Test Driven Development (TDD)

“Product Construction or Implementation Phase”

Tools: *Virtual Studio Code*

Framework: *Python-Django*

Required Installations:

For Testing Python Core Code: *pip install pytest*

For Testing Django-Framework: *pip install pytest-django*

For Generating Test Report: *pip install pytest-cov*

For helping to generate instances of Django models: *pip install mixer*

Flows of Process (FOP)

1. In Root Project Repo “ecom”: creating a file *test_setting.py*

test_setting.py

```
from .settings import *
DATABASES={
    "default":{
        "ENGINE":"django.db.backends.sqlite3",
        "NAME":":memory",
    }
   }
EMAIL_BACKEND='django.core.mail.backends.locmem.EmailBackend'
```

2. In Root Project Repo “ecom”: creating a file pytest.ini

pytest.ini

```
[pytest]
DJANGO_SETTINGS_MODULE = ecom.test_setting
addopts = --nomigrations --cov=. --cov-report=html
```

3. In inner root “ecom” repo: creating a folder “tests”

   in “tests” folder create 2 files “__init__.py” and “test_models.py”

test_models.py

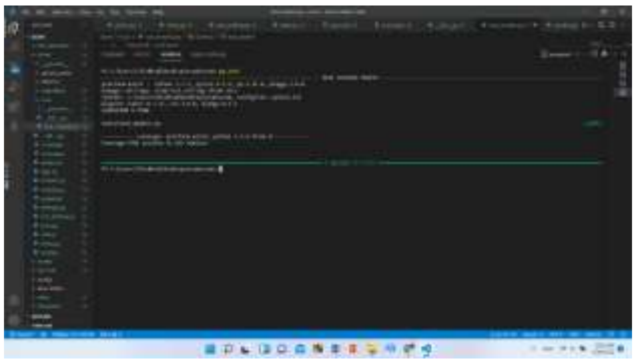
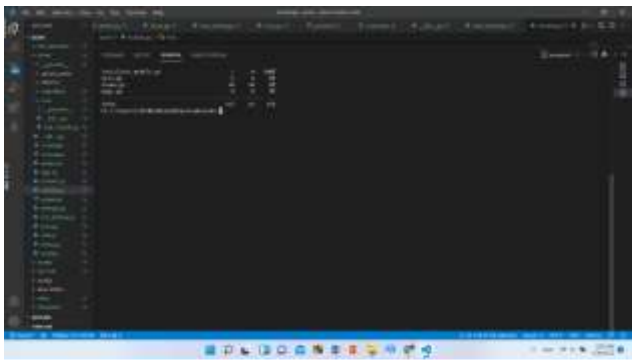
```
import pytest
from mixer.backend.django import mixer
pytestmark = pytest.mark.django_db
class TestPost:
    def test_model(self):
        obj = mixer.blend('ecom.Post')
        assert obj.pk == 1, 'Should create a Post instance'
```

