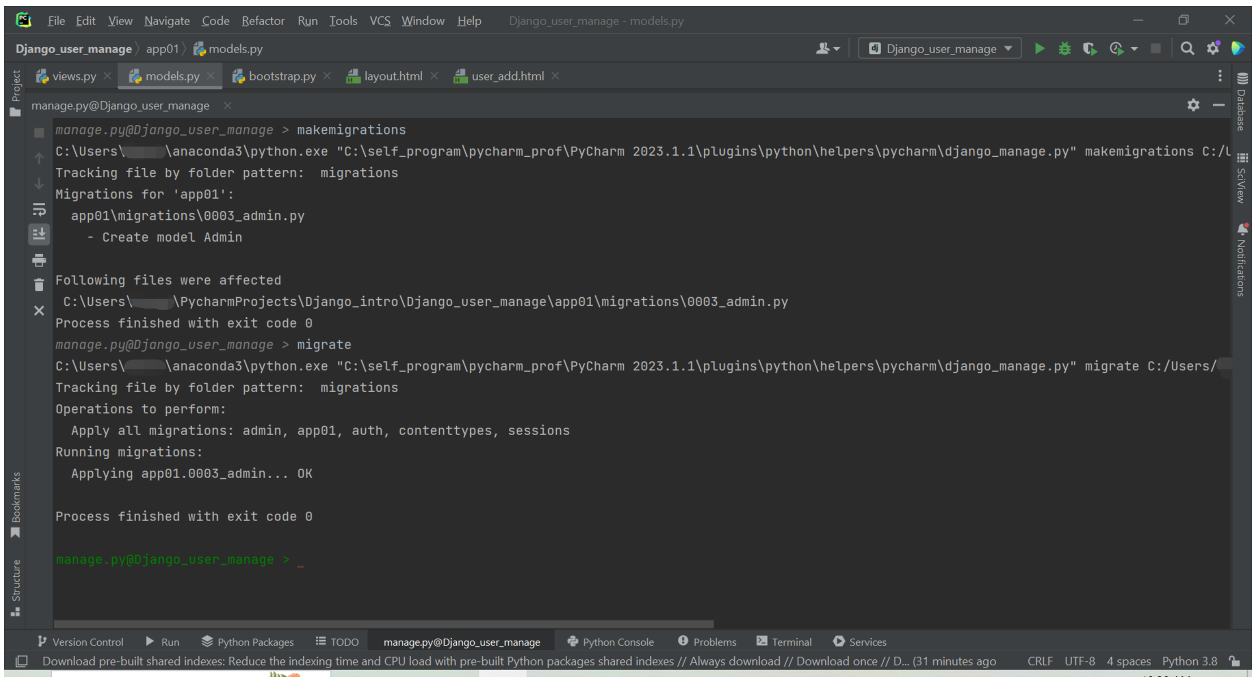
User Management - continue - Administrator

1. create table

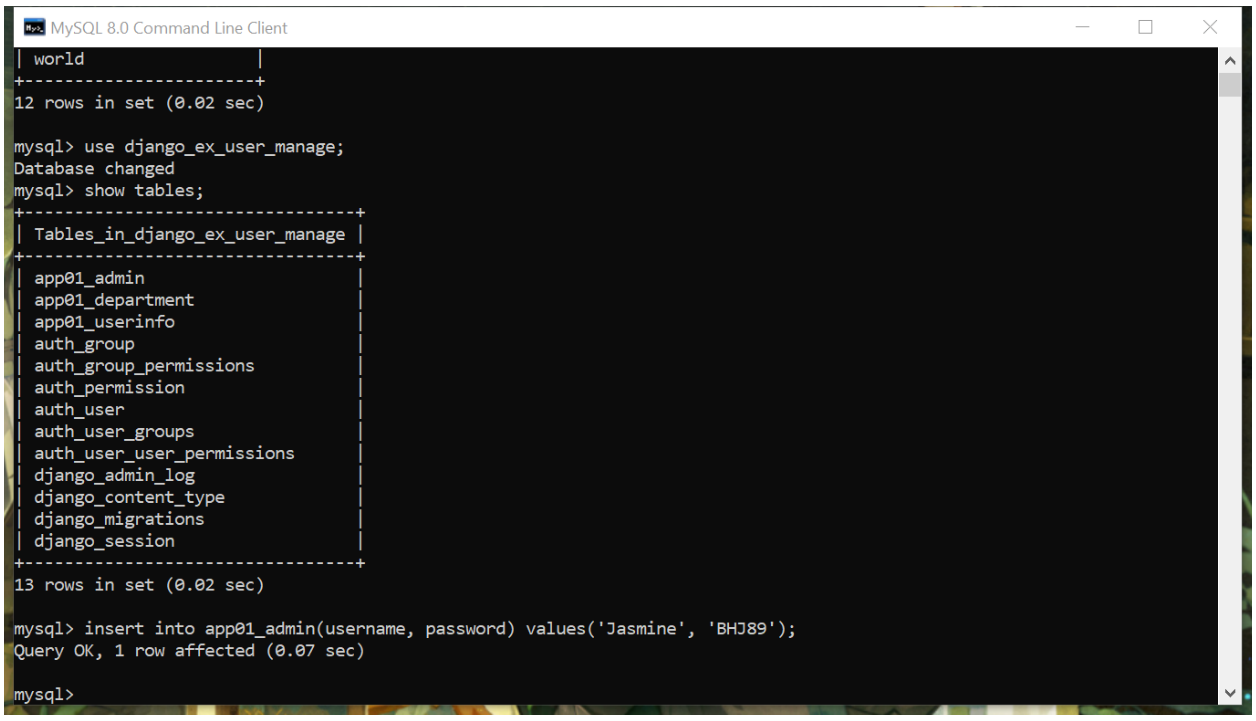
```
# models.py
class Admin(models.Model):
    username = models.CharField(verbose_name='Admin_name', max_length=32)
    password = models.CharField(verbose_name='Password', max_length=64)
```



