KAZAKH KBTU BRITISH
TECHNICAL
UNIVERSITY

BACKEND FRAMEWORK (DJANGO)

Lesson 4

django REST framework

https://www.django-rest-framework.org

Django Rest Framework JWT

http://jpadilla.github.io/django-rest-framework-jwt/

What is JWT?

What is a JSON Web Token?

A JSON web token, or JWT (“jot”) for short, is a standardized, optionally validated and/or encrypted container format that is used to securely transfer information between two parties.

What is a JSON Web Token?

JWTs consist of three parts separated by dots (.), which are:

- header
- payload
- signature

JWT typically looks like the following:

```
xxxxx.yyyyy.zzzzz
```

Header

The header typically consists of two parts: the type of the token, which is JWT, and the hashing algorithm such as HMAC SHA256 or RSA.

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

this JSON is Base64Url encoded to form the first part of the JWT.

Payload

The second part of the token is the payload, which contains the claims. Claims are statements about an entity (typically, the user) and additional metadata.

```
{
  "sub": "1234567890",
  "name": “KBTU FIT",
  "admin": true
}
```

The payload is then Base64Url encoded to form the second part of the JWT.

Signature

To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that.

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

The signature is used to verify that the sender of the JWT is who it says it is and to ensure that the message was’t changed in the way.

Putting all together

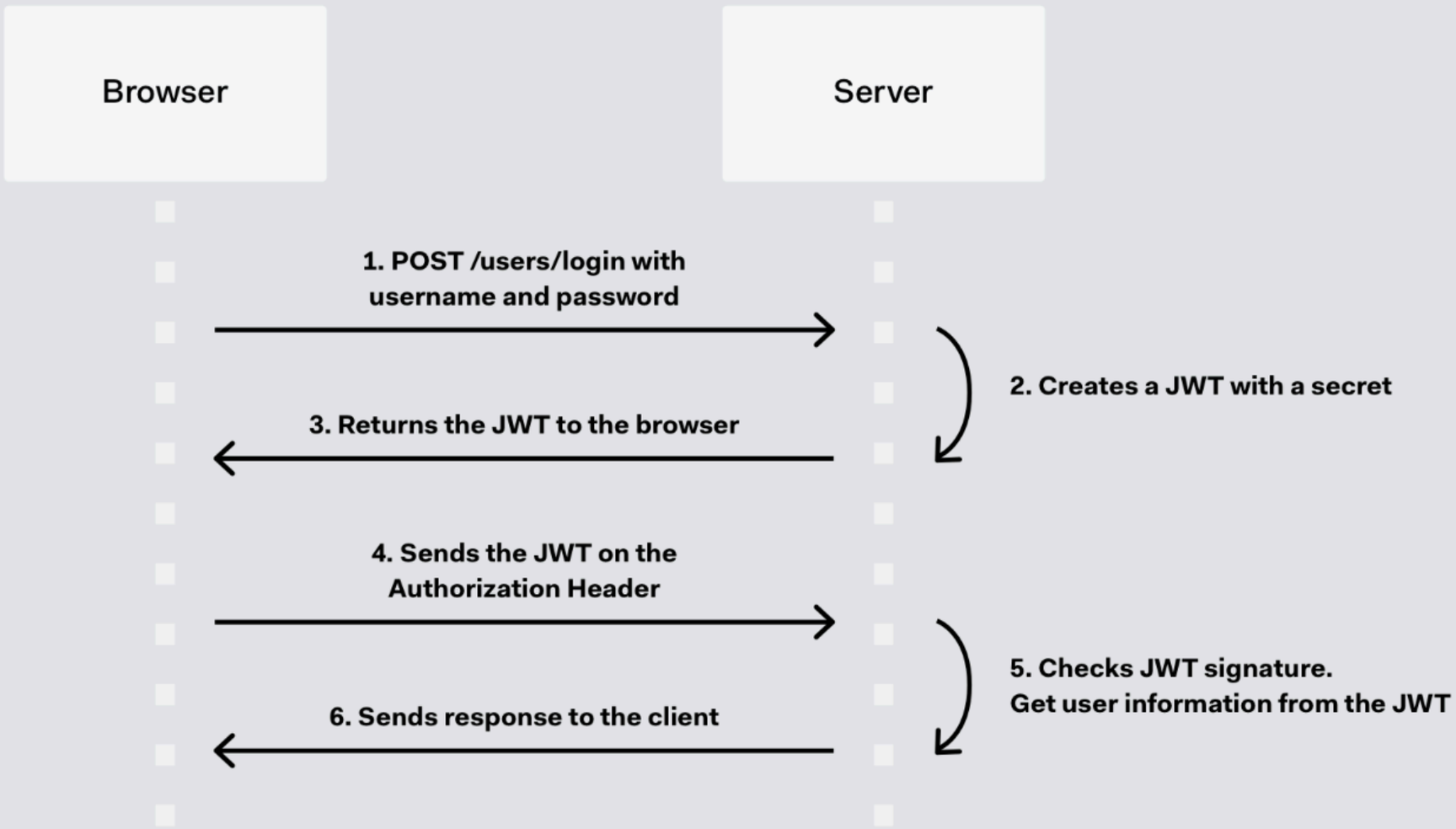
The following shows a JWT that has the previous header and payload encoded and it is signed with a secret.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Access to resource

