

Viva : <https://chatgpt.com/share/68c167a2-0ce0-800b-a776-86244e4bef12>

implement unit 1 - 6 all syllabus



Perfect! We can create a full Django mini-project that covers Unit I to VI practically. I'll provide a **step-by-step** guide with code, explanations, and folder structure, so you can implement all units in one project. Let's do this in a single cohesive project called `mysite` with an app `myapp`.

## Step 0: Setup

1. Create project folder:

bash

Copy code

```
mkdir ~/Desktop/DjangoFullProject
cd ~/Desktop/DjangoFullProject
```

2. Create virtual environment:

bash

Copy code

```
python3 -m venv env
source env/bin/activate # Mac/Linux
env\Scripts\activate   # Windows
```

3. Install Django:

bash

Copy code

```
pip install django
```




## Unit I – Project and App Setup

### 1. Create project

```
bash

django-admin startproject mysite
cd mysite
```

 Copy code

### 2. Create app

```
bash


python manage.py startapp myapp
```

 Copy code

### 3. Add app to `mysite/settings.py`

```
python

INSTALLED_APPS = [
    ...
    'myapp',
]
```

 Copy code

## Unit II – Views and URLs

### 1. Create simple views in `myapp/views.py`

python

 Copy code

```
from django.http import HttpResponse
from django.shortcuts import render

def home(request):
    return HttpResponse("Welcome to MySite Home Page!")

def about(request):
    return HttpResponse("This is the About Page.")

def greet_user(request, username):
    return HttpResponse(f"Hello, {username}!")
```

### 2. Map URLs in `myapp/urls.py`

python

 Copy code

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='home'),
    path('about/', views.about, name='about'),
    path('greet/<str:username>/', views.greet_user, name='greet_user'),
]
```

### 3. Include app URLs in `mysite/urls.py`



### 3. Include app URLs in `mysite/urls.py`

python

 Copy code

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapp.urls')),
]
```

## Unit III – Templates, Debugging and Testing

### 1. Create template folder

arduino


 Copy code

```
myapp/
├── templates/
│   └── myapp/
│       ├── home.html
│       └── about.html
```

### 2. Sample `home.html`

## 2. Sample `home.html`

html


 Copy code

```
<!DOCTYPE html>
<html>
<head>
  <title>Home</title>
</head>
<body>
  <h1>Welcome, {{ username }}!</h1>
  <ul>
    {% for item in items %}
      <li>{{ item }}</li>
    {% endfor %}
  </ul>
  {% if show_footer %}
    <footer>MySite Footer</footer>
  {% endif %}
</body>
</html>
```

## 3. Update `views.py` to render templates

### 3. Update `views.py` to render templates

python

 Copy code


```
def home(request):
    items = ['Django', 'Python', 'Web Development']
    context = {'username': 'Sanuj', 'items': items, 'show_footer': True}
    return render(request, 'myapp/home.html', context)

def about(request):
    return render(request, 'myapp/about.html')
```

## Unit IV – Forms in Django

### 1. Create form in `myapp/forms.py`

python

 Copy code


```
from django import forms

class ContactForm(forms.Form):
    name = forms.CharField(max_length=50)
    email = forms.EmailField()
    message = forms.CharField(widget=forms.Textarea)
```

### 2. Create form view in `views.py`

## 2. Create form view in `views.py`

python


 Copy code

```
from .forms import ContactForm
from django.shortcuts import redirect

def contact(request):
    if request.method == 'POST':
        form = ContactForm(request.POST)
        if form.is_valid():
            print(form.cleaned_data)
            return redirect('home')
    else:
        form = ContactForm()
    return render(request, 'myapp/contact.html', {'form': form})
```

## 3. `contact.html` template


html

 Copy code

```
<h1>Contact Us</h1>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Send</button>
</form>
```

## 4. Add URL

python

 Copy code

#### 4. Add URL

python

 Copy code

```
path('contact/', views.contact, name='contact'),
```

## Unit V – Models, Migrations, and Admin

### 1. Create model in `myapp/models.py`

python

 Copy code

```
from django.db import models

class Product(models.Model):
    name = models.CharField(max_length=100)
    price = models.FloatField()
    description = models.TextField()

    def __str__(self):
        return self.name
```

