الجمهورية اليمنية

جامعة إب

كلية العلوم

قسم علوم الحاسوب وتقنية المعلومات

تكليف مقرر

هندسة برمجيات - عملي

Software Engineering

المحاضرة السابعة

عمل الطالب :

أسامة سعيد محمد حمود سعيد - مجموعة A

إشراف :

م. مالك المصنف

2025 - 2026

1- أنظمة ال Authentication :

- JWT Authentication System (JSON Web Token) :

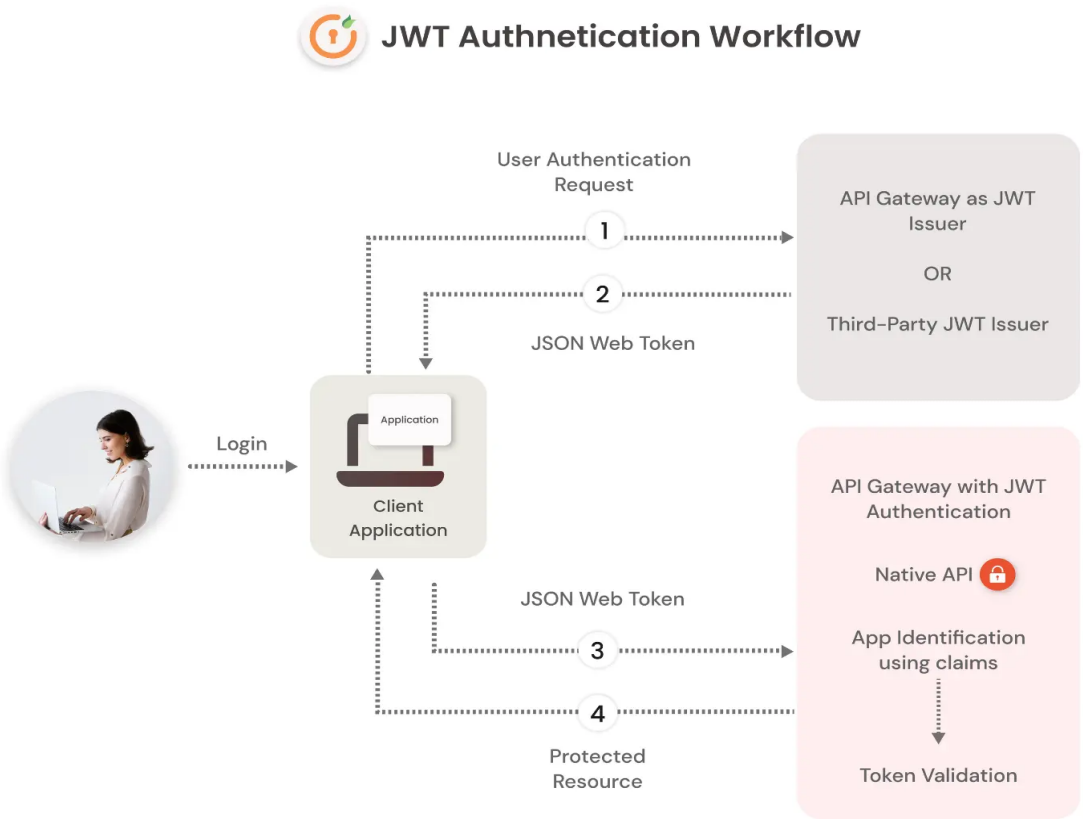
- السيرفر يُنشئ JWT يتكون من Header + Payload + Signature
- يُرسل للعميل بدون تخزينه في السيرفر.
- كل طلب لاحق يحتوي علىJWT ، والسيرفر يتحقق من التوقيع فقط (لا يحتاج قاعدة بيانات).

مميزاته :

- لا يحتاج تخزين في السيرفر (stateless)
- مناسب جدًا لـ Microservices وDistributed Systems
- يحتوي معلومات المستخدم (claims) داخل التوكن نفسه.
- أداء عالي في التحقق (فقط تحقق من التوقيع).

عيوبه :

- حجم التوكن كبير (يزيد من حجم الطلبات).
- لا يمكن إلغاء JWT بسهولة إلا بآليات إضافية (blacklist / short expiry + refresh token)
- إذا انكشف سر التوقيع (secret key) كل التوكنات معرضة للخطر.



