IFB299 IT Project Design and Development Personal Portfolio

(Team No. 70)

(Kaushal Kishorbhai Limbasiya – N10032029)

(GitHub: IFB299-Group-70-S.K.R.A.M)

Artefact 1 – (Logical Software Diagram of Car Rental Company)

In considering specific requirements from Car Rental Company, the logical software diagram implemented for this scenario will be of three-tier pattern. More specifically, the board of CRC expects the systems to provide few functions namely, to browse the numbers of different cars that are picked up or returned in some stored and car recommendations to customers. The above requirements can be fulfilled by utilising a separate business logic (here CRC Business Logic) which will be connected with web interface (here CRC Web Interface) and database (here CRC Database). As a result, the pattern of software architecture of CRC is three-tier. The following image illustrates the flow of Logical Software Diagram of Car Rental Company (CRC).

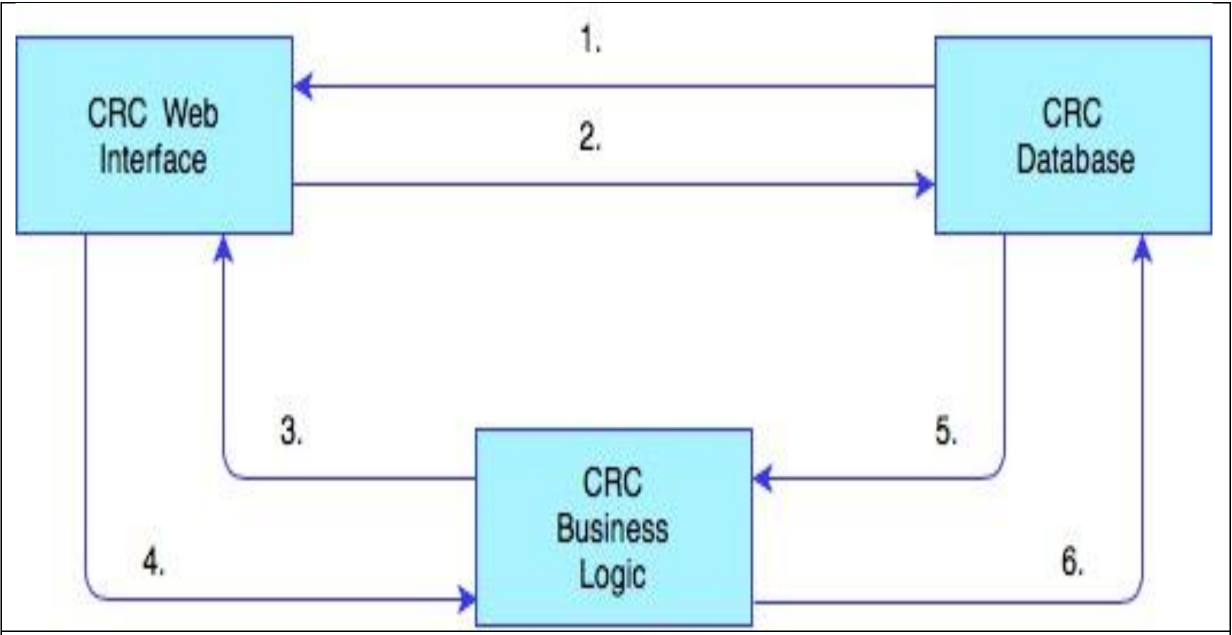


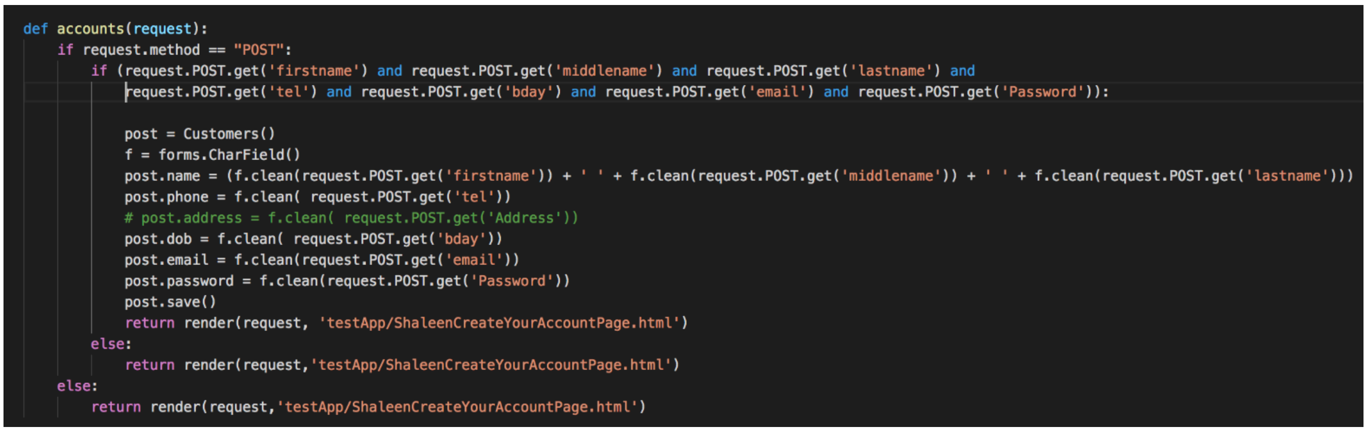
Figure 1. Logical Software Diagram of Car Rental Company (CRC)

The Logical Software Diagram of CRC have various flows (here numerical represented as 1, 2, 3, 4, 5 and 6) within CRC Web Interface, CRC Database, CRC Business Logic.

More specifically, the flow 1 illustrates the requirement of CRC board for recording the history of rental services of CRC's customers. Over here, the board will only need aid from CRC Web Interface and CRC Database. The flow 3, 4, 5 and 6 illustrates the requirement of CRC board to browse the numbers of different cars that are picked up or returned in some stored and car recommendations to customers utilising CRC Business Logic (technical) and fetching the results from CRC Database and again responding back to CRC Web Interface were member of CRC board is waiting for answer.

Artefact 2 – (Updating Customer’s Detail into the Database using sign-up page)