File Edit View Navigate Code Refactor Run Tools VCS Window Help Django_user_manage - models.py
Django_user_manage 〉 app01 〉 models.py
Django_user_manage
views.py
models.py
bootstrap.py
layout.html
user_add.html
manage.py@Django_user_manage
Version Control
Run
Python Packages
TODO
manage.py@Django_user_manage
Python Console
Problems
Terminal
Services

```
manage.py@Django_user_manage > makemigrations
C:\Users\[redacted]\anaconda3\python.exe "C:\self_program\pycharm_prof\PyCharm 2023.1.1\plugins\python\helpers\pycharm\django_manage.py" makemigrations C:/U
Tracking file by folder pattern:  migrations
Migrations for 'app01':
  app01\migrations\0003_admin.py
    - Create model Admin

Following files were affected
 C:\Users\[redacted]\PycharmProjects\Django_intro\Django_user_manage\app01\migrations\0003_admin.py
Process finished with exit code 0
manage.py@Django_user_manage > migrate
C:\Users\[redacted]\anaconda3\python.exe "C:\self_program\pycharm_prof\PyCharm 2023.1.1\plugins\python\helpers\pycharm\django_manage.py" migrate C:/Users/
Tracking file by folder pattern:  migrations
Operations to perform:
  Apply all migrations: admin, app01, auth, contenttypes, sessions
Running migrations:
  Applying app01.0003_admin... OK

Process finished with exit code 0

manage.py@Django_user_manage >
```

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // D... (31 minutes ago   CRLF UTF-8 4 spaces Python 3.8



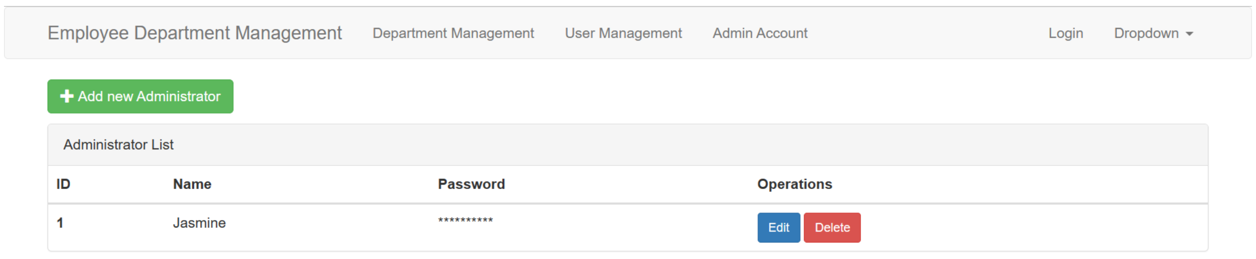
MySQL 8.0 Command Line Client

```
| world                  |
+------------------------+
12 rows in set (0.02 sec)

mysql> use django_ex_user_manage;
Database changed
mysql> show tables;
+---------------------------------+
| Tables_in_django_ex_user_manage |
+---------------------------------+
| app01_admin                     |
| app01_department                |
| app01_userinfo                  |
| auth_group                      |
| auth_group_permissions          |
| auth_permission                 |
| auth_user                       |
| auth_user_groups                |
| auth_user_user_permissions      |
| django_admin_log                |
| django_content_type             |
| django_migrations               |
| django_session                  |
+---------------------------------+
13 rows in set (0.02 sec)

mysql> insert into app01_admin(username, password) values('Jasmine', 'BHJ89');
Query OK, 1 row affected (0.07 sec)

mysql>
```

