Dalhousie University
Department of Engineering Mathematics
ENGM4620
Python for Engineers

Dealership Service Maintenance Web Application

Prepared for: Dr Issam Hammad

Created by:
Abdulla Sadoun B00900541
Abdul Hameed Al Busaid B00832820

Table of Contents

.

Repository Link

LINK TO OUR REPOSITORY: https://github.com/AbdullaSadoun/DealerMaintenance

Purpose

The main purpose of this web application is to aid and automate the process of maintenance and service for dealerships and garages. The software will help administrators, mechanics and customers swiftly book appointments and check what services can be done on their selected vehicles..

Objectives

The main objectives of this project are:

- Demonstrate basic use and understanding of Python
- Demonstrate understanding of using Methods
- Demonstrate basic understanding of Django, ORM and the rest of its feature
- Demonstrate basic use of creating a simple SQL Database
- Use of Admin and authentication in web development
- Use of frontend elements like css and html to create user friendly pages.

Requirements

```
asgiref==3.8.1
chardet==5.2.0
Django==5.0.4
pillow==10.3.0
Python==3.10.13
reportlab==4.1.0
sqlparse==0.4.4
typing_extensions==4.11.0
```

How to use software

After ensuring that you have all the required installations from the requirements file, Open up the project and in your terminal ensure by using “cd” command that you are in the DealerMaintenance directory, once complete make sure you have activated the virtual environment by typing in the following command in the terminal: “source env/bin/activate”. After

ensuring you are have (env) by your cli, use the terminal type in: “python manage.py runserver”. You have now launched the app then you can either type in the terminal “open http://127.0.0.1:8000” if you are using mac or for windows: “start http://127.0.0.1:8000” or you can copy paste the following link into your web browser: http://127.0.0.1:8000

Demo

Demo is completed in the video.

Design

When it comes to designing the web application we have reused an identical design to that we have completed for project one. We have included the same figure0 (format taken from System Analysis) to highlight the different processes, stores and terminators that are used by this web app. The different systems and how they interact with each other are underscored in the figure, showing how the solution is reached.

