Our group has chosen to implement a back end system and virtual users in order to further secure our created website, for the Road Traffic Authority (RTA). Our website can allow users to apply for a license, apply to renew their license, register a vehicle, pay fines, view their merit points and list a vehicle as destroyed. Additionally this website will also work for those who register as a Road Safety Officer which can approve or reject license renewals, issue vehicles with fines and demerits, approve or reject license applications and issue members of the public with a license number, approve payments and revoke licenses and approve or flag for investigation on the sale or destruction of a vehicle.

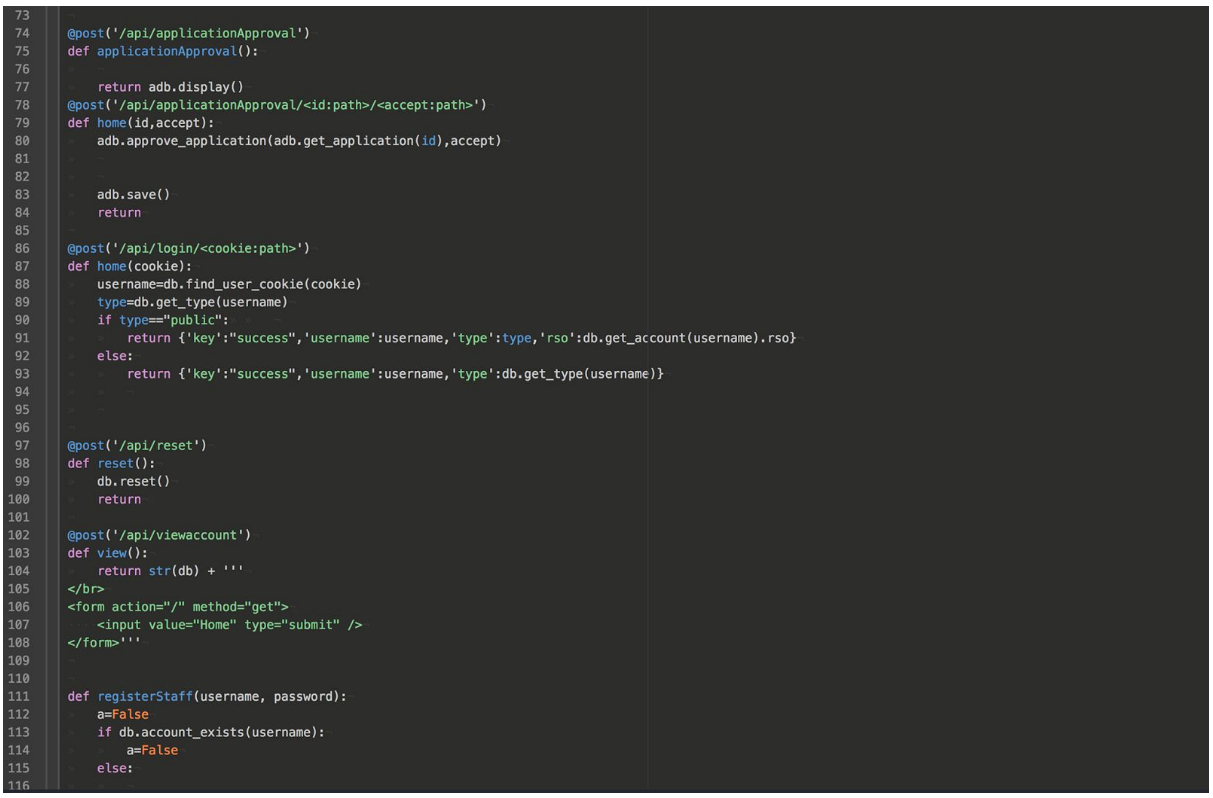
The purpose of a back-end database is to allow the website to store information in a format so that information can be readily retrieved and used when necessary. A database is a structured set of information which is held separate to the front end of the site. The addition of a back-end database adds an additional layer of security. In our website, the database is stored on the back-end, on a separate port to our front-end website. By doing this, someone who wants to access the data (users/staff/admin) need to validate their identity by entering their user details and password, where the website then accesses this information from the back end. In this way, data will only be accessed by logged in users, which is done in the form of sending requests for information, which can be confirmed or denied based on the role of the user.

