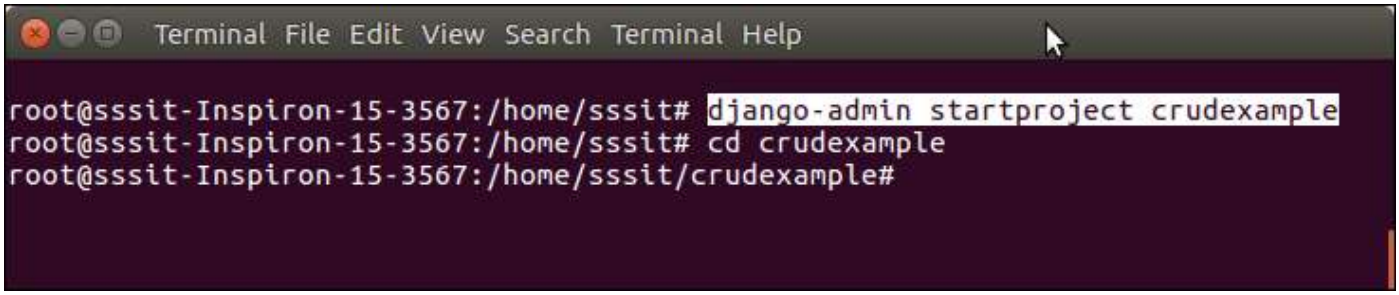
Django CRUD (Create Read Update Delete) Example

To create a Django application that performs CRUD operations, follow the following steps.

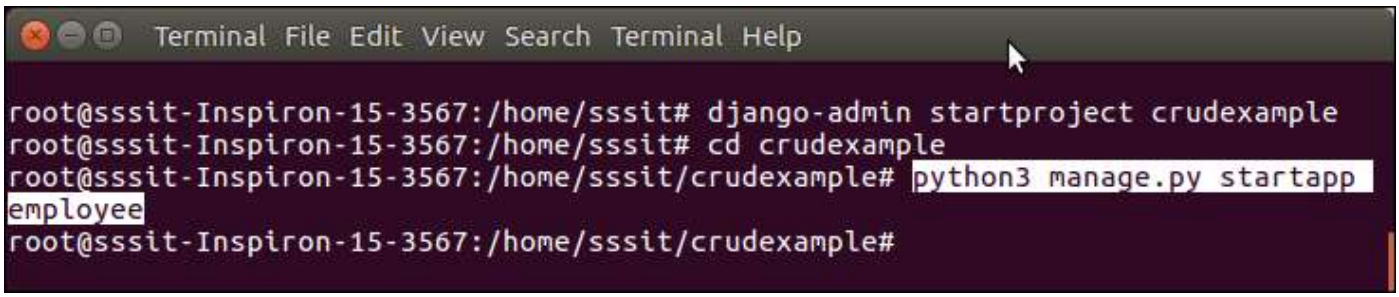
1. Create a Project

```
$ django-admin startproject crudexample
```



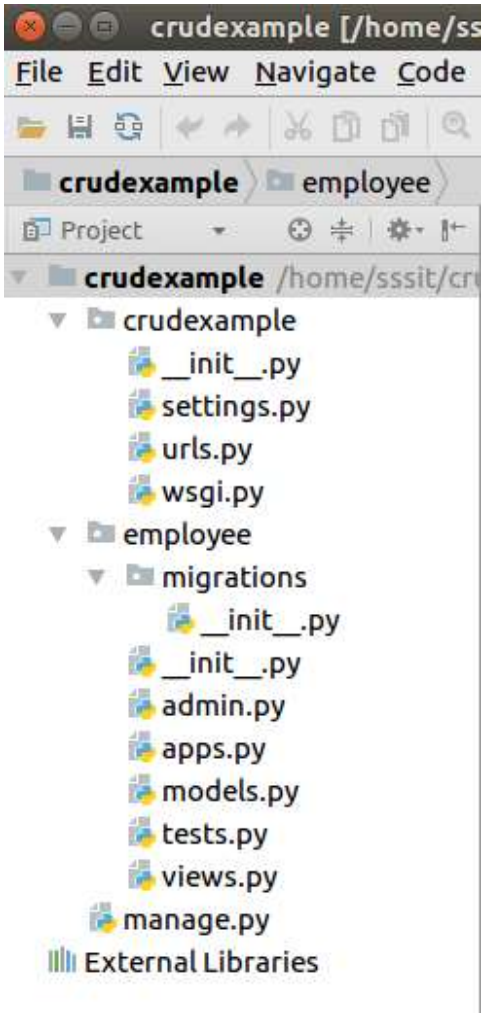
2. Create an App

```
$ python3 manage.py startapp employee
```



3. Project Structure

Initially, our project looks like this:



4. Database Setup

Create a database **djangodb** in mysql, and configure into the **settings.py** file of django project. See the example.

// settings.py

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

5. Create a Model

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

// models.py

```
from django.db import models
class Employee(models.Model):
    eid = models.CharField(max_length=20)
    ename = models.CharField(max_length=100)
    eemail = models.EmailField()
    econtact = models.CharField(max_length=15)
    class Meta:
```

```
    db_table = "employee"
```

6. Create a ModelForm

// forms.py

```
from django import forms
from employee.models import Employee
class EmployeeForm(forms.ModelForm):
  class Meta:
    model = Employee
    fields = "__all__"
```

7. Create View Functions

// views.py

```
from django.shortcuts import render, redirect
from employee.forms import EmployeeForm
from employee.models import Employee
# Create your views here.
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})
def show(request):
```

```
    employees = Employee.objects.all()
    return render(request,"show.html",{'employees':employees})
def edit(request, id):
    employee = Employee.objects.get(id=id)
    return render(request,'edit.html', {'employee':employee})
def update(request, id):
    employee = Employee.objects.get(id=id)
    form = EmployeeForm(request.POST, instance = employee)
    if form.is_valid():
        form.save()
        return redirect("/show")
    return render(request, 'edit.html', {'employee': employee})
def destroy(request, id):
    employee = Employee.objects.get(id=id)
    employee.delete()
    return redirect("/show")
```

8. Provide Routing

Provide URL patterns to map with views function.

// urls.py

```
from django.contrib import admin
```

```
from django.urls import path
from employee import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('emp', views.emp),
    path('show',views.show),
    path('edit/<int:id>', views.edit),
    path('update/<int:id>', views.update),
    path('delete/<int:id>', views.destroy),
]
```

9. Organize Templates

Create a **templates** folder inside the **employee** app and create three (index, edit, show) html files inside the directory. The code for each is given below.

