

# ساخت سایت و برنامه‌ی مدیریت پسورد با قابلیت لاگین به کمک تشخیص چهره

نوید حسن زاده، یاسمون صفائیان، سروش دهقان

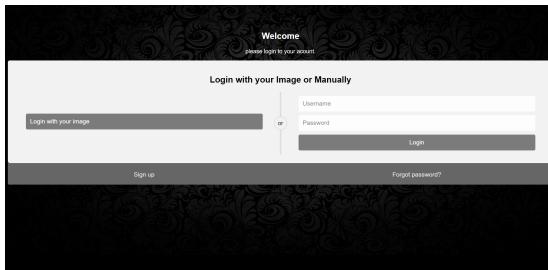
۱۳۹۷ تیر ۲۶

## ۱ مقدمه

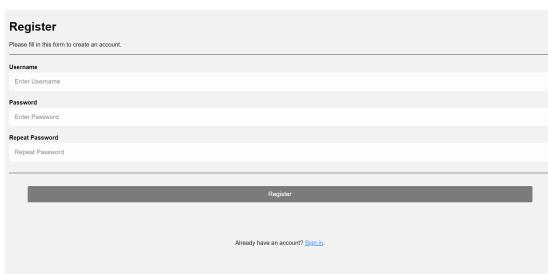
هدف این پروژه طراحی صفحه‌ی وبی است که کاربرانش بتوانند تنها به وسیله دوربین لپ تاپ و تصویر خود وارد سایت شوند به عبارت دیگر بتواند کاربران مجاز را با تشخیص چهره آنها مشخص کند. پروژه در زمینه پردازش تصاویر می‌باشد.



شکل ۱: ورود با استفاده از تصویر.



شکل ۲: صفحه login طراحی شده.



شکل ۳: صفحه sign up طراحی شده.

## ۲ روند پروژه

### ۱.۲ گام اول : طراحی صفحات وب

برای این پروژه صفات وب مورد نیاز به کمک زبان html نوشته شده و این صفحات با Django به هم مربوط شده اند.

صفحه login که در **شکل ۲** مشاهده می شود اولین صفحه ای که کاربر با آن رویه رو می شود می باشد. در این صفحه فرد با زدن نام کاربری و رمز عبور در صورتی که قبل از این عضو شده باشد وارد سایت می شود. همین طور با فشردن کلید image your with login وارد صفحه ای می شود که دوربین لپ تاپ را روشن کرده و با گرفتن عکس فرد او را لاگین می کند. در صورتی که فرد قبل از این عضو شده باشد با فشردن کلید sign up وارد صفحه ای برای ورود می شود.

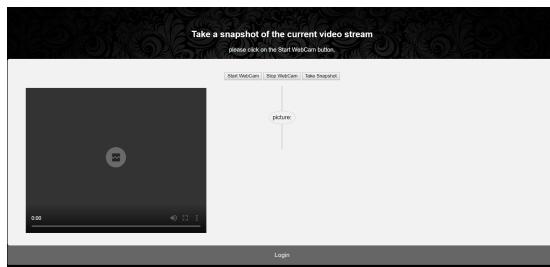
همانطور که اشاره شد صفحه sign up که در **شکل ۳** نشان داده شده است که در این صفحه کاربر نام کاربری و کلمه عبور خود را وارد می کند و در صورتی که کلمه عبور و تکرار آن یکسان باشند با زدن کلید register وارد صفحه اصلی سایت می شود.

صفحه image your with login نیز در **شکل ۴** قابل مشاهده است.

