Employee Contact Management System (Django MySQL)

Virtual Environment

It is suggested to have a dedicated virtual environment for each Django project, and one way to manage a virtual environment is venv, which is included in Python.

The name of the virtual environment is your choice, in this tutorial we will call it `myworld`.

Type the following in the command prompt, remember to navigate to where you want to create your project:

Windows:

```
py -m venv myworld
```

This will set up a virtual environment, and create a folder named "myworld" with subfolders and files, like this:

```
myworld
  Include
  Lib
  Scripts
  pyvenv.cfg
```

Then you have to activate the environment, by typing this command:

Windows:

```
myworld\Scripts\activate.bat
```

Once the environment is activated, you will see this result in the command prompt:

Windows:

```
(myworld) F:\Sannya\SSGC\Django\myworld>
```

Install Django

Now, that we have created a virtual environment, we are ready to install Django.

Django is installed using pip, with this command:

Windows:

```
(myworld) F:\Sannya\SSGC\Django\myworld>py -m pip install Django
```

My First Project

Once you have come up with a suitable name for your Django project, like mine: `django_mysql_crud`, navigate to where in the file system you want to store the code (in the virtual environment), I will navigate to the `myworld` folder, and run this command in the command prompt:

```
django-admin startproject django_mysql_crud
```

Django creates a `django_mysql_crud` folder on my computer, with this content:

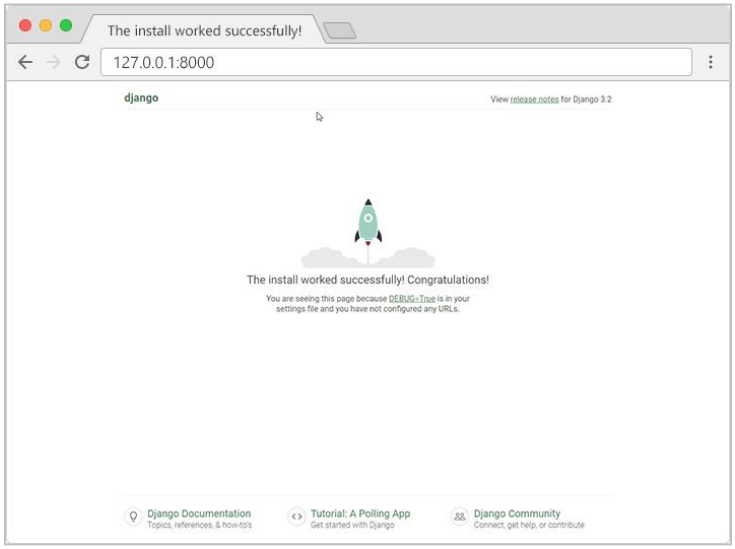
```
django_mysql_crud
    manage.py
    django_mysql_crud/
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
```

In order to run the project, open Python IDE(Pycharm in my case) and navigate to this project. After the folder is open, open the terminal and enter the following command to run the project.

```
Py manage.py runserver
```

Open a new browser window and type [127.0.0.1:8000](127.0.0.1:8000) in the address bar.

The result:



Create App

I will name my app `employee`.

Start by navigating to the selected location where you want to store the app, in my case the `django_msql_crud` folder, and run the command below.

If the server is still running, and you are not able to write commands, press [CTRL] [BREAK], or [CTRL] [C] to stop the server and you should be back in the virtual environment.

```
py manage.py startapp employee
```

Django creates a folder named `employee` in my project, with this content:

```
django_msql_crud
    manage.py
    django_msql_crud/
    employee/
        migrations/
            __init__.py
        __init__.py
        admin.py
        apps.py
        models.py
        tests.py
        views.py
```

Database setup

Create a database **employees** in mysql, and configure into the **settings.py** file of django project and modify the ENGINE from sqlite3 to mysql i.e.

```
'ENGINE': 'django.db.backends.mysql',
```

//Settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'employees',
        'USER': 'root',
        'PASSWORD': 'microsoftBt21',
        'HOST':'localhost',
        'PORT':'3306'
    }
}
```

Contacts Table of Employees:

```
DESCRIBE contacts
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
emp_id	int	NO	PRI	NULL	
emp_contact	varchar(20)	YES		NULL	
emp_name	varchar(255)	YES		NULL	
emp_des	varchar(255)	YES		NULL	

Create a Model

Put the following code into **models.py** file.

