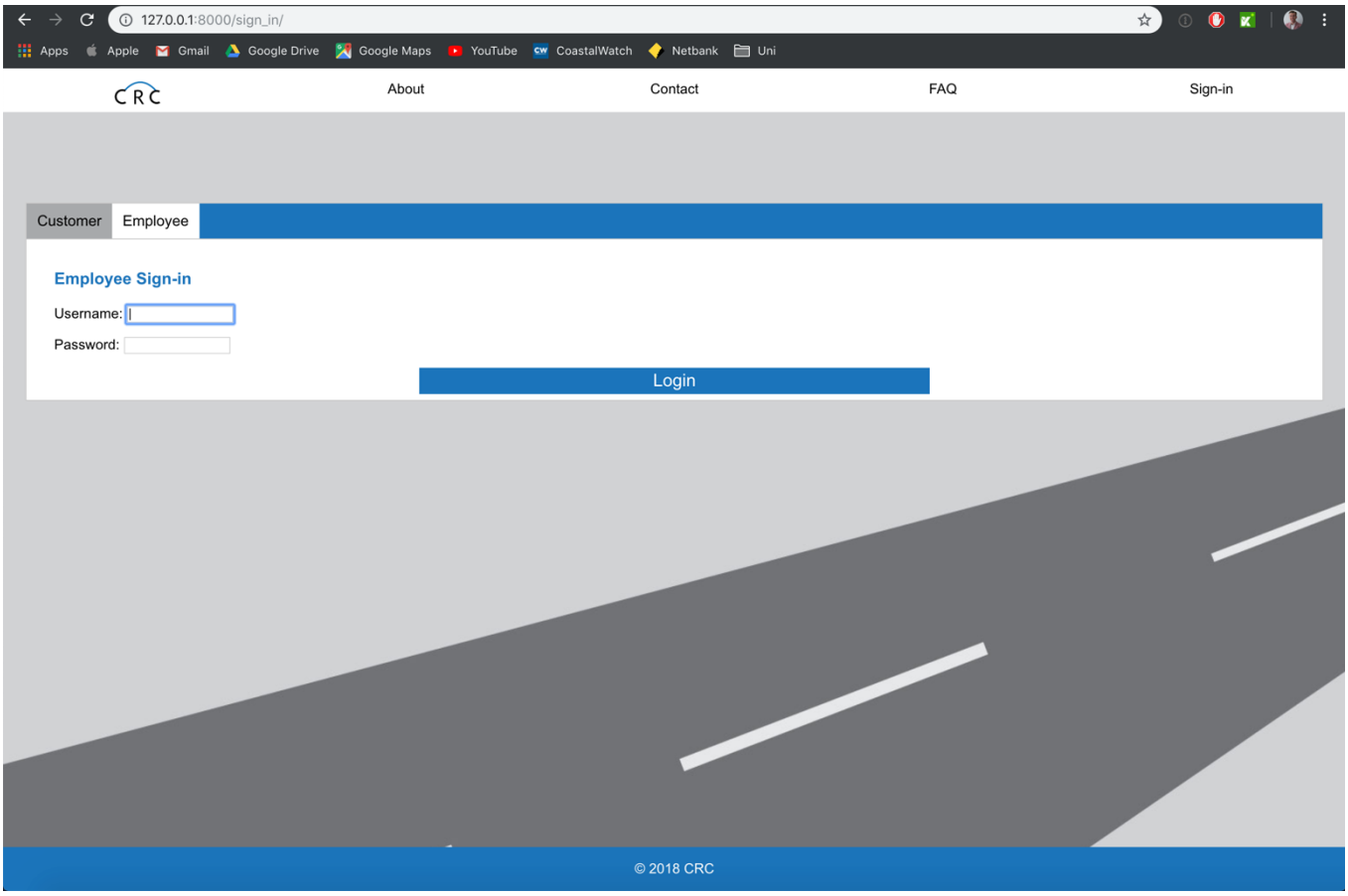
Where It Is Used

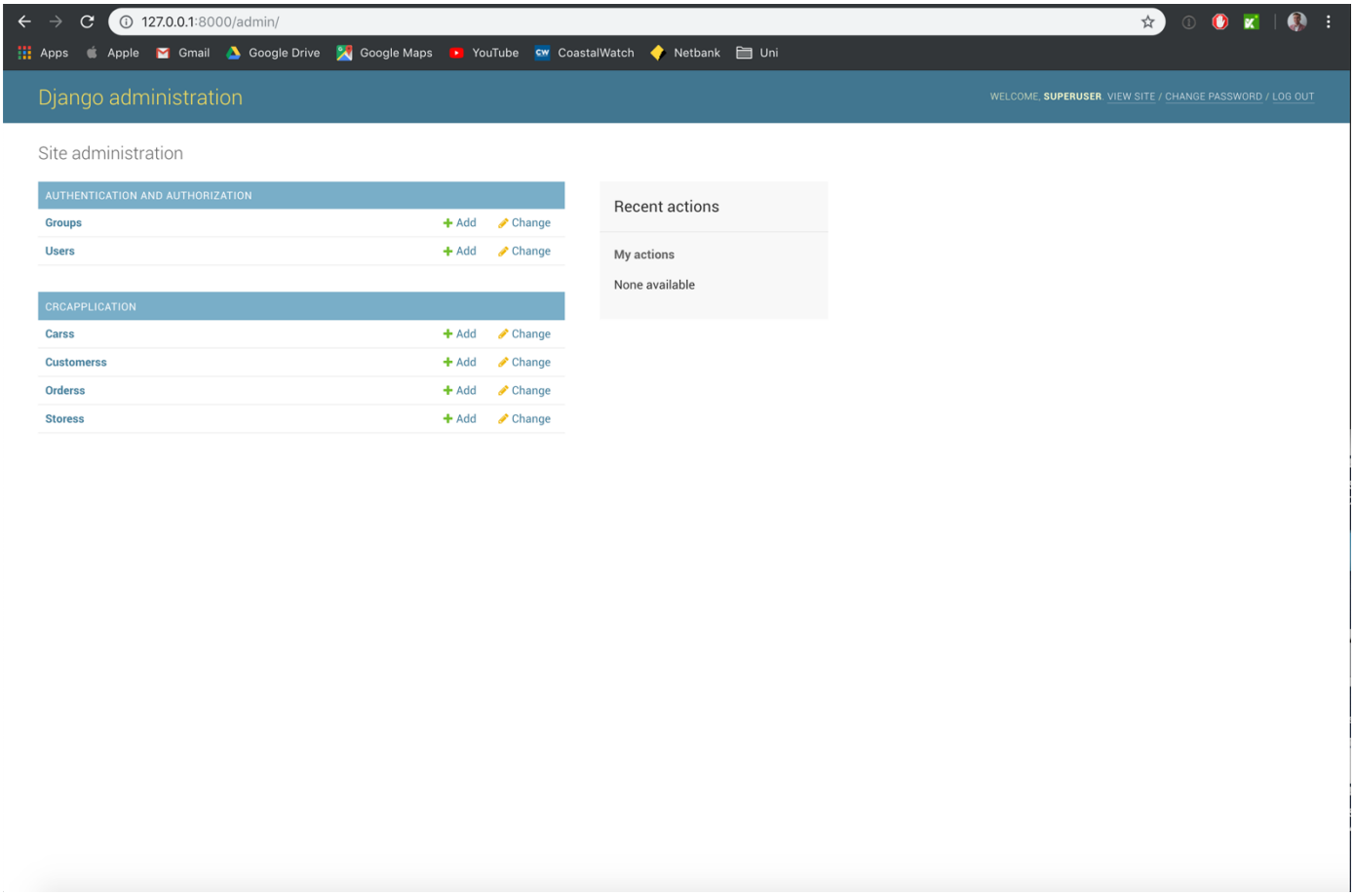
Github Commit – taylorbindon/IFB299/Admin Page and Employee Login – Pre Migrations

Github Commit – taylorbindon/IFB299/Added Primary Keys to DB and models.py

Github Commit – taylorbindon/IFB299/Admin Page with models

P.T.O for login page and admin page examples.


127.0.0.1:8000/sign_in/
Apps
Apple
Gmail
Google Drive
Google Maps
YouTube
CoastalWatch
Netbank
Uni
CRC
About
Contact
FAQ
Sign-in
Customer
Employee
Employee Sign-in
Username:
Password:
Login
© 2018 CRC


127.0.0.1:8000/admin/
Apps
Apple
Gmail
Google Drive
Google Maps
YouTube
CoastalWatch
Netbank
Uni
Django administration
WELCOME, SUPERUSER. VIEW SITE / CHANGE PASSWORD / LOG OUT
Site administration
AUTHENTICATION AND AUTHORIZATION
Groups + Add Change
Users + Add Change
CRCAPPLICATION
Carss + Add Change
Customerss + Add Change
Orderss + Add Change
Storess + Add Change
Recent actions
My actions
None available

Artefact 5 – Logout and Session Access

Description

Although the employee pages do not appear in the customer navigation bar, these pages could still be accessed if a user knew the correct url. I implemented session information and middleware that redirects the user if they are not authenticated with the auth_user table in the database, in order to counteract this issue. This session access heavily depends upon the user being authenticated and logging in, and logging out. The Django logout function, which is run whenever a user reaches the logout page, clears the session information for the authenticated user from the database, returning it to the access permissions of a regular anonymous user.

This artefact also required the implementation of middleware. The middleware.py file that I implemented handles requests that are being sent to the server and determines whether or not a user is authenticated, in turn determining what page the user views. The middleware that was implemented will redirect an anonymous user to the sign in page whenever they do not have any session information to prove that they are logged in.

How It Contributed to the Project

Although this artefact is not a major requirement, as set out in the briefing, it is important to produce a product that is both functional and secure. It would be extremely detrimental, from a security perspective, to allow a customer, or anyone on the internet for that matter, to access the admin page. This is because the security goals of the Car Rental Company would be breached and data could be leaked, destroyed, or taken advantage of by attackers.

Where It Is Used

The middleware.py runs on every request and checks to see if the user is authenticated. If the user is *not* authenticated, the middleware redirects the user to the sign in page.

Github Commit – taylorbindon/IFB299/Session access restrictions – need logout page

Github Commit – taylorbindon/IFB299/Middleware Update – correct access restrictions