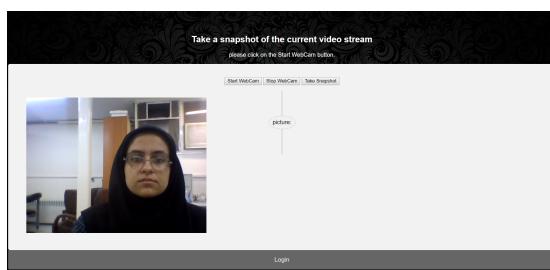
در این صفحه با زدن کلید WebCam start دوربین لپ تاپ روشن شده و تصویر فرد همانند **شکل ۵**

## ۱.۲ گام اول: طراحی صفحات وب

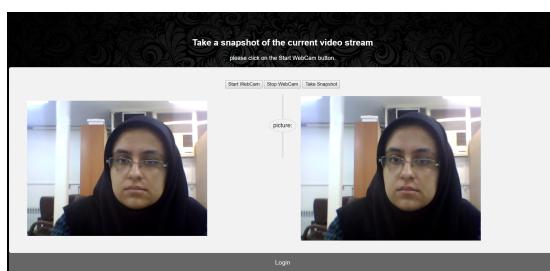
۲ روند پروژه



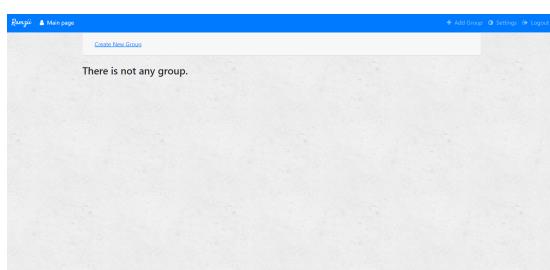
شکل ۴: صفحه طراحی شده برای ورود با تصویر.



شکل ۵: تصویر نشان داده شده از فرد.



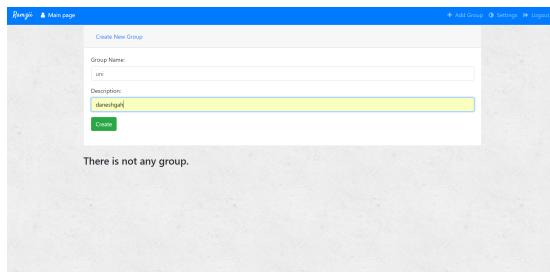
شکل ۶: تصویر ذخیره شده از فرد.



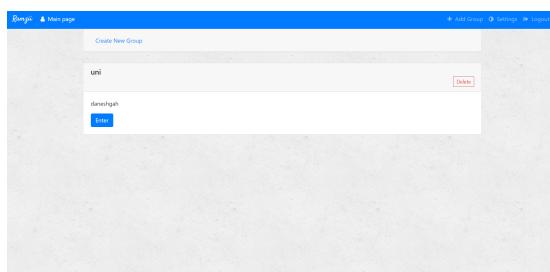
شکل ۷: صفحه اصلی وب سایت.

## ۱.۲ گام اول: طراحی صفحات وب

۲ روند پروژه



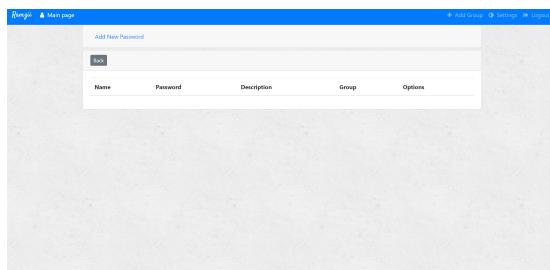
شکل ۸: صفحه اصلی وب سایت.



شکل ۹: اضافه شدن گروهی برای پسورد ها.

روبه روی دوربین نمایش داده می شود.

با زدن کلید Snapshot Take عکس گرفته می شود و ذخیره می شود. **شکل ۶** با زدن کلید login در صورتی که کاربر قبلًا عضو باشد وارد صفحه اصلی می شود. صفحه اصلی در **شکل ۷** نشان داده شده است. با ورود کاربر جدید صفحه خالی است و گروه و پسوردی درست نشده است. با کلیک بر روی group new create صفحه **شکل ۸** باز میشود. با وارد کردن نام و توضیح در مورد گروه و کلیک بر روی create گروه ساخته شده **شکل ۹** و ما دوباره در صفحه اصلی خواهیم بود. اگر روی enter کلیک کنیم وارد صفحه مربوط به هر گروه شده **شکل ۱۰** و میتوانیم پسورد هایی



شکل ۱۰: صفحه نشان دهنده محتوا گروه ها.

