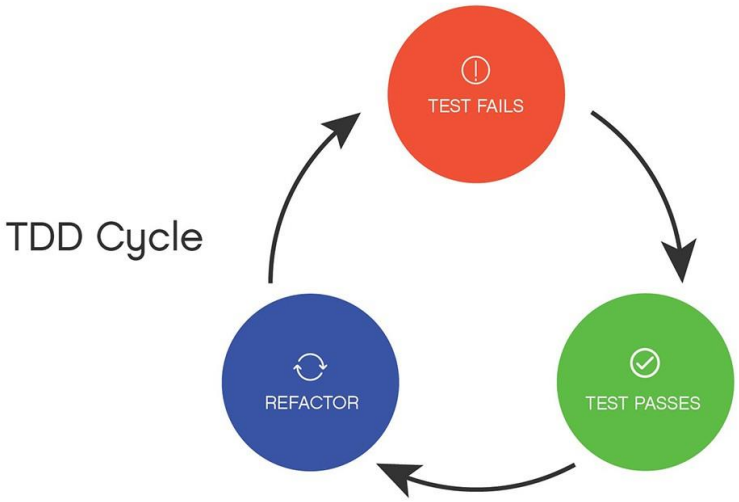
Test Driven Development Report for "Dhar-Hobe"- A Bangladeshi Product Renting System

Introduction

Test Driven Development is the process in which test cases are written before the code that validates those cases. It depends on repetition of a very short development cycle. Test driven Development is a technique in which automated Unit test are used to drive the design and free decoupling of dependencies. The steps are as follows:



1. Add a test – Write a test case that describe the function completely. In order to make the test cases the developer must understand the features and requirements using user stories and use cases.
2. Run all the test cases and make sure that the new test case fails.
3. Write the code that passes the test case
4. Run the test cases
5. Refactor code – This is done to remove duplication of code.
6. Repeat the above-mentioned steps again and again.

Pre-requisites

Tools: Virtual Studio Code

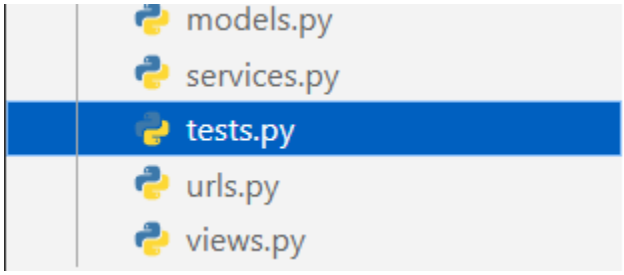
Framework: Python-Django

Unit Testing Tool: Django's built in testing framework for unit testing.

For Generating Test Report: pip install coverage

Procedure

1. Create tests.py file to write testcases.



2. From django.test import django's built in unit testing module TestCase. Also import Requestfactory and Client for simulating various kind of test cases.
3. Import all the files and modules you want to perform unit testing on. In our project's case, we have imported User, elements of models.py, views.py and services.py.

```
1  from django.test import TestCase, RequestFactory,Client
2  from django.contrib.auth.models import User
3  from django.urls import reverse
4  from .models import Renter, Product, RenterProduct
5  from .views import *
6  from .views import admin_page
7  from .services import EmailService,ProductService
8
```

4. Now write down the testcases. Follow the format mentioned in Unit Testing for test case writing. After adding test cases, to run the test cases of the functions in views.py, write *python manage.py test* in the terminal.

```
PS E:\4-2\SQA\Project\SQ_Project\Dhar-Hobe\BharaHobe> py manage.py test
```

Test Fails

Feature: Review Add Product Request

We have added 7 test cases. But 3 of them failed.

```
----------------------------------
Ran 7 tests in 0.073s

FAILED (failures=3)
```

```
======================================================================
FAIL: test_approved_requests (app_rentApproval.tests.ApprovedRequestsViewTest)
Test case to verify the behavior of the approved_requests view.
----------------------------------------------------------------------
======================================================================
FAIL: test_disapproved_requests (app_rentApproval.tests.DisapprovedRequestsViewTest)
Test case to verify the behavior of the disapproved_requests view.


FAIL: test_add_product_rqsts_with_no_products (app_rentApproval.tests.AddProductRqstsViewTest)
Test case to verify the behavior of the add_product_rqsts view when no products are available for rent.
```