Our back-end takes an object-oriented approach by storing user information as python objects. As a user opens the website (at port 8080) and submits their user info inputs their username and password, a request is sent to the back-end (port 8081). Once they have been logged in, this line of communication between the user and the website operations remains connected.

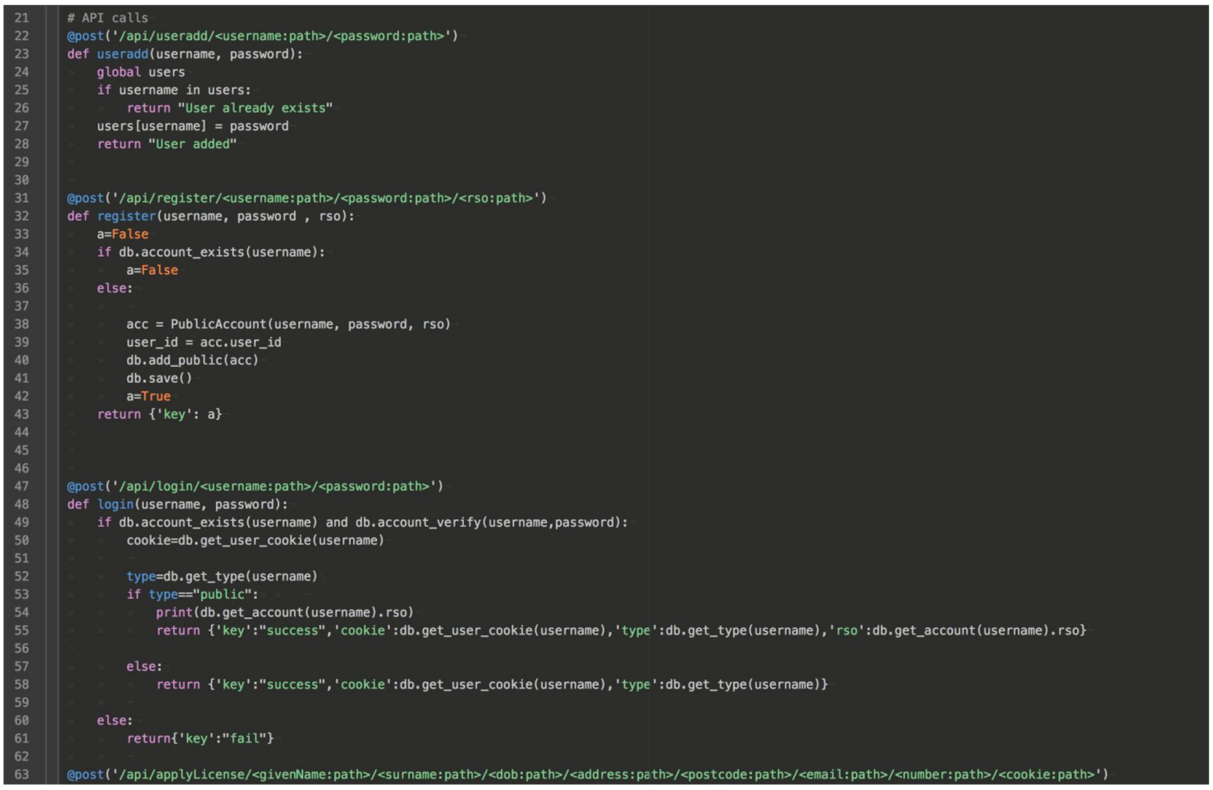
If a user is logged in, they will continue operating on the front-end, with the ability to send requests to the back end. For example, if the user is attempting to apply for licence, then they will need to fill out a form, which is on the front end. Then, this information is sent to be stored in the back end of the database.

