**Personal Portfolio - Sprint 2**

Group 118

Aliya Rahmanovic – n9365877

GitHub: <https://github.com/ZacAtkinson/IFB299>

**Artefact 1** – Test Cases

NOTE: Khrystel assisted in making this

A test case is a specification of the inputs, execution conditions, testing procedure, and expected results that define a single test to be executed to achieve a particular software testing objective, such as to exercise a particular program path or to verify compliance with a specific requirement. In other words, the a test case consists of a test case description, entry criteria, exit criteria,expected result, actual result and test execution result (pass/fail).

Test cases were created for the project to assist in making sure the website was fully functional and had no software defects or errors when used. The test cases contributed greatly to the project by making sure the website was refined at all stages and all features were still functional.

Refer to Appendix A for full test case document

**Artefact 2** – CSS Style Sheet

The CSS Style sheet is an external document which HTML elements reference and dictates how the elements are to be displayed on screen, paper, or in other media. It can control the layout of multiple web pages all at once. External stylesheets are stored in CSS files.

The CSS style sheet that was specifically created for the website greatly contributed to the project by helping fulfil our desired UI designs. It also helped in making sure the website was unique in design, neat and followed a consistent design style.

Refer to GitHub for the CSS style sheet: testsite > sampleapp > static > sampleapp > css > **navstyle.css**

**Artefact 3** – User Test

A user test is a test in which an outside user (someone not involved in development) is called to test the website. The user’s insight and feedback can help the developers in having an insight of what works and what doesn’t work. It assists in giving developers feedback on certain elements and an idea of what needs improving or what needs to be removed.

It contributed to the project by allowing me to receive outside feedback and gave me ideas on what I can improve and what areas need the most attention. It also helped in giving me an idea on how the client would react to the website as well. Furthermore, it helped me understand what features needed urgent fixing.

Refer to Appendix B for “Test and Report Feedback”

**Artefact 4** – Developed additional functionality to meet specifications

Further development of the site was done including the Log In and Log out functionality, refining the Search Vehicles to include Vehicle Recommendations, developing employee functions such as Deals, Analysis – Sales Leaderboard for stores, Leaderboard for most ordered Vehicles, Store Performance per year and Store Performance on a specific date.

This added functionally greatly contributed to the project by providing greater functionality to the website and assisted in meeting crucial criteria and meeting the demands of the client.

**Views Developed:**

Recommendedvehicles

Analysis

Storeanalysis

Leaderboard

Deals

Mostusedvehicles

Example of one of the views used in the project: analysis view:

def analysis(request):

allmonths = ["January", "Feburary", "March", "April", "May", "June", "July", "August", "September",

"October", "November", "December"]

error = False

if 'month' in request.GET:

month = request.GET['month']

year = request.GET['year']

if not month:

error = True

else:

# query = Order.objects.all().query

# query.group\_by = ['Order\_PickupStore']

# results = QuerySet(query=query, model=Order)

for x in range(11):

if(allmonths[x] == month):

query = x

month = x+1

break

if(month >= 10):

month = str(month)

date = year + "-" + month

else:

month = str(month)

date = year + "-0" + month

#.values() is basically GROUP BY

results = Orders.objects.filter(order\_pickupdate\_\_startswith=date).values('order\_pickupstore').annotate(num\_orders=Count('order\_pickupstore')).order\_by('-num\_orders')

#results = Order.objects.annotate(num\_books=Count(date))

# if(month == 'September'):

# month = '9'

orders = Orders.objects.filter(order\_pickupdate\_\_startswith=date)

return render(request, 'snippets/analysis\_results.html', {'orders': orders, 'query': allmonths[query], 'year': year, 'results': results})

return render(request, 'snippets/analysis\_page.html', {'error': error})

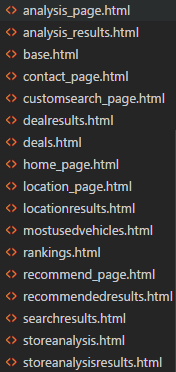
Refer to GitHub for full functionality code

**Artefact 5** – Developed additional HTML pages to meet specifications

HTML is the standard mark-up language for producing web pages and web applications. With CSS and JavaScript, it is apart of a trio of cornerstone technologies for the World Wide Web. To compliment the additional functionalities that were developed (Artefact 4) and in the previous sprint, several HTML pages were created

These HTML pages contributed to the project by making sure all the data that is retrieved from the views/code was properly formatted and displayed in a manner which is easy for the user to view. It also made sure that the entire website could even display anything at all.

