FORMS PRE-CLASS SETUP

```
# CREATING VIRTUAL ENVIRONMENT
# windows
py -m venv env
# windows other option
python -m venv env
# linux / Mac OS
vitualenv env

# ACTIVATING ENVIRONMENT
# windows
.\env\Scripts\activate
# linux / Mac OS
source env/bin/activate

# PACKAGE INSTALLATION
# if pip does not work try pip3 in linux/Mac OS
pip install django
# alternatively python -m pip install django
pip install python-decouple
django-admin --version
django-admin startproject forms .
```

add a gitignore file at same level as env folder

go to terminal

```
py manage.py migrate
py manage.py runserver
```

click the link with CTRL key pressed in the terminal and see django rocket.

go to terminal, stop project, add app

```
py manage.py startapp student
```

go to settings.py and **add 'student' app** to **installed apps** and add below lines to end of settings.py

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

go to students/models.py

```
from django.db import models

class Student(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    number = models.IntegerField(blank=True, null=True)
    profile_pic = models.ImageField(upload_to='profile_pics', blank=True)

    def __str__(self):
        return f"{self.last_name} {self.first_name}"
```

go to terminal

```
pip install pillow
pip freeze > requirements.txt
py manage.py makemigrations
py manage.py migrate
```

create template folder as student/templates/student

base.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
</head>

<body>
    {% block container %}
    {% endblock container %}
</body>

</html>
```

index.html

```
{% extends "student/base.html" %}
{% block container %}
<h1>Home Page</h1>
```

```
<h3>Student App</h3>

{% endblock container %}
```

student.html

```
{% extends "student/base.html" %}
{% block container %}
<form action="">
  <label for="">student name</label>
  <input type="text" />
  <input type="submit" value="OK" />
</form>
{% endblock container %}
```

go to student/views.py

```
from django.shortcuts import render

def index(request):
    return render(request, 'student/index.html')

def student_page(request):
    return render(request,'student/student.html')
```

go to forms/urls.py

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

from student.views import index

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

go to student/urls.py

```
from django.urls import path

from .views import student_page
```

```
urlpatterns = [
    path('', student_page, name='student'),
]
```

go to terminal

```
py manage.py createsuperuser
py manage.py runserver
```

IN-CLASS STARTS HERE

run server and explain urls and form.html

go to students/forms.py

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

class StudentForm(forms.Form):
    first_name = forms.CharField(max_length=50)
    last_name = forms.CharField(max_length=50)
    number = forms.IntegerField(required=False)
```

go to student/views.py and amend student_page

```
from .forms import StudentForm

def student_page(request):
        form = StudentForm()
        context = {
            'form': form
        }
    return render(request,'student/student.html', context)
```

explain sending form

go to student/templates/student/student.html and amend below lines

```
% extends "student/base.html" %}
{% block container %}

<h2>Student Form</h2>

{% comment %}
```

```
<form action="">
  <label for="">student name</label>
  <input type="text" />
  <input type="submit" value="OK" />
</form>
{% endcomment %}

<form action="" method="post" enctype="multipart/form-data">
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="OK" />
</form>

{% endblock container %}
```

explain get, post, enctype and CSRF

go to student/forms.py and amend StudentForm and use forms.ModelForm class

```
from .models import Student
class StudentForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = ["first_name", "last_name", "number", "profile_pic"]
        labels = {"first_name": "Name"}
```

Form Save Process

go to student/views.py and amend student_page

long way of data save process

```
def student_page(request):
#     # create an empty form
#     form = StudentForm()
#     if request.method == 'POST':
#         # create form and fill in this form with user data inside request.POST
#         form = StudentForm(request.POST)
#         # check form validation
#         if form.is_valid():
#             # get user data inside form
#             student_data = {
#                 "first_name": form.cleaned_data.get('first_name'),
#                 "last_name": form.cleaned_data.get('last_name'),
#                 "number": form.cleaned_data.get('number')
#                 # "profile_pic": form.cleaned_data.get('profile_pic'),
#             }
#             # database save process
#             # by specifiying fields
#             # student =
Student(first_name=student_name,last_name=student_surname, number=student_number)
#             # or by kwargs, spreaind the student_data dict
#             student = Student(**student_data)
#             # uploaded files are not inside request.POST, instead they are in
request.FILES
#             if 'profile_pic' in request.FILES:
#                 student.profile_pic = request.FILES['profile_pic']
#             student.save()
#             return redirect('index')

#     context = {
#         'form': form
#     }
#     return render(request, 'student/student.html', context)
```

