now implement these practicals/Experments : List of practicals

- Set up a Django project, create an app, and execute basic manage.py commands.
- Organize the project structure and explore the initial settings.
- Develop views for different functionalities (e.g., displaying content, handling forms).
- Map views to specific URLs and handle various HTTP requests.
- Create and integrate templates in Django for different pages or components.
- Debug a Django application, utilizing Django's built-in debugging features.
- Implement basic tests to check functionality across different parts of the app.

List of Practicals / Experiments:

Session 2025-26 Page:1/2

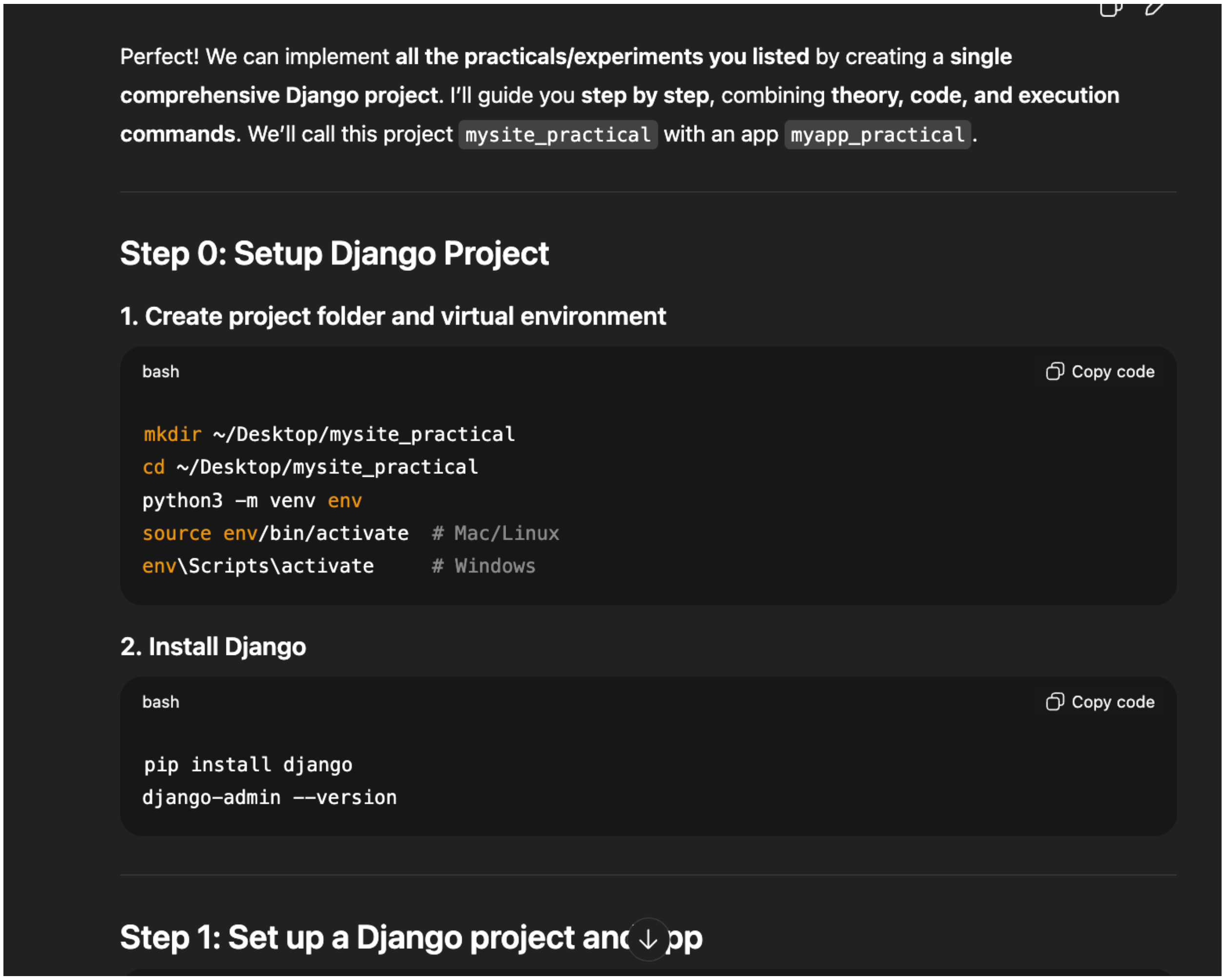
Text Books:

1. BUILDING WEBSITES WITH DJANGO by AWANISH RANJAN, BPB PUBLICATIONS

References:

1. DESIGNING MICROSERVICES USING DJANGO by SHAYANK JAIN, BPB PUBLICATIONS

- Build forms using Django for user input or data submission.
- Validate form data and implement CSRF protection to ensure security.
- Design and create models representing different data entities.
- Apply migrations to reflect changes in the database schema.
- Utilize Django Admin to manage and interact with model data.
- Implement session-based functionality using cookies for user interaction.
- Manage user creation, authentication, and permissions within the Django system.
- Set up login/logout functionality and secure specific views with authentication checks.



Perfect! We can implement **all the practicals/experiments you listed** by creating a **single comprehensive Django project**. I'll guide you **step by step**, combining **theory, code, and execution commands**. We'll call this project `mysite_practical` with an app `myapp_practical`.

