**CRUD project**

**step1**: create new environment --> conda create -n EnvName python=3.8.1

**step2**: Activate newly created environment

**step3**: Check if django is installed or not --> django-admin version

**step4**: If not Install Django --> pip install django

**step5**: Check if Django is successfully installed --> django-admin version

**step6**: Create project --> django-admin startproject ProjectName

**step7**: Navigate to that project --> cd ProjectName

**step8**: Create app for that project -->

python managee.py startapp AppName

**step9**: Create static\enroll folder inside enroll app.

**step10**: Create css and js in static\enroll folder

**step11**: Create templates folder inside enroll app.

**step12**: Create base.html file inside templates folder.

**step13**: Download bootstrap to get compiled CSS and JavaScript.

**step14**: Copy bootstrap.min.css file in css folder and

bootstrap.min.js file in js folder

**step15**: This doesn’t include documentation, source files, or any optional

JavaScript dependencies like Popper. so we have to download this.

Scroll down in bootstrap page and you will find popper.js link.

Open this link and copy the code and paste that code in popper.js

file which you have to create in js folder

**step16**: Add enroll app in installed apps in project's settings.py file.

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'enroll',

]

**setp17**: Create "addandshow.html" file in templates.

which will insert and shows data from database.

**step18**: Create "updatestudents.html" file in templates.

which will used to update.

**step19**: in base.html file we have to add bootstrap.css

for this we have to use link.css tag

<link rel="stylesheet" href="{%static 'enroll/css/bootstrap.css'%}">

<link rel="stylesheet" href="style.css">

We have to add **"{%static 'enroll/css/bootstrap.css'%}"** in href

<link rel="stylesheet" href="{% static 'enroll/css/bootstrap.min.css' %}">

**step20**: change title in base.html file as you want.

<title> CRUD Project </title>

<title>CRUD project</title>

**step21**: we have used static, so we have to load it first

otherwise we will get an error.

{% load static %}

{% load static %}

**step22**: we have to add js in body tag by using "Script src" tag.

 <script src="{% static 'enroll/js/bootstrap.js' %}"></script>

 <script src="{% static 'enroll/js/popper.js' %}"></script>

**step23**: add div.container inn body tag

<div class="container mt-5"></div>

**step24**: give heading using h2 tag in div container

**<h2> Function Based View ModelForm CRUD Project </h2>**

<div class="container mt-5">

        <h2> Function Based View ModelForm CRUD Project</h2>

    </div>

**step25**: To center this h2 tag we have to write "text-center"

To make it colorful we have to use "alert alert-danger"

 <div class="container mt-5">

        <h2 class="text-center alert alert-danger"> Function Based View ModelForm CRUD Project</h2>

    </div>

**step26**: create block content

{% block content %} {% endblock content %}

now whenever any file extend this block all this base.html code

will go to that file.

<!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>CRUD project</title>

    <link rel="stylesheet" href="{% static 'enroll/css/bootstrap.min.css' %}">

</head>

<body>

    <div class="container mt-5">

        <h2 class="text-center alert alert-danger"> Function Based View ModelForm CRUD Project</h2>

        {% block content %}

        {% endblock content %}

    </div>

    <script src="{% static 'enroll/js/bootstrap.js' %}"></script>

    <script src="{% static 'enroll/js/popper.js' %}"></script>

</body>

</html>

**step27**: extends this base.html file in

"addandshow.html" & "updatestudents.html" file

{% extends 'enroll/base.html' %}

{% block content %}

code will go here

{% endblock content %}

{% extends 'enroll/base.html' %}

{% block content %}

Code will go here

{% endblock content %}

**step28**: First render template to check if this is working or not

then we will start writing code

templates are rendered by using view function.

