

## Test Driven Development (TDD)

**Tools:** Virtual Studio Code

**Framework:** Python-Django

**Required Installations:**

**For Testing Python Core Code:** pip install pytest

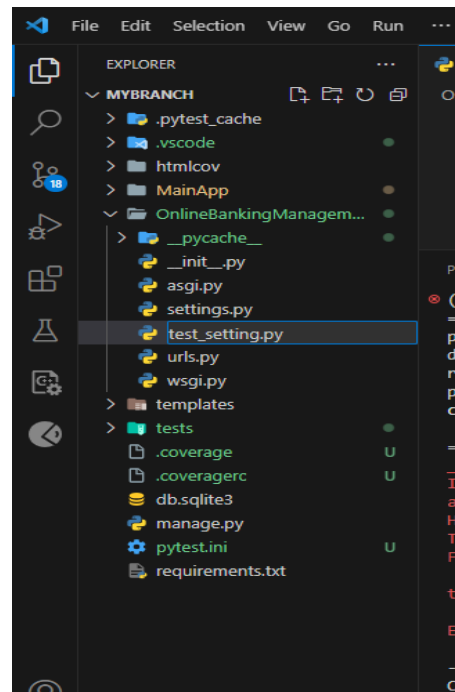
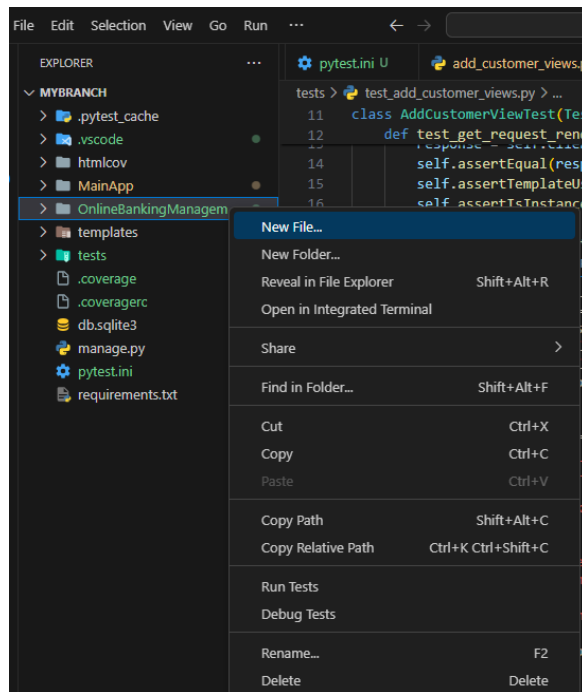
**For Testing Django-Framework:** pip install pytest-django

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

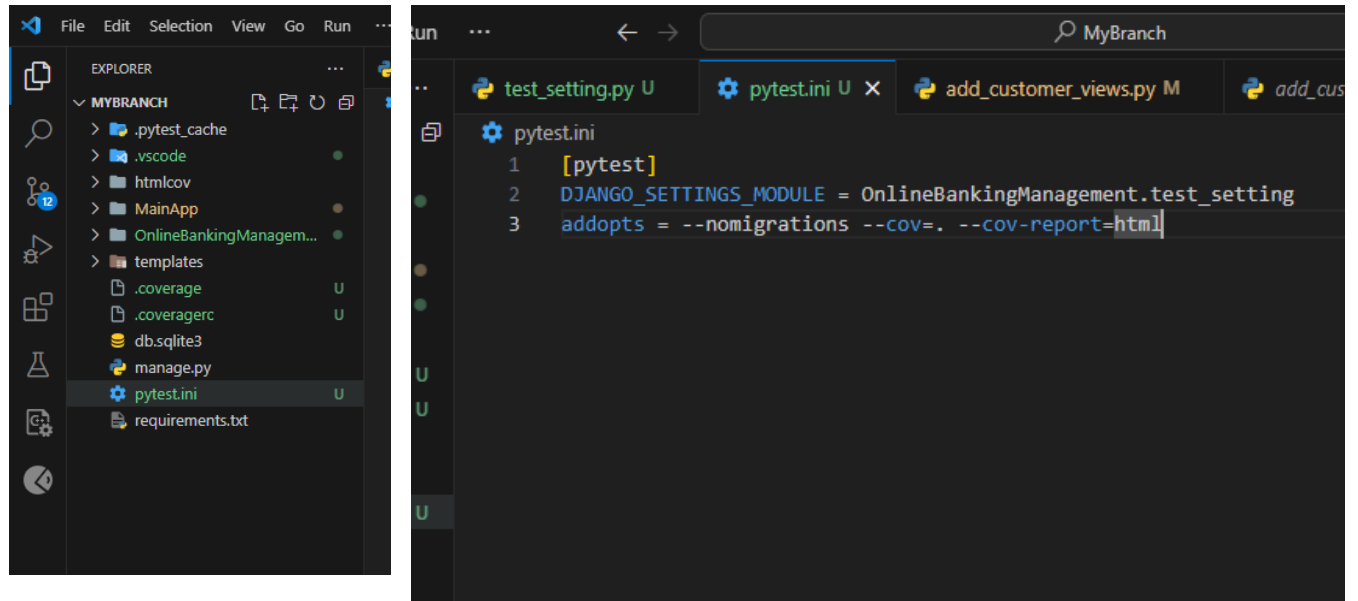
**For helping to generate instances of Django models:** pip install mixer