Step 0: Setup Django Project

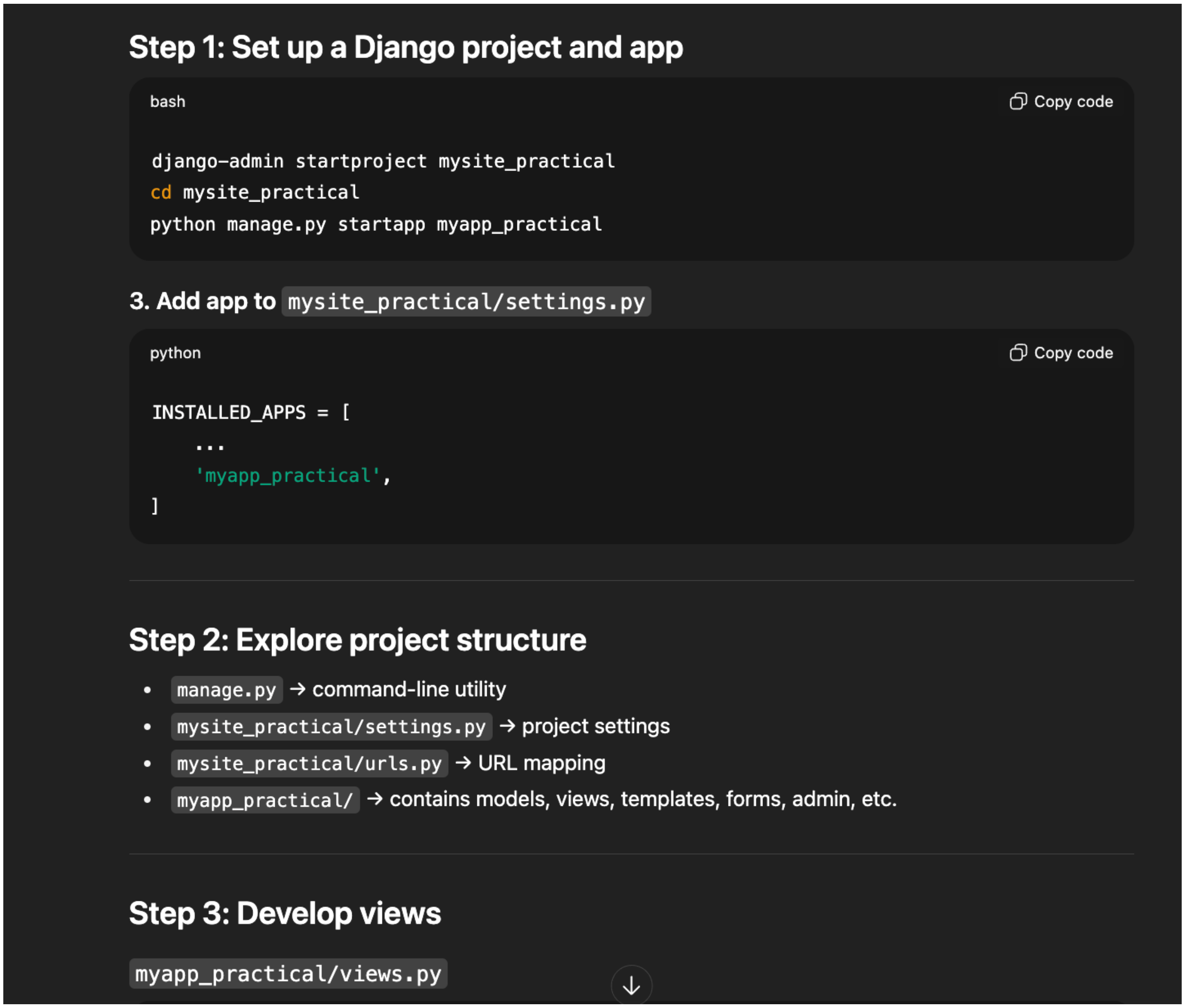
1. Create project folder and virtual environment

```bash
mkdir ~/Desktop/mysite_practical
cd ~/Desktop/mysite_practical
python3 -m venv env
source env/bin/activate  # Mac/Linux
env\Scripts\activate     # Windows
```

2. Install Django

```bash
pip install django
django-admin --version
```

Step 1: Set up a Django project and pp



Step 1: Set up a Django project and app

```bash
django-admin startproject mysite_practical
cd mysite_practical
python manage.py startapp myapp_practical
```

3. Add app to `mysite_practical/settings.py`

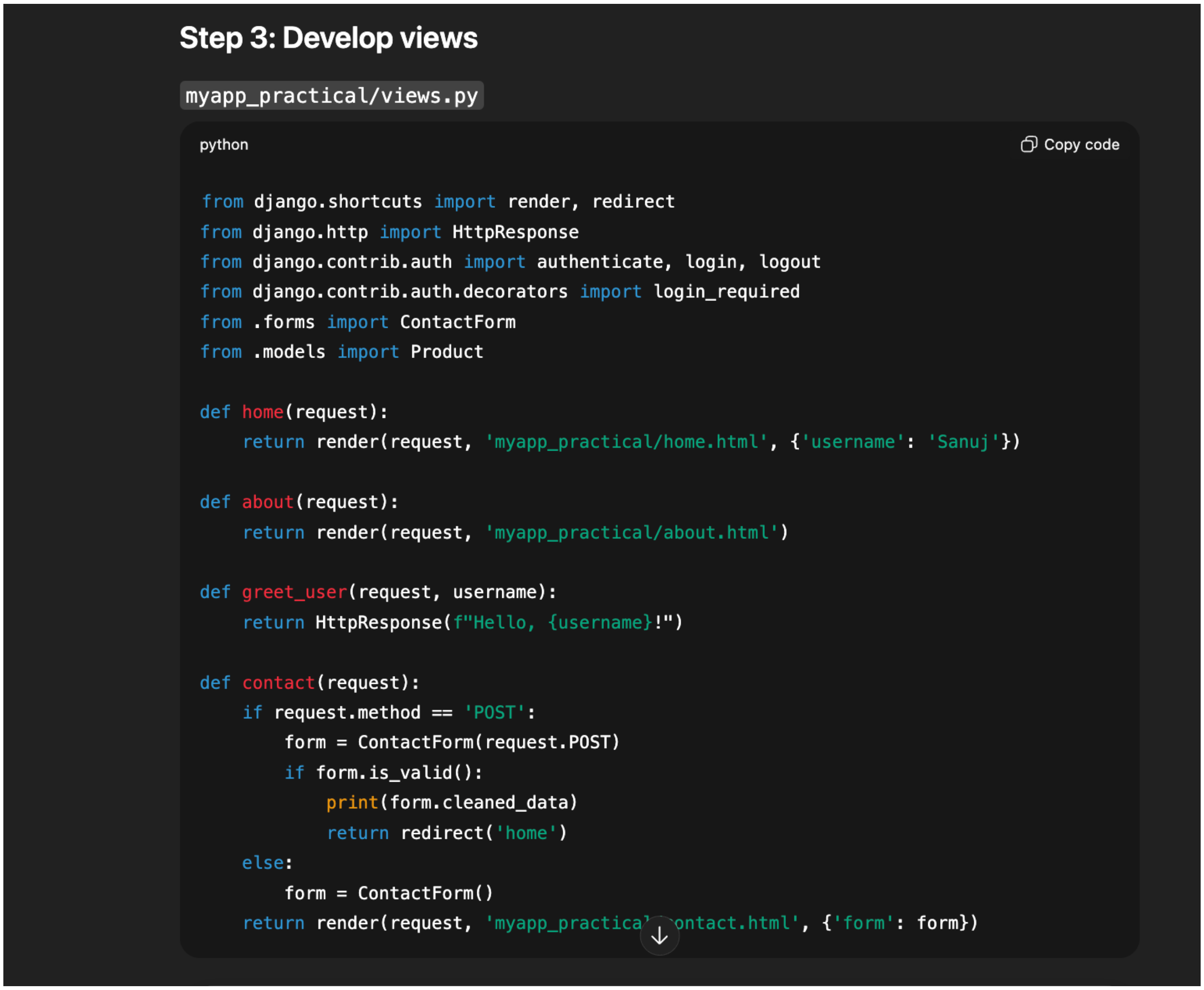
```python
INSTALLED_APPS = [
    ...
    'myapp_practical',
]
```

Step 2: Explore project structure

- `manage.py` → command-line utility
- `mysite_practical/settings.py` → project settings
- `mysite_practical/urls.py` → URL mapping
- `myapp_practical/` → contains models, views, templates, forms, admin, etc.