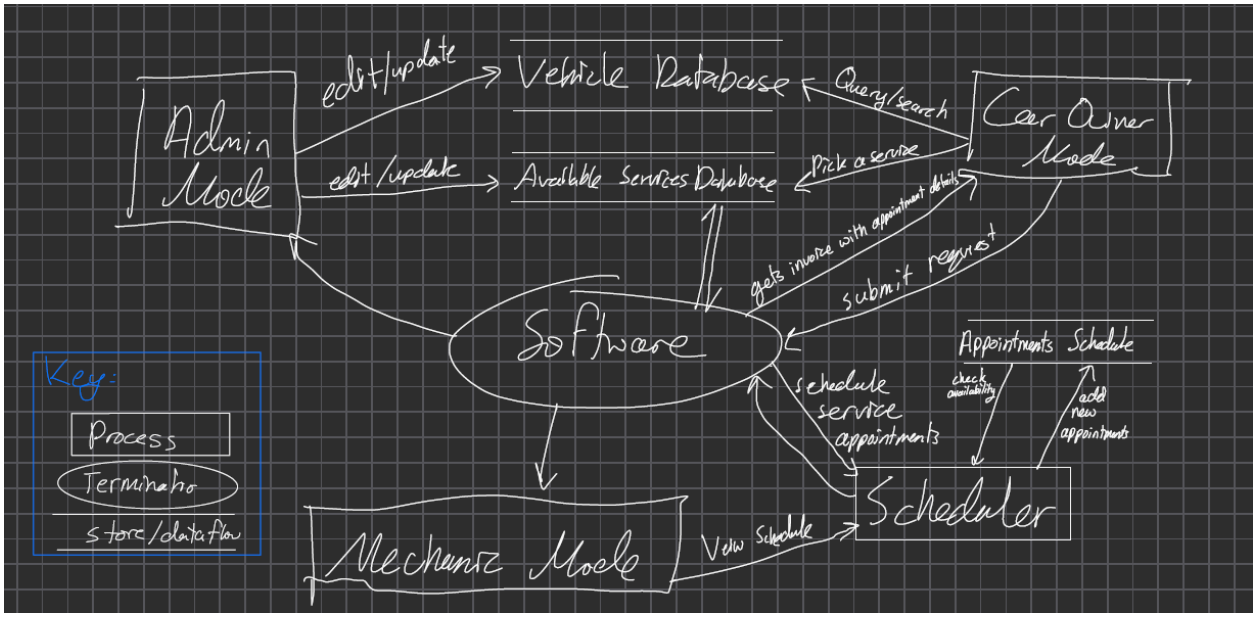
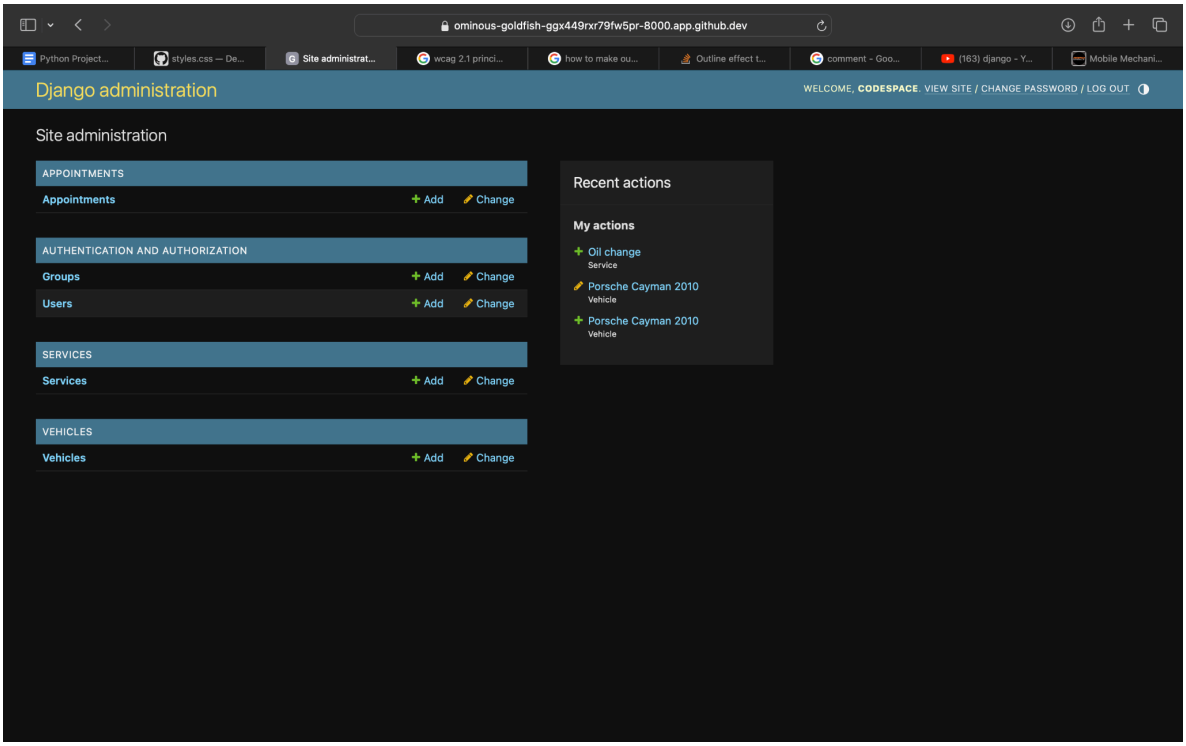