### Flows of Process (FOP):

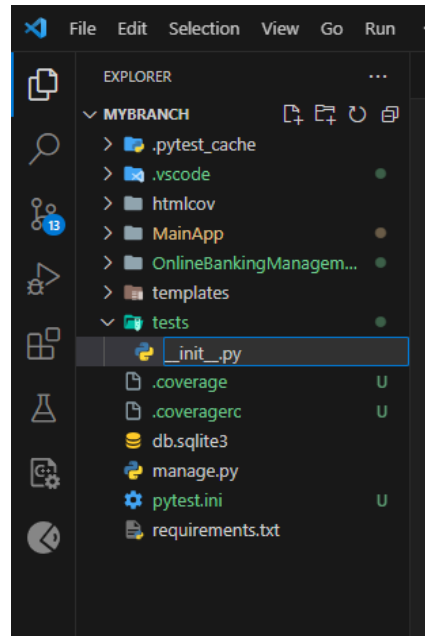
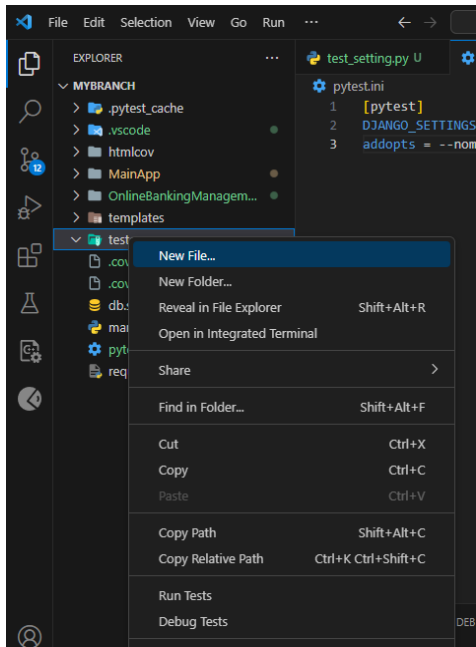
1. In Root Project Repo “OnlineBankingManagement”: creating a file test\_setting.py



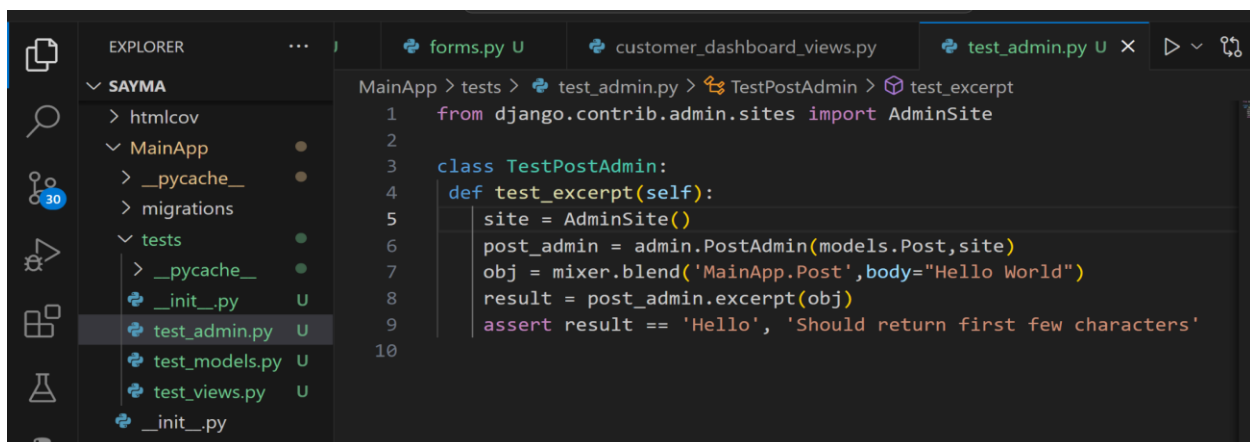
2. In Root Project Repo “OnlineBankingManagement”: creating a file pytest.ini