2. show page

```
def admin_list(request):
    queryset = models.Admin.objects.all()

    context = {
        'queryset': queryset
    }

    return render(request, 'admin_list.html', context)
```

```
{% extends 'layout.html' %}

{% block content %}
    <div class="container">
        <div>
            <a href="/admin/add/" class="btn btn-success" style="margin-bottom: 10px">
                <span class="glyphicon glyphicon-plus" aria-hidden="true"></span>
                Add new Administrator
            </a>
        </div>

        <div class="bs-example" data-example-id="panel-without-body-with-table">
            <div class="panel panel-default">
                <!-- Default panel contents -->
                <div class="panel-heading">Administrator List</div>

                <!-- Table -->
                <table class="table">
                    <thead>
                    <tr>
                        <th>ID</th>
                        <th>Name</th>
                        <th>Password</th>
                        <th>Operations</th>
                    </tr>
                    </thead>
                    <tbody>
                    {% for obj in queryset %}
                        <tr>
                            <th scope="row">{{ obj.id }}</th>
                            <td>{{ obj.username }}</td>
                            <td>**********</td>
                            <td>
                                <a class="btn btn-primary btn-sm" href="/admin/{{ obj.id }}/edit/">Edit</a>
                                <a class="btn btn-danger btn-sm" href="/admin/{{ obj.id }}/delete/">Delete</a>
                            </td>
                        </tr>
                    {% endfor %}

                    </tbody>
                </table>
            </div>
        </div>
        <nav aria-label="Page navigation">
            <ul class="pagination">
                {{ page_string }}
            </ul>
        </nav>


    </div>
{% endblock %}
```



Administrator List

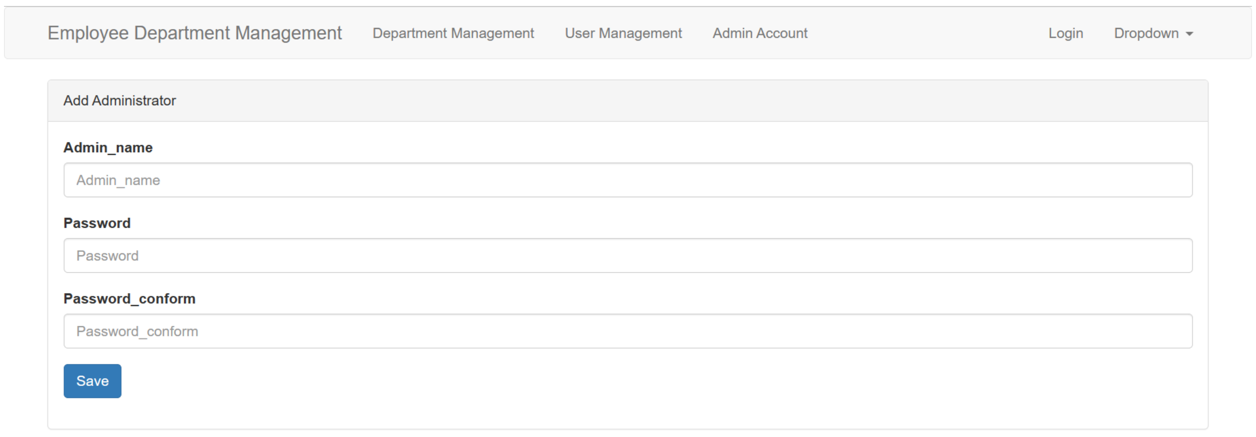
ID	Name	Password	Operations
1	Jasmine	**********	Edit Delete

3. Add Administrator

add framework reuse

```
# views.py
class AdminModelForm(BootStrapModelForm):
    confirm_password = forms.CharField(label='Password_conform')

    class Meta:
        model = models.Admin
        fields = ['username', 'password', 'confirm_password']


def admin_add(request):

    form = AdminModelForm()
    context = {
        'title': 'Add Administrator',
        'form': form,
    }

    return render(request, 'change.html', context)
```

```
# change.html
{% extends 'layout.html' %}
{% load static %}
{% block css %}
    <link rel="stylesheet" href="{% static "plugins/bootstrap-datepicker-master/dist/css/bootstrap-datepicker.css" %}">
{% endblock %}

{% block content %}
    <div class="container">
        <div class="panel panel-default">
            <div class="panel-heading">{{ title }}</div>
            <div class="panel-body">
                <form method="post" novalidate>
                    {% csrf_token %}
                    {% for field in form %}
                        <div class="form-group">
                            <label>{{ field.label }}</label>
                            {{ field }}
                            <span>{{ field.errors.0 }}</span>
                        </div>

                    {% endfor %}
                    <div class="form-group">
                        <button type="submit" class="btn btn-primary">Save</button>
                    </div>
                </form>
            </div>
        </div>

    </div>
{% endblock %}
{% block js %}
    <script src="{% static "plugins/bootstrap-datepicker-master/dist/js/bootstrap-datepicker.js" %}"></script>
<script>
    $(function () {
        $('#id_create_time').datepicker({
            format: 'yyyy-mm-dd',
            startDate: '0',
            autoclose: true
        });
    })
</script>
{% endblock %}
```