Step 3: Develop views

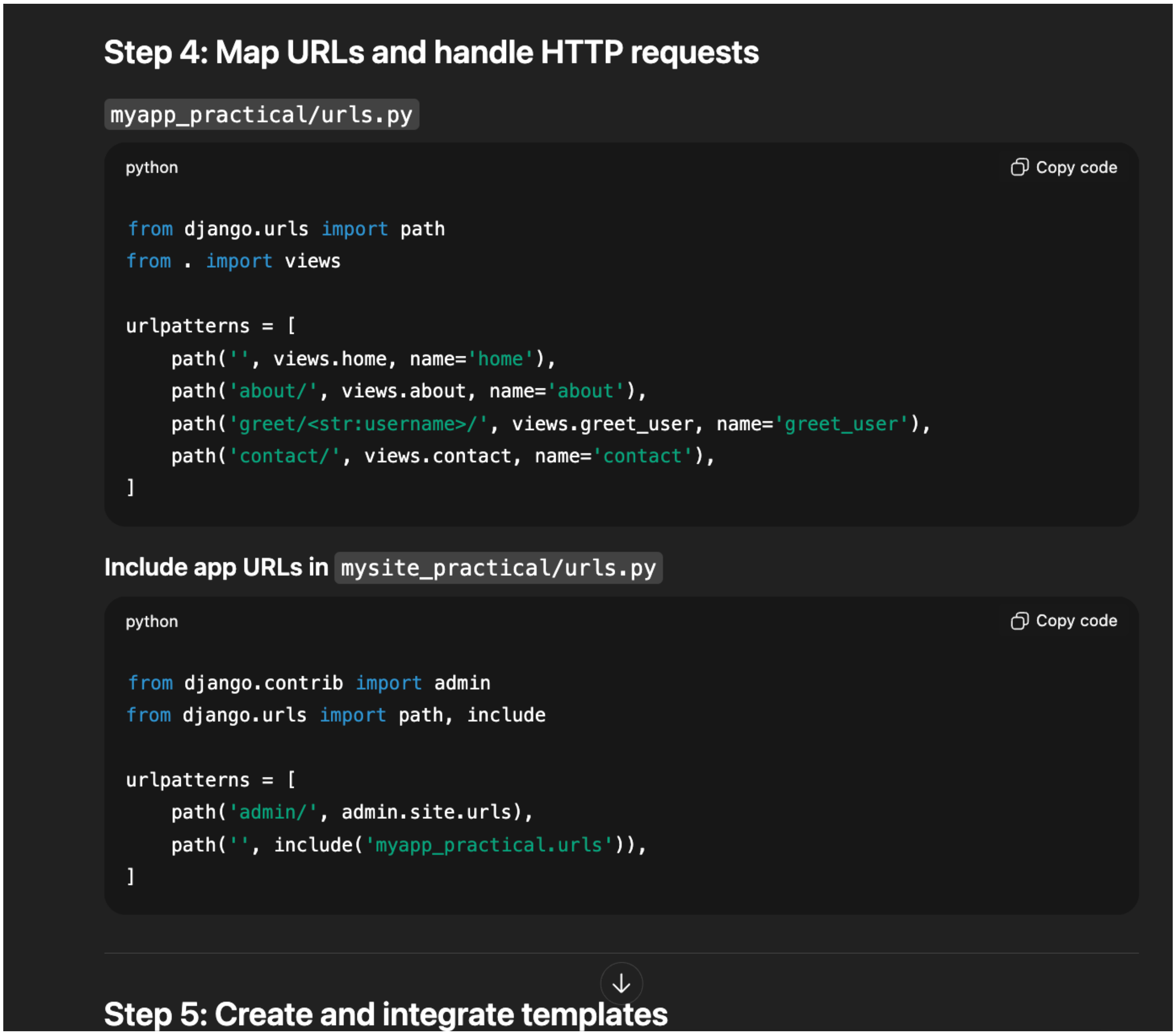
`myapp_practical/views.py`



Step 3: Develop views

`myapp_practical/views.py`

```python
from django.shortcuts import render, redirect
from django.http import HttpResponse
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from .forms import ContactForm
from .models import Product

def home(request):
    return render(request, 'myapp_practical/home.html', {'username': 'Sanuj'})

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

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

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_practical/contact.html', {'form': form})
```



