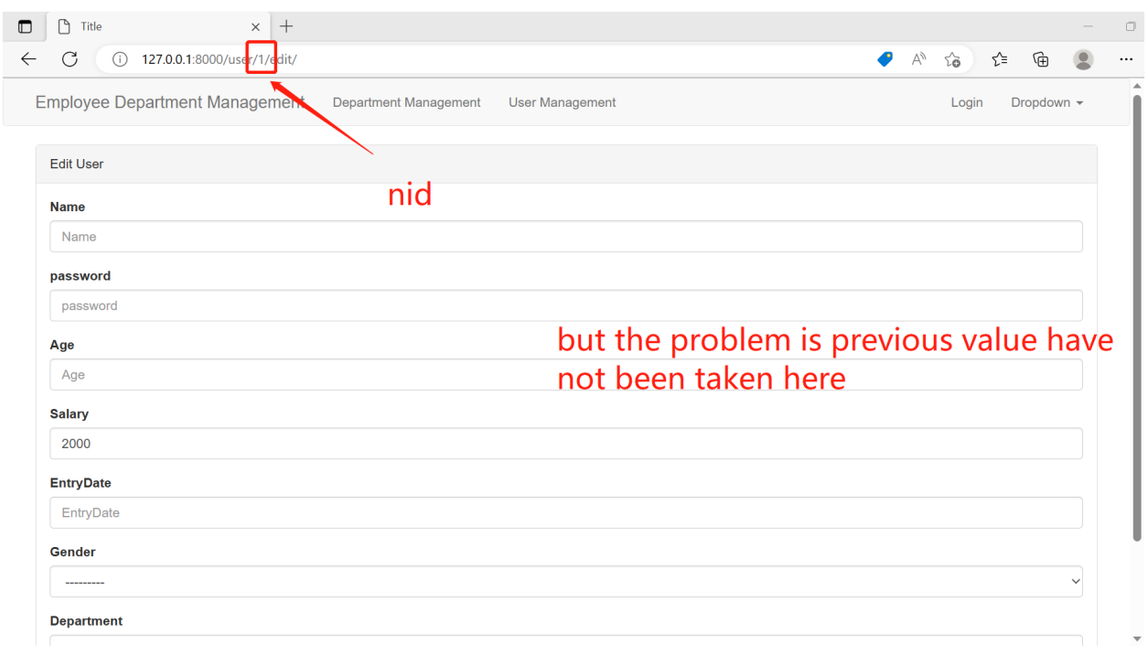
User Management - continue

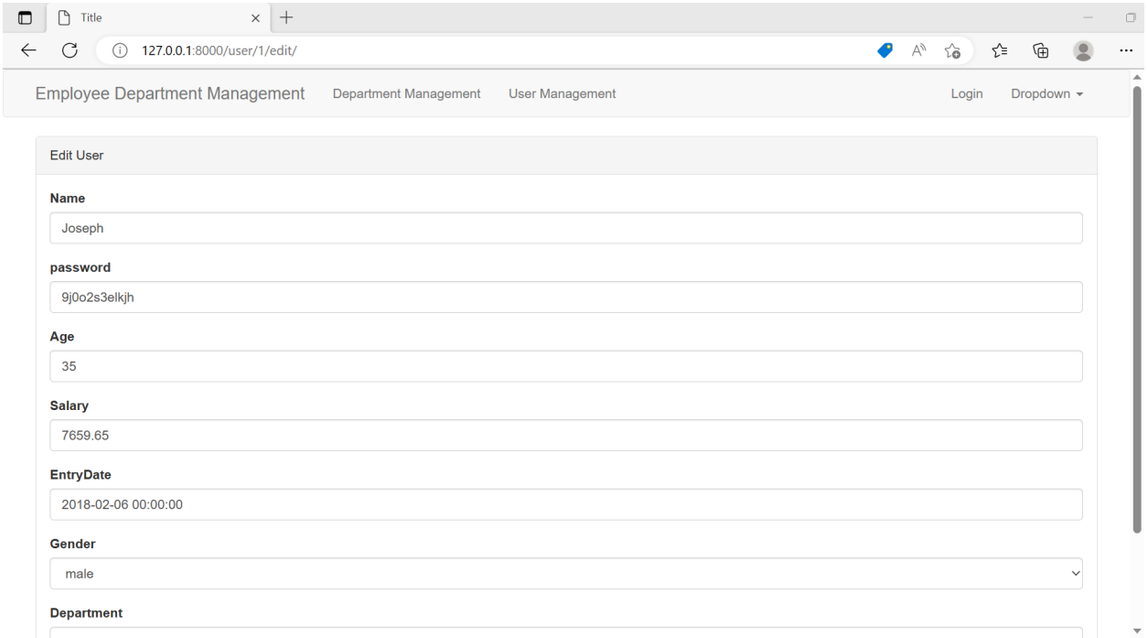
1. edit User

```
def user_edit(request, nid):

    form = UserModelForm()

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

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

{% block content %}
    <div class="container">
        <div class="panel panel-default">
            <div class="panel-heading"> Edit User</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 %}
```



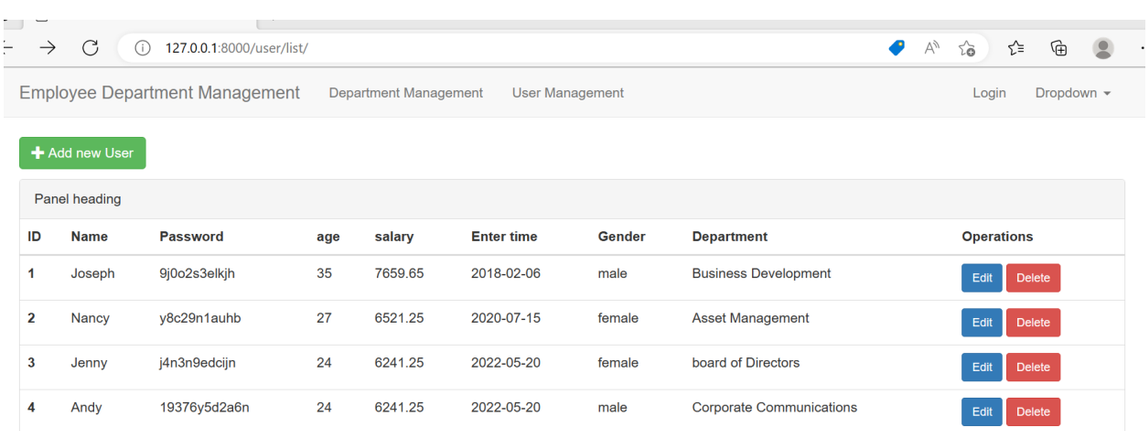

```
def user_edit(request, nid):
    # get data from database used nid
    row_object = models.UserInfo.objects.filter(id=nid).first()
    form = UserModelForm(instance=row_object)

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

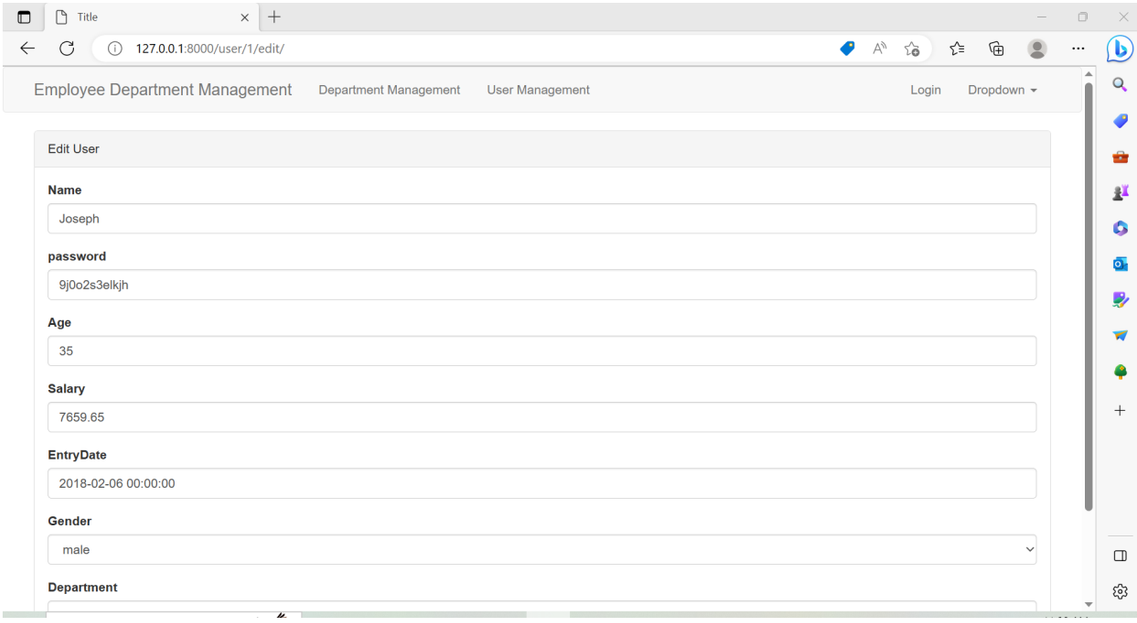


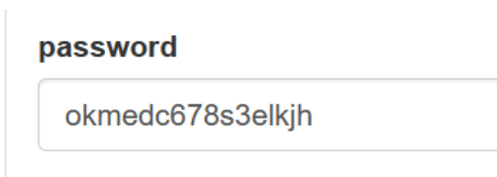

```
def user_edit(request, nid):
    if request.method == 'GET':
        # get data from database used nid
        row_object = models.UserInfo.objects.filter(id=nid).first()
        form = UserModelForm(instance=row_object)

        return render(request, 'user_edit.html', {'form': form})

    form = UserModelForm(data=request.POST)
    if form.is_valid():
        form.save()
        return redirect('/user/list/')
    # error message
    return render(request, 'user_add.html', {'form': form})
```

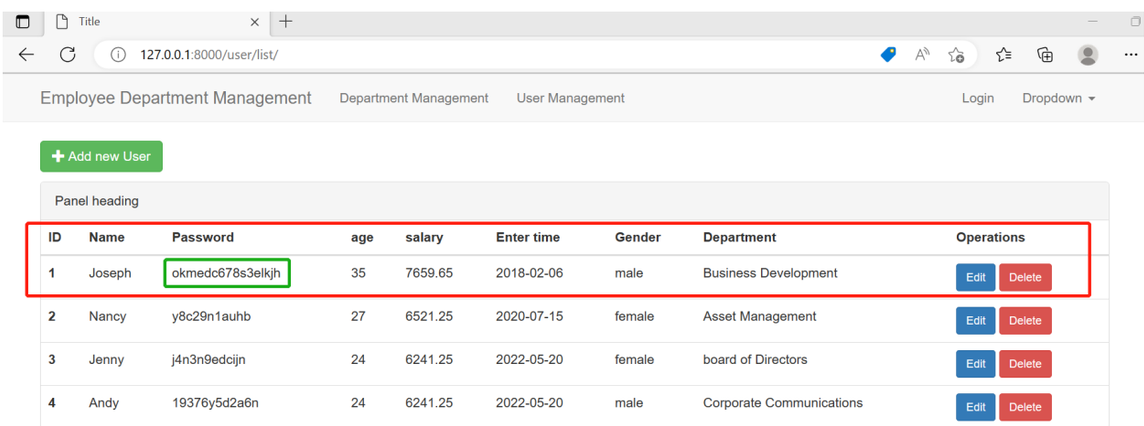
But this is not add, it is edit. If we only use save(), it will just simply add this record to the table.

