توضیحات کد:

following: به کمک این متد، کاربر پزشک مورد علاقه خود را به لیست اضافه میکند. از آنجایی که میخواهیم اطلاعاتی را که از سمت کاربر به سمت سرور ارسال میشود را در پایگاه داده ذخیره کنیم از متد POST استفاده می‌کنیم.

```
@api_view(['POST'])
def following(request):

    if request.method == 'POST':
        data = request.data
        favoriteObj = favoriteSerializer(data=data)

        if favoriteObj.is_valid():
            favoriteObj.save()
            return Response("Done", status=CREATED)
        else:
            return Response(favoriteObj.errors, status=INVALID_DATA)
```

API استفاده شده برای این قابلیت بصورت زیر است. در بدنه این API دو ویژگی user_id و doctor_id در فرمت JSON ارسال می‌شود.

```
path('follow/', views.following),
```

comment: به کمک این متد کاربر کامنت خود را برای پزشک مورد نظر اعلام می‌کند. از آنجایی که میخواهیم اطلاعاتی را که از سمت کاربر به سمت سرور ارسال میشود را در پایگاه داده ذخیره کنیم از متد POST استفاده می‌کنیم.

```
@api_view(['POST'])
def comment(request):

    if request.method == 'POST':
        data = request.data
        commentObj = commentSerializer(data=data)

        if commentObj.is_valid():
            commentObj.save()
            return Response("Done", status=CREATED)
        else:
            return Response(commentObj.errors, status=INVALID_DATA)
```

API استفاده شده برای این قابلیت بصورت زیر است. در بدنه این API سه ویژگی user_id و doctor_id و comment در فرمت JSON ارسال می‌شود.

```
path('comment/', views.comment),
```

search: به کمک این متد، کاربر میتواند با نام یا کدنظام پزشکان جستوجو کند. از آنجایی که میخواهیم اطلاعاتی را از سمت سرور دریافت کنیم از متد GET استفاده می‌کنیم. درصورتی که عبارت جستوجو عدد باشد در ستون مربوط به کدنظام کوئری زده میشود درغیراینصورت در ستون نام این عمل انجام می‌شود.

```
@api_view(['GET'])
def search(request, keyword):

    if request.method == "GET":

        if keyword.isdigit():
            try:
                output = doctor.objects.get(dNumber=int(keyword))
                _output = doctorSerializer(output)
                return Response(_output.data, status=SUCCEEDED_REQUEST)
            except ObjectDoesNotExist:
                return Response("Not Found", status=SUCCEEDED_REQUEST)

        else:
            output = doctor.objects.filter(name__contains=keyword).all()
            _output = doctorSerializer(output, many=True)
            if not _output.data:
                return Response("Not Found", status=SUCCEEDED_REQUEST)
            return Response(_output.data, status=SUCCEEDED_REQUEST)
```

API استفاده شده برای این قابلیت بصورت زیر است. در URL کافیست عبارت مورد نظر را بصورت زیر وارد می‌کنیم.

```
path('search/keyword=<str:keyword>/', views.search),
```

set_appointment: به کمک این متد کاربر کامنت خود را برای پزشک مورد نظر اعلام می‌کند. از آنجایی که میخواهیم اطلاعاتی را که از سمت کاربر به سمت سرور ارسال میشود را در پایگاه داده ذخیره کنیم از متد POST استفاده می‌کنیم. درصورتی که وقت مورد نظر از قبل رزرو شده باشد یا زمان درستی نباشد (به عبارت دیگر پزشک در آن زمان حضور نداشته باشد)، ذخیره سازی در پایگاه داده صورت نمیگیرد.

```
@api_view(['POST'])
def set_appointment(request):

    if request.method == 'POST':
        appointmentData = request.data
        day = datetime.strptime(appointmentData['date'], "%Y-%m-%d").date().strftime("%A")

        if working_time.objects.filter(doctor_id=appointmentData['doctor_id'],
                                       day=day,
                                       start_time__contains=appointmentData["time"]).exists():

            if not appointment.objects.filter(doctor_id=appointmentData['doctor_id'],
                                              date__contains=appointmentData['date'],
                                              time__contains=appointmentData["time"]).exists():

                appointmentObj = appointmentSerializer(data=appointmentData)

                if appointmentObj.is_valid():
                    appointmentObj.save()
                    return Response("Reserved successfully", status=CREATED)
                else:
                    return Response(appointmentObj.errors, status=INVALID_DATA)
            else:
                return Response("This time was reserved", status=INVALID_DATA)
        else:
            return Response("This time is not correct", status=INVALID_DATA)
```

API استفاده شده برای این قابلیت بصورت زیر است. در بدنه این API چهار ویژگی user_id و doctor_id، date و time در فرمت JSON ارسال می‌شود.

```
path('setAppointment/', views.set_appointment),
```

show_times: به کمک این متد، کاربر میتواند زمان حضور پزشک مورد نظر را در روز مشخص مشاهده کند (به عبارت دیگر زمان‌هایی که میتواند رزور کند را به او نشان می‌دهیم). از آنجایی که میخواهیم اطلاعاتی را از سمت سرور دریافت کنیم از متد GET استفاده می‌کنیم.

```
@api_view(['GET'])
def show_times(request, date, doctorID):

    if request.method == 'GET':
        day = datetime.strptime(date, "%Y-%m-%d").date().strftime("%A")
        doctor_times = working_time.objects.filter(day=day, doctor_id=doctorID).all()
        appointment_times = appointment.objects.filter(date=date, doctor_id=doctorID).all()
        suggestion_times = []
        flag = 0
        for dt in doctor_times:
            dtime = dt.start_time
            for at in appointment_times:
                atime = at.time
                if dtime == atime:
                    flag = 1
                    break
            if flag != 1:
                suggestion_times.append(dt)
            else:
                flag = 0
        _suggestion_times = workingTimeSerializer(suggestion_times, many=True)
        if _suggestion_times is None:
            return Response("There is no time.", status=SUCCEEDED_REQUEST)
        return Response(_suggestion_times.data, status=SUCCEEDED_REQUEST)
```