Add Administrator

Admin_name

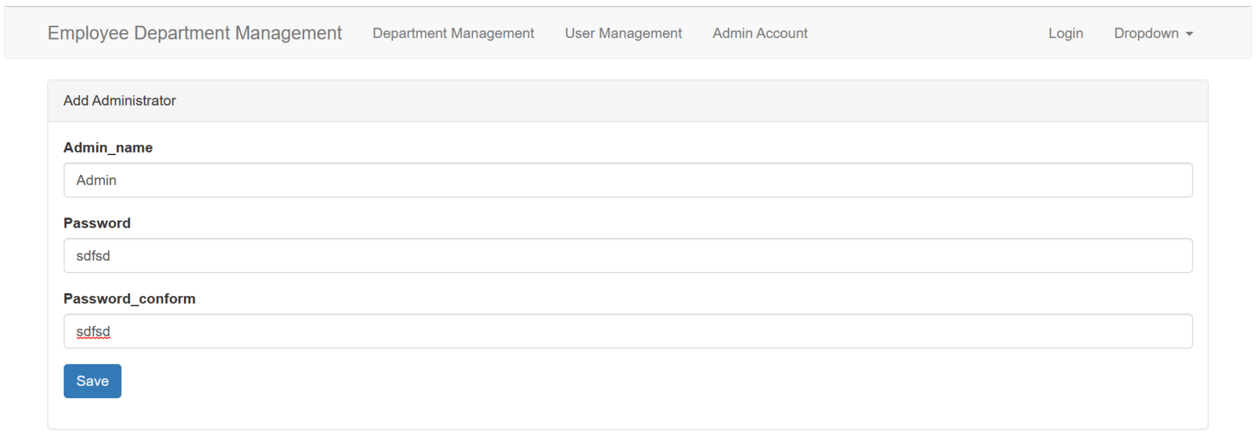
Admin_name

Password

Password

Password_conform

Password_conform



Add Administrator

Admin_name

Admin

Password

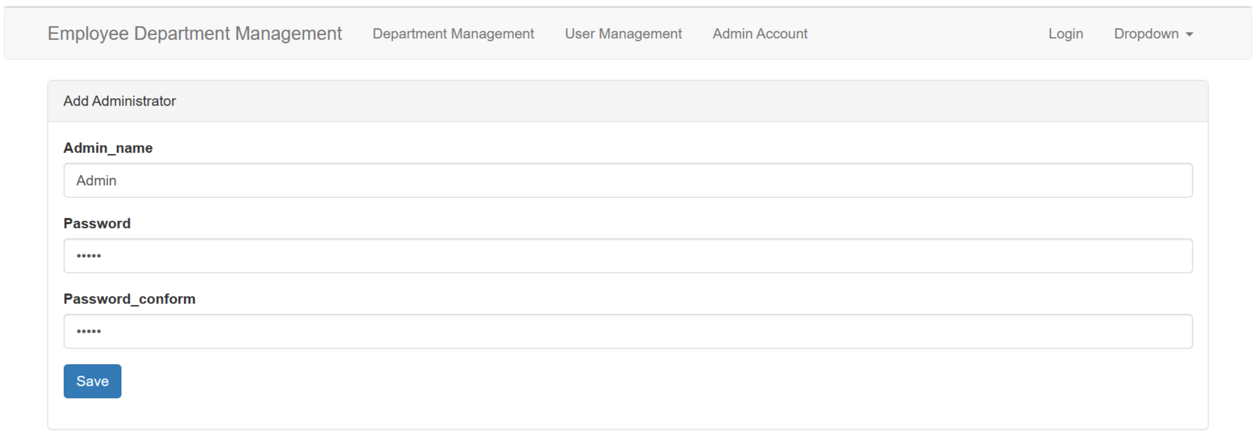
sdfsd

Password_conform

sdfsd

Change password input

```
class AdminModelForm(BootStrapModelForm):

    confirm_password = forms.CharField(label='Password_conform',
                                       widget=forms.PasswordInput)

    class Meta:
        model = models.Admin
        fields = ['username', 'password', 'confirm_password']
        widgets = {
            'password': forms.PasswordInput
        }

def admin_add(request):

    form = AdminModelForm()
    context = {
        'title': 'Add Administrator',
        'form': form,
    }

    return render(request, 'change.html', context)
```



