Yuma Vehicle Manager, Demonstration Application

Student ID: 730096317

Python Version:

To run this application, please use Python version 3.10+. The recommended version for this application is version 3.12.2.

Start the application by running the file `'main.py'` from the top-level folder `'sysdev2'`. In the terminal, this location can be described as `'{usersLocalFileSystem}/sysdev2/main.py'`

Note: On macOS, there is a known issue in Tkinter where text fails to load on older versions of Python 3. If you start the application and the text is not loading correctly, please use the recommended version of Python.

Installing libraries:

This application makes use of Tkinter (For GUI), Flake8 (Linting) and Pytest (Unit testing). The final two can be installed through pip by issuing this command from a terminal: `pip install flake8 pytest pytest-mock`

Note: Since Tkinter is not available through pip, please ensure it is correctly installed in your python installation

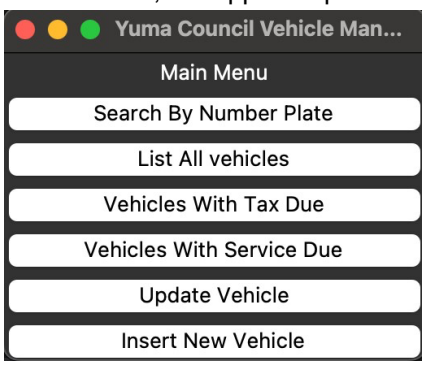
Starting the App:

To start the application, run the file `main.py`. This will reset the DB and then open the application.

Note: If you do not wish for the database to be reset to its demo state before starting the app, you can easily remove the code by editing the main.py file - The code is surrounded by the comment 'Reset DB'

Main Window:

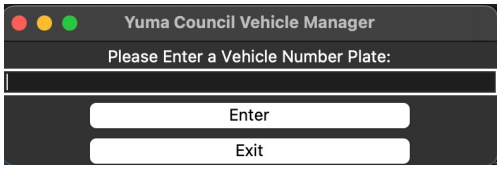
Once started, the app will open its main window:



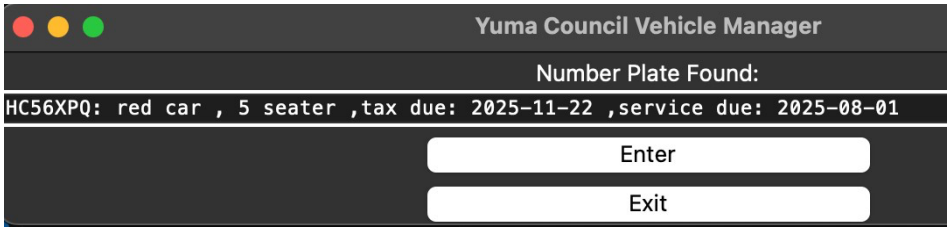
This contains all the current options offered by the vehicle manager app.

Search By Number Plate:

This opens a window that allows you to get a vehicle's details from its number plate
First, you are prompted to enter the number plate of the vehicle:

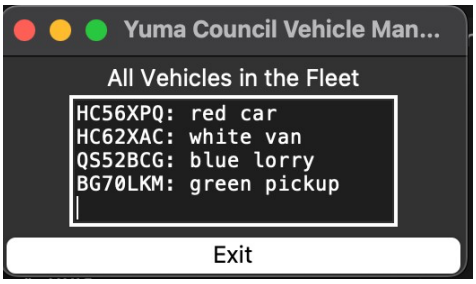


Then, if a valid number plate is entered, the information of that vehicle is displayed:



List All Vehicles:

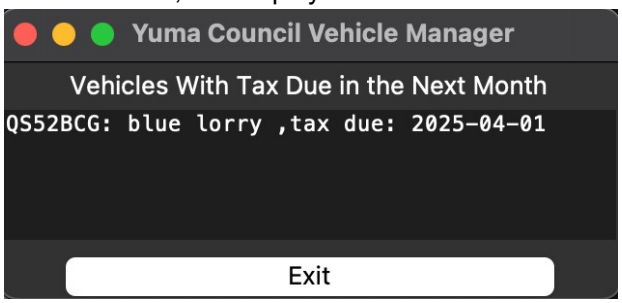
When selected, this displays all the vehicles currently in the database:



Note: You can scroll to see more vehicles if more than five are in the database

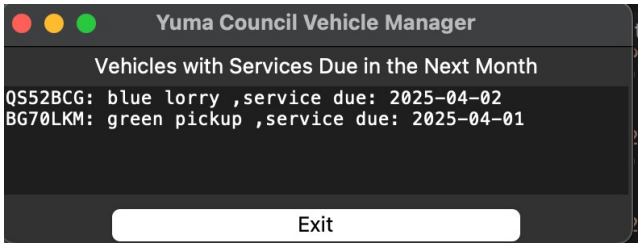
Vehicles With Tax Due:

When selected, this displays all vehicles with their tax due in the next month:



Vehicles With Service Due:

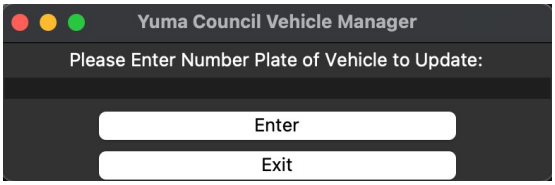
When selected, this displays all vehicles with a service due in the next month:



Update Vehicle:

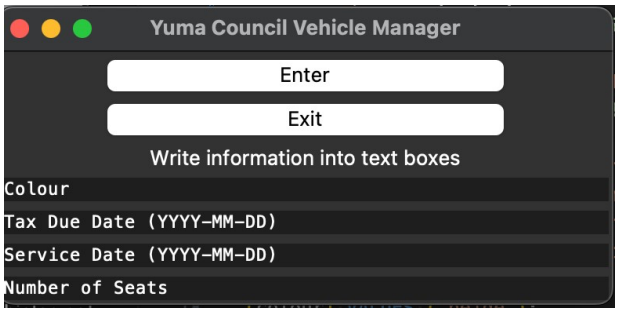
When selected, this will allow you to update some of the values of a vehicle in the database.

Firstly, you will be prompted to enter the number plate of the vehicle you wish to update:



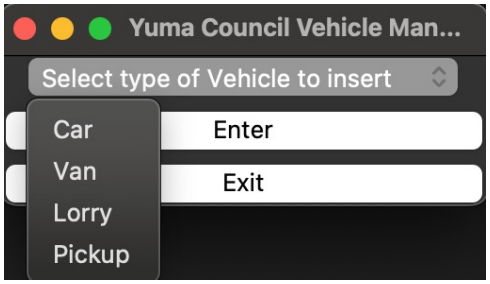
Once a valid number plate has been entered, you can update the vehicle's values by entering the new values in the given text fields.

Note: Only the text fields with valid inputs will be updated. For example, if you enter one valid and one invalid input, only the valid input will be changed in the database. You must also enter at least one valid input for an update—more info on valid inputs is in the section below.



Insert New Vehicle:

This allows you to insert a new vehicle into the database. You are first asked for the type of vehicle you wish to insert:



Then you can add the information for the vehicle you are inserting, clicking enter when done.
Note: All fields must be filled with a valid input, as described below.

Valid Inputs: For specific inputs to be valid, they must meet specific criteria.

- Number plate: must follow the 'LLNNLLL' Format, where L is a Letter and N is a number
- Colour: Must be one of the nineteen valid colours as provided by the UK's DVLA; these colours can be found in the valid_inputs.py file
- Tax Due Date/ Service Date: Must be valid dates following the YYYY-MM-DD format. Must also occur in the future since a new vehicle being entered cannot have its tax or service due in the past
- Cargo Capacity: Must be an integer, which is the vehicle's cargo capacity in litres
- Cab type: For pickups, it can either be 'single' or 'double', or for lorries, it can be 'day' or 'sleeper'

Tests:

This project contains unit tests for the application; the source code can be found in the 'src/tests' directory.
To use Pytest to run the unit tests, use this command:

```
pytest src/tests/test_run_sql.py
```

You can also get a coverage report for the tests by issuing this command:

`coverage run --source=src -m pytest -v src/tests && coverage report -m`

Note: Flake8 linting has been disabled for the tests directory since it contains counterintuitive warnings to unit testing. For example, using the `assert` keyword in Python is crucial in unit testing but would cause the warning `Use of assert detected. The enclosed code will be removed when compiling to optimised byte code. Flake8(S101)` In flake8.

If you wish to enable flake8 linting on this directory, please remove or edit the `.flake8` file.