// index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Index</title>
    {% load staticfiles %}
    <link rel="stylesheet" href="{% static 'css/style.css' %}"/>
```

```
</head>
<body>
<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</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.eid }}
   </div>
  </div>
  <div class="form-group row">
   <label class="col-sm-2 col-form-label">Employee Name:</label>
   <div class="col-sm-4">
    {{ form.ename }}
   </div>
  </div>
   <div class="form-group row">
   <label class="col-sm-2 col-form-label">Employee Email:</label>
   <div class="col-sm-4">
    {{ form.eemail }}
   </div>
  </div>
   <div class="form-group row">
   <label class="col-sm-2 col-form-label">Employee Contact:</label>
   <div class="col-sm-4">
    {{ form.econtact }}
   </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-primary">Submit</button>
    </div>
  </div>
    </div>
</form>
</body>
</html>
```

// show.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Employee Records</title>
    {% load staticfiles %}
    <link rel="stylesheet" href="{% static 'css/style.css' %}"/>
</head>
<body>
<table class="table table-striped table-bordered table-sm">
    <thead class="thead-dark">
    <tr>
        <th>Employee ID</th>
        <th>Employee Name</th>
        <th>Employee Email</th>
        <th>Employee Contact</th>
        <th>Actions</th>
    </tr>
    </thead>
    <tbody>
```

```
{% for employee in employees %}
   <tr>
      <td>{{ employee.eid }}</td>
      <td>{{ employee.ename }}</td>
      <td>{{ employee.eemail }}</td>
      <td>{{ employee.econtact }}</td>
      <td>
          <a href="/edit/{{ employee.id }}"><span class="glyphicon glyphicon-pencil" >Edit</span>
</a>
         <a href="/delete/{{ employee.id }}">Delete</a>
      </td>
   </tr>
{% endfor %}
   </tbody>
</table>
<br>
<br>
<center><a href="/emp" class="btn btn-primary">Add New Record</a></center>
</body>
</html>
```

// edit.html

```
<!DOCTYPE html>
<html lang="en">
```

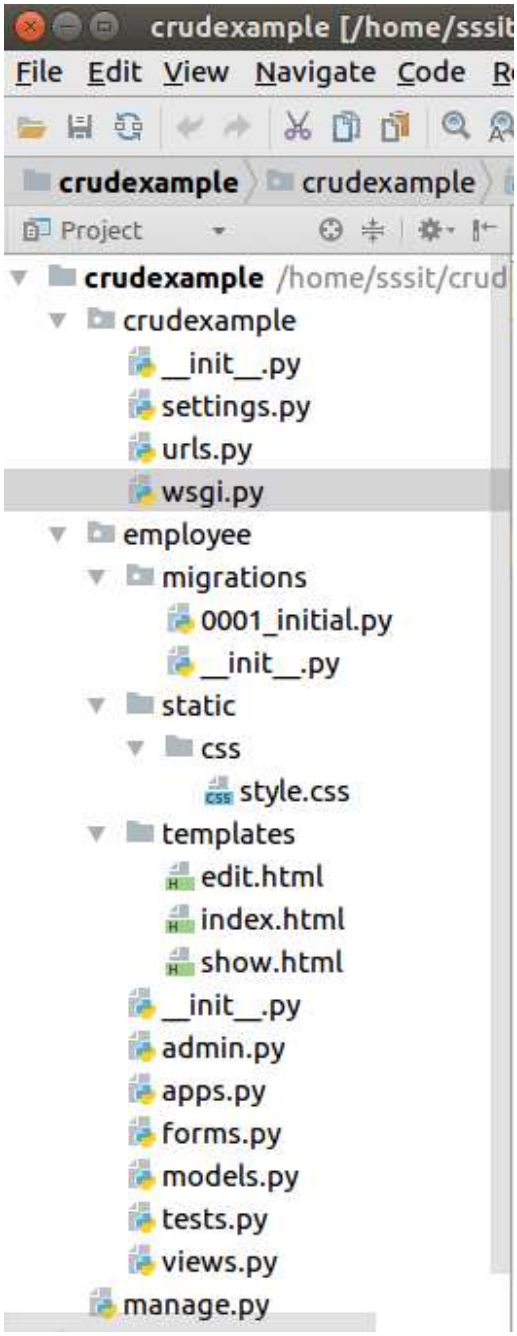
```
<head>
   <meta charset="UTF-8">
   <title>Index</title>
   {% load staticfiles %}
   <link rel="stylesheet" href="{% static 'css/style.css' %}"/>
</head>
<body>
<form method="POST" class="post-form" action="/update/{{employee.id}}">
      {% csrf_token %}
   <div class="container">
<br>
   <div 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="eid" id="id_eid" required maxlength="20" value="
{{ employee.eid }}"/>
   </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="ename" id="id_ename" required maxlength="100" value="
{{ employee.ename }}" />
   </div>
  </div>
   <div class="form-group row">
   <label class="col-sm-2 col-form-label">Employee Email:</label>
   <div class="col-sm-4">
```

```
            <input type="email" name="eemail" id="id_eemail" required maxlength="254" value="
{{ employee.eemail }}" />
    </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="econtact" id="id_econtact" required maxlength="15" value="
{{ employee.econtact }}" />
    </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>
```

10. Static Files Handling

Create a folder **static/css** inside the **employee** app and put a css inside it. Download the css file here Click Here.

11. Project Structure



12. Create Migrations

Create migrations for the created model employee, use the following command.

```
$ python3 manage.py makemigrations
```

```
Terminal File Edit View Search Terminal Help
root@sssit-Inspiron-15-3567:/home/sssit/crudexample# python3 manage.py makemigra
tions
Migrations for 'employee':
  employee/migrations/0001_initial.py
    - Create model Employee
root@sssit-Inspiron-15-3567:/home/sssit/crudexample#
```

After migrations, execute one more command to reflect the migration into the database. But before it, mention name of app (employee) in INSTALLED_APPS of settings.py file.

