Here's a complete example of a Django app implementing your requirements using function-based views:

- Display list of courses.
- After clicking a course, display list of lectures.
- After clicking a lecture, show video and PDF notes.
- Admin can manage CRUD operations on lectures (video + notes).

App structure (e.g. `courses/`)

Here’s a quick file structure example:

```
myproject/
│
├── courses/
│   ├── models.py
│   ├── views.py
│   ├── urls.py
│   ├── forms.py
│   ├── templates/
│   │   └── courses/
│   │       ├── course_list.html
│   │       ├── lecture_list.html
│   │       ├── lecture_detail.html
│   │       ├── lecture_form.html
│   │       └── lecture_confirm_delete.html
│
├── media/
│   └── pdf_notes/    ← uploaded files go here
│
├── manage.py
└── myproject/
    └── settings.py
```

Step 1:- models.py in `courses` app:-:-

```
   from django.db import models


class Course(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField(blank=True)

    def __str__(self):
        return self.title

class Lecture(models.Model):
    course = models.ForeignKey(Course, related_name='lectures',
on_delete=models.CASCADE)
    title = models.CharField(max_length=200)
    video_url = models.URLField()
    pdf_notes = models.FileField(upload_to='pdf_notes/')

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

Step 2:- forms.py in `courses` app:-

```
from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import AuthenticationForm
from .models import Lecture

class LectureForm(forms.ModelForm):
    class Meta:
        model = Lecture
        fields = ['course', 'title', 'video_url', 'pdf_notes']

class AdminSignupForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput)
    confirm_password = forms.CharField(widget=forms.PasswordInput)

    class Meta:
        model = User
        fields = ['username', 'email', 'password']

    def clean(self):
        cleaned_data = super().clean()
        if cleaned_data.get('password') != cleaned_data.get('confirm_password'):
            self.add_error('confirm_password', 'Passwords do not match')
```

Step 3:- views.py in `courses` app:-:-

```
from django.shortcuts import render, get_object_or_404, redirect
from .models import Course, Lecture
from .forms import LectureForm, AdminSignupForm
from django.contrib.auth.forms import AuthenticationForm
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.models import User
from django.contrib.auth.decorators import login_required, user_passes_test
from django.contrib import messages

def course_list(request):
    courses = Course.objects.all()
    return render(request, 'courses/course_list.html', {'courses': courses})

def lecture_list(request, course_id):
    course = get_object_or_404(Course, pk=course_id)
    lectures = course.lectures.all()
    return render(request, 'courses/lecture_list.html', {'course': course,
'lectures': lectures})

def lecture_detail(request, lecture_id):
    lecture = get_object_or_404(Lecture, pk=lecture_id)
    return render(request, 'courses/lecture_detail.html', {'lecture': lecture})

def admin_required(view_func):
    return login_required(user_passes_test(lambda u: u.is_staff)(view_func))

@admin_required
def lecture_create(request):
    if request.method == 'POST':
        form = LectureForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            return redirect('courses:course_list')
    else:
        form = LectureForm()
    return render(request, 'courses/lecture_form.html', {'form': form, 'action':
'Create'})

@admin_required
def lecture_update(request, lecture_id):
    lecture = get_object_or_404(Lecture, pk=lecture_id)
    if request.method == 'POST':
```

```
        form = LectureForm(request.POST, request.FILES, instance=lecture)
        if form.is_valid():
            form.save()
            return redirect('courses:lecture_detail', lecture_id=lecture.id)
    else:
        form = LectureForm(instance=lecture)
    return render(request, 'courses/lecture_form.html', {'form': form, 'action':
'Update'})

@admin_required
def lecture_delete(request, lecture_id):
    lecture = get_object_or_404(Lecture, pk=lecture_id)
    if request.method == 'POST':
        lecture.delete()
        return redirect('courses:course_list')
    return render(request, 'courses/lecture_confirm_delete.html', {'lecture':
lecture})

def admin_signup(request):
    if request.method == 'POST':
        form = AdminSignupForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.set_password(form.cleaned_data['password'])
            user.is_staff = True
            user.save()
            messages.success(request, 'Admin account created. Please log in.')
            return redirect('courses:admin_login')
    else:
        form = AdminSignupForm()
    return render(request, 'courses/admin_signup.html', {'form': form})

def admin_login(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            user = authenticate(
                username=form.cleaned_data.get('username'),
                password=form.cleaned_data.get('password')
            )
            if user and user.is_staff:
                login(request, user)
                return redirect('courses:course_list')
            messages.error(request, 'Invalid credentials or not an admin.')
    else:
```

```
        form = AuthenticationForm()
    return render(request, 'courses/admin_login.html', {'form': form})

def admin_logout(request):
    logout(request)
    return redirect('courses:course_list')
```

✅ Step 4 `urls.py` in `courses` app:-

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

app_name = 'courses'

urlpatterns = [
    path('', views.course_list, name='course_list'),
    path('course/<int:course_id>/', views.lecture_list, name='lecture_list'),
    path('lecture/<int:lecture_id>/', views.lecture_detail,
name='lecture_detail'),

    # Lecture management
    path('admin/lecture/create/', views.lecture_create, name='lecture_create'),
    path('admin/lecture/<int:lecture_id>/update/', views.lecture_update,
name='lecture_update'),
    path('admin/lecture/<int:lecture_id>/delete/', views.lecture_delete,
name='lecture_delete'),

    # Admin auth
    path('admin/signup/', views.admin_signup, name='admin_signup'),
    path('admin/login/', views.admin_login, name='admin_login'),
    path('admin/logout/', views.admin_logout, name='admin_logout'),
]
```

Step 5 :- ✅ Templates (examples)

Put these under `templates/courses/`.

course_list.html

```
<h1>Courses</h1>
<ul>
  {% for course in courses %}
    <li><a href="{% url 'courses:lecture_list' course.id %}">{{ course.title
}}</a></li>
  {% endfor %}
</ul>
<a href="{% url 'courses:admin_login' %}">Admin Login</a>
```

lecture_list.html

```
<h2>Lectures for {{ course.title }}</h2>
<ul>
  {% for lecture in lectures %}
    <li><a href="{% url 'courses:lecture_detail' lecture.id %}">{{
lecture.title }}</a></li>
  {% endfor %}
</ul>
```

lecture_detail.html

```
<h2>{{ lecture.title }}</h2>
<iframe width="560" height="315" src="{{ lecture.video_url }}"></iframe>
<br><br>
<a href="{{ lecture.pdf_notes.url }}" download>Download Notes</a>
```

lecture_form.html

```
<h2>{{ action }} Lecture</h2>
<form method="post" enctype="multipart/form-data">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">{{ action }}</button>
</form>
```

lecture_confirm_delete.html

```
html
CopyEdit
<h2>Delete Lecture</h2>
<form method="post">
    {% csrf_token %}
    <p>Are you sure?</p>
    <button type="submit">Yes, Delete</button>
</form>
```

admin_signup.html

```
<h2>Admin Signup</h2>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Sign Up</button>
</form>
```

admin_login.html

```
<h2>Admin Login</h2>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Login</button>
</form>
```

Step 6 :- ⚙️ `settings.py` additions

```
import os

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

]
```

Step 7:- ⚙️ Project `urls.py`:-

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

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

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```