

## Pertemuan 6: Pengujian Aplikasi

### Capaian Pembelajaran

Mahasiswa mampu memahami cara menulis kode untuk melakukan pengujian aplikasi web pada framework Django  
mahasiswa mampu memahami WEB API  
mahasiswa memahami cara kerja pencarian data  
mahasiswa mampu mengimplementasi rancangan laman pencarian dan penampilan hasil pencarian di platform web

### Studi kasus :

Menerapkan pengujian pada aplikasi sederhana yang sudah dibuat pada Pertemuan sebelumnya

### Rincian Tugas Praktikum:

#### Tahap Persiapan

1. Mengaktifkan virtual environment dari proyek yang akan digunakan
2. Melakukan instalasi library yang akan digunakan.

#### Teori Singkat

Pengujian aplikasi adalah bagian penting dari siklus pengembangan aplikasi perangkat lunak.

Tujuan melakukan pengujian:

1. Menjamin kualitas
2. Memastikan kepuasan pengguna
3. Mendukung perubahan dan pemeliharaan
4. Meningkatkan kepercayaan
5. Mengurangi biaya dan waktu
6. Kepatuhan dan keamanan

Ada banyak tipe, level dan klasifikasi pengujian. Beberapa metode pengujian otomatis yang penting adalah:

1. Unit test: melakukan verifikasi perilaku dari komponen tunggal (dilakukan di level kelas atau fungsi)
2. Integration test: melakukan verifikasi bagaimana sekelompok komponen bekerja jika digunakan secara bersama sama.
3. Regresion test: pengujian perangkat lunak yang bertujuan memastikan bahwa perubahan yang dilakukan pada kode program tidak merusak fungsionalitas yang sebelumnya sudah dapat berjalan dengan baik.

Django menyediakan framework pengujian yang dikembangkan diatas library **unittest**. Library ini mencakup fungsi-pengujian unit test dan integration test. Ada beberapa kelas test yang

tersedia, yang paling umum digunakan adalah **TestCase (django.test.TestCase)**. Pada saat digunakan, maka kelas ini akan membuat sebuah database sebelum dilakukan pengujian dan menjalankan setiap fungsi pengujian pada transaksinya sendiri. Kelas ini punya **Client** yang dapat digunakan untuk mensimulasikan seorang user berinteraksi dengan kode pada level view.

Pengujian aplikasi harus dilakukan pada semua aspek dari kode yang kita buat, bukan pustaka atau fungsi yang disediakan oleh Python atau Django. Contoh Pada model Pengguna, kita tidak perlu melakukan pengujian untuk memastikan bahwa sebuah atribut disimpan benar dalam format Char atau Integer atau Float, karena Charfield atau FloatField adalah bawaan dari Django yang tidak perlu kita uji.

Jadi apa yang harus diuji? Yang harus diuji adalah label (dalam hal ini misalnya First Name, Last Name, Address) dan ukuran dari field yang dialokasikan untuk teks karena ini adalah bagian dari desain yang kita buat dan kemungkinan merupakan sesuatu yang bisa gagal atau berubah diwaktu yang akan datang.

PYTHON



```
class Author(models.Model):
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)
    date_of_birth = models.DateField(null=True, blank=True)
    date_of_death = models.DateField('Died', null=True, blank=True)

    def get_absolute_url(self):
        return reverse('author-detail', args=[str(self.id)])

    def __str__(self):
        return '%s, %s' % (self.last_name, self.first_name)
```

(ref: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Testing>)

Pada kasus diatas, metode `get_absolute_url()` dan `__str__()` harus dicek berperilaku sesuai yang dibutuhkan. Sedangkan fungsi `reverse` yang merupakan bawaan django kita percayai sudah diimplementasikan secara benar.

### Struktur Pengujian

Pengujian aplikasi dilakukan di level app. Django akan menggunakan modul built-in test discovery pada working directory pada semua file yang berpola `test*.py`

**Pertama**, hapus terlebih dahulu file `tests.py` bawaan dari app Django nya, Kemudian siapkan sebuah modul “tests” dan buat file terpisah untuk setiap modul yang dites. File `__init__.py`

adalah file kosong dengan nama khusus untuk menandai tests sebagai sebuah modul (package).

```
└─ app_name
    └─ tests
        ├── __init__.py
        ├── test_forms.py
        ├── test_models.py
        └─ test_views.py
```