Step 4: Map URLs and handle HTTP requests

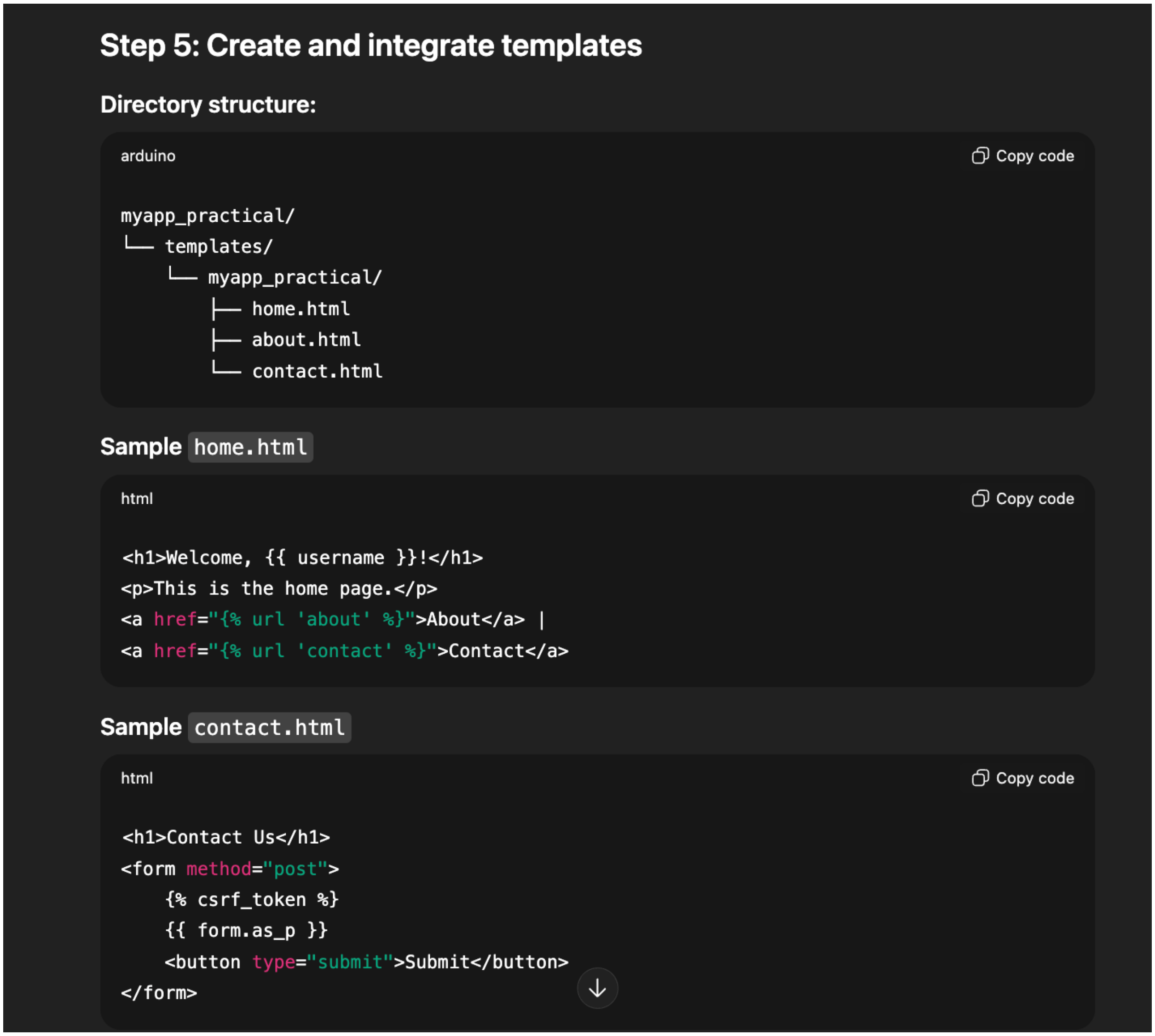
`myapp_practical/urls.py`

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

Include app URLs in `mysite_practical/urls.py`

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

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

Step 5: Create and integrate templates



Step 5: Create and integrate templates

Directory structure:

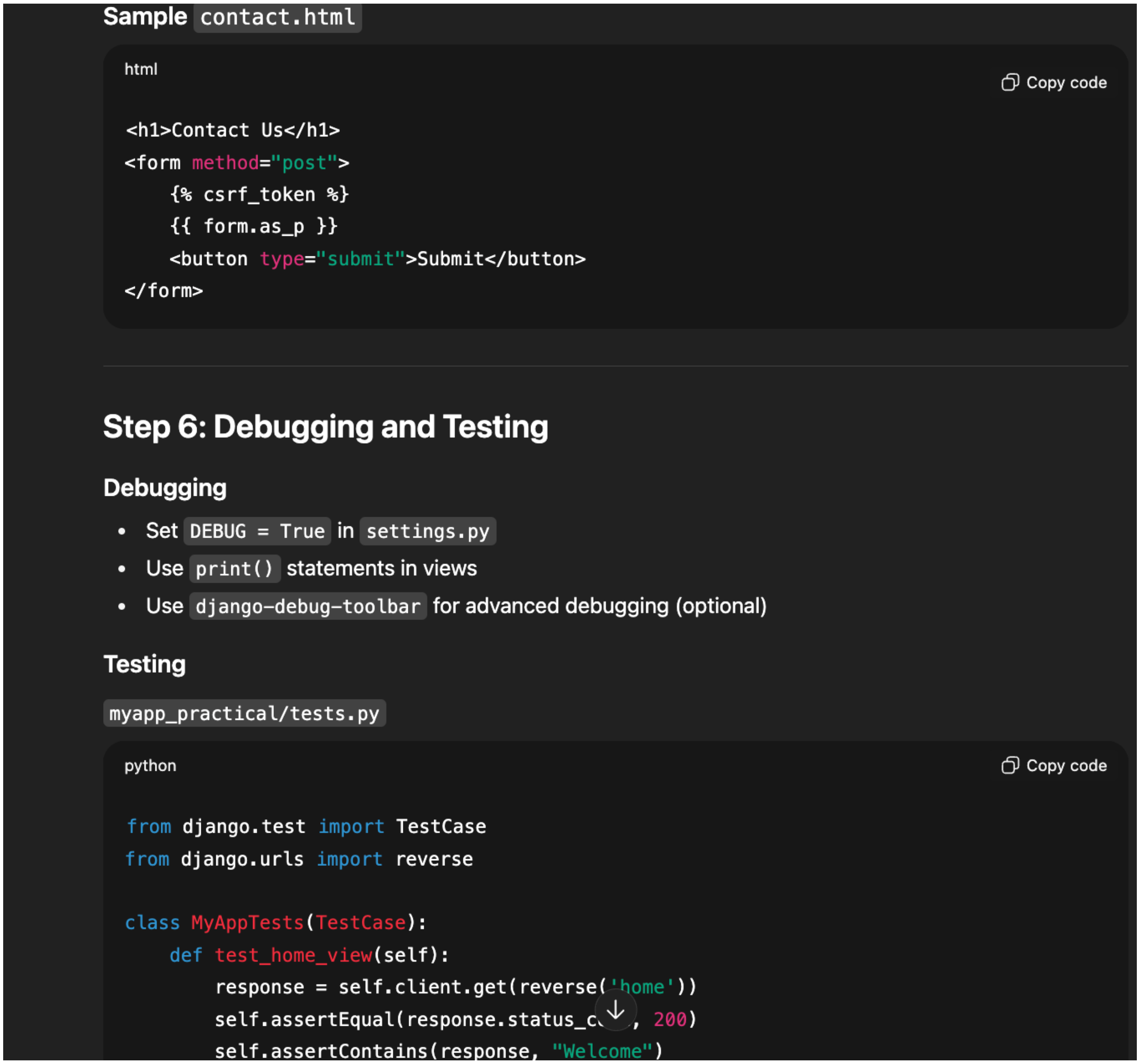
```
myapp_practical/
└── templates/
    └── myapp_practical/
        ├── home.html
        ├── about.html
        └── contact.html
```

Sample `home.html`

```html
<h1>Welcome, {{ username }}!</h1>
<p>This is the home page.</p>
<a href="{% url 'about' %}">About</a> |
<a href="{% url 'contact' %}">Contact</a>
```