3. In inner root “MainApp” repo: creating a folder “tests” in “tests” folder create 4 files “\_\_init\_\_.py”, “test\_models.py”, “test\_views.py” and “test\_admin.py”



test\_admin.py

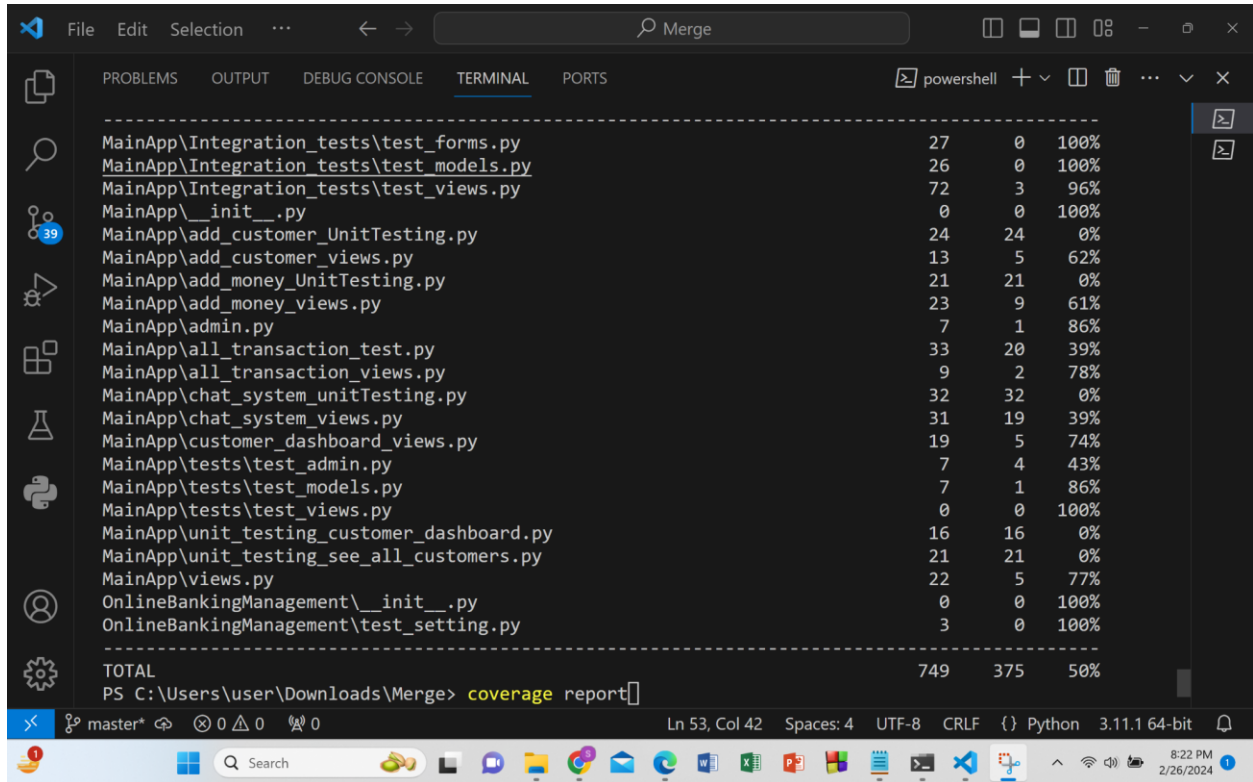


admin.py



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

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

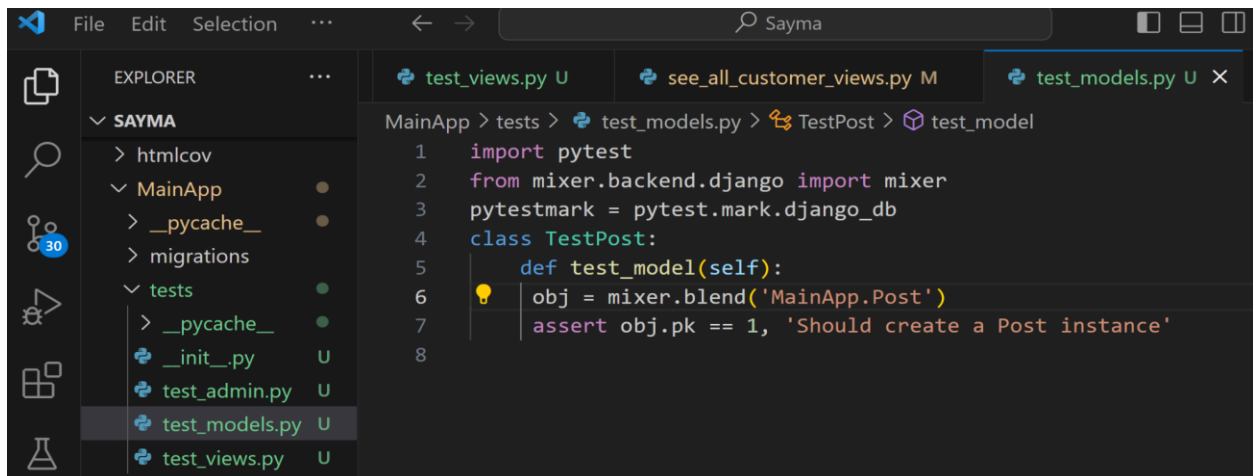


The screenshot shows a VS Code terminal window with a coverage report. The report lists various Python files and their corresponding line counts, missed lines, and coverage percentages. The files are organized into a table with three columns: File, Lines, Missed, and Coverage. The files include integration tests, unit tests, and views for a Django application. The terminal also shows the command `coverage report` being executed in a PowerShell prompt.

File	Lines	Missed	Coverage
MainApp\Integration_tests\test_forms.py	27	0	100%
MainApp\Integration_tests\test_models.py	26	0	100%
MainApp\Integration_tests\test_views.py	72	3	96%
MainApp\__init__.py	0	0	100%
MainApp\add_customer_UnitTesting.py	24	24	0%
MainApp\add_customer_views.py	13	5	62%
MainApp\add_money_UnitTesting.py	21	21	0%
MainApp\add_money_views.py	23	9	61%
MainApp\admin.py	7	1	86%
MainApp\all_transaction_test.py	33	20	39%
MainApp\all_transaction_views.py	9	2	78%
MainApp\chat_system_UnitTesting.py	32	32	0%
MainApp\chat_system_views.py	31	19	39%
MainApp\customer_dashboard_views.py	19	5	74%
MainApp\tests\test_admin.py	7	4	43%
MainApp\tests\test_models.py	7	1	86%