```
def user_edit(request, nid):
    if request.method == 'GET':
        # get data from database used nid
        row_object = models.UserInfo.objects.filter(id=nid).first()
        form = UserModelForm(instance=row_object)

        return render(request, 'user_edit.html', {'form': form})

    # get data from POST, save data to id=nid
    row_object = models.UserInfo.objects.filter(id=nid).first()
    form = UserModelForm(data=request.POST, instance=row_object)
    if form.is_valid():
        form.save()
        return redirect('/user/list/')
    # error message
    return render(request, 'user_add.html', {'form': form})
```









simplify

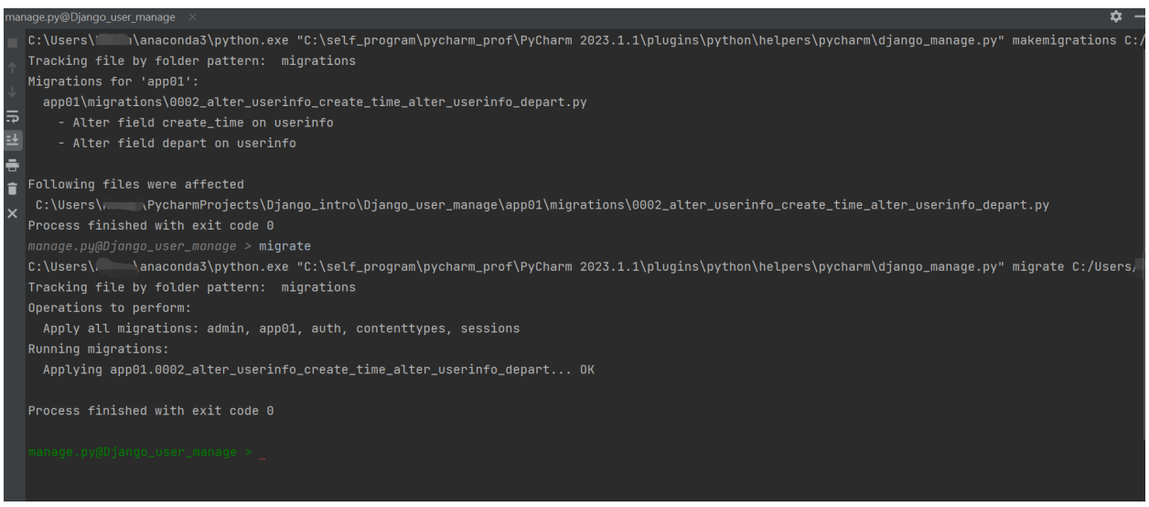
```
def user_edit(request, nid):
    # get data from database used nid
    row_object = models.UserInfo.objects.filter(id=nid).first()
    if request.method == 'GET':
        form = UserModelForm(instance=row_object)
        return render(request, 'user_edit.html', {'form': form})

    # get data from POST, save data to id=nid
    form = UserModelForm(data=request.POST, instance=row_object)
    if form.is_valid():
        form.save()
        return redirect('/user/list/')
    # error message
    return render(request, 'user_add.html', {'form': form})
```

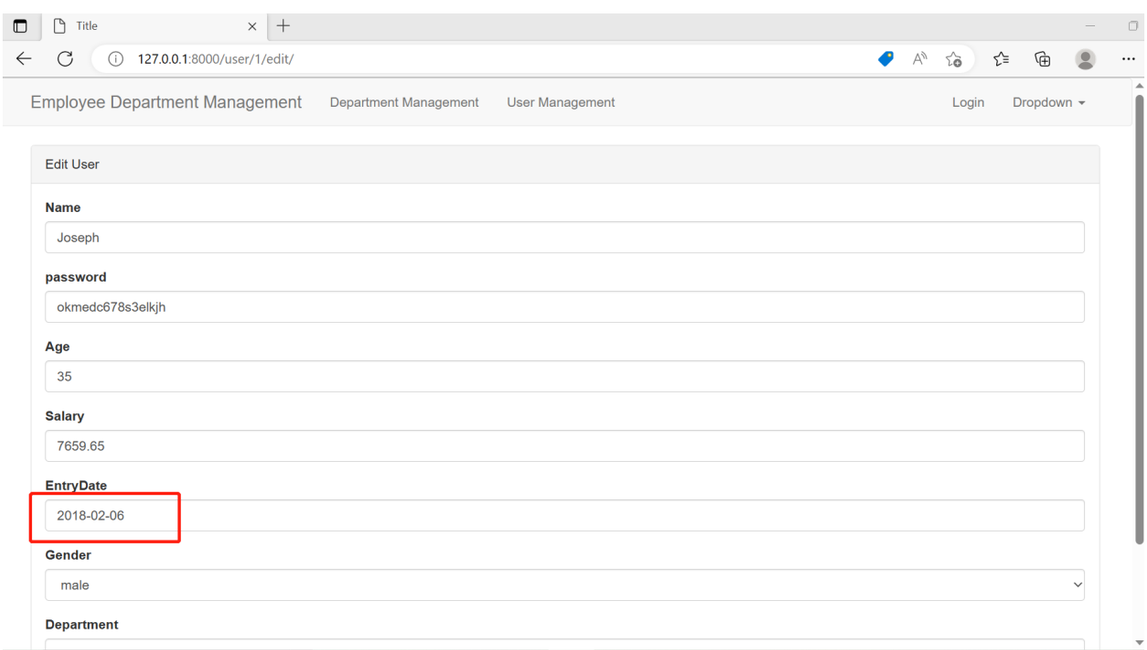
Change table

Datetime - Date

```
class UserInfo(models.Model):
    """employee info"""
    name = models.CharField(verbose_name='Name', max_length=16)
    password = models.CharField(verbose_name='Password', max_length=64)
    age = models.IntegerField(verbose_name='Age')
    salary = models.DecimalField(verbose_name='Salary', max_digits=10,\
                                 decimal_places=2, default=2000)
```

```
    create_time = models.DateField(verbose_name='EntryDate')
    """
    Join Department
    On id
    """
    # this column will be named as depart_id
    # only id in Department table can appear in this col
    # if delete rows in Department table, also drop related records in this table
    # "on_delete=models.CASCADE"
    depart = models.ForeignKey(verbose_name='Department', to='Department', to_field='id', on_delete=models.CASCADE)
    # if delete rows in Department table, set related records NA in this table
    # "null=True, blank=True, on_delete=models.SET_NULL"
    # depart = models.ForeignKey(to='Department', to_field='id', null=True, blank=True, on_delete=models.SET_NULL)

    """
    also, if we do not want to connect to other tables, and the values are really simple,
    like "gender", only male and female,
    we can also use tuple
    """
    gender_choices = (
        (1, 'male'),
        (2, 'female')
    )
    gender = models.SmallIntegerField(verbose_name='gender', choices=gender_choices)
```






```
        # form.save: save all info from custom input in form
        # if we have some info which does not from custom input
        # we have:
        # from.instance.<keyName> = <value>
```

2. Delete User

add some user for delete

Employee Department Management Department Management User Management Login Dropdown

+ Add new User

Panel heading

ID	Name	Password	age	salary	Enter time	Gender	Department	Operations
1	Joseph	okmedc678s3elkjh	35	7659.65	2018-02-06	male	Business Development	Edit Delete
2	Nancy	y8c29n1auhb	27	6521.25	2020-07-15	female	Asset Management	Edit Delete
3	Jenny	j4n3n9edcijn	24	6241.25	2022-05-20	female	board of Directors	Edit Delete
4	Andy	19376y5d2a6n	24	6241.25	2022-05-20	male	Corporate Communications	Edit Delete
5	Oscar	0l9v3r9ijnbgt5	28	8956.43	2017-03-23	male	Creative Services	Edit Delete
6	Tobin	w9n2t0ntkv345	32	8493.34	2017-02-18	male	Customer Experience	Edit Delete
7	Lilac	A4a9i8t1n3	25	6043.35	2022-06-25	female	Corporate Communications	Edit Delete
8	Marigold	2l7t6eS1w7e4	24	6457.65	2021-09-24	female	board of Directors	Edit Delete

add urls