Gambar xx. Struktur modul pengujian Django app

#### Tahapan Pengujian Modul

1. Aktifkan virtual envirotnmen
2. Siapkan modul test pada app yang akan diuji, dengan file didalam modul seperti Gambar xx.
3. Tambahkan kode program berikut :

```
from django.test import TestCase

class YourTestClass(TestCase):
    @classmethod
    def setUpTestData(cls):
        print("setUpTestData: Run once to set up non-modified data for all class methods.")
        pass

    def setUp(self):
        print("setUp: Run once for every test method to set up clean data.")
        pass

    def test_false_is_false(self):
        print("Method: test_false_is_false.")
        self.assertFalse(False)

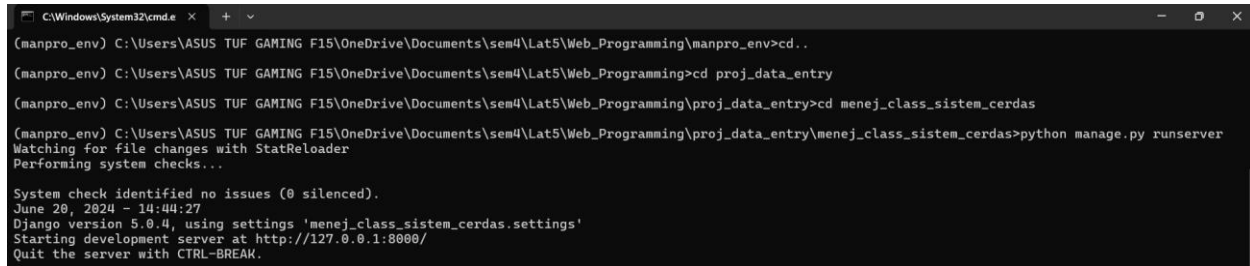
    def test_false_is_true(self):
        print("Method: test_false_is_true.")
        self.assertTrue(False)

    def test_one_plus_one_equals_two(self):
        print("Method: test_one_plus_one_equals_two.")
        self.assertEqual(1 + 1, 2)
```

Penjelasan kode: AssertTrue, AssertFalse, dan AssertEqual adalah standard assertion yang disediakan oleh unittest. Django juga menyediakan beberapa assertion khusus: (cek di <https://docs.djangoproject.com/en/5.0/topics/testing/tools/#assertions>)

4. Jalankan perintah **python manage.py test** di cmd dengan status virtual environmentnya aktif

Hasil menjalankan perintah tersebut:



```
C:\Windows\System32\cmd.exe
(manpro_env) C:\Users\ASUS TUF GAMING F15\OneDrive\Documents\sem4\Lat5\Web_Programming\manpro_env>cd..
(manpro_env) C:\Users\ASUS TUF GAMING F15\OneDrive\Documents\sem4\Lat5\Web_Programming>cd proj_data_entry
(manpro_env) C:\Users\ASUS TUF GAMING F15\OneDrive\Documents\sem4\Lat5\Web_Programming\proj_data_entry>cd menej_class_sistem_cerdas
(manpro_env) C:\Users\ASUS TUF GAMING F15\OneDrive\Documents\sem4\Lat5\Web_Programming\proj_data_entry\menej_class_sistem_cerdas>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 20, 2024 - 14:44:27
Django version 5.0.4, using settings 'menej_class_sistem_cerdas.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Teliti hasil running program, jawab pertanyaan berikut dan lampirkan dalam laporan.

Berapa kali fungsi setUpTestData dipanggil dari menjalankan pengujian?

1. Berapa kali fungsi setup dipanggil dan kapan?
2. Mana fungsi pengujian yang berhasil dijalankan?
3. Mana fungsi pengujian yang gagal dijalankan? Berikan penjelasan singkat kenapa gagal?

5. Lakukan pengujian ulang dengan menambahkan atribut **--verbosity**

**python manage.py test --verbosity 2**

(level verbosity yang dimungkinkan : 0, 1, 2, dan 3 dengan default 1)

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure for 'WEB\_PROGRAMMING' with various files and folders. The code editor shows a Python file named 'test\_form.py' with the following code:

```
1 from django.test import TestCase
2
3 class YourTestClass(TestCase):
4     @classmethod
5     def setUpTestData(cls):
6         print("setUpTestData: Run once to set up non-modified data for all class methods.")
7         pass
8
9     def setUp(self):
10        print("setUp: Run once for every test method to set up clean data.")
11        pass
12
13    def test_false_is_false(self):
14        print("Method: test_false_is_false.")
15        self.assertFalse(False)
16
17    def test_false_is_true(self):
18        print("Method: test_false_is_true.")
19        self.assertTrue(False)
20
21    def test_one_plus_one_equals_two(self):
22        print("Method: test_one_plus_one_equals_two.")
23        self.assertEqual(1 + 1, 2)
24
```

At the bottom of the code editor, there is a status bar with the text 'Ln 2'.

6. Untuk melakukan pengujian yang saling independen pada komputer multiprocessor, pengujian dapat dipercepat dengan menambahkan atribut `-parallel auto`

Kita bisa melakukan pengujian pada modul tertentu dengan menambahkan sebagai argumen setelah kata test

BASH