- **Session Authentication System :**

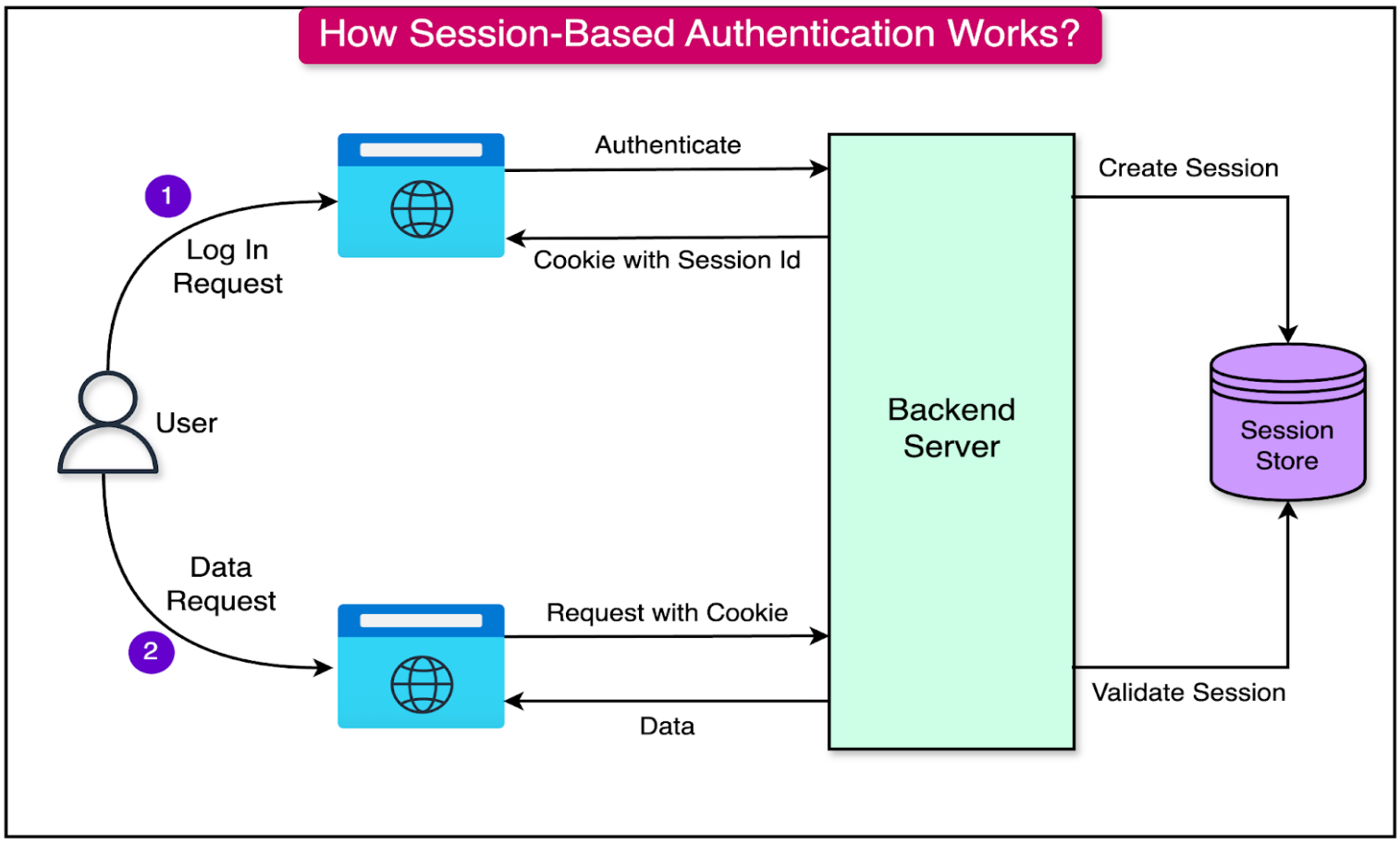
- يعتمد على تخزين الجلسة (session) في السيرفر (عادةً داخل قاعدة بيانات أو في الذاكرة).
- يرسل السيرفر Session ID للعميل ويخزنها في Cookie
- كل طلب لاحق يرسل نفس الـ Session ID ويتم التحقق منه بالسيرفر.