Sample `contact.html`

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



Sample `contact.html`

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

Step 6: Debugging and Testing

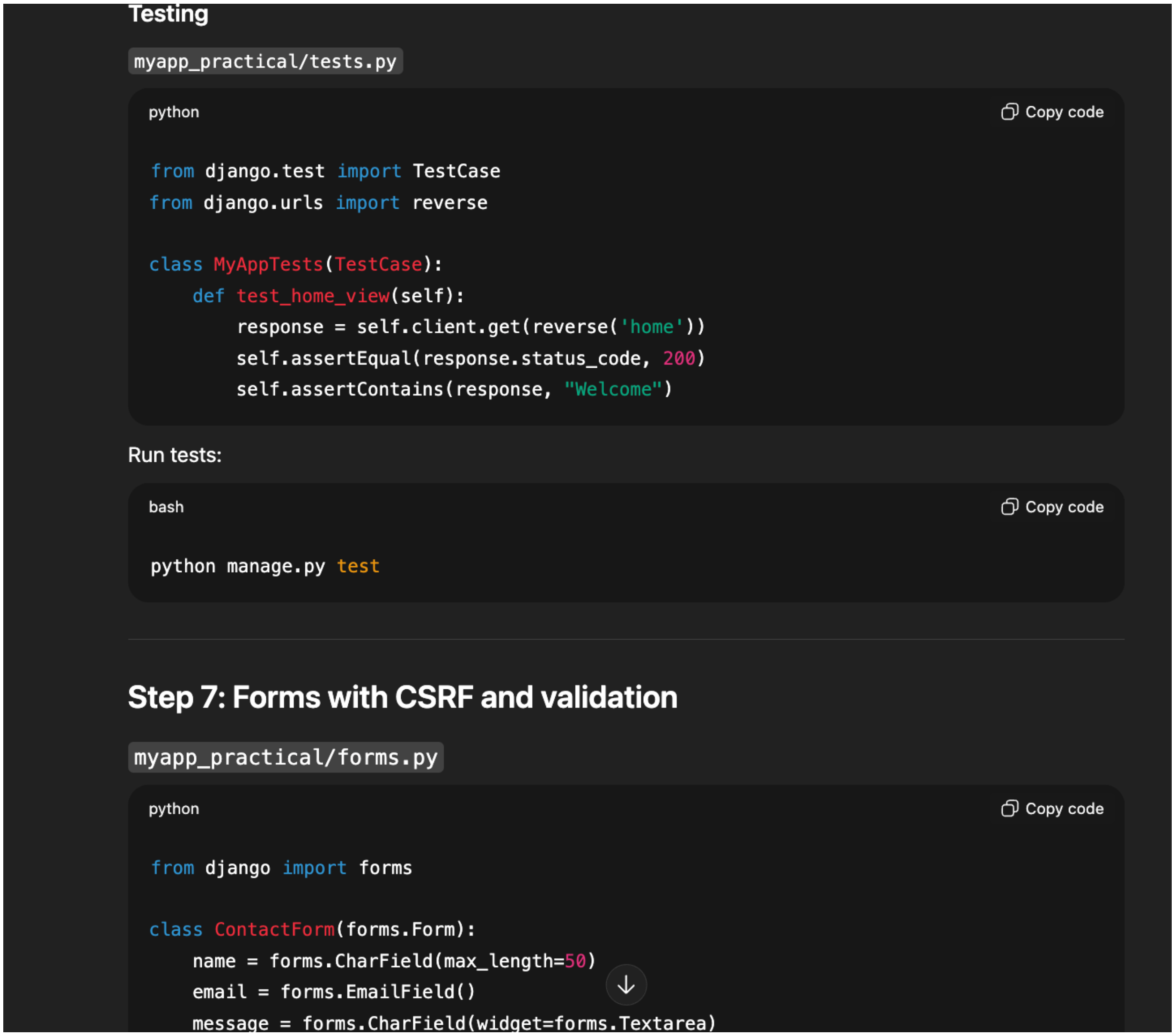
Debugging

- Set `DEBUG = True` in `settings.py`
- Use `print()` statements in views
- Use `django-debug-toolbar` for advanced debugging (optional)

Testing

`myapp_practical/tests.py`

```python
from django.test import TestCase
from django.urls import reverse

class MyAppTests(TestCase):
    def test_home_view(self):
        response = self.client.get(reverse('home'))
        self.assertEqual(response.status_c[illegible], 200)
        self.assertContains(response, "Welcome")
```



Testing

`myapp_practical/tests.py`

```python
from django.test import TestCase
from django.urls import reverse

class MyAppTests(TestCase):
    def test_home_view(self):
        response = self.client.get(reverse('home'))
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, "Welcome")
```

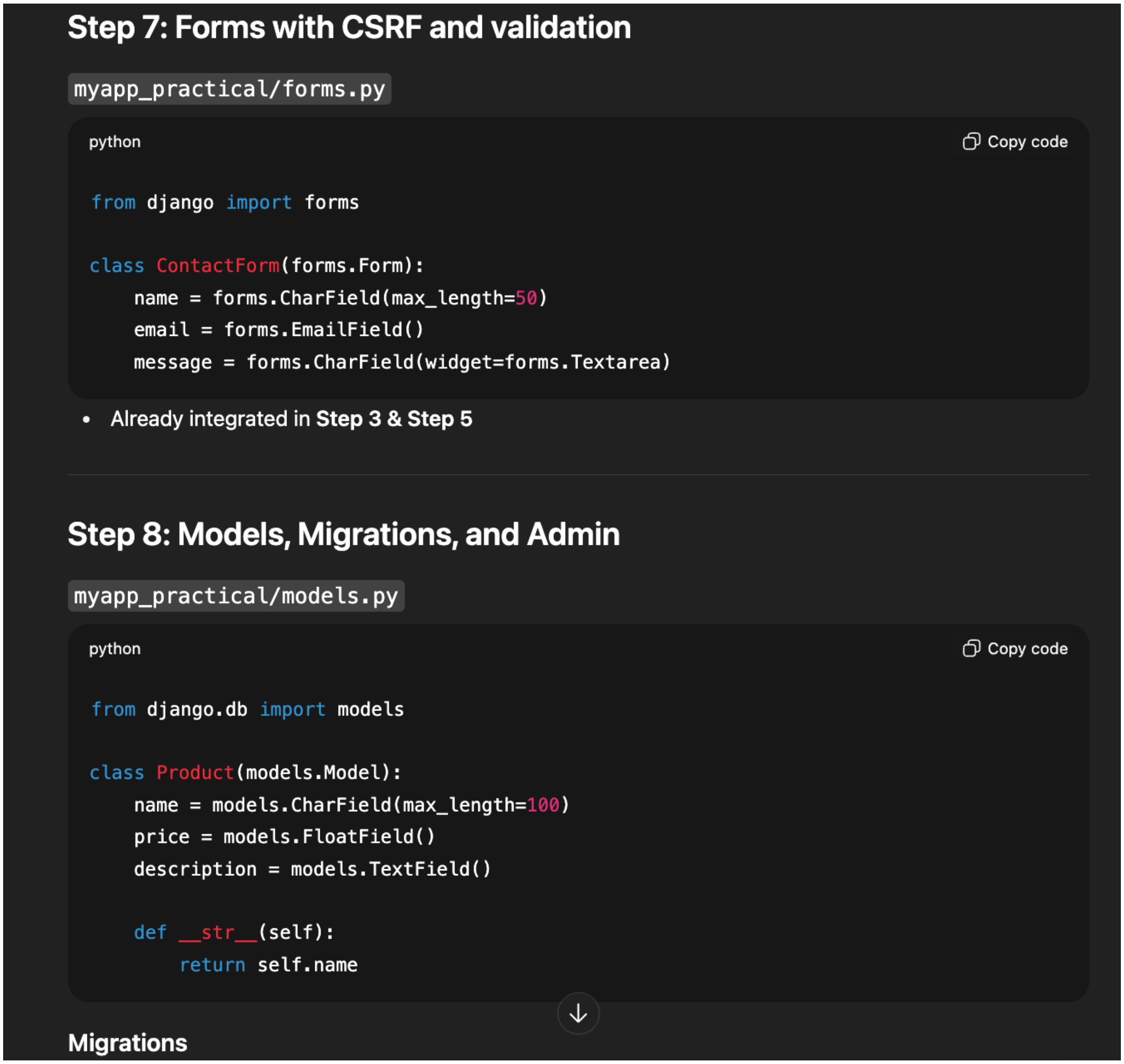
Run tests:

```bash
python manage.py test
```