models.py

```
class Post(models.Model):
    body = models.TextField()
```

4. Write a command in terminal for testing models.py : py.test

5. Checking Coverage Report For Models:Write a command in Terminal: coverage report

6. In “test” folder create a file : “test_admin.py”

test_admin.py

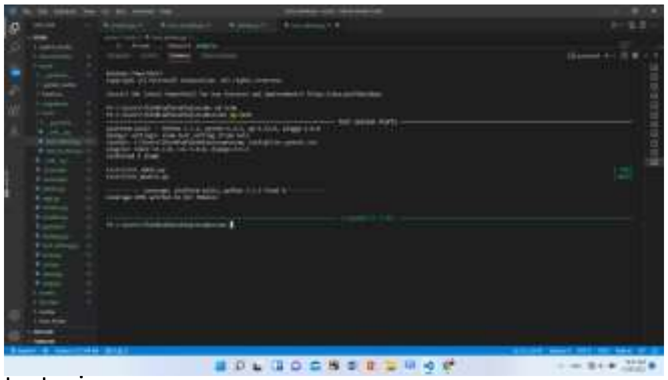
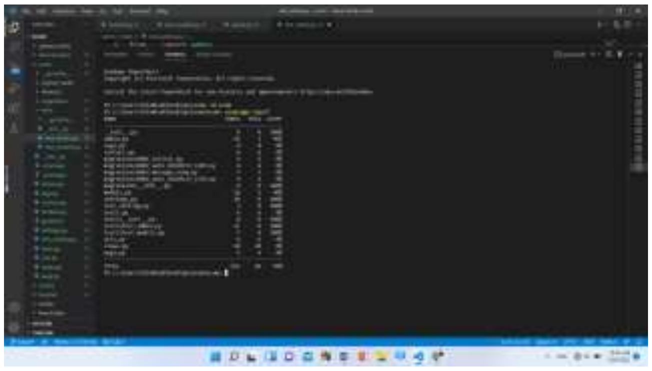
```
class TestPostAdmin:
    def test_excerpt(self):
        site = AdminSite()
        post_admin = admin.PostAdmin(models.Post,site)
        obj = mixer.blend('ecom.Post',body="Hello World")
        result = post_admin.excerpt(obj)
        assert result == 'Hello', 'Should return first few 
characters'
```

admin.py

```
class PostAdmin(admin.ModelAdmin):
    models = models.Post
    list_display = ('excerpt', )
    def excerpt(self, obj):
        return obj.get_excerpt(5)
admin.site.register(models.Post, PostAdmin)
```

7. Then run a command in terminal : py.test

8. Checking Coverage Report For admin:Write a command in Terminal: coverage report

test_view.py

```
from django.test import RequestFactory
from .. import views
class TestHomeView:
    def test_anonymous(self):
        req= RequestFactory().get('/')
        resp=views.HomeView.as_view()(req)
        assert resp.status_code==200, 'Should be callable by an
anyone'
```

views.py

```
from django.views.generic import TemplateView

class HomeView(TemplateView):
    template_name = 'ecom/home.html'
```

10. Then run a command in terminal : py.test

11. Checking Coverage Report For admin:Write a command in Terminal: coverage report

12. Testing Authentication

test_view.py

```
import pytest
from django.test import RequestFactory
from django.contrib.auth.models import AnonymousUser
from mixer.backend.django import mixer
pytestmark = pytest.mark.django_db
from .. import views
class TestAdminView:
    def test_anonymous(self):
        req = RequestFactory().get('/')
        req.user=AnonymousUser()
        resp=views.AdminView.as_view()(req)
        assert 'login' in resp.url
    def test_superuser(self):
        user = mixer.blend('auth.User',is_superuser=True)
        req = RequestFactory().get('/')
        req.user=user
        resp=views.AdminView.as_view()(req)
        assert resp.status_code == 200, 'Authenticated user can
access'
```

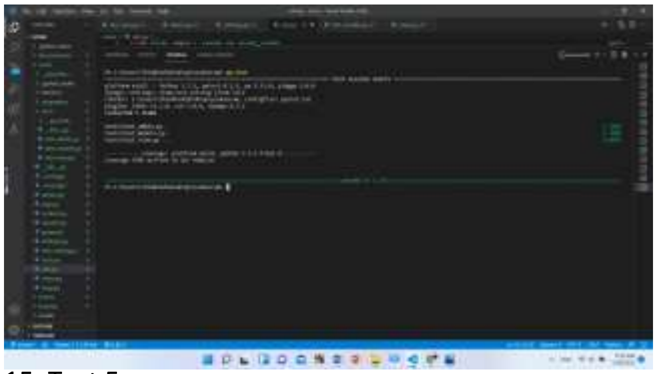
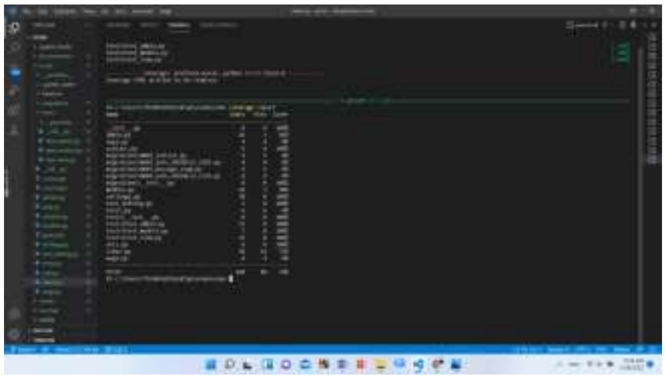
views.py

```
from django.views.generic import TemplateView
from django.contrib.auth.decorators import login_required
from django.utils.decorators import method_decorator

class AdminView(TemplateView):
    template_name='ecom/home.html'
    @method_decorator(login_required)
    def dispatch(self, request, *args, **kwargs):
        return super().dispatch(request, *args, **kwargs)
```