For this go to app's view.py file and create view function.

insert data means add student's information/data in database.

this is done by using view function that we are creating.

def add\_show(request):

return render(request, 'enroll/addandshow.html)

def add\_show(request):

    return render(request, 'enroll/addandshow.html')

**step29**: To call this function we have to write url in project’s urls.py file

first we have to import views from enroll.

from enroll import views

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', views.add\_show, name = 'addandshow'),

]

when we click on home, so this add\_show function will have to be getting called which will go in view.py file and will render 'addandshow.html' template. and will show result whatever we have written in that template.

**Step30**: In models.py file we have to add data

from django.db import models

# Create your models here.

class User(models.Model):

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

    Email = models.EmailField(max\_length=100)

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

To run model class we have to migrate the changes then only it will work. But first we have to register this model in admin if you want to cross verify the data.

**Step31**: In admin.py file

First import User from .models

from django.contrib import admin

from .models import User

# Register your models here.

@admin.register(User) # This is just a decorator

class UserAdmin(admin.ModelAdmin):

    list\_display = ('id', 'Name', 'Email', 'Password')

**step32**: python manage.py makemigrations

python manage.py migrate

python manage.py createsuperuser

now again runserver

go to admin page and enter the user name and password.

You can see that in enroll application User table has been created.

**Step33** : now we have to make form from our model class

Create **“forms.py”** file in enroll app

**step34**: in forms.py file --> from django import forms

now import model class --> from .models import User

**create class** -->

class StudentRegistration(forms.ModelForm):

class Meta: # create meta class

model = User #in model we write model class name

fields = ['Name','Email','Password']

you can use list or tuple

here we are making filed of forms

Here, we are deciding what fields can be shown in form that first it will be showing Name then Email & Password

from django import forms

from .models import User

class StudentRegistration(forms.ModelForm):

    class Meta:

        model = User

        fields = ['Name', 'Email', 'Password']

Now we have to use this form but for that we have to go to view function.

**step35**: We have to pass this form inside the template in views.py file

but for that first we have to make object of form then that object

we have to pass in the templte in views.py file

If our request is coming through "POST" that means

if method == "POST" then we have to make one object of form

if the reuqest is coming thorugh 'GET' then it will go to else part

but in this case it will generate a blank form

from django.shortcuts import render

from .forms import StudentRegistration

# Create your views here.

def add\_show(request):

    if request.method == "POST":

        fm = StudentRegistration(request.POST)

    else:

        fm = StudentRegistration()

    return render(request, 'enroll/addandshow.html', {'form': fm})

Step36: Now this 'from' key will become variable in addandshow.html file

we have to first write "form:POST tag"

don't write anything in "action" and

in method we have to write "POST"

then {{form.as\_p}} but first we have to give {% csrf\_token %}

Create one button in form tag i.e**.** type **input:b** tag and type="submit"

because we are creating a submit button that's why

then we have to give class="btn btn-success" and value="Add"

{% extends 'enroll/base.html' %}

{% block content %}

<h2> Add and Show Student </h2>

<form action="" method="post"></form>

{% csrf\_token %}

{{form.as\_p}}

<input type="Submit" class="btn btn-success" value="Add">

</form>

{% endblock content %}

step37: Now runserver and see if the form is created and is showing on the

website or not. yes, it is created but this is a html form

not bootstrap form, because all the form is not in same indentation

because we have not added bootstrap classes in this form.

step38: Go to bootstap site and go to documentation and search for "form"

then see classes in form

we should have "form-control" class in input. then only

our input field will be worked as bootstrap

step39: go to forms.py file and use "widgets"

in this first key that we will give is "name" because

we want to add class in name then same for email & password.

widgets = {

"Name":forms.TextInput(attrs={'class':'form-control'}),

"Email":forms.EmailInput(attrs={'class':'form-control'}),

"Password":forms.PasswordInput(attrs={'class':'form-control'}),

}

But in Password we have to write Password input, because

if we not use this then password will be seen in text. and if

we use PasswordInput then we will see password in ...

but we don't want this in this big rows

we have to add some style

step40: In base.html file we have written code in container block.

so we have to make raw in container and in that raw we have to put

addandshow code.

create div.row tag and in that row we can create one div

div.col-sm-4 tag

that means in 4 columns the form will be created.