مميزاته :

- آمن جدًا إذا تم تفعيل HTTPS + CSRF Tokens
- سهل التنفيذ وموجود في معظم الـ Framework
- يمكن التحكم في الجلسة من السيرفر.

عيوبه :

- يحتاج السيرفر لتخزين بيانات لكل جلسة (غير عملي عند وجود ملايين المستخدمين).
- غير مناسب لـ APIs و Mobile Apps (لأنه يعتمد على Cookies)



- **Token Authentication System (Basic Token) :**

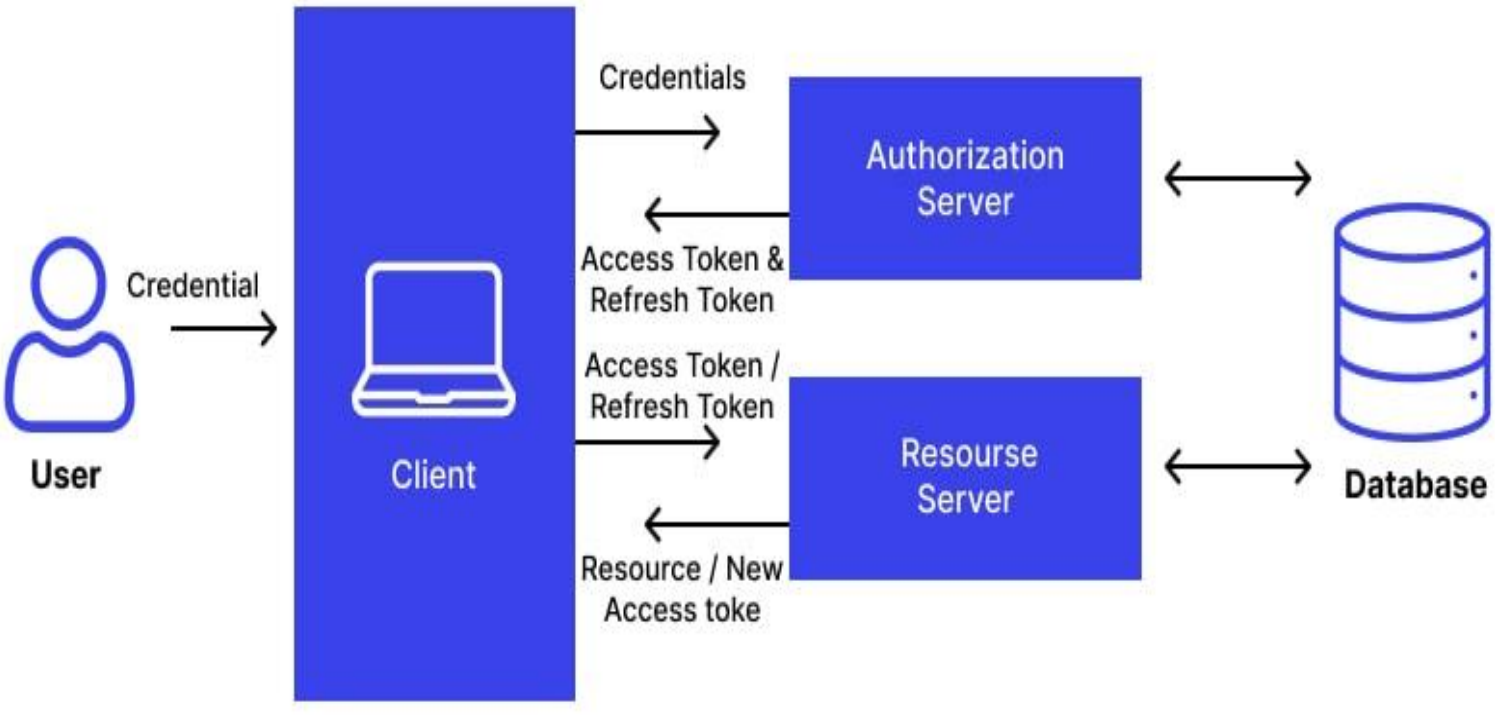
- السيرفر يُنشئ Token (رمز عشوائي) عند تسجيل الدخول ويخزنه في قاعدة بيانات.
- العميل يخزن الـ Token ويرسله مع كل طلب (عادةً في الـ Authorization Header)
- السيرفر يطابقه مع ما هو مخزن في قاعدة البيانات.

