CRUD OPERATIONS

C - Create (Create data | Insert)

R - Retrieve (Select Query)

U - Update (Update Query)

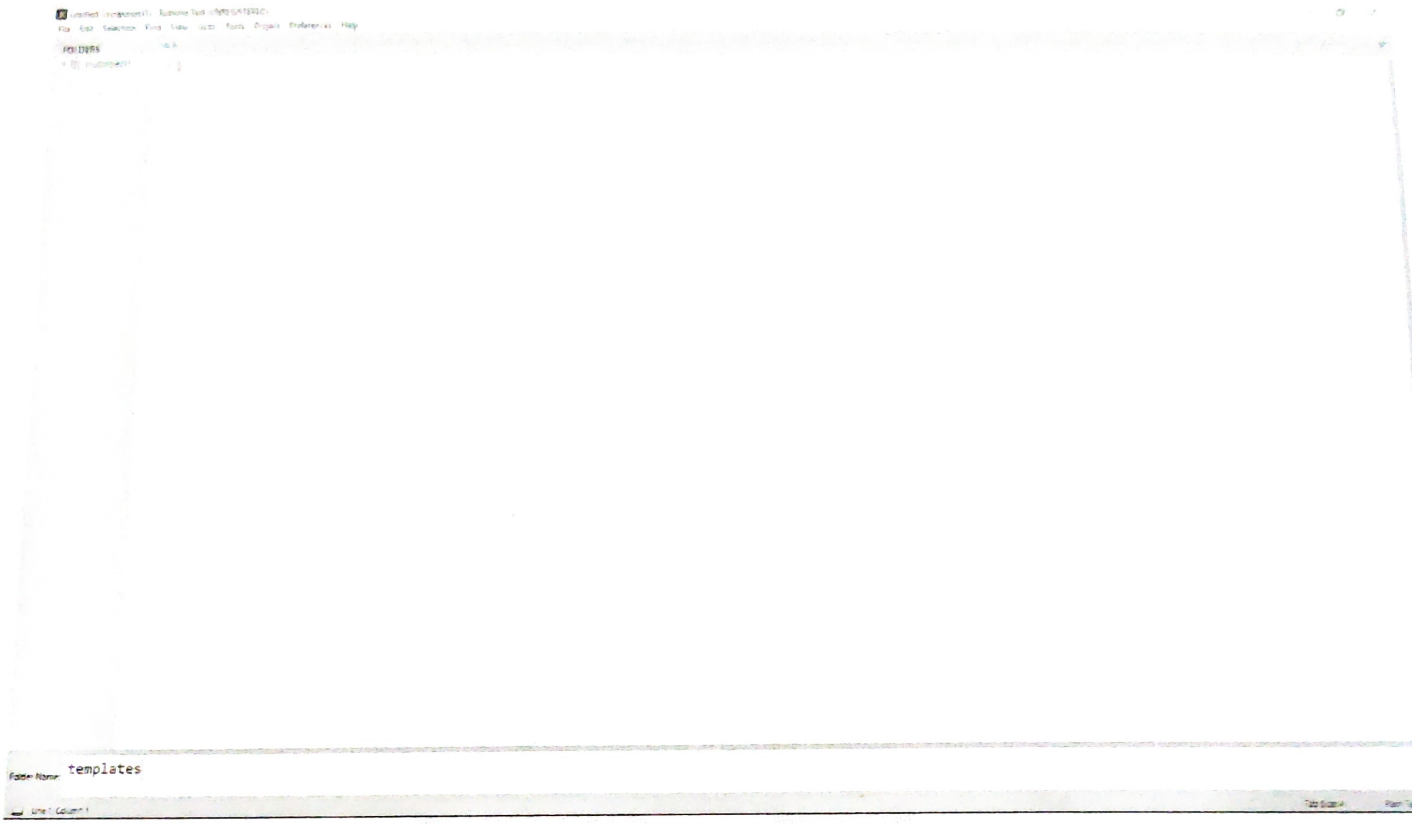
D - Delete (Delete Query)