Step 7: Forms with CSRF and validation

`myapp_practical/forms.py`

```python
from django import forms

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



Step 7: Forms with CSRF and validation

`myapp_practical/forms.py`

```python
from django import forms

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

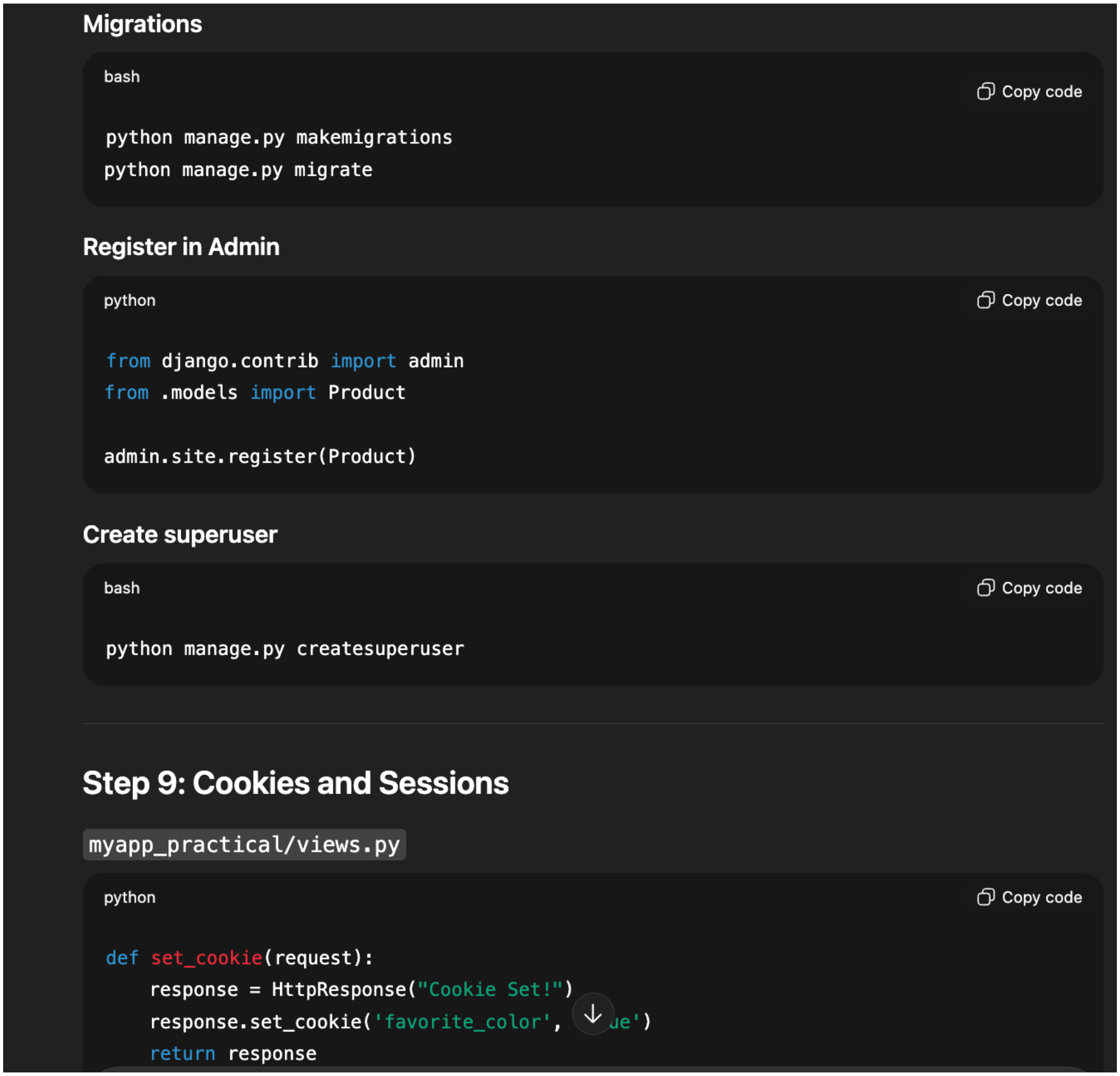
- Already integrated in **Step 3 & Step 5**

Step 8: Models, Migrations, and Admin

`myapp_practical/models.py`

```python
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
```

Migrations



Migrations

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

Register in Admin

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

admin.site.register(Product)
```

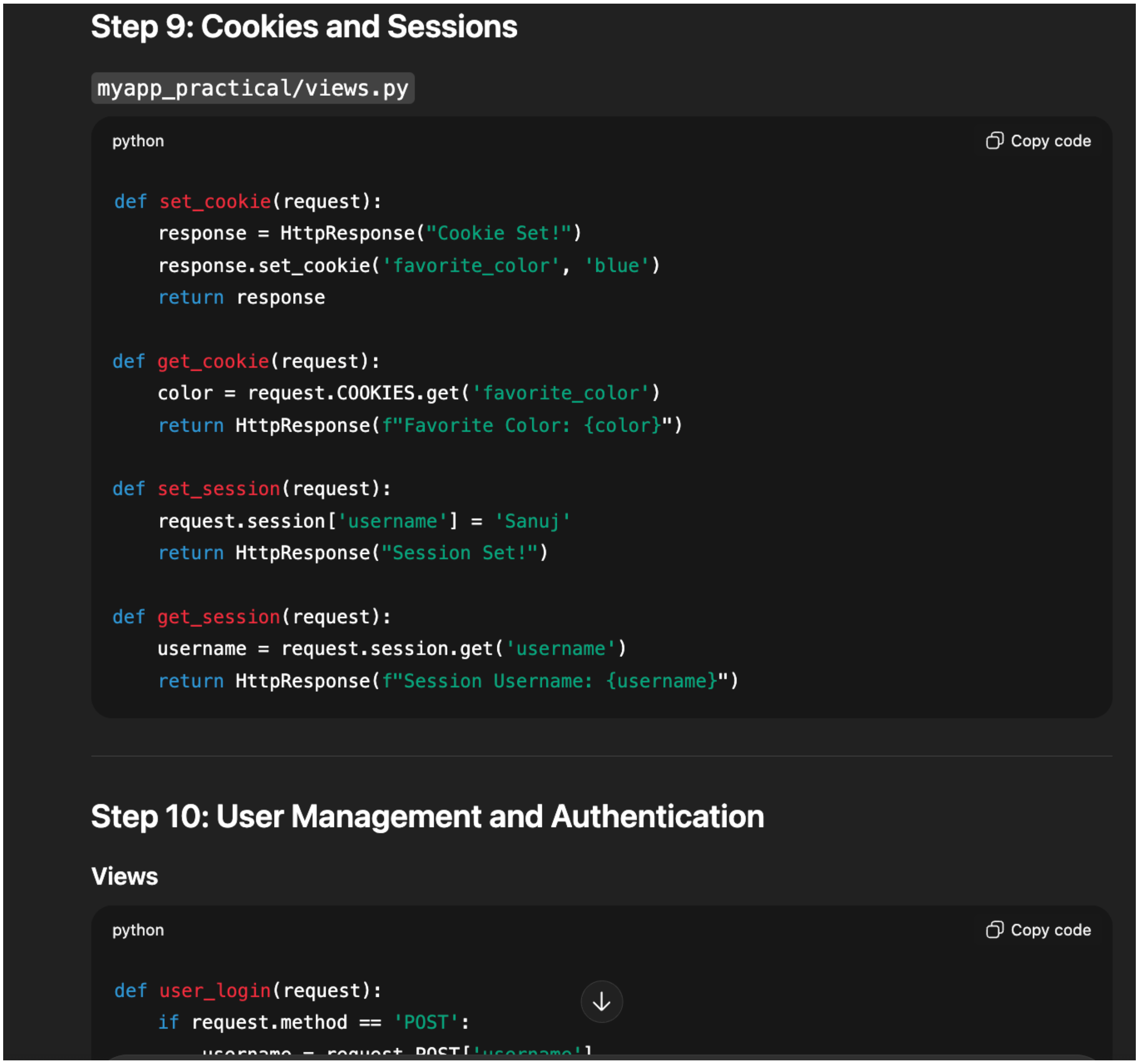
Create superuser