{% extends 'enroll/base.html' %}

{% block content %}

<div class="row">

    <div class="col-sm-4">

        <h3> Add and Show Student </h3>

        <form action="" method="post"></form>

        {% csrf\_token %}

        {{form.as\_p}}

        <input type="Submit" class="btn btn-success" value="Add">

        </form>

    </div>

</div>

{% endblock content %}

To center the text we have to add class

class="text-center alert alert-info"

<h3 class="text-center alert alert-info"> Add new Student </h3>

{% extends 'enroll/base.html' %}

{% block content %}

<div class="row">

    <div class="col-sm-4">

        <h3 class="text-center alert alert-info"> Add new Student </h3>

        <form action="" method="post"></form>

        {% csrf\_token %}

        {{form.as\_p}}

        <input type="submit" class="btn btn-success" value="Add">

        </form>

    </div>

</div>

{% endblock content %}

**step41**: runserver again

we have created the frontend but if when we enter data then

data will not be saved in database

so to save the data in database we have to write some code.

for that go to views.py file

**step42**: when the method is POST and when we insert the data and it will

requested through POST method and the form will be submitted if we

will click on add button then this data should be saved to database

**step43**: go to views.py file

if our form will be valid then save the data to the database else

 if fm.is\_valid():

            fm.save()

here, we have validate the form and if the data has been inserted then it will be saved to database.

Or we can take particular one by one cleaned data and save it to the database

For that we have to make object for our model class.

**Step44**: from .models import User

 if fm.is\_valid():

            nm = fm.cleaned\_data['Name']

            em = fm.cleaned\_data['Email']

            pw = fm.cleaned\_data['Password']

# Then you have to create one object of model class

Reg = User(Name=nm, Email=em, Password=pd)

# Now to save this data we have to use reg instead of fm

reg.save() # instead of fm.save()

Now if I don’t want to save password in database then I don’t have to write password stuff in database like this

if fm.is\_valid():

            nm = fm.cleaned\_data['Name']

            em = fm.cleaned\_data['Email']

# Then you have to create one object of model class

Reg = User(Name=nm, Email=em)

# Now to save this data we have to use reg instead of fm

reg.save() # instead of fm.save()

Now only name and email id will be saved to database, password would not be saved to database.

If I want that after saving the data to database, the enter data will be gone from the form then I have to write blank form after saving the data to the database like this.

from django.shortcuts import render

from .forms import StudentRegistration

# Create your views here.

def add\_show(request):

    if request.method == "POST":

        fm = StudentRegistration(request.POST)

        if fm.is\_valid():

            nm = fm.cleaned\_data['Name']

            em = fm.cleaned\_data['Email']

            reg = User(Name=nm, Email=em)

            reg.save()

            fm = StudentRegistration()

    else:

        fm = StudentRegistration()

    return render(request, 'enroll/addandshow.html', {'form': fm})

step45: The data that we have entered has to be shown to user on that page

so that user can able to see that data that what i have entered.

For that go to views.py file

In views.py file

stud = User.objects.all() # this will give you all data

and we have to pass this stud object in templates as well.

stud = User.objects.all()

    return render(request, 'enroll/addandshow.html', {'form': fm,

    'stu':stud})

from django.shortcuts import render

from .forms import StudentRegistration

from .models import User

# Create your views here.

def add\_show(request):

    if request.method == "POST":

        fm = StudentRegistration(request.POST)

        if fm.is\_valid():

            nm = fm.cleaned\_data['Name']

            em = fm.cleaned\_data['Email']

            reg = User(Name=nm, Email=em)

            reg.save()

            fm = StudentRegistration()

    else:

        fm = StudentRegistration()

     stud = User.objects.all()

    return render(request, 'enroll/addandshow.html', {'form': fm,

    'stu':stud})

**step46**: go to addandshow.html file and edit this file to show data

below the col-sm-4 div add another div

add div.col-sm-8 tag then create h4 tag

<h4> Show Student information </h4>

to make it better --> make it center and add color.

