Environment setup for Django → create a project and create an app inside the project

1.
```
C:\Users\dksr1\OneDrive\Desktop\python>django-admin startproject sastra
```

2. Go to **sastra** folder and Creating app with name **swi.**

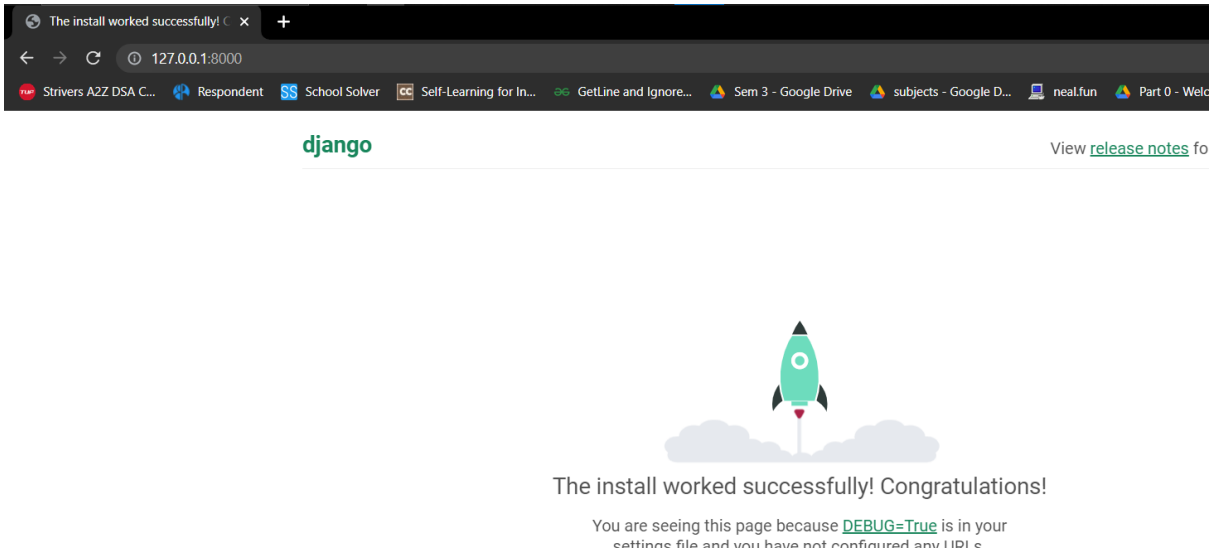
```
C:\Users\dksr1\OneDrive\Desktop\python>cd sastra

C:\Users\dksr1\OneDrive\Desktop\python\sastra>dir
 Volume in drive C is OS
 Volume Serial Number is C20D-361E

 Directory of C:\Users\dksr1\OneDrive\Desktop\python\sastra

22-12-2022  13:46    <DIR>          .
22-12-2022  13:46    <DIR>          ..
22-12-2022  13:46               684 manage.py
22-12-2022  13:46    <DIR>          sastra
               1 File(s)            684 bytes
               3 Dir(s)  35,384,086,528 bytes free

C:\Users\dksr1\OneDrive\Desktop\python\sastra>py manage.py startapp swi

C:\Users\dksr1\OneDrive\Desktop\python\sastra>_
```

3. Create a folder with name **templates/** in **sastra** folder.
4. In **sastra folder,** we should have the following files

Name	Date modified	Type	Size
sastra	22-12-2022 13:47	File folder	
swi	22-12-2022 13:47	File folder	
templates	22-12-2022 13:51	File folder	
manage.py	22-12-2022 13:46	Python Source File	1 KB

5. Run the project using command
   py manage.py runserver

   then reach the browser and go with the url http://127.0.0.1:8000/ which can be copied from cmd after running the server. If everything goes well, output should be like shown below >>>



---------- **setup part is over** ----------

Now we have to open 6 necessary files. Coding and modification of these files is enough for the Django project for exam.