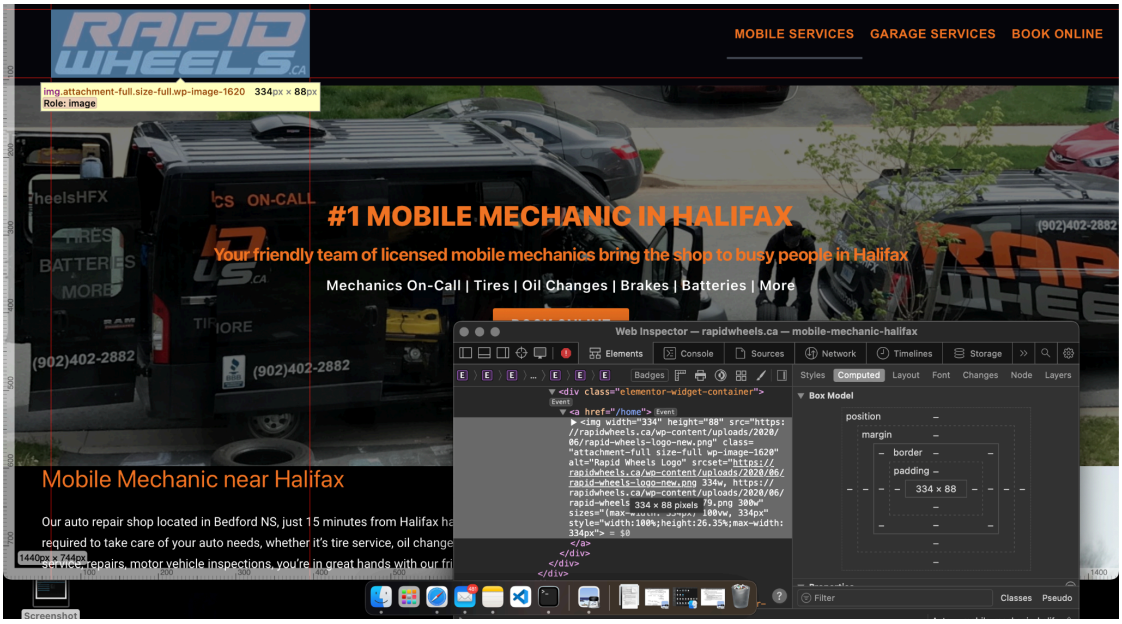
Figure 0: Software Flowchart

Implementation

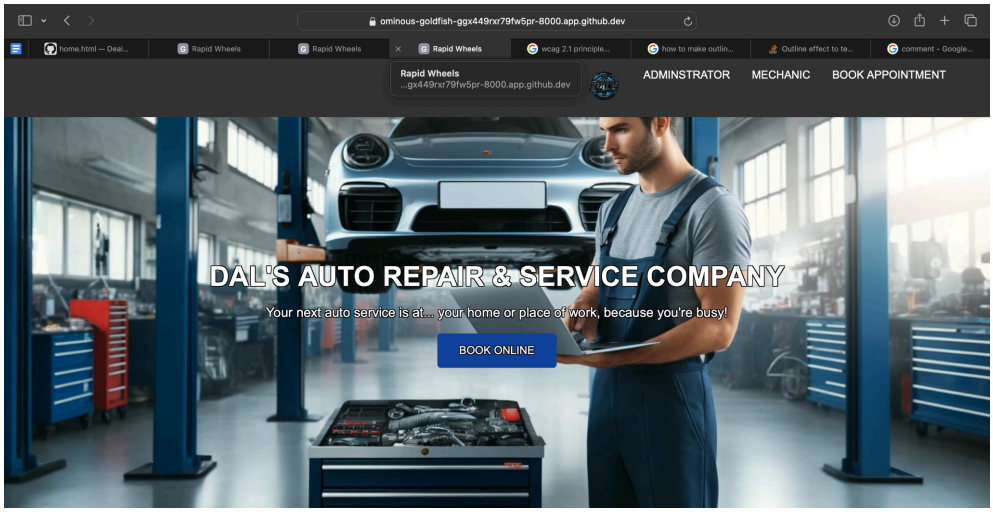
The majority of the features we had in the first version of the software were successfully implemented into the second revision (the web app). For the admin mode which was in the previous version we have opted for using the default admin mode offered by django. This was decided as we did not have enough time to implement a new admin interface due to the time constraint. The following figure will demonstrate how this webpage looks like.



