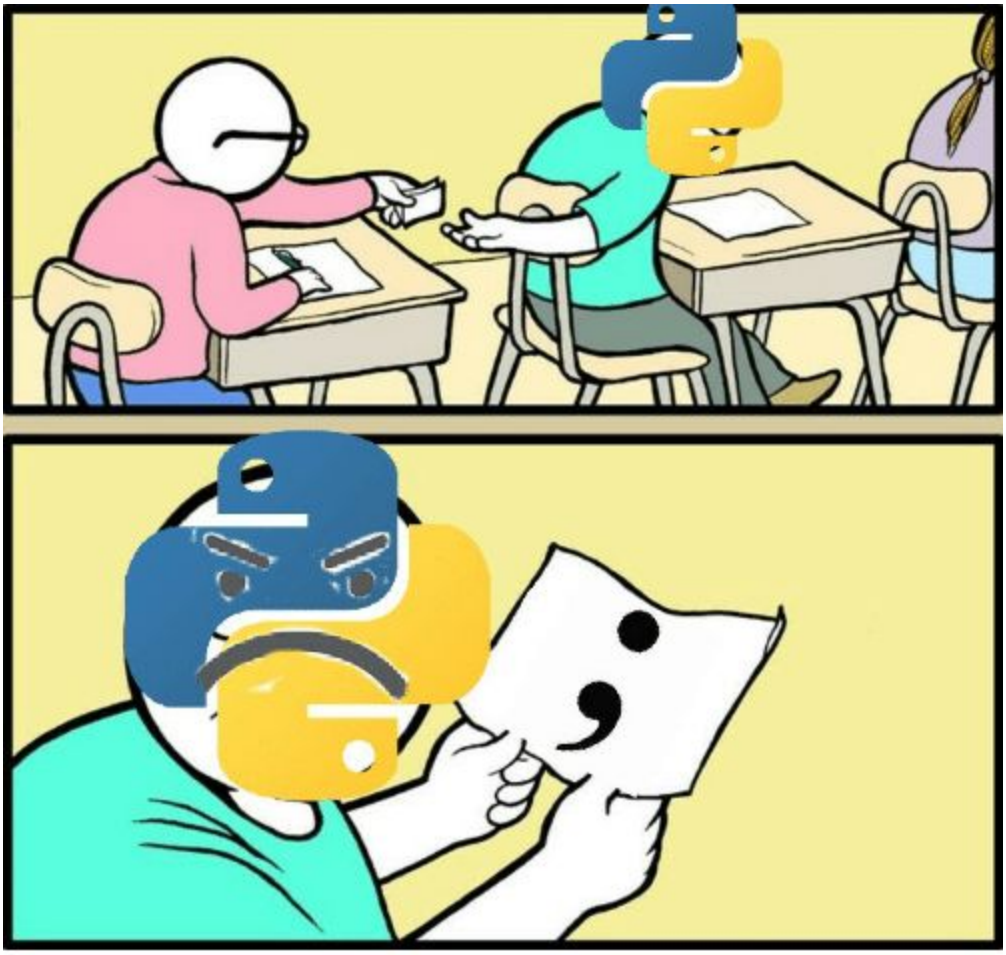