At the time of failure, the views functions for the 3 of them were as below:

```python
def add_product_rqsts(request):
    """
    View function to display and handle product requests.

    If a POST request is received, it handles the action based on the request_i
    Otherwise, it renders the page with a list of product requests.

    Parameters:
        request (HttpRequest): The HTTP request object.

    Returns:
        HttpResponse: Renders the add product requests page with a list of proc
    """
    if request.method == 'POST':
        request_id = request.POST.get('request_id')
        action = request.POST.get('action')

        renter_product = RenterProductModel.objects.get(id=request_id)
        ProductService().handle_action(action, renter_product)

    requests = RenterProductModel.objects.filter(m_is_approved='pending')

    context = {'requests': requests}
    return render(request, 'app_rentApproval/add_product_rqsts.html', context)
```

```python
def approved_requests(request):
    """
    View function to display approved product requests.

    Parameters:
        request (HttpRequest): The HTTP request object.

    Returns:
        HttpResponse: Renders the approved requests page with a list of approved pro
    """
    approved_requests = RenterProductModel.objects.filter(m_is_approved='approved')
    context = {'approved_requests': approved_requests}
    return render(request, 'app_rentApproval/approved_requests.html', context)
```

```python
def disapproved_requests(request):
    """
    View function to display disapproved product requests.

    Parameters:
        request (HttpRequest): The HTTP request object.

    Returns:
        HttpResponse: Renders the disapproved requests page with a list of disapproved pr
    """
    disapproved_requests = RenterProductModel.objects.filter(m_is_approved='disapproved')
    context = {'disapproved_requests': disapproved_requests}
    return render(request, 'app_rentApproval/disapproved_requests.html', context)
```

Test Passes

Feature: Review Add Product Request

The test_add_product_rqsts_with_no_products was failing as there were no condition in the views for add_product_rqsts to handle the situation when there are no products for adding requests. After putting the condition, the test case run successfully for that function.

```
    if not requests:
        return HttpResponse('No add products requests for rent is available')
```

We have checked if the function is now performing ok for the test case by the following command

```
python manage.py test app_rentApproval.tests.AddProductRqstsViewTest.test_add_product_rqsts_with_no_products
```

```
----------------------------------------------------------------------
Ran 1 test in 0.005s

OK
Destroying test database for alias 'default'...
```

Also, we have modified the template code in approved_rqsts.html and disapproved_rqsts.html as there were some error. After modifying the needed codes in views.py file and template files, we have successfully passed all the test cases written in the tests.py file.

```
----------------------------------------------------------------------
Ran 7 tests in 0.066s

OK
Destroying test database for alias 'default'...
```

Test Fails

Feature: Review Return Product Request

For TDD, we have written the test cases first for ReturnRequest feature.

```python
#Sprint2 testing
class ReturnRequestViewTest(TestCase):

    def setUp(self):
        self.factory = RequestFactory()
        self.url = reverse('return_requests')

    def test_return_requests(self):
        rentee = RenteeModel.objects.create(m_username='testuser', m_password='testpassword', m_email='test@example.com',
                                            m_address='Test Address', m_phone_number='1234567890')
        product1 = ProductModel.objects.create(m_name='Product 1', m_description='Description 1', m_rental_price=10.0,
                                               m_quantity_available=5)
        product2 = ProductModel.objects.create(m_name='Product 2', m_description='Description 2', m_rental_price=15.0,
                                               m_quantity_available=8)

        ReturnRequestModel.objects.create(m_rentee=rentee, m_product=product1)
        ReturnRequestModel.objects.create(m_rentee=rentee, m_product=product2)

        # Create a GET request to the view
        request = self.factory.get(self.url)
        request.user = rentee
        response = return_requests(request)

        self.assertEqual(response.status_code, 200)
        self.assertContains(response, 'Product 1')
        self.assertContains(response, 'Product 2')

    def test_return_requests_with_no_products(self):
        rentee = RenteeModel.objects.create(m_username='testuser', m_password='testpassword', m_email='test@example.com',
                                            m_address='Test Address', m_phone_number='1234567890')
        request = self.factory.get(self.url)
        request.user = rentee
        response = return_requests(request)
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, 'No return requests for product is available')

