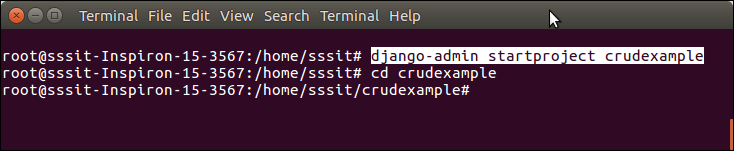
Django CRUD (Create Read Update Delete) Example

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

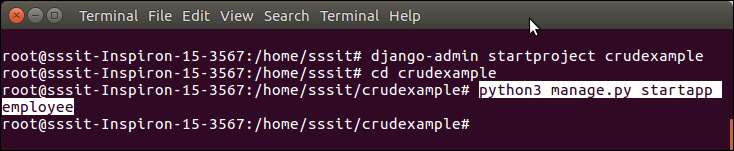
**1. Create a Project**

1. $ django-admin startproject crudexample



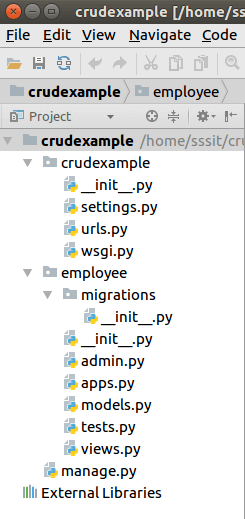
**2. Create an App**

1. $ 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**

1. DATABASES = {
2. 'default': {
3. 'ENGINE': 'django.db.backends.mysql',
4. 'NAME': 'djangodb',
5. 'USER':'root',
6. 'PASSWORD':'mysql',
7. 'HOST':'localhost',
8. 'PORT':'3306'
9. }
10. }

**5. Create a Model**

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

**// models.py**

1. from django.db **import** models
2. **class** Employee(models.Model):
3. eid = models.CharField(max\_length=20)
4. ename = models.CharField(max\_length=100)
5. eemail = models.EmailField()
6. econtact = models.CharField(max\_length=15)
7. **class** Meta:
8. db\_table = "employee"

**6. Create a ModelForm**

**// forms.py**

1. from django **import** forms
2. from employee.models **import** Employee
3. **class** EmployeeForm(forms.ModelForm):
4. **class** Meta:
5. model = Employee
6. fields = "\_\_all\_\_"

**7. Create View Functions**

**// views.py**

1. from django.shortcuts **import** render, redirect
2. from employee.forms **import** EmployeeForm
3. from employee.models **import** Employee
4. # Create your views here.
5. def emp(request):
6. **if** request.method == "POST":
7. form = EmployeeForm(request.POST)
8. **if** form.is\_valid():
9. **try**:
10. form.save()
11. **return** redirect('/show')
12. except:
13. pass
14. **else**:
15. form = EmployeeForm()
16. **return** render(request,'index.html',{'form':form})
17. def show(request):
18. employees = Employee.objects.all()
19. **return** render(request,"show.html",{'employees':employees})
20. def edit(request, id):
21. employee = Employee.objects.get(id=id)
22. **return** render(request,'edit.html', {'employee':employee})
23. def update(request, id):
24. employee = Employee.objects.get(id=id)
25. form = EmployeeForm(request.POST, instance = employee)
26. **if** form.is\_valid():
27. form.save()
28. **return** redirect("/show")
29. **return** render(request, 'edit.html', {'employee': employee})
30. def destroy(request, id):
31. employee = Employee.objects.get(id=id)
32. employee.delete()
33. **return** redirect("/show")

**8. Provide Routing**

Provide URL patterns to map with views function.