A screenshot of a web-based password management application. The interface includes a header with navigation links like 'Design', 'Main page', 'Add Group', 'Settings', and 'Logout'. Below the header is a form titled 'Add New Password' with fields for 'Name' (containing 'saeid'), 'Password' (containing '123'), and 'Description' (containing 'taghize'). At the bottom of the form is a green 'Add' button. To the right of the form is a table titled 'List' showing a single row with the data from the form.

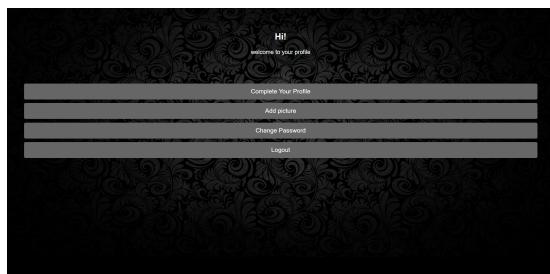
شکل ۱۱: اضافه کردن پسورد به گروه ها.

A screenshot of the same web application after the password has been added. The 'List' table now displays a single row with the following data: Name: 'saeid', Password: '123', Description: 'taghize', Group: 'gr1', and Options: an 'edit' button.

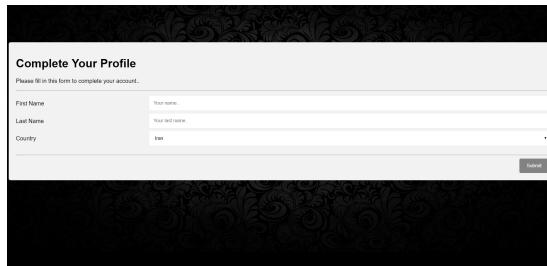
شکل ۱۲: پسورد اضافه شده به گروه ها.

که مربوط به هر گروه هستند را مشاهده کنیم. با کلیک روی **delete** گروه ایجاد شده پاک میشود. با کلیک بر روی **setting** در نوار بالای صفحه صفحه تنظیمات باز میشود. اگر گروه تازه ایجاد شده باشد پسوردی در آن نیست.

با کلیک بر روی **password new add** صفحه **شکل ۱۱** باز میشود. با وارد کردن نام، رمز عبور و توضیح و با زدن کلید **add** پسورد جدید به صفحه گروه **شکل ۱۲** اضافه می شود. با کلیک بر روی **delete** پسورد ایجاد شده پاک می شود و با کلیک بر روی **back** وارد صفحه اول می شویم. صفحه **setting** در **شکل ۱۳** نشان داده شده است. در این صفحه با کلیک بر روی **profile your complete** وارد صفحه **شکل ۱۴** می شویم.



شکل ۱۳: اضافه کردن پسورد به گروه ها.



شکل ۱۴: پسورد اضافه شده به گروه ها.



شکل ۱۵: اضافه کردن پسورد به گروه ها.

شوم. با وارد کردن اطلاعات و کلیک بر روی **submit** وارد صفحه تنظیمات شده و اطلاعات وارد شده ذخیره می شود. با کلیک بر روی **picture add** وارد صفحه **شکل ۱۵** زیر می شود و با گرفتن تصویر و کلیک روی **save** تصویر را ذخیره می کند. با کلیک روی **password change** وارد صفحه **شکل ۱۶** شده و در صورتی که پسورد جدید و تکرار آن یکسان باشد پسورد جدید را ذخیره کرده و با زدن کلید **save** به صفحه تنظیمات بر میگردد. با کلیک بر روی **logout** وارد صفحه ای اول شده و کاربر را از سایت خارج میکند.



شکل ۱۶: پسورد اضافه شده به گروه ها.