Created function which helps to update customer’s details like firstname, middlename and lastname, Date of Birth, email address and password the insert while signing up in the CRC. It helps to keep record of the customers those who sign-up into website. Customer enters data into sign-up page and enter data is sent to database with the help of python and Django.
It uses method called POST to input entered data to the database.

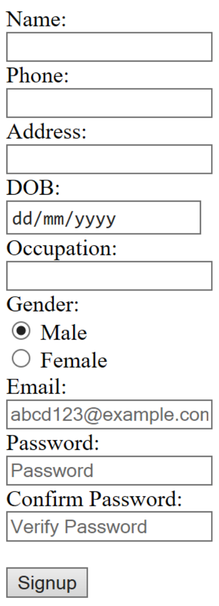


```python
def accounts(request):
    if request.method == "POST":
        if (request.POST.get('firstname') and request.POST.get('middlename') and request.POST.get('lastname') and
            request.POST.get('tel') and request.POST.get('bday') and request.POST.get('email') and request.POST.get('Password')):

            post = Customers()
            f = forms.CharField()
            post.name = (f.clean(request.POST.get('firstname')) + ' ' + f.clean(request.POST.get('middlename')) + ' ' + f.clean(request.POST.get('lastname')))
            post.phone = f.clean( request.POST.get('tel'))
            # post.address = f.clean( request.POST.get('Address'))
            post.dob = f.clean( request.POST.get('bday'))
            post.email = f.clean(request.POST.get('email'))
            post.password = f.clean(request.POST.get('Password'))
            post.save()
            return render(request, 'testApp/ShaleenCreateYourAccountPage.html')
        else:
            return render(request,'testApp/ShaleenCreateYourAccountPage.html')
    else:
        return render(request,'testApp/ShaleenCreateYourAccountPage.html')
```

Artefact 3 – (Html of signup page without CSS to check functionality)

This simple signup page which is used to check the functionality of the Django code as it submits the entered information to database. This was just used by me to test while coding. It has various fields like name, telephone number, Date of birth, occupation of customer, Gender of customer, email address and password. This all information are stored in the database after clicking on the submit button.
Password and confirm password are working field in this html page. Data is only sent to database when all the fields are filled and password and confirm password matches.