```
# Run the specified module
python3 manage.py test catalog.tests

# Run the specified module
python3 manage.py test catalog.tests.test_models

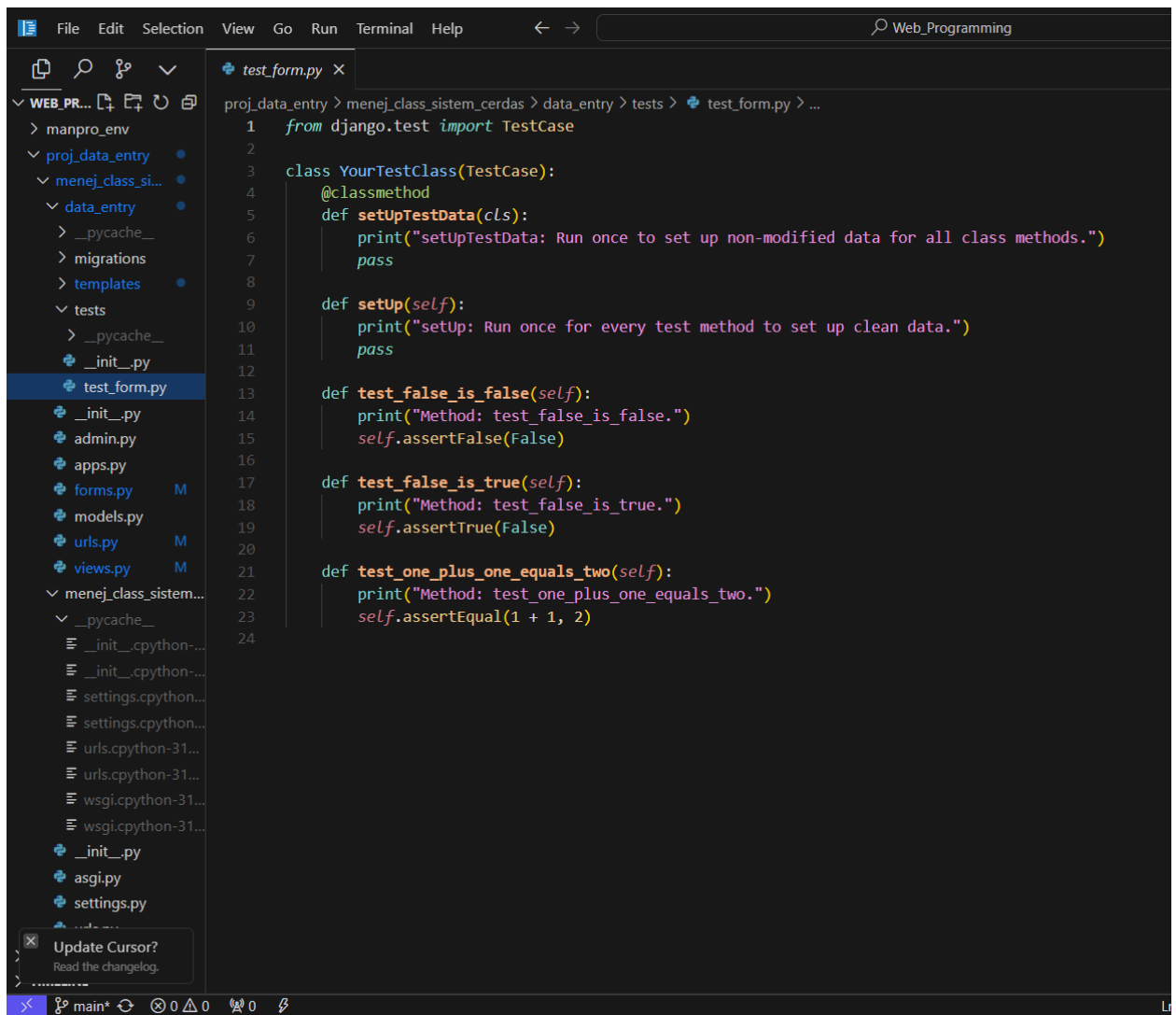
# Run the specified class
python3 manage.py test catalog.tests.test_models.YourTestClass

# Run the specified method
python3 manage.py test
catalog.tests.test_models.YourTestClass.test_one_plus_one_equals_two
```

## Pengujian Model

### Langkah:

1. Siapkan kode untuk pengujian model Pengguna



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'manpro\_env', 'proj\_data\_entry', 'menej\_class\_si...', 'data\_entry', 'tests', and files like 'test\_form.py'. The code editor shows the content of 'test\_form.py'.