## ۲.۲ گام دوم: طراحی بک اند

مدل user برای لایکین شدن در سایت نیاز داریم تا از کاربر username و password را گرفته و لایکین کنیم. به این منظور لازم است تا مدلی برای یوزر درست کنیم تا اطلاعات کاربران در آن ذخیره شود.

```
class User(models.Model):
    group = models.ForeignKey(Group, on_delete=models.CASCADE, default='No Name')
    username = models.CharField(max_length=100)
    password = models.CharField(max_length=1000)
    def __str__(self):
        return 'username=' + self.username + ' password=' + self.password
```

این کلاس دو پارامتر password و username دارد و برای نمایش آن تابع `str` است. مدل group: محتوی صفحه اصلی سایت گروه هایی است که هر کاربر برای خود ایجاد کرده است. برای ذخیره گروه ها مشابه قبل نیاز به مدلی برای group که شامل groupdesc ، groupname است برای ذخیره مشخصات گروه ها. علاوه بر این از آنجایی که هر گروه متعلق به صفحه مربوط به یک یوزر خاص است و گروه های کاربران متفاوت با یکدیگر فرق دارد، باید در مدل group یک foreignkey به گروه اختصاص دهیم که مشخص کند هر گروه متعلق به کدام کاربر است.

```
class Group(models.Model):
    user = models.ForeignKey(user, on_delete=models.CASCADE)
    groupName = models.CharField(max_length=40)
    groupDesc = models.CharField(max_length=1000)
    groupImage = models.CharField(max_length=1000)

    def get_absolute_url(self):
        return reverse('manager:passwords', kwargs={'pk' : self.pk})

    def __str__(self):
        return 'Name=' + self.groupName + ' Description=' + self.groupDesc
```

مدل password: با وارد شدن به صفحه هر گروه تمام پسورد های مربوط به آن گروه نمایش داده میشود. برای ایجاد چنین قابلیتی نیاز داریم تا مدلی برای password ایجاد کنیم که شامل name ، desc باشد. میدانیم که هر پسورد که در صفحه یک گروه ایجاد میشود مربوط به همان گروه است و نباید در صفحه دیگر گروه ها نمایش داده شود. برای دست یافتن به این امکان مشابه مدل گروه، یک foreignkey به گروه group اختصاص دهیم که مشخص کند هر پسورد متعلق به کدام گروه است.

```
class Password(models.Model):
    group = models.ForeignKey(Group, on_delete=models.CASCADE)
    # passID = models.CharField(max_length=20)
    # userName = models.CharField(max_length=20)
    name = models.CharField(max_length=100)
```

```

desc = models.CharField(max_length=1000)
password = models.CharField(max_length=100)
def __str__(self):
    return 'Group' + self.group.groupName + 'Pass. Name=' + self.name
    + ' Description=' + self.desc

url(r'^login/(.)*$', views.login, name='login')

def login(request , i):
    # import pdb;pdb.set_trace()
    inputs = request.get_full_path()
    values = inputs.split('?')[1].split('&')
    username = values[0].split('=')[1]
    passw = values[1].split('=')[1]
    # print(passw)
    # print(user.objects.filter(username=user)[0])
    all_users = user.objects.all()
    print(all_users)
    for users in all_users:
        print(users.username)
        if users.username == username:
            print("yes")
            if users.password == passw:
                print("yesyes")
                r = redirect('manager:index')
                r.set_cookie('user',username)
                r.set_cookie('permission', 'T')
                return r
            else:
                print("no")
                return redirect('index')
        else:
            print("no no")
    return redirect('index')

```

در این حالت نام کاربری و پسورد در درخواست وجود دارد. آنها را از درخواست جدا کرده و در متغیر های `username` و `password` می‌بینیم. بعد از آن نیاز داریم تا چک کنیم که آیا یوزرنیم فرد در بین نام‌های کاربری ذخیره شده در دیتابیس وجود دارد یا خیر. در صورتی که نام کاربری مشابهی پیدا کرد باید چک کند که آیا پسورد وارد شده هم با پسورد مربوط به آن یوزرنیم در دیتابیس مشابه هست یا خیر. در صورتی که یوزرنیم و پسورد درست وارد شده بودند نیاز داریم تا به صفحه اصلی سایت با url به نام `manager::index` هدایت شویم. اما از آنجایی که نیاز داریم تا گروه‌هایی که مربوط به خود آن کاربر هستند را نمایش دهیم

کاربر را به عنوان کوکی همراه درخواست ارسال میکنیم.