مميزاته :

- أبسط من JWT
- يسهل إلغاء أي Token مباشرة من قاعدة البيانات.
- مناسب لو APIs صغيرة أو داخل أنظمة داخلية.

عيوبه :

- السيرفر يحتاج lookup لكل Token
- أقل أداء من JWT
- لا يحتوي بيانات إضافية (مثل User Role أوExpiry) داخل التوكن نفسه



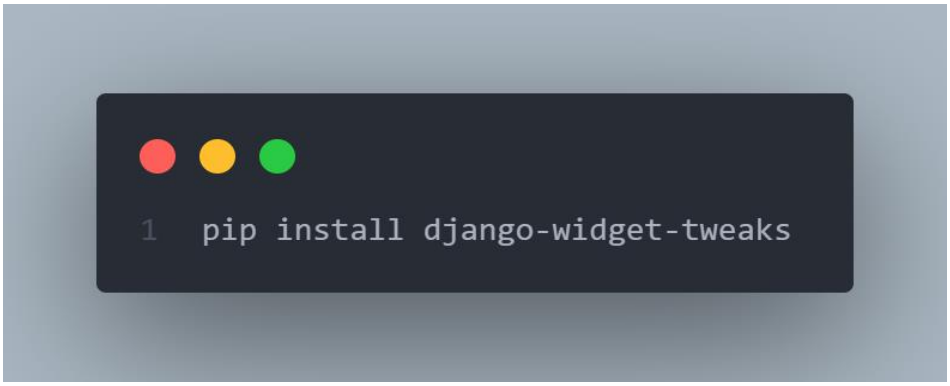
- **مقارنة بين الأنواع الثلاثة :**

وجه المقارنة / النظام	Session Authentication	Token Authentication	JWT Authentication
طريقة التخزين	Session IDمحفوظ في السيرفر + Cookie في العميل	Tokenمحفوظ في قاعدة البيانات والسيرفر + عند العميل	JWTمحفوظ عند العميل فقط (عادةً في LocalStorage أو Cookie)
اعتماد السيرفر	السيرفر لازم يتذكر كل Session (stateful)	السيرفر لازم يخزن كل Token (stateful)	السيرفر لا يخزن شيء عن الجلسات(stateless)
الحجم المرسل	صغير جدًا (Session IDقصير)	متوسط (Tokenقصير عشوائي)	أكبر (JWTيحتويPayload + توقيع)
الأداء	ممتاز للمشاريع الصغيرة (لكن يحتاج ذاكرة لتخزين Sessions)	أبطأ قليلًا لو قاعدة البيانات كبيرة (lookupلكل طلب)	أسرع على السيرفر (لاlookup) لكن أبطأ قليلاً بسبب حجم الـ JWT
الأمان	يعتمد على Cookie + CSRF Protection	يعتمد على سرية الـ Token + HTTPS	يعتمد على توقيع HMAC أو RSA + HTTPS
قابلية التوسع	ضعيف عند الـ load العالي (لأن السيرفر لازم يتذكر كل Session)	متوسط (لكن مازال يحتاج قاعدة بيانات Tokens)	عالي جدًا (Statelessالسيرفر لا يتذكر شيء)
إدارة الجلسة	يمكن إنهاء Session من السيرفر مباشرة	يمكن إلغاء Token بحذفه من DB	لا يمكن إلغاء JWT بسهولة (إلا بـ blacklist أو تقليل زمن الانتهاء)
سهولة الاستخدام	مدمج افتراضيًا في أغلب Frameworks (Django, Laravel)	بسيط وسهل التنفيذ	يتطلب مكتبات للتوقيع والتحقق
حماية من الهجمات	عرضة لـ CSRF (إذا Cookie بدون حماية)	عرضة للسرقة (Token Hijacking)	عرضة للسرقة + مشكلة صعوبة إلغاء التوكن قبل انتهاء صلاحيته
أفضل استخدام	مواقع تقليدية (Web Appsقديمة تعتمد على Cookies)	APIs بسيطة أو داخل نفس الشبكة	REST ،Microservices APIs ،Mobile Apps ، أنظمة موزعة
الهيكلية	**Stateful** السيرفر يخزن كل Session في الذاكرة أو DB	**Stateful** السيرفر يخزن كل Token في قاعدة بيانات	السيرفر لا يخزن أي شيء عن الجلسة
كيف يتم التحقق؟	- عند تسجيل الدخول السيرفر ينشئ Session ويخزنها في DB. - يرسل Session ID للعميل داخل Cookie. - مع كل طلب العميل يرسل الـ Cookie. - السيرفر يبحث عن Session IDويطابقها مع المستخدم.	- عند تسجيل الدخول السيرفر ينشئ Token (رمز عشوائي) ويخزنه في DB. - يرسل الـ Token للعميل (عادةً في Header) - مع كل طلب العميل يرسل الـ Token في Authorization Header - السيرفر يبحث عن الـ Tokenفي DB ويتأكد من صلاحيته.	- عند تسجيل الدخول السيرفر ينشئJWT (Header + Payload + Signature) ويرسله للعميل. - العميل يخزن الـ JWT (في LocalStorage أو Cookie) - مع كل طلب العميل يرسل الـ JWT في Authorization Header. - السيرفر يتحقق من **التوقيع الرقمي** فقط (بدون الحاجة لقاعدة بيانات).