// settings.py

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

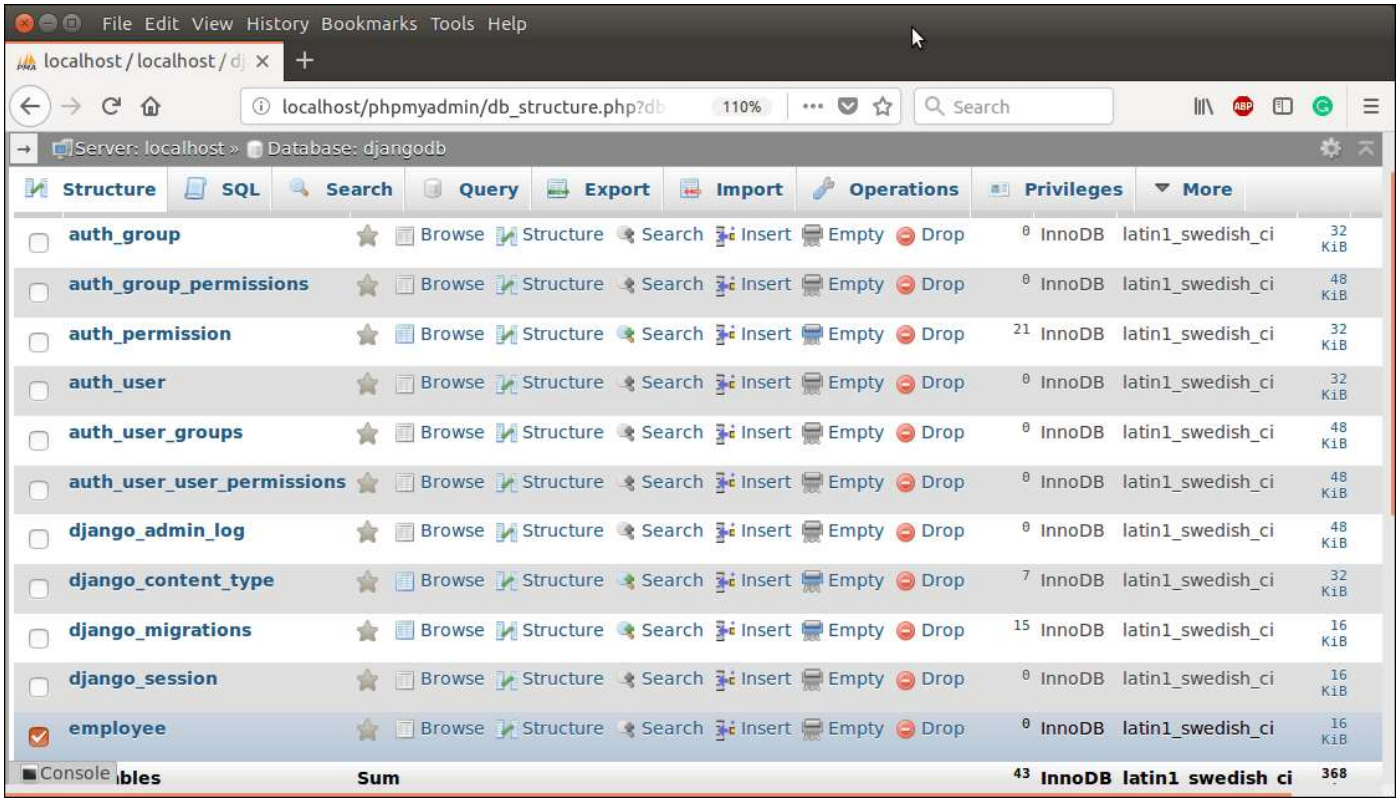
Run the command to migrate the migrations.

```
$ python3 manage.py migrate
```

```
Terminal File Edit View Search Terminal Help
root@sssit-Inspiron-15-3567:/home/sssit/crudexample# python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, employee, 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 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 employee.0001_initial... OK
  Applying sessions.0001_initial... OK
root@sssit-Inspiron-15-3567:/home/sssit/crudexample#
```

Now, our application has successfully connected and created tables in database. It creates 10 default tables for handling project (session, authentication etc) and one table of our model that we created.

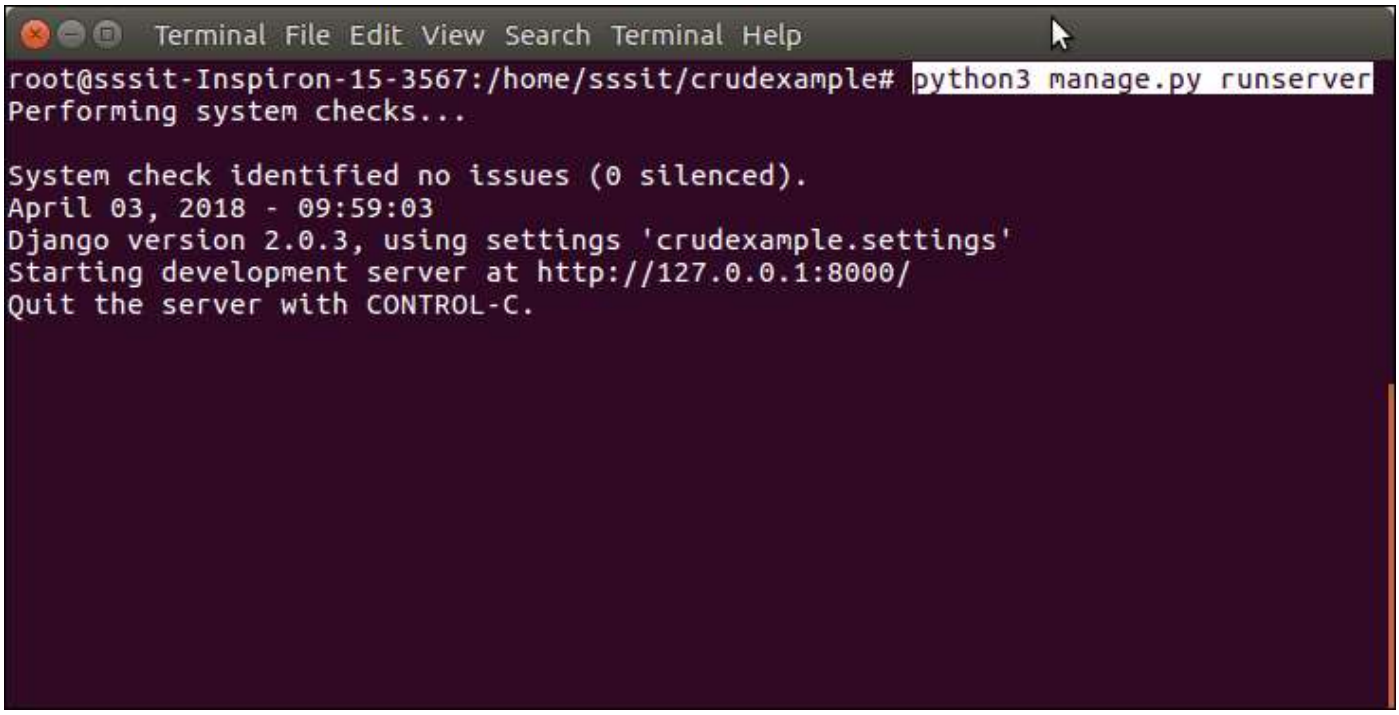
See list of tables created after migrate command.