// models.py

```python
from django.db import models

# Create your models here.
8 usages  Sannya Wasim
class Employees(models.Model):
    emp_id = models.IntegerField(primary_key=True, max_length=10)
    emp_name = models.CharField(max_length=255)
    emp_des = models.CharField(max_length=255)
    emp_contact = models.CharField(max_length=255)
    Sannya Wasim
    class Meta:
        db_table = 'contacts'
```

Create a ModelForm

// forms.py

```python
from django import forms
from employee.models import Employees

5 usages  Sannya Wasim
class EmployeeForm(forms.ModelForm):
    Sannya Wasim
    class Meta:
        model = Employees
        fields = "__all__"
```

CRUD Operations involving Views, Templates and URLs

Views

Django views are Python functions that takes http requests and returns http response, like HTML documents.

A web page that uses Django is full of views with different tasks and missions.

Views are usually put in a file called `views.py` located on your app's folder.

Templates

Django provides a convenient way to generate dynamic HTML pages by using its template system.

A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

URLs

Every page on the Internet needs its own URL. This way your application knows what it should show to a user who opens that URL. In Django, we use something called URLconf (URL configuration). URLconf is a set of patterns that Django will try to match the requested URL to find the correct view.

INSERT Operation

Views.py

```
Usage  new *
def emp(request):
    if request.method == "POST":
        form = EmployeeForm(request.POST)
        if form.is_valid():
            try:
                form.save()
                return redirect('/show')
            except:
                pass
    else:
                form = EmployeeForm()
    return render(request, 'index.html', {'form':form})
```

Index.html

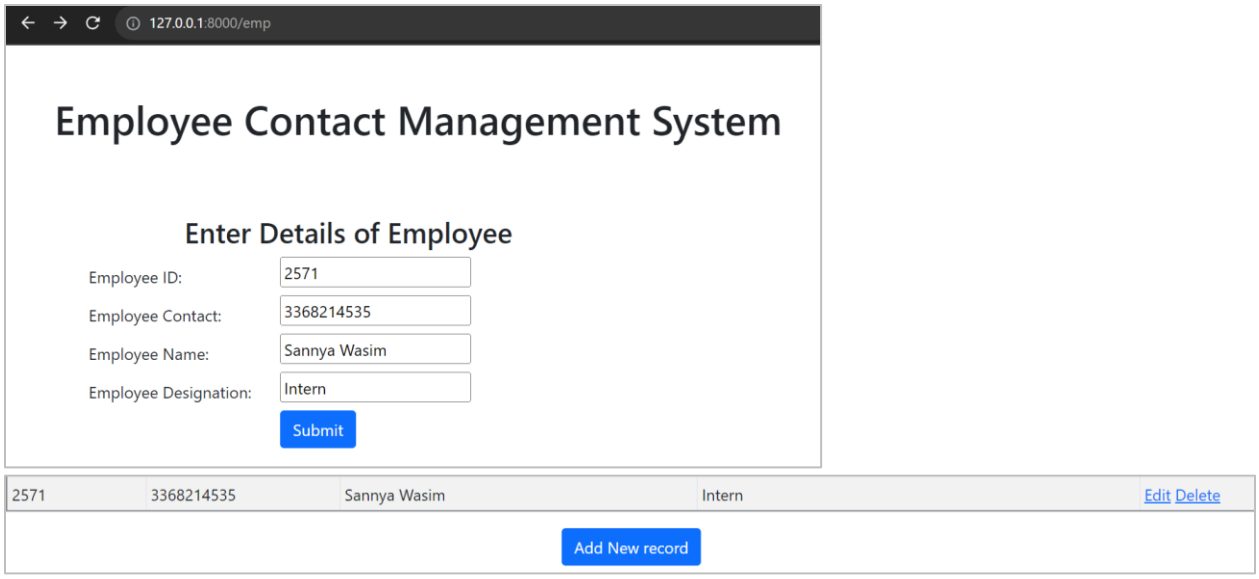
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
<!--    {% load static %}-->
<!--    <link rel="stylesheet" href="{% static 'css/style.css' %}"/>-->
    {% load bootstrap5 %}
  {% bootstrap_css %}
  {% bootstrap_javascript %}
</head>
<body>
<h1 class="m-5">Employee Contact Management System</h1>
    <form method="POST" class="post-form" action="/emp">
        {% csrf_token %}
        <div class="container">
            <br>
            <div class="form-group row">
                <label class="col-sm-1 col-form-label"></label>
                <div class="col-sm-4">
                    <h3>Enter Details of Employee</h3>
                </div>
            </div>
```