علاوه بر آن نیاز داریم تا فقط کاربرانی که login انجام داده اند بتوانند وارد صفحه اصلی سایت بشوند و سایر افراد با وارد کردن آدرس <http://127.0.0.1:8000/manager/> توانند به صفحه سایت دسترسی داشته باشند. برای دست یابی به این هدف، متغیری به عنوان permission را نیز همراه با درخواست ورود در کوکی T تنظیم میکنیم. برای هدایت به صفحه اصلی سایت، url و سپس تابع زیر فراخوانی میشود.

```
url(r'^$', main_views.index, name='index')
```

```
def index(request):
    #import pdb;pdb.set_trace()
    if (request.COOKIES.get('permission') != 'T'):
        return redirect('index')
    else:

        userObj =
        get_object_or_404(user, pk=user.objects.filter(username=request.COOKIES.get('user'))))
        all_groups = userObj.group_set.all()
        # template = loader.get_template('manager/index.html')
        context = {'all_groups': all_groups, 'userid':2}
        r = render(request, 'index_manage.html', context)

        html = ''
        # return HttpResponse(template.render(context, request))
    return r
```

این تابع ابتدا چک میکند آیا کوکی permission T هست یا نه، اگر نبود دوباره ما را به صفحه لاگین هدایت میکند، اما اگر T بود یعنی کاربری لاگین کرده بود، به کمک کد زیر ابتدا object یوزر را از روی کوکی یوزرنیم که همراه درخواست لاگین ارسال شده است تشخیص میدهد و سپس تمام گروه های مربوط به آن یوزر را در صفحه نمایش میدهد.

```
userObj=get_object_or_404(user,pk=user.objects.filter(username=request.COOKIES.get('user')))
all_groups = userObj.group_set.all()
```

خروج از سایت: برای خروج کاربر با کلیک بر روی لینک مورد نظر، تابع و url زیر فراخوانی میشود.

```
url(r'^logout$', views.logout, name='logout')
```

```
def logout(request):
    r = redirect('index')
    r.delete_cookie('user')
    r.delete_cookie('name')
    r.set_cookie('permission', 'F')
    return r
```

این تابع تمام کوکی هایی که هنگام ورود به سایت مشخص شده بودند را پاک میکند و کوکی `permis` را نیز `F` میکند تا دیگر کسی با تایپ `url` مربوط به صفحه اصلی نتواند وارد آن شود و سپس ما را به صفحه `login` هدایت میکند.

ایجاد یک کاربر جدید: برای ایجاد یک کاربر جدید `url` و تابع زیر فراخوانی میشود.

```
url(r'^sign_up/(.)*$', views.sign_up, name='sign_up')
```

```
def sign_up(request, i):
    # import pdb;pdb.set_trace()
    print(request)
    inputs = request.get_full_path()
    values = inputs.split('?')[1].split('&')
    all_users = user.objects.all()
    if all_users:
        for users in all_users:
            if users.username != values[0].split('=')[1]:
                if values[1].split('=')[1] == values[2].split('=')[1] :
                    newuser = user()
                    newuser.username = values[0].split('=')[1]
                    newuser.password = values[1].split('=')[1]
                    newuser.save()
                    r = redirect('manager:index')
                    r.set_cookie('name', newuser.username)
                    r.set_cookie('permission', 'T')
                    return r
            else:
                return redirect('register:register_new')
        else:
            return redirect('register:register_new')
    else:
        if values[1].split('=')[1] == values[2].split('=')[1]:
            newuser = user()
            newuser.username = values[0].split('=')[1]
            newuser.password = values[1].split('=')[1]
            newuser.save()
            r = redirect('manager:index')
            r.set_cookie('name', newuser.username)
            r.set_cookie('permission', 'T')
            return r
```

درخواستی که برای این تابع ارسال میشود شامل پوزنیم، پسورد و تکرار پسورد است. ابتدا نیاز داریم تا پوزنیم، پسورد و تکرار پسورد را از درخواست ارسال شده جدا کنیم. اگر قبل از این کاربری در سایت عضو شده بود ابتدا چک میکند که پوزنیم وارد شده تکراری نباشد، اگر پوزنیم تکراری بود مارا به همان صفحه `sign up` هدایت میکند. اما اگر پوزنیم تکراری نبود چک میکند که پسورد و تکرار پسورد یکسان است یا نه. اگر یکسان بود `object` جدیدی از جنس `user` ساخته و پوزنیم و پسورد این `object` را با پوزنیم و پسورد

وارد شده توسط کاربر پر کرده و یوزر جدید را ذخیره میکند.