### 2. Make migrations

bash

 Copy code

```
python manage.py makemigrations
python manage.py migrate
```

### 3. Register model in `myapp/admin.py`





### 3. Register model in `myapp/admin.py`

python


 Copy code

```
from django.contrib import admin
from .models import Product

admin.site.register(Product)
```

### 4. Create superuser

bash


 Copy code

```
python manage.py createsuperuser
```

## Unit VI – Cookies, Sessions, Users and Authentication

### 1. Cookie Example in `views.py`

python

 Copy code

```
def set_cookie(request):
    response = HttpResponse("Cookie Set!")
    response.set_cookie('favorite_color', 'blue')
    return response


def get_cookie(request):
    color = request.COOKIES.get('favorite_color')
    return HttpResponse(f"Favorite Color: {color}")
```

### 2. Session Example



## 2. Session Example

python


 Copy code

```
def set_session(request):
    request.session['username'] = 'Sanuj'
    return HttpResponse("Session Set!")

def get_session(request):
    username = request.session.get('username')
    return HttpResponse(f"Session Username: {username}")
```

## 3. User Authentication

python

 Copy code

```
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required

def user_login(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('home')
    return render(request, 'myapp/login.html')
```

### 3. User Authentication

python

 Copy code

```
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required

def user_login(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('home')
        return render(request, 'myapp/login.html')

@login_required
def user_logout(request):
    logout(request)
    return redirect('home')
```

### Project Folder Structure

bash

 Copy code

```
mysite/
├── mysite/
│   ├── settings.py
│   └── urls.py
```

## ✓ Project Folder Structure

bash

 Copy code

```
mysite/
├── mysite/
│   ├── settings.py
│   ├── urls.py
│   └── ...
├── myapp/
│   ├── templates/myapp/home.html
│   ├── templates/myapp/contact.html
│   ├── forms.py
│   ├── models.py
│   ├── views.py
│   ├── urls.py
│   └── admin.py
└── manage.py
```

This one project covers all Units I-VI:

- **Unit I:** Project & App setup
- **Unit II:** Views & URLs, HTTP requests
- **Unit III:** Templates, loops, if-else, template inheritance
- **Unit IV:** Forms, GET/POST, CSRF
- **Unit V:** Models, Migrations, Admin
- **Unit VI:** Cookies, Sessions, User Auth

## ◆ 2. Template Tags

- **Definition:** Special instructions inside `{% %}` for logic, loops, conditions, inheritance, etc.

### Common Template Tags


Tag	Purpose	Example
<code>{% if %}</code>	Conditional statements	<code>{% if age &gt;= 18 %} Adult {% else %} Minor {% endif %}</code>
<code>{% for %}</code>	Loop over lists	<code>{% for student in students %} {{ student.name }} {% endfor %}</code>
<code>{% block %}</code>	Define blocks in base template	<code>{% block content %} ... {% endblock %}</code>
<code>{% extends %}</code>	Inherit from base template	<code>{% extends "base.html" %}</code>
<code>{% include %}</code>	Include another template	<code>{% include "header.html" %}</code>
<code>{% url 'name' %}</code>	Reverse URL mapping	<code>&lt;a href="{% url 'home' %}"&gt;Home&lt;/a&gt;</code>
<code>{% csrf_token %}</code>	CSRF protection in forms	<code>&lt;form method="post"&gt;{% csrf_token %} &lt;/form&gt;</code>
<code>{% empty %}</code>	Executes if loop is empty	<code>{% for i in items %}{{ i }}{% empty %}No Items{% endfor %}</code>
<code>{% comment %}</code>	Add comments	<code>{% comment %} This is a comment {% endcomment %}</code>
<code>{% static %}</code>	Load static files	<code>&lt;img src="{% static 'images/logo.png' %}"&gt;</code>



Perfect 👍 You already listed the most **common Django template tags** with their purpose and examples. Let me now **implement all of them together in a single working example** so you can directly use it in your Django project.