Add Administrator

Admin_name

Admin

Password

•••••

Password_conform

•••••

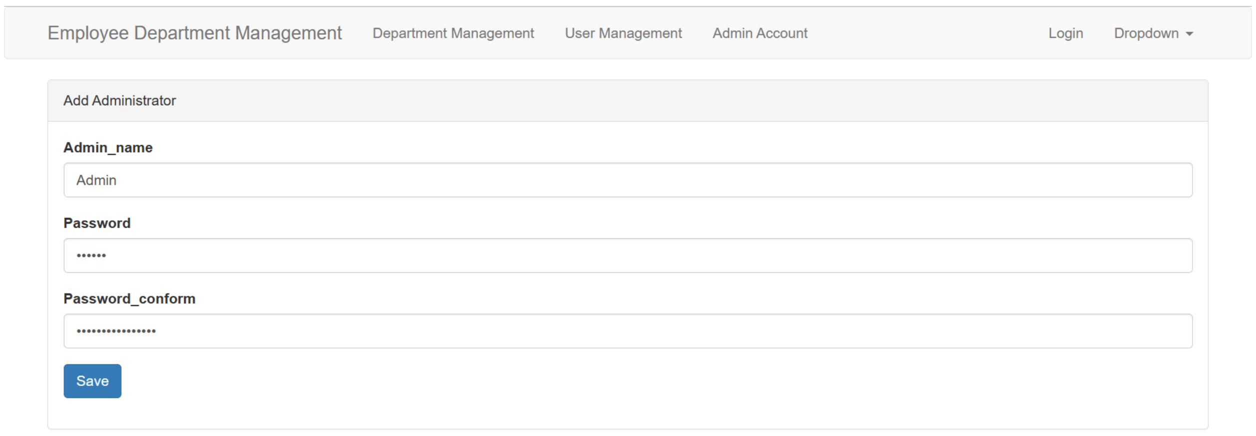
take password from POST request

```
class AdminModelForm(BootStrapModelForm):

    confirm_password = forms.CharField(label='Password_conform',
                                       widget=forms.PasswordInput)

    class Meta:
        model = models.Admin
        fields = ['username', 'password', 'confirm_password']
        widgets = {
            'password': forms.PasswordInput
        }

def admin_add(request):

    if request.method =='GET':
        form = AdminModelForm()
        context = {
            'title': 'Add Administrator',
            'form': form,
        }

        return render(request, 'change.html', context)

    form = AdminModelForm(data=request.POST)
    if form.is_valid():
        form.save()

        return redirect('/admin/list/')

    context = {
        'title': 'Add Administrator',
        'form': form,
    }
    return render(request, 'change.html', context)
```

But this will not confirm whether password and password confirm are the same.

password confirm

```
class AdminModelForm(BootStrapModelForm):

    confirm_password = forms.CharField(label='Password_conform',
                                       widget=forms.PasswordInput)

    class Meta:
        model = models.Admin
        fields = ['username', 'password', 'confirm_password']
        widgets = {
            'password': forms.PasswordInput
        }

    def clean_confirm_password(self):
        confirm = self.cleaned_data.get("confirm_password")
        pwd = self.cleaned_data.get("password")
        if confirm != pwd:
            raise ValidationError("Password not match")
        return confirm


def admin_add(request):

    if request.method =='GET':
        form = AdminModelForm()
        context = {
            'title': 'Add Administrator',
            'form': form,
        }

        return render(request, 'change.html', context)

    form = AdminModelForm(data=request.POST)
    if form.is_valid():
        # get all input
        # form.cleaned_data

        form.save()

        return redirect('/admin/list/')

    context = {
        'title': 'Add Administrator',
        'form': form,
    }
    return render(request, 'change.html', context)
```


Employee Department Management Department Management User Management Admin Account Login Dropdown