13. Then run a command in terminal : py.test

14. Checking Coverage Report For autentication: Write a command in Terminal: coverage report

15. Test Forms

in tests repo create test_form.py and in inner ecom repo create forms.py

test_form.py

```
import pytest
pytestmark = pytest.mark.django_db
from .. import forms
class TestPostForm:
    def test_form(self):
        form = forms.PostForm(data={})
        assert form.is_valid() is False, 'Should be invalid if 
no data given'

        form = forms.PostForm(data={'body':'Hello'})
        assert form.is_valid() is False, 'Should be invalid if 
too short'
        assert 'body' in form.errors, 'Should have body field 
error'

        form = forms.PostForm(data = {'body':'Hello 
World!!!!!!!'})
        assert form.is_valid() is True, 'Should be Valid if 
enough'
```

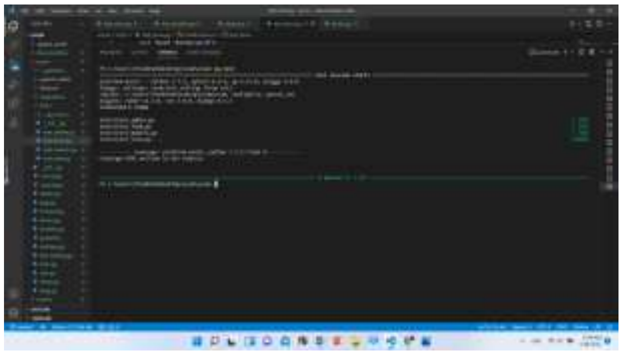
forms.py

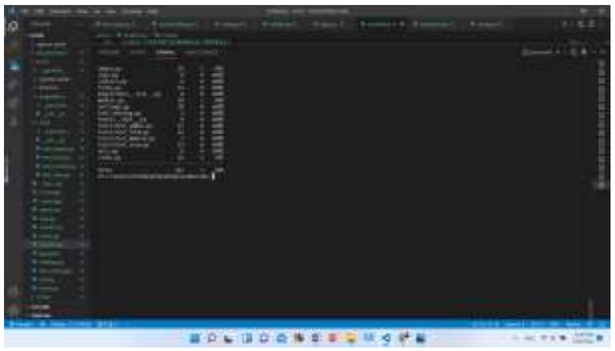
```
from pyexpat import model
from attr import fields
```

```python
from django import forms
from . import models
class PostForm(forms.ModelForm):
    class Meta:
        model = models.Post
        fields=('body',)
    def clean_body(self):
        data = self.cleaned_data.get('body')
        if len(data) <=5:
            raise forms.ValidationError('Message is too short')
        return data
```

16. Then run a command in terminal : py.test

17. Checking Coverage Report For forms: Write a command in Terminal: coverage report


[illegible]


[illegible]

Individual 6 TDD Reports:

[illegible]

[illegible]

[illegible]

[illegible]

