Personal Portfolio

Group Number: Team 92 – Blink 192
Name: Taylor Bindon
Student Number: N9296506
GitHub: taylorbindon

Artefact 1 – Logical and Physical Diagrams

Description

The logical and physical diagrams were prepared prior to implementing the Django framework. The logical diagram illustrates how the web framework will operate and provide content to users. The physical diagram depicts how the system will be physically implemented. The three tiered architecture system has been implemented for this application as there are components that rely on user facing processes only, mid layer operations that do not connect to the database, and full system utilisation where database queries are run. Therefore, some pages require different levels of access to different data, this framework helps illustrate and organise these processes.

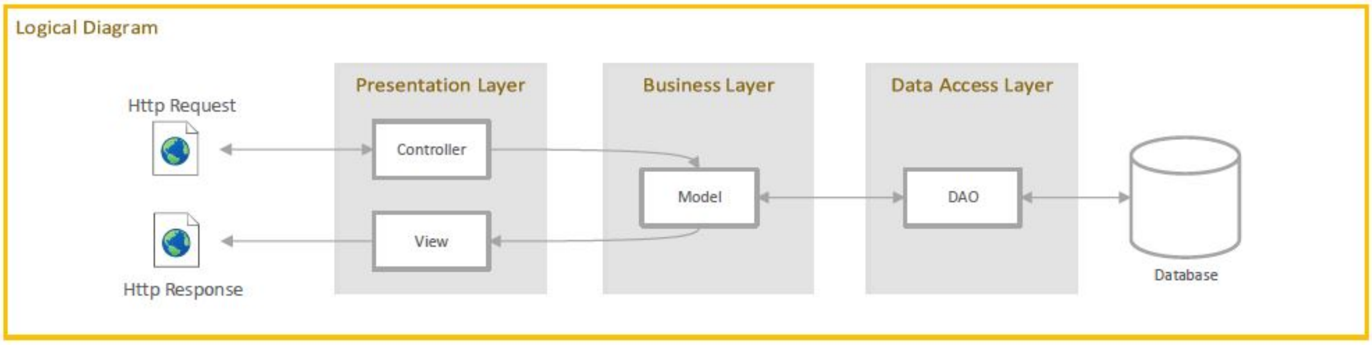
How It Contributed to the Project

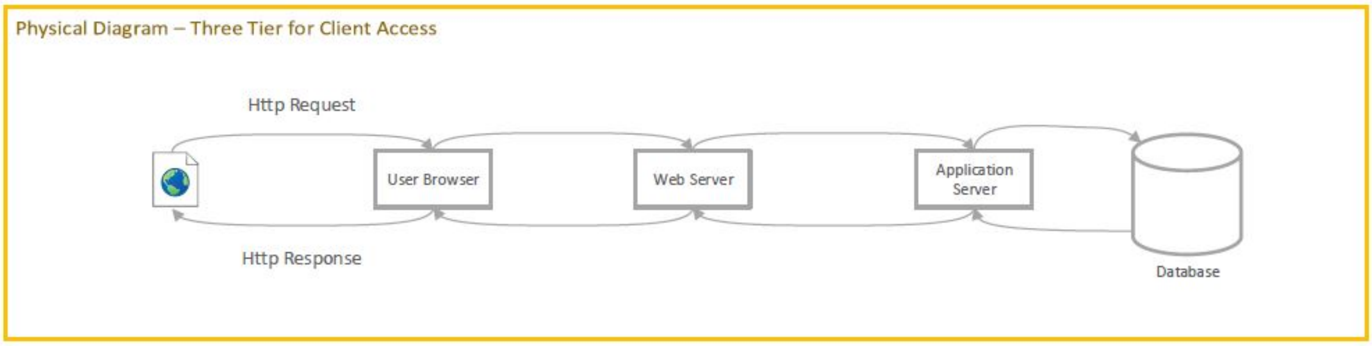
The logical diagram helped provide the group and client with a deeper understand of how the web framework will operate. The physical diagram further aided this depiction by separating physical sources of data into their respective systems. Instead of referring to the systems as layers, treating them as physical objects helped clarify the system architecture for various members of the group and for the client.

The three tiered architecture system enables users to receive information faster, for example users that are navigating pages do not require database queries, thus allowing the deliverance of user facing processes to be quicker. Furthermore, accessing mid layer business logic and database queries only when required will provide greater system efficiency and make the app feel far more user friendly, thus enhancing user experience.