```
            <div class="form-group row">
                <label class="col-sm-2 col-form-label">Employee ID: </label>
                <div class="col-sm-4">
                    {{ form.emp_id}}
                </div>
            </div>
            <div class="form-group row">
                <label class="col-sm-2 col-form-label">Employee Contact: </label>
                <div class="col-sm-4">
                    {{ form.emp_contact}}
                </div>
            </div>
            <div class="form-group row">
                <label class="col-sm-2 col-form-label">Employee Name: </label>
                <div class="col-sm-4">
                    {{ form.emp_name}}
                </div>
            </div>
            <div class="form-group row">
                <label class="col-sm-2 col-form-label">Employee Designation: </label>
                <div class="col-sm-4">
                    {{ form.emp_des}}
                </div>
            </div>
            <div class="form-group row">
                <label class="col-sm-2 col-form-label"></label>
                <div class="col-sm-4">
                    <button type="submit" class="btn btn-primary">Submit</button>
                </div>
            </div>
        </div>

    </form>
</body>
</html>
```

Urls.py

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('show', views.show),
    path('emp', views.emp),
```

Output:



Display Operation

Views.py

```python
from django.shortcuts import render, redirect
from employee.forms import EmployeeForm
from employee.models import Employees

# Create your views here.
1 usage  Sannya Wasim
def show(request):
    employees = Employees.objects.all()
    return render(request, 'show.html', {'contacts': employees})
```

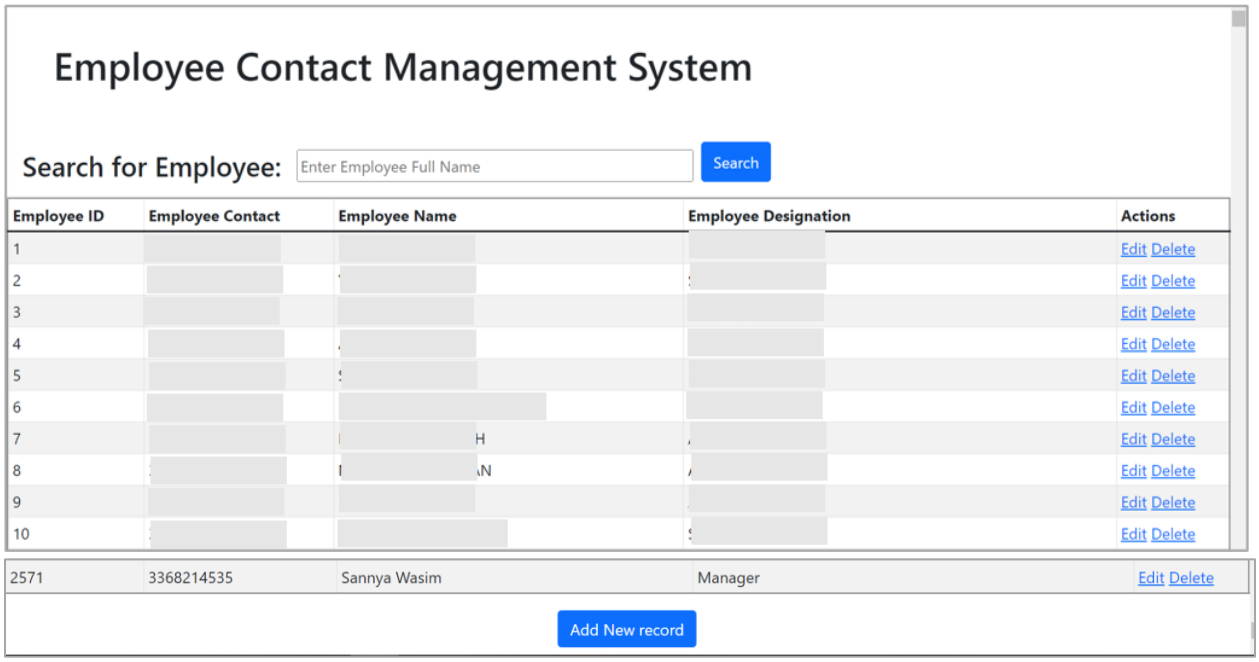
Show.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Employees</title>
    {% load bootstrap5 %}
  {% bootstrap_css %}
  {% bootstrap_javascript %}
</head>
<body>
<h1 class="m-5">Employee Contact Management System</h1>