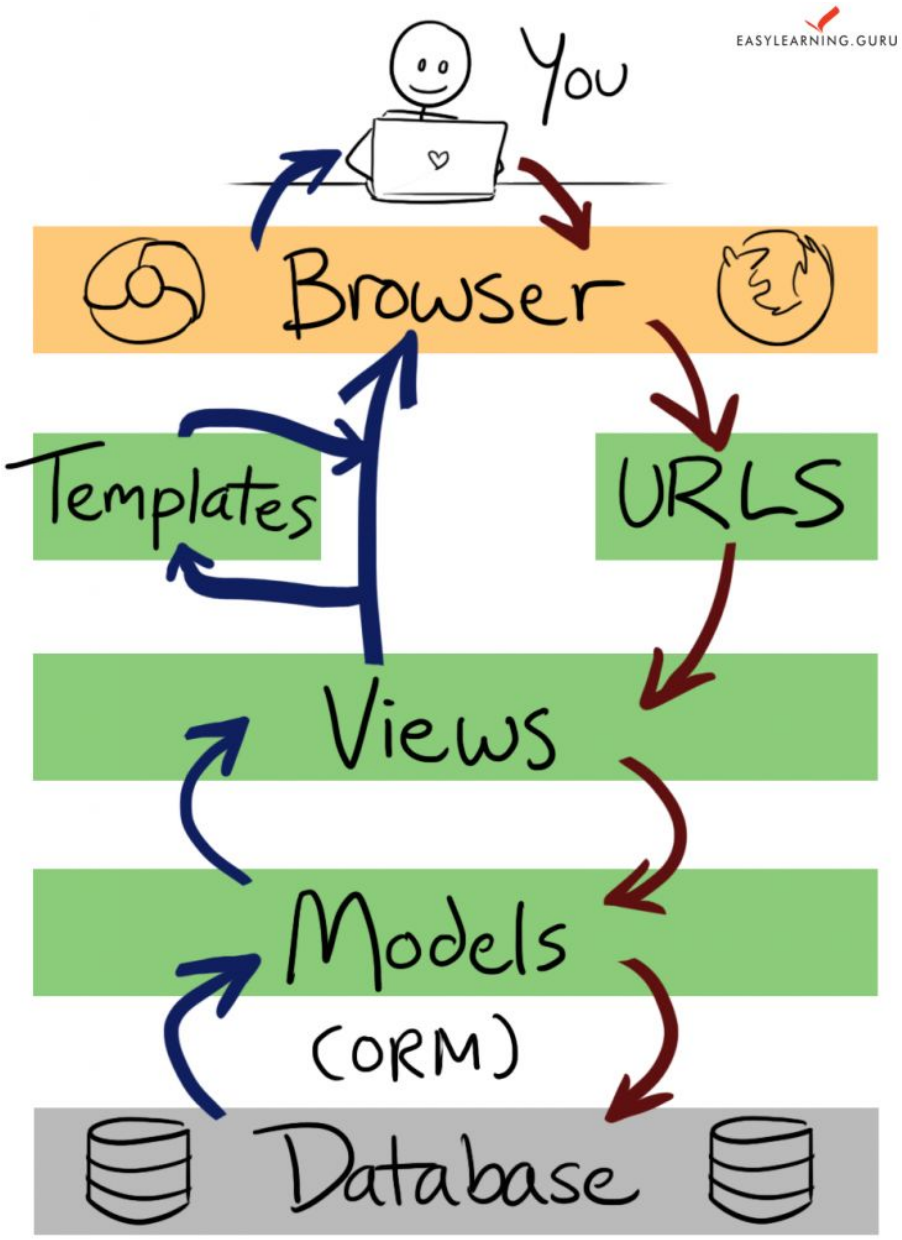
Django Project Guide