→From **swi** folder, open : views.py, models.py, admin.py
→From **sastra** folder, open : settings.py, urls.py

→Create a file with name **forms.py** in **swi/** folder and open it

---------- **file opening part is also over** ----------

3 - Minor and easy changes in settings.py now 😊

1 . go to settings.py

In **INSTALLED_APPS,** add swi as below

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'swi',
]
```

2 . in settings.py

import os

3. in templates variable in settings.py, in DIRS[] variable, do the following change

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

No need of settings files anymore……..close it 😊

---------- **close settings.py file** ----------

Now we will play with models.py and forms.py

models.py will contain data related to administration block and forms.py will contain data which is necessary for taking input users from website.

Confused?????

Let us go through some examples.

Take bus depo as an example. Govt should store bus, its destination cities names, and total seats available in a bus in database. But there is no need for travellers to see or know total available seats in bus and all destinations bus is going. Conductor will go to each person and ask what is the traveller destination and number of seats he want. So, government feeds some data of bus in database and that data is made from *models.py* but data that traveller fills to get the ticket is made from *forms.py*.

Lets get back to the current question now…….

In models.py

Write the following code

```
swi > models.py > student > __str__
from django.db import models

# Create your models here.
class student(models.Model):
    student_name = models.CharField(max_length=15)
    student_reg_no = models.CharField(max_length=10)
    student_password = models.CharField(max_length=8)
    student_attendance = models.CharField(max_length=3)
    student_cgpa = models.CharField(max_length=7)

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

In forms.py

```
from django import forms

# Create your forms here.
class student_details(forms.Form):
    student_details_name = forms.CharField(max_length=15)
    student_details__reg_no = forms.CharField(max_length=10)
    student_details_password = forms.CharField(max_length=8)
    student_details_attendance = forms.CharField(max_length=3)
    student_details_cgpa = forms.CharField(max_length=7)
```

The magic is this forms.py will create its own webpage instead of any html file to take inputs from user……..

Work with models.py and forms.py is over but keep them open for future reference of variable names 😊

---------- don't close forms.py and models.py 😉😊----------

we will register our model with the database now……..

➔ Go to admin.py

Then import student class from models using

from .models import student

```
urls.py    admin.py ×    models.py
swi > admin.py
1  from django.contrib import admin
2  from .models import student
3  # Register your models here.
4  admin.site.register(student)
```

---------- close admin.py now ----------

Lets register superuser which is like owner of db…

CMD TIME

1. First do python manage.py makemigrations
2. Then do python manage.py migrate

```
C:\Users\dksr1\OneDrive\Desktop\python\sastra>python manage.py makemigrations
Migrations for 'swi':
  swi\migrations\0001_initial.py
    - Create model student

C:\Users\dksr1\OneDrive\Desktop\python\sastra>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, swi
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
  Applying swi.0001_initial... OK

C:\Users\dksr1\OneDrive\Desktop\python\sastra>
```

3. Creating a super user

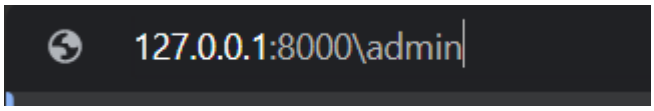
```
C:\Users\dksr1\OneDrive\Desktop\python\sastra>python manage.py createsuperuser
Username (leave blank to use 'dksr1'): dk
Email address: 09e80z@gmail.com
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

C:\Users\dksr1\OneDrive\Desktop\python\sastra>
```

Remember the password and username also……..

