API Documentaion

مدل های دیتابیسی کنونی:

```python
from django.db import models
from django.contrib.auth.models import User

class StudentProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    stu_no = models.CharField(verbose_name="Student Number", max_length=10, blank=False, null=False)
    is_ta = models.BooleanField(default=False)
    phone_no = models.CharField(verbose_name="Phone Number", blank=False, null=False)

class ProfessorProfile(models.Model):

    user = models.OneToOneField(User, on_delete=models.CASCADE)
    national_no = models.CharField(verbose_name="National Number", max_length=10, unique=True, blank=False)
    students = models.ManyToManyField('StudentProfile', through='StudentProfessor', related_name='professors', default=None)
    def __str__(self):
        return f"{self.user} - {self.national_no}"
#intermediary model for adding some elemnts like datetime for relations if necessary
class StudentProfessor(models.Model):
    student = models.ForeignKey(StudentProfile, on_delete=models.CASCADE, default=None)
    professor = models.ForeignKey(ProfessorProfile, on_delete=models.CASCADE, default=None)

class Course(models.Model):
    name = models.CharField(max_length=100, verbose_name="Course Name")
    term = models.IntegerField(verbose_name="Term")
    required_TAs = models.IntegerField(verbose_name="Required TAs")
    num_applicants = models.IntegerField(verbose_name="Number of Applicants", default=0)
    num_tas = models.IntegerField(verbose_name="Number of TAs", default=0)
    section = models.IntegerField(verbose_name="Section")
    professor = models.ForeignKey('ProfessorProfile', on_delete=models.CASCADE, related_name='courses')

    def __str__(self):
        return f"{self.name} - Term {self.term} - Section {self.section}"
```

API ها:

Login

url : http://127.0.0.1:8000/users/login/

method : POST

ورودی های الزامی :

Login

```
POST /users/login/
```

```
HTTP 400 Bad Request
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "username": [
        "This field is required."
    ],
    "password": [
        "This field is required."
    ]
}
```

خروجی: در حالت صحیح کد 200 در حالت هر حالت نادرست 400

خروجی درحالی که کاربر استاد باشد:

```
POST /users/login/
```

```
HTTP 200 OK
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "token": "82d574cedc113f5ee0a8a02eb4af80e076125721",
    "user_data": {
        "id": 1,
        "role": "professor",
        "username": "testuser",
        "first_name": "bro",
        "last_name": "yechi_dige",
        "national_no": "99546448",
        "email": "test@example.com"
    }
}
```

از قسمت role آن میتوان متوجه پروفسور بودن آن شد همچنین مقدار id متعلق به یوزری است که در پروفایل استاد استفاده شده. این در قسمت های بعدی استفاده زیادی خواهد داشت.

خروجی درحالی که کاربر دانشجو باشد:

Login

```
POST /users/login/
```

```
HTTP 200 OK
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "token": "d34b0d6f08bcafdb86b0022c6105e332111a11cc",
    "user_data": {
        "username": "testuserst",
        "first_name": "yechi",
        "last_name": "yechi_dige",
        "phone_no": "886454665",
        "stu_no": "1222334455",
        "is_ta": false,
        "email": "test@example.com"
    }
}
```

Professor register(sign-up)

url : http://127.0.0.1:8000/users/professor-register/

method : POST

ورودی های الزامی :

Professor Register

```
POST /users/professor-register/
```

```
HTTP 400 Bad Request
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "user": [
        "This field is required."
    ],
    "national_no": [
        "This field is required."
    ],
    "password2": [
        "This field is required."
    ]
}
```

ورودی های اصلی(ورودی های که به طور عادی انتظار انها را داریم) :


```
{
        "user": {
           "username": "testuser",
           "first_name": "yechi",
           "last_name": "yechi_dige",
           "email": "test@example.com",
           "password": "testpassword"
        },
        "national_no": "1222334455",
        "password2": "testpassword"
}
```

خروجی در صورت اطلاعات تکراری یا غلط:

Professor Register

```
POST /users/professor-register/
```

```
HTTP 400 Bad Request
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "user": {
        "username": [
            "A user with that username already exists."
        ]
    },
    "national_no": [
        "professor profile with this National Number already exists."
    ]
}
```