MainApp\tests\test_views.py	0	0	100%
MainApp\unit_testing_customer_dashboard.py	16	16	0%
MainApp\unit_testing_see_all_customers.py	21	21	0%
MainApp\views.py	22	5	77%
OnlineBankingManagement\__init__.py	0	0	100%
OnlineBankingManagement\test_setting.py	3	0	100%
TOTAL	749	375	50%

PS C:\Users\user\Downloads\Merge> `coverage report`

test\_models.py



The screenshot shows a VS Code editor window with the `test_models.py` file open. The file is located in the `MainApp > tests > test_models.py` directory. The code defines a `TestPost` class with a `test_model` method. The method uses `mixer.blend` to create a `Post` instance and asserts that its `pk` attribute is equal to 1.

```
1 import pytest
2 from mixer.backend.django import mixer
3 pytestmark = pytest.mark.django_db
4 class TestPost:
5     def test_model(self):
6         obj = mixer.blend('MainApp.Post')
7         assert obj.pk == 1, 'Should create a Post instance'
8
```

model.py

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

### Test\_views.py

```
MainApp > tests > test_views.py > TestAdminView > test_superuser  
1  from django.test import RequestFactory  
2  from .. import views  
3  class TestHomeView:  
4      def test_anonymous(self):  
5          req= RequestFactory().get('/')  
6          resp=views.HomeView.as_view()(req)  
7          assert resp.status_code==200, 'Should be callable by an anyone'  
8
```

### views.py

```
from django.views.generic import TemplateView  
class HomeView(TemplateView):  
    template_name = 'OnlineBankingManagement/home.html'
```