<h4 class=text-center alert alert-info> Show Student information </h4>

<div class="col-sm-8">

        <h4 class="text-center alert alert-info">Show Student Information</h4>

**step47**: runserver and check if show student information is showing or not

**step48**: Now we have to make one table which will show the student info

and table has been created if and only if there is any record saved

and for that first we have to check it using if statement

 {% if stu %}

        {% endif %}

{% extends 'enroll/base.html' %}

{% block content %}

<div class="row">

    <div class="col-sm-4">

        <h4 class="text-center alert alert-info"> Add New Student </h4>

        <form action="" method="post">

        {% csrf\_token %}

        {{form.as\_p}}

        <input type="submit" class="btn btn-success" value="Add">

        </form>

    </div>

    <div class="col-sm-8">

        <h4 class="text-center alert alert-info">

            Show Student Information</h4>

        {% if stu %}

        <h4> Table Data </h4>

        {% else %}

        <h4 class="text-center alert alert-warning"> No Records </h4>

        {% endif %}

    </div>

</div>

{% endblock content %}

If there is data available in stu then it will show Table data else it will show No records. This is I have written just to check if this is working or not.

But actually i have to create table to show the data if data is available and for that i just go to bootstrap and copy the code of table tag.

**step49**: copy the table with hoverable rows and paste it to addandshow file

{% if stu %}

            <table class="table table-hover">

                <thead>

                    <tr>

                      <th scope="col">ID</th>

                      <th scope="col">Name</th>

                      <th scope="col">Email</th>

                      <th scope="col">Password</th>

                    </tr>

                  </thead>

            </table>

        {% else %}

        <h4 class="text-center alert alert-warning"> No Records </h4>

        {% endif %}

after creating the table we have to add data that we have passed in stu. we have to fetch the data from stu and show that into this table. And for that we will use for loop we are going to write one by one data. and we have write this loop inside <tbody>

{% if stu %}

            <table class="table table-hover">

                <thead>

                    <tr>

                      <th scope="col">ID</th>

                      <th scope="col">Name</th>

                      <th scope="col">Email</th>

                      <th scope="col">Password</th>

                      <th scope="col">Action</th>

                    </tr>

                  </thead>

                  <tbody>

                      {% for st in stu %}

                      <tr>

                          <th scope="row"> {{st.id}} </th>

                          <td> {{st.Name}} </td>

                          <td> {{st.Email}} </td>

                          <td> {{st.Password}} </td>

                      </tr>

                      {% endfor %}

                  </tbody>

            </table>

        {% else %}

        <h4 class="text-center alert alert-warning"> No Records </h4>

        {% endif %}

If we refresh and see then we are able to see data in the table that we have entered till now but there is another column called Action and it is empty but we want buttons there in that column. We are not going to make buttons for each and every data. So we add edit and delete button in the row scope after the data that means after <td> i.e. after Name, Email and Password.

To make button small we write “btn-sm” we are making two buttons edit and delete using “a tag”

<td>

      <a href="#" class="btn btn-warning btn-sm">Edit</a>

      <a href="#" class="btn btn-danger btn-sm">Delete</a>

</td>

Now refresh the page and you will see that the buttons are added now we want to add some space between these two columns i.e., Add new student and show new student. So for that we have to make column size as 7 from 8 and offset-1, so this will make gap of one column.