خروجی در صورت درخواست صحیح (برگرداندن اطلاعات کاربری که ثبت شده)

Student register(sign-up)

url : http://127.0.0.1:8000/users/student-register/

method : POST

ورودی های الزامی :

Student Register

```
POST /users/student-register/
```

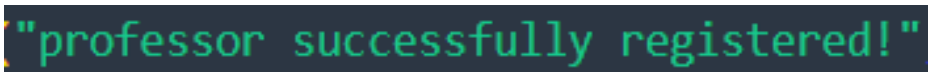
```
HTTP 400 Bad Request
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "user": [
        "This field is required."
    ],
    "stu_no": [
        "This field is required."
    ],
    "phone_no": [
        "This field is required."
    ],
    "password2": [
        "This field is required."
    ]
}
```

ورودی های اصلی(ورودی های که به طور عادی انتظار انها را داریم) :


```
{
        "user": {
           "username": "testuser9",
           "first_name": "yechi",
           "last_name": "yechi_dige",
           "email": "test@example.com",
           "password": "testpassword"
        },
        "stu_no": "1222334455",
        "phone_no": "0991199853",
        "password2": "testpassword"
}
```

خروجی صحیح:



خروجی در صورت اطلاعات غلط شبیه قبلی

Logout

url : http://127.0.0.1:8000/users/logout/

method :GET

ورودی های الزامی :

بدون اینکه وارد شده باشیم اجازه دسترسی نداریم و در ریکوئست گت باید توکن موجود باشد.

Logout

```
GET /users/logout/
```

```
HTTP 401 Unauthorized
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
WWW-Authenticate: Token

{
    "detail": "Authentication credentials were not provided."
}
```

ورودی صحیح:

```
Authorization: token b2d617908123bd501e62f9bc8c81e3301e7e4aa5
```

باید هدری با این مشخصات به طوری که توکن بعد عبارت token باید با یک فاصله نوشته شود پاس داده شود

خروجی با اطلاعات غلط:

شبیه عکس ورودی های الزامی

خروجی با اطلاعات صحیح :

تنها کد 200 برگردانده میشود.

ProfessorCourseAPIView

url : http

://127.0.0.1:8000/users/professor/get-lesson/

method :GET

درسهای مربوط به هر استاد داده میشوند در صورت احراز هویت شده بودن.

ورودی های الزامی :

بدون اینکه وارد شده باشیم اجازه دسترسی نداریم و در ریکوئست گت باید توکن موجود باشد

مشابه عکس ورودی الزامی Logout

ورودی صحیح:

```
Authorization: token b2d617908123bd501e62f9bc8c81e3301e7e4aa5
```

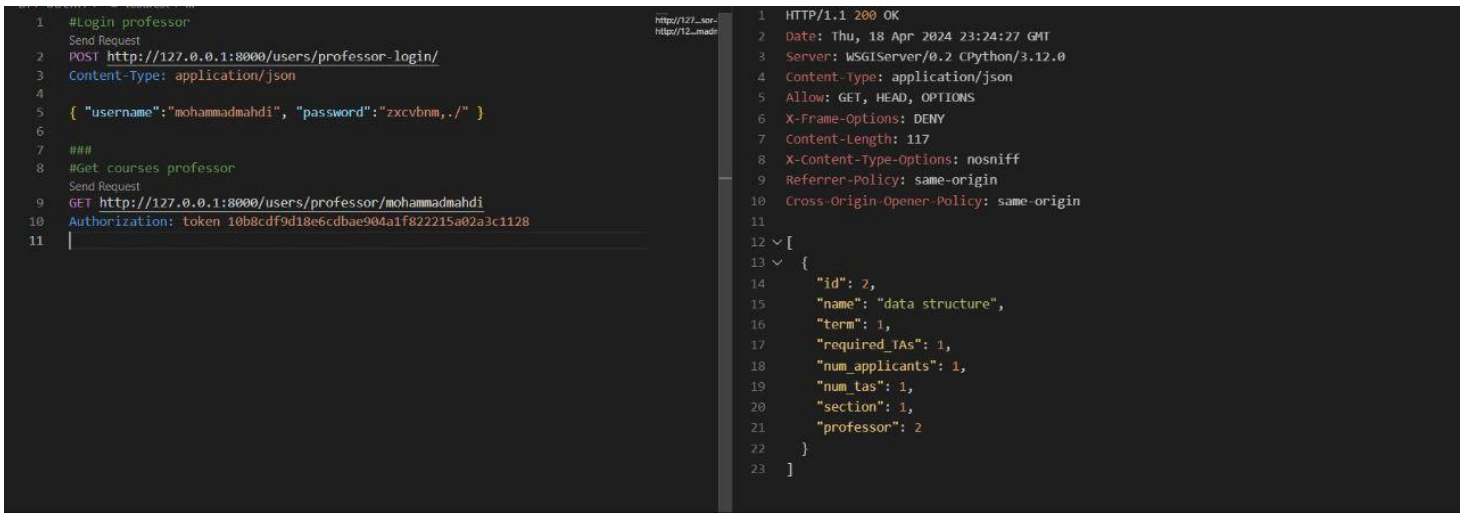
باید هدری با این مشخصات به طوری که توکن بعد عبارت token باید با یک فاصله نوشته شود پاس داده شود