Add Administrator

Admin_name

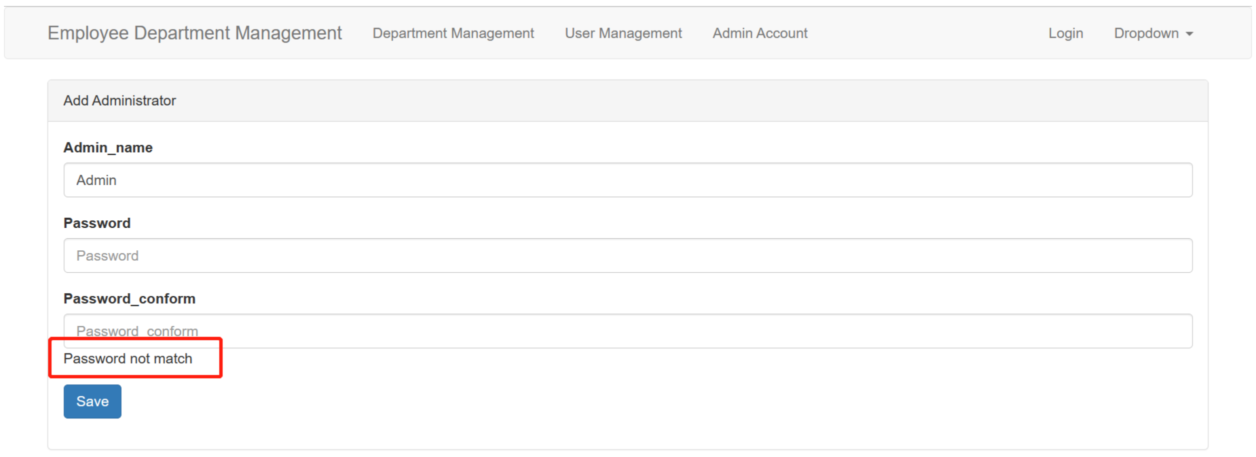
Admin

Password

••••••

Password_conform

••••••••••••••••


Employee Department Management Department Management User Management Admin Account Login Dropdown

Add Administrator

Admin_name

Admin

Password

Password

Password_conform

Password_conform

Password not match

If want to keep previous password input

add: render_value=True in widget

```
confirm_password = forms.CharField(label='Password_conform',
                                   widget=forms.PasswordInput(render_value=True))
```

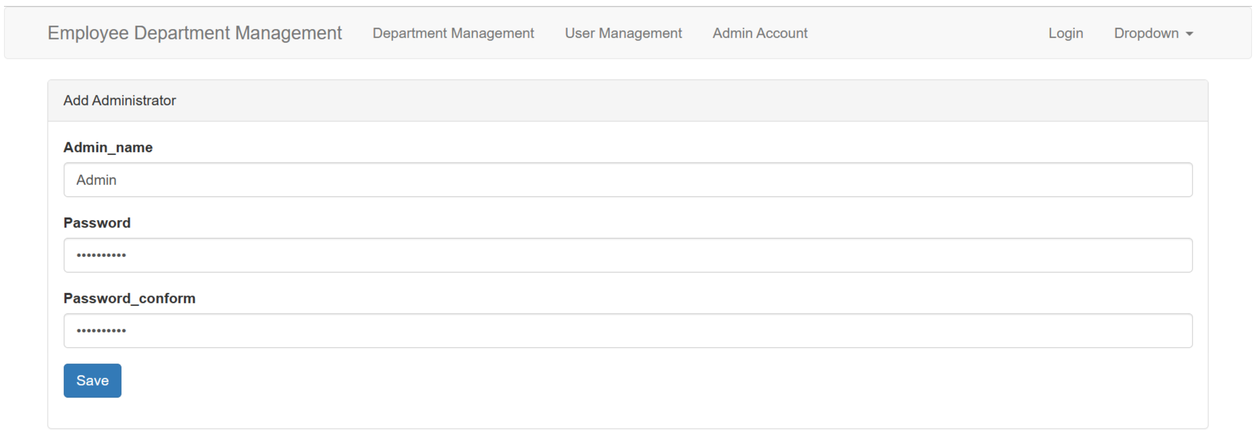
```
class AdminModelForm(BootStrapModelForm):

    confirm_password = forms.CharField(label='Password_conform',
                                       widget=forms.PasswordInput(render_value=True))

    class Meta:
        model = models.Admin
        fields = ['username', 'password', 'confirm_password']
        widgets = {
            'password': forms.PasswordInput(render_value=True)
        }

    def clean_confirm_password(self):
        confirm = self.cleaned_data.get("confirm_password")
        pwd = self.cleaned_data.get("password")
        if confirm != pwd:
            raise ValidationError("Password not match")
        return confirm
```

encrypt password

```
# encrypt.py
from django.conf import settings
import hashlib
```

```
def md5(data_string):
    obj = hashlib.md5(settings.SECRET_KEY.encode('utf-8'))
    obj.update(data_string.encode('utf-8'))
    return obj.hexdigest()
```

```
# views.py
from app01.utils.encrypt import md5
class AdminModelForm(BootStrapModelForm):

    # confirm_password = forms.CharField(label='Password_conform',
    #                                    widget=forms.PasswordInput(render_value=True))
    confirm_password = forms.CharField(label='Password_conform',
                                       widget=forms.PasswordInput)

    class Meta:
        model = models.Admin
        fields = ['username', 'password', 'confirm_password']
        widgets = {
            # 'password': forms.PasswordInput(render_value=True),
            'password': forms.PasswordInput(),
        }

    def clean_password(self):
        # we will exe clean password, return encrypt password
        pwd = self.cleaned_data.get("password")
        return md5(pwd)

    def clean_confirm_password(self):
        # what we get are already been encrypted,
        # thus we also need to encrypted password confirm for later confirm
        confirm = md5(self.cleaned_data.get("confirm_password"))
        pwd = self.cleaned_data.get("password")
        if confirm != pwd:
            raise ValidationError("Password not match")
        return confirm
```