We found the learning curve to be extremely steep especially when it comes to learning html and css as we have never been exposed to frontend before. To aid us we watched a couple videos and then proceeded to find a website with similar functionality to ours. We then inspected the content of the website using our web browser built in “f12” function and tried to imitate a similar interface to the website, the website used is linked below in the references section and is highlighted in the figure below.



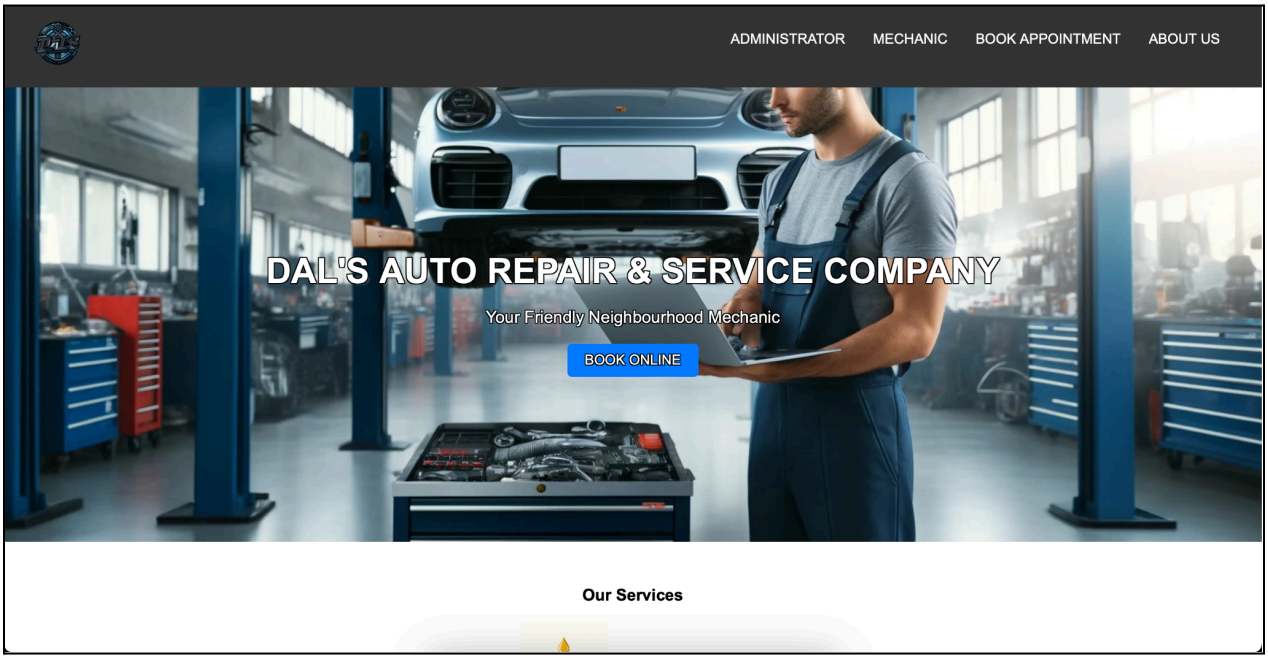
As for our home page this is what it initially looked like:



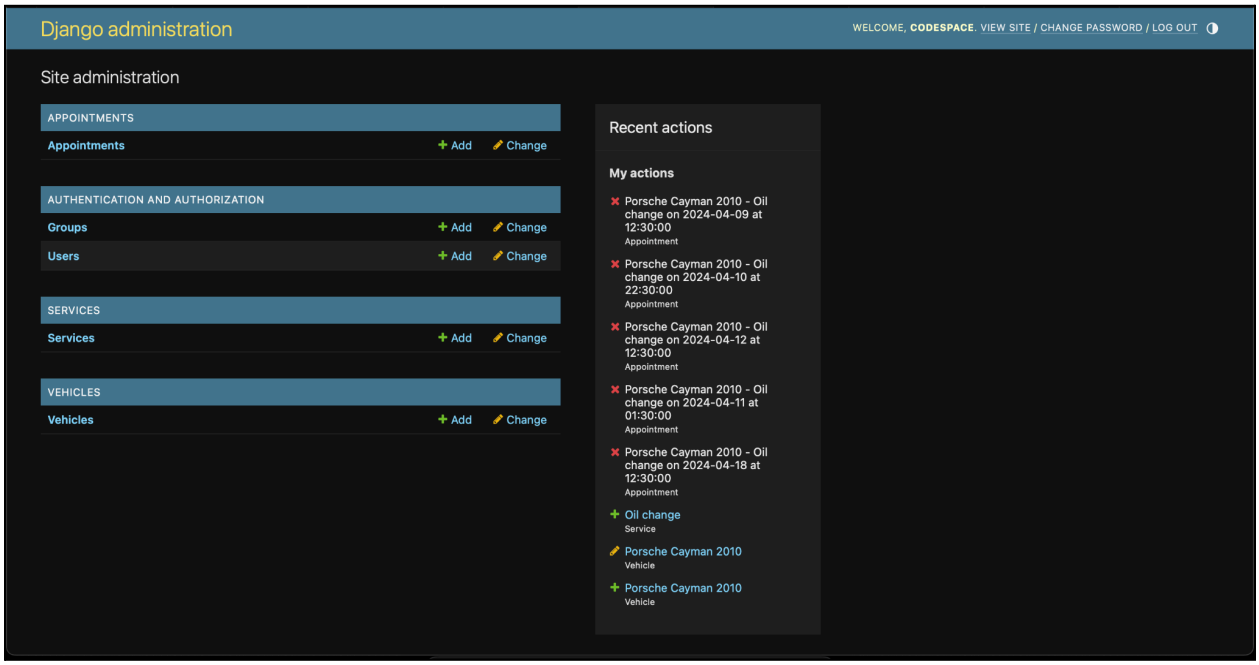
For the logo and the background image we have obtained these photos by using GPT4's DALL-E which is their image "generation" solution. Unfortunately some of the functions have had to go to accommodate for others, these functions can easily be added later on, these include but are not limited to the time and cost estimates for the services.

We have tried to try out as many new features as possible, so we decided to add a downloadable pdf that the user can generate once they have filled out the form, this can later become the receipt or the confirmation with a certain qr code to be swiftly scanned and checked in when going to the garage. We tried getting rid of the tedious tasks like filling out the object fields for services that are not exactly necessary, we just need to demonstrate it works, the object list can be updated later.

We will now present more photos of our final product:

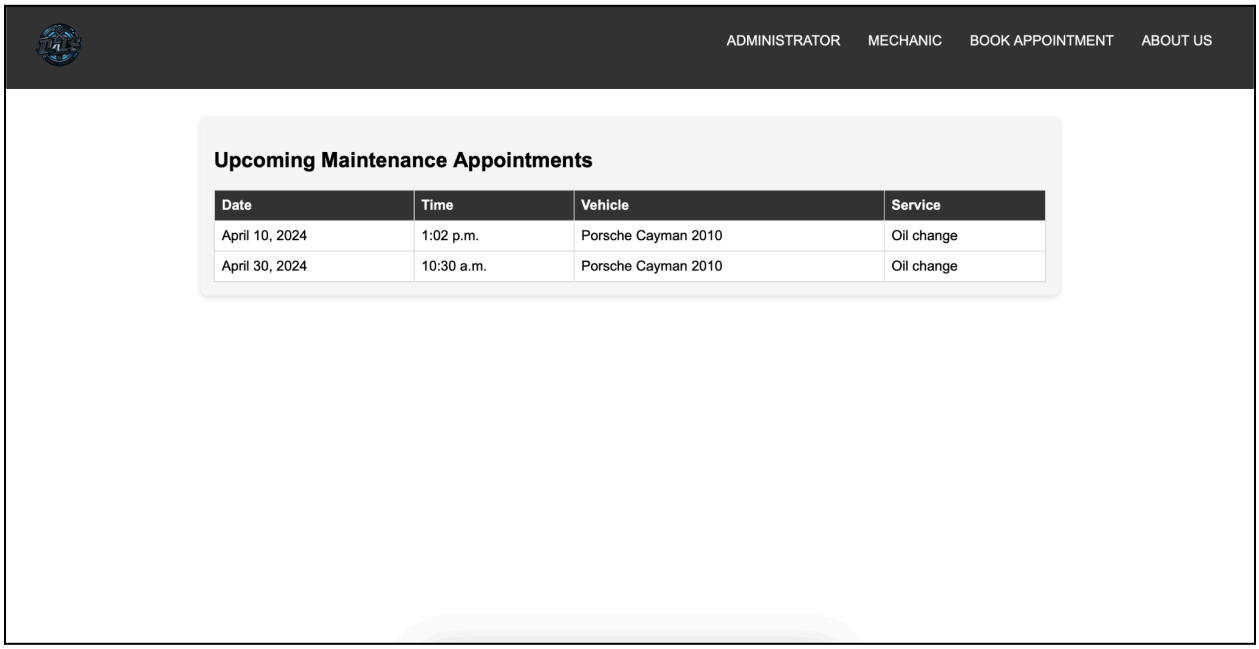


Final homepage of our web-app

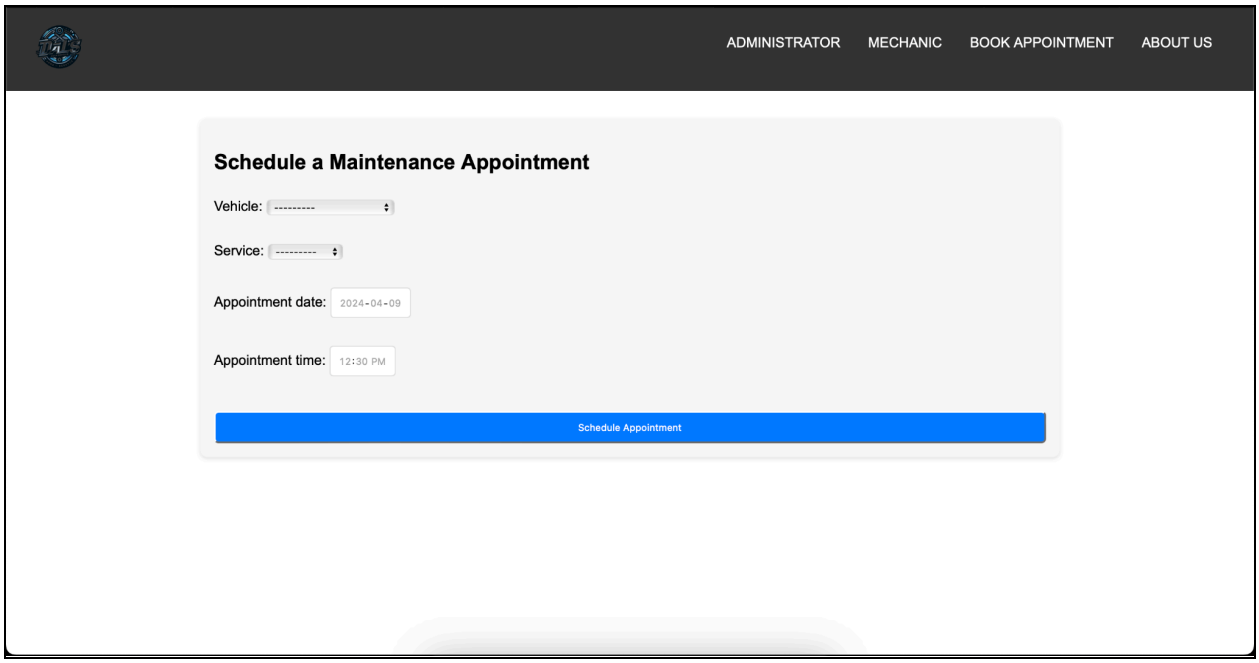


The Default Django Admin Interface

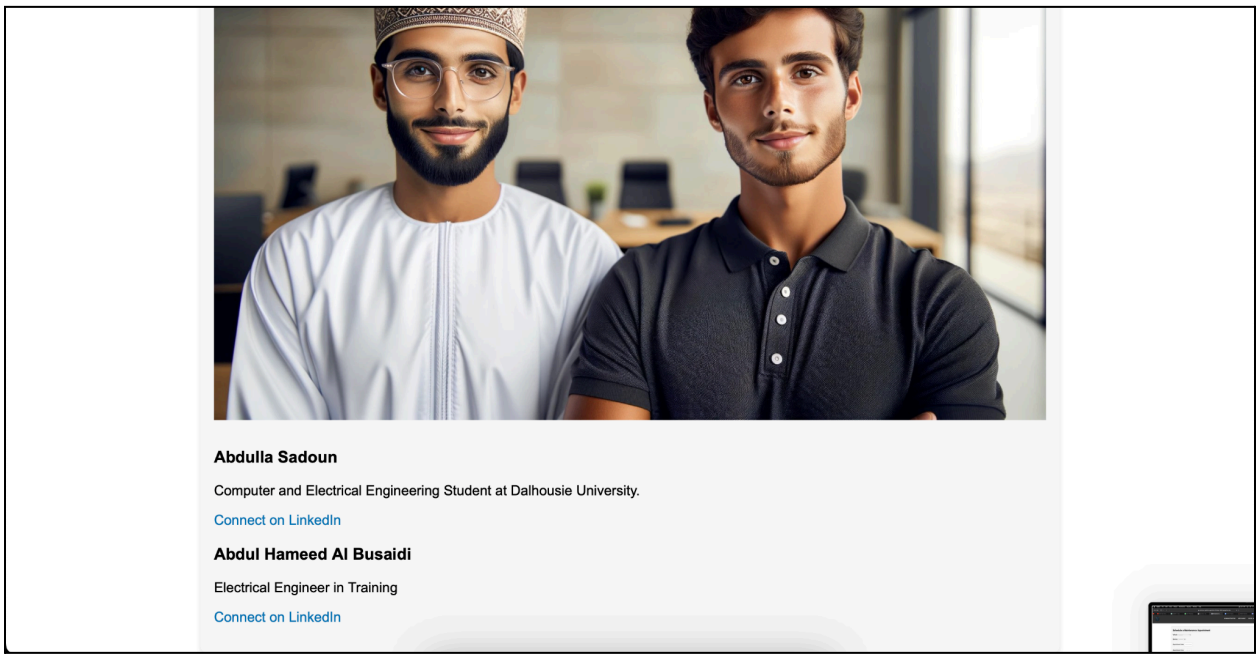
In this page, the user would be able to login as admin using the username: “codespace” and password “theadmin”. Once the user logs in they would have superuser powers as they can manipulate the content of all the contents in the database (methods).



The Mechanic’s Schedule



The Form to schedule an appointment



About Us/Contact Us page

The image included is also another DALL-E Image which is a combination of our linked in profile pictures. Or at least DALL-E's interpretation of it.

Most of the features we had in mind were successfully implemented and carried over. In an ideal scenario, the program has proven useful in saving cost and time for the dealership.