<form class="d-flex p-2 m-auto" action="{% url 'search' %}">
    <h3 class="m-2">Search for Employee: </h3>
    <input type="text" name="ename" placeholder="Enter Employee Full Name" class="m-2" size="50"/>
    <button type="submit" class="btn btn-primary mb-2">Search</button>
</form>
<table class="table table-striped table-bordered table-sm" border="2">
    <thead>
    <tr>
        <th>Employee ID</th>
        <th>Employee Contact</th>
        <th>Employee Name</th>
        <th>Employee Designation</th>
        <th>Actions</th>
    </tr>
    </thead>
    <tbody>
    {% for employee in contacts %}
    <tr>
        <td>{{employee.emp_id}}</td>
        <td>{{employee.emp_contact}}</td>
        <td>{{employee.emp_name}}</td>
        <td>{{employee.emp_des}}</td>
        <td><a href="/update/{{employee.emp_id}}"><span class="glyphicon glyphicon-pencil">Edit</span>
        <a href="/delete/{{employee.emp_id}}"><span class="glyphicon glyphicon-pencil">Delete</span></
    </tr>
    {% endfor %}
    </tbody>
</table>
<center><a class="btn btn-primary mb-2" href="/emp">Add New record</a></center>
</body>
</html>
```

Urls.py

```python
from django.contrib import admin
from django.urls import path
from employee import views

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

Output:



UPDATE Operation

Views.py

```python
def update(request, id):
    employee = Employees.objects.get(emp_id=id)
    if request.method == 'POST':
        form = EmployeeForm(request.POST, instance=employee)
        if form.is_valid():
            form.save()
            return redirect('/show')
    else:
        form = EmployeeForm(instance=employee)

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

Edit.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Edit Employees</title>
<!--    {% load static %}-->
<!--    <link rel="stylesheet" href="{% static 'css/style.css' %}"/>-->
    {% load bootstrap5 %}
  {% bootstrap_css %}
  {% bootstrap_javascript %}
</head>
<body>
<form method="POST" class="post-form" action="/update/{{employee.emp_id}}">
    {% csrf_token %}
    <div class="container">
        <br>
        <div class="form-group row">
            <label class="col-sm-1 col-form-label"></label>
            <div class="col-sm-4">
                <h3>Update Details</h3>
            </div>
        </div>
        <div class="form-group row">
```