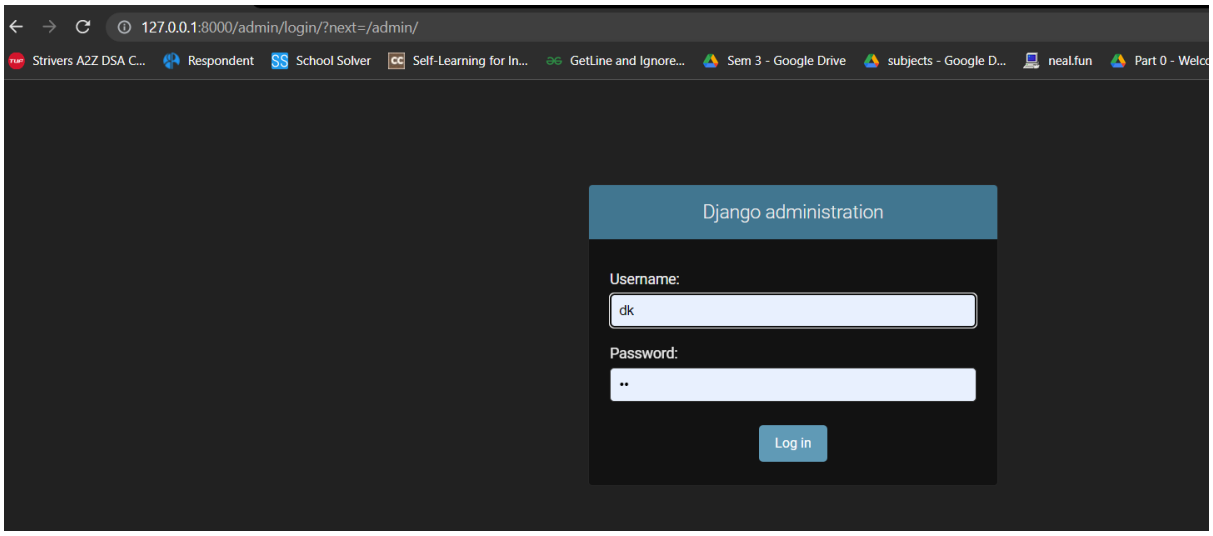
To run this now

python manage.py runserver

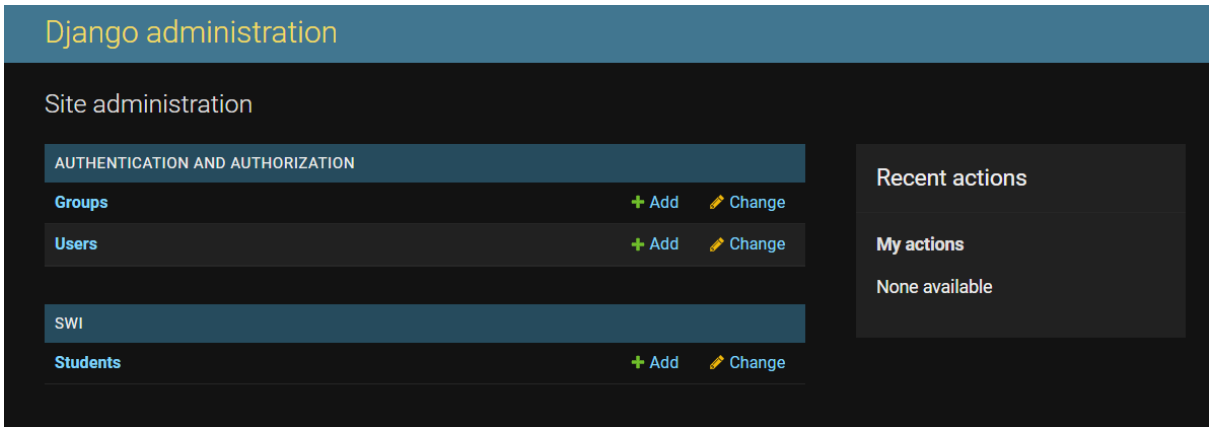
Then *go to url and search like shown below->*

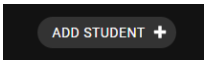


➔ *The page should be loaded like this if you search for the above line.*



After clicking login , we should be able to see like :

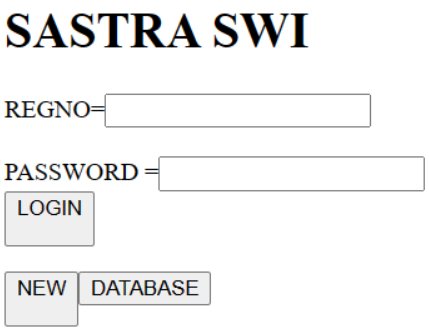


Click on students and use this 

button and add few students details which are the fields what we created in models.py

----------→ time to become a web developer 🤣

We will create a homepage like this :



Entering correct details will print students details from database…

We can add new students and can see the whole database also…

For this we create 3 html files in templates folder, follow the instructions carefully from now.