**// urls.py**

1. from django.contrib **import** admin
2. from django.urls **import** path
3. from employee **import** views
4. urlpatterns = [
5. path('admin/', admin.site.urls),
6. path('emp', views.emp),
7. path('show',views.show),
8. path('edit/<int:id>', views.edit),
9. path('update/<int:id>', views.update),
10. path('delete/<int:id>', views.destroy),
11. ]

**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**

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <title>Index</title>
6. {% load staticfiles %}
7. <link rel="stylesheet" href="{% static 'css/style.css' %}"/>
8. </head>
9. <body>
10. <form method="POST" **class**="post-form" action="/emp">
11. {% csrf\_token %}
12. <div **class**="container">
13. <br>
14. <div **class**="form-group row">
15. <label **class**="col-sm-1 col-form-label"></label>
16. <div **class**="col-sm-4">
17. <h3>Enter Details</h3>
18. </div>
19. </div>
20. <div **class**="form-group row">
21. <label **class**="col-sm-2 col-form-label">Employee Id:</label>
22. <div **class**="col-sm-4">
23. {{ form.eid }}
24. </div>
25. </div>
26. <div **class**="form-group row">
27. <label **class**="col-sm-2 col-form-label">Employee Name:</label>
28. <div **class**="col-sm-4">
29. {{ form.ename }}
30. </div>
31. </div>
32. <div **class**="form-group row">
33. <label **class**="col-sm-2 col-form-label">Employee Email:</label>
34. <div **class**="col-sm-4">
35. {{ form.eemail }}
36. </div>
37. </div>
38. <div **class**="form-group row">
39. <label **class**="col-sm-2 col-form-label">Employee Contact:</label>
40. <div **class**="col-sm-4">
41. {{ form.econtact }}
42. </div>
43. </div>
44. <div **class**="form-group row">
45. <label **class**="col-sm-1 col-form-label"></label>
46. <div **class**="col-sm-4">
47. <button type="submit" **class**="btn btn-primary">Submit</button>
48. </div>
49. </div>
50. </div>
51. </form>
52. </body>
53. </html>

**// show.html**

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <title>Employee Records</title>
6. {% load staticfiles %}
7. <link rel="stylesheet" href="{% static 'css/style.css' %}"/>
8. </head>
9. <body>
10. <table **class**="table table-striped table-bordered table-sm">
11. <thead **class**="thead-dark">
12. <tr>
13. <th>Employee ID</th>
14. <th>Employee Name</th>
15. <th>Employee Email</th>
16. <th>Employee Contact</th>
17. <th>Actions</th>
18. </tr>
19. </thead>
20. <tbody>
21. {% **for** employee in employees %}
22. <tr>
23. <td>{{ employee.eid }}</td>
24. <td>{{ employee.ename }}</td>
25. <td>{{ employee.eemail }}</td>
26. <td>{{ employee.econtact }}</td>
27. <td>
28. <a href="/edit/{{ employee.id }}"><span **class**="glyphicon glyphicon-pencil" >Edit</span></a>
29. <a href="/delete/{{ employee.id }}">Delete</a>
30. </td>
31. </tr>
32. {% endfor %}
33. </tbody>
34. </table>
35. <br>
36. <br>
37. <center><a href="/emp" **class**="btn btn-primary">Add New Record</a></center>
38. </body>
39. </html>