header of each http request must contain

```
Authorization: JWT <token>
```


Browser
Server
1. POST /users/login with username and password
2. Creates a JWT with a secret
3. Returns the JWT to the browser
4. Sends the JWT on the Authorization Header
5. Checks JWT signature. Get user information from the JWT
6. Sends response to the client

Django Multiple User Types

Django Multiple User Types

1. No matter what strategy you pick, or what is your business model, always use one, and only one Django model to handle the authentication

2. Never user the built-in Django User model directly, even if the built-in Django User implementation fulfill all the requirements of your application

Django Multiple User Types

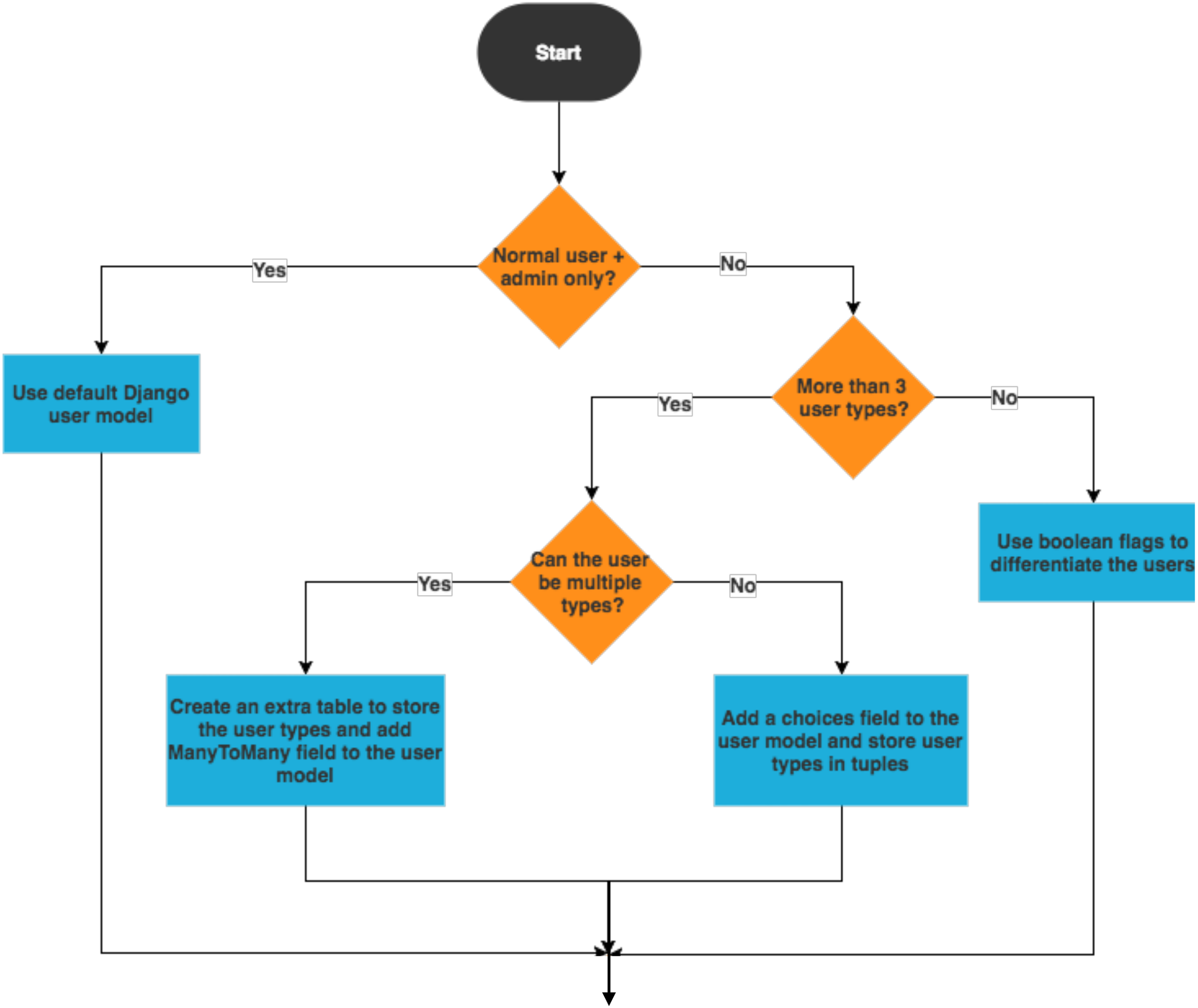
1. How much information specific to each type of user you need to maintain?