```
path('user/<int:nid>/delete/', views.user_delete),
```

function

```
def user_delete(request, nid):
    models.UserInfo.objects.filter(id=nid).delete()
    return redirect('/user/list/')
```

Attention

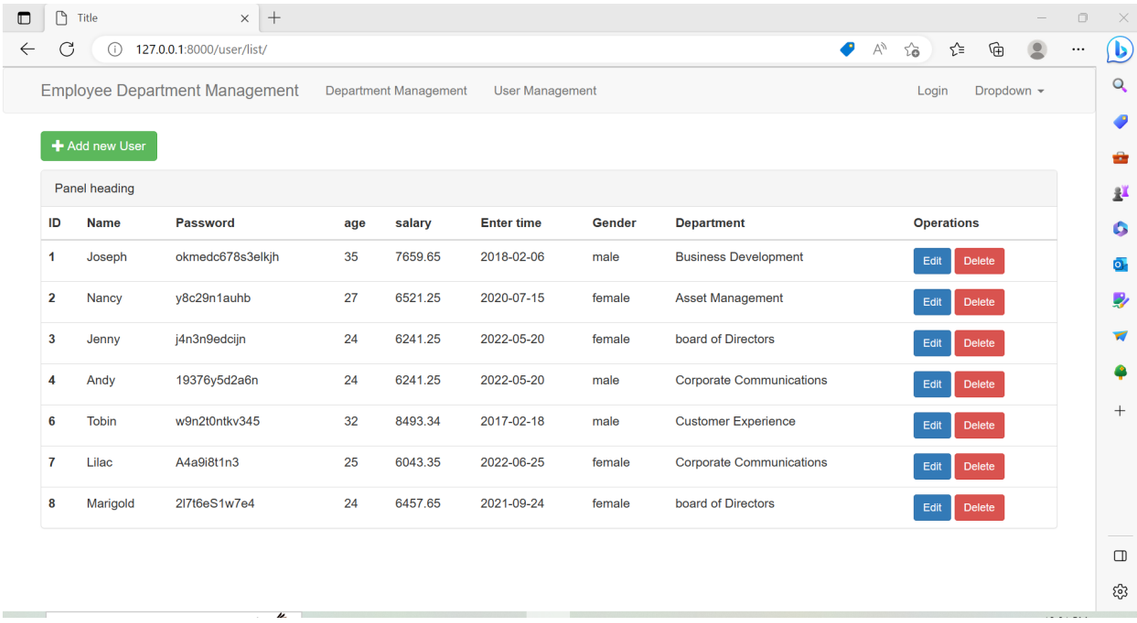
1. must add urls-function pairs
2. may not have an exact html page

Employee Department Management Department Management User Management Login Dropdown

+ Add new User

Panel heading

ID	Name	Password	age	salary	Enter time	Gender	Department	Operations
1	Joseph	okmedc678s3elkjh	35	7659.65	2018-02-06	male	Business Development	Edit Delete
2	Nancy	y8c29n1auhb	27	6521.25	2020-07-15	female	Asset Management	Edit Delete
3	Jenny	j4n3n9edcijn	24	6241.25	2022-05-20	female	board of Directors	Edit Delete
4	Andy	19376y5d2a6n	24	6241.25	2022-05-20	male	Corporate Communications	Edit Delete
5	Oscar	0l9v3r9ijnbgt5	28	8956.43	2017-03-23	male	Creative Services	Edit Delete
6	Tobin	w9n2t0ntkv345	32	8493.34	2017-02-18	male	Customer Experience	Edit Delete
7	Lilac	A4a9i8t1n3	25	6043.35	2022-06-25	female	Corporate Communications	Edit Delete
8	Marigold	2l7t6eS1w7e4	24	6457.65	2021-09-24	female	board of Directors	Edit Delete



Regular expression check

```python
class PrettyModelForm(forms.ModelForm):
    mobile = forms.CharField(
        label="手机号",
        validators=[RegexValidator(r'^159[0-9]+$', '数字必须以159开头')],
    )
```

```python
# 验证: 方式2
def clean_mobile(self):
    txt_mobile = self.cleaned_data["mobile"]

    if len(txt_mobile) != 11:
        # 验证不通过
        raise ValidationError("格式错误")

    # 验证通过, 用户输入的值返回
    return txt_mobile
```

```python
class PrettyModelForm(forms.ModelForm):
    # 验证：方式1
    # mobile = forms.CharField(
    #     label="手机号",
    #     validators=[RegexValidator(r'^1[3-9]\d{9}$', '手机号格式错误'), ],
    # )

    class Meta:
        model = models.PrettyNum
        # fields = "__all__"
        # exclude = ['level']
        fields = ["mobile", 'price', 'level', 'status']

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        for name, field in self.fields.items():
            field.widget.attrs = {"class": "form-control", "placeholder": field.label}

    # 验证：方式2
    def clean_mobile(self):
        txt_mobile = self.cleaned_data["mobile"]

        if len(txt_mobile) != 11:
            # 验证不通过
            raise ValidationError("格式错误")

        # 验证通过，用户输入的值返回
        return txt_mobile
```

字段 + 正则

clean_字段名

获取用户输入的值

钩子方法

unable change some cols

```python
class PrettyEditModelForm(forms.ModelForm):
    mobile = forms.CharField(disabled=True,label="手机号")

    class Meta:
        model = models.PrettyNum
        fields = ['mobile', 'price', 'level', 'status']

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        for name, field in self.fields.items():
            field.widget.attrs = {"class": "form-control", "placeholder": field.label}
```

disable=True

× allowed duplicate

不允许手机号重复。

- 添加：【正则表达式】【手机号不能存在】

```python
# [obj,obj,obj]
queryset = models.PrettyNum.objects.filter(mobile="1888888888")

obj = models.PrettyNum.objects.filter(mobile="1888888888").first()

# True/False
exists = models.PrettyNum.objects.filter(mobile="1888888888").exists()
```

python

```python
# 验证：方式2
def clean_mobile(self):
    txt_mobile = self.cleaned_data["mobile"]

    exists = models.PrettyNum.objects.filter(mobile=txt_mobile).exists()
    if exists:
        raise ValidationError("手机号已存在")

    # 验证通过，用户输入的值返回
    return txt_mobile
```

- 编辑：【正则表达式】【手机号不能存在】

排除自己以外，其他的数据是否手机号是否重复？

```python
# id!=2 and mobile='1888888888'
models.PrettyNum.objects.filter(mobile="1888888888").exclude(id=2)
```

python

self.instance.pk

primary key of the choosen row.

get id of the edit row.

```python
# 验证：方式2
def clean_mobile(self):

    # 当前编辑的哪一行的ID
    # print(self.instance.pk)

    txt_mobile = self.cleaned_data["mobile"]
    exists = models.PrettyNum.objects.exclude(id=self.instance.pk).filter(mobile=txt_mobile).exists()
    if exists:
        raise ValidationError("手机号已存在")

    # 验证通过，用户输入的值返回
    return txt_mobile
```

Search

9.5 搜索手机号

```python
models.PrettyNum.objects.filter(mobile="19999999991",id=12)
```

法一

```python
data_dict = {"mobile":"19999999991","id":123}
models.PrettyNum.objects.filter(**data_dict)
```

法二

python

```python
def pretty_list(request):
    """ 靓号列表 """

    q = models.PrettyNum.objects.filter(mobile="19999999991", id=6)
    print(q)

    # select * from 表 order by level desc;
    queryset = models.PrettyNum.objects.all().order_by("-level")
    return render(request, 'pretty_list.html', {"queryset": queryset})
```

```python
#  [<PrettyNum: PrettyNum object (6)>  ]
q1 = models.PrettyNum.objects.filter(mobile="19999999991", id=6)
print(q1)

data_dict = {"mobile": "19999999991", "id": 6}
q2 = models.PrettyNum.objects.filter(**data_dict)
print(q2)
```

```python
models.PrettyNum.objects.filter(id=12)        # 等于12
models.PrettyNum.objects.filter(id__gt=12)    # 大于12
models.PrettyNum.objects.filter(id__gte=12)   # 大于等于12
models.PrettyNum.objects.filter(id__lt=12)    # 小于12
models.PrettyNum.objects.filter(id__lte=12)   # 小于等于12

data_dict = {"id__lte":12}
models.PrettyNum.objects.filter(**data_dict)
```

```python
models.PrettyNum.objects.filter(mobile="999")                # 等于
models.PrettyNum.objects.filter(mobile__startswith="1999")   # 筛选出以1999开头
models.PrettyNum.objects.filter(mobile__endswith="999")      # 筛选出以999结尾
models.PrettyNum.objects.filter(mobile__contains="999")      # 筛选出包含999

data_dict = {"mobile__contains":"999"}
models.PrettyNum.objects.filter(**data_dict)
```

python

int

```python
models.PrettyNum.objects.filter(id=12)        # 等于12
models.PrettyNum.objects.filter(id__gt=12)    # 大于12
models.PrettyNum.objects.filter(id__gte=12)   # 大于等于12
models.PrettyNum.objects.filter(id__lt=12)    # 小于12
models.PrettyNum.objects.filter(id__lte=12)   # 小于等于12
```

python

str

```python
models.PrettyNum.objects.filter(mobile="999")                # 等于
models.PrettyNum.objects.filter(mobile__startswith="1999")   # 筛选出以1999开头
models.PrettyNum.objects.filter(mobile__endswith="999")      # 筛选出以999结尾
models.PrettyNum.objects.filter(mobile__contains="999")      # 筛选出包含999
```

python

```python
def pretty_list(request):
    """ 靓号列表 """
    data_dict = {}
    value = request.GET.get('q')
    if value:
        data_dict["mobile__contains"] = value

    res = models.PrettyNum.objects.filter(**data_dict)
    print(res)
```

create an empty dict, if there is an input in the searching bar, add that value to the dict, else, the dict will still be empty.

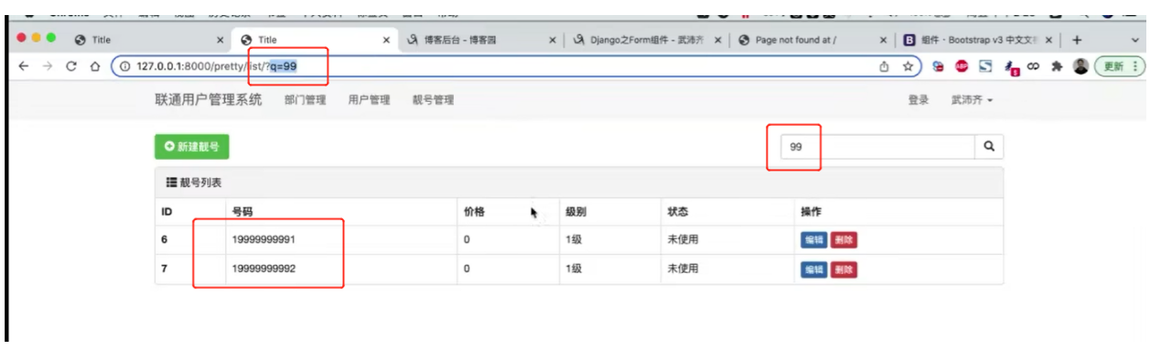
```python
def pretty_list(request):
    """ 靓号列表 """
    data_dict = {}
    value = request.GET.get('q')
    if value:
        data_dict["mobile__contains"] = value

    queryset = models.PrettyNum.objects.filter(**data_dict).order_by("-level")

    return render(request, 'pretty_list.html', {"queryset": queryset})
```

```html
<div class="container">
    <div style="margin-bottom: 10px" class="clearfix">
        <a class="btn btn-success" href="/pretty/add/">
            <span class="glyphicon glyphicon-plus-sign" aria-hidden="true"></span>
            新建靓号
        </a>

        <div style="float: right;width: 300px;">
            <div class="input-group">
                <input type="text" class="form-control" placeholder="Search for...">
                <span class="input-group-btn">
                    <button class="btn btn-default" type="button">
                        <span class="glyphicon glyphicon-search" aria-hidden="true"></span>
                    </button>
                </span>
            </div>
```

```html
<div style="float: right;width: 300px;">
    <form method="get">
        <div class="input-group">
            <input type="text" name="q" class="form-control" placeholder="Search for...">
            <span class="input-group-btn">
            <button class="btn btn-default" type="submit">
                <span class="glyphicon glyphicon-search" aria-hidden="true"></span>
            </button>
            </span>
        </div>
    </form>
</div>
</div>
```