- مكتبة **django-widget-tweaks** :

مرونة أكبر لتخصيص Model Forms و Form Fields داخل القوالب (templates) بدون الحاجة لتعريف كل شيء في الكود.

تثبيت المكتبة :



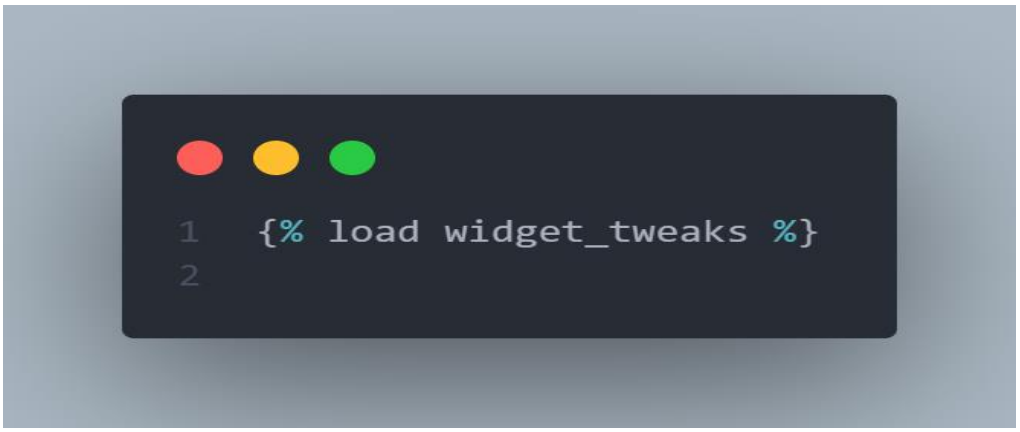
```
pip install django-widget-tweaks
```

الاعداد في sttings.py :

```
INSTALLED_APPS = [
    "unfold",
    'students.apps.StudentsConfig',
    'teachers.apps.TeachersConfig',
    'courses.apps.CoursesConfig',
    'profiles.apps.ProfilesConfig',
    'user',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    "django_cleanup.apps.CleanupConfig", # هنا حيث تفضل ان تكون بعد التطبيقات التي تحتوي على Models
    "widget_tweaks",
]
```

طريقة الاستخدام :

نعمل اول شيء load في أي templates :



```
{% load widget_tweaks %}
```

إضافة CSS Classes أوAttributes :