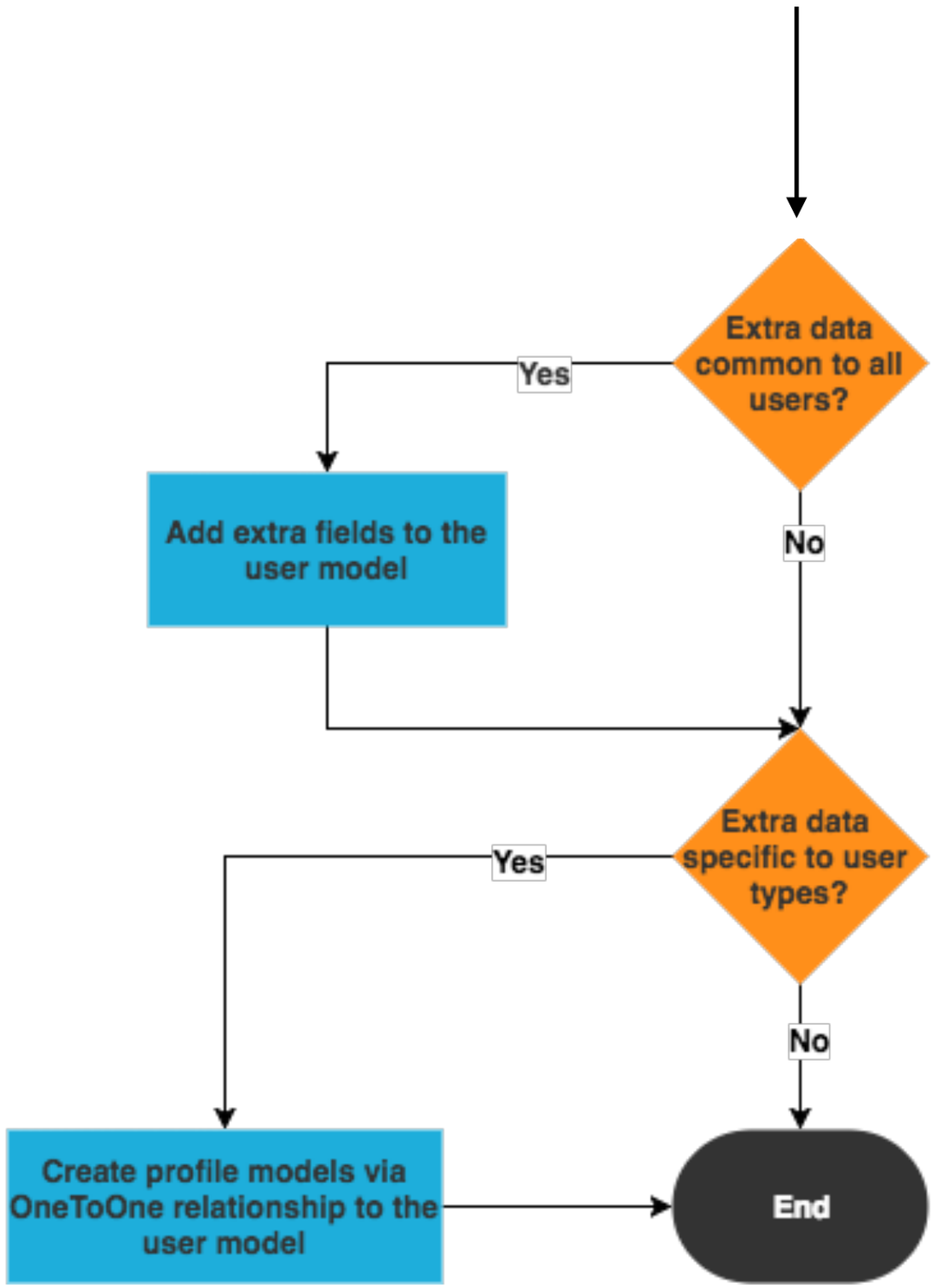
2. Can the users have more than one role in the application? E.g. can a User be a Student and Teacher at the same time?

3. How many different types of users the application will need to manage?

Django Multiple User Types

1. is_staff, is_superuser
2. multiple roles at the same time
    1. is_student, is_teacher
3. only one role
    1. choice field
4. many user types and multiple roles
    1. create an extra table and create a many to many relationship with user


Start
Normal user + admin only?
Yes
No
Use default Django user model
More than 3 user types?
Yes
No
Use boolean flags to differentiate the users
Can the user be multiple types?
Yes
No
Create an extra table to store the user types and add ManyToMany field to the user model
Add a choices field to the user model and store user types in tuples


Extra data common to all users?
Yes
No
Add extra fields to the user model
Extra data specific to user types?
Yes
No
Create profile models via OneToOne relationship to the user model
End

Questions?