The HTML pages created for the project:



Example of one of the HTML pages, customsearch\_page.html

{% extends 'snippets/base.html' %}

{% block title %}CRC - Custom Search{% endblock title %}

{% block content %}

<h2 align=center>Custom Search</h2>

<h2 align=center>Looking for a specific vehicle?</h2>

{% if error %}

<p align=center id='error'>Please submit a make.</p>

<div align=center>

<form action=" " method="get">

<h2> Make </h2>

<input type="text" name="make">

<h2> Model </h2>

<input type="text" name="model">

<h2> Series </h2>

<input type="text" name="series">

<h2> Year </h2>

<input type="text" name="year">

<h2> Seating Capacity </h2>

<input type="text" name="capacity">

<h2> Transmission </h2>

<input type="text" name="transmission">

<p><input type="submit" value="Search"></p>

</div>

{% else %}

<div align=center>

<form action="searchresults/" method="get">

<h2> Make </h2>

<input type="text" name="make">

<h2> Model </h2>

<input type="text" name="model">

<h2> Series </h2>

<input type="text" name="series">

<h2> Year </h2>

<input type="text" name="year">

<h2> Seating Capacity </h2>

<input type="text" name="capacity">

<h2> Transmission </h2>

<input type="text" name="transmission">

<p><input type="submit" value="Search"></p>

</div>

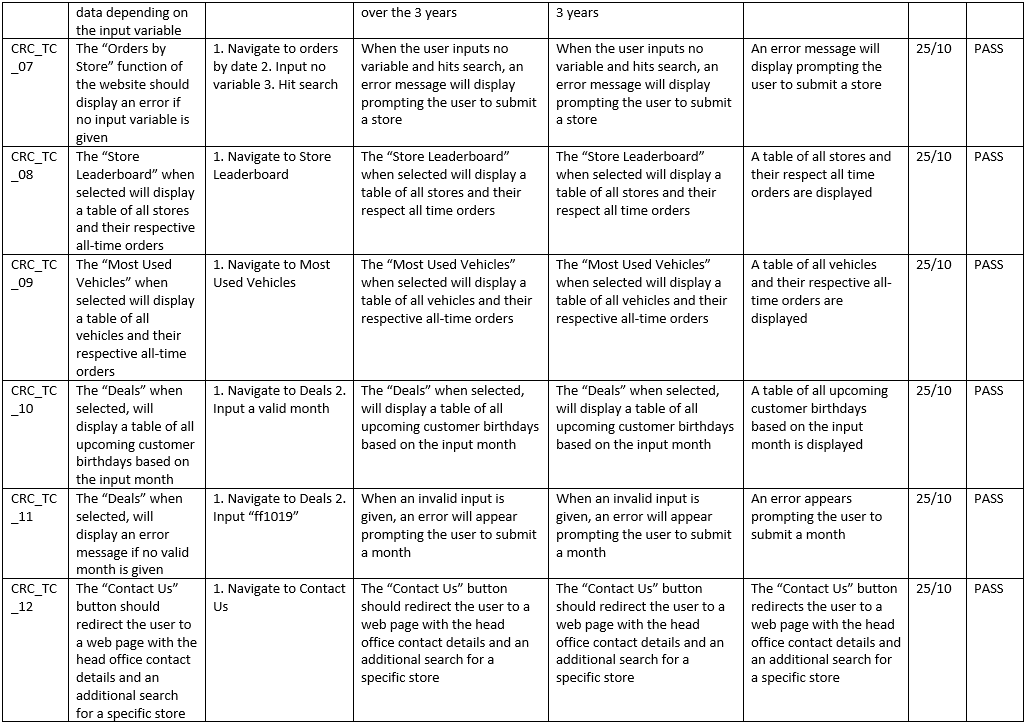
{% endif %}

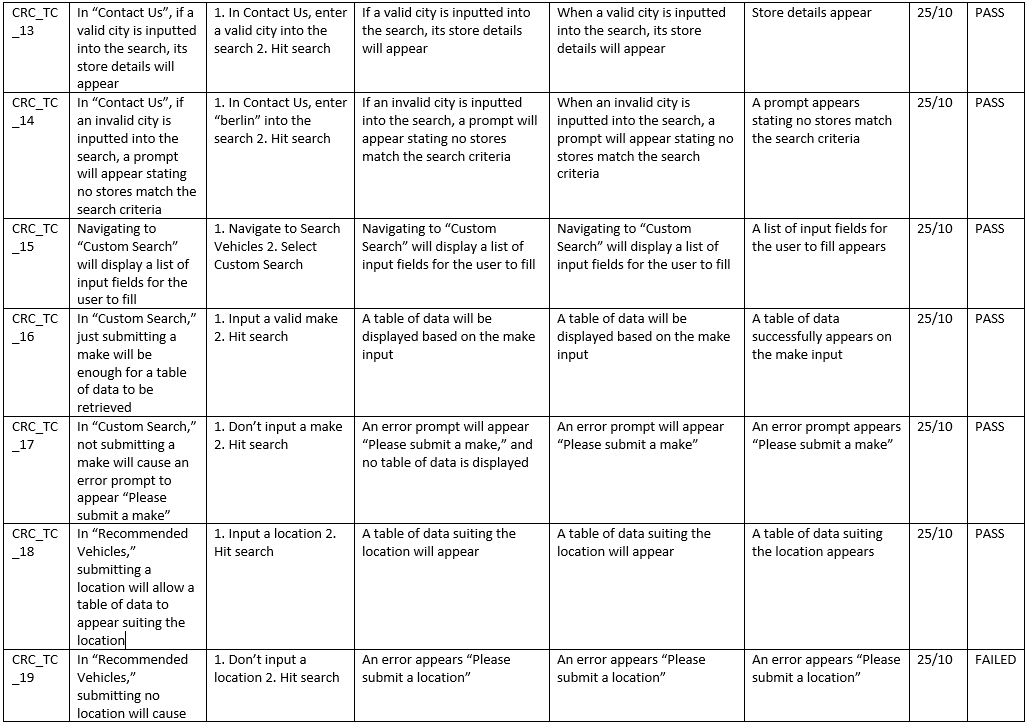
{% endblock content %}

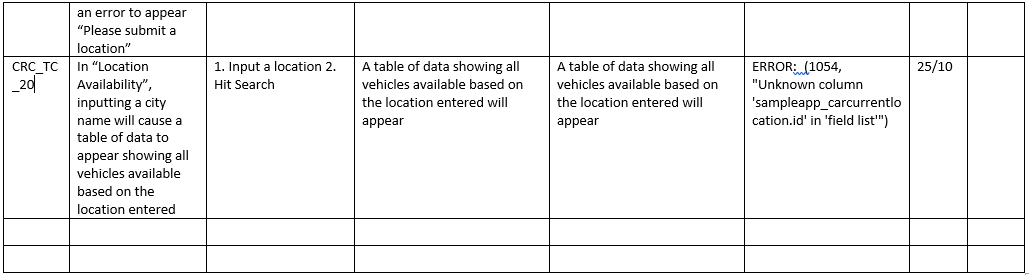
To view all of the HTML pages head to the GitHub, go to the source code and head to templates > snippets

**Appendix A – Test Cases Document**









**Appendix B – Test Report Feedback**

**Test and Feedback Report**

**Participant Name: Christian Davis   
Date of Test: 24/10/2018**

**Any problems?**

* When I tried inputting values into some of the input fields, an error displayed. Specifically, in Vehicle Location, Order by dates, Recommended vehicles and Location Availability. Trying to input a value in those sections and hitting search gave errors
* Some of the pages were not aligned/formatted correctly, some text’s indentation and formatting are inconsistent compared to the rest of the site