1. PROJECT : CREATE A STUDENT WEB APPLICATION

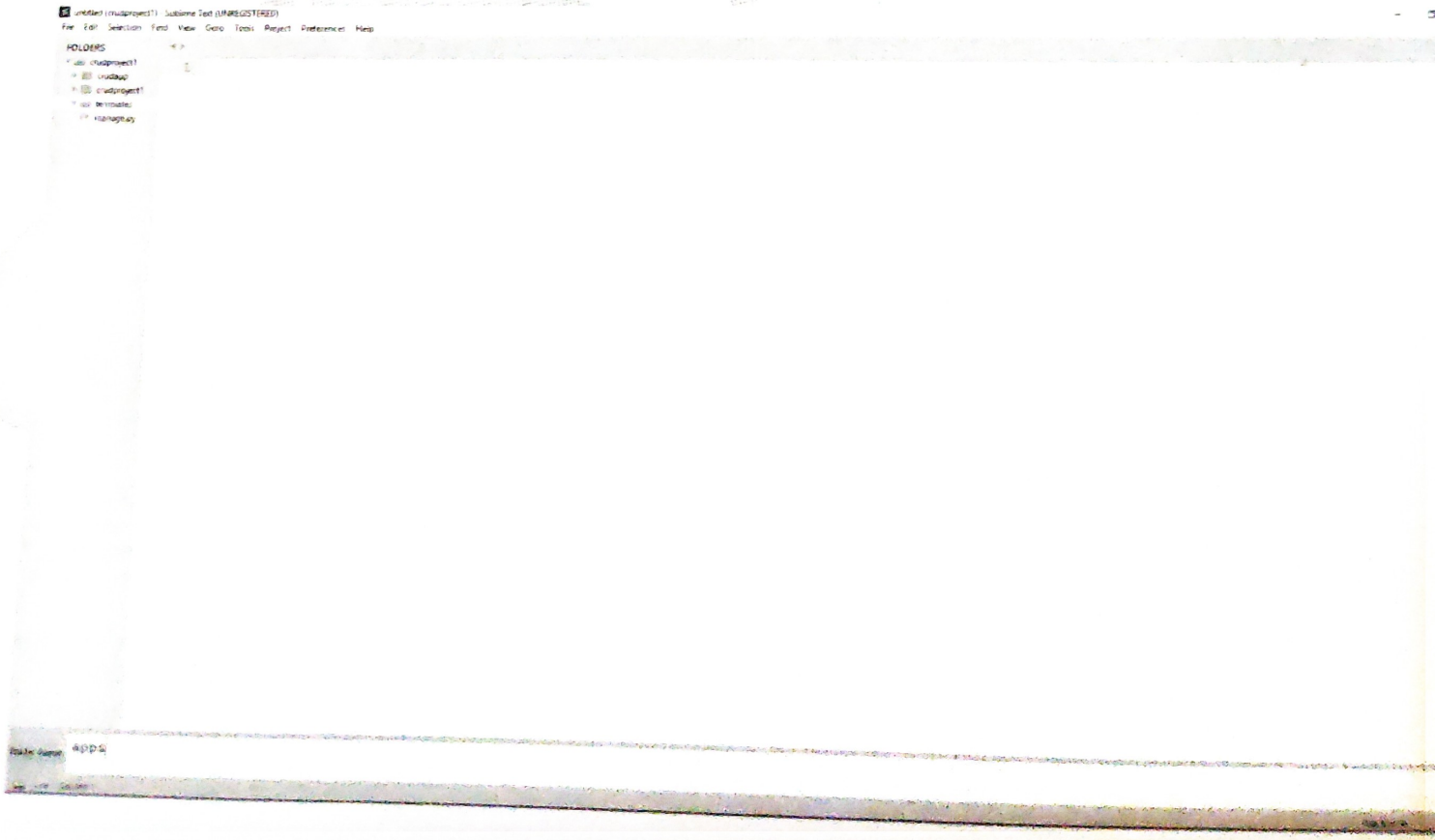
1. Create **Project and Application.**

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.16299.1087]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\systech>d:

D:\>cd Django

D:\Django>django-admin startproject crudproject1

D:\Django>cd crudproject1

D:\Django\crudproject1>python manage.py startapp crudapp

D:\Django\crudproject1>
```

2. *Click* **Templates.**



3. Create **apps** folder.



4. Configuring **Setting** file.

```
from pathlib import Path
import os
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
T_dir=os.path.join(BASE_DIR,'templates')

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-8-_w%!&h!dhrzkdvf#3q$10ud!%-sanr)q()fnu4jbo4_4eco'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

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

5. Configuring **Settings** file (add Template file).

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [T_dir],
        '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',
            ],
        },
    },
]
```

6. Creating **models.**

models.py

```
from django.db import models
class Student(models.Model):
    sno=models.IntegerField()
    name=models.CharField(max_length=64)
    age=models.IntegerField()
    dep=models.CharField(max_length=256)
```

7. Type command : **python manage.py makemigrations**

```
D:\Django\crudproject1>python manage.py makemigrations
Migrations for 'crudapp':
  crudapp\migrations\0001_initial.py
    - Create model Student

D:\Django\crudproject1>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, crudapp, sessions
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 crudapp.0001_initial... OK
  Applying sessions.0001_initial... OK

D:\Django\crudproject1>
```

8. Type command: **python manage.py createsuperuser**

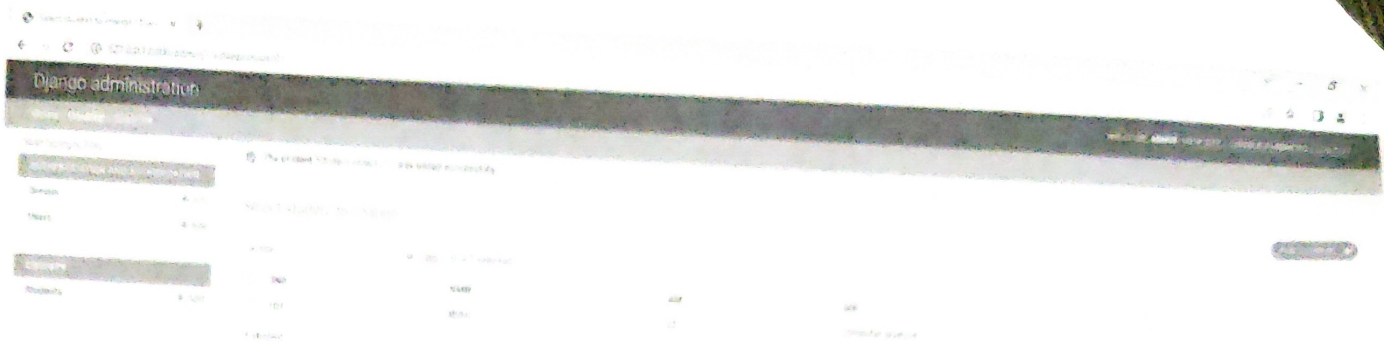
```
D:\Django\crudproject1>python manage.py createsuperuser
Username (leave blank to use 'systech'): admin
Email address: admin@gmail.com
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

D:\Django\crudproject1>
```

9. Configuring **Admin** file.

admin.py

```
from django.contrib import admin
from crudapp.models import Student
class StudentAdmin(admin.ModelAdmin):
    list_display=['sno','name','age','dep']
admin.site.register(Student,StudentAdmin)
```

10. Store in admin panel.



11. Configuring **View file.**

views.py