short way of save process

```
def student_page(request):
    # if request.method is POST then fill in the form with user data otherwise
create a blank form
    form = StudentForm(request.POST or None)
    # check form validation
    if form.is_valid():
        # save to database
        student = form.save()
        # uploaded files are not inside request.POST, instead they are in
request.FILES
        if 'profile_pic' in request.FILES:
            # if a profile_pic uploaded then add it to student and then save
student to db
            student.profile_pic = request.FILES.get('profile_pic')
            student.save()
        # after save POST status should be ended with redirect
        return redirect('index')
    context = {
        'form': form
    }
    return render(request,'student/student.html', context)
```

explain POST, and how to save student

go to terminal

```
py manage.py createsuperuser
```

navigate to admin panel and show that student model does not exist go to student/admin.py

```
from django.contrib import admin

from .models import Student
# Register your models here.
admin.site.register(Student)
```

navigate to admin panel and show student model there and display recorded students

Alternative way of form saving process

go to student/views.py and amend student_page

```
def student_form(request):
    form = StudentForm()
    if request.method == 'POST':
        form = StudentForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            return redirect('/student/')

    context = {
        'form': form
    }
    return render(request, 'student/student.html', context)
```

explain form.save and request FILES

BOOTSTRAP

go to student/templates/student/base.html and add bootstrap

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <!-- <link rel="stylesheet" href="../static/css/style.css" /> -->
    <link rel="stylesheet" href="{% static 'css/style.css' %}" />
    <!-- CSS only -->
    <link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
      crossorigin="anonymous"
    />
  </head>

  <body>
    <div style="margin-top: 100px; margin-bottom: 100px" class="container">
      {% block container %}
      {% endblock container %}
    </div>
    <!-- JavaScript Bundle with Popper -->
    <script

src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
      integrity="sha384-
ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
      crossorigin="anonymous"
    ></script>
  </body>
</html>
```

Styling with CSS

create static folder as student/static/student

style.css

```
h3 {
  background-color: orchid;
}
```

CRISPY FORMS

go to terminal

```
pip install django-crispy-forms
pip freeze > requirements.txt
```

go to settings.py

```
INSTALLED_APPS = (
    ...
    'crispy_forms',
)

CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

go to student/templates/student/student.html and crispy tags

```
{% extends "student/base.html" %}
{% block container %}

<div style="width:300px;">
    {% load crispy_forms_tags %}
    <form action="" method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form | crispy}}
        <br>
        <input type="submit" value="OK" />
    </form>
</div>

{% endblock container %}
```

Messages

go to student/views.py and import messages end send success message

```
from django.contrib import messages
def student_form(request):
    form = StudentForm()
    if request.method == 'POST':
        form = StudentForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            messages.success(request, "Student added successful")
            return redirect('/student/')
    context = {
        'form': form
    }
    return render(request, 'student/student.html', context)
```

go to student/templates/student/base.html and add messages codes

```
<!DOCTYPE html>
{% load static %}
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <!-- <link rel="stylesheet" href="../static/css/style.css" /> -->
    <link rel="stylesheet" href="{% static 'css/style.css' %}" />
    <!-- CSS only -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"
        integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous" />
</head>

<body>
    <div style="margin-top: 100px; margin-bottom: 100px" class="container">

        {% if messages %}
        {% for message in messages %}
        {% if message.tags == "error" %}
        <div class="alert alert-danger">{{ message }}</div>
        {% else %}
        <div class="alert alert-{{ message.tags }}">{{ message }}</div>
        {% endif %}
        {% endfor %}
        {% endif %}

        {% block container %}
        {% endblock container %}
    </div>
    <!-- JavaScript Bundle with Popper -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
crossorigin="anonymous">
    </script>
</body>

</html>
```

Bonus - Custom Validation for Form

go to student/forms.py

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

class StudentForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = '__all__'
        # fields = ["first_name", "last_name", "number", "profile_pic"]
        labels = {"first_name": "Name"}
    # common validator
    def clean(self):
        # first_name = self.cleaned_data.get("first_name")
        # last_name = self.cleaned_data.get("last_name")
        # number = self.cleaned_data.get("number")

        # form_data = self.cleaned_data
        form_data = super().clean()
        number = form_data.get('number')

        if not number == None and not 1000 < int(number) < 10000:
            raise forms.ValidationError("Numbers should be between 1000 and
10000")

        return form_data

    # validator to a specific field
    # def clean_number(self):

    #     number = self.cleaned_data.get("number")

    #     if not number == None and not 1000 < number < 10000:
    #         raise forms.ValidationError("Numbers should be between 1000 and
10000")

    #     return number
```