```html
at: right;width: 300px;">
d="get">
ass="input-group">
put type="text" name="q" class="form-control" placeholder="Search for..." value="{{ search_data }}">
an class="input-group-btn">
tton class="btn btn-default" type="submit">
 <span class="glyphicon glyphicon-search" aria-hidden="true"></span>
utton>
n>
```

```python
def pretty_list(request):
    """ 靓号列表 """
    data_dict = {}
    search_data = request.GET.get('q', "")
    if search_data:
        data_dict["mobile__contains"] = search_data

    queryset = models.PrettyNum.objects.filter(**data_dict).order_by("-level")

    return render(request, 'pretty_list.html', {"queryset": queryset, "search_data": search_data})
```



3. Paging

add more info to User management

ID	Name	Password	age	salary	Enter time	Gender	Department	Operations
1	Joseph	okmedc678s3elkjh	35	7659.65	2018-02-06	male	Business Development	Edit Delete
2	Nancy	y8c29n1auhb	27	6521.25	2020-07-15	female	Asset Management	Edit Delete
3	Jenny	j4n3n9edcijn	24	6241.25	2022-05-20	female	board of Directors	Edit Delete
4	Andy	19376y5d2a6n	24	6241.25	2022-05-20	male	Corporate Communications	Edit Delete
6	Tobin	w9n2t0ntkv345	32	8493.34	2017-02-18	male	Customer Experience	Edit Delete
7	Lilac	A4a9i8t1n3	25	6043.35	2022-06-25	female	Corporate Communications	Edit Delete
8	Marigold	2l7t6eS1w7e4	24	6457.65	2021-09-24	female	board of Directors	Edit Delete
9	Archer	A4c8e53edcvgy7	30	8023.45	2018-03-21	male	Business Development	Edit Delete
10	Brooks	b0o028oo84	32	7835.32	2019-01-23	male	board of Directors	Edit Delete
11	Fletcher	6l3t3h4rf	24	6235.65	2021-01-23	male	Customer Experience	Edit Delete
12	Evie	4v8e37ie	23	6024.25	2022-04-03	female	Customer Experience	Edit Delete
13	Lillian	9l0kil0ane2q	26	8756.75	2019-06-01	female	Asset Management	Edit Delete

we can decide how many records shown in one page:

```
def user_list(request):
    queryset = models.UserInfo.objects.all()[0:5]

    return render(request, 'user_list.html', {'queryset': queryset})
```

Employee Department Management Department Management User Management Login Dropdown

+ Add new User

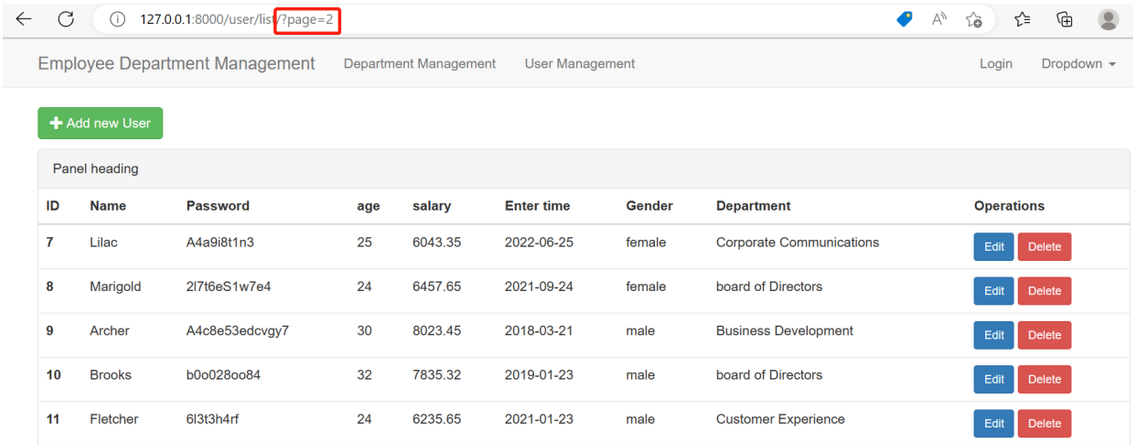
Panel heading

ID	Name	Password	age	salary	Enter time	Gender	Department	Operations
1	Joseph	okmedc678s3elkjh	35	7659.65	2018-02-06	male	Business Development	Edit Delete
2	Nancy	y8c29n1auhb	27	6521.25	2020-07-15	female	Asset Management	Edit Delete
3	Jenny	j4n3n9edcijn	24	6241.25	2022-05-20	female	board of Directors	Edit Delete
4	Andy	19376y5d2a6n	24	6241.25	2022-05-20	male	Corporate Communications	Edit Delete
6	Tobin	w9n2t0ntkv345	32	8493.34	2017-02-18	male	Customer Experience	Edit Delete

idea:

get page from get request, calculate the start and end index of each page.

```
def user_list(request):
    # page, default value=1
    page = int(request.GET.get('page', 1))
    start = (page-1)*5
    end = page*5
    queryset = models.UserInfo.objects.all()[start:end]

    return render(request, 'user_list.html', {'queryset': queryset})
```



ID	Name	Password	age	salary	Enter time	Gender	Department	Operations
7	Lilac	A4a9i8t1n3	25	6043.35	2022-06-25	female	Corporate Communications	Edit Delete
8	Marigold	2l7t6eS1w7e4	24	6457.65	2021-09-24	female	board of Directors	Edit Delete
9	Archer	A4c8e53edcvgy7	30	8023.45	2018-03-21	male	Business Development	Edit Delete
10	Brooks	b0o028oo84	32	7835.32	2019-01-23	male	board of Directors	Edit Delete
11	Fletcher	6l3t3h4rf	24	6235.65	2021-01-23	male	Customer Experience	Edit Delete

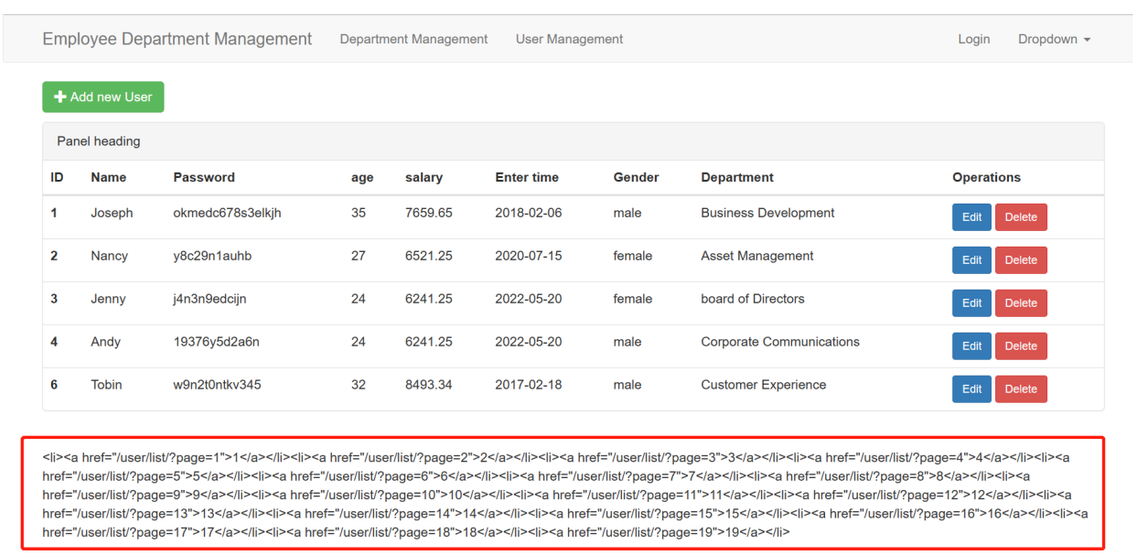
of course, we can add pages info in html

```
<li><a href="?page=1">1</a></li>
<li><a href="?page=2">2</a></li>
```

but we can also create pages in python

```
def user_list(request):
    # page, default value=1
    page = int(request.GET.get('page', 1))
    start = (page-1)*5
    end = page*5

    queryset = models.UserInfo.objects.all()[start:end]
    # for obj in queryset:
    #     print(obj.id, obj.name, obj.salary, obj.age)
    #     print(obj.create_time, obj.create_time.strftime('%Y-%m-%d'))
    #     print(obj.gender, obj.get_gender_display())
    #     print(obj.depart_id, obj.depart.department)

    page_str_list = []
    for i in range(1, 20):
        ele = '<li><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
        page_str_list.append(ele)
    page_string = "".join(page_str_list)

    return render(request, 'user_list.html', {'queryset': queryset, 'page_string': page_string})
```

but actually, this will not give what we want:



ID	Name	Password	age	salary	Enter time	Gender	Department	Operations
1	Joseph	okmedc678s3elkjh	35	7659.65	2018-02-06	male	Business Development	Edit Delete
2	Nancy	y8c29n1auhb	27	6521.25	2020-07-15	female	Asset Management	Edit Delete
3	Jenny	j4n3n9edcijn	24	6241.25	2022-05-20	female	board of Directors	Edit Delete
4	Andy	19376y5d2a6n	24	6241.25	2022-05-20	male	Corporate Communications	Edit Delete
6	Tobin	w9n2t0ntkv345	32	8493.34	2017-02-18	male	Customer Experience	Edit Delete

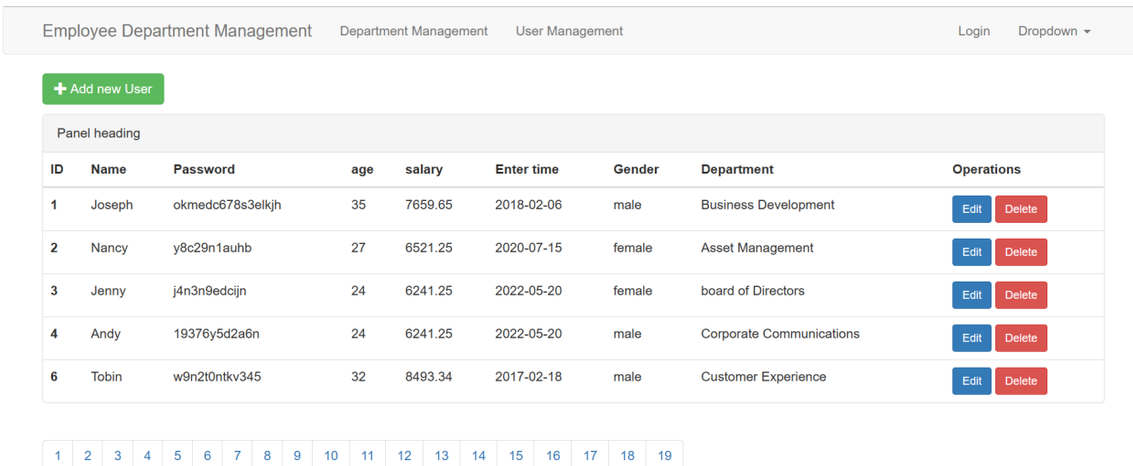
```
<li><a href="/user/list/?page=1">1</a></li><li><a href="/user/list/?page=2">2</a></li><li><a href="/user/list/?page=3">3</a></li><li><a href="/user/list/?page=4">4</a></li><li><a
href="/user/list/?page=5">5</a></li><li><a href="/user/list/?page=6">6</a></li><li><a href="/user/list/?page=7">7</a></li><li><a href="/user/list/?page=8">8</a></li><li><a
href="/user/list/?page=9">9</a></li><li><a href="/user/list/?page=10">10</a></li><li><a href="/user/list/?page=11">11</a></li><li><a href="/user/list/?page=12">12</a></li><li><a
href="/user/list/?page=13">13</a></li><li><a href="/user/list/?page=14">14</a></li><li><a href="/user/list/?page=15">15</a></li><li><a href="/user/list/?page=16">16</a></li><li><a
href="/user/list/?page=17">17</a></li><li><a href="/user/list/?page=18">18</a></li><li><a href="/user/list/?page=19">19</a></li>
```

to solving this, we need to mark these strings as “safe” to translate them into html. This will need to import the model:

```
from django.utils.safestring import mark_safe
```

and

```
    page_str_list = []
    for i in range(1, 20):
        ele = '<li><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
        page_str_list.append(ele)
    page_string = mark_safe("".join(page_str_list))

    return render(request, 'user_list.html', {'queryset': queryset, 'page_string': page_string})
```

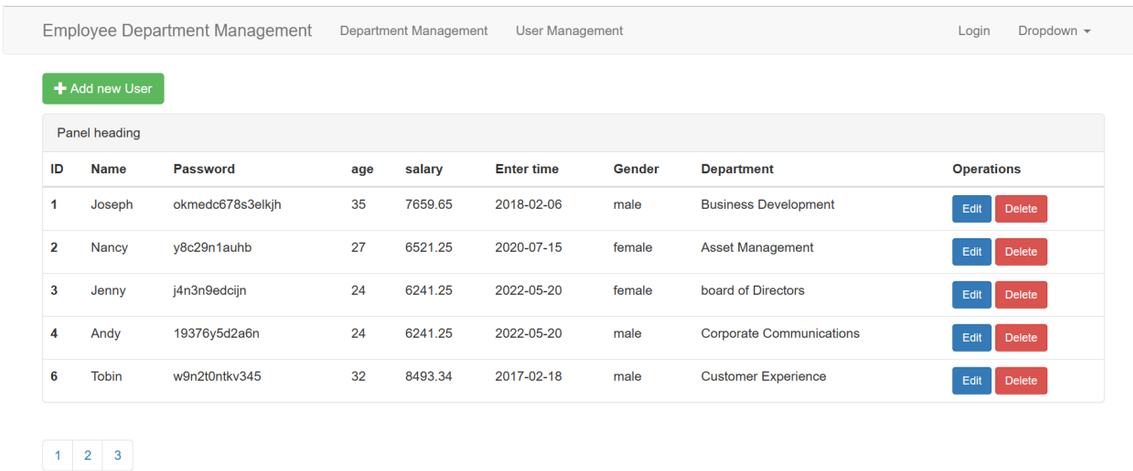


How to decide how many pages we need?

```
# # get total number of record
total_count = models.UserInfo.objects.all().count()
```

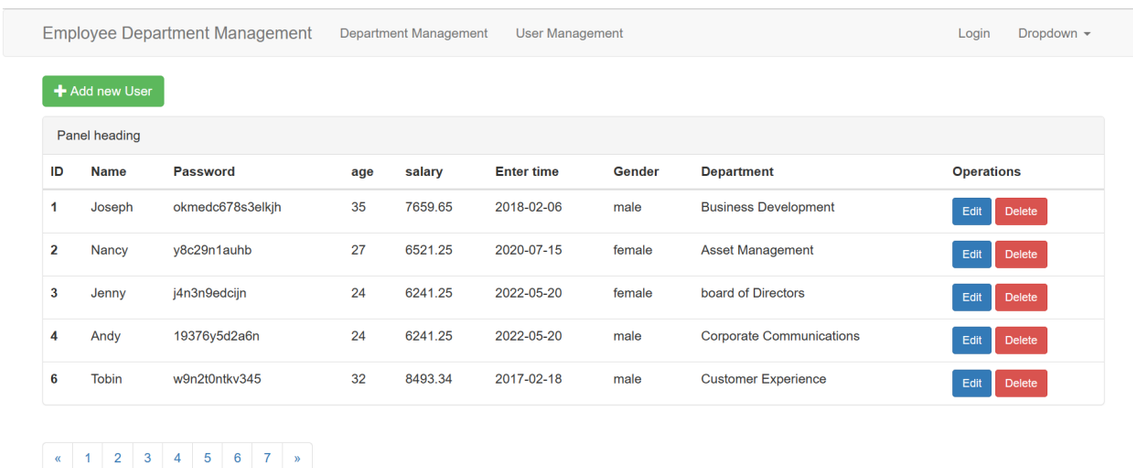
```
# # total pages
total_page = math.ceil(total_count / page_size)
```

```
def user_list(request):
    # get total number of record
    total_count = models.UserInfo.objects.all().count()

    # page, default value=1
    page = int(request.GET.get('page', 1))
    page_size = 5
    start = (page-1)*page_size
    end = page*page_size

    # total pages
    total_page = math.ceil(total_count / page_size)

    queryset = models.UserInfo.objects.all()[start:end]

    page_str_list = []
    for i in range(1, total_page+1):
        ele = '<li><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
        page_str_list.append(ele)
    page_string = mark_safe("".join(page_str_list))

    return render(request, 'user_list.html', {'queryset': queryset, 'page_string': page_string})
```



If there are many pages and we do not want to show all of them

add more info into UserInfo table:

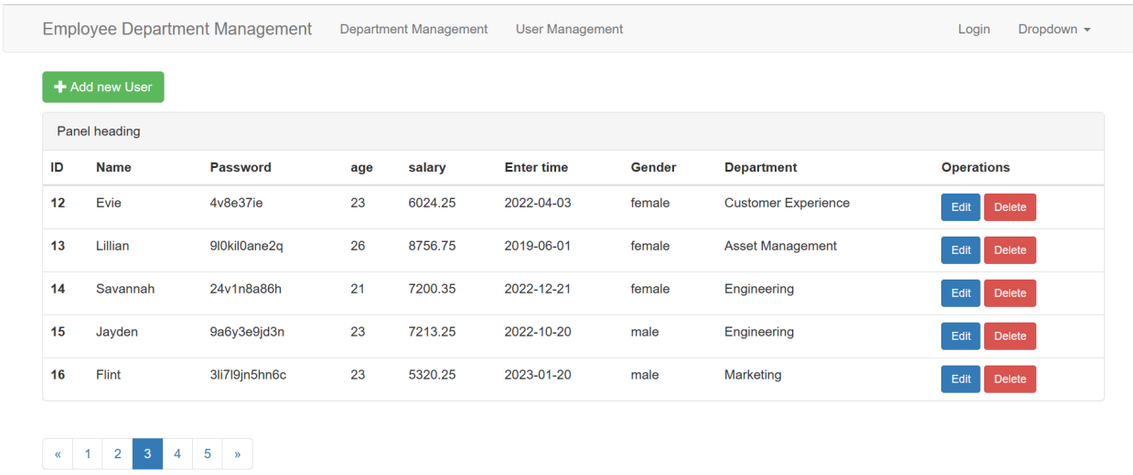


idea: only show 5 pages [page-2:page+2]

```
def user_list(request):
    # get total number of record
    total_count = models.UserInfo.objects.all().count()

    # page, default value=1
    page = int(request.GET.get('page', 1))
    page_size = 5
    start = (page-1)*page_size
    end = page*page_size

    # total pages
    total_page = math.ceil(total_count / page_size)

    queryset = models.UserInfo.objects.all()[start:end]

    # only show 5 pages
    start_page = page-2
    end_page = page+2

    queryset = models.UserInfo.objects.all()[start:end]

    page_str_list = []
    for i in range(start_page, end_page+1):
        ele = '<li><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
        page_str_list.append(ele)
    page_string = mark_safe("".join(page_str_list))

    return render(request, 'user_list.html', {'queryset': queryset, 'page_string': page_string})
```

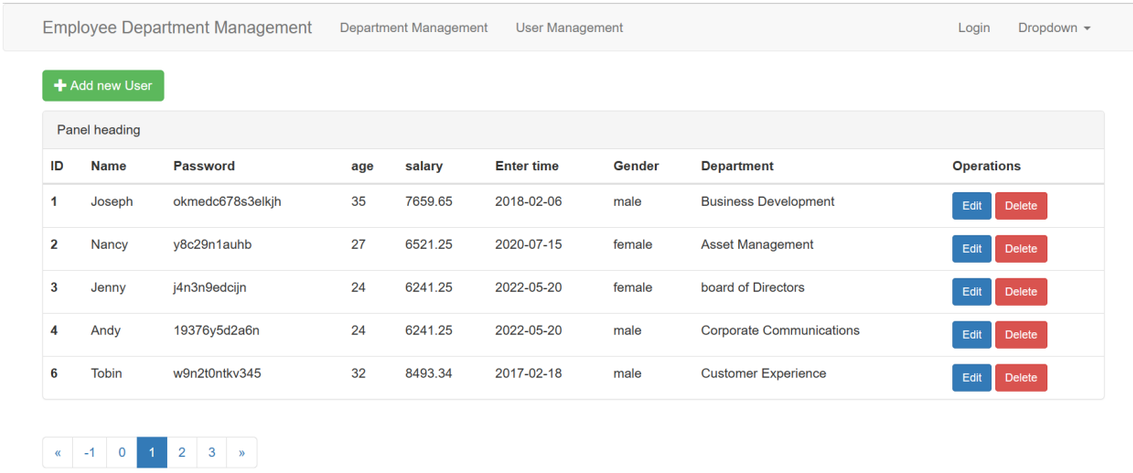
show page number