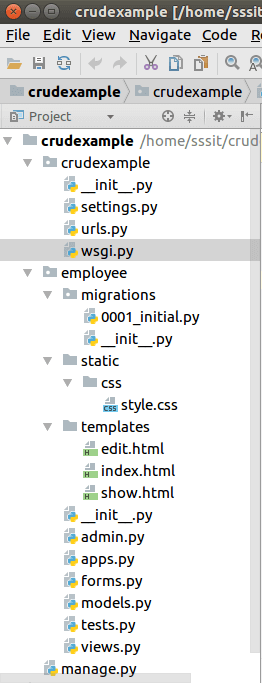
**// edit.html**

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <title>Index</title>
6. {% load staticfiles %}
7. <link rel="stylesheet" href="{% static 'css/style.css' %}"/>
8. </head>
9. <body>
10. <form method="POST" **class**="post-form" action="/update/{{employee.id}}">
11. {% csrf\_token %}
12. <div **class**="container">
13. <br>
14. <div **class**="form-group row">
15. <label **class**="col-sm-1 col-form-label"></label>
16. <div **class**="col-sm-4">
17. <h3>Update Details</h3>
18. </div>
19. </div>
20. <div **class**="form-group row">
21. <label **class**="col-sm-2 col-form-label">Employee Id:</label>
22. <div **class**="col-sm-4">
23. <input type="text" name="eid" id="id\_eid" required maxlength="20" value="{{ employee.eid }}"/>
24. </div>
25. </div>
26. <div **class**="form-group row">
27. <label **class**="col-sm-2 col-form-label">Employee Name:</label>
28. <div **class**="col-sm-4">
29. <input type="text" name="ename" id="id\_ename" required maxlength="100" value="{{ employee.ename }}" />
30. </div>
31. </div>
32. <div **class**="form-group row">
33. <label **class**="col-sm-2 col-form-label">Employee Email:</label>
34. <div **class**="col-sm-4">
35. <input type="email" name="eemail" id="id\_eemail" required maxlength="254" value="{{ employee.eemail }}" />
36. </div>
37. </div>
38. <div **class**="form-group row">
39. <label **class**="col-sm-2 col-form-label">Employee Contact:</label>
40. <div **class**="col-sm-4">
41. <input type="text" name="econtact" id="id\_econtact" required maxlength="15" value="{{ employee.econtact }}" />
42. </div>
43. </div>
44. <div **class**="form-group row">
45. <label **class**="col-sm-1 col-form-label"></label>
46. <div **class**="col-sm-4">
47. <button type="submit" **class**="btn btn-success">Update</button>
48. </div>
49. </div>
50. </div>
51. </form>
52. </body>
53. </html>

**10. Static Files Handling**

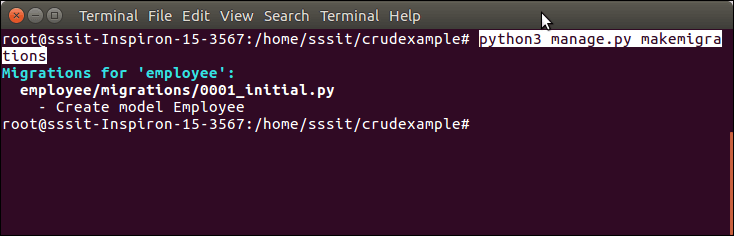
Create a folder **static/css** inside the **employee** app and put a css inside it. Download the css file here [Click Here.](https://www.javatpoint.com/style.css)

**11. Project Structure**



**12. Create Migrations**

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

1. $ python3 manage.py makemigrations

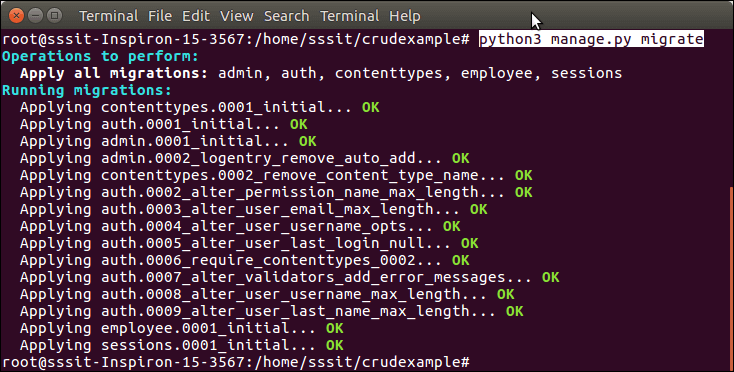
After migrations, execute one more command to reflect the migration into the database. But before it, mention name of app (employee) in INSTALLED\_APPS of settings.py file.

**// settings.py**

1. INSTALLED\_APPS = [
2. 'django.contrib.admin',
3. 'django.contrib.auth',
4. 'django.contrib.contenttypes',
5. 'django.contrib.sessions',
6. 'django.contrib.messages',
7. 'django.contrib.staticfiles',
8. 'employee'
9. ]