```
1 from django.test import TestCase
2
3 class YourTestClass(TestCase):
4     @classmethod
5     def setUpTestData(cls):
6         print("setUpTestData: Run once to set up non-modified data for all class methods.")
7         pass
8
9     def setUp(self):
10        print("setUp: Run once for every test method to set up clean data.")
11        pass
12
13    def test_false_is_false(self):
14        print("Method: test_false_is_false.")
15        self.assertFalse(False)
16
17    def test_false_is_true(self):
18        print("Method: test_false_is_true.")
19        self.assertTrue(False)
20
21    def test_one_plus_one_equals_two(self):
22        print("Method: test_one_plus_one_equals_two.")
23        self.assertEqual(1 + 1, 2)
24
```

Update Cursor?  
Read the changelog.

2. Jalankan perintah python manage.py test

```
C:\Windows\System32\cmd.e X + v
System check identified no issues (0 silenced).

-----
Ran 0 tests in 0.000s
OK

(manpro_env) C:\KULIAH\semester 4\PWB\PWB 6\web_programming\proj_data_entry\menej_class_sistem_cerdas>cd menej_class_sis
tem_cerdas

(manpro_env) C:\KULIAH\semester 4\PWB\PWB 6\web_programming\proj_data_entry\menej_class_sistem_cerdas\menej_class_sistem
_cerdas>cd..

(manpro_env) C:\KULIAH\semester 4\PWB\PWB 6\web_programming\proj_data_entry\menej_class_sistem_cerdas>python manage.py t
est
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
SetupTestData dijalankan
test_email_label dijalankan
.test_object_name_is_email dijalankan
.test_password_max_length dijalankan
.
-----
Ran 3 tests in 0.008s
OK
Destroying test database for alias 'default'...