Name:

Phone:

Address:

DOB:
dd/mm/yyyy

Occupation:

Gender:
● Male
○ Female

Email:
abcd123@example.con

Password:
Password

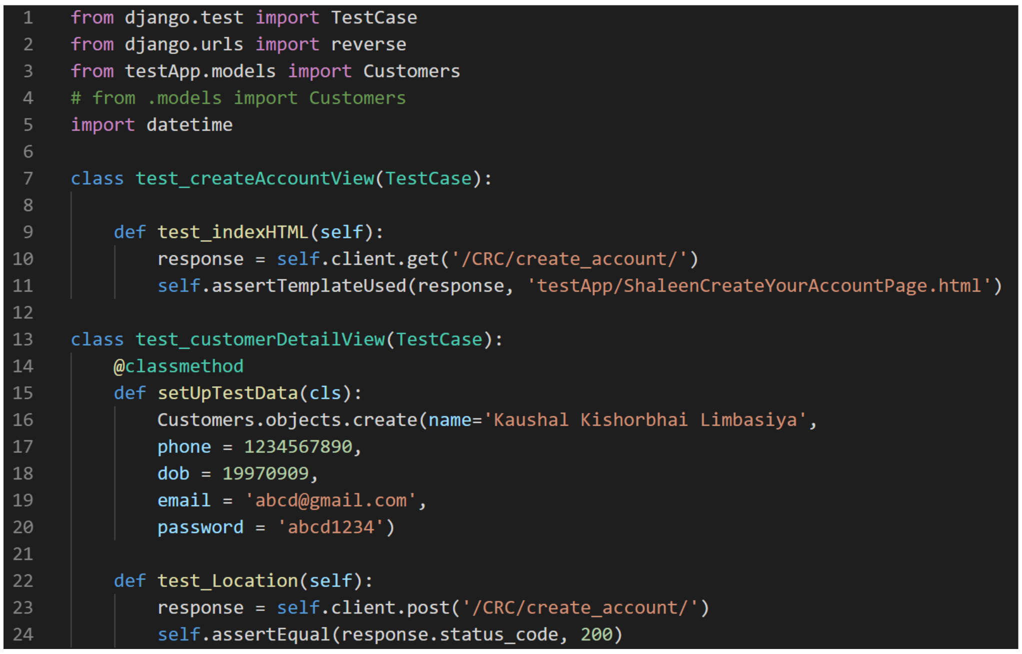
Confirm Password:
Verify Password

Artefact 4 – (Unit Testing of Database)

Unit test helps us to check working of the code. I ran unit test for inserting data into database. This test shows error if the data is not sent to a database or data inserted is wrong. The test "test_createAccountView" is carried out calling correct html page.

The test "test_customerDetailView" is carried out to know if the data is entered to the database correctly.
If data is not correctly entered to the database, it shows an error 200.

This testing helps us to know that signup page is filled correctly and data sent to database is at correct attribute.



```python
from django.test import TestCase
from django.urls import reverse
from testApp.models import Customers
# from .models import Customers
import datetime

class test_createAccountView(TestCase):

    def test_indexHTML(self):
        response = self.client.get('/CRC/create_account/')
        self.assertTemplateUsed(response, 'testApp/ShaleenCreateYourAccountPage.html')

class test_customerDetailView(TestCase):
    @classmethod
    def setUpTestData(cls):
        Customers.objects.create(name='Kaushal Kishorbhai Limbasiya',
        phone = 1234567890,
        dob = 19970909,
        email = 'abcd@gmail.com',
        password = 'abcd1234')

    def test_Location(self):
        response = self.client.post('/CRC/create_account/')
        self.assertEqual(response.status_code, 200)
```