اگر کاربری از قبل در دیتابیس موجود نباشد هم مستقیما object را ساخته و سایر کارها را انجام میدهد. بعد از کلیک بر روی register باید ما را به url و تابع زیر از بخش api هدایت کند تا برای الگوریتم تشخیص چهره هم یک کاربر با همان مشخصات ایجاد کرده و پوشه ای برای تصاویر آن ایجاد کند. برای ارسال نام کاربری فرد به این آدرس دوباره از کوکی استفاده کرده و نام کاربری را به عنوان name در کوکی ذخیره میکند و سپس از آن استفاده میکند.

```
url(r'^new/$', views.new)

def new(request):
    if request.method == "GET":
        print(request.COOKIES.get('name'))
        name1 = request.COOKIES.get('name')
        print(request)
        # if request.GET.get("username", None)
        # is not None and request.GET.get("email", None) is not None:
        print("heloooooo")
        user = User.objects.create_user(name1, '')
        user.first_name = name1
        user.last_name = name1
        user.save()
        training_folder = os.path.join(TRAINED_FACES_PATH, str(user.pk))
        if not os.path.exists(training_folder):
            os.makedirs(training_folder)
    return redirect('manager:index')
```

بعد از انجام تابع و ایجاد کاربر جدید برای تشخیص چهره باید به صفحه اصلی سایت هدایت شود. ایجاد گروه جدید: برای ایجاد گروه جدید با کلیک بر روی لینک مرتبط با ایجاد گروه url و تابع زیر فراخوانی میشود

```
url(r'^createGroup/(.)*$', views.createGroup, name='createGroup')

def createGroup(request, i):
    inputs = request.get_full_path()
    values = inputs.split('?')[1].split('&')
    userObj = get_object_or_404(user, pk=user.objects.filter(username=request.COOKIES.get('user'))[0])
    newGroup = Group()
    newGroup.user = userObj
    newGroup.groupName = values[0].split('=')[1]
    newGroup.groupDesc = values[1].split('=')[1]
    newGroup.save()
    return redirect('manager:index' )
```

درخواستی که برای این تابع ارسال میشود شامل نام و توضیح مربوط به گروه میباشد. ابتدا باید name را از درخواست جدا کنیم. از آنجایی که برای ورود به این صفحه یا توسط تابع login و یا signup

وارد شدیم، یوزرنیم کاربر وارد شده در کوکی ذخیره شده است و چون گروه یک متغیر `key foreign` از مدل یوزر دارد ابتدا باید `object` مربوط به یوزرنیم کاربر را پیدا کرده و تغییر `user (متغیر foreign)` را مساوی آن `object` قرار دهیم و سپس سایر فیلد های مربوط به گروه را پر کرده و آن را ذخیره کنیم. بعد از ذخیره گروه جدید باید دوباره به صفحه اصلی سایت هدایت شویم.

ورود به صفحه گروه: برای ورود به صفحه ای که در آن پسوردهای مربوط به هر گروه نمایش داده میشود `url` و تابع زیر فراخوانی میشود.

```
url(r'^^(?P<pk>[0-9]+)/$', views.passwordsPage, name='passwords')
```

```
def passwordsPage(request, pk):
    group = get_object_or_404(Group, pk=pk)
    return render(request, 'passwordsPage.html',
                  {'passes': group.password_set.all(), 'group_id': group.id})
```

با کلیک روی نام هر گروه `id` مربوط به آن گروه در `url` ارسال میشود. با `url` زیر `id` هر پست در متغیر `pk` ذخیره میشود. همراه درخواست برای تابع فرستاده می شود. ابتدا باید تشخیص بدیم که گروه مربوط به این `pk` کدام است و `object` آن را تشخیص دهیم و سپس تما پسوردهای مربوط به آن گروه را در صفحه نمایش دهیم.