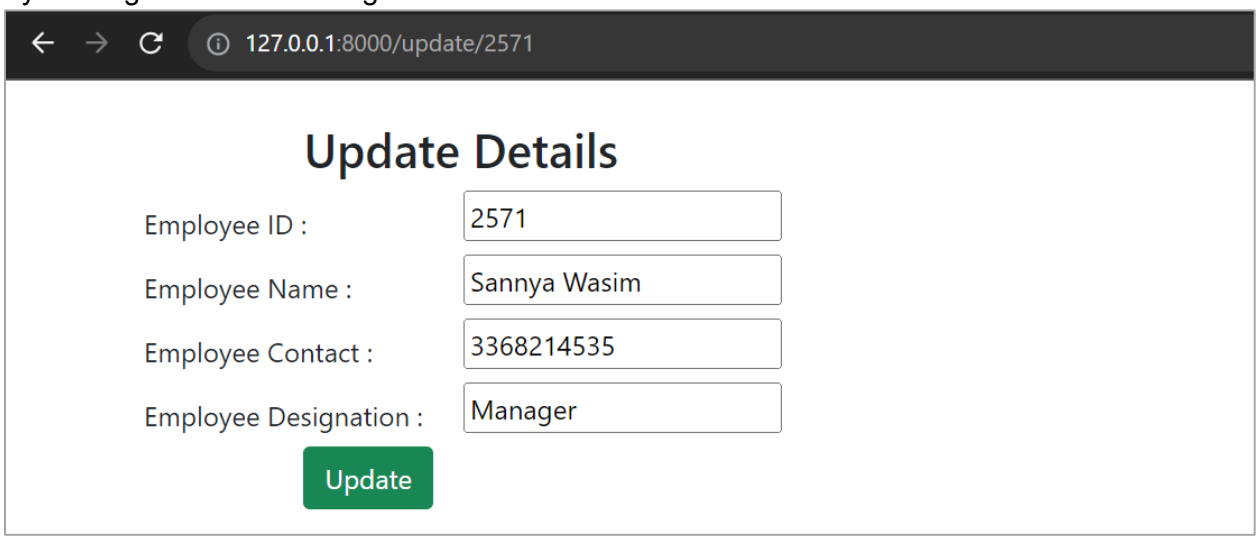
```
            <label class="col-sm-2 col-form-label">Employee ID :</label>
            <div class="col-sm-4">
                <input type="text" name="emp_id" id="id_eid" required maxlength="20" value="{{ employe
            </div>
        </div>
        <div class="form-group row">
            <label class="col-sm-2 col-form-label">Employee Name :</label>
            <div class="col-sm-4">
                <input type="text" name="emp_name" id="id_ename" required maxlength="100" value="{{ em
            </div>
        </div>
        <div class="form-group row">
            <label class="col-sm-2 col-form-label">Employee Contact :</label>
            <div class="col-sm-4">
                <input type="text" name="emp_contact" id="id_econtact" required maxlength="100" value=
            </div>
        </div>
        <div class="form-group row">
            <label class="col-sm-2 col-form-label">Employee Designation :</label>
            <div class="col-sm-4">
                <input type="text" name="emp_des" id="id_edes" required maxlength="100" value="{{ empl
            </div>
            </div>
        </div>
        <div class="form-group row">
            <label class="col-sm-1 col-form-label"></label>
            <div class="col-sm-4">
                <button type="submit" class="btn btn-success">Update</button>
            </div>
        </div>
    </div>
</form>
</body>
</html>
```

Urls.py

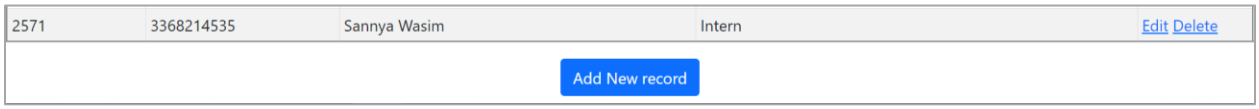
```
from django.contrib import admin
from django.urls import path
from employee import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('show', views.show),
    path('update/<int:id>', views.update),
```

Output:

By clicking on the edit link against each field:



Record Updated where designation set to "**Intern**".



DELETE Operation:

Views.py:

```
def destroy(request, id):
    employee = Employees.objects.get(emp_id = id)
    employee.delete()
    return redirect('/show')
```

Show.html

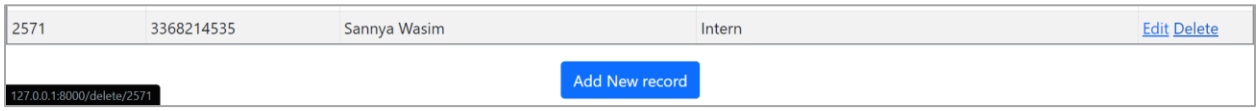
```
    {% for employee in contacts %}
    <tr>
        <td>{{employee.emp_id}}</td>
        <td>{{employee.emp_contact}}</td>
        <td>{{employee.emp_name}}</td>
        <td>{{employee.emp_des}}</td>
        <td><a href="/update/{{employee.emp_id}}"><span class="glyphicon glyphicon-pencil">Edit</span>
        <a href="/delete/{{employee.emp_id}}"><span class="glyphicon glyphicon-pencil">Delete</span></
    </tr>
    {% endfor %}
```