```
class='active'
```

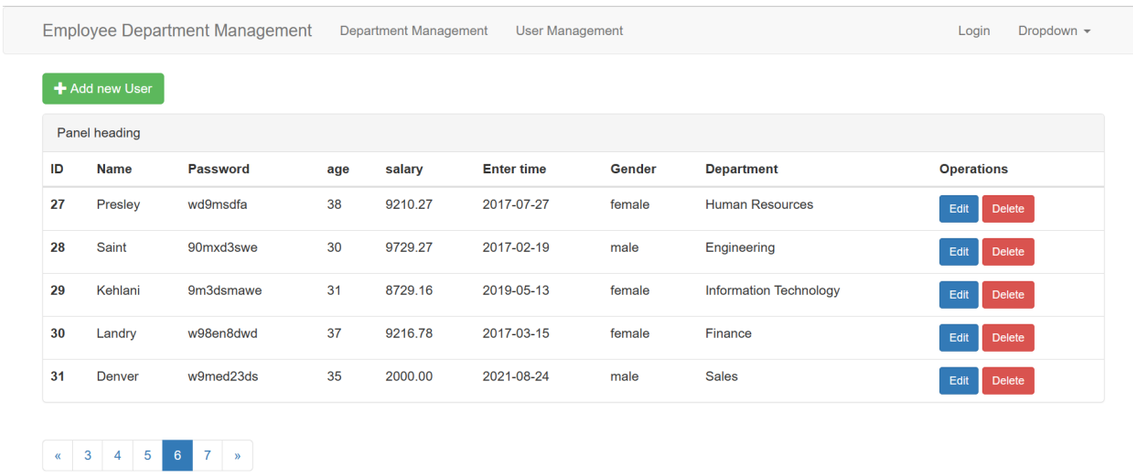


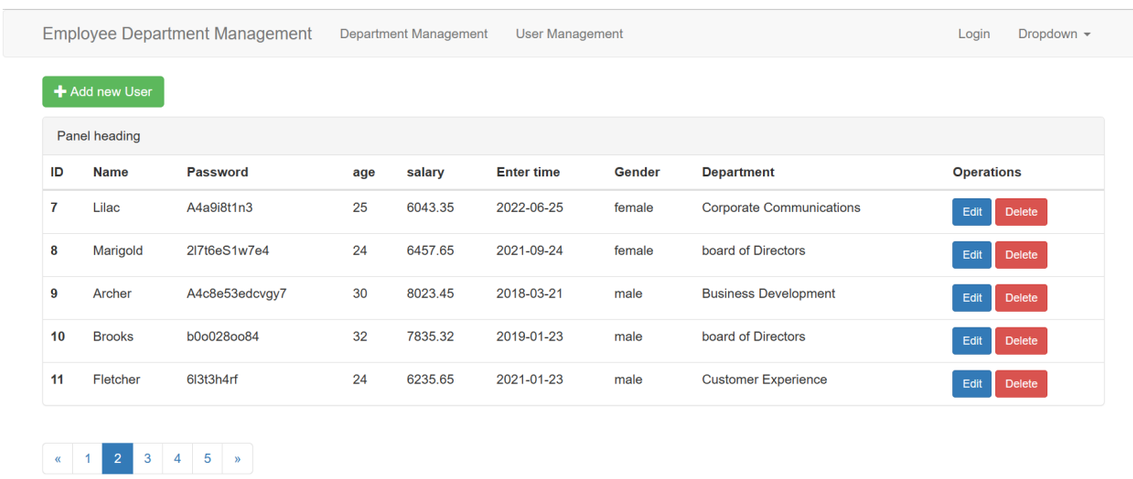

```
def depart_edit(request, nid):
    if request.method == 'GET':
        row_obj = models.Department.objects.filter(id=nid).first()
        return render(request, 'depart_edit.html', {"row_obj": row_obj})

    depart = request.POST.get('Depart')
    models.Department.objects.filter(id=nid).update(department=depart)
    return redirect('/depart/list/')

def user_list(request):
    # get total number of record
    total_count = models.UserInfo.objects.all().count()

    # page, default value=1
    page = int(request.GET.get('page', 1))
    page_size = 5
    start = (page-1)*page_size
    end = page*page_size

    # total pages
    total_page = math.ceil(total_count / page_size)

    # only show 5 pages
    start_page = page-2
    end_page = page+2

    queryset = models.UserInfo.objects.all()[start:end]

    page_str_list = []
    for i in range(start_page, end_page+1):
        if i == page:
            ele = '<li class="active"><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
        else:
            ele = '<li><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
        page_str_list.append(ele)
    page_string = mark_safe("".join(page_str_list))

    return render(request, 'user_list.html', {'queryset': queryset, 'page_string': page_string})
```

However, this also cause problems, when you are at page 1, it will show page 0 and page -1, which is quite nonsence.




```
def user_list(request):
    # get total number of record
    total_count = models.UserInfo.objects.all().count()

    # page, default value=1
    page = int(request.GET.get('page', 1))
    page_size = 5
    start = (page-1)*page_size
    end = page*page_size

    # total pages
    total_page = math.ceil(total_count / page_size)

    plus = 2
    if total_page <= 2 * plus + 1:
        start_page = 1
        end_page = total_page
    else:
        if page <= plus:
            start_page = 1
            end_page = 2 * plus +1
        elif page <= total_page - plus:
            start_page = page - plus
            end_page = page + plus
        else:
            start_page = total_page - 2*plus
            end_page = total_page

    queryset = models.UserInfo.objects.all()[start:end]

    page_str_list = []
    for i in range(start_page, end_page+1):
        if i == page:
            ele = '<li class="active"><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
        else:
            ele = '<li><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
        page_str_list.append(ele)
    page_string = mark_safe("".join(page_str_list))

    return render(request, 'user_list.html', {'queryset': queryset, 'page_string': page_string})
```

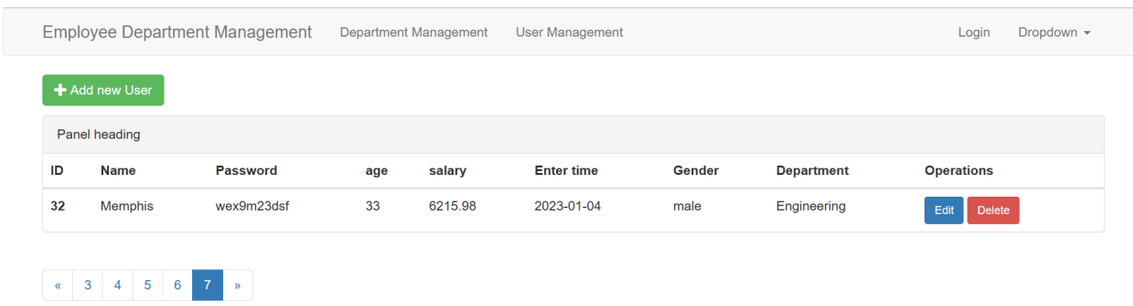




similarly:

```
def user_list(request):
    # get total number of record
    total_count = models.UserInfo.objects.all().count()

    # page, default value=1
    page = int(request.GET.get('page', 1))
    page_size = 5
    start = (page-1)*page_size
    end = page*page_size

    # total pages
    total_page = math.ceil(total_count / page_size)

    plus = 2
    if total_page <= 2 * plus + 1:
        start_page = 1
        end_page = total_page
    else:
        if page <= plus:
            start_page = 1
            end_page = 2 * plus +1
        elif page <= total_page - plus:
            start_page = page - plus
            end_page = page + plus
        else:
            start_page = total_page - 2*plus
            end_page = total_page

    queryset = models.UserInfo.objects.all()[start:end]
    # for obj in queryset:
    #     print(obj.id, obj.name, obj.salary, obj.age)
    #     print(obj.create_time, obj.create_time.strftime('%Y-%m-%d'))
    #     print(obj.gender, obj.get_gender_display())
    #     print(obj.depart_id, obj.depart.department)

    page_str_list = []
```

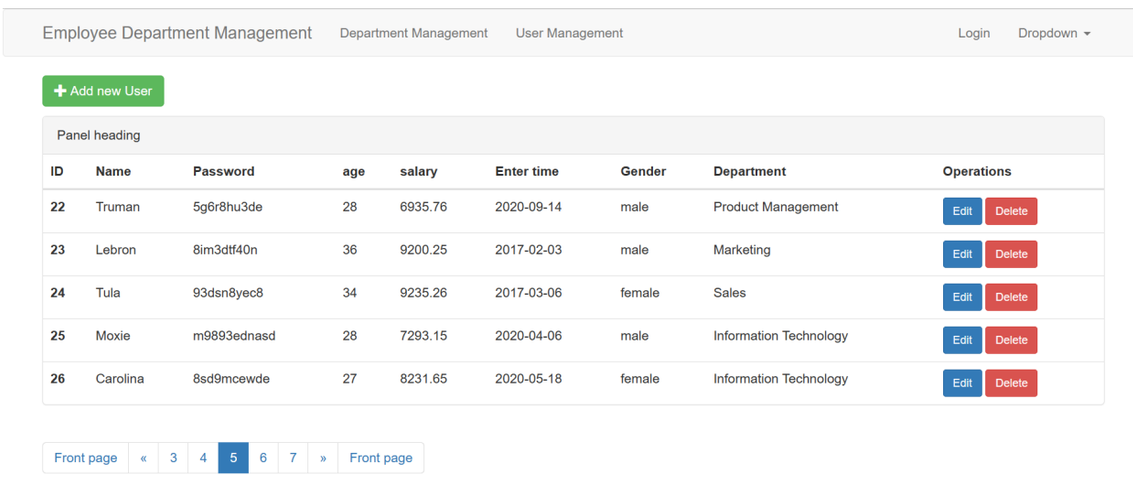
```
    # add previous page
    if page > 1:
        prev = '<li><a href="/user/list/?page={}">&laquo;</a></li>'.format(page-1)
    else:
        prev = '<li><a href="/user/list/?page={}">&laquo;</a></li>'.format(1)
    page_str_list.append(prev)
    for i in range(start_page, end_page+1):
        if i == page:
            ele = '<li class="active"><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
        else:
            ele = '<li><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
        page_str_list.append(ele)