Where It Is Used

The chart is primarily used to provide the group with an understanding of the system architecture.





Artefact 2 – Database Setup (Shared)

Description

The database setup work was shared between myself and Jeremy Barnes. Jeremy established the database tables and populated the database, whilst I helped filter the data to enable it to be used in the database. From these steps, a database schema was created and I connected it to our Django web framework.

How It Contributed to the Project

By having the database setup and connected to our web application, the group was enabled to test and use the database to manipulate data and experiment with how reports will be generated for the client. This artefact is crucial to the project, as without the database established and connected to the framework, data could not be retrieved to produce reports for the client. Furthermore, the environment was setup so that all the other group members needed to do was install the relevant software, mysqlclient etc., through pip install commands.

Where It Is Used

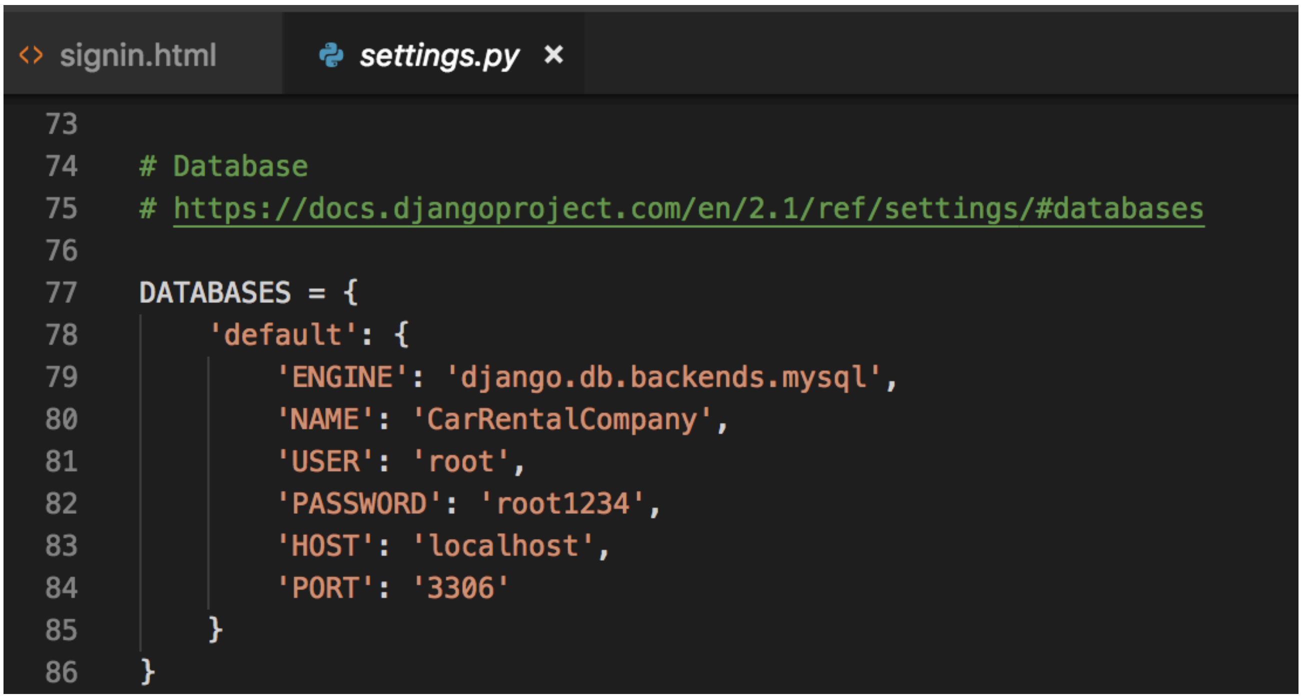
Github Commit – taylorbindon/IFB299/Database Connection

- This commit included code for the virtual environment and the connection to the unpopulated database.