Run Server

To run server use the following command.

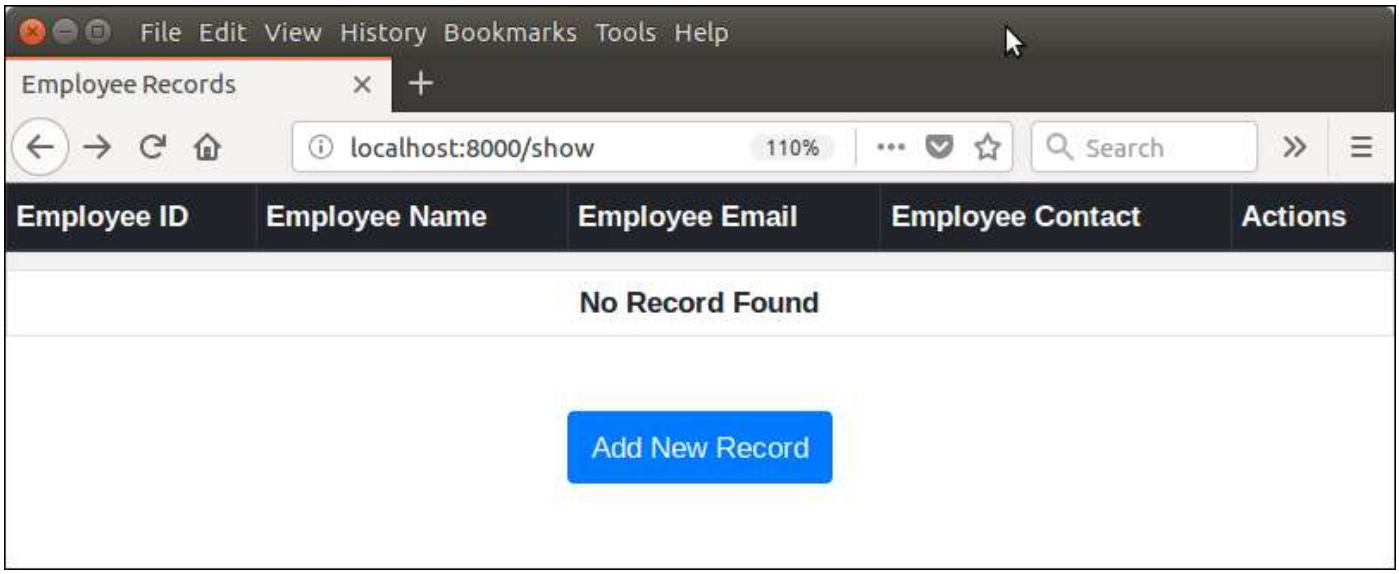
```
$ python3 manage.py runserver
```



Access to the Browser

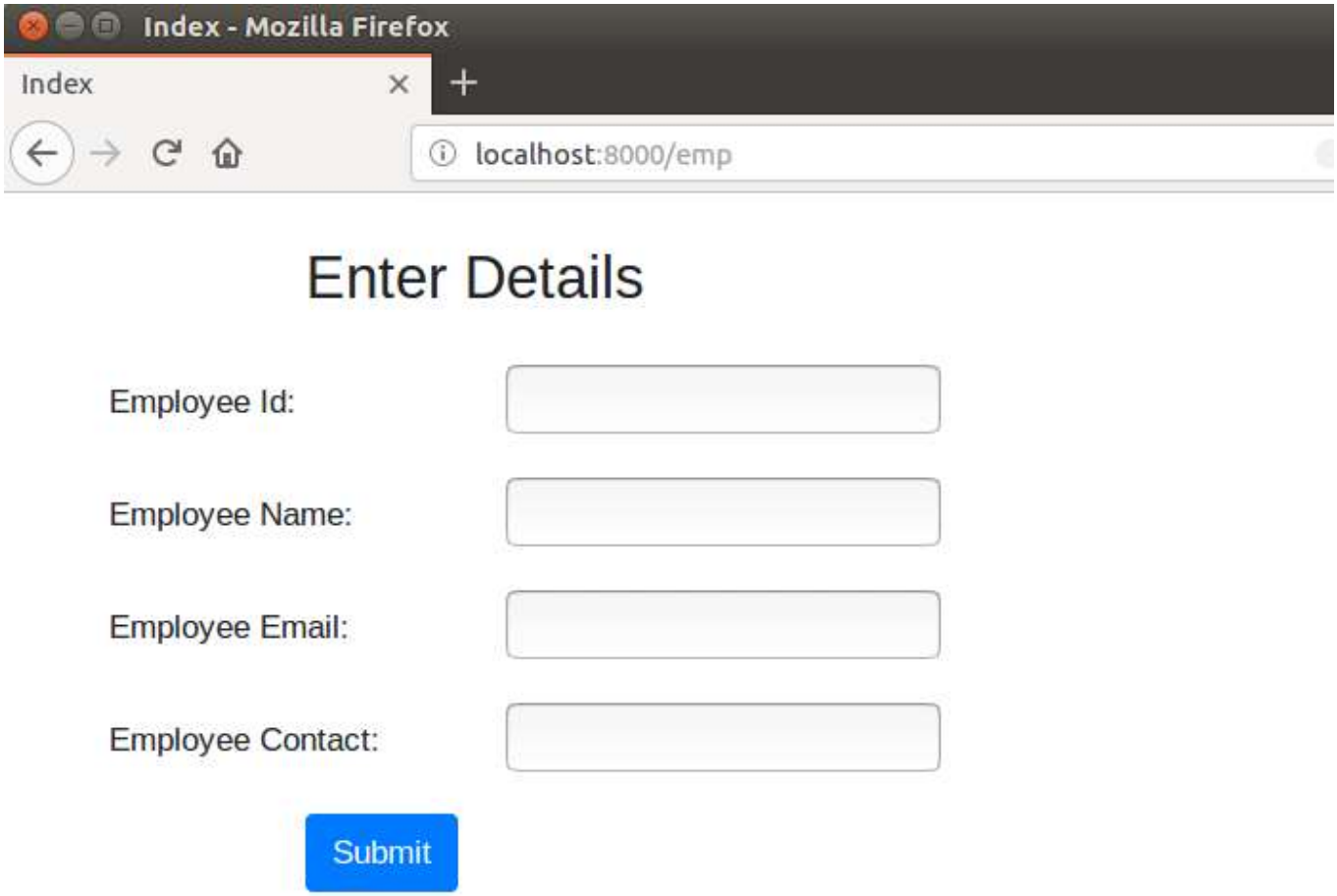
Access the application by entering **localhost:8000/show**, it will show all the available employee records.

