SEE U

Software Revision and Control 2

Hang Xie 1025910
Jiasheng Yu 1025957
Rui Chen 1025567
Jinhang Wu 1025891

Version: (1.0) **Date: (05/10/2020)**

Front-end

Our group has made a consensus that we will switch from the previous framework to Vue.js, as we cannot build the running environment of flutter in our computers.

Since our aim is to build a mobile app that can share our campus life with other students, it is unwise to implement our platform in Android and iOS separately, which requires a large amount of time and specialty. Thus, thanks to the HBuilderX, we tackle the problems of the cross-domain problem by adopting mixed development in our project, where we could only develop one code but could be deployed into different operating systems.

Figure 1. HBuilderX Platform

For the first week of our revision, our front end successfully designed the UI of login and registration page, as it shows below.

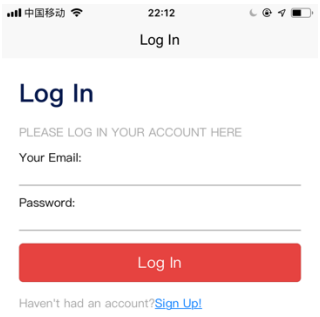

Figure 2. The Login Page of SeeU

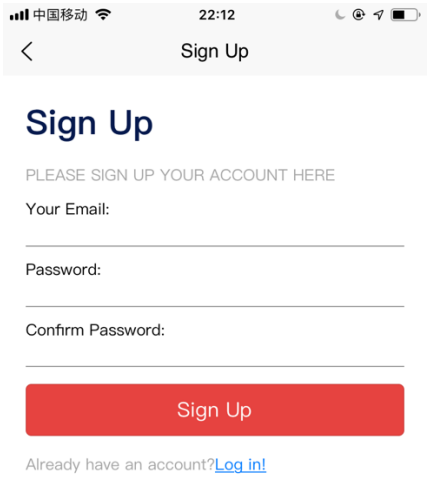

Figure 3. The Sign Up Page of SeeU

I have tested the login and registration interfaces with Andy, and it could interact with the database and run successfully.

Besides, the bonus of this week is that I have also developed one of the main pages of SeeU. Although we haven't tested the interface and revise the CSS detail, we are thrilled that we have developed the very first prototype of our SeeU.

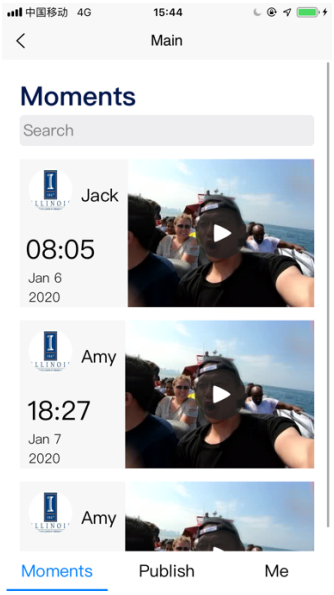

Figure 4. The Moments Page of SeeU

All in all, the work of our front-end went smoothly without too many bugs and accidents compared with the back-end. We are also working actively with the back-end to cope with any possible problems they may encounter.

Back-End

This week, we firstly build some test interfaces such as add_user and show_users. The test tool we used is postman.

Test example:

- add_user:
  For example, add Calvin.

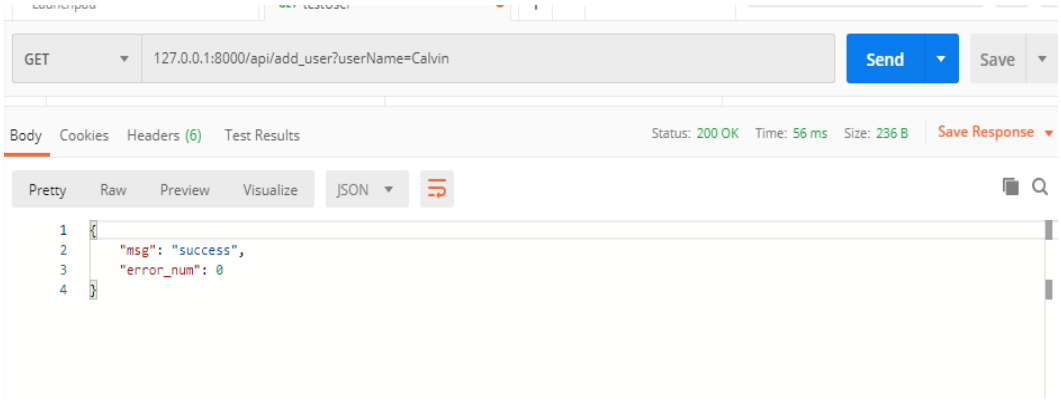

Figure 5. Test of Add User Interface

- show_users:

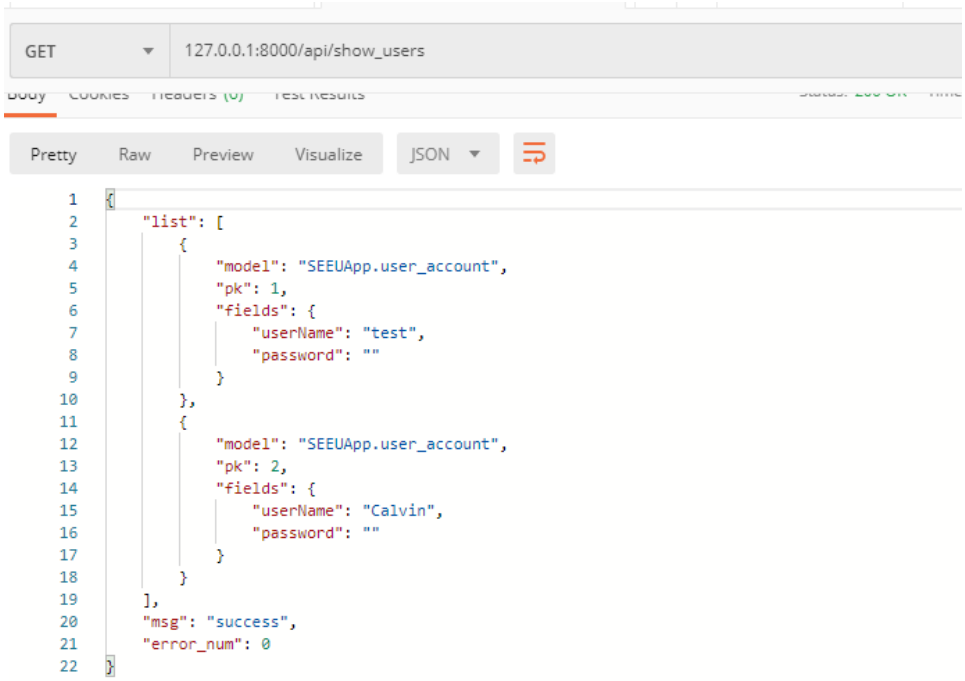

Figure 6. Test of Show Users Interface

Bug

1. Forbidden (CSRF cookie not set.): ****

Just comment on this line

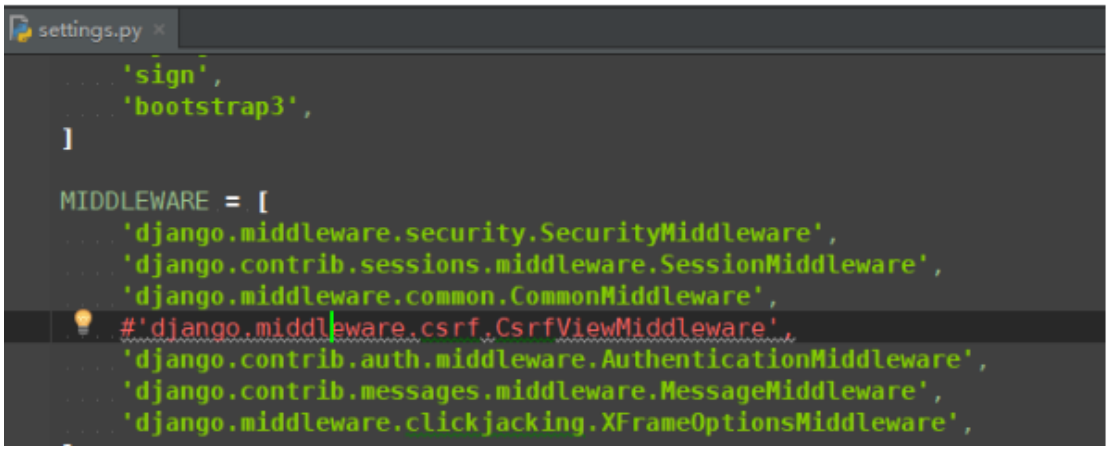

Figure 7. Code Issue 1

Solved

2. The object has no attribute 'strip'" and Django cannot use request.POST.get to obtain the String

We can the interface body and change to request.body

```
@require_http_methods(["POST"])
def user_register(request):
    response = {}
    try:
        email = MyUser(email=request.POST.get('email'))
        password = MyUser(password=request.POST.get('password'))
        user = User.objects.create_user(email, password)
        user.save()
        response['msg'] = 'success'
        response['error_num'] = 0
    except Exception as e:
        response['msg'] = str(e)
        response['error_num'] = 1
```

Still have bugs and cannot be solved so far.

