Husky Planet

Husky planet is a platform where users are able to share ideas and discuss common problems in exploring any fields.

Architecture

Husky planet is a client-server system including frontend based on Bootstrap3, backend implemented with Django 2.2.9, and RDS instance provided by AWS.

Django Server

There are three critical components in django reflecting working pipeline and process logic.

- Viewer

Take login view as example.

```
def loginView(request):
    page = 'login'

    if (request.user.is_authenticated):
        return redirect('home')

    if (request.method == 'POST'):
        email = request.POST.get('email').lower()
        password = request.POST.get('password')
        # check if user exists
        try:
            user = User.objects.get(email=email)
        except:
            messages.error(request, "User does not exist")
        # authenticate
        user = authenticate(request, email=email, password=password)
        if (user is not None):
            login(request, user)
            return redirect('home')
        else:
            messages.error(request, "Username or password incorrect")

    context = {
        'page': page,
    }

    return render(request, 'base/login_register.html', context)
```

We get request data from `request` object. First, we check if the request type is `post`, return invalid request method when it is not required method `post`. Then we will check if the login email is in our database system. If so, then proceed to check password.

If all things go well, we will login user with `login` method. And meanwhile, redirect `home` page to user.

- Controller

Controller takes job of signaling critical methods or middleware to invoke third-party functionality.

- Models

```
# TODO: Customize User Model
class User(AbstractUser):
    name = models.CharField(max_length=200, null=True)
    email = models.EmailField(null=True, unique=True)
    bio = models.TextField(null=True)

    avatar = models.ImageField(null=True, default="avatar.svg")

    # when creating superuser, comment out following code
    USERNAME_FIELD = 'email'

    REQUIRED_FIELDS = []
```

We have defined four models in our system including `User`, `Message`, `Room`, and `Topic`. Take `User` as example, `User` model rewrite default admin model of Django. We define new login item as email rather than username before.

On great thing about Django.model is that it provides a function called `migration`, which is capable of tranforming Django models to Database objects in SQL language. Thereby, we are not bothered to write tedious SQL language.

Ralational Database System

In the system, we use remote RDS provided by AWS and select PostgreSQL as our default database system.

```
# setting.py

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': os.environ.get('DB_NAME'),
        'USER': os.environ.get('DB_USERNAME'),
        'PASSWORD': os.environ.get('DB_PASSWD'),
        'HOST': os.environ.get('DB_ENDPOINT'),
        'PORT': '5432'
    }
}
```

We are able to connect RDS with python database adapter `psycopg2`.

Online Demo

You are able to play with it on our demo server.

Developer: Minzhi Qu, Yanliu Wang
Email: quminzhi@gmail.com
Team: Horizon Team