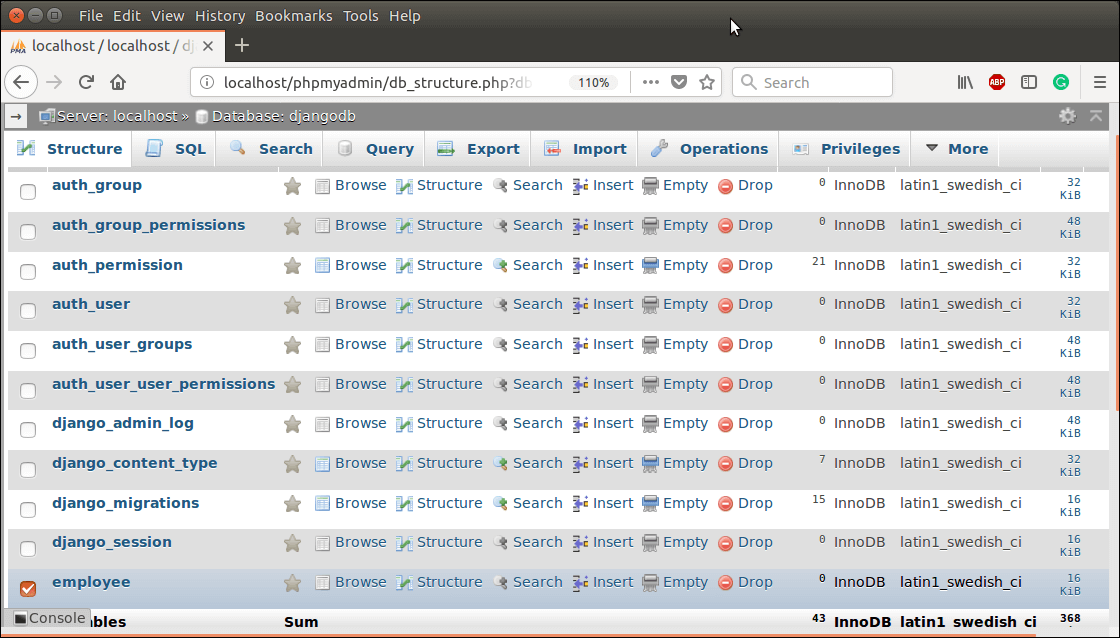
Run the command to migrate the migrations.

1. $ python3 manage.py migrate

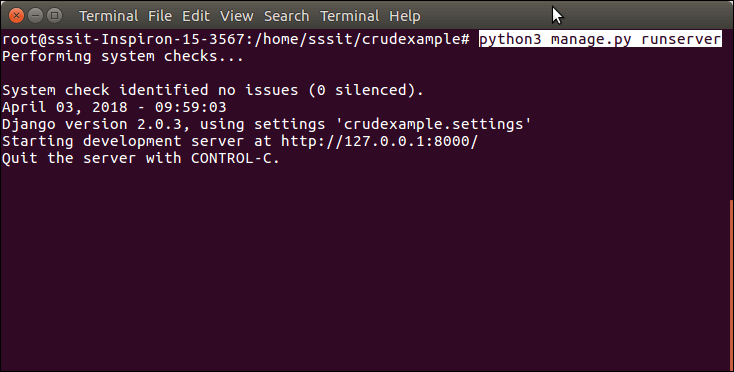


Now, our application has successfully connected and created tables in database. It creates 10 default tables for handling project (session, authentication etc) and one table of our model that we created.

See list of tables created after migrate command.

Run Server

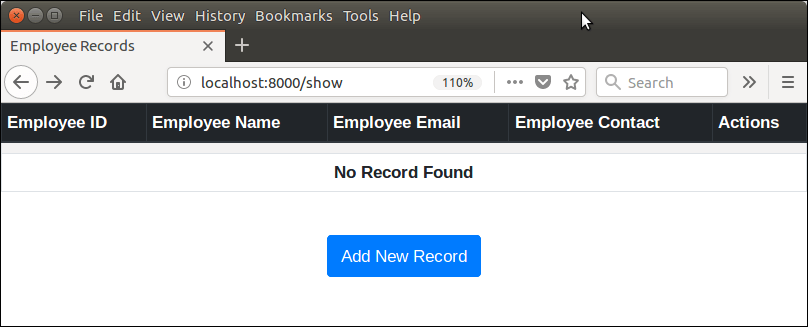
To run server use the following command.