```
from django.shortcuts import render
from crudapp.models import Student

def read(request):
    student=Student.objects.all()
    return render(request,'apps/result.html',{'s':student})
```

12. Configuring url file.

urls.py

```
from django.contrib import admin
from django.urls import path
from crudapp import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('result/',views.read)
]
```

13. Create html file.

result.html

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<center>
<body>
<h2>STUDENT INFORMATION</h2>
<table border='2'>
<thead>
    <th>Student No</th>
    <th>Student Name</th>
    <th>Student Age</th>
    <th>Student Department</th>
</thead>
{%for i in s%}
<tr>
    <td>{{i.sno}}</td>
    <td>{{i.name}}</td>
    <td>{{i.age}}</td>
    <td>{{i.dep}}</td>
</tr>
{%endfor%}
</table>
</body>
</center>
</html>
```

14. Verify the result.

STUDENT INFORMATION

Student No	Student Name	Student Age	Student Department
101	abdul	25	computer science

15. Create **form.py**

forms.py

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


class StudentForm(forms.ModelForm):
    class Meta:
        model=Student
        fields='__all__'
```

16. Configuring **View** file

```
from django.shortcuts import render
from crudapp.models import Student
from crudapp.forms import StudentForm

def read(request):
    student=Student.objects.all()
    return render(request,'apps/result.html',{'s':student})
def insert(request):
    form=StudentForm()
    if request.method=="POST":
        form=StudentForm(request.POST)
        if form.is_valid():
            form.save()
    return render(request,'apps/insert.html',{'form':form})
```

17. Create inserting html file.

insert.html

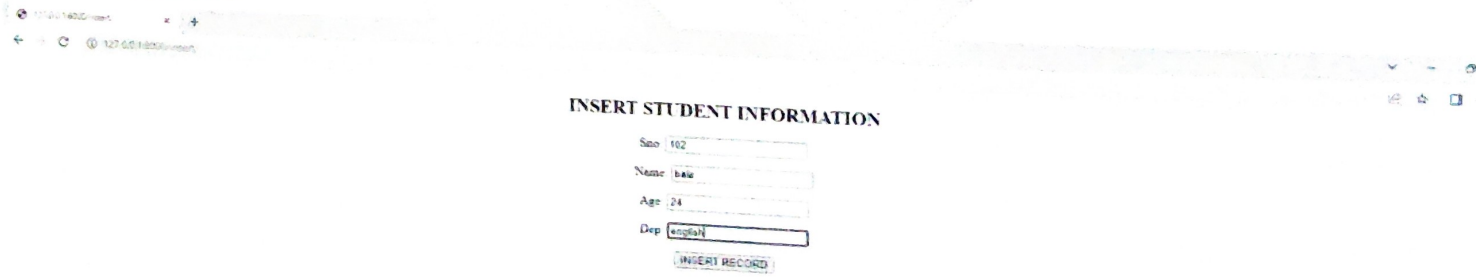
```
<!DOCTYPE html>
<html>
<head>
    <title></title>
        <!-- Latest compiled and minified CSS -->
</head>
<center>
<body>
<h2>INSERT STUDENT INFORMATION</h2>
<form method="POST">
    {{form.as_p}}
    {%csrf_token%}
    <input type="submit" value="INSERT RECORD">
</form>
</body>
</center>
</html>
```

18. Configuring url file

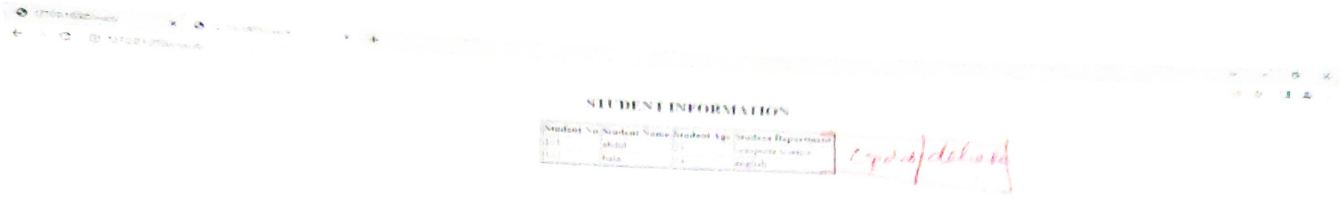
urls.py