(manpro_env) C:\KULIAH\semester 4\PWB\PWB 6\web_programming\proj_data_entry\menej_class_sistem_cerdas>
```

### Pengujian View

Validasi perilaku view dilakukan dengan modul pengujian Client. Kelas ini berlaku seperti sebuah web browser dummy yang dapat kita gunakan untuk mensimulasikan GET dan POST pada URL dan mengobservasi respon. Kita bisa melihat segala sesuatu

### Studi Kasus 2:

#### Mahasiswa mampu memahami cara kerja pencarian data

Cara kerja pencarian data (digenerate dengan Chat GPT)

Pencarian data di aplikasi web dilakukan dengan beberapa langkah dasar:

1. Input dari Pengguna: Pengguna memasukkan kata kunci atau parameter pencarian ke dalam kolom pencarian di antarmuka aplikasi web.
2. Pemrosesan Permintaan: Server web menerima permintaan pencarian dari pengguna dan memulai proses pemrosesan.
3. Pengindeksan Data: Data di dalam aplikasi web biasanya diindeks terlebih dahulu. Ini berarti bahwa informasi-informasi tertentu dari setiap item data disimpan dalam struktur yang mempercepat pencarian. Pengindeksan dapat dilakukan dengan berbagai cara, termasuk membangun struktur data seperti indeks terbalik (inverted index) untuk teks atau menggunakan database yang dioptimalkan untuk pencarian.
4. Pencocokan: Pada tahap ini, sistem mencocokkan kata kunci yang dimasukkan oleh pengguna dengan data yang telah diindeks. Ini bisa menjadi proses yang kompleks, terutama jika ada banyak data yang perlu dipertimbangkan atau jika diterapkan dalam konteks yang lebih rumit seperti pencarian berbasis teks atau pencarian dalam basis data terstruktur.

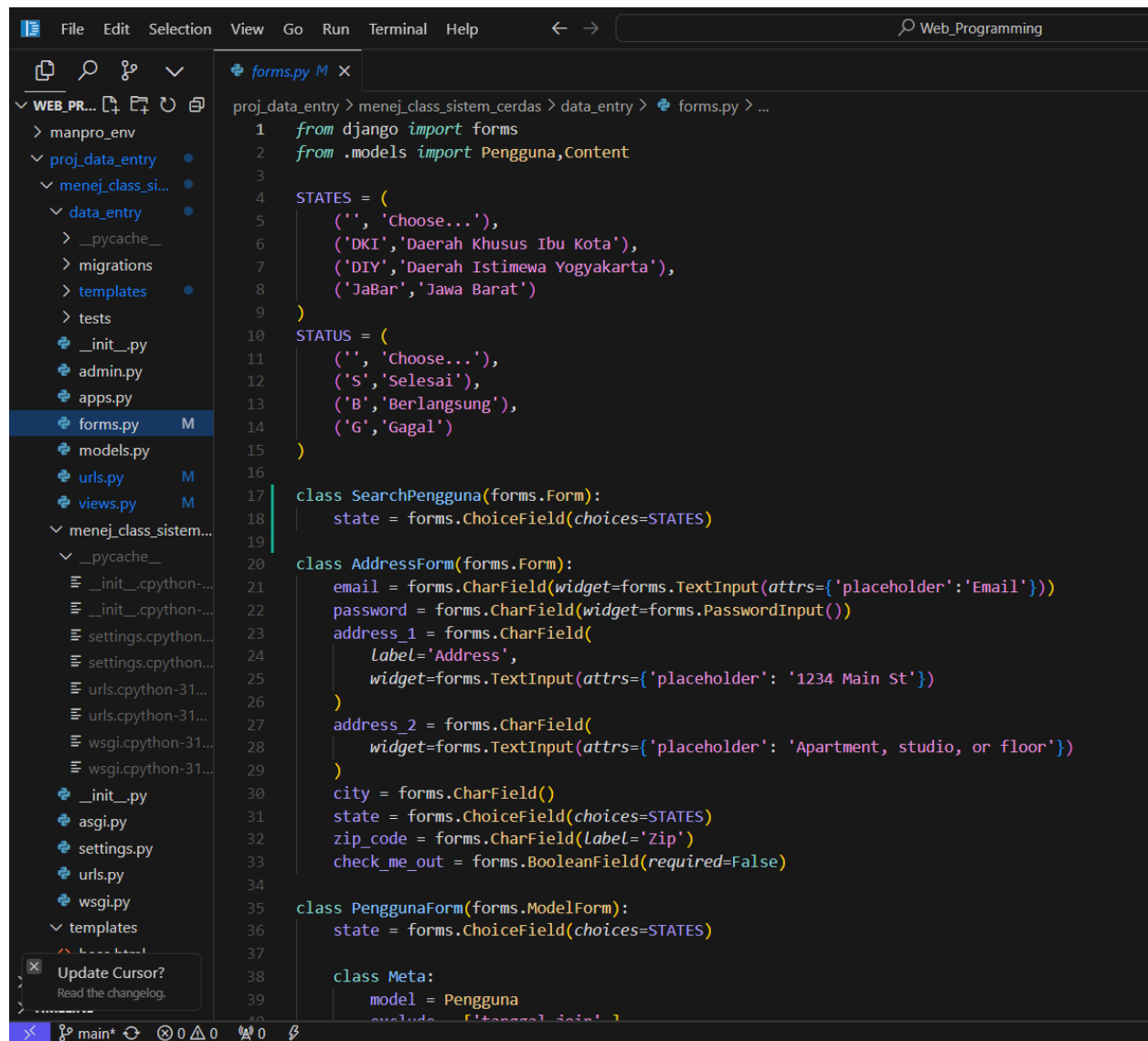


5. **Ranking:** Setelah data yang cocok ditemukan, mereka sering kali diberi peringkat berdasarkan relevansi. Metrik relevansi dapat bervariasi tergantung pada jenis aplikasi dan preferensi pengembang. Misalnya, dalam pencarian web, algoritma seperti PageRank digunakan untuk memberikan peringkat pada halaman-halaman web berdasarkan otoritas dan relevansi mereka terhadap kata kunci.
6. **Pengembalian Hasil:** Setelah peringkat ditentukan, hasil pencarian yang paling relevan biasanya dikembalikan ke pengguna dalam antarmuka pengguna aplikasi web. Ini bisa berupa daftar item yang sesuai dengan kata kunci atau informasi terkait lainnya.
7. **Pengoptimalan:** Proses pencarian ini sering kali dioptimalkan secara terus-menerus untuk meningkatkan kinerja, akurasi, dan kecepatan pencarian. Ini bisa melibatkan penggunaan teknik-teknik seperti caching, pembaruan indeks secara berkala, atau penggunaan algoritma pencarian yang lebih canggih.

Penting untuk dicatat bahwa implementasi pencarian data di aplikasi web dapat bervariasi tergantung pada jenis aplikasi, skala data, dan kebutuhan pengguna. Beberapa aplikasi web mungkin memerlukan teknik pencarian yang lebih kompleks daripada yang lain tergantung pada kompleksitas data dan kasus penggunaan.

#### Tahapan Implementasi:

1. **Memastikan keberadaan sumberdata**  
Pencarian data kedalam database dapat dilakukan dengan melakukan query pada sebuah tabel atau sejumlah tabel didalam database.  
Pada praktikum sebelumnya telah diimplementasikan model pengguna dan konten. Pada model pengguna kita dapat melakukan pencarian daftar pengguna dengan kriteria tertentu misalnya asal kota (city) dan asal propinsi (state) atau kriteria yang lain. Pada model Content, misalnya kita bisa melakukan pencarian konten berdasarkan Authornya.
2. **Merancang dan mengimplementasikan antarmuka pencarian**  
Pencarian kedalam database dilakukan menggunakan kata kunci. Kata kunci dapat berupa satu atau lebih kriteria untuk melakukan pencarian. Misalnya pada model Pengguna, kita bisa mencari semua pengguna yang berasal dari suatu kota/city yang keywordnya bisa kita sajikan dalam bentuk list yang dapat diakses pengguna. Untuk mengimplementasikan fungsi pencarian tersebut maka perlu dibuat rancangan Form dan komponennya terlebih dahulu dan diletakkan di file forms.py untuk menampilkan form dan menangani pencarian maka perlu dilakukan implementasi di views.py. Link URL untuk mengakses fungsi pencarian didaftarkan di urls.py
  1. Mendefinisikan komponen untuk pencarian di forms.py