1. $ python3 manage.py runserver

Access to the Browser

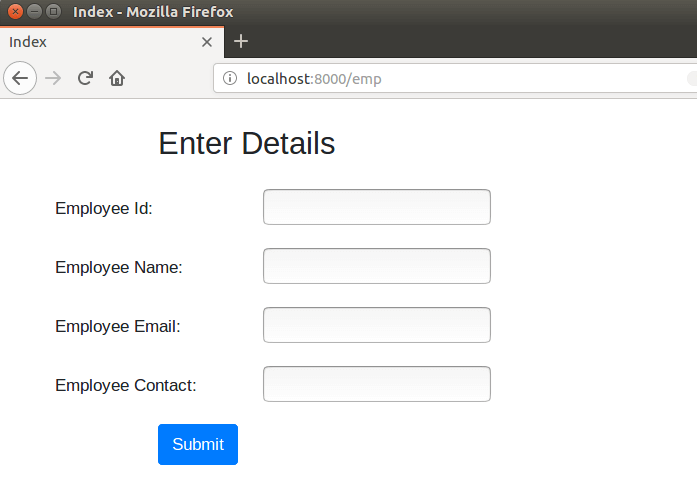
Access the application by entering **localhost:8000/show**, it will show all the available employee records.

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

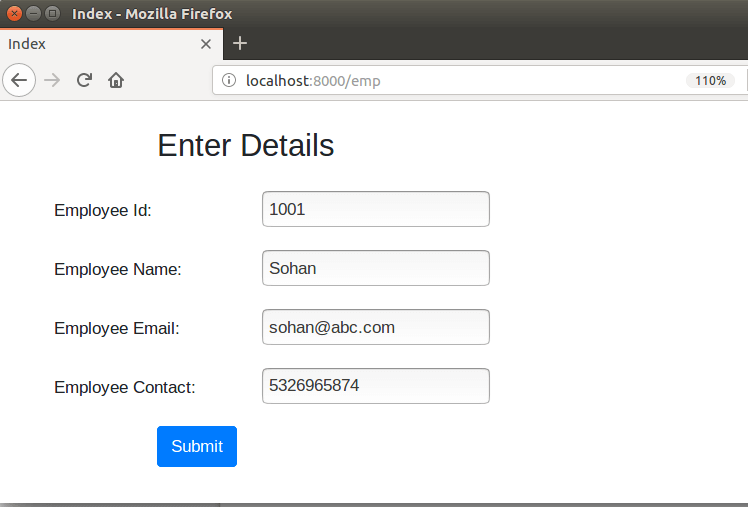


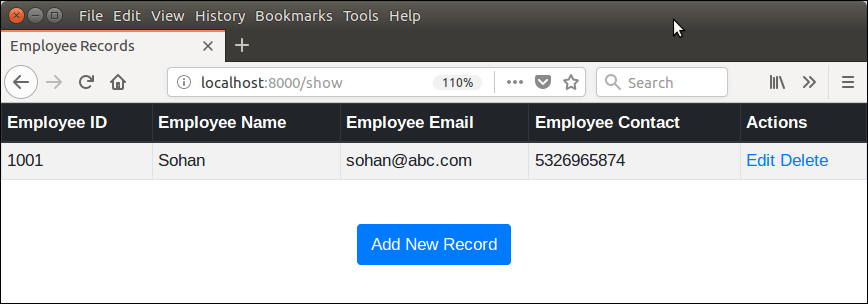
Adding Record

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

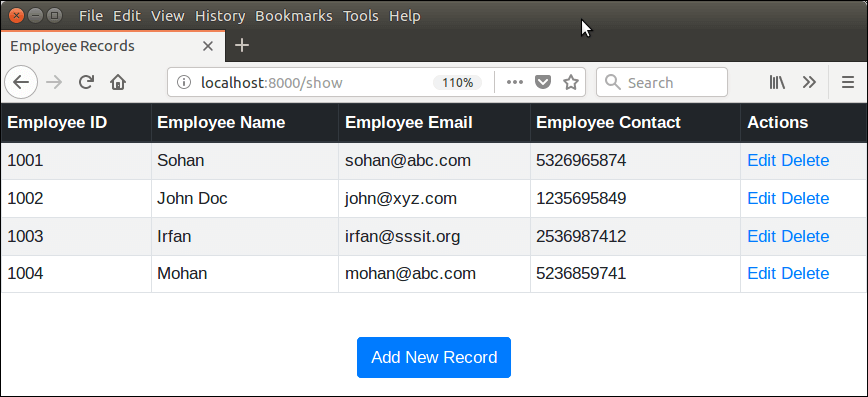


Filling the details.