```
from django.contrib import admin
from django.urls import path
from crudapp import views
urlpatterns = [
    path('admin/', admin.site.urls),
     path('result/',views.read),
     path('insert/',views.insert)
]
```

19. Insert the record



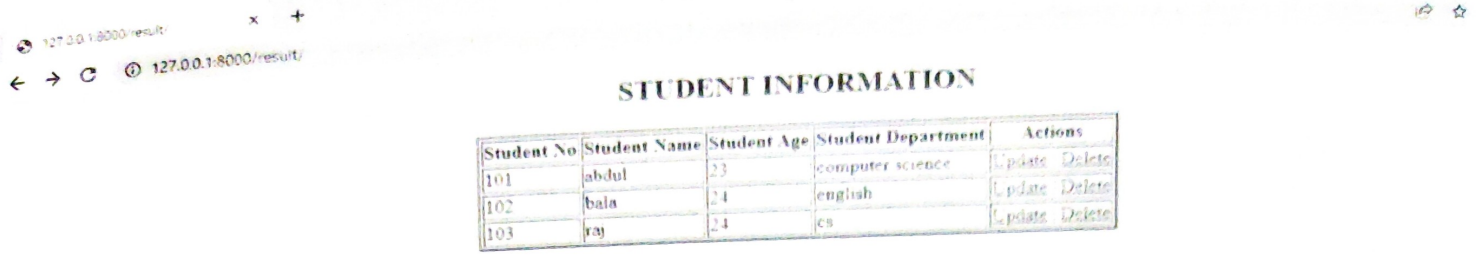
20. Verify the result.



21. Creating result html page.

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<center>
<body>
<h2>STUDENT INFORMATION</h2>
<table border='2'>
<thead>
    <th>Student No</th>
    <th>Student Name</th>
    <th>Student Age</th>
    <th>Student Department</th>
    <th>Actions</th>

</thead>
{%for i in s%}
<tr>
    <td>{{i.sno}}</td>
    <td>{{i.name}}</td>
    <td>{{i.age}}</td>
    <td>{{i.dep}}</td>
    <td><a href="#">Update</a> | <a href="#">Delete</a></td>

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

22. Verify the result.



23. Configuring **View** file

views.py

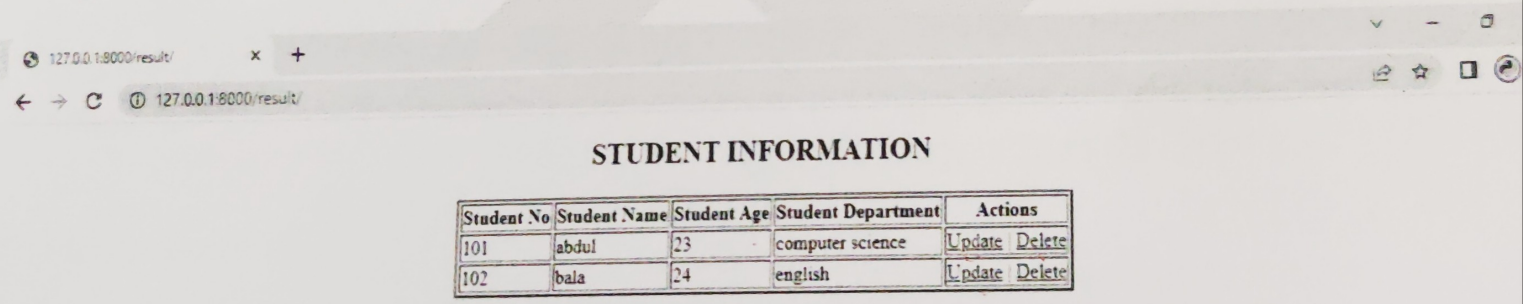
```
from django.shortcuts import render,redirect
from crudapp.models import Student
from crudapp.forms import StudentForm