A cheat sheet provided by your course teacher :D



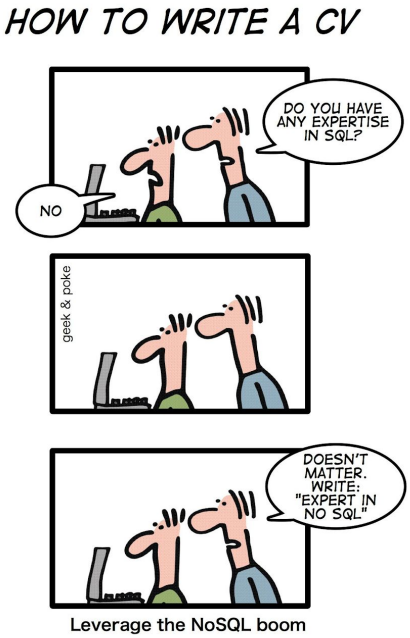
Make the path from Django to Browser



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

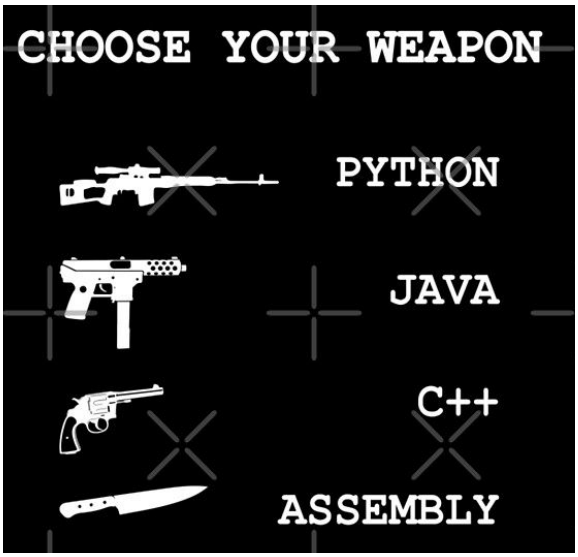


Create Schema diagram

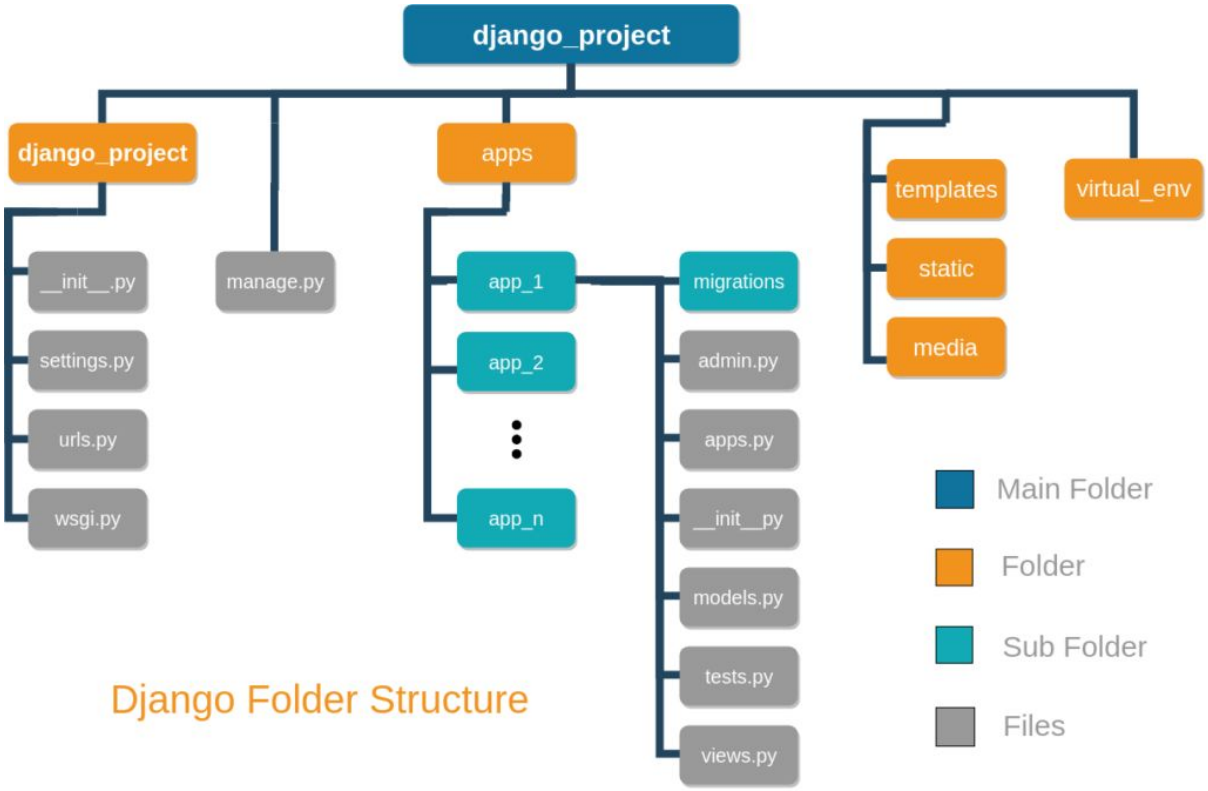
1. Use the standard rules to derive the schema diagram from ERD

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]



Create APPs

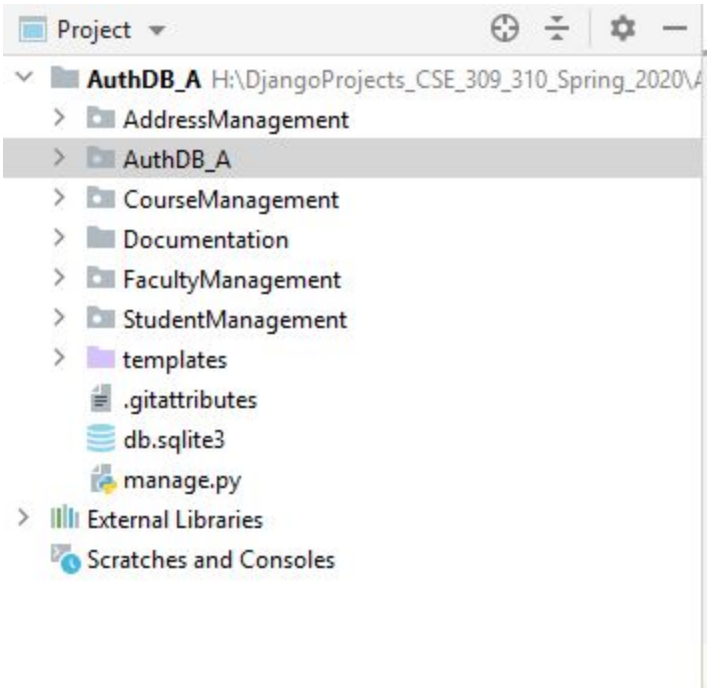


Divide the project into different apps based on the functionality. There is no right or wrong design here. But there are good and bad designs. Good design will help you to manage the project efficiently.

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

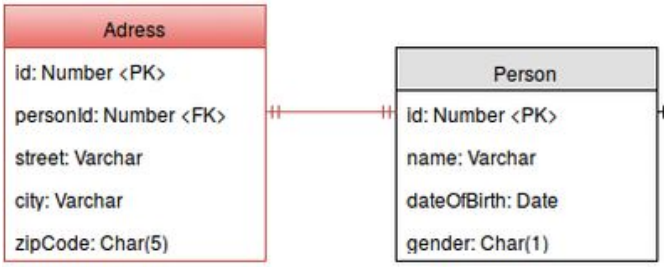
a. StudentManagement
b. FacultyManagment
c. CourseManagement
d. StaffManagement
e. Administration
f. ClubManagement
g. Accounts
h. EventManagement
i. Department

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

```python
from django.db import models
from FacultyManagement.models import Advisor
from CourseManagement.models import Course
# Create your models here.

class Student(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField(max_length=200, unique=True)
    contact_no = models.IntegerField(blank=True, null=True)

    advisor = models.ForeignKey(Advisor, on_delete=models.SET_DEFAULT, default=1)

    # s_id = models.IntegerField(unique=True)
    # id = models.IntegerField(primary_key=True)

    def __str__(self):
        return self.name


class Student_Course(models.Model):
    student = models.ForeignKey(Student, on_delete=models.CASCADE)
    course = models.ForeignKey(Course, on_delete=models.SET_NULL, null=True, blank=True)

    def __str__(self):
        return self.student.name + " : " + self.course.code

```

CourseManagement/models.py

```python
from django.db import models

# Create your models here.

class Course(models.Model):
    name = models.CharField(max_length=200)
    code = models.CharField(max_length=20, unique=True)

    def __str__(self):
        return self.code

```

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:

```
from django.forms import ModelForm

class AuthorForm(ModelForm):
    class Meta:
        model = Author
        fields = '__all__'
```

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

```python
from django.shortcuts import render
from .models import Student
from .forms import StudentForm

# Create your views here.

def showStudents(request):
    studentList = Student.objects.all()
    context = {
        'students' : studentList
    }
    return render(request, 'StudentManagement/studentlist.html', context)

```

When you only know HTML and develop a web application

Create HTML pages

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

templates/StudentManagement/studentlist.html

```
{% for s in students %}
    <h2>Student info:  {{ s.id }} </h2>
    {{ s.name }} <br>
    {{ s.email }} <br>
    {{ s.contact_no }} <br>

{% endfor %}
```

When someone asks how is CSE 309 & 310 going ?