Add Administrator

Admin_name

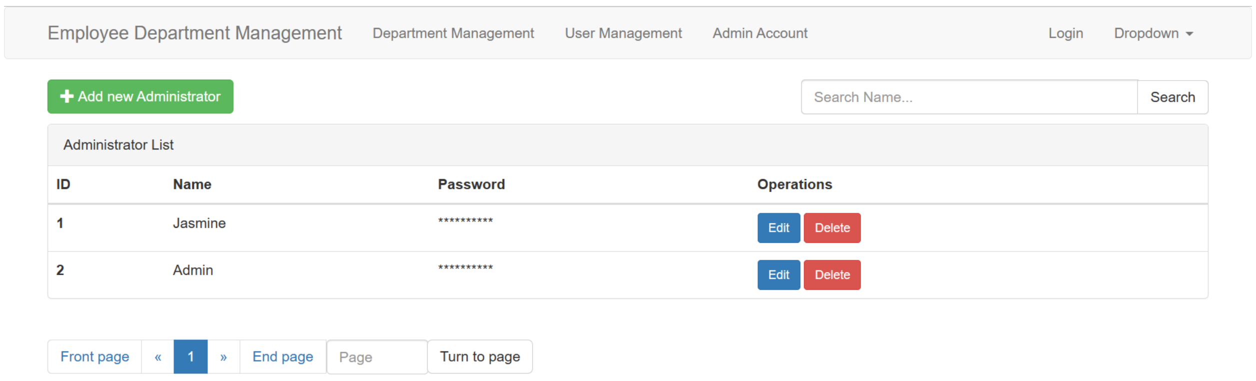
Admin

Password

••••••••••

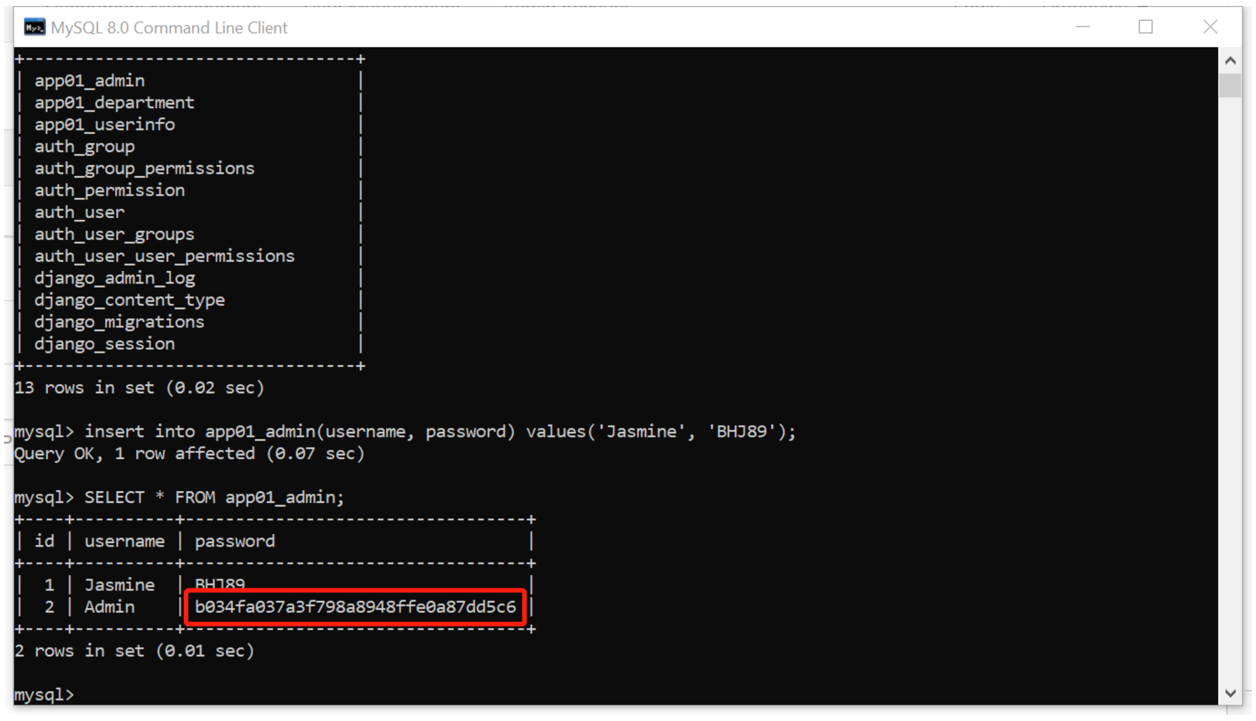
Password_conform

••••••••••



Administrator List

ID	Name	Password	Operations
1	Jasmine	**********	Edit Delete
2	Admin	**********	Edit Delete



```
+-------------------------------+
| app01_admin                   |
| app01_department              |
| app01_userinfo                |
| auth_group                    |
| auth_group_permissions        |
| auth_permission               |
| auth_user                     |
| auth_user_groups              |
| auth_user_user_permissions    |
| django_admin_log              |
| django_content_type           |
| django_migrations             |
| django_session                |
+-------------------------------+
13 rows in set (0.02 sec)

mysql> insert into app01_admin(username, password) values('Jasmine', 'BHJ89');
Query OK, 1 row affected (0.07 sec)

mysql> SELECT * FROM app01_admin;
+----+----------+----------------------------------+
| id | username | password                         |
+----+----------+----------------------------------+
|  1 | Jasmine  | BHJ89                            |
|  2 | Admin    | b034fa037a3f798a8948ffe0a87dd5c6 |
+----+----------+----------------------------------+
2 rows in set (0.01 sec)

mysql>
```

4. Edit Administrator

```
class AdminEditModelForm(BootStrapModelForm):

    # confirm_password = forms.CharField(label='Password_conform',
    #                                    widget=forms.PasswordInput(render_value=True))
    confirm_password = forms.CharField(label='Password_conform',
                                      widget=forms.PasswordInput)

    class Meta:
        model = models.Admin
        fields = ['password', 'confirm_password']
        widgets = {
            # 'password': forms.PasswordInput(render_value=True),
            'password': forms.PasswordInput(),
        }

    def clean_password(self):
        # we will exe clean password, return encrypt password
        pwd = self.cleaned_data.get("password")
        return md5(pwd)

    def clean_confirm_password(self):
        # what we get are already been encrypted,
        # thus we also need to encrypted password confirm for later confirm
        confirm = md5(self.cleaned_data.get("confirm_password"))
        pwd = self.cleaned_data.get("password")
        if confirm != pwd:
            raise ValidationError("Password not match")
        return confirm
```

```
def admin_edit(request, nid):

    # check if id exist
    row_object = models.Admin.objects.filter(id=nid).first()
    if not row_object:
        return redirect('/admin/list/')

    if request.method == 'GET':
        form = AdminEditModelForm(instance=row_object)
        context = {
            'title': 'Edit Administrator',
            'form': form,
        }
        return render(request, 'change.html', context)

    form = AdminEditModelForm(data=request.POST, instance=row_object)
    if form.is_valid():
        form.save()
        return redirect('/admin/list/')

    context = {
        'title': 'Edit Administrator',
        'form': form,
    }
    return render(request, 'change.html', context)
```

not allowed using previous password

```
class AdminEditModelForm(BootStrapModelForm):

    # confirm_password = forms.CharField(label='Password_conform',
    #                                     widget=forms.PasswordInput(render_value=True))
    confirm_password = forms.CharField(label='Password_conform',
                                       widget=forms.PasswordInput)

    class Meta:
        model = models.Admin
        fields = ['password', 'confirm_password']
        widgets = {
            # 'password': forms.PasswordInput(render_value=True),
            'password': forms.PasswordInput(),
        }

    def clean_password(self):
        # we will exe clean password, return encrypt password
        pwd = self.cleaned_data.get("password")
        md5_pwd = md5(pwd)
        # self.instance: object we pass through instance
        # AdminEditModelForm(instance=row_object)
        # thus self.instance -> row_object
        # self.instance.pk (primary key-> id)
        exist = models.Admin.objects.filter(id=self.instance.pk, password=md5_pwd).first()
        if exist:
            raise ValidationError("Can not use previous password")

        return md5(pwd)

    def clean_confirm_password(self):
        # what we get are already been encrypted,
        # thus we also need to encrypted password confirm for later confirm
        confirm = md5(self.cleaned_data.get("confirm_password"))
        pwd = self.cleaned_data.get("password")
        if confirm != pwd:
            raise ValidationError("Password not match")
        return confirm

def admin_edit(request, nid):

    # check if id exist
    row_object = models.Admin.objects.filter(id=nid).first()
    if not row_object:
        return redirect('/admin/list/')

    if request.method == 'GET':
        form = AdminEditModelForm(instance=row_object)
        context = {
            'title': 'Edit Administrator',
            'form': form,
        }
        return render(request, 'change.html', context)

    form = AdminEditModelForm(data=request.POST, instance=row_object)
    if form.is_valid():
        form.save()
        return redirect('/admin/list/')

    context = {
        'title': 'Edit Administrator',
        'form': form,
    }
    return render(request, 'change.html', context)
```