```

After running the test cases, we can see that the test case fails.

```
PS E:\4-2\SQA\Project\SQ_Project\Dhar-Hobe\BharaHobe> python manage.py test app_rentApproval.tests.ReturnRequestViewTest
Found 2 test(s).
Creating test database for alias 'default'...
System check identified some issues:

WARNINGS:
?: (staticfiles.W004) The directory 'E:\4-2\SQA\Project\SQ_Project\Dhar-Hobe\BharaHobe\static' in the STATICFILES_DIRS sett
ing does not exist.

System check identified 1 issue (0 silenced).
.F
======================================================================
FAIL: test_return_requests_with_no_products (app_rentApproval.tests.ReturnRequestViewTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "E:\4-2\SQA\Project\SQ_Project\Dhar-Hobe\BharaHobe\app_rentApproval\tests.py", line 270, in test_return_requests_wit
h_no_products
    self.assertContains(response, 'No return requests for product is available')
  File "C:\Python310\lib\site-packages\django\test\testcases.py", line 658, in assertContains
    self.assertTrue(
AssertionError: False is not true : Couldn't find 'No return requests for product is available' in response

----------------------------------------------------------------------
Ran 2 tests in 0.024s

FAILED (failures=1)
Destroying test database for alias 'default'...
```

At that time, the body of the return_requests function in views.py was not fully implemented. That's why test case failed.

```
117 #SPRINT 2 views for return requests
118 def return_requests(request):
119     return_requests=ReturnRequestModel.objects.filter(m_is_approved='pending')
120     context = {'return_requests':return_requests}
121     return render(request,'app_rentApproval/return_product_rqsts.html',context)
```

Test Passes

After modifying the body of the return_requests function in views.py according to the test functions in tests.py, the code looks as following:

```
117 #SPRINT 2 views for return requests
118 def return_requests(request):
119
120     return_requests=ReturnRequestModel.objects.filter(m_is_approved='pending')
121
122     if not return_requests:
123         return HttpResponse('No return requests for product is available')
124
125     context = {'return_requests':return_requests}
126     return render(request,'app_rentApproval/return_product_rqsts.html',context)
```

After implementing the function body according to the test function, the test cases passes.

```
PS E:\4-2\SQA\Project\SQ_Project\Dhar-Hobe\BharaHobe> python manage.py test app_rentApproval.tests.ReturnRequestViewTest
Found 2 test(s).
Creating test database for alias 'default'...
System check identified some issues:

WARNINGS:
?: (staticfiles.W004) The directory 'E:\4-2\SQA\Project\SQ_Project\Dhar-Hobe\BharaHobe\static' in the STATICFILES_DIRS se

System check identified 1 issue (0 silenced).
..
----------------------------------------------------------------------
Ran 2 tests in 0.021s

OK
Destroying test database for alias 'default'...
PS E:\4-2\SQA\Project\SQ_Project\Dhar-Hobe\BharaHobe>
```

Results

To measure the coverage and accuracy of the test cases upon the modules, write in the command line the following commands.

```
pip install coverage
coverage run manage.py test
coverage report
```

Basically, we have tested the modules in our project app and found the following coverage rate:

```
app_rentApproval\apps.py                                                    4      0   100%
app_rentApproval\migrations\0001_initial.py                                 5      0   100%
app_rentApproval\migrations\0002_renter.py                                  4      0   100%
app_rentApproval\migrations\0003_renterproduct.py                           5      0   100%
app_rentApproval\migrations\0004_rename_price_product_rental_price.py       4      0   100%
app_rentApproval\migrations\0005_renterproduct_is_approved.py               4      0   100%
app_rentApproval\migrations\0006_renterproduct_approved_at.py               4      0   100%
app_rentApproval\migrations\0007_alter_renterproduct_is_approved.py         4      0   100%
app_rentApproval\migrations\0008_productmodel_rentermodel_renterproductmodel_and_more.py    5      0   100%
app_rentApproval\migrations\__init__.py                                     0      0   100%
app_rentApproval\models.py                                                 29      3    90%
app_rentApproval\services.py                                               12      0   100%
app_rentApproval\tests.py                                                  94      0   100%
app_rentApproval\urls.py                                                    3      0   100%
app_rentApproval\views.py                                                  43      4    91%
manage.py                                                                  12      2    83%
rentapp\__init__.py                                                         0      0   100%
rentapp\migrations\__init__.py                                              0      0   100%
rentapp\tests.py                                                            1      0   100%
rentapp\views.py                                                            3      1    67%
------------------------------------------------------------------------------------------------
TOTAL                                                                     316     32    90%
```

So, the total coverage is 90%.

References:

1. https://www.geeksforgeeks.org/test-driven-development-tdd/
2. https://brainhub.eu/library/test-driven-development-tdd
3. https://github.com/JU-CSE-27/swe-wiki/blob/master/resources/TDD.pdf