Initially, there is no record. So, it shows no record message.

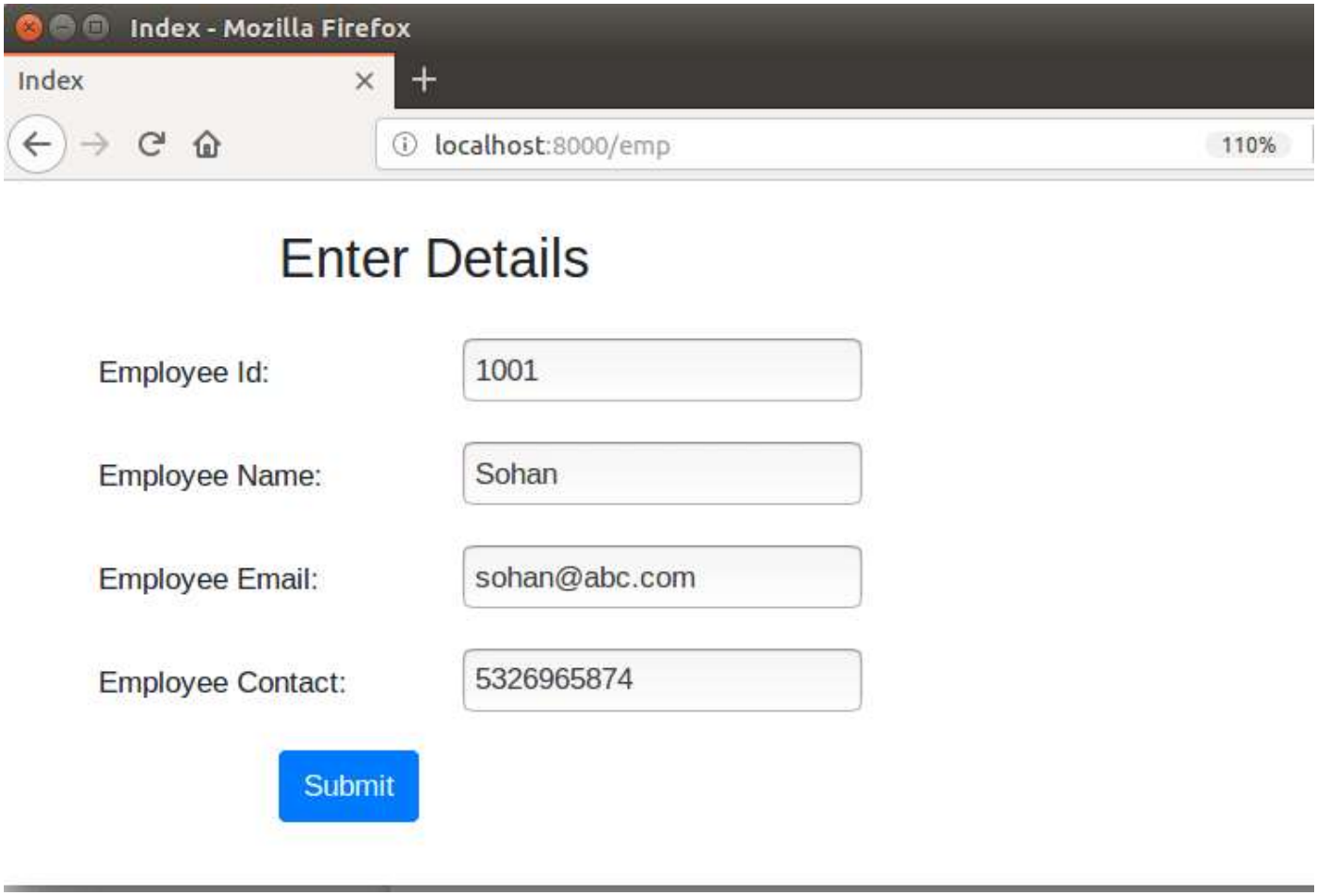


Adding Record

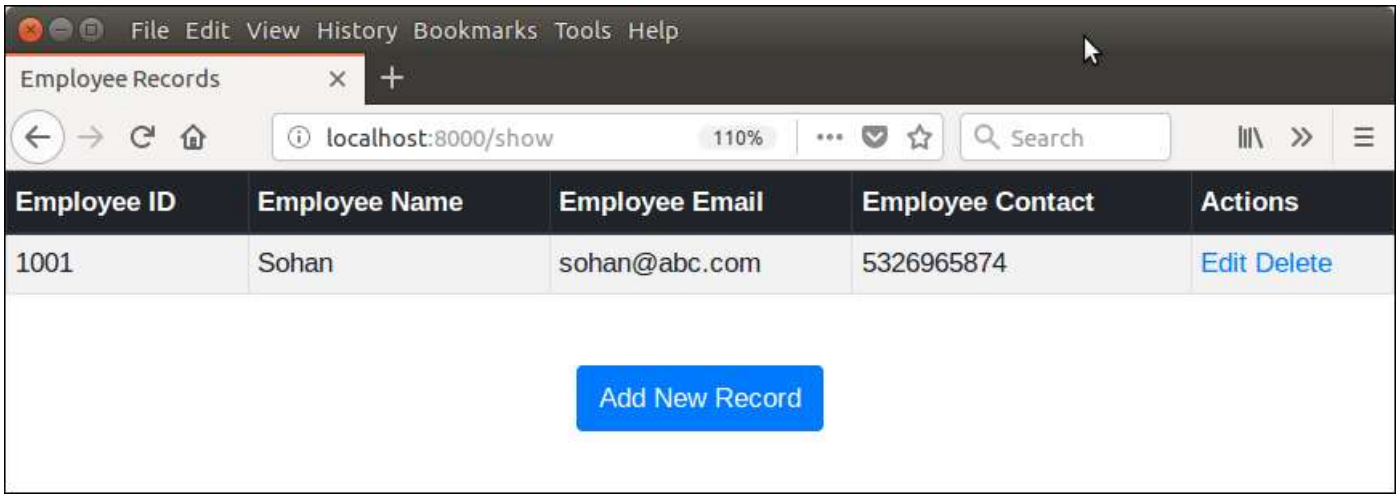
Click on the **Add New Record** button and fill the details. See the example.



Filling the details.