اضافه کردن پسورد جدید: برای اضافه کردن پسورد جدید با کلیک بر روی لینک مربوط، `url` و تابع زیر فراخوانی می شود.

```
url(r'^^(.)*addPassword/(.)*$', views.addPassword, name='addPassword'),
```

```
def addPassword(request, i, j):
    inputs = request.get_full_path()
    values = inputs.split('?')[1].split('&')
    id2 = int(values[0].split('=')[1])
    group = get_object_or_404(Group, pk=id2)
    newPass = Password()
    newPass.group = group
    newPass.name = values[1].split('=')[1]
    newPass.password = values[2].split('=')[1]
    newPass.desc = values[3].split('=')[1]
    newPass.save()
    return redirect('manager:passwords', pk=group.id)
```

درخواست ارسال شده برای این تابع شامل نام، پسورد و توضیح در مورد آن پسورد است همچنین این درخواست از صفحه مربوط به گروه ارسال شده است و `id` مربوط به گروه را همراه با درخواست میفرستد. ابتدا نیاز داریم تا `name`، `password` و `desc` را از درخواست جدا کنیم. به کمک `id` گروه مشخص میکنیم که این `id` مربوط به کدام گروه است و `object` گروه را ایجاد میکنیم. سپس متغیر `key (foreign group)` را برابر `object` گروه قرار میدهیم. سایر متغیر های پسورد را نیز تکمیل کرده و

پسورد را ذخیره میکنیم. بعد از ذخیره پسورد نیاز داریم تا به صفحه گروه همان پسورد هدایت شویم. برای این منظور نیاز است تا id گروه را همراه درخواست خود ارسال کنیم.

پاک کردن گروه: برای پاک کردن گروه url و تابع زیر فراخوانی می شود.

```
url(r'^(.)*deleteGroup/(.)*$', views.deleteGroup, name='deleteGroup')

def deleteGroup(request, i,j):
    inputs = request.get_full_path()
    values = inputs.split('?')[1].split('&')
    print(values)
    group = get_object_or_404(Group, pk=int(values[0].split('=')[1]))
    group.delete()
    return redirect('manager:index')
```

ابتدا نیاز داریم تا به کمک id گروه که درون درخواست وجود دارد تشخیص بدیم که id متعلق به کدام گروه است. گروه را پاک کرده و دوباره به صفحه گروه ها هدایت میشویم.

پاک کردن پسورد: برای پاک کردن پسورد url و تابع زیر استفاده می شود.

```
url(r'^(.)*deletePassword/(.)*$', views.deletePassword, name='deletePassword'),

def deletePassword(request, i, j):
    inputs = request.get_full_path()
    values = inputs.split('?')[1].split('&')
    group = get_object_or_404(Group, pk=int(values[0].split('=')[1]))
    password = group.password_set.all().filter(pk=int(values[1].split('=')[1]))[0]
    password.delete()
    return redirect('manager:passwords', pk=group.id)
```

ابتدا نیاز داریم تا به کمک id گروه که درون درخواست وجود دارد تشخیص بدیم که id متعلق به کدام گروه است. با تشکیل object گروه پسوردی که دارای مشخصاتی که میخواهیم پاک کنیم را پیدا کرده و آن را پاک میکنیم و دوباره به صفحه گروه ها هدایت می شویم.

ورود به صفحه های ذخیره عکس و ورود به کمک عکس: برای ورود به این صفحات url ها و توابع زیر فراخوانی می شود.

```
url(r'^save$', image_views.image_save, name='image_save'),

def image_login(request):
    return render(request, 'login_picture.html')

# login
url(r'^login$', image_views.image_login, name='image_login'),

def image_save(request):
    return render(request, 'save_picture.html')
```