def read(request):
    student=Student.objects.all()
    return render(request,'apps/result.html',{'s':student})
def insert(request):
    form=StudentForm()
    if request.method=="POST":
        form=StudentForm(request.POST)
        if form.is_valid():
            form.save()
    return render(request,'apps/insert.html',{'form':form})
def delete(request,id):
    s=Student.objects.get(id=id)
    s.delete()
    return redirect('/result')
```

24. Configuring url file

urls.py

```
from django.contrib import admin
from django.urls import path
from crudapp import views
urlpatterns = [
    path('admin/', admin.site.urls),
     path('result/',views.read),
     path('insert/',views.insert),
     path('delete/<int:id>', views.delete)
]
```

25. Verify the result.



STUDENT INFORMATION

Student No	Student Name	Student Age	Student Department	Actions
101	abdul	23	computer science	Update Delete
102	bala	24	english	Update Delete

26. Configuring **View** file

views.py

```
from django.shortcuts import render,redirect
from crudapp.models import Student
from crudapp.forms import StudentForm

def read(request):
    student=Student.objects.all()
    return render(request,'apps/result.html',{'s':student})
def insert(request):
    form=StudentForm()
    if request.method=="POST":
        form=StudentForm(request.POST)
        if form.is_valid():
            form.save()
    return render(request,'apps/insert.html',{'form':form})
def delete(request,id):
    s=Student.objects.get(id=id)
    s.delete()
    return redirect('/result')
def update(request,id):
    student=Student.objects.get(id=id)
    if request.method=="POST":
        form=StudentForm(request.POST, instance=student)
        if form.is_valid():
            form.save()
        return redirect('/result')
    return render(request,'apps/update.html',{'s':student})
```

27. Creating html file

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
        <!-- Latest compiled and minified CSS -->
</head>
<center>
<body>
<h2>UPDATE STUDENT INFORMATION</h2>
<form method="POST">
    {%csrf_token%}
    Student Number: <input type="text" name="sno" value="{{s.sno}}"><br>
    Student Name: <input type="text" name="name" value="{{s.name}}"><br>
    Student Age: <input type="text" name="age" value="{{s.age}}"><br>
    Department: <input type="text" name="dep" value="{{s.dep}}"><br>

    <input type="submit" value="UPDATE RECORD">
</form>
</body>
</center>
</html>
```