<div class="col-sm-7 offset-1">

        <h4 class="text-center alert alert-info">

            Show Student Information</h4>

{% extends 'enroll/base.html' %}

{% block content %}

<div class="row">

    <div class="col-sm-4">

        <h4 class="text-center alert alert-info"> Add New Student </h4>

        <form action="" method="post">

        {% csrf\_token %}

        {{form.as\_p}}

        <input type="submit" class="btn btn-success" value="Add">

        </form>

    </div>

    <div class="col-sm-7 offset-1">

        <h4 class="text-center alert alert-info">

            Show Student Information</h4>

        {% if stu %}

            <table class="table table-hover">

                <thead>

                    <tr>

                      <th scope="col">ID</th>

                      <th scope="col">Name</th>

                      <th scope="col">Email</th>

                      <th scope="col">Password</th>

                      <th scope="col">Action</th>

                    </tr>

                  </thead>

                  <tbody>

                      {% for st in stu %}

                      <tr>

                          <th scope="row"> {{st.id}} </th>

                          <td> {{st.Name}} </td>

                          <td> {{st.Email}} </td>

                          <td> {{st.Password}} </td>

                          <td>

                              <a href="#" class="btn btn-warning btn-sm">Edit</a>

                              <a href="#" class="btn btn-danger btn-sm">Delete</a>

                          </td>

                      </tr>

                      {% endfor %}

                  </tbody>

            </table>

        {% else %}

        <h4 class="text-center alert alert-warning"> No Records </h4>

        {% endif %}

    </div>

</div>

{% endblock content %}

Now I want to show password in the column as well. So I have to add this

pd = fm.cleaned\_data['Password']

            reg = User(Name=nm, Email=em, Password=pd)

**step50**: But if I want to edit or delete some data then the buttons are not working right now so for that I have to write some code for that. We have to write view function for delete and have to add some url

We have to make one form in which if we click on delete button the form will be submitted and it will go to “POST” request when we delete some data.

When we make object by writing User.object.all() but we don’t want to delete all so to delete particular data we have to write

Pi = User.object.get(pk=”id”) # pk means primary key

And from where id will come it will come from dynamic url that we have added in the parenthesis when we have created a class called Delete\_Data

And then we have to delete this object by using delete method.

Pi.delete()

Then we have to redirect this and for that we have to import HttpResponseRedirect first

Return HttpResponseRedirect(‘/’)

# This Function will Delete Item from Table

def Delete\_Data(request, id):

    if request.method == 'POST':

        pi = User.objects.get(pk=id)

        pi.delete()

        return HttpResponseRedirect('/')

To run this Delete\_Data function we have to define url for this and to define urls go to urls.py file

path('delete/<int:id>/', views.Delete\_Data, name="deletedata)

path('delete/<int:id>/', views.Delete\_Data, name='deletedata')

but how id will come to urls <int:id>, it will come from addandshow..html file where we have written Delete button in a tag remove that tag and add one form using "form:POST tag" and after that we have to create submit button by using "input:button tag" and change type form button to submit and give value as Delete and give some class so that it will look like a button

<input type='submit' class='btn btn-danger' value='Delete'>

<td>

      <a href="#" class="btn btn-warning btn-sm">Edit</a>

      <form action="" method="post">

             <input type="submit" class="btn btn-danger" value="Delete">

      </form>

</td>

But if we refresh the page, now the Delete button has been showing in second line. So we have to make it inline.

for that we have to add one class of bootstrap which will be used to make it inline.

<form action="" method="post" class="d-inline">

                                  <input type="submit" class="btn btn-danger"

                                  value="Delete">

                              </form>

When we click on Delete button one url will be generated so we have to add this to action

<form action=”{% url ‘deletedata’ %}” method=”POST” class=”d-inline”>

 <form action="{% url 'deletedata' st.id %}" method="post" class="d-inline">

                                  <input type="submit" class="btn btn-danger"

                                  value="Delete">

                              </form>

<form action="{%url 'deletedata' st.id %}" method="POST" class="d-inline">

{% url 'deletedata' st.id %} this will make url

**Always remeber whenever we add form we have to add csrf\_token otherwise it will give you an error.**

You can cross-verify in admin page

**step51**: Now the next thing we have to do is update the data, when we click on edit button it will update the data. if we click on edit button nothing is happening here because i have not created any function for that we have to create the function in views.py file

In the process of updating the data we have to do two things, first we have to retrieve the data but for specific id and we have to save that data for the specific id.