## ۳.۲ گام سوم : الگوریتم تشخیص چهره

ایجاد کاربر جدید برای تشخیص چهره: برای ایجاد کاربر جدید همان طور که در قسمت های قبل توضیح داده شد url و تابع زیر فراخوانی می شود.

```
url(r'^new/$', views.new),  
  
def new(request):  
    if request.method == "GET":  
        print(request.COOKIES.get('name'))  
        name1 = request.COOKIES.get('name')  
        print(request)  
        # if request.GET.get("username", None)  
        # is not None and request.GET.get("email", None) is not None:  
        print("heloooooo")  
        user = User.objects.create_user(name1, '')  
        user.first_name = name1  
        user.last_name = name1  
        user.save()  
        training_folder = os.path.join(TRAINED_FACES_PATH, str(user.pk))  
        if not os.path.exists(training_folder):  
            os.makedirs(training_folder)  
    return redirect('manager:index')
```

که کاربری با نامی مشابه بوزنیم فرد sign up کننده می سازد.  
اضافه کردن عکس های جدید و train کردن مدل با عکس جدید: برای این منظور url و تابع زیر فراخوانی می شود.

```
url(r'^train/$', views.train),  
  
def train(request , img):  
    img = img[22:]  
    name1 = request.COOKIES.get('user')  
    user= User.objects.filter(first_name=name1)[0]  
    if request.method == "GET":  
        # check to see if an image was uploaded  
        #if request.GET.get("imageBase64", None)  
        #is not None and request.GET.get("user", None) is not None:  
            # grab the uploaded image  
        image = _grab_image(base64_string=img)  
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  
        rects = detector.detectMultiScale(image, scaleFactor=1.1, minNeighbors=5,  
                                         minSize=(30, 30), flags=0)  
  
        # construct a list of bounding boxes from the detection  
        rects = [(int(x), int(y), int(x + w), int(y + h)) for (x, y, w, h) in rects]  
        if len(rects) == 0:  
            return JsonResponse({"error": "No faces detected"})
```

```

else:
    x, y, w, h = rects[0]
    cv2.imwrite(
        TRAINED_FACES_PATH + "/" + str(user.pk) + "/" + str(uuid.uuid4()) + ".jpg",
        cv2.resize(image[y:h, x:w], (256, 256)))
return redirect('image:image_save')

تشخیص چهره: برای این منظور url و تابع زیر فراخوانی میشود

url(r'^recognize/$', views.recognize),

def recognize(request, img):
    img = img[22:]
    # initialize the data dictionary to be returned by the request
    data = {}
    # check to see if this is a get request
    if request.method == "GET":
        # check to see if an image was uploaded
        #if request.GET.get("imageBase64", None) is not None:
            # grab the uploaded image
        image = _grab_image(base64_string=img)

        # otherwise, assume that a URL was passed in

        # convert the image to grayscale, load the face cascade detector,
        # and detect faces in the image
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        rects =
        detector.detectMultiScale(image, scaleFactor=1.1, minNeighbors=5,
                                   minSize=(30, 30), flags=0)

        # construct a list of bounding boxes from the detection
        rects = [(int(x), int(y), int(x + w),
                  int(y + h)) for (x, y, w, h) in rects]
        if len(rects) == 0:
            data.update({"detected": False})
        else:
            x, y, w, h = rects[0]
            recognizer.setThreshold(THRESHOLD)
            identity, confidence = recognizer.predict(
                cv2.resize(image[y:h, x:w], (256, 256)))
        )
        smile = smiledetector.detectMultiScale(
            image[y:h, x:w],
            scaleFactor=1.7,
            minNeighbors=22,
            minSize=(25, 25),
            flags=0)

```

```

smiling = False if len(smile) == 0 else True
try:
    user = User.objects.get(id=identity)
    user = {
        "first_name": user.first_name,
        "last_name": user.last_name,
        "username": user.username,
        "email": user.email,
        "id": user.pk,
    }
except User.DoesNotExist:
    user = ""

# update the data dictionary with the faces detected
data.update({"detected": True, "identity":
    identity, "user": user, "box": rects, "smiling": smiling})

if user is not "":
    r = redirect('manager:index')
    print('Bande khoda', user)
    r.set_cookie('permission', 'T')
    r.set_cookie('login_user', user)
    r.set_cookie('user', user['username'])

    # return a JSON response
    return r
else:
    return redirect('image:image_login')

```

<https://gitlab.com/navidhz/ramzii.git>