```
1 from django import forms
2 from .models import Pegguna, Content
3
4 STATES = (
5     ('', 'Choose...'),
6     ('DKI', 'Daerah Khusus Ibu Kota'),
7     ('DIY', 'Daerah Istimewa Yogyakarta'),
8     ('JaBar', 'Jawa Barat')
9 )
10 STATUS = (
11     ('', 'Choose...'),
12     ('S', 'Selesai'),
13     ('B', 'Berlangsung'),
14     ('G', 'Gagal')
15 )
16
17 class SearchPegguna(forms.Form):
18     state = forms.ChoiceField(choices=STATES)
19
20 class AddressForm(forms.Form):
21     email = forms.CharField(widget=forms.TextInput(attrs={'placeholder': 'Email'}))
22     password = forms.CharField(widget=forms.PasswordInput())
23     address_1 = forms.CharField(
24         label='Address',
25         widget=forms.TextInput(attrs={'placeholder': '1234 Main St'})
26     )
27     address_2 = forms.CharField(
28         widget=forms.TextInput(attrs={'placeholder': 'Apartment, studio, or floor'})
29     )
30     city = forms.CharField()
31     state = forms.ChoiceField(choices=STATES)
32     zip_code = forms.CharField(label='Zip')
33     check_me_out = forms.BooleanField(required=False)
34
35 class PeggunaForm(forms.ModelForm):
36     state = forms.ChoiceField(choices=STATES)
37
38     class Meta:
39         model = Pegguna
40         exclude = ('tanggal_dipin', )
```

2. Mendefinisikan penanganan pencarian di fungsi search\_pengguna\_by\_state()  
Di bagian awal, didefinisikan terlebih dahulu form untuk melakukan pencarian  
Jika user mengirimkan data melalui form dengan request method POST maka
  1. Ambil keyword pencari dari form
  2. Lakukan kueri kedalam database
  3. Kirimkan data ke template.
  4. Lakukan pemanggilan fungsi render