1. Go to templates folder and create **home.html**

```
<html>
    <head>
        <title>SWI LOGIN</title>
        <h1>SASTRA SWI</h1>
    </head>
<body>

<form action="login">
    REGNO=<input type:"text" name="a"> <br><br>
    PASSWORD =<input type:"text" name="b"> <br>
    <button type:"submit">LOGIN
</form>

<form action="new">
    <button type:"submit">NEW
</form>

<form action="data">
    <button type:"submit">DATABASE
</form>

</body>
</html>
```

When we click submit button, registration number will be saved as variable a and password as variable b .

We can see three form actions which are 3 buttons here. One is login <using specific student details to fetch only one student details from database> .

Another one is **new** which is used to add new student details to database.

Last one is databse button whose action is **data** here which fetches all data from data base.

When we click **new** button , a form will be loaded on to screen to take input.

This is done from enter.html

```
<html>
    <head>
    <title>HOME</title>
        <h1>ADD DETAILS<h1>
    </head>
    <body>
        <form action="" method="post" nonvalidate>
            {%csrf_token%}
            {{form}}
            <button type="submit">submit</button>
        </form>
    </body>
</html>
```

In this file, we use action=”” and method as post

{{form}} is a variable used in html and this form variable is passed from python code. The form displayed here is what we used/coded in forms.py

To display details of either one person or for whole database, we use same html file **details.html**

```
<html>
    <style>
        table th,td{
            border:1px solid black
        }
    </style>
    <head>
        <title>DETAILS</title>
        <h1>STUDENT DETAILS</h1>
    </head>
    <body>
        <table>
        <tr>
            <th>NAME</th>
            <th>REGNO</th>
            <th>ATTENDANCE</th>
        </tr>
        {%for i in x%}
        <tr>
            <td>{{i.student_name}}</td>
            <td>{{i.student_regno}}</td>
            <td>{{i.student_attendance}}</td>
        </tr>
```

```
        {%endfor%}
        </table>
    </body>
</html>
```

It shows data as a table and

{%for I in x%}

        - - - - - - - -

{%endfor%}

is syntax for a for loop and we pass that x from python code.

Now we connect these files to views.py and urls.py

Home.html is root page . so

In views.py , create a function to render thehome.html -> html page

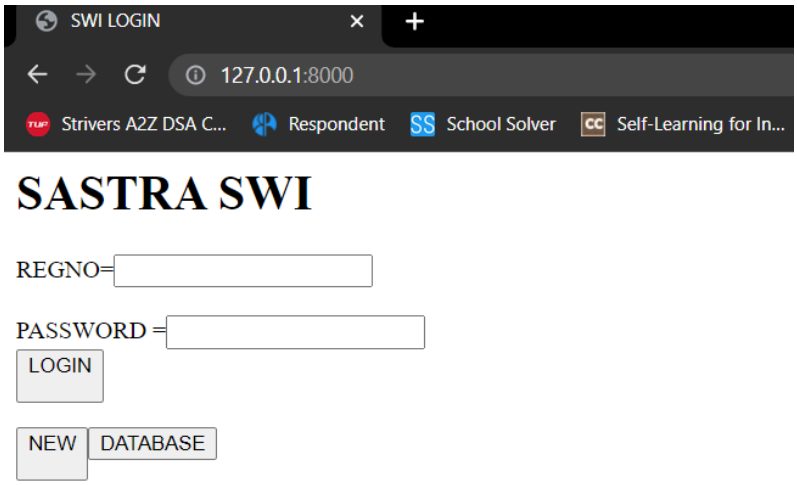
```
from django.shortcuts import render

# Create your views here.
def home_page(request):
    return render(request, 'home.html')
```