28. Configuring url file.

urls.py

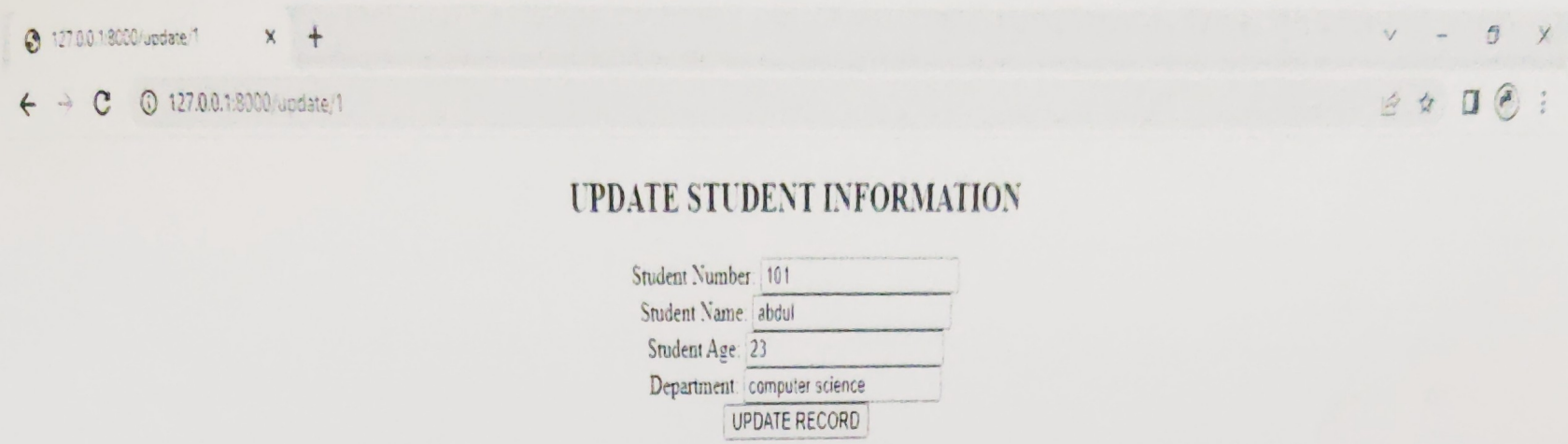
```
from django.contrib import admin
from django.urls import path
from crudapp import views
urlpatterns = [
    path('admin/', admin.site.urls),
     path('result/',views.read),
     path('insert/',views.insert),
     path('delete/<int:id>', views.delete),
     path('update/<int:id>', views.update)
]
```

29. Creating update and delete html page.

result.html

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<center>
<body>
<h2>STUDENT INFORMATION</h2>
<table border='2'>
<thead>
    <th>Student No</th>
    <th>Student Name</th>
    <th>Student Age</th>
    <th>Student Department</th>
    <th>Actions</th>
</thead>
{%for i in s%}
<tr>
    <td>{{i.sno}}</td>
    <td>{{i.name}}</td>
    <td>{{i.age}}</td>
    <td>{{i.dep}}</td>
    <td><a href="/update/{{ i.id }}">Update</a> | <a href="/delete/{{ i.id }}">Delete</a></td>
</tr>
{%endfor%}
</table>
</body>
</center>
</html>
```

OUTPUT:

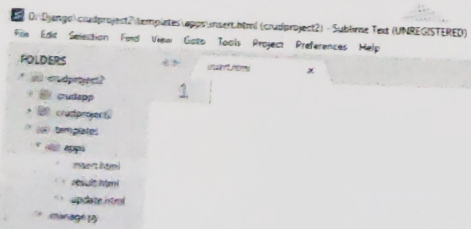

127.0.0.1:8000/update/1
127.0.0.1:8000/update/1
UPDATE STUDENT INFORMATION
Student Number: 101
Student Name: abdul
Student Age: 23
Department: computer science
UPDATE RECORD

2. PROJECT : CREATE A PRODUCT WEB APPLICATION

1. Create Project and Application

```
Microsoft Windows [Version 10.0.16299.1087]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\systech>d:

D:\>cd Django

D:\Django>django-admin startproject crudproject2

D:\Django>
D:\Django>cd crudproject2

D:\Django\crudproject2>python manage.py startapp crudapp

D:\Django\crudproject2>
```

2. Create templates, apps folder and create three html page.



3. Add **app name and template directory** in Settings file

```
from pathlib import Path
import os
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
T_dir=os.path.join(BASE_DIR,'templates')

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-8k+5b+l@s-q5s@%kepq!3b^$2-c@(0q5+c7o-b#ha*o()6s

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

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

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [T_dir],
        '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',
            ],
        },
    },
]
```

4. Create **Database in model** file

models.py

```
from django.db import models

class Product(models.Model):
    pno=models.IntegerField()
    name=models.CharField(max_length=64)
    price=models.IntegerField()
    warenty=models.CharField(max_length=100)
```

5. Type command : **python manage.py makemigrations**

Python manage.py migrate

```
D:\Django\crudproject2>python manage.py makemigrations
Migrations for 'crudapp':
  crudapp\migrations\0001_initial.py
    - Create model Product