Github Commit – taylorbindon/IFB299/Database Population

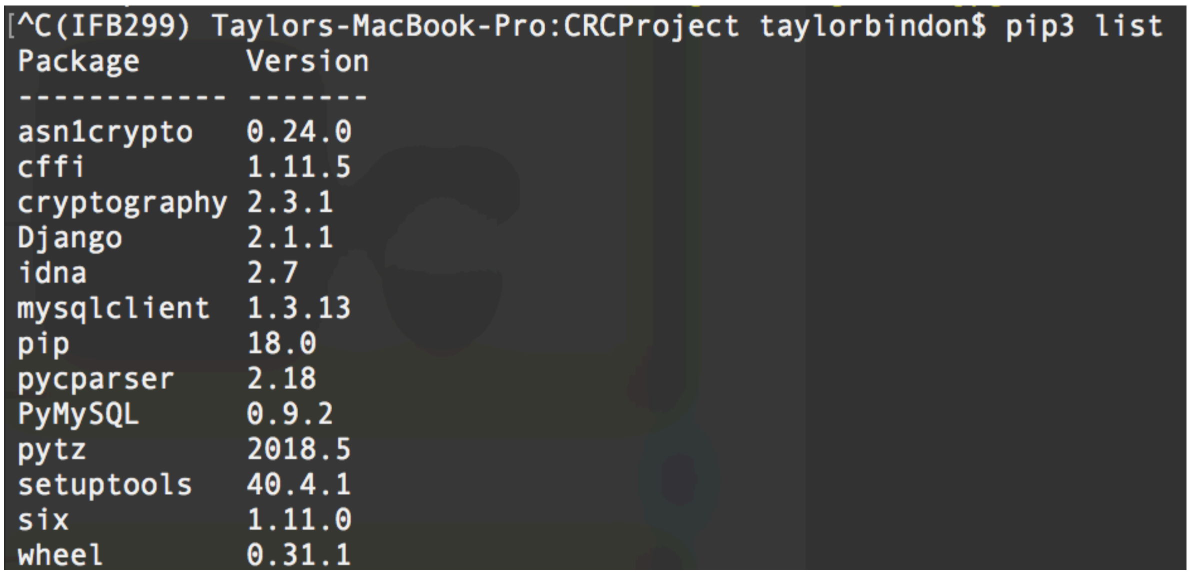
- This commit included the fully populated and connected database.

IFB299/Application/CRCProject/CRC/settings.py code that illustrated the connection details that connect the application to the database.



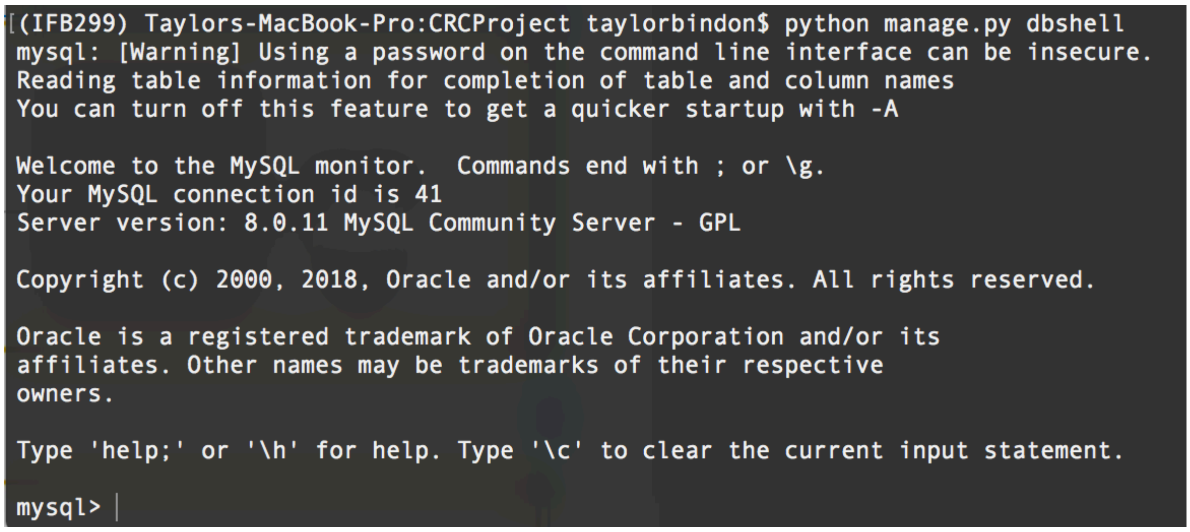
```
# Database
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'CarRentalCompany',
        'USER': 'root',
        'PASSWORD': 'root1234',
        'HOST': 'localhost',
        'PORT': '3306'
    }
}
```