**step52**: To update the data 🡪 go to updatestudents.html file

But first we have to make one function for that in views.py file.

now here also we have to make our request dynamic becasue we have to work on specific data and have to update the item for specific id. we have to render updatestudents.html template when Update\_date function will be getting called. but we have to add this in urls.

we have to make our url dynamic so for that as we have done in delete url same we have to do for update url

path('<int:id/', views.Update\_data, name=updatedata)

whichever id will be called according to that Update\_data function will be getting called and run the view function and updatestudents template will be rendered. and when will url got hit when we click on Edit button.

so in href we have to write

<a href="{% url 'name that we have wriiten inside urls.py file'

that means 'updatedata' st.id %}

in views.oy file 🡪

# This Function will Update/Edit the Item

def Update\_Data(request, id):

    return render(request, 'enroll/updatestudents.html', {'id': id})

in urls.py file 🡪

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', views.add\_show, name = 'addandshow'),

    path('delete/<int:id>/', views.Delete\_Data, name='deletedata'),

    path('<int:id>/', views.Update\_Data, name='updatedata'),

]

In updatestudents.html file 🡪

{% extends 'enroll/base.html' %}

{% block content %}

<h4> Update Student {{id}} </h4>

{% endblock content %}

In addandshow.html file 🡪

<td>

   <a href="{% url 'updatedata' st.id %}" class="btn btn-warning btnsm">Edit</a>

   <form action="{% url 'deletedata' st.id %}" method="post" class="d-inline">

       {% csrf\_token %}

      <input type="submit" class="btn btn-danger" value="Delete">

   </form>

</td>

Now when we refresh the page and click on edit button of first id then it is showing “Update Student 1”

**step53**: Now we have to edit the code, we do not want blank form we want form with data, so that we can edit it. we cannot edit the blank form.

In views.py file 🡪

# This Function will Update/Edit the Item

def Update\_Data(request, id):

    if request.method == "POST":

        pi = User.objects.get(pk=id)

fm = StudentRegistration(request.POST, instance=pi)

if fm.is\_valid():

            fm.save()

    return render(request, 'enroll/updatestudents.html', {'id': id})

or we can write cleaned\_data if we want to clean data one by one but there is no need to do this because we have saved this data after cleaning it so now we can edit and save this data directly.

This all will be done if any only if we click on update button that means if we click on update button then the request will come through POST but if we don't click on update button then the request will come through get.

That means when you click on Edit button so request will come through get and for get we have to write some code, that means we write this code in else part.

else:

            pi = User.objects.get(pk=id)

            fm = StudentRegistration(instance=pi)

    return render(request, 'enroll/updatestudents.html', {‘id’: id}

and you have to change in return part also because we have to pass data not id so we have to write form not id

return render(request, 'enroll/updatestudents.html', {'form':fm})

return render(request, 'enroll/updatestudents.html', {'form': fm})

def Update\_Data(request, id):

    if request.method == 'POST':

        pi = User.objects.get(pk=id)

        fm = StudentRegistration(request.POST, instance=pi)

        if fm.is\_valid():

            fm.save()

    else:

        pi = User.objects.get(pk=id)

        fm = StudentRegistration(instance=pi)

    return render(request, 'enroll/updatestudents.html', {'form': fm})

