

Міністерство освіти і науки, молоді та спорту України  
Національний Технічний Університет України  
«Київський Політехнічний Інститут»  
Навчально-науковий комплекс  
«Інститут прикладного системного аналізу»  
Кафедра системного проектування

**Лабораторна робота №5**

З курсу: «Проектування інформаційних систем»

На тему: «Модульне тестування (Unit-тести) та рефакторинг»

Виконала:

Студентка 4 курсу

Групи ДА-72

Лупяк А.С.

## **Завдання:**

1. Розробити код програми архітектурної моделі. Використовувати Test Driven Development.
2. Провести рефакторинг коду програми, щоб задовольнити вимоги технічного завдання.

Тестуємо функціонал для прогнозування попиту на велопрокат

```
from django.test import TestCase
```

```
from core.forms import SearchForm
```

```
from core.models import History
```

```
class BicycleTestClass(TestCase):
```

```
    def setUp(self):
```

```
        self.form_data = {'date': '10.12.2020', 'amount_of_hours': 3, 'start_time': '15.00', 'weather':  
3}
```

```
    def test_check_forecast_label(self):
```

```
        form = SearchForm(self.form_data)
```

```
        field_label = fav_obj._meta.get_field('weather').verbose_name
```

```
        self.assertEqual(field_label, 'weather')
```

```
    def test_search_form_valid(self):
```

```
        form = SearchForm(self.form_data)
```

```
        self.assertTrue(form.is_valid())
```

```
    def test_forecast_result(self):
```

```
        form = SearchForm(self.form_data)
```

```
        result = forecast(form)
```

```
        self.assertTrue(result)
```

Тестуємо функціонал авторизації

```
def test_search_added_to_history(self):  
    form = SearchForm(self.form_data)  
    n_before = History.objects.all().count()  
    r = submit_form(form)  
    r.save()  
    n_after = History.objects.all().count()  
    self.assertEqual(r.resp_code, 200)  
    self.assertEqual(n_before + 1, n_after)
```

```
from django.test import TestCase, Client  
from user.models import Favorite  
from user.forms import UserLoginForm, UserRegisterForm  
from django.contrib.auth import (  
    authenticate,  
    login,  
)
```

```
class UserTestClass(TestCase):  
  
    def setUp(self):  
        self.user_data = {  
            'username': 'AAAAA',  
            'password': '21345466789JHGF',  
        }  
  
    def test_user_registartion_valididation(self):  
        form = UserRegisterForm(self.user_data)  
        self.assertTrue(form.is_valid())
```

```
wrong_registration_data = {  
    'username': '0',  
    'password': '',  
}  
  
form = UserRegisterForm(wrong_registration_data)  
  
self.assertFalse(form.is_valid())
```

```
def test_user_authentication(self):
```

```
    form = UserRegisterForm(self.user_data)
```

```
    if form.is_valid():
```

```
        user = form.save(commit=False)
```

```
        password = form.cleaned_data.get('password')
```

```
        user.set_password(password)
```

```
        user.save()
```

```
        self.assertTrue(authenticate(username=user.username, password=password))
```

```
        resp = self.client.get('/login')
```

```
        self.assertEqual(resp.status_code, 200)
```

```
        resp = self.client.get('/logout')
```

```
        self.assertEqual(resp.status_code, 302)
```

```
def test_user_login_validation(self):
```

```
    form = UserLoginForm(self.user_data)
```

```
    if form.is_valid():
```

```
        username = form.cleaned_data.get("username")
```

```
        password = form.cleaned_data.get("password")
```

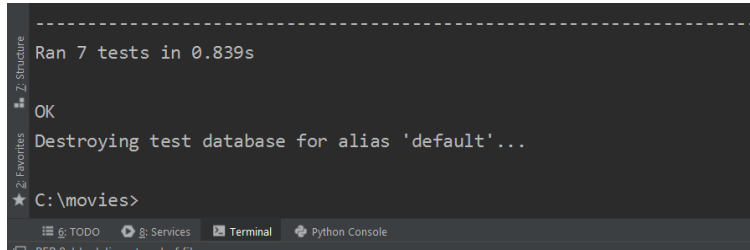
```
        user = authenticate(username=username, password=password)
```

```
        c = Client()
```

```
        c.get('/login', self.user_data)
```

```
        self.assertTrue(login(c.get('/login'), user))
```