API استفاده شده برای این قابلیت بصورت زیر است. در URL کافیست doctor_id و date را ارسال کنیم.

```
path('showTimes/date=<str:date>/id=<int:doctorID>/', views.show_times),
```

register_userinfo: به کمک این متد، کاربر عادی می‌تواند اطلاعات شخصی خود را(نظیر نام و نام خانوادگی، شماره تماس و ...) در سامانه ثبت کند. از آنجایی که می‌خواهیم اطلاعاتی را به سمت سرور بفرستیم از متد POST استفاده می‌کنیم.

```
@api_view(['POST'])
def register_userinfo(request):

    if request.method == 'POST':
        data = request.data
        infoObj = normalUserSerializer(data=data)

        if infoObj.is_valid():
            infoObj.save()
            return Response("Done", status=CREATED)
        else:
            return Response(infoObj.errors, status=INVALID_DATA)
```

API استفاده شده برای این قابلیت بصورت زیر است. در بدنه این API چهار ویژگی user_id و phone، address و name در فرمت JSON ارسال می‌شود.

```
path('registerinfo/', views.register_userinfo),
```

edit_userinfo: به کمک این متد، کاربر عادی می‌تواند اطلاعات شخصی خود را(نظیر نام و نام خانوادگی، شماره تماس و ...) در سامانه ویرایش کند. از آنجایی که کاربر می‌تواند یک یا چند مورد از اطلاعاتش را ویرایش کند، از متد PATCH استفاده می‌کنیم.

```
@api_view(['PATCH'])
def edit_userinfo(request , user_id):

    if request.method == 'PATCH':
        try:
            found = normal_user.objects.get(user_id=user_id)
        except found.DoesNotExist:
            return Response(status=status.HTTP_404_NOT_FOUND)

        data = request.data
        infoObj = normalUserSerializer(found, data=data , partial=True)

        if infoObj.is_valid():
            infoObj.save()
            return Response("Updated successfully", status=SUCCEEDED_REQUEST)
        else:
            return Response(infoObj.errors, status=INVALID_DATA)
```

API استفاده شده برای این قابلیت بصورت زیر است. در URL نیاز است user_id را بفرستیم و در بدنه این API از یک تا سه ویژگی شامل phone، address و name را می‌توانیم در فرمت JSON ارسال کنیم.

```
path('editinfo/user_id=<str:user_id>', views.edit_userinfo),
```

filter: به کمک این متد، کاربر عادی می‌تواند لیست تمامی پزشکان موجود در سامانه را مشاهده کند و نیز می‌تواند در صورت تمایل، این لیست را با توجه به شهر، حوزه تخصصی پزشک و تحصیلات پزشک، فیلتر کند. از آنجایی که می‌خواهیم اطلاعاتی را از سمت سرور دریافت کنیم از متد GET استفاده می‌کنیم.

```
@api_view(['GET'])
def filter(request,city='',education='',field=''):
    city = request.GET.get('city')
    education = request.GET.get('education')
    field = request.GET.get('field')
    if request.method == "GET":
        try:
            output = doctor.objects.all()
            _output = doctorSerializer(output,many=True)
            if city:
                output = output.filter(address__contains = city).all()
                _output = doctorSerializer(output,many=True)
            if education:
                output = output.filter(education = education).all()
                _output = doctorSerializer(output,many=True)
            if field:
                output = output.filter(field = field).all()
                _output = doctorSerializer(output,many=True)
            return Response(_output.data, status=SUCCEEDED_REQUEST)
        except ObjectDoesNotExist:
            return Response("Not Found", status=SUCCEEDED_REQUEST)
```

API استفاده شده برای این قابلیت بصورت زیر است. استفاده از فیلترها اختیاری است و نیازی نیست هر سه فیلتر بر لیست پزشکان اعمال شوند.

```
url(r'^getlist(?:/city=(?P<city>[a-zA-Z]+))?(?:/education=(?P<education>[a-zA-Z]+))?(?:/field=(?P<field>[a-zA-Z]+))?/$',views.filter),
```

doctorinfo: به کمک این متد، کاربر عادی می‌تواند اطلاعات پزشک مورد نظر خود (نظیر نام و نام خانوادگی، شماره تماس، آدرس مطب، تحصیلات و ...) و نیز ساعات کاری پزشک را مشاهده کند. از آنجایی که می‌خواهیم اطلاعاتی را از سمت سرور دریافت کنیم از متد GET استفاده می‌کنیم.

```
@api_view(['GET'])
def doctorinfo(request, doctorid):

    if request.method == "GET":
        try:
            output1 = doctor.objects.get(id=doctorid)
            _output1 = doctorSerializer(output1)
            output2 = working_time.objects.filter(doctor_id=doctorid).all()
            _output2 = workingTimeSerializer(output2 , many=True)
            output = {'info': _output1.data , 'working_times': _output2.data}
            return Response(output, status=SUCCEEDED_REQUEST)
        except ObjectDoesNotExist:
            return Response("Not Found", status=SUCCEEDED_REQUEST)
```

API استفاده شده برای این قابلیت بصورت زیر است. تنها در URL نیاز است doctorid را ارسال کنیم.

```
path('doctorinfo/doctorid=<str:doctorid>/', views.doctorinfo),
```