Urls.py

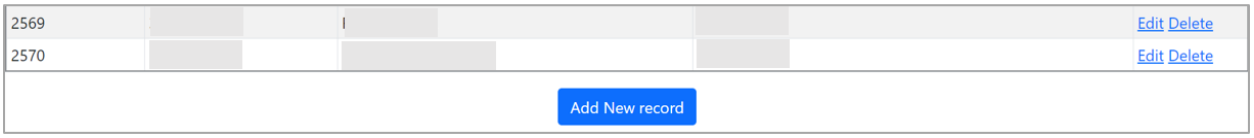
```python
from django.contrib import admin
from django.urls import path
from employee import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('show', views.show),
    path('update/<int:id>', views.update),
    path('delete/<int:id>', views.destroy),
```

Output:

Before deleting:



After deleting:



SEARCH Operation:

Views.py

```python
def search(request):
    ename = request.GET.get('ename')
    if ename:
        employee = Employees.objects.filter(emp_name__icontains=ename)
    else:
        employee = Employees.objects.none()
    return render(request, 'search.html', {'employee': employee})
```

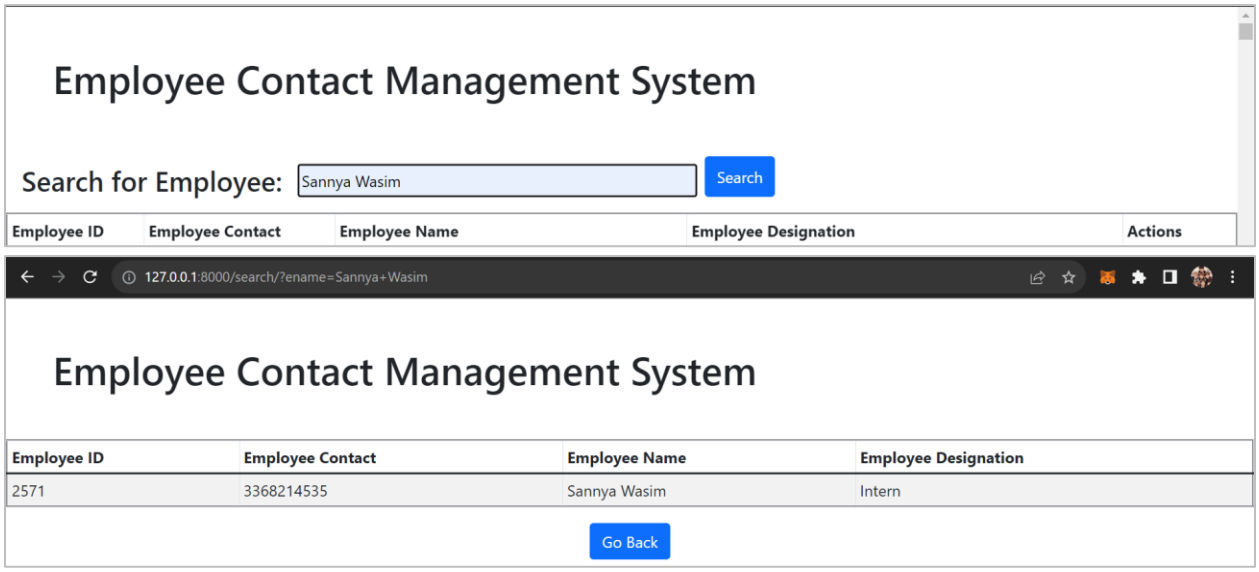
Search.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Employees</title>
    {% load bootstrap5 %}
  {% bootstrap_css %}
  {% bootstrap_javascript %}
</head>
<body>
<h1 class="m-5">Employee Contact Management System</h1>
<table class="table table-striped table-bordered table-sm" border="2">
    <thead>
    <tr>
        <th>Employee ID</th>
        <th>Employee Contact</th>
        <th>Employee Name</th>
        <th>Employee Designation</th>
    </tr>
    </thead>
    <tbody>
    {% for a in employee %}
    <tr>
        <td>{{a.emp_id}}</td>
        <td>{{a.emp_contact}}</td>
        <td>{{a.emp_name}}</td>
        <td>{{a.emp_des}}</td>
    </tr>
    {% endfor %}
    </tbody>
</table>
<center><a class="btn btn-primary mb-2" href="/show">Go Back</a></center>
</body>
</html>
```

Urls.py

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('show', views.show),
    path('emp', views.emp),
    path('update/<int:id>', views.update),
    path('delete/<int:id>', views.destroy),
    path('search/', views.search, name='search'),
]
```

Output:



Errors Encountered and their solutions

Error:

It is impossible to add a non-nullable field 'member_contact' to member without specifying a default. This is because the database needs something to populate existing rows. Please select a fix: 1) Provide a one-off default now (will be set on all existing rows with a null value for this column) 2) Quit and manually define a default value in models.py.