Our backend database consists of account information (such as username, password, e-mail), application information (name, address etc.) and vehicle information (registration, type, address etc.) Additionally, the back-end of our website contains the methods which the users of our website need. So for example, if a Road Safety Officer wants to approve a licence, they will import from the licence application from the database, approve/deny the application and then the state of the application is stored in the back of the database.

In order for the back end database system to work, ensure that the run.py file and the back end.py file are running at the same time which will demonstrate how our back end system works.

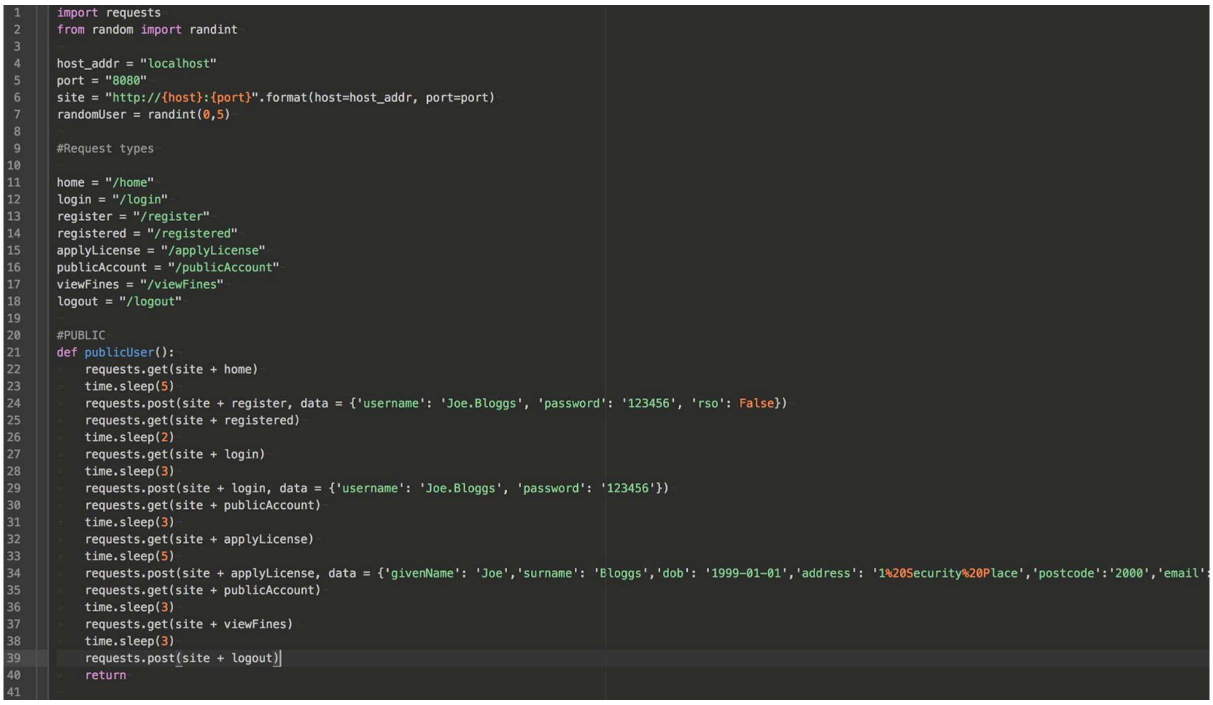


```python

@post('/api/applicationApproval')
def applicationApproval():

    return adb.display()
@post('/api/applicationApproval/<id:path>/<accept:path>')
def home(id,accept):
    adb.approve_application(adb.get_application(id),accept)



    adb.save()
    return

@post('/api/login/<cookie:path>')
def home(cookie):
    username=db.find_user_cookie(cookie)
    type=db.get_type(username)
    if type=="public":
        return {'key':"success",'username':username,'type':type,'rso':db.get_account(username).rso}
    else:
        return {'key':"success",'username':username,'type':db.get_type(username)}



@post('/api/reset')
def reset():
    db.reset()
    return

@post('/api/viewaccount')
def view():
    return str(db) + '''
</br>
<form action="/" method="get">
    <input value="Home" type="submit" />
</form>'''


def registerStaff(username, password):
    a=False
    if db.account_exists(username):
        a=False
    else:
```



```python
# API calls
@post('/api/useradd/<username:path>/<password:path>')
def useradd(username, password):
    global users
    if username in users:
        return "User already exists"
    users[username] = password
    return "User added"


@post('/api/register/<username:path>/<password:path>/<rso:path>')
def register(username, password , rso):
    a=False
    if db.account_exists(username):
        a=False
    else:

        acc = PublicAccount(username, password, rso)
        user_id = acc.user_id
        db.add_public(acc)
        db.save()
        a=True
    return {'key': a}



@post('/api/login/<username:path>/<password:path>')
def login(username, password):
    if db.account_exists(username) and db.account_verify(username,password):
        cookie=db.get_user_cookie(username)

        type=db.get_type(username)
        if type=="public":
            print(db.get_account(username).rso)
            return {'key':"success",'cookie':db.get_user_cookie(username),'type':db.get_type(username),'rso':db.get_account(username).rso}

        else:
            return {'key':"success",'cookie':db.get_user_cookie(username),'type':db.get_type(username)}

    else:
        return{'key':"fail"}

@post('/api/applyLicense/<givenName:path>/<surname:path>/<dob:path>/<address:path>/<postcode:path>/<email:path>/<number:path>/<cookie:path>')
```