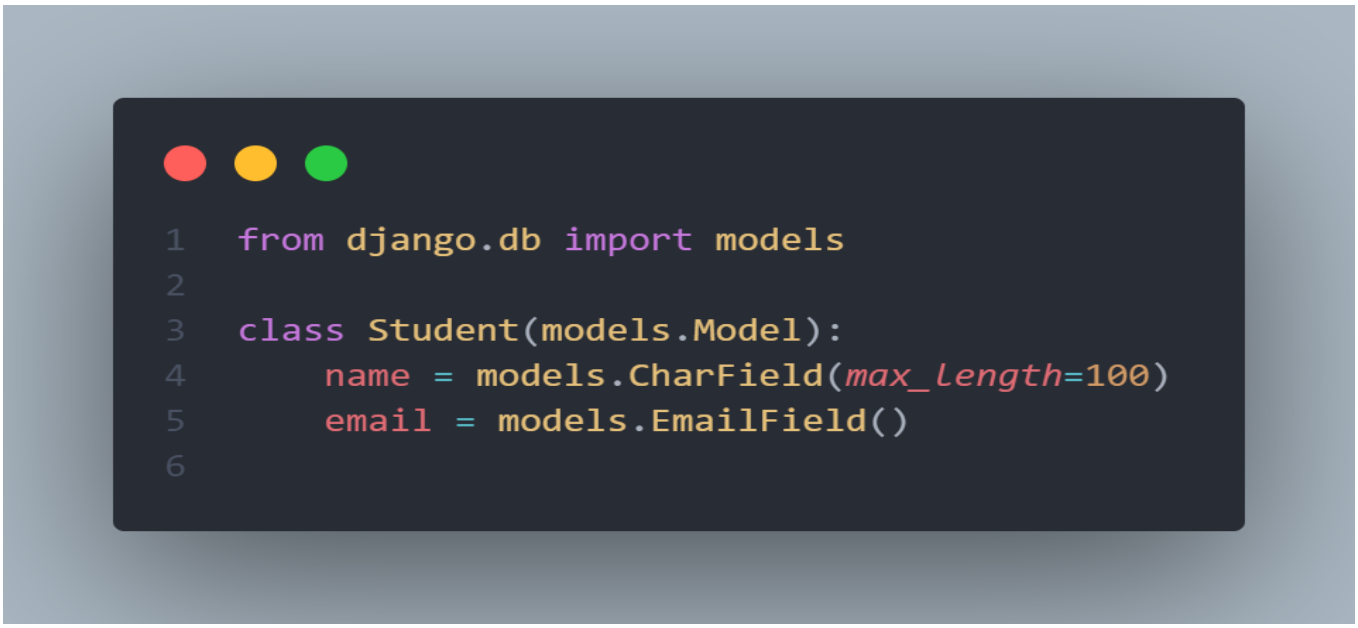
```
{{ form.username|add_class:"form-control my-custom-class" }}
{{ form.password|attr:"placeholder:أدخل كلمة المرور" }}
```

استخدام أكثر من تعديل :

```
{{ form.email|add_class:"form-control"|attr:"placeholder:أدخل بريدك"|attr:"autocomplete:off" }}
```

مثال كامل على ModelForm مع Bootstrap :