```bash
python manage.py createsuperuser
```

Step 9: Cookies and Sessions

`myapp_practical/views.py`

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



Step 9: Cookies and Sessions

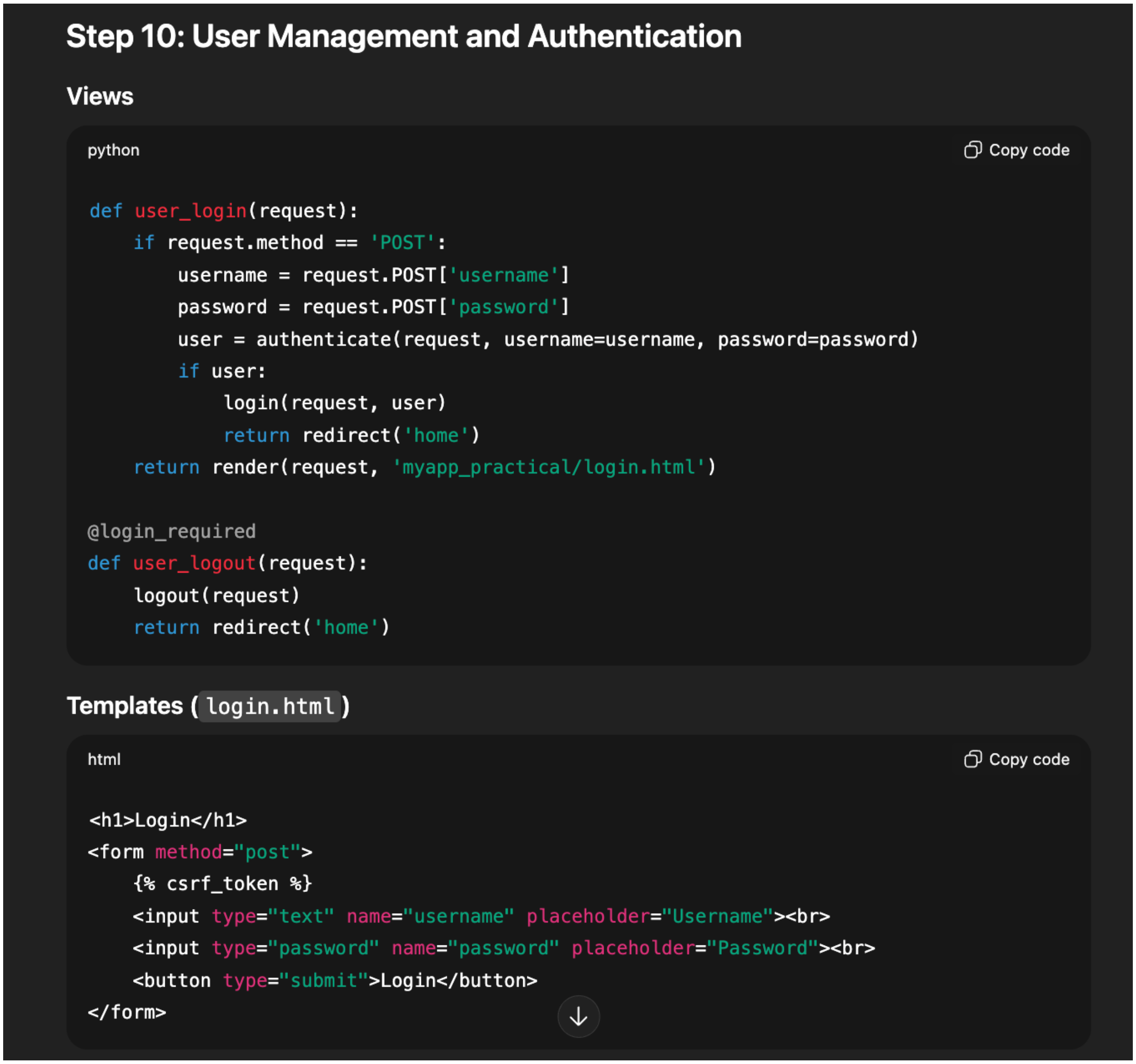
`myapp_practical/views.py`

```python
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}")

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}")
```

Step 10: User Management and Authentication

Views