Some of the features we missed out on include: calculating and analyzing the timing and costs, proper scheduling algorithm, ability to create a username, password account to save the customer's info, update them, add them to a mailing list, ability to add multiple mechanics.

Testing/Edge Cases —

Test 1: Inputting new records

Purpose/Objective: The purpose of this test is to see whether the website saves the new records added to the vehicle table and the available services table between uses.
Test Configuration: I have run the program and went into admin mode where I have previously logged in to., I will add a new record of a vehicle model, and then I will proceed by adding a service. I will then try to see if the results are saved between runs.
Expected Results: The software should have no problems saving the new record and it should be ready for customers to use when inputting an appointment request.
Actual Results: The program passed the test and successfully added and remember the records between runs
Pass/Fail: Pass

Test 2: Duplicated Car Models

Purpose/Objective: This main objective of this test is to see how the program reacts when the user user enters the same car model twice.
Test Configuration: I have run the web app and entered the admin or superuser page, I will then add the same record twice.
Expected Results: I expect the program to save both records as we did not accommodate a feature to eliminate duplicates.
Actual Results: The program did indeed save both records and didn't try to eliminate duplicates which it should do in an ideal scenario.
Pass/Fail: Fail

Test 3: Deleting Previous Records

Purpose/Objective: The purpose of this test is to see if the deleting records functionality of the system works properly and deleted records are permanently removed from the system even between runs.
Test Configuration: I will run the program in admin mode, try deleting a record from the vehicle model and a record from the services model in our database, once done, I'll view the tables making sure they are deleted from the programs memory then I will rerun the program, go to admin mode and check the tables to see that the record specified are still deleted.
Expected Results: The program should be able to delete the records.
Actual Results: The program successfully got rid of the deleted records and managed to remember the models after the update.

Pass/Fail: Pass

Test 4: Trying to Overbook Mechanic

Purpose/Objective: The purpose of this test is to check whether the functionality of finding an available time slot for the mechanic works.
Test Configuration: I will try to book 2 different appointments on the same day and at the same time. I'll then check the schedule to see if they are assigned at different times.
Expected Results: I think this part of the code did not work properly during the implementation and might fail the test.
Actual Results: The program failed to assign different times for the service appointments and overbooked the mechanic
Pass/Fail: Fail

Contribution

Our group has continued to utilize the same strategy to tackle this task, we have been learning the content together, implementing the code and various design elements together. We have had numerous brainstorming sessions to decide on many of the little things in the web app especially when it comes to the overall scheme and frontend of the application. But I think we have reached a pretty good conclusion given the time constraint we were put in. We have continued to use our in-person collaborative approach as we have in the previous projects which has proven itself better suited to our group. We have also followed that approach for the creation of this document and for the video presentation/demo.

Conclusion and Reflections

In conclusion, this project was successful, but due to shortage of time and high course load at the end of our academic year,, we were not able to implement all the features we originally planned on having in the project. We also couldn't optimize the code and make it more readable or to follow standard principles or regular style.. This can be overcome next time with better planning, and with utilization methods like agile scrum.

Future development of the project may be done if the group members decided on exploring the potential of this idea, and a plethora of features was originally intended to make it to the final project still on the roadmap . Although in the meantime, no future development is confirmed and the group will halt development after the project submission.

In the future, More of the backend database methods will be fixed, like creating time, cost estimates, eliminating duplicates, and accommodating multiple mechanics. Additionally it would be nice to have account registration for each type of user as well as better UI.

References

- Solution Used for Comparison: https://ari.app/demo/
- For learning django usage:
    - https://youtu.be/cUaZ0ElJ760?si=dixWme6xjC8lIdGk
    - https://youtu.be/nGIg40xs9e4?si=-oYhVKLDQNfgnJT8
- Frontend inspiration: https://rapidwheels.ca/mobile-mechanic-halifax/
- IDE used for development: Github Codespace

LINK TO OUR REPOSITORY: https://github.com/AbdullaSadoun/DealerMaintenance