* The login doesn’t work at all, inputting the supplied login information does not work and it keeps saying the login information entered is incorrect
* The website logo isn’t displayed, a broken image is displayed instead in the navigation bar
* Some of the result tables could use extra information, for example some display the store ID rather than the store’s name. Displaying the store’s name would help in readability. Furthermore, in Store Performance, the input could instead accept a city name instead of a store ID, helping readability.

**Any positives?**

* Simple and intuitive UI design, the site looks clean and not to overbearing
* The result tables are informative and display only key information
* The navigation bar operates smoothly

**Participant’s closing thoughts:**

The overall website design is simple, but maybe too simple and could use some extra work as some sections felt empty. The errors were abundant and need working on. The result tables were formatted neatly and **when** they appeared they appeared very quickly with no interruptions.

**Developer Notes and Outcomes:**

* Need to fix the Vehicle Location, Order by dates and Recommended vehicles to actually take user input and not display errors
* Work on the UI to add a bit more to the overall design of the website
* Try getting the CRC logo to display in the navigation bar, if still continues to not display replace with text “CRC Rentals”
* Redesign the HTML pages to fix indentation and formatting problems
* Revisit the authentication process for logging in to make sure it works
* Add more employee functionality and add reporting functionality
* Go into the database and try to add foreign keys to work so that the result tables display names rather than IDs, then update models to accommodate foreign key functionality
* Instead of showing Django errors, try catching the errors and redisplaying them as part of the site. This is to make sure it doesn’t look unprofessional and break out of the website
* Update the CSS style sheet to fix indentation and formatting problems
* Add more test cases to the test case document, specifically to test if errors are still showing up. (In Vehicle Location, Order by dates, Recommended vehicles and Location Availability