```
File Edit Selection View Go Run Terminal Help
Web_Programming

views.py M x
proj_data_entry > menej_class_sistem_cerdas > data_entry > views.py > ...
1 from django.shortcuts import render
2 from .forms import AddressForm, PenggunaForm, ContentForm, SearchPengguna
3 from .models import Pengguna, Content
4 from django.http import JsonResponse
5
6 # Create your views here.
7 def set_data_entry(request):
8     form = AddressForm()
9     context = {
10         'form': form,
11     }
12     return render(request, 'data_entry/input_data_1.html', context)
13
14 def set_pengguna(request):
15     list_pengguna = Pengguna.objects.all()
16     context = None
17     form = PenggunaForm(None)
18     if request.method == "POST":
19         form = PenggunaForm(request.POST)
20         if form.is_valid():
21             form.save()
22             list_pengguna = Pengguna.objects.all()
23             context = {
24                 'form': form,
25                 'list_pengguna': list_pengguna,
26             }
27             return render(request, 'data_entry/input_data_1.html', context)
28     else:
29         list_pengguna = Pengguna.objects.all()
30         context = {
31             'form': form,
32             'list_pengguna': list_pengguna,
33         }
34         return render(request, 'data_entry/input_data_1.html', context)
35
36 def view_pengguna(request, id):
37     try:
38         pengguna = Pengguna.objects.get(pk=id)
39         return render(request, 'data_entry/pengguna_detail.html', {'user_id': pengguna.id})
40     except Pengguna.DoesNotExist:
```

3. Mengimplementasikan template untuk menampilkan form dan tabel hasil pencarian

FileEditSelectionViewGoRunTerminalHelpWeb\_Programming

list\_pengguna.html U x

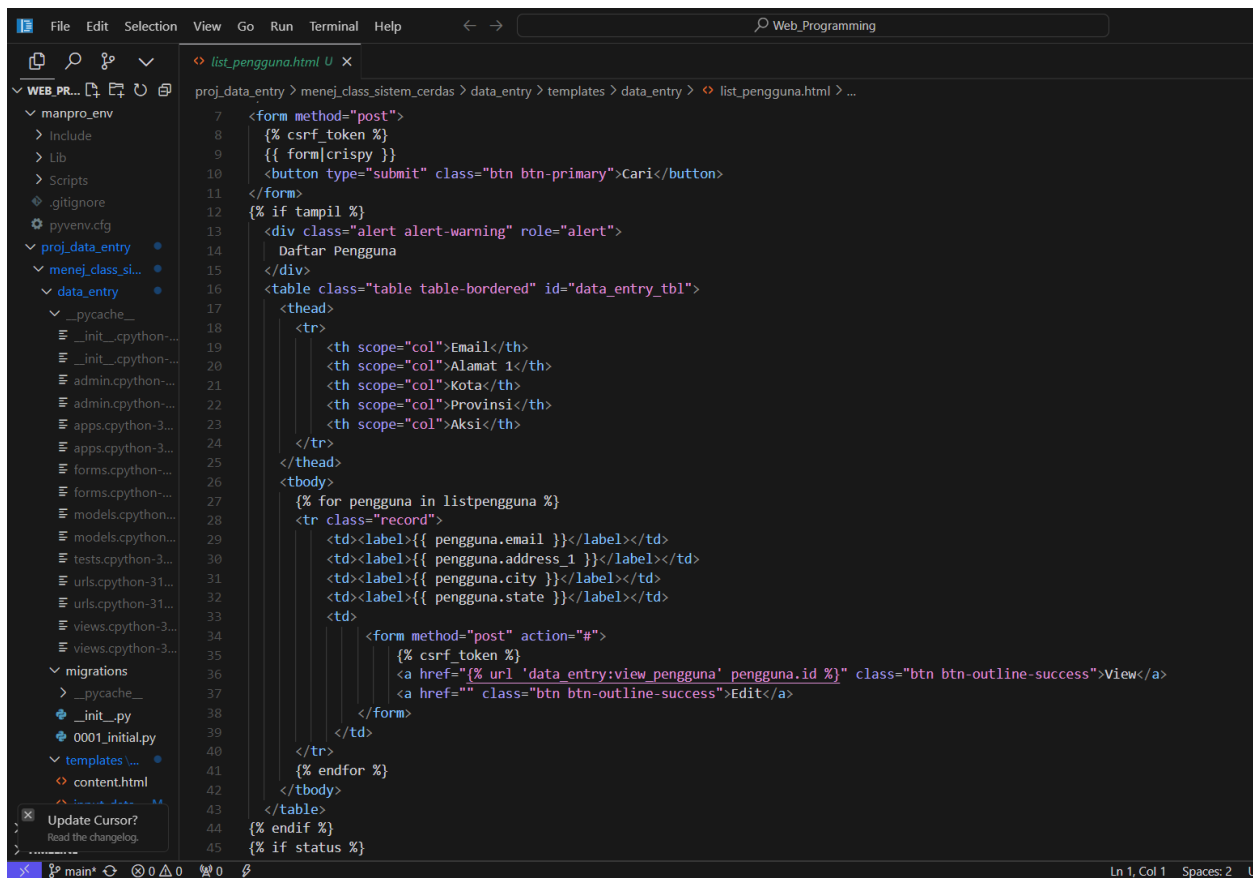
proj\_data\_entry > menjej\_class\_sistem\_cerdas > data\_entry > templates > data\_entry > list\_pengguna.html > ...

WEB\_PR...  
manpro.env  
Include  
Lib  
Scripts  
.gitignore  
pyvenv.cfg  
proj\_data\_entry  
menjej\_class\_si...  
data\_entry  
\_\_pycache\_\_  
\_\_init\_\_.cpython-...  
\_\_init\_\_.cpython-...  
admin.cpython-...  
admin.cpython-...  
apps.cpython-3...  
apps.cpython-3...  
forms.cpython-...  
forms.cpython-...  
models.cpython-...  
models.cpython-...  
tests.cpython-3...  
urls.cpython-31...  
urls.cpython-31...  
views.cpython-3...  
views.cpython-3...  
migrations  
\_\_pycache\_\_  
\_\_init\_\_.py  
0001\_initial.py  
templates\...  
content.html

