Django app that generates CSV and PDF files by retrieving data from models

**1. Create a Django Project and App**

First, create a Django project and an app within the project:

django-admin startproject report\_generator\_project

cd report\_generator\_project

python manage.py startapp report\_generator\_app

**2. Define Model for Data**

In `report\_generator\_app/models.py`, define a model to represent the data you want to generate reports for:

from django.db import models

class ReportData(models.Model):

name = models.CharField(max\_length=100)

age = models.IntegerField()

email = models.EmailField()

def \_\_str\_\_(self):

return self.name

**3. Create Views for Generating Reports**

In `report\_generator\_app/views.py`, create views for generating CSV and PDF reports:

from django.http import HttpResponse

from django.template.loader import get\_template

from xhtml2pdf import pisa

import csv

from .models import ReportData

def generate\_csv(request):

response = HttpResponse(content\_type='text/csv')

response['Content-Disposition'] = 'attachment; filename="report.csv"'

writer = csv.writer(response)

writer.writerow(['Name', 'Age', 'Email'])

queryset = ReportData.objects.all()

for item in queryset:

writer.writerow([item.name, item.age, item.email])

return response

def generate\_pdf(request):

template\_path = 'report\_generator\_app/pdf\_template.html'

queryset = ReportData.objects.all()

context = {'data': queryset}

response = HttpResponse(content\_type='application/pdf')

response['Content-Disposition'] = 'attachment; filename="report.pdf"'

template = get\_template(template\_path)

html = template.render(context)

pisa\_status = pisa.CreatePDF(html, dest=response)

if pisa\_status.err:

return HttpResponse('PDF generation failed')

return response

**4. Create HTML Template for PDF Generation**

Create an HTML template to render data for PDF generation. Save it as `pdf\_template.html` in the `report\_generator\_app/templates/report\_generator\_app` directory:

<!DOCTYPE html>

<html>

<head>

<title>Report</title>

</head>

<body>

<h2>Report</h2>

<table border="1">

<tr>

<th>Name</th>

<th>Age</th>

<th>Email</th>

</tr>

{% for item in data %}

<tr>

<td>{{ item.name }}</td>

<td>{{ item.age }}</td>

<td>{{ item.email }}</td>

</tr>

{% endfor %}

</table>

</body>

</html>

**5. Configure URLs**

Configure URLs in `report\_generator\_app/urls.py`:

from django.urls import path

from . import views

urlpatterns = [

path('generate-csv/', views.generate\_csv, name='generate\_csv'),

path('generate-pdf/', views.generate\_pdf, name='generate\_pdf'),

]

**6. Include App URLs in Project URLs**

Include app URLs in the project's `urls.py`:

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('report\_generator\_app.urls')),

]

**7. Install Required Packages**

You'll need to install the `xhtml2pdf` package for PDF generation:

pip install xhtml2pdf

**Explanation:**

**- \*\*Model\*\*:**

`ReportData` model represents the data for which reports will be generated.

**- \*\*Views\*\*:**

`generate\_csv` retrieves data from the database and generates a CSV file. `generate\_pdf` retrieves data and renders it in an HTML template, then converts it to a PDF file using `xhtml2pdf`.

- **\*\*Templates\*\*:**

`pdf\_template.html` contains the HTML structure for the PDF report, with data passed from the view.

- **\*\*URLs\*\*:**

URLs are configured to map to the `generate\_csv` and `generate\_pdf` views.

**- \*\*Settings\*\*:**

Make sure to configure `STATIC\_ROOT` and `STATIC\_URL` settings in `settings.py` to serve static files, including the PDF template.

That's it! With this setup, you can access `/generate-csv/` and `/generate-pdf/` URLs to generate CSV and PDF reports, respectively, based on the data stored in the `ReportData` model.