Coverage for **contact.py**: 100%

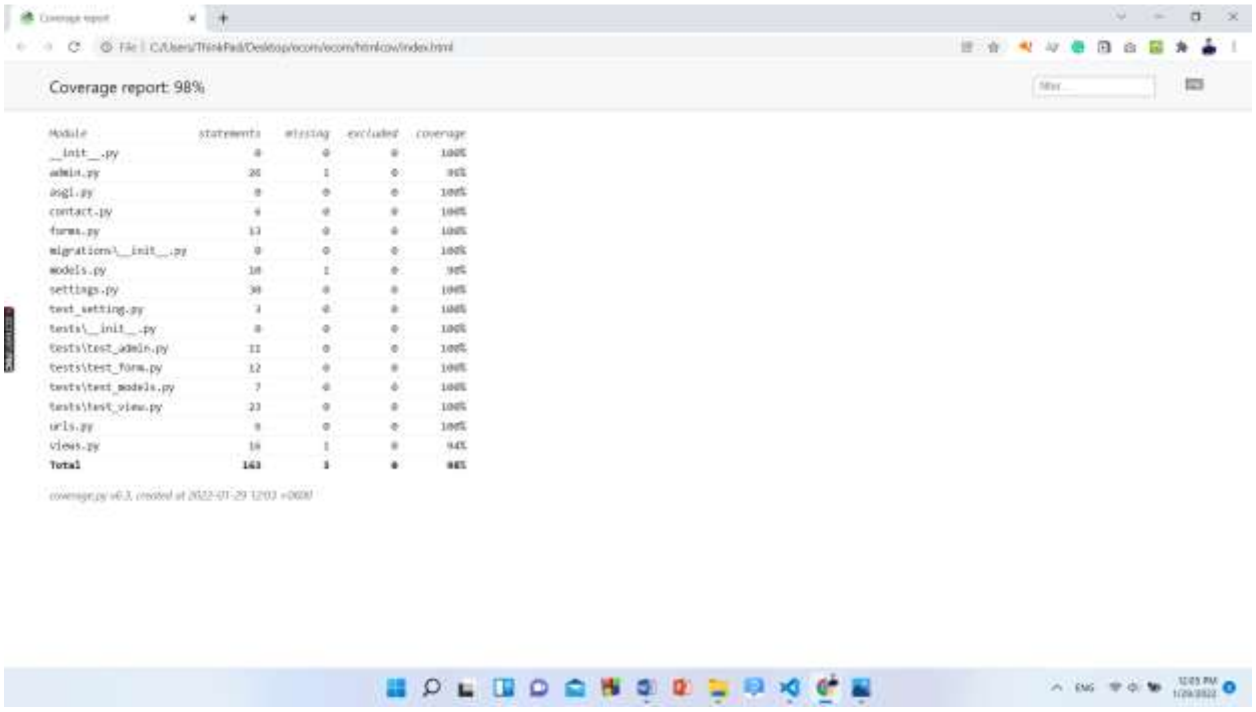
Coverage for **admin.py**: 96%

Conclusion:

In brief, we tested six modules and found 98% accuracy tested result-

1. urls.py: Coverage rate: 100%
2. views.py: Coverage rate: 98%
3. settings.py: Coverage rate: 100%
4. forms.py: Coverage rate: 100%
5. contact.py: Coverage rate: 100%
6. admin.py: Coverage rate: 98%

Total Coverage Rate: 98%


C:/Users/ThinkPad/Desktop/ecom/ecom/htmlcov/index.html

Coverage report: 98%

Module	statements	missing	excluded	coverage
__init__.py	0	0	0	100%
admin.py	26	1	0	96%
asgi.py	0	0	0	100%
contact.py	4	0	0	100%
forms.py	13	0	0	100%
migrations__init__.py	0	0	0	100%
models.py	18	1	0	94%
settings.py	30	0	0	100%
test_setting.py	3	0	0	100%
tests__init__.py	0	0	0	100%
tests\test_admin.py	11	0	0	100%
tests\test_form.py	12	0	0	100%
tests\test_models.py	7	0	0	100%
tests\test_view.py	23	0	0	100%
urls.py	0	0	0	100%
views.py	16	1	0	94%
Total	**163**	**3**	**0**	**98%**

coverage.py v6.3, created at 2022-01-29 12:03 +0600

References:

1. For clear photos, please check the following link :

   https://github.com/JU-CSE-27/swe-wiki/tree/master/test_reports

2. For viewing report in HTML format:

   https://sew-tdd.netlify.app/