6. Write a command in terminal for testing models.py and views.py : `py.test`

7. Checking Coverage Report For Models and views: Write a command in Terminal:  
`coverage report`

	106	38	64%
MainApp\forms.py	106	38	64%
MainApp\login_logout_views.py	16	9	44%
MainApp\login_unit_testing_SB.py	27	27	0%
MainApp\migrations\0002_message.py	6	6	0%
MainApp\migrations\0002_userprofile_phone_no.py	4	4	0%
MainApp\migrations\0003_merge_0002_message_0002_userprofile_phone_no.py	4	4	0%
MainApp\migrations\__init__.py	0	0	100%
MainApp\models.py	35	0	100%
MainApp\money_transfer_UnitTesting.py	20	20	0%
MainApp\money_transfer_views.py	38	25	34%
MainApp\profile_update_unit_test_SB.py	23	23	0%
MainApp\profile_update_views.py	30	16	47%
MainApp\see_all_customer_views.py	12	0	100%
MainApp\tests.py	15	15	0%
MainApp\tests\__init__.py	0	0	100%
MainApp\tests\test_admin.py	7	4	43%
MainApp\tests\test_models.py	7	1	86%
MainApp\tests\test_views.py	7	0	100%
MainApp\unit_testing_customer_dashboard.py	16	16	0%
MainApp\unit_testing_see_all_customers.py	21	21	0%
MainApp\views.py	25	5	80%
OnlineBankingManagement\__init__.py	0	0	100%
OnlineBankingManagement\test_setting.py	3	0	100%
TOTAL	759	375	51%

PS C:\Users\user\Downloads\Merge>

## test\_forms.py

```

MainApp > tests > test_forms.py > TestPostForm > test_form
1  import pytest
2  pytestmark = pytest.mark.django_db
3  from .. import forms
4  class TestPostForm:
5      def test_form(self):
6          form = forms.PostForm(data={})
7          assert form.is_valid() is False, 'Should be invalid if no data given'
8
9          form = forms.PostForm(data={'body': 'Hello'})
10         assert form.is_valid() is False, 'Should be invalid if too short'
11         assert 'body' in form.errors, 'Should have body field error'
12
13         form = forms.PostForm(data = {'body': 'Hello World!!!!!!!!'})
14         assert form.is_valid() is True, 'Should be Valid if enough'

```

## form.py

```

MainApp > forms.py > PostForm > Meta > clean_body
189
190 from pyexpat import model
191 from attr import fields
192 from django import forms
193 from . import models
194 class PostForm(forms.ModelForm):
195     class Meta:
196         model = models.Post
197         fields=('body',)
198         def clean_body(self):
199             data = self.cleaned_data.get('body')
200             if len(data) <=5:
201                 raise forms.ValidationError('Message is too short')
202             return data
203

```

8. Then run a command in terminal : `py.test`

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

PS C:\Users\user\Downloads\Merge> coverage report				
Name	Stmts	Miss	Cover	
-----	-----	-----	-----	
MainApp\Integration_tests\test_forms.py	27	0	100%	
MainApp\Integration_tests\test_models.py	26	0	100%	
MainApp\Integration_tests\test_views.py	72	3	96%	
MainApp\__init__.py	0	0	100%	
MainApp\add_customer_views.py	13	5	62%	
MainApp\add_money_views.py	23	9	61%	
MainApp\admin.py	9	1	89%	
MainApp\all_transaction_views.py	9	2	78%	
MainApp\chat_system_views.py	31	19	39%	
MainApp\customer_dashboard_views.py	19	5	74%	
MainApp\forms.py	119	42	65%	
MainApp\login_logout_views.py	16	9	44%	
MainApp\migrations\__init__.py	0	0	100%	
MainApp\models.py	35	0	100%	
MainApp\money_transfer_views.py	38	25	34%	
MainApp\profile_update_views.py	30	16	47%	
MainApp\see_all_customer_views.py	12	0	100%	
MainApp\tests.py	0	0	100%	
MainApp\tests\__init__.py	0	0	100%	
MainApp\tests\test_admin.py	9	3	67%	
MainApp\tests\test_forms.py	12	3	75%	
MainApp\tests\test_models.py	7	1	86%	
MainApp\tests\test_views.py	25	0	100%	
MainApp\urls.py	5	0	100%	
MainApp\views.py	33	5	85%	
OnlineBankingManagement\__init__.py	0	0	100%	
OnlineBankingManagement\settings.py	20	0	100%	
OnlineBankingManagement\test_setting.py	3	0	100%	
OnlineBankingManagement\urls.py	3	0	100%	
-----	-----	-----	-----	
TOTAL	596	148	75%	
PS C:\Users\user\Downloads\Merge>				

**Conclusion:**

In brief, we tested six modules and found 75% accuracy tested result-

1. urls.py: Coverage rate: 100%
2. views.py: Coverage rate: 85%
3. settings.py: Coverage rate: 100%
4. forms.py: Coverage rate: 65%
5. admin.py: Coverage rate: 89%

Total Coverage Rate: 75%