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

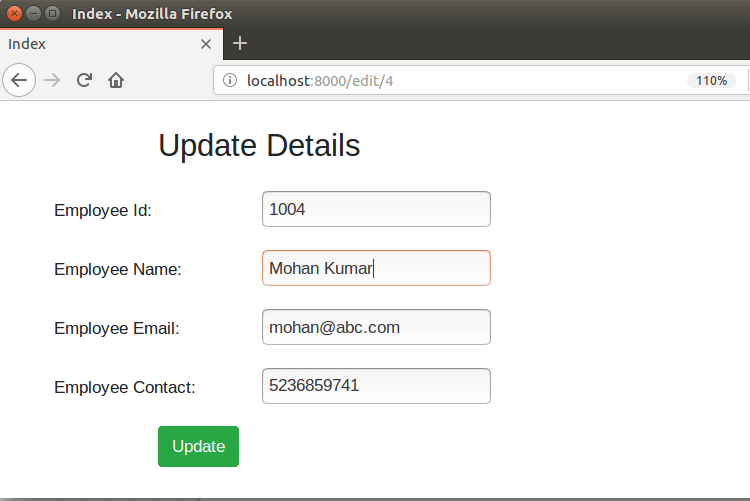
This section also allows, update and delete records from the **actions** column.

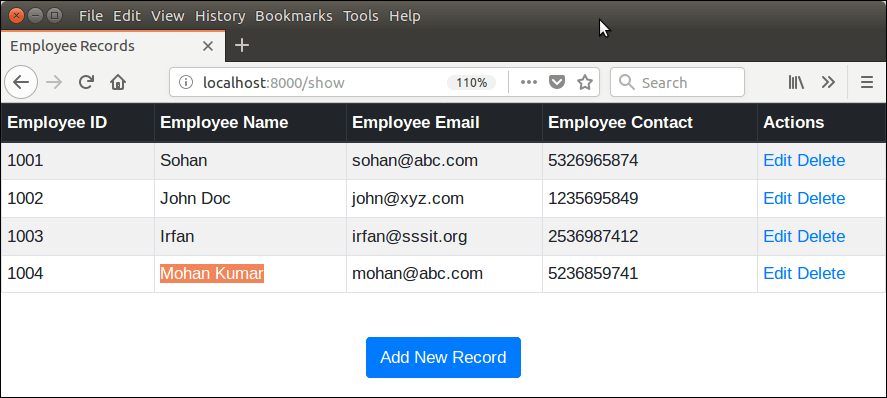
After saving couple of records, now we have following records.

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.

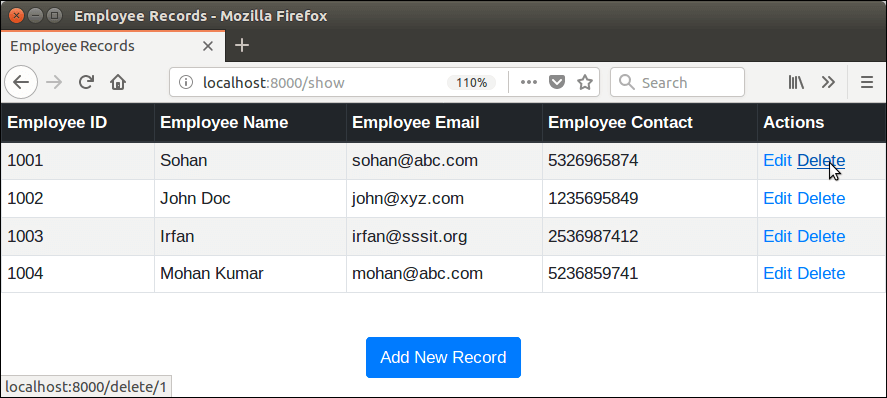


Click on update button and it redirects to the following page. See name is updated.

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