## Результат виконання тестів



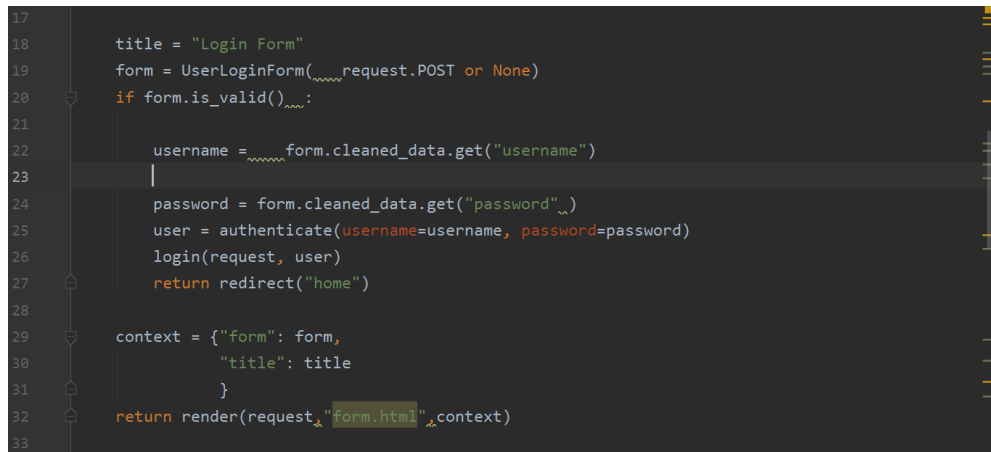
```
-----
Ran 7 tests in 0.839s

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

C:\movies>
```

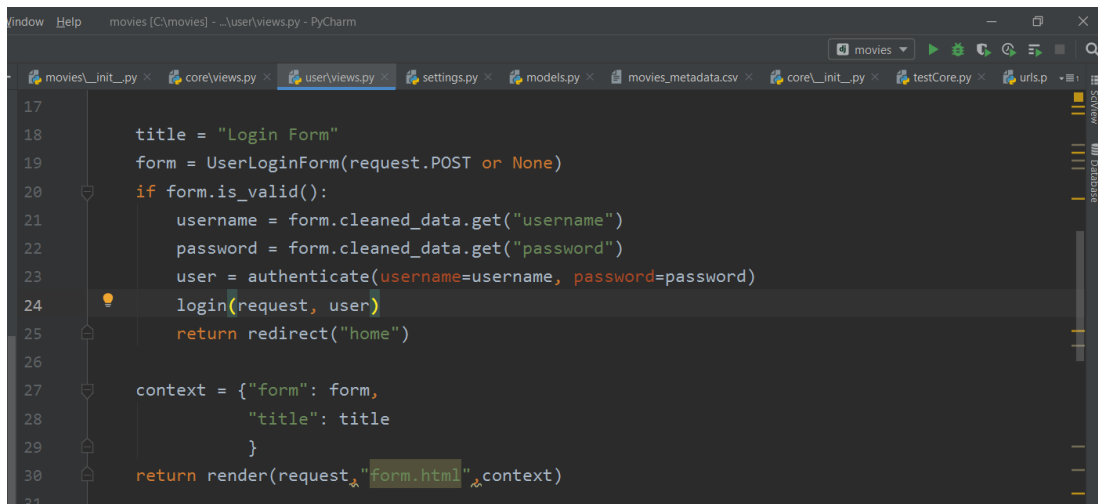
## 2. Проведемо Рефакторинг

До



```
17
18     title = "Login Form"
19     form = UserLoginForm(request.POST or None)
20     if form.is_valid():
21
22         username = form.cleaned_data.get("username")
23         |
24         password = form.cleaned_data.get("password")
25         user = authenticate(username=username, password=password)
26         login(request, user)
27         return redirect("home")
28
29     context = {"form": form,
30               "title": title
31               }
32     return render(request, "form.html", context)
33
```

Після



```
17
18     title = "Login Form"
19     form = UserLoginForm(request.POST or None)
20     if form.is_valid():
21         username = form.cleaned_data.get("username")
22         password = form.cleaned_data.get("password")
23         user = authenticate(username=username, password=password)
24         login(request, user)
25         return redirect("home")
26
27     context = {"form": form,
28               "title": title
29               }
30     return render(request, "form.html", context)
31
```

## Висновок

При виконанні роботи ознайомились з процедурою рефакторингу, створювали ПЗ за допомогою Test Driven Development. Для рефакторингу коду було застосовано вбудований у PyCharm інструмент для рефакторингу. Тести виконались успішно за 1.861 секунди. Вони були написані до розробки

коду, тому такий підхід можна назвати – TDD (Test Driven Development) методологією.