ملف models.py

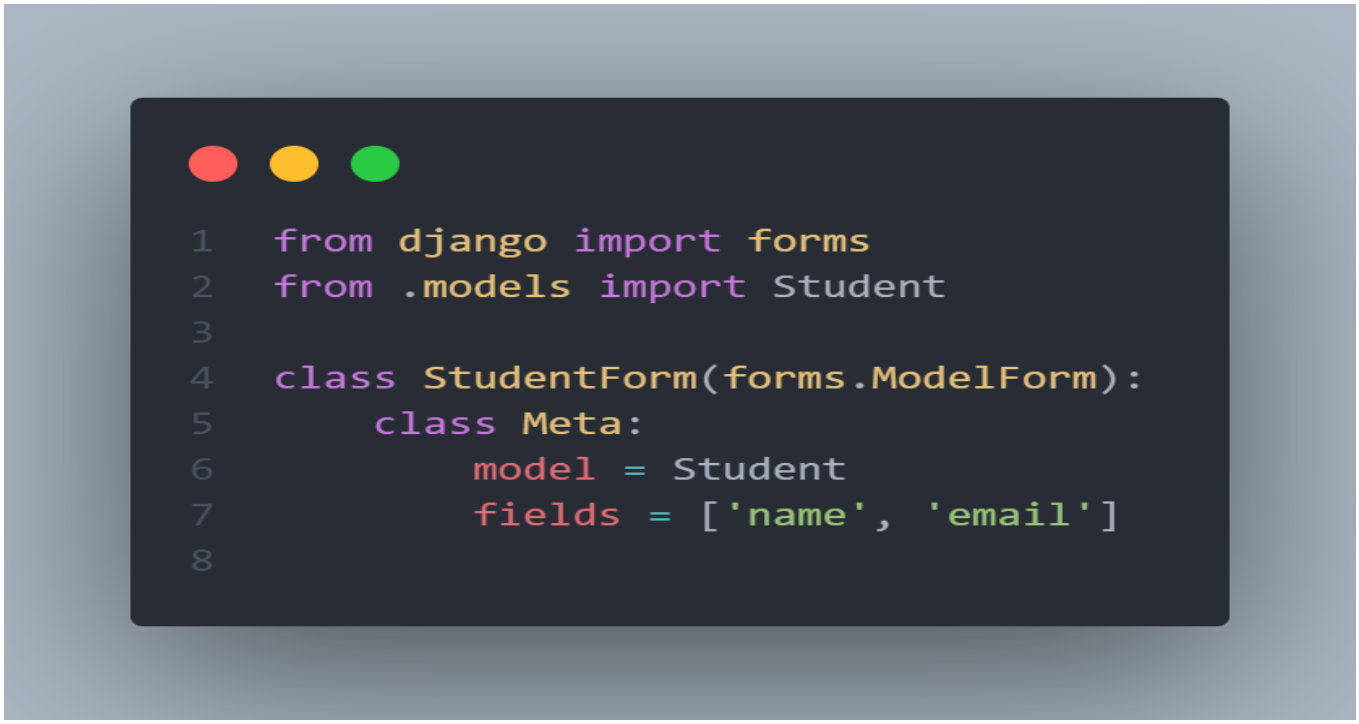


```python
from django.db import models

class Student(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField()

```

ملف forms.py



```python
from django import forms
from .models import Student

class StudentForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = ['name', 'email']

```

ملف template.html

```
{% load widget_tweaks %}

<form method="post">
    {% csrf_token %}

    <div class="mb-3">
        <label for="id_name">الاسم</label>
        {{ form.name|add_class:"form-control"|attr:"placeholder:أدخل اسمك" }}
    </div>

    <div class="mb-3">
        <label for="id_email">البريد الإلكتروني</label>
        {{ form.email|add_class:"form-control"|attr:"placeholder:example@email.com" }}
    </div>

    <button type="submit" class="btn btn-primary">حفظ</button>
</form>

```

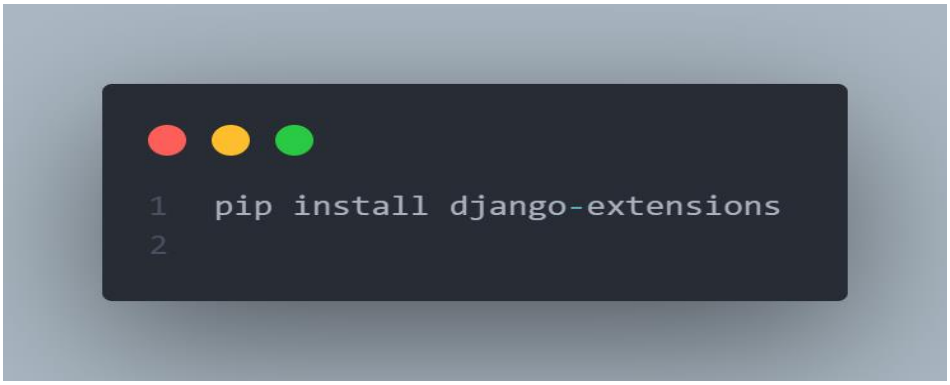
مقارنة بين django-widget-tweaks و django-crispy-forms :