```
7 <form method="post">
8     {% csrf_token %}
9     {{ form|crispy }}
10    <button type="submit" class="btn btn-primary">Cari</button>
11  </form>
12  {% if tampil %}
13    <div class="alert alert-warning" role="alert">
14      Daftar Pengguna
15    </div>
16    <table class="table table-bordered" id="data_entry_tbl">
17      <thead>
18        <tr>
19          <th scope="col">Email</th>
20          <th scope="col">Alamat 1</th>
21          <th scope="col">Kota</th>
22          <th scope="col">Provinsi</th>
23          <th scope="col">Aksi</th>
24        </tr>
25      </thead>
26      <tbody>
27        {% for pengguna in listpengguna %}
28          <tr class="record">
29            <td><label>{{ pengguna.email }}</label></td>
30            <td><label>{{ pengguna.address_1 }}</label></td>
31            <td><label>{{ pengguna.city }}</label></td>
32            <td><label>{{ pengguna.state }}</label></td>
33            <td>
34              <form method="post" action="#">
35                {% csrf_token %}
36                <a href="{% url 'data_entry:view_pengguna' pengguna.id %}" class="btn btn-outline-success">View</a>
37                <a href="#" class="btn btn-outline-success">Edit</a>
38              </form>
39            </td>
40          </tr>
41        {% endfor %}
42      </tbody>
43    </table>
44    {% endif %}
45    {% if status %}
```

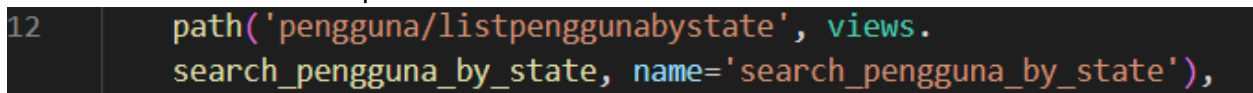
Update Cursor?  
Read the changelog.

Ln 1, Col 1 Spaces: 2 U



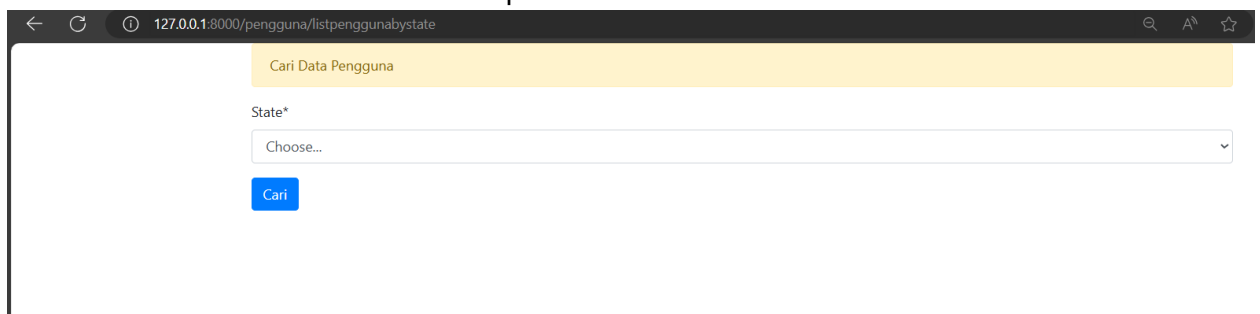
```
7 <form method="post">
8   {% csrf_token %}
9   {{ form|crispy }}
10  <button type="submit" class="btn btn-primary">Cari</button>
11 </form>
12 {% if tampil %}
13 <div class="alert alert-warning" role="alert">
14   Daftar Pengguna
15 </div>
16 <table class="table table-bordered" id="data_entry_tbl">
17   <thead>
18     <tr>
19       <th scope="col">Email</th>
20       <th scope="col">Alamat 1</th>
21       <th scope="col">Kota</th>
22       <th scope="col">Provinsi</th>
23       <th scope="col">Aksi</th>
24     </tr>
25   </thead>
26   <tbody>
27     {% for pengguna in listpengguna %}
28     <tr class="record">
29       <td><label>{{ pengguna.email }}</label></td>
30       <td><label>{{ pengguna.address_1 }}</label></td>
31       <td><label>{{ pengguna.city }}</label></td>
32       <td><label>{{ pengguna.state }}</label></td>
33       <td>
34         <form method="post" action="#">
35           {% csrf_token %}
36           <a href="{% url 'data entry:view_pengguna' pengguna.id %}" class="btn btn-outline-success">View</a>
37           <a href="#" class="btn btn-outline-success">Edit</a>
38         </form>
39       </td>
40     </tr>
41     {% endfor %}
42   </tbody>
43 </table>
44 {% endif %}
45 {% if status %}
```

4. Mendaftarkan fungsi pencarian di urls.py  
Tambahkan path untuk akses menu search



```
12 path('pengguna/listpenggunabystate', views.  
search_pengguna_by_state, name='search_pengguna_by_state'),
```

3. Run server dan akses url untuk akses pencarian



Pencarian berhasil

127.0.0.1:8000/pengguna/listpenggunabystate

Cari Data Pengguna

State\*

Daerah Khusus Ibu Kota

Cari

Daftar Pengguna

Email	Alamat 1	Kota	Provinsi	Aksi
abdul90@gmail.com	mdicche	Jakarta	DKI	<div>ViewEdit</div>
abdul90@gmail.com	mdicche	Jakarta	DKI	<div>ViewEdit</div>
abdul91@gmail.com	mdicche	Jakarta	DKI	<div>ViewEdit</div>
wayan@gmail.com	JL KEBON NANAS SELATAN 1 RT/RW 001/008	KOTA JAKARTA TIMUR	DKI	<div>ViewEdit</div>
trisnaardika27@gmail.com	JL KEBON NANAS SELATAN 1 RT/RW 001/008	KOTA JAKARTA TIMUR	DKI	<div>ViewEdit</div>