خروجی با اطلاعات غلط:

شبیه عکس ورودی های الزامی logout

خروجی با اطلاعات صحیح :

کد 200 برگردانده میشود. همراه لیستی از دروس



CourseRegisterView

Method: post

url : http

://127.0.0.1:8000/users/ professor/create-lesson/

ورودی ها:

```
data = {
    "name": "data structure",
    "term": 4022,
    "required_TAs": 5,
    "num_applicants": 10,
    "num_tas": 3,
    "section": 2,
    "professor": 1  # Assuming ProfessorProfile has an auto-incremented id
            }
```

فیلد پروفسور همان آیدی ای را می گیرد که در قسمت لاگین بر می گردانیم.

خروجی:

```
f"lesson '{name}' is added to database",
```

جای name اسم درس قرار می گیرید

نکته:

اسم درس دوبار نمی تواند تکرار شود.

ProfessorDetailsView

method:get

url : http://127.0.0.1:8000/users/ professor/get-detail/<int:professor_id>

ورودی:

شماره آیدی پروفسور (گرفته شده در لاگین)

خروجی:

{"professor":{"user":1,"national_no":"99546448","students":[]},"students":[]}

لیست دانشجو های اول دانشجو هایی هستند که متعلق به استادند. و لیست دانشجو های دوم ریز اطلاعات مربوط به آن دانشجویان.

StudentProfilePartialUpdateView

method:patch

url : http://127.0.0.1:8000/users/student/profile/<int:id>/

اطلاعات دانشجویان به طور تکه ای آپدیت میکند یعنی لازم نیست حتما کل آن یکجا آپدیت شود قسمت های جدا هم اپدیت می شود. فقط ویژگی ها خاص دانشجو مثل شماره تلفن و شماره دانشجویی و تی ای بودن می تواند آپدیت شود. اسم و فامیل و ایمیل و ... با آپدیت یوزر آپدیت میشود

ورودی:

مثلا میخوام فقط stu_no بودن رو آپدیت کنم.

```
data = {
    'stu_no':'99546448',

}
```

اگر چیزای دیگه خواستم اونا رو هم اضافه می کنم. همچنین ای دی همونی هست که از لاگین گرفتیم.(مال یوزره)

خروجی:

اطلاعات جدید دانشجو

ProfessorProfilePartialUpdateView

method:patch

url :http://127.0.0.1:8000/users/student/profile/<int:id>/

ورودی

شبیه آپدیت دانشجو یعنی مثلا چیزی مثل کدملی رو آپدیت می کنی نه اسم و فامیل چون اون مال آپدیت یوزر می شه.

خروجی

شبیه آپدیت دانشجو

UserPartialUpdateView

method:patch

url :http://127.0.0.1:8000/users/ update/<int:id>/

ورودی

```
data = {
    'first_name':"bro",
}
```

ویژگی های یوزر به طور بخش بخش یا کامل آپدیت می شود. آیدی هم همان آیدی ای است که موقع لاگین میگیریم.

خروجی

اطلاعات جدید یوزر

CourseDeleteView

method:delete

url :http://127.0.0.1:8000/users/ professor/delete-lesson/<str:name>/

ورودی:

اسم درس را برای ورودی باید داده شود.

خروجی :ندارد