وجه المقارنة / المكتبة	django-widget-tweaks	django-crispy-forms
الفكرة الأساسية	تعديلات مباشرة على عناصر الـ Formداخل الـtemplate (CSS classes, attributes).	مكتبة كاملة لبناء وتنسيق النماذج بطريقة ديناميكية ومرتبة.
طريقة الاستخدام	تعتمد على template filters مثل add_class, attr.	تعتمد على FormHelper و Layout objects (Field, Row, Column).
سهولة التعلم	سهلة جدًا وبسيطة (تضيف كلاس أو Attribute في السطر نفسه)	أعمق وأكثر تعقيدًا لأنها تعطيك تحكم كامل في الـlayout.
المناسب لها	تحسينات سريعة على تصميم الـ forms (مثلاً مع Bootstrap)	أنظمة تحتاج نموذج متكامل مع ترتيب الأعمدة والأسطر والحقول بشكل منظم.
مثال عملي	`{{ form.username	add_class:"form-control"
التوافق	يعمل مع أي CSS framework (Bootstrap, Tailwind, ...).	يدعم Bootstrap 4/5, Tailwind (بإضافات Foundation...)
الاعتماد على السيرفر	لا يغير شيء في الـ Python form نفسه (التغيير فقط في الـ template)	يغير على مستوى الـ form definition باستخدام FormHelper.
مستوى المرونة	عالي جدًا لتعديلات صغيرة ومباشرة.	عالي جدًا لكن يحتاج شغل أكثر (مناسب لمشاريع ضخمة).

- مكتبة **django-extensions** :

مكتبة قوية تضيف مجموعة أوامر إضافية لإدارة مشروع Django ، ومن أهمها الأمر graph_models اللي يستخرج ال ERD (Entity Relationship Diagram) أو Schema للموديلات.

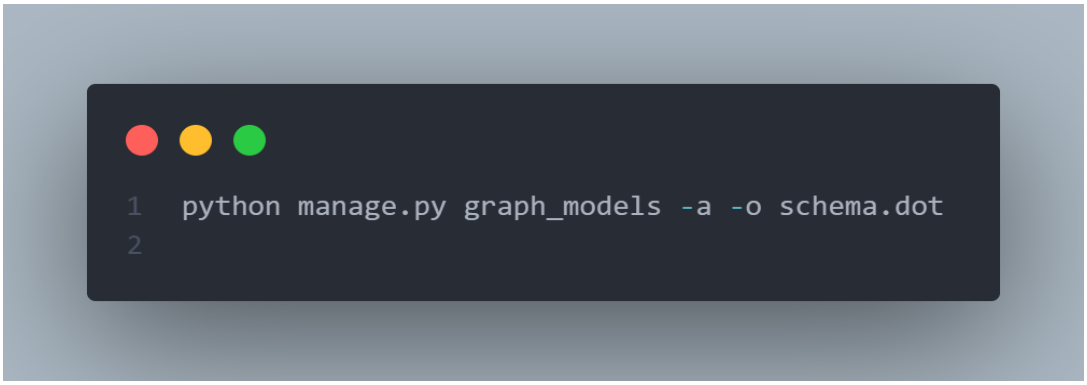
تثبيت المكتبة :



إضافة المكتبة للمشروع :

```
INSTALLED_APPS = [
    "unfold",
    'students.apps.StudentsConfig',
    'teachers.apps.TeachersConfig',
    'courses.apps.CoursesConfig',
    'profiles.apps.ProfilesConfig',
    'user',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    "django_cleanup.apps.CleanupConfig", # هنا حيث تفضل ان تكون بعد التطبيقات التي تحتوي على Models
    "django_extensions",
]
```

استخرج كود الـ Schema :



```
python manage.py graph_models -a -o schema.dot
```

a- = كل التطبيقات (apps)

o- schema.dot= يخرج ملف نصي (DOT format)

نفتح ملف المشروع :

سنجد ملف اسمه schema.dot بجانب المشروع

نفتحه بأي محرر نصوص (Notepad, VS Code) وننسخ المحتوى.

استخدام موقع رسم الـ Schema :

نفتح موقع **Graphviz Online** ثم نلصق المحتوى المنسوخ في مربع الإدخال، ونضغط **Generate** يطلع لنا الـ **Schema Diagram** لمشروعك

ال Schema للتنفيذ حق المحاضرات بااستخدام هذه المكتبة :