pip3 list command that shows the software that was installed. Difficulties were encountered when installing mysqlclient, easily fixed in Mac Terminal on my local machine with a brew command, that took much longer to configure on other group members' local machines.



```
[^C(IFB299) Taylors-MacBook-Pro:CRCProject taylorbindon$ pip3 list
 Package      Version
 ------------ -------
 asn1crypto   0.24.0
 cffi         1.11.5
 cryptography 2.3.1
 Django       2.1.1
 idna         2.7
 mysqlclient  1.3.13
 pip          18.0
 pycparser    2.18
 PyMySQL      0.9.2
 pytz         2018.5
 setuptools   40.4.1
 six          1.11.0
 wheel        0.31.1
```

Using manage.py to run the database shell shows that mysqlclient works and can interface with the database.



```
[(IFB299) Taylors-MacBook-Pro:CRCProject taylorbindon$ python manage.py dbshell
 mysql: [Warning] Using a password on the command line interface can be insecure.
 Reading table information for completion of table and column names
 You can turn off this feature to get a quicker startup with -A

 Welcome to the MySQL monitor.  Commands end with ; or \g.
 Your MySQL connection id is 41
 Server version: 8.0.11 MySQL Community Server - GPL

 Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

 Oracle is a registered trademark of Oracle Corporation and/or its
 affiliates. Other names may be trademarks of their respective
 owners.

 Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

 mysql>
```

Artefact 3 – Django Environment

Description

In this artefact, I was responsible for creating the Github repository, implementing the skeleton web framework, installing software and enabling the virtual development environment, connecting views and pages together, and connecting the database.

As only one group member, Jeremy, had used Github prior to commencing this subject, I tried to teach and guide other members how to use the software. Furthermore, as all of us were fairly unfamiliar with web frameworks, I took on the responsibility of learning about how web frameworks operate and how they should be implemented. This is reflected in the Github commit history where the commits I delivered related to the framework implementation, some minor page setup testing, and database connection.

How It Contributed to the Project

By establishing a consistent development environment, the team was enabled to create individual html pages, css style sheets, bootstrap implementation, and add static files to the framework. This also enabled html pages to be linked together to create a smooth, user friendly, web application. Furthermore, without the framework established, the database could not be connected or tested to ensure its operation.

As I was the only team member learning about and implementing the framework, other group members completed html pages and other artefacts outside of the environment. Unfortunately there is a gap in knowledge amongst the group pertaining to web frameworks and Github. Understandably, it is unlikely that group members have been exposed to or used web frameworks before. However, Github is not hard to learn and the process of pushing and pulling is not onerous. The intent is for the html pages to be consolidated into the framework, ideally by individual members should they learn about the framework and how to use Github.

Going forward, it will be presented in a meeting that everybody *must* be well acquainted with Github and Django in order for the project to move ahead.

Where It Is Used

Github Commit – taylorbindon/IFB299/Skeleton Framework (Draft)

- This commit was the initial implementation of the framework for all group members to look through and understand how Django operates.

Github-Commit – taylorbindon/IFB299/Database Connection

- This commit included additional files to implement test html pages so that individually produced html pages can be consolidated into the framework.

Artefact 4 – Database Testing

Description

Example queries were used to test the connection to the database, and the database information itself, in MySQL and Django. Testcases are yet to be implemented as the group needs to first understand the web framework itself.

How It Contributed to the Project

These queries verified that the database was connected and could be accessed locally. This means that the database will be operational and usable by the client to generate reports from the data stored in the database.

Where It Is Used

Github Commit – taylorbindon/IFB299/Import Models from Database

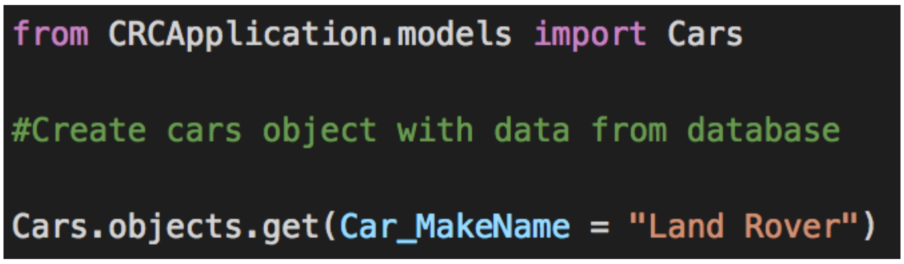
- This commit shows the models that were generated from the database in preparation for being used to query the database.
- Specifically IFB299/Application/CRCProject/CRCApplication/models.py

An SQL query will be used by all of the search functions on different pages. These queries will operate similar to the example below, however, they will use variables as entered by the user instead of a hardcoded query.