D:\Django\crudproject2>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, crudapp, sessions
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 crudapp.0001_initial... OK
  Applying sessions.0001_initial... OK

D:\Django\crudproject2>
```

6. Type command : **python manage.py createsuperuser.**

```
D:\Django\crudproject2>python manage.py createsuperuser
Username (leave blank to use 'systech'): admin
Email address: admin@gmail.com
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

D:\Django\crudproject2>
```

7. Configuring **admin** file.

admin.py

```
from django.contrib import admin
from crudapp.models import Product
class ProductAdmin(admin.ModelAdmin):
    list_display=['pno','name','price','warenty']
admin.site.register(Product,ProductAdmin)
```

8. Creating **forms.py**

forms.py

```
from django import forms
from crudapp.models import Product

class ProductForm(forms.ModelForm):
    class Meta:
        model=Product
        fields='__all__'
```

9. Configuring **url** file

urls.py

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('result/', views.read),
    path('insert/', views.insert),
    path('delete/<int:id>', views.delete),
    path('update/<int:id>', views.update)
]
```

10.Configuring **View** file

```
from django.shortcuts import render,redirect
from crudapp.models import Product
from crudapp.forms import ProductForm
def read(request):
    product=Product.objects.all()
    return render(request,'apps/result.html',{'p':product})
def insert(request):
    form=ProductForm()
    if request.method=="POST":
        form=ProductForm(request.POST)
        if form.is_valid():
            form.save()
    return render(request,'apps/insert.html',{'form':form})
def delete(request,id):
    p=Product.objects.get(id=id)
    p.delete()
    return redirect('/result')
def update(request,id):
    product=Product.objects.get(id=id)
    if request.method=="POST":
        form=ProductForm(request.POST, instance=product)
        if form.is_valid():
            form.save()
        return redirect('/result')
    return render(request,'apps/update.html',{'p':product})
```

11.Creating **Inserting html** file.

```
<html>


<center>
<body>
<h2 style="color: red">INSERT PRODUCT INFORMATION</h2>
<form method="POST" style="color: blue">
    {{form.as_p}}
    {%csrf_token%}
    <input type="submit" value="INSERT RECORD">
</form>
</body>
</center>
</html>
```

12.Configuring **Result** page.

result.html

```
<!DOCTYPE html>
<html>
<style>
    table{
        border-color: white;}
    th{
        color: blue;
        border-color: white;}</style>
<center>
<body>
<h2 style="text-align: center;color: red">PRODUCT LISTS</h2>
<table border='2'>
<thead>
    <th>Product No</th>
    <th>Product Name</th>
    <th>Product Price</th>
    <th>Product Warenty</th>
    <th>Actions</th>
</thead>
{%for i in p%}
<tr>
    <td>{{i.pno}}</td>
    <td>{{i.name}}</td>
    <td>{{i.price}}</td>
    <td>{{i.warenty}}</td>
    <td><a href="/update/{{ i.id }}">Update</a> | <a href="/delete/{{ i.id }}">Delete</a></td>
</tr>
{%endfor%}
</table>
</body>
</center>
</html>
```

13.Configuring **update** page.

update.html

```
<!DOCTYPE html>
<html>
<center>
<body>
<h2 style="color: red">UPDATE PRODUCT INFORMATION</h2>
<form method="POST" style="color: blue">
    {%csrf_token%}
    PRODUCT NO: <input type="text" name="NO" value="{{p.pno}}"><br>
    PRODUCT NAME: <input type="text" name="name" value="{{p.name}}"><br>
    PRODUCT PRICE: <input type="text" name="price" value="{{p.price}}"><br>
    PRODUCT WARENTY: <input type="text" name="warenty" value="{{p.warenty}}"><br>

    <input type="submit" value="UPDATE RECORD">
</form>
</body>
</center>
</html>
```

OUTPUT: