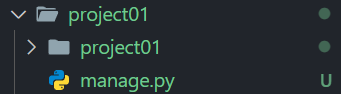
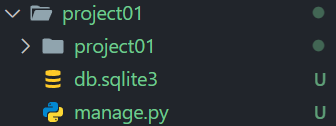
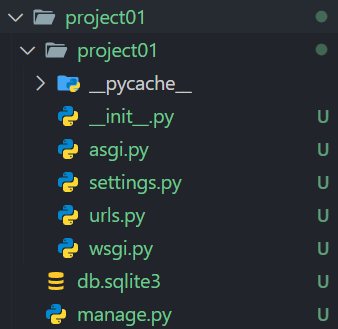
**Install**

We will use uv to create and a virtual environment where we install all libraries

1. Setup file : uv init <file-name> [ if filename not provide, it create files in current folder ]
2. Create a Virtual Env : uv venv [ venv is basically filename ]
3. Activate Virtual Env : .venv\Scripts\activate
4. Install Django : uv add Django

**Create Your Project directory**

1. django-admin startproject <project-name>
2. then go into you project : cd <project-name>  
   
3. start your project server : python manage.py