Django Auth System

* Every Django application comes with authentication system.
  + The system allows users to login, logout, reset password etc.
  + The authentication system has multiple modules like groups, groups permission, user permissions.
  + Users table have all details every user has inside the app.
  + The app is found inside the apps inside setting file.

Middleware section (in setting app):

* When user make a request at some point there will be passed request object to the views file.
* The middleware functions are executed one by one.
* This middleware either return a response or pass the request object to the next middleware.
  + When middleware returns a response, it will stop executing the next middleware functions.
  + One of these middleware functions is authentication middleware, this middleware add a .user property to the request object.
  + It contains the user information.

Create a new User.

* Inside user table we have all information related to user and user authentication.
* Sometimes I want to customize the user model.
* To customize user model, I have two approaches one is inherit user model and extend it the other is create a profile and add one to one relation.
* When to use each one?
  + Extending user model:
    - when I want to add extra information related to authentication process.
  + One to one relation:
    - When I want to have information related to user such as address, etc.…
    - An example is student profile, employee profile or customer profile.

Configure inherited user model.

1. Model file (app/model): usually the app name should be core.
   1. Create the model and inherit it from AbstractUser that is imported from Django contrib auth model.
      1. From Django.contrib.auth.model import AbstractUser.
2. Add the app inside settings inside apps.
   1. Inside settings file create AUTH\_USER\_MODEL = ‘app.ModelName’
   2. In apps add appName.
3. Make migrations of the new extended model.
   1. I must create the inherited model before I create migrations because it will cause me troubles later on.
      1. The problem with this approach is that I will end up with migration problems because the initial 0001 uses User from auth to update table that I changed.
      2. One way to solve this is to drop and create database.
4. App Admin file.
   1. The admin site will remove the user because I was using base user class that I no longer use.
   2. To solve this problem, go to app where new User model live then:
      1. Register the new user model to the new admin user that is inherited from UserAdmin.
      2. From Django.contrib.auth.admin import UserAdmin
      3. Then inherit it to the new class
      4. Register my new User Model to the new admin class.
   3. To customize the admin fields, I can visit the UserAdmin model and see where fields are defined then I override the property I want in my case it’s fields.

Associate user model with profile.

* Rewatch later

Groups and permissions:

* Groups are used to create set of permissions and add users to that group instead of adding these permissions to users themselves.
* When I create my models Django add default permission to each model like crud operations.
* To benefit from that I first create group and assign permissions to that group so when I add users to it.
* Then I assign user.
* Next when user login will see these roles.

Customize permissions:

* To customize permission, I go to the model and create meta class.
  + Inside it I add permissions list of tuples.
  + Inside meta class I define permissions = [

(‘permission\_definition’, ‘do something’)

]