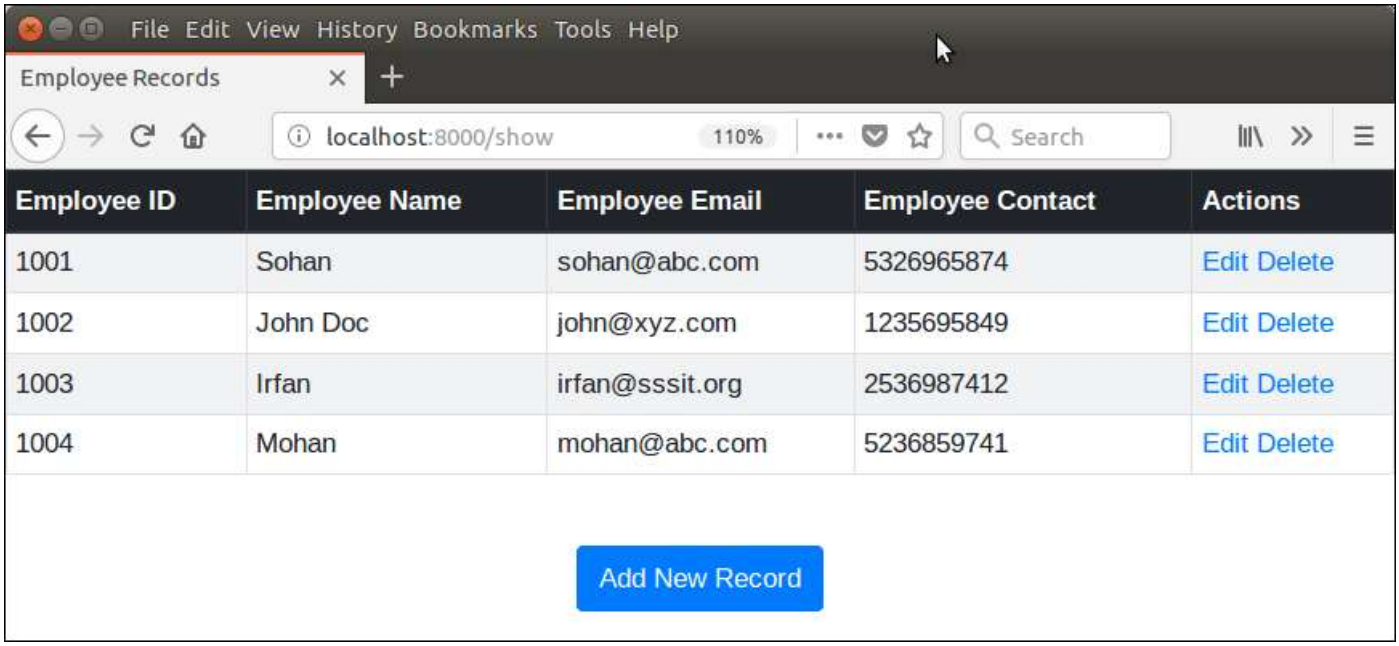


Submit the record and see, after submitting it shows the saved record.



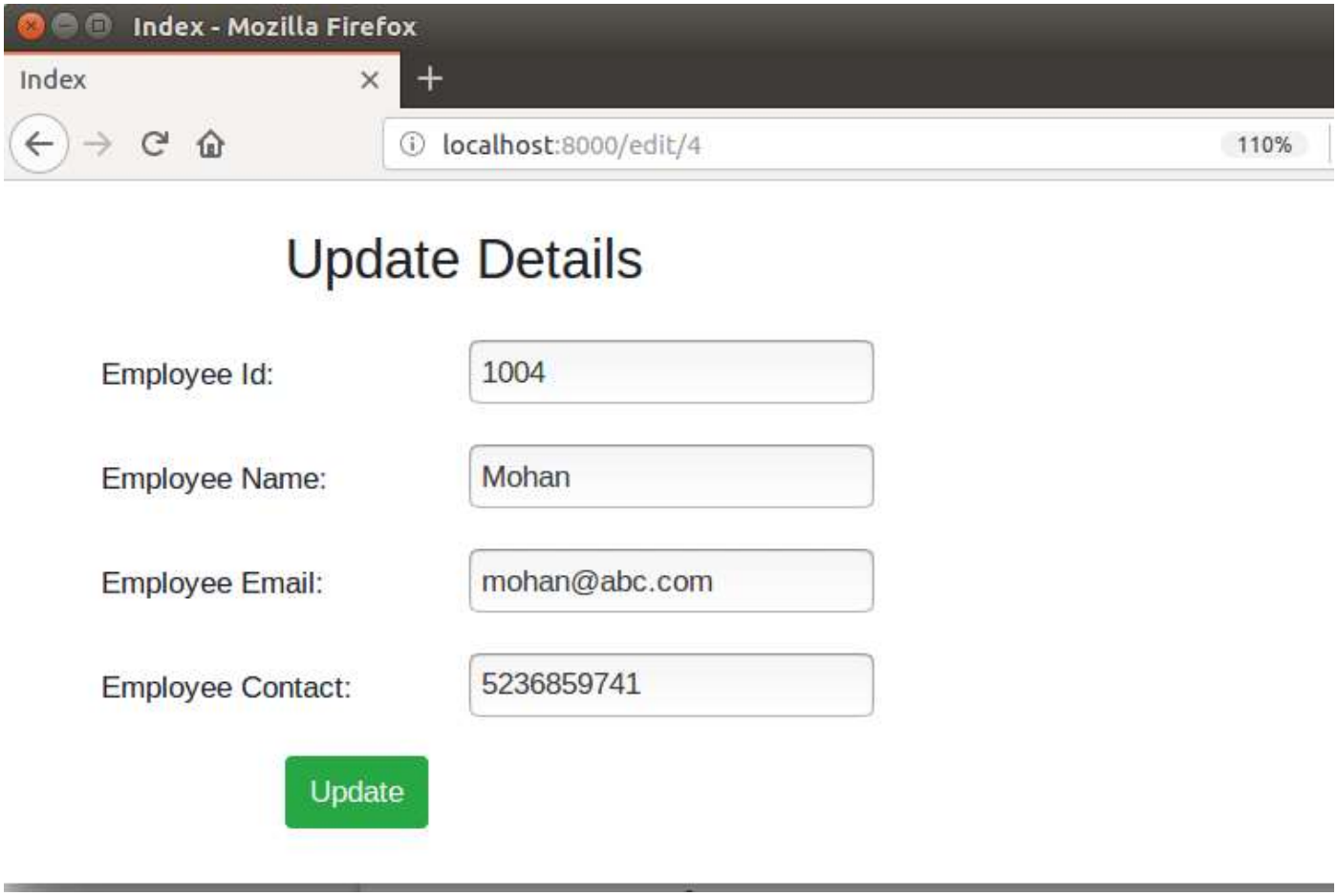
This section also allows, update and delete records from the **actions** column.