**step54**: Now we are sending form so we have to write some code in updatestudents.html file

First make row tag by writing “div.row” tag

<div class="row">

then we make small 8 columns and offset 2 columns by using **div.col-sm-8 tag**

<div class="col-sm-8 offset-2">

Then write h4 tag

<h4> Update Student information </h4>

now we add some classes of bootstrap to make it attractive

<h4 class="alert alert-info"> Update Student Information </h4>

now create form by using form:POST tag

<form action="" method="post">

write csrf\_token by using tag csrf\_token tag

{% csrf\_token %}

Display the form by using form variable that is written inside views.py file

{{form.as\_p}}

now create one button which will update the data by using input:submit tag & give some bootstrap class to make it attractive

<input type="submit" class="btn btn-success" value="Update">

{% extends 'enroll/base.html' %}

{% block content %}

<div class="row">

    <div class="col-sm-8 offset-2">

        <h4 class="alert alert-info"> Update Student Information </h4>

        <form action="" method="post">

            {% csrf\_token %}

            {{form.as\_p}}

            <input type="submit" class="btn btn-success" value="Update">

        </form>

    </div>

</div>

{% endblock content %}

**step55**: if we refresh the page and when we click on edit button the data will be displayed that is name, email data are being displayed but password is not being displayed, it is blank

**step56**: now we have to create one “back to home” button.

To create back to home button, we create one tag by using "a tag" and we set one url in href and to go to home page we have to add name of home url that we have written in urls.py file.

So, for homepage the url name is "addandshow", so we have to write this name in url and we add some class of bootstrap so that this will look like a button.

<a href="{% url 'addandshow' %}" class="btn btn-info"> Back To Home </a>

If we refresh the page and when click on edit button we can see that "Back To Home" button has added and it is working as well.

**step57**: Now to show password in update page, if we want to show this then we have to change widgets.

So, to change widgets go to forms.py file and we have to make some changes in password that means we have to make render\_value True, by default render\_value is False so we have to make it True to show the password in update page.

**In forms.py file** 🡪

widgets = {

            "Name":forms.TextInput(attrs={'class':'form-control'}),

            "Email":forms.EmailInput(attrs={'class':'form-control'}),

            "Password":forms.PasswordInput(render\_value=True, attrs=

            {'class':'form-control'})

        }

from django import forms

from .models import User

class StudentRegistration(forms.ModelForm):

    class Meta:

        model = User

        fields = ['Name', 'Email', 'Password']

        widgets = {

            "Name":forms.TextInput(attrs={'class':'form-control'}),

            "Email":forms.EmailInput(attrs={'class':'form-control'}),

            "Password":forms.PasswordInput(render\_value=True, attrs=

            {'class':'form-control'})

        }

Now if we refresh the page and when we click on edit button now the password is showing in dot dot format.

When you click on edit button request will come through get that's why we can see the id in search query and that id will be passed in st.id as st.1 in updatedata url in addandshow.html file and then it will go to updatedata url in urls.py file id 1 will be passed in <int:1> and then Update\_Data function will be getting called in views.py and as id 1 will be passed in view function and it will check if request.method=="POST" but here the method is get so this part that means if block will be skipped and it will go to else block and there id will be passed as 1 i.e. pk=1 (pk=id) means primary key=1 means what primary keys will always be generated by itself that means automatically. and whatever id we will pass that means whatever edit button we will click on that id will be passed and will saved in pi object and then we will pass that pi object in instance variable i.e. instance=pi so that we will get the data in fm according to the id that we have passed that means whatever edit button we have clilked on. according to that id data will be passed in fm and then it will show only that data with the form in update page becasue we have rendered that data using udadatestudent.html template.

Now if we change the data in update page and when we click on update button so now this is a form, so now the request will be passed through POST method, then again it will go to Update\_Data function and check the method if the method is POST then it will go to if block and then id will be passed in pk that means pk=id and it will saved to pi object. then we saved this pi in instance variable and the request is post so it will be saved to fm object and will validate that data in fm object and will save that data.

Now the updated data will be saved to the database. and then we have to click on Back to home button to go to home page.