In urls.py do the following changes

```
from swi import views
```

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

from swi import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('',views.home_page),
]
```

By default with the address 127.0.0.1:8080, the html page will be rendered



Try clicking those buttons, everthing will show error 😛because we haven't linked other files yet. So, lets work on tasks of each button.

3 targets → 1. Login 2. New 3. Database

Lets go in reverse order

3. database button -> on clicking this button , it should print all the details in database on screen as a table. It uses details.html page.

Import these in views.py

```
swi > views.py > data_button
from django.shortcuts import render
from .models import student
from .forms import student_details
from django.http import HttpResponse
```

```
def data_button(request):
    students_data_from_db = student.objects.all()
    return render(request, 'details.html', {'x': students_data_from_db})
```

This function will print everything from database

Feeling bored ::::)(take a break – its overwhelming if done in a single shot

Target 2 . new button to add new user to database

In views.py add this function

```
from django.shortcuts import render
from .models import student
from .forms import student_details
from django.http import HttpResponse
# Create your views here.
def home_page(request):
    return render(request, 'home.html')

def data_button(request):
    students_data_from_db = student.objects.all()
    return render(request, 'details.html', {'x':students_data_from_db})

def new_button(request):
    if request.method == "POST":

        form = student_details(request.POST)
        if form.is_valid():
            name_from_web_page = form.cleaned_data['student_details_name']
            password_from_web_page =
form.cleaned_data['student_details_password']
            reg_no_from_web_page =
form.cleaned_data["student_details__reg_no"]
            attendance_from_Web_page =
form.cleaned_data["student_details_attendance"]
            cgpa_from_web_page = form.cleaned_data["student_details_cgpa"]

            # create new object to save data entered from web page form to
database
            new_student = student()
            new_student.student_name = name_from_web_page
            new_student.student_reg_no = reg_no_from_web_page
            new_student.student_attendance = attendance_from_Web_page
            new_student.student_cgpa = cgpa_from_web_page
            new_student.student_password = password_from_web_page
            new_student.save()
            return HttpResponse("added the object response")

    else:
        form = student_details()
```

```
    return render(request,'enter.html',{'form':form})
```

Observe the variable names here clearly, first we used student_details_name …… which is from forms.py.

Now in urls.py -> lets add url for this :)

Last target is login button

In views.py

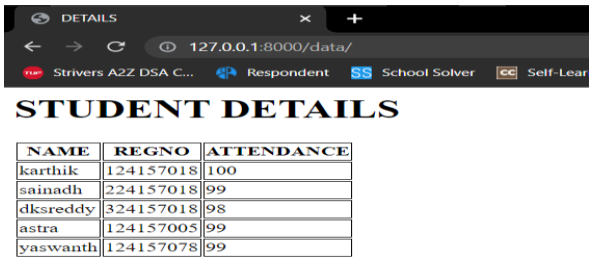
```
def login_button(request):
    reg_no_typed_on_page = request.GET['a']
    password_typed_on_page = request.GET['b']
    students_data_from_db = student.objects.filter(student_reg_no =
reg_no_typed_on_page)
    for i in students_data_from_db:
        if(int(i.student_password) == int(password_typed_on_page)):
            return render(request,'details.html',{'x':students_data_from_db})
        else:
            return HttpResponse("invalid password")
```

and in urls.py

```
"""
from django.contrib import admin
from django.urls import path

from swi import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('',views.home_page),
    path('data/',views.data_button),
    path('new/',views.new_button),
    path('login/',views.login_button),
]
```

finally everything is done 😊

On database button



NAME	REGNO	ATTENDANCE
karthik	124157018	100
sainadh	224157018	99
dksreddy	324157018	98
astra	124157005	99
yaswanth	124157078	99

On hitting new button

ADD DETAILS

Student details name: **Student details reg no:** **Student details password:**
Student details attendance: **Student details cgpa:** submit

On hitting login button

STUDENT DETAILS

NAME	REGNO	ATTENDANCE
sastraite	123456789	100