```python
def user_login(request):
    if request.method == 'POST':
        username = request.POST['username']
```



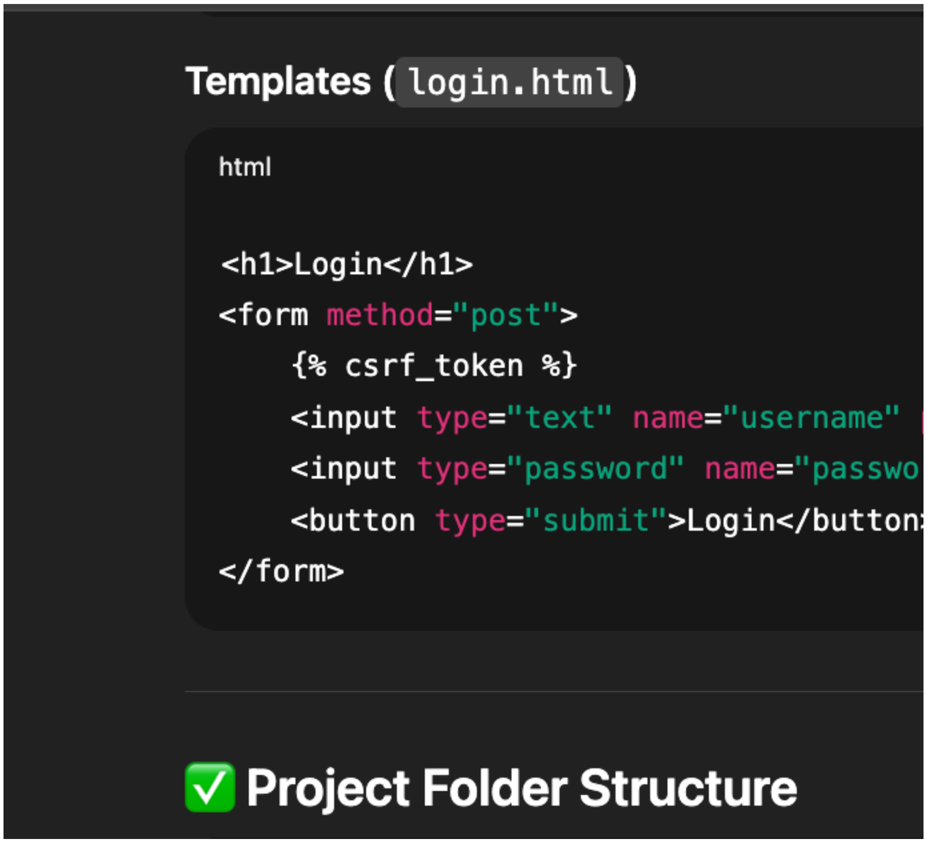
Step 10: User Management and Authentication

Views

```python
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:
            login(request, user)
            return redirect('home')
    return render(request, 'myapp_practical/login.html')

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

Templates (`login.html`)

```html
<h1>Login</h1>
<form method="post">
    {% csrf_token %}
    <input type="text" name="username" placeholder="Username"><br>
    <input type="password" name="password" placeholder="Password"><br>
    <button type="submit">Login</button>
</form>
```

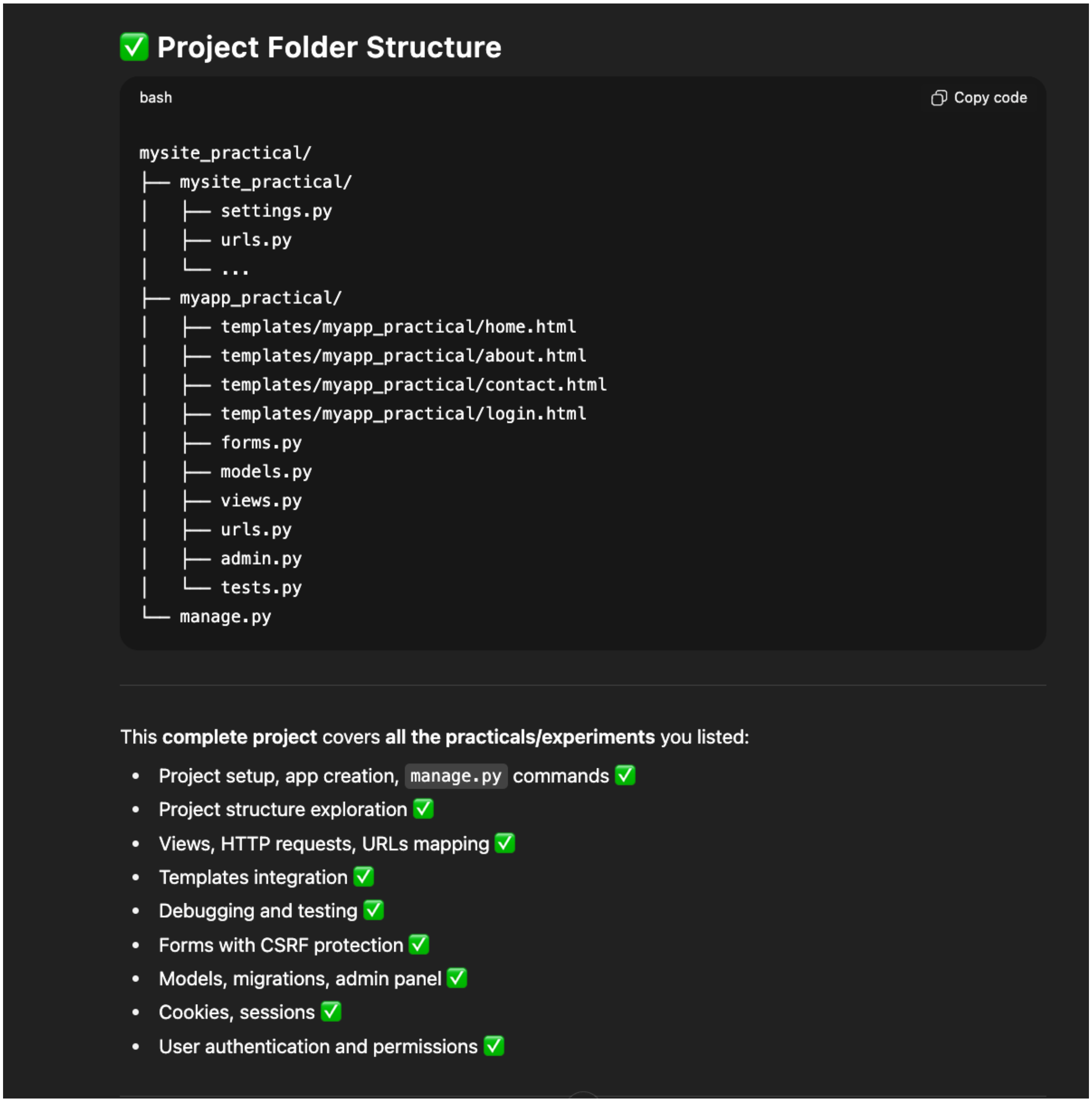


Templates (`login.html`)

html

```html
<h1>Login</h1>
<form method="post">
    {% csrf_token %}
    <input type="text" name="username"
    <input type="password" name="passwo
    <button type="submit">Login</button
</form>
```

✅ Project Folder Structure



✅ Project Folder Structure

```bash
mysite_practical/
├── mysite_practical/
│   ├── settings.py
│   ├── urls.py
│   └── ...
├── myapp_practical/
│   ├── templates/myapp_practical/home.html
│   ├── templates/myapp_practical/about.html
│   ├── templates/myapp_practical/contact.html
│   ├── templates/myapp_practical/login.html
│   ├── forms.py
│   ├── models.py
│   ├── views.py
│   ├── urls.py
│   ├── admin.py
│   └── tests.py
└── manage.py
```

This **complete project** covers **all the practicals/experiments** you listed:

- Project setup, app creation, `manage.py` commands ✅
- Project structure exploration ✅
- Views, HTTP requests, URLs mapping ✅
- Templates integration ✅
- Debugging and testing ✅
- Forms with CSRF protection ✅
- Models, migrations, admin panel ✅
- Cookies, sessions ✅
- User authentication and permissions ✅

https://
chatgpt.com/
share/
68c167a2-0ce
0-800b-
a776-86244e4
bef12