After saving couple of records, now we have following records.

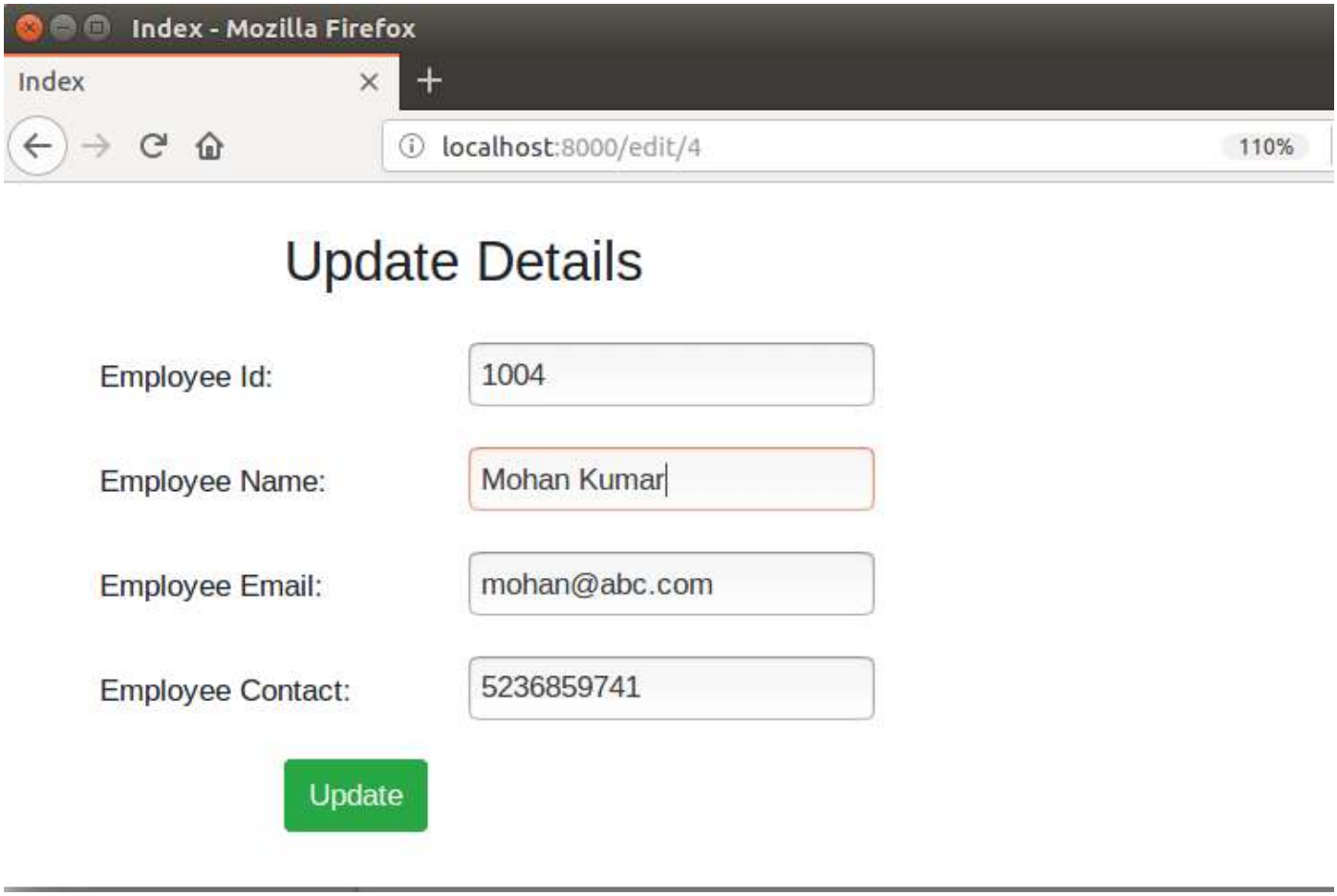


Update Record

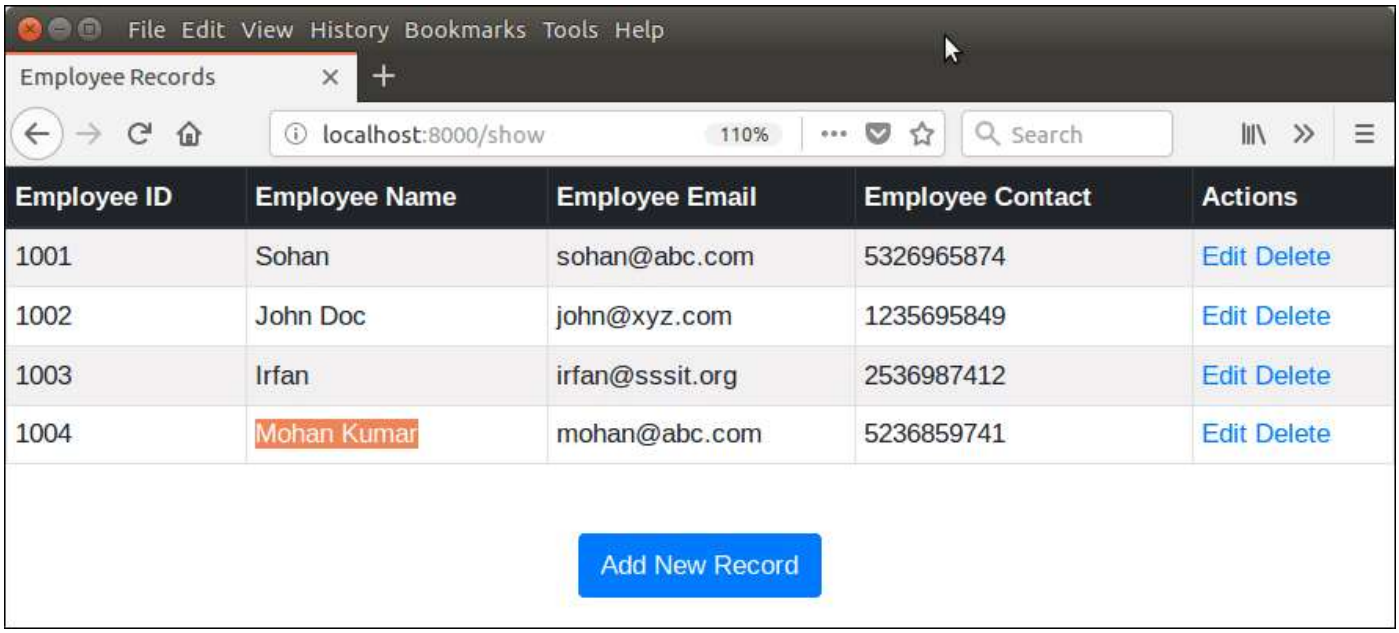
Lets update the record of **Mohan** by clicking on **edit** button. It will display record of Mohan in edit mode.