SQL Query

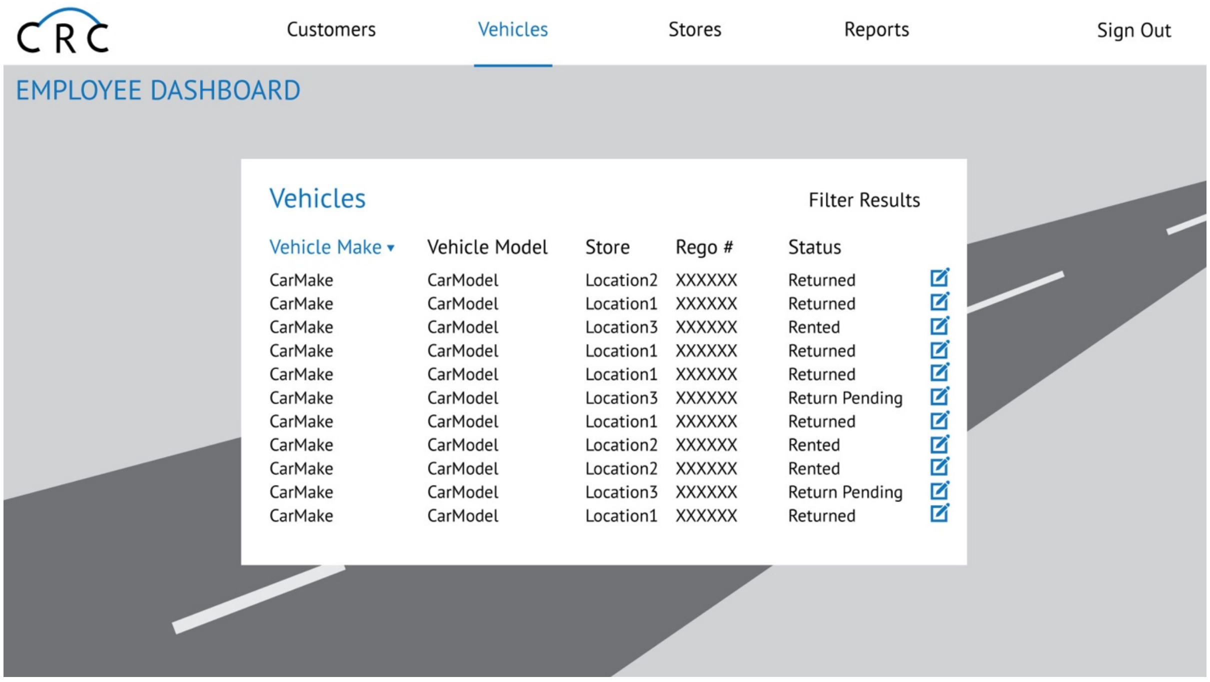
```
SELECT * FROM CarRentalCompany.cars
Where CarRentalCompany.cars.Car_MakeName = "Land Rover";
```

Django Equivalent



```
from CRCApplication.models import Cars

#Create cars object with data from database

Cars.objects.get(Car_MakeName = "Land Rover")
```

These queries return entries in the 'Cars' table that have the 'Car_MakeName' field as 'Land Rover'. This is an example of how the queries will return information in the model layer, to be presented to the user as a report in the following format:



Vehicle Make ▾	Vehicle Model	Store	Rego #	Status	
CarMake	CarModel	Location2	XXXXXX	Returned	✎
CarMake	CarModel	Location1	XXXXXX	Returned	✎
CarMake	CarModel	Location3	XXXXXX	Rented	✎
CarMake	CarModel	Location1	XXXXXX	Returned	✎
CarMake	CarModel	Location1	XXXXXX	Returned	✎
CarMake	CarModel	Location3	XXXXXX	Return Pending	✎
CarMake	CarModel	Location1	XXXXXX	Returned	✎
CarMake	CarModel	Location2	XXXXXX	Rented	✎
CarMake	CarModel	Location2	XXXXXX	Rented	✎
CarMake	CarModel	Location3	XXXXXX	Return Pending	✎
CarMake	CarModel	Location1	XXXXXX	Returned	✎