✓ `base.html`

django

 Copy code

```
<!DOCTYPE html>
<html>
<head>
  <title>Django Template Tags Demo</title>
  {% load static %}
  <link rel="stylesheet" href="{% static 'css/style.css' %}">
</head>
<body>

  {% include "header.html" %}

  <div class="container">
    {% block content %}
    <!-- Child templates will replace this block -->
    {% endblock %}
  </div>

  {% include "footer.html" %}

</body>
</html>
```



✓ index.html

django

Copy code

```
{% extends "base.html" %}

{% block content %}

<h2>Common Template Tags Demo</h2>

<!-- 1. if / else -->
{% if age >= 18 %}
    <p>You are an <strong>Adult</strong>.</p>
{% else %}
    <p>You are a <strong>Minor</strong>.</p>
{% endif %}

<!-- 2. for loop + empty -->
<h3>Student List:</h3>
<ul>
{% for student in students %}
    <li>{{ forloop.counter }}. {{ student.name }}</li>
{% empty %}
    <li>No students available.</li>
{% endfor %}
</ul>

<!-- 3. url tag -->
<p>Go back to <a href="{% url 'home' %}">Home</a></p>

<!-- 4. csrf_token -->
<form method="post">
    {% csrf_token %}
    <input type="text" name="username" placeholder="Enter Name">
    <button type="submit">Submit</button>
</form>

<!-- 5. comment -->
{% comment %}
This is a comment and will not appear in the HTML output.
{% endcomment %}

<!-- 6. static tag -->


{% endblock %}
```

✓ header.html



html


Copy code

```
<!-- 6. static tag -->


{% endblock %}
```

#### ✓ header.html


html

 Copy code

```
<header>
  <h1>Welcome to My Django App</h1>
  <nav>
    <a href="{% url 'home' %}">Home</a> |
    <a href="{% url 'about' %}">About</a>
  </nav>
</header>
```

#### ✓ footer.html

html

 Copy code

```
<footer>
  <p>© 2025 My Django Project</p>
</footer>
```

#### ✓ views.py



✓ views.py

python

Copy code

```
from django.shortcuts import render

def home(request):
    students = [
        {"name": "Alice"},
        {"name": "Bob"},
        {"name": "Charlie"},
    ]
    context = {
        "age": 20,
        "students": students
    }
    return render(request, "index.html", context)


def about(request):
    return render(request, "about.html")
```

### 🔑 Covered Tags:

- {% if %} / {% else %} → Conditional rendering
- {% for %} / {% empty %} → Loop with empty fallback
- {% block %} → Content placeholder in base template
- {% extends %} → Inheriting base.html
- {% include %} → Reuse header/footer templates
- {% url %} → Reverse URL resolution
- {% csrf\_token %} → Security in forms
- {% comment %} → Template comments
- {% static %} → Load static files

## Django Project Folder Layout Explained

pgsql

 Copy code

```
FirstProject/
├── venv/           ← Virtual Environment (optional but recommended)
├── first/          ← Project folder (settings & configurations)
│   ├── __init__.py ← Marks directory as a Python package
│   ├── asgi.py     ← Entry point for ASGI servers (async deployment)
│   ├── settings.py ← Project settings (database, apps, middleware, etc.)
│   ├── urls.py     ← Project-level URL mappings (routes)
│   └── wsgi.py     ← Entry point for WSGI servers (deployment)
├── myapp/          ← Application folder (your actual app logic)
│   ├── __init__.py ← Marks directory as Python package
│   ├── admin.py    ← Register models for Django Admin
│   ├── apps.py     ← App configuration file
│   ├── migrations/ ← Database migration files (auto-created when models change)
│   │   └── __init__.py
│   ├── models.py   ← Database models (tables as Python classes)
│   ├── tests.py    ← Unit tests for your app
│   └── views.py    ← Business logic (functions/classes returning responses)
└── manage.py       ← Command-line utility to runserver, migrations, etc.
```