Solution:

The error message you're encountering indicates that you're trying to add a non-nullable field (member_contact) to the Member model without specifying a default value. Since the field is non-nullable and you have existing rows in the database, Django needs to know how to populate the existing rows with a value for the new field.

You can add a default parameter to the member_contact field in your Member model:

```
from django.db import models

class Member(models.Model):
    member_name = models.CharField(max_length=255)
    member_des = models.CharField(max_length=255, default='')  # Specify an
    member_contact = models.IntegerField(default=11111111111)
```

Error:

```
def update(request, id):
    employee = Employees.objects.get(emp_id=id)
    form = EmployeeForm(request.POST, instance=employee)
    if form.is_valid():
        form.save()
        # return render(request, "show.html", {'contacts':employee})
        return redirect('/show')
```

Solution:

The error you're encountering is because the update view in your views.py does not have a return statement for all possible code paths. In Django, views are expected to return an HttpResponse object, but in your update view, you're not returning anything explicitly in the case where the form is not valid.

To fix this issue, you should ensure that your update view returns an appropriate response for both valid and invalid form submissions. Here's how you can do it:

```
def update(request, id):
    employee = Employees.objects.get(emp_id=id)
    if request.method == 'POST':
        form = EmployeeForm(request.POST, instance=employee)
        if form.is_valid():
            form.save()
            return redirect('/show')
    else:
        form = EmployeeForm(instance=employee)

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

Error:

so basically when using django, I made a model of Employees:

```
class Employees(models.Model):
        emp_id = models.CharField(max_length=10)
        emp_name = models.CharField(max_length=255)
        emp_des = models.CharField(max_length=255)
        emp_contact = models.CharField(max_length=255)
class Meta:
```

db_table = 'contacts'

and when I perform migration, it creates 0001_initial.py:

Generated by Django 4.2.4 on 2023-08-30 08:25

from django.db import migrations, models

class Migration(migrations.Migration):

initial = True dependencies = [ ]

operations = [migrations.CreateModel(name='Employees', fields=[# ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')), ('emp_id', models.CharField(max_length=10)), ('emp_name', models.CharField(max_length=255)), ('emp_des', models.CharField(max_length=255)), ('emp_contact', models.CharField(max_length=255)),], options={ 'db_table': 'contacts', },),]

My 'contacts' table in database was already created with columns as **emp_id', 'emp_name', 'emp_des', emp_contact where emp_id** was not set to Auto Increment attribute. I want that when I use this exisiting table, it doesnt produce this: **('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),** because since id field is not in the table, it is still finding it.

Error: OperationalError: (1054, "Unknown column 'contacts.id' in 'field list'")

Solution:

It seems like you want to use an existing table named "contacts" and map it to a Django model named "Employees" without creating a new primary key field (such as the default 'id' field that Django creates). You can achieve this by explicitly specifying the primary key field in your model to match the existing table structure.

Here's how you can modify your model to achieve this:

```
from django.db import models

class Employees(models.Model):
    emp_id = models.IntegerField(primary_key=True)  # Use IntegerField for
    emp_name = models.CharField(max_length=255)
    emp_des = models.CharField(max_length=255)
    emp_contact = models.CharField(max_length=255)

    class Meta:
        db_table = 'contacts'
```

In this modified model:

I've added primary_key=True to the emp_id field, which indicates that this field will serve as the primary key for the model. Since your existing table doesn't have the default 'id' field, this will ensure that Django uses the emp_id as the primary key.

I've removed the id field that was automatically generated by Django, as it's not present in your existing table.

With this configuration, when you perform migrations, Django will recognize that you've specified the emp_id field as the primary key and won't try to create the default 'id' field.

Remember that if you make changes to your model like this, you should create a new migration to update the existing schema. You can do this with:

```
python manage.py makemigrations employee
```

Then, apply the migration with:

```
python manage.py migrate employee
```