## Django Project Guide

*A cheat sheet provided by your course teacher :D*



## 

## 

## Make the path from Django to Browser

## 

## Create an ER diagram

1. Find the entities or classes of the project
2. Analyse the relationship between the entities
3. Use a professional tool to create the ERD
4. Add necessary attributes to every entity

Note: ER diagram might change during the development process

## Create Schema diagram

1. Use the standard rules to derive the schema diagram from ERD

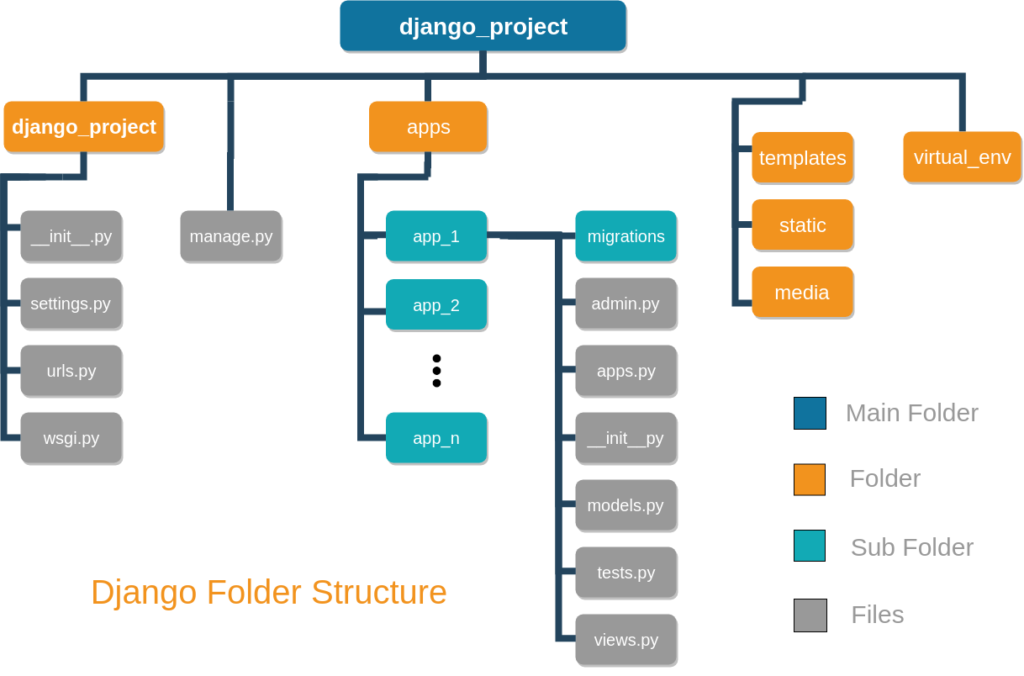
## Create Project and Upload to GitHub

1. One member should create a new project (without any virtual environment)
2. Share the project with your team member and me.

[Create and Upload a Django Project to GitHub](https://docs.google.com/document/d/18lDpMRtHBOBf-bVzoQ2MwzB-fpB4jbBthYj9_2Lf52c/edit?usp=sharing)



## Create APPs



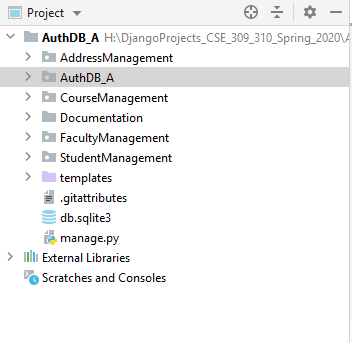
Divide the project into different apps based on the functionality. There is no right or wrong design here. But there are good and bad designs. Good design will help you to manage the project efficiently.



For example: If you want to develop an university management system, you might create the following apps

1. StudentManagement
2. FacultyManagment
3. CourseManagement
4. StaffManagement
5. Administration
6. ClubManagement
7. Accounts
8. EventManagement
9. Department

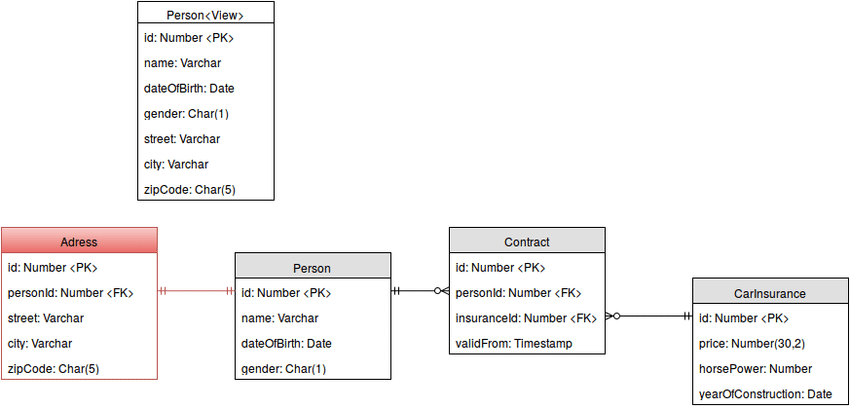
and so on



## Implement models

1. Every app has its own models.py
2. Implement models according to the schema diagram (within the appropriate app)
3. Import tables from other models if necessary
4. Use foreign key attribute to create the relationships among the tables
5. Add “on delete” attribute to the foreign key function
6. Determine the dependencies

For example: consider the following tables

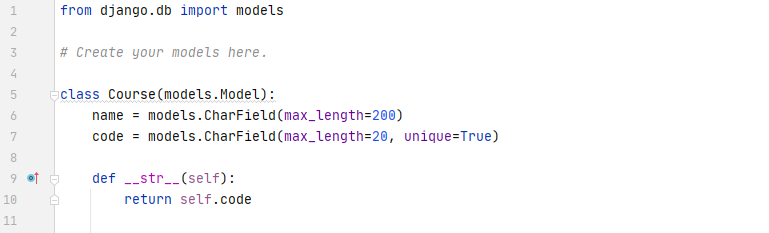


Person is independent as it is not pointing to another table. On the other hand, Address table is pointing to the Person table (with a foreign key personid)

### StudentManagement/models.py



### CourseManagement/models.py

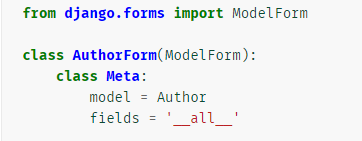


## 

## Create forms

1. Initially forms.py is not created inside an app. So, inside the app directory create a new form.py
2. Import model Class from models.py
3. Create a new class for form and inherit with ModelForm
4. Create a class Meta
5. Inside the Meta, add mode and fields
6. fields == ‘\_\_all\_\_’ means all fields of the model

Example:

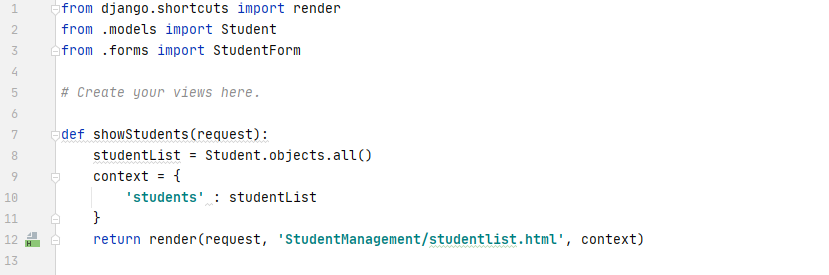


*Note: you don’t need to create forms for next week’s update.*

## Create views

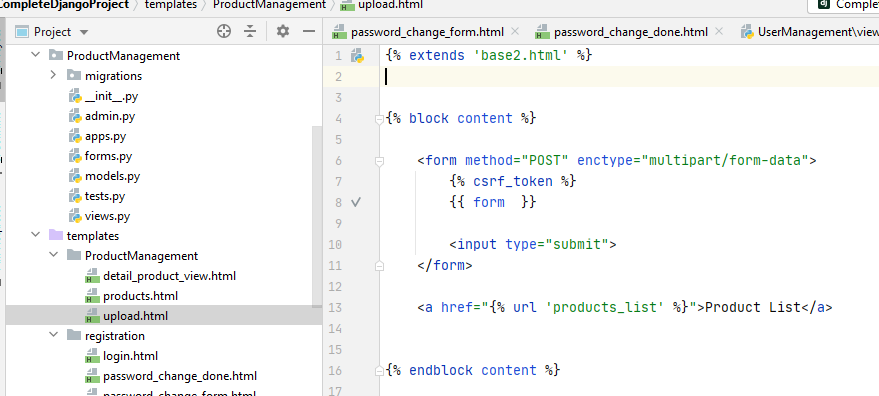
* Show database table in HTML page
  + Import the model Class
  + Define a function and search the table for all objects
  + Add the resulting query set in the context
  + Pass the context in the render function
  + Also add the request and html page in the render function

StudentManagement/views.py

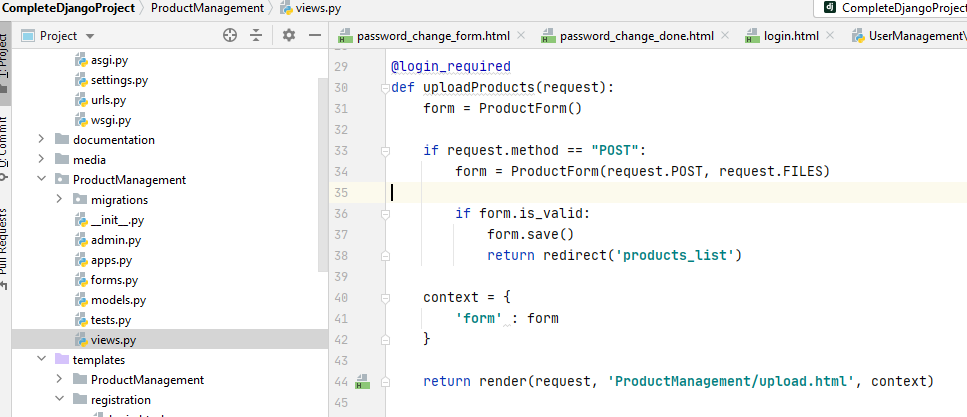
****

* Take input to database table from HTML page

1. Create anHTML page to take input. Just use form in the HTML page



1. Create a views function to upload books



1. Create an url for this function



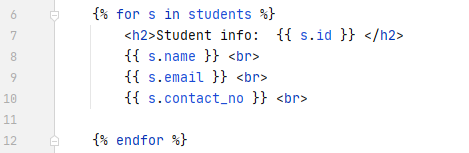


## Create HTML pages

1. Create directories with the APPs name
2. To show database table
   1. Inside the template/app add a HTML page (example: showStudents.html)
   2. Use a for loop to iterate through the query set
   3. Print the information using {{ }}
3. To insert the data into database table
   1. *Will be added later. Stay tuned!*

#### 

#### templates/StudentManagement/studentlist.html

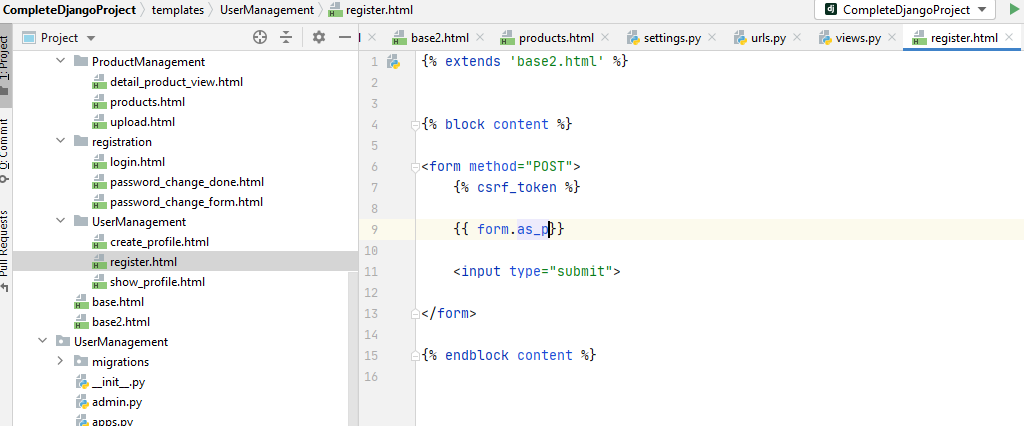


## When someone asks how is CSE 309 & 310 going ?

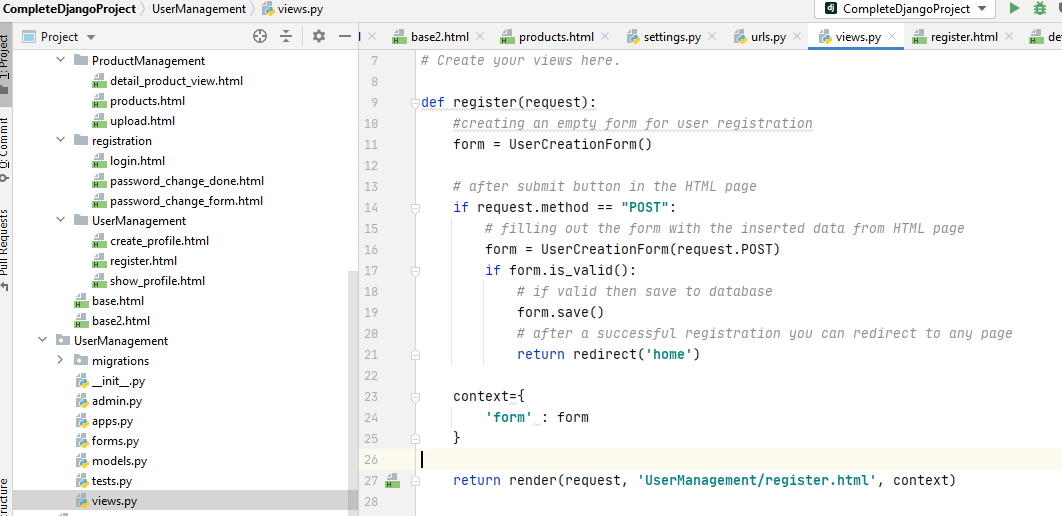


## User Registration (Sign up)

1. Create a UserManagement App
2. Create an HTML page for user registration



1. Write codes in views.py



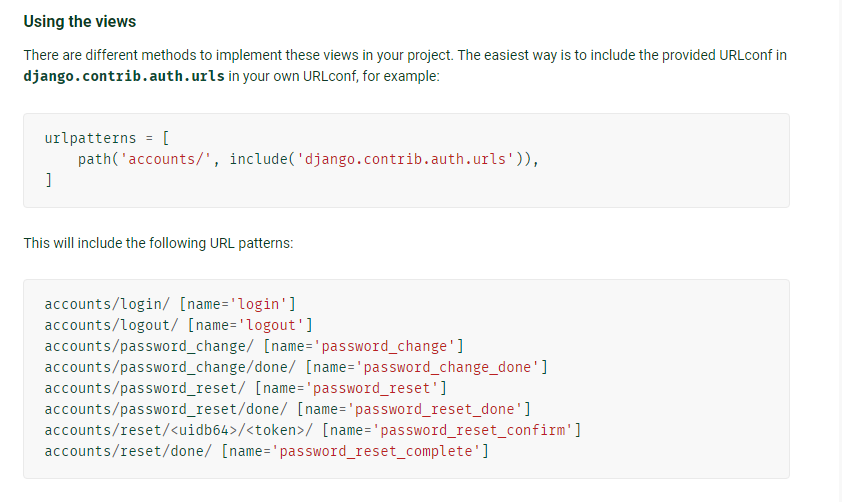
1. Create an url path and done!

path(**'signup/'**, user\_views.register, name=**'register'**),

## Login

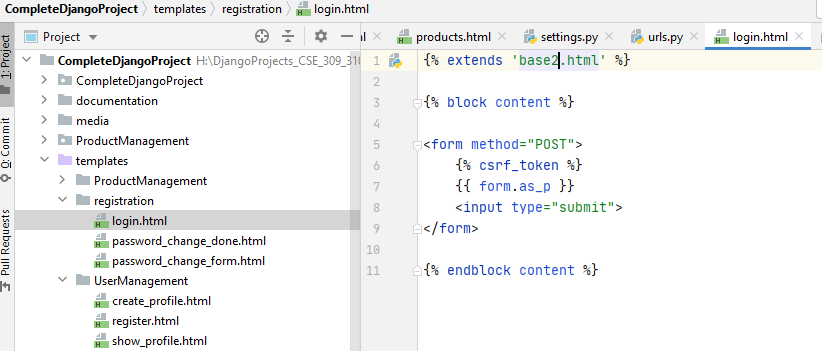
1. For authentication we will use the build in modules. Write this path in the urls.py

**path('accounts/', include('django.contrib.auth.urls')),**

****

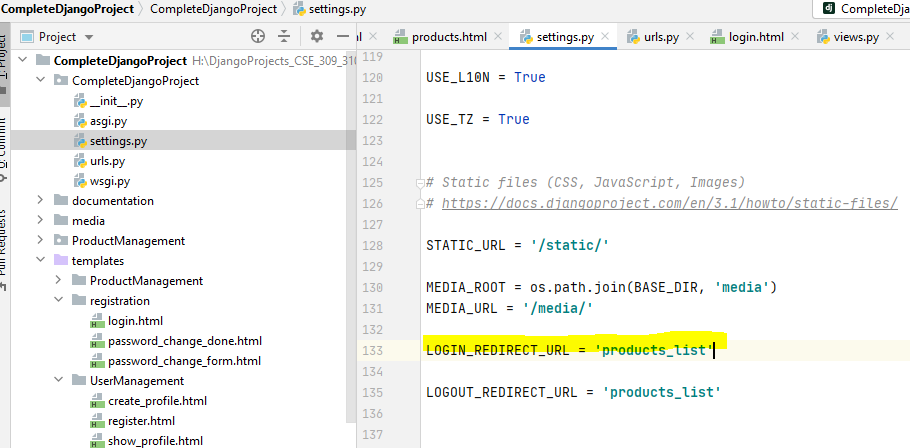
**Ref:** [**https://docs.djangoproject.com/en/3.1/topics/auth/default/**](https://docs.djangoproject.com/en/3.1/topics/auth/default/)

1. Create a “registration” directory in “templates” and add a “login.html” page.



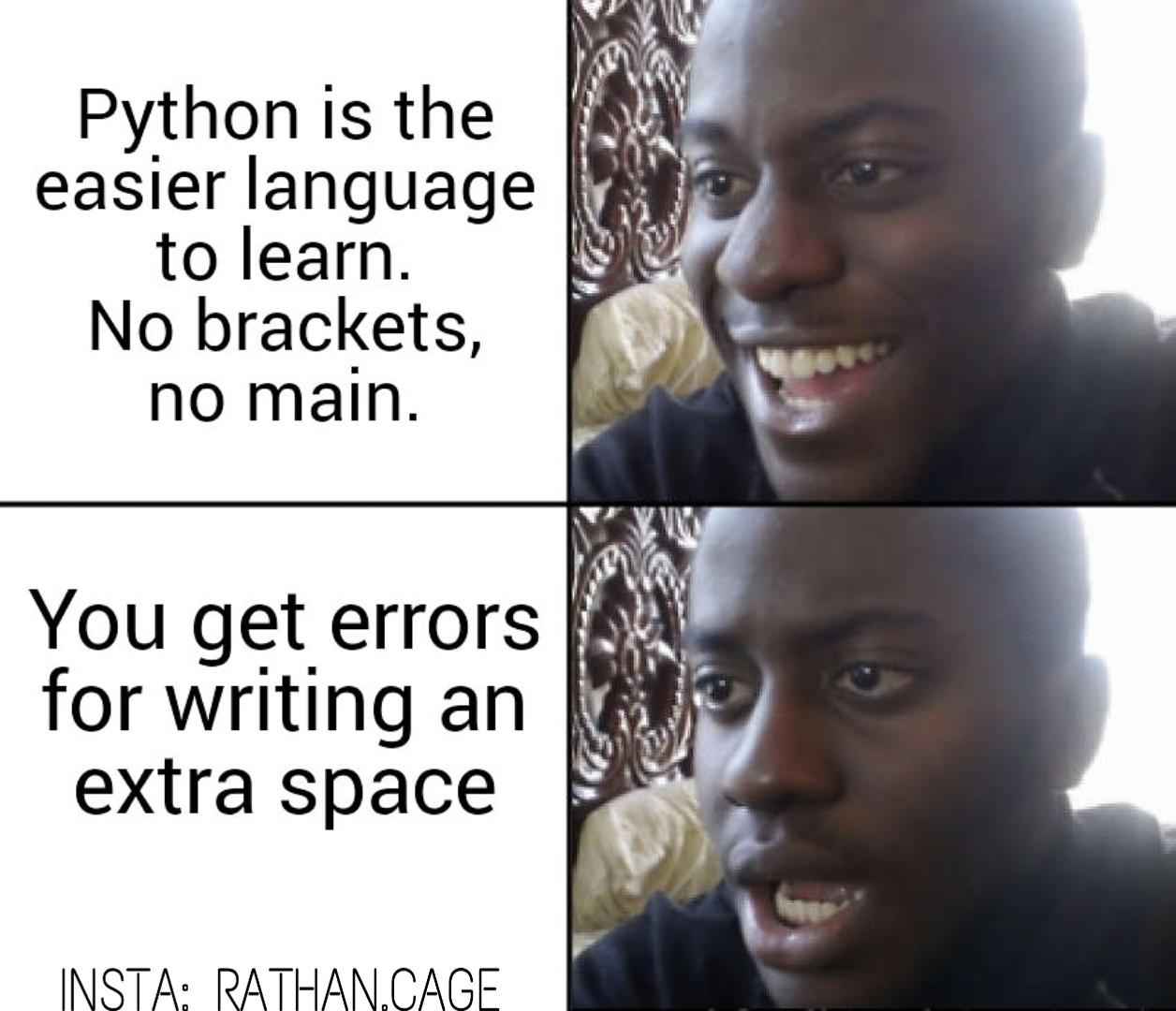
1. You need to set Login route by adding a this line in the setting.py

**LOGIN\_REDIRECT\_URL = '*your\_url*'**

****

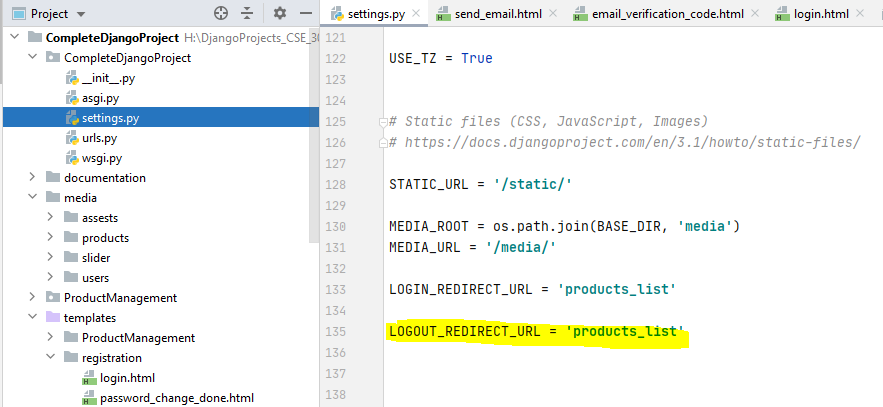
1. Now you can access the login page by using this url after the home url in browser

**/accounts/login/**



## Logout

1. Just add **LOGOUT\_REDIRECT\_URL = 'your-url' in settings.py**



1. Now you can access the logout function by using this url after the home url in browser

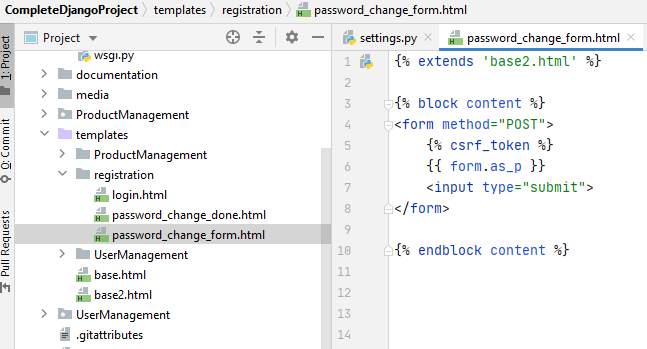
**/accounts/logout/**

1. Done!

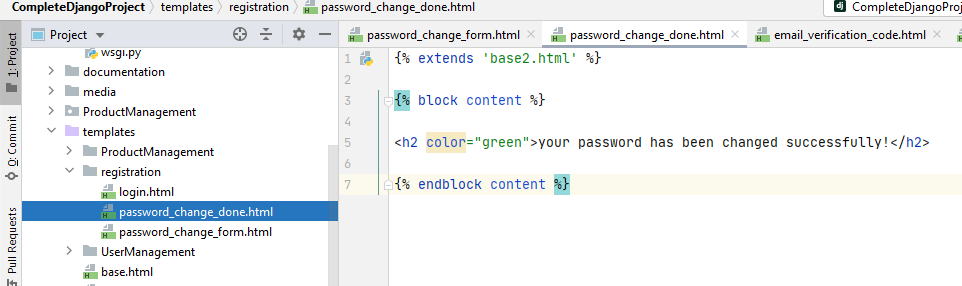


## Change Password

1. In “templates/registration” directory add a “password\_change\_form.htm” page



1. In “templates/registration” directory add a “password\_change\_done.html” page



1. Now you can access the password function by using this url after the home url in browser

**/accounts/password\_change/**