Virtual users are widespread in today’s cyber world which may make it hard to distinguish from real users, as they conduct the exact same activities. We have implemented the use of virtual users in order to secure our website from unwanted bots. It works through using randint to pick a user and then by using requests to obtain get and post requests to direct the chosen user. It works through incorporating an if statement that depending on the number chosen, it chooses a random user from 6 and runs the code through various tasks.



```python
import requests
from random import randint

host_addr = "localhost"
port = "8080"
site = "http://{host}:{port}".format(host=host_addr, port=port)
randomUser = randint(0,5)

#Request types

home = "/home"
login = "/login"
register = "/register"
registered = "/registered"
applyLicense = "/applyLicense"
publicAccount = "/publicAccount"
viewFines = "/viewFines"
logout = "/logout"

#PUBLIC
def publicUser():
    requests.get(site + home)
    time.sleep(5)
    requests.post(site + register, data = {'username': 'Joe.Bloggs', 'password': '123456', 'rso': False})
    requests.get(site + registered)
    time.sleep(2)
    requests.get(site + login)
    time.sleep(3)
    requests.post(site + login, data = {'username': 'Joe.Bloggs', 'password': '123456'})
    requests.get(site + publicAccount)
    time.sleep(3)
    requests.get(site + applyLicense)
    time.sleep(5)
    requests.post(site + applyLicense, data = {'givenName': 'Joe','surname': 'Bloggs','dob': '1999-01-01','address': '1%20Security%20Place','postcode':'2000','email':
    requests.get(site + publicAccount)
    time.sleep(3)
    requests.get(site + viewFines)
    time.sleep(3)
    requests.post(site + logout)
    return

```

The virtual users re-enact the types of users that are applicable to our website. Two of the users are members of the public who are applying for licenses and listing vehicles for sale. Moreover, another two virtual users are acting as Road Safety Officers who approve applications and checks for payments of fines. Finally, an administrator virtual user has been created which can perform a reset of the database of our website and also view other existing accounts.

The additional security features that have been added from our initial draft website have improved the security and functionality of our website. By the inclusion of a backend on a separate port, an additional layer of security is provided to the website information as users must be logged on in order to send requests and retrieve information. The addition of the virtual users has navigation scenarios for different user types, showcasing the adaptability and capabilities of simple bots in today’s online world.