Lets, suppose I update **mohan** to **mohan kumar** then click on the update button. It updates the record immediately. See the example.



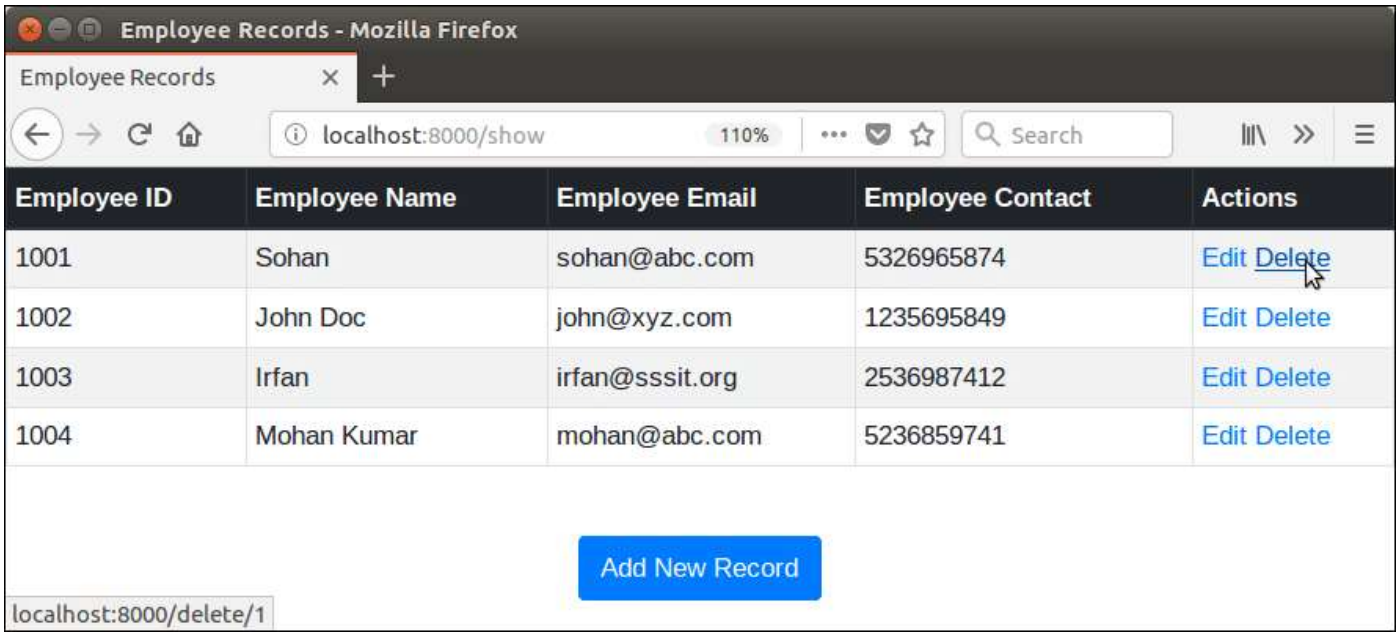
Click on update button and it redirects to the following page. See name is updated.



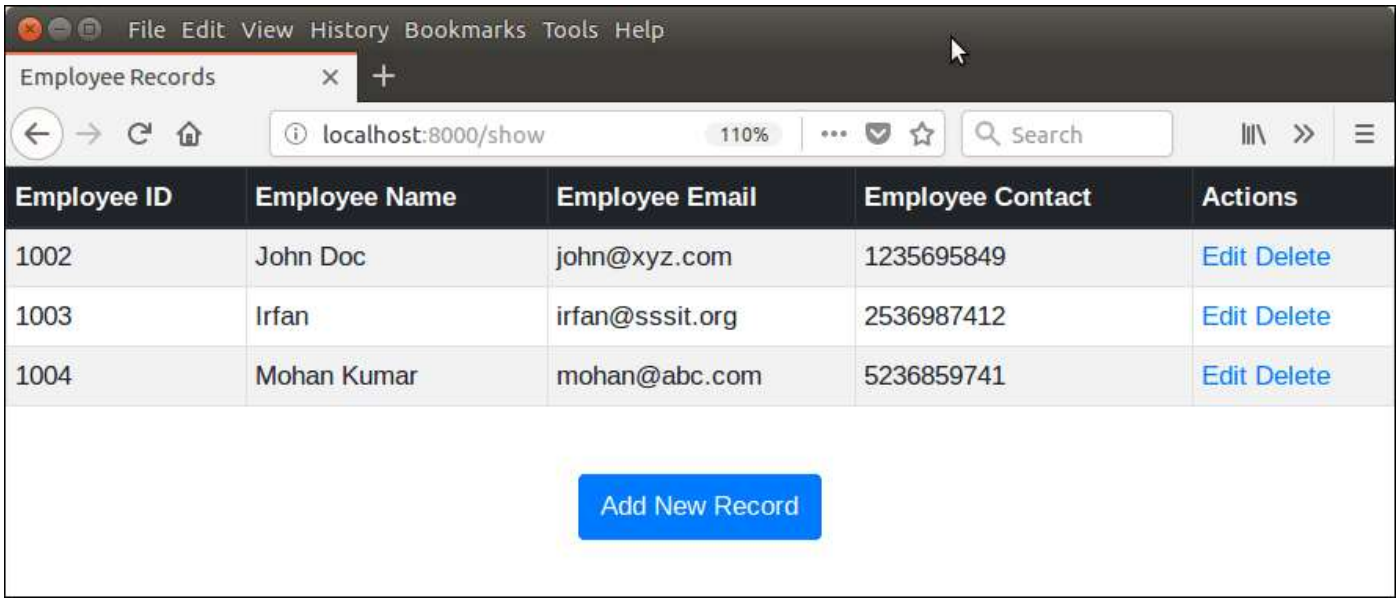
Same like, we can delete records too, by clicking the **delete** link.

Delete Record

Suppose, I want to delete **Sohan**, it can be done easily by clicking the delete button. See the example.



After deleting, we left with the following records.



Well, we have successfully created a CRUD application using Django.

← Prev Next →

B.Tech / MCA