3. After combining the front-end and back-end

Run front-end:

```
(venv) D:\SeniorProject\SeeU\SeeU_frontend>npm run dev
npm ERR! code ENOENT
npm ERR! syscall open
npm ERR! path D:\SeniorProject\SeeU\SeeU_frontend\package.json
npm ERR! errno -4058
npm ERR! enoent ENOENT: no such file or directory, open 'D:\SeniorProject\SeeU\SeeU_frontend\package.json'
npm ERR! enoent This is related to npm not being able to find a file.
npm ERR! enoent
```

Figure 8. Issues Occurred When Combining Front-end and Back-end

Cannot be solved.

4. npm WARN package.json: No repository field

Solution:

As dan_nl stated, you can add a private fake repository in package.json. You don't even need name and version for it:

```
{
  ...,
  "repository": {
    "private": true
  }
}
```

Better still: Set the private flag directly. This way npm doesn't ask for a README file either:

```
{
  "name": ...,
  "description": ...,
  "version": ...,
  "private": true
}
```

Figure 9. Solution of npm WARN Issue

But still cannot solve.

5. When we post the request, we cannot post the information to our database. It shows nothing.

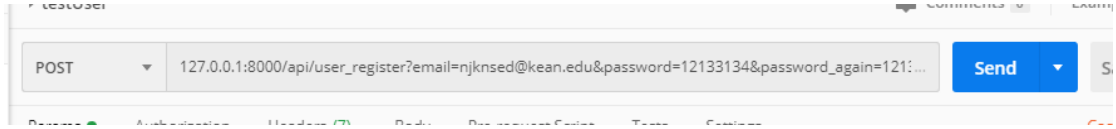

Figure 10 (a). Troubleshooting During the Test of the Interface

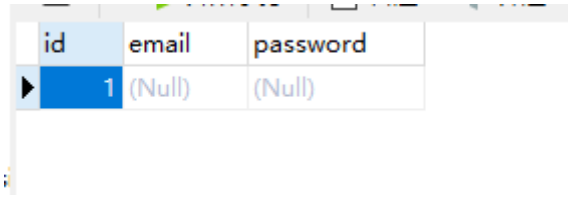

Figure 10 (b). Troubleshooting During the Test of the Interface

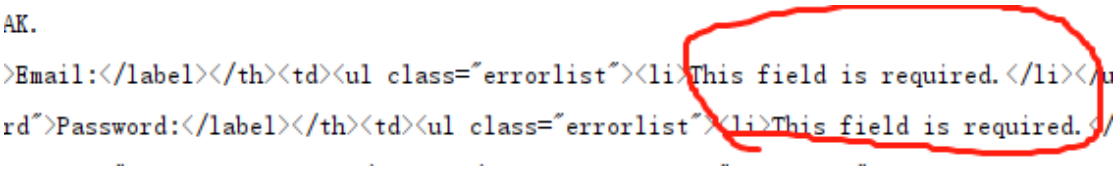

Figure 10 (c). Troubleshooting During the Test of the Interface

Analysis

For this problem, we are still dealing with it. Our idea is that the issue is raised by the Django API “request.POST”

```
def user_register(request):
    response = {}
    if request.method == 'POST':
        reg_form = RegisterForm(request.POST)
        print(reg_form)
        try:
```

Figure 11. Detail Concerning How We Tackle the Problem in Figure 10

So, it means our URL post request is accepted by the server correctly, but there is something wrong in our request body.

Andy’s Front-end Design

Not only Calvin developed the front-end, but I also developed one for sparing:

Here are the UI interface and test example:

Figure 12. Andy’s Design of UI

It is really similar to Tik Tok, so users may also like this one.
“test txt” may display some information related to this video.

If you want to download this front-end, please access another GitHub address:
https://github.com/JiashengYu618/seeu_front.git