Artefact 5 – Class Diagram (Shared)

Description

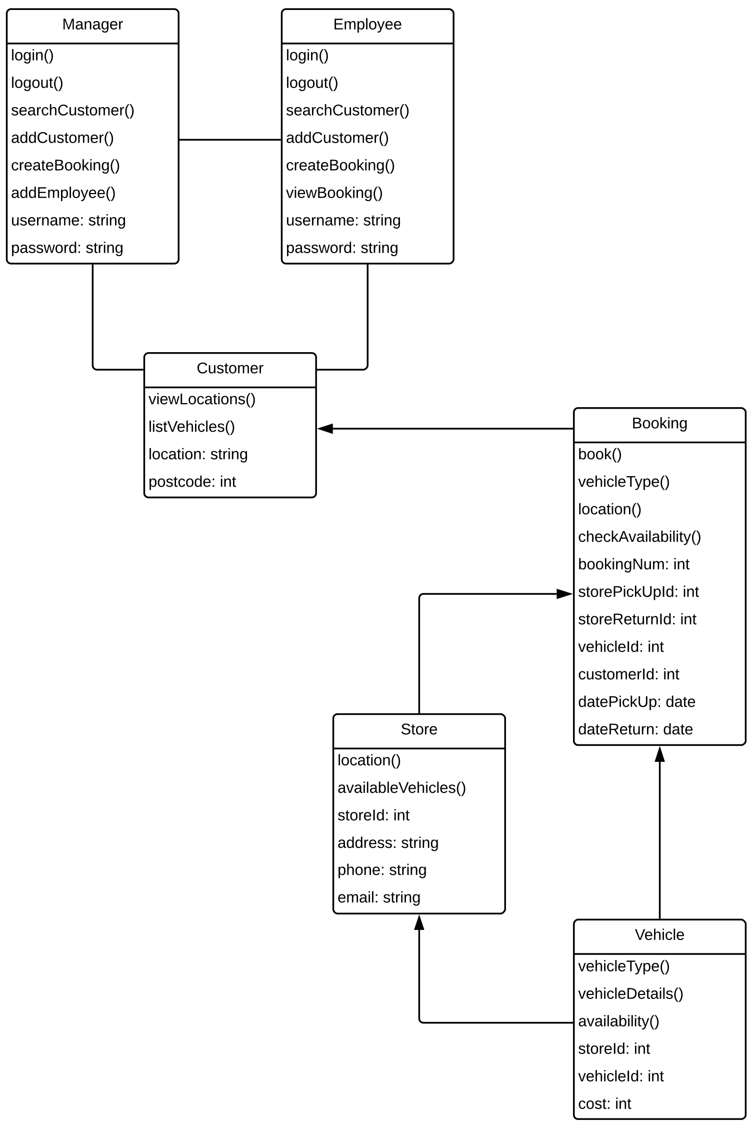
For this artefact, Jeremy created the initial mockup of the class diagram and I implemented the UML styling and relations between classes.

How It Contributed to the Project

This diagram provides the group a high level perspective of the necessary functionality, and functions to be implemented, for the application. This diagram serves as a guide as to what data will be need, and how it should be manipulated, in order for the application to run as per the client's requests.

Where It Is Used

The diagram is mainly used as an overview to help guide developers with how the data should be treated in the web framework itself.



Manager
login()
logout()
searchCustomer()
addCustomer()
createBooking()
addEmployee()
username: string
password: string
Employee
login()
logout()
searchCustomer()
addCustomer()
createBooking()
viewBooking()
username: string
password: string
Customer
viewLocations()
listVehicles()
location: string
postcode: int
Booking
book()
vehicleType()
location()
checkAvailability()
bookingNum: int
storePickUpId: int
storeReturnId: int
vehicleId: int
customerId: int
datePickUp: date
dateReturn: date
Store
location()
availableVehicles()
storeId: int
address: string
phone: string
email: string
Vehicle
vehicleType()
vehicleDetails()
availability()
storeId: int
vehicleId: int
cost: int