    # add next page
    if page < total_page:
        prev = '<li><a href="/user/list/?page={}">&raquo;</a></li>'.format(page+1)
    else:
        prev = '<li><a href="/user/list/?page={}">&raquo;</a></li>'.format(total_page)
    page_str_list.append(prev)

    page_string = mark_safe("".join(page_str_list))

    return render(request, 'user_list.html', {'queryset': queryset, 'page_string': page_string})
```



add front and end page

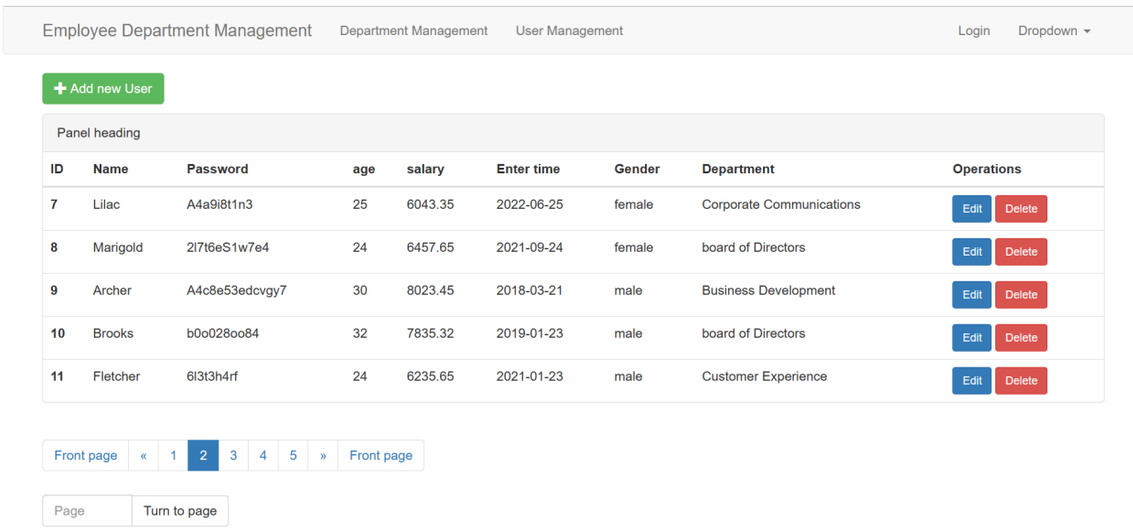
```
def user_list(request):
    # get total number of record
    total_count = models.UserInfo.objects.all().count()

    # page, default value=1
    page = int(request.GET.get('page', 1))
    page_size = 5
    start = (page-1)*page_size
    end = page*page_size

    # total pages
    total_page = math.ceil(total_count / page_size)

    plus = 2
    if total_page <= 2 * plus + 1:
        start_page = 1
        end_page = total_page
    else:
        if page <= plus:
            start_page = 1
            end_page = 2 * plus +1
        elif page <= total_page - plus:
            start_page = page - plus
            end_page = page + plus
        else:
            start_page = total_page - 2*plus
            end_page = total_page

    queryset = models.UserInfo.objects.all()[start:end]

    page_str_list = []
    # add first page
    front = '<li><a href="/user/list/?page={}">Front page</a></li>'.format(1)
    page_str_list.append(front)
    # add previous page
    if page > 1:
```

```
        prev = '<li><a href="/user/list/?page={}">&laquo;</a></li>'.format(page-1)
    else:
        prev = '<li><a href="/user/list/?page={}">&laquo;</a></li>'.format(1)
    page_str_list.append(prev)
    for i in range(start_page, end_page+1):
        if i == page:
            ele = '<li class="active"><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
        else:
            ele = '<li><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
        page_str_list.append(ele)

    # add next page
    if page < total_page:
        prev = '<li><a href="/user/list/?page={}">&raquo;</a></li>'.format(page+1)
    else:
        prev = '<li><a href="/user/list/?page={}">&raquo;</a></li>'.format(total_page)
    page_str_list.append(prev)

    # add last page
    last = '<li><a href="/user/list/?page={}">Front page</a></li>'.format(total_page)
    page_str_list.append(last)

    page_string = mark_safe("".join(page_str_list))

    return render(request, 'user_list.html', {'queryset': queryset, 'page_string': page_string})
```



add turn to page

```
<form method="get">
            <div class="input-group" style="width: 200px">
                <input type="text" name="page" class="form-control" placeholder="Page">

                <span class="input-group-btn">
                <button type="submit" class="btn btn-default">Turn to page</button>
                </span>
            </div>
        </form>
```



wrapper Class

```
import math
from django.utils.safestring import mark_safe


class Pagination(object):
    def __init__(self, request, queryset, page_size=5, page_param='page'):
        page = request.GET.get(page_param, '1')
        if page.isdecimal():
            page = int(page)
        else:
            page = 1
        self.page = page
        self.plus = int(page_size / 2)
        self.page_size = page_size

        self.start = (page-1) * page_size
        self.end = page * page_size

        self.page_queryset = queryset[self.start:self.end]

        total_count = queryset.count()
        total_page_count = math.ceil(total_count / page_size)

        self.total_page = total_page_count
        print(type(self.plus))


    def html(self):
        if self.total_page <= 2 * self.plus + 1:
            start_page = 1
            end_page = self.total_page
        else:
            if self.page <= self.plus:
                start_page = 1
                end_page = 2 * self.plus + 1
            elif self.page <= self.total_page - self.plus:
                start_page = self.page - self.plus
                end_page = self.page + self.plus
            else:
                start_page = self.total_page - 2 * self.plus
                end_page = self.total_page
        self.start_page = start_page
        self.end_page = end_page
        page_str_list = []
        # add first page
        front = '<li><a href="/user/list/?page={}">Front page</a></li>'.format(1)
        page_str_list.append(front)
        # add previous page
        if self.page > 1:
            prev = '<li><a href="/user/list/?page={}">&laquo;</a></li>'.format(self.page - 1)
        else:
            prev = '<li><a href="/user/list/?page={}">&laquo;</a></li>'.format(1)
        page_str_list.append(prev)
        for i in range(self.start_page, self.end_page + 1):
            if i == self.page:
                ele = '<li class="active"><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
            else:
```

```
                ele = '<li><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
            page_str_list.append(ele)

        # add next page
        if self.page < self.total_page:
            prev = '<li><a href="/user/list/?page={}">&raquo;</a></li>'.format(self.page + 1)
        else:
            prev = '<li><a href="/user/list/?page={}">&raquo;</a></li>'.format(self.total_page)
        page_str_list.append(prev)

        # add last page
        last = '<li><a href="/user/list/?page={}">Front page</a></li>'.format(self.total_page)
        page_str_list.append(last)

        page_string = mark_safe("".join(page_str_list))
        return page_string
```

```
from app01.utils.pageination import Pagination
def user_list(request):
    queryset = models.UserInfo.objects.all().order_by('-age')
    page_object = Pagination(request, queryset)
    page_queryset = page_object.page_queryset
    page_string = page_object.html()

    context = {
        'queryset': page_queryset,
        'page_string': page_string
    }

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

add prev, front, last, end:

```
import math
from django.utils.safestring import mark_safe


class Pagination(object):
    """
    :param

    """
    def __init__(self, request, queryset, page_size=5, page_param="page"):
        page = request.GET.get(page_param, '1')
        if page.isdecimal():
            page = int(page)
        else:
            page = 1
        self.page = page
        self.plus = int(page_size / 2)
        self.page_size = page_size

        self.start = (page-1) * page_size
        self.end = page * page_size

        self.page_queryset = queryset[self.start:self.end]

        total_count = queryset.count()
        total_page_count = math.ceil(total_count / page_size)

        self.total_page = total_page_count
        print(type(self.plus))


    def html(self):
        if self.total_page <= 2 * self.plus + 1:
            start_page = 1
            end_page = self.total_page
        else:
            if self.page <= self.plus:
                start_page = 1
                end_page = 2 * self.plus + 1
            elif self.page <= self.total_page - self.plus:
                start_page = self.page - self.plus
                end_page = self.page + self.plus
            else:
                start_page = self.total_page - 2 * self.plus
                end_page = self.total_page
        self.start_page = start_page
        self.end_page = end_page
        page_str_list = []
        # add first page
        front = '<li><a href="/user/list/?page={}">Front page</a></li>'.format(1)
        page_str_list.append(front)
        # add previous page
        if self.page > 1:
            prev = '<li><a href="/user/list/?page={}">&laquo;</a></li>'.format(self.page - 1)
```

```
        else:
            prev = '<li><a href="/user/list/?page={}">&laquo;</a></li>'.format(1)
        page_str_list.append(prev)
        for i in range(self.start_page, self.end_page + 1):
            if i == self.page:
                ele = '<li class="active"><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
            else:
                ele = '<li><a href="/user/list/?page={}">{}</a></li>'.format(i, i)
            page_str_list.append(ele)

        # add next page
        if self.page < self.total_page:
            prev = '<li><a href="/user/list/?page={}">&raquo;</a></li>'.format(self.page + 1)
        else:
            prev = '<li><a href="/user/list/?page={}">&raquo;</a></li>'.format(self.total_page)
        page_str_list.append(prev)

        # add last page
        last = '<li><a href="/user/list/?page={}">Front page</a></li>'.format(self.total_page)
        page_str_list.append(last)

        page_str_list.append("""
        <li>
                    <form method="get">

                        <input type="text" name="page" class="form-control" placeholder="Page"
                               style="width: 100px; display:inline-block; float: left;">

                        <span class="input-group-btn">
                        <button type="submit" class="btn btn-default" style="border-radius: 5px">Turn to page</button>
                        </span>

                    </form>
                </li>
        """)

        page_string = mark_safe("".join(page_str_list))
        return page_string
```

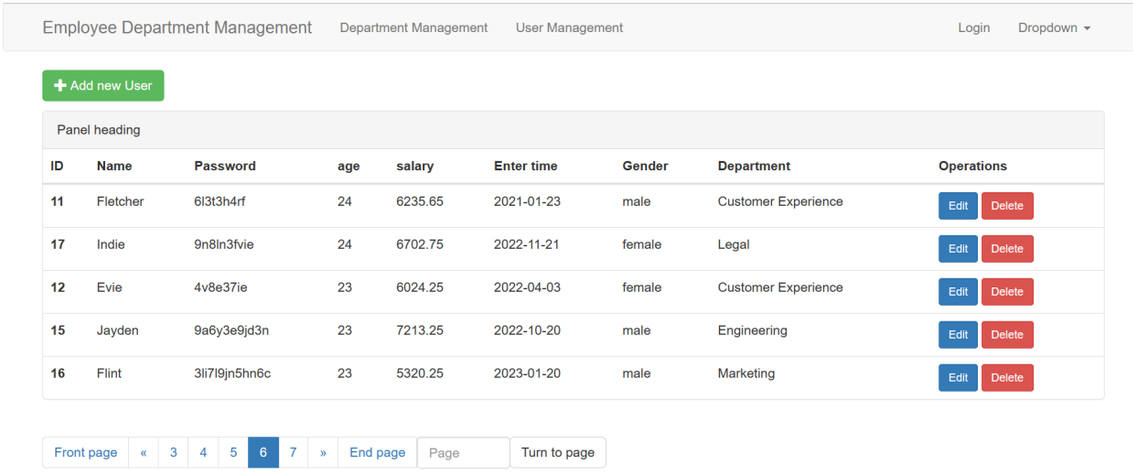
if we get multiple value using GET method, we might lost other values while turning the page.

```
request.GET.urlencode()
# concat all values from urls
# but default not allowed changes
```

```
from django.http.request import QueryDict
import copy
query_dict = copy.deepcopy(request.GET)
# allowed changes
query_dict._mutable = True
self.query_dict = query_dict
query_dict.setlist('page', [11])
# concat
query_dict.urlencode()
```

```
import math
from django.utils.safestring import mark_safe
from django.http.request import QueryDict
import copy


class Pagination(object):
    """
    :param

    """
    def __init__(self, request, queryset, page_size=5, page_param="page"):
        page = request.GET.get(page_param, '1')
        if page.isdecimal():
            page = int(page)
        else:
            page = 1
        query_dict = copy.deepcopy(request.GET)
        # allowed changes
        query_dict._mutable = True
        self.query_dict = query_dict

        self.page = page
        self.page_param = page_param
        self.plus = int(page_size / 2)
        self.page_size = page_size

        self.start = (page-1) * page_size
        self.end = page * page_size
```

```
        self.page_queryset = queryset[self.start:self.end]

        total_count = queryset.count()
        total_page_count = math.ceil(total_count / page_size)

        self.total_page = total_page_count
        print(type(self.plus))


    def html(self):
        if self.total_page <= 2 * self.plus + 1:
            start_page = 1
            end_page = self.total_page
        else:
            if self.page <= self.plus:
                start_page = 1
                end_page = 2 * self.plus + 1
            elif self.page <= self.total_page - self.plus:
                start_page = self.page - self.plus
                end_page = self.page + self.plus
            else:
                start_page = self.total_page - 2 * self.plus
                end_page = self.total_page
        self.start_page = start_page
        self.end_page = end_page

        self.query_dict.setlist(self.page_param, [1])

        page_str_list = []
        # add first page
        front = '<li><a href="?page={}">Front page</a></li>'.format(self.query_dict.urlencode())
        page_str_list.append(front)
        # add previous page
        if self.page > 1:
            self.query_dict.setlist(self.page_param, [self.page - 1])
            prev = '<li><a href="?{}">&laquo;</a></li>'.format(self.query_dict.urlencode())
        else:
            self.query_dict.setlist(self.page_param, [1])
            prev = '<li><a href="?{}">&laquo;</a></li>'.format(self.query_dict.urlencode())
        page_str_list.append(prev)
        for i in range(self.start_page, self.end_page + 1):
            if i == self.page:
                self.query_dict.setlist(self.page_param, [i])
                ele = '<li class="active"><a href="?{}">{}</a></li>'.format(self.query_dict.urlencode(), i)
            else:
                self.query_dict.setlist(self.page_param, [i])
                ele = '<li><a href="?{}">{}</a></li>'.format(self.query_dict.urlencode(), i)
            page_str_list.append(ele)

        # add next page
        if self.page < self.total_page:
            self.query_dict.setlist(self.page_param, [self.page + 1])
            prev = '<li><a href="?{}">&raquo;</a></li>'.format(self.query_dict.urlencode())
        else:
            self.query_dict.setlist(self.page_param, [self.total_page])
            prev = '<li><a href="?{}">&raquo;</a></li>'.format(self.query_dict.urlencode())
        page_str_list.append(prev)

        # add last page
        self.query_dict.setlist(self.page_param, [self.total_page])
        last = '<li><a href="?{}">End page</a></li>'.format(self.query_dict.urlencode())
        page_str_list.append(last)

        page_str_list.append("""
        <li>
                    <form method="get">

                        <input type="text" name="page" class="form-control" placeholder="Page"
                               style="width: 100px; display:inline-block; float: left;">

                        <span class="input-group-btn">
                        <button type="submit" class="btn btn-default" style="border-radius: 5px">Turn to page</button>
                        </span>

                    </form>
                </li>
        """)

        page_string = mark_safe("".join(page_str_list))
        return page_string
```



But this method will not make sence for search page

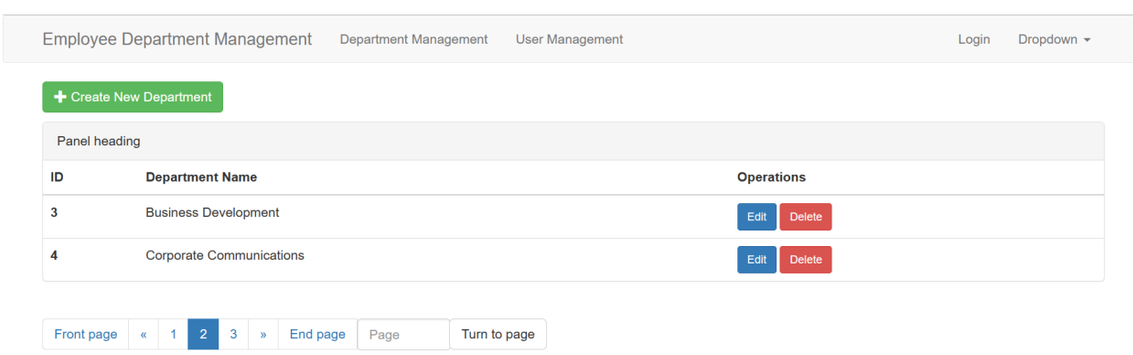
4. Add paging to Department list

1. add page bars to depart_list.html

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

2. add class in function in views.py

```
def depart_list(request):
    # # get data from dataset
    # queryset = models.Department.objects.all()
    #
    # return render(request, 'depart_list.html', {'queryset': queryset})
    queryset = models.Department.objects.all()
    page_object = Pagination(request, queryset, page_size=2)
    page_queryset = page_object.page_queryset
    page_string = page_object.html()

    context = {
        'queryset': page_queryset,
        'page_string': page_string
    }
    return render(request, 'depart_list.html', context)
```



Please do remember to change

1. queryset (you may need to order by or not, but do remember to change valid col name)
2. table name

5. Time (JS)

ModelForm

```
class UserModelForm(forms.ModelForm):
    password = forms.CharField(min_length=8, label='password')
    class Meta:
        model = models.UserInfo
        fields = ['name', 'password', 'age', 'salary', 'create_time', 'gender', 'depart']

        widgets = {"create_time": forms.TextInput(attrs={"type": "date"})}
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        for name, field in self.fields.items():
            field.widget.attrs = {"class": "form-control", "placeholder": field.label}
```

jQuery

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="{% static "plugins/bootstrap-3.4.1-dist/css/bootstrap.css" %}">
    {% block css %}{% endblock %}
</head>
<body>
<nav class="navbar navbar-default">
    <div class="container">
        <!-- Brand and toggle get grouped for better mobile display -->
        <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
                    data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">Employee Department Management</a>
        </div>

        <!-- Collect the nav links, forms, and other content for toggling -->
        <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
            <ul class="nav navbar-nav">
                {#                <li class="active"><a href="#">Link <span class="sr-only">(current)</span></a></li>#}
```

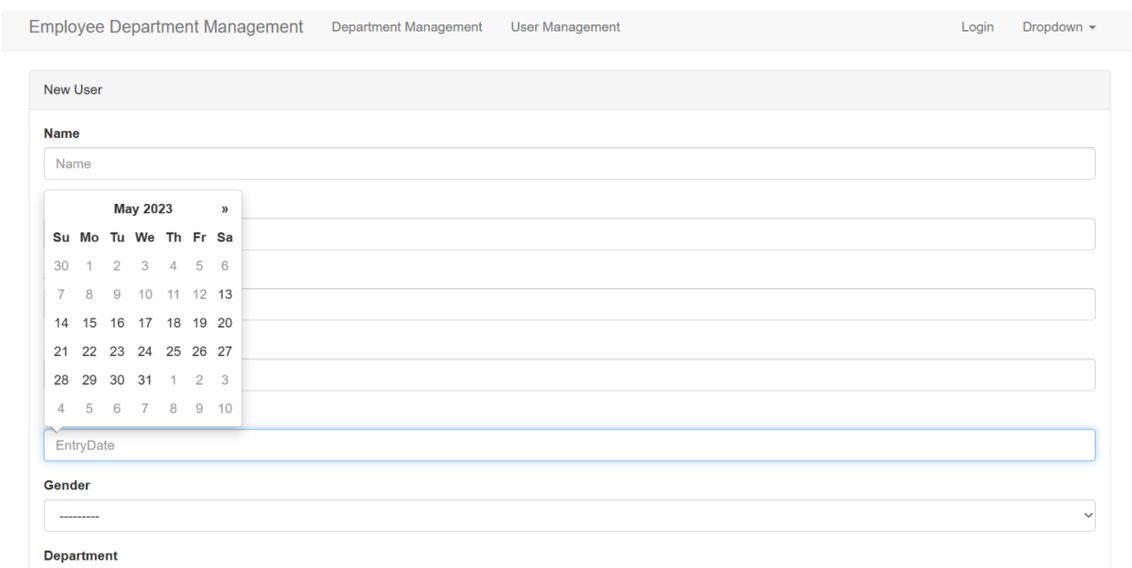
```
                <li><a href="/depart/list/">Department Management</a></li>
                <li><a href="/user/list/">User Management</a></li>

            </ul>

            <ul class="nav navbar-nav navbar-right">
                <li><a href="#">Login</a></li>
                <li class="dropdown">
                    <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true"
                       aria-expanded="false">Dropdown <span class="caret"></span></a>
                    <ul class="dropdown-menu">
                        <li><a href="#">Action</a></li>
                        <li><a href="#">Another action</a></li>
                        <li><a href="#">Something else here</a></li>
                        <li role="separator" class="divider"></li>
                        <li><a href="#">Separated link</a></li>
                    </ul>
                </li>
            </ul>
        </div><!-- /.navbar-collapse -->
    </div><!-- /.container-fluid -->
</nav>

<div>
    {% block content %}{% endblock %}
</div>

<script src="{% static "js/jquery-3.6.4.min.js" %}"></script>
<script src="{% static "plugins/bootstrap-3.4.1-dist/js/bootstrap.min.js" %}"></script>
{% block js %}{% endblock %}
</body>
</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"> New User</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 %}
```

💡 {% extends 'layout.html' %} must shown in the first line

```
{% 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"> Edit User</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">Update</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 %}
```



6. Bootstrap style class inheritance

```
# bootstrap.py
class BootStrapModelForm(forms.ModelForm):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        for name, field in self.fields.items():
            if field.widget.attrs:
                field.widget.attrs['class'] = 'form-control'
                field.widget.attrs['placeholder'] = field.label
            else:
                field.widget.attrs = {"class": "form-control",
                                      "placeholder": field.label}
```

```
# views.py
from app01.utils.bootstrap import BootStrapModelForm

class UserModelForm(BootStrapModelForm):
    password = forms.CharField(min_length=8, label='password')
    class Meta:
```

```
        model = models.UserInfo
        fields = ['name', 'password', 'age', 'salary', 'create_time', 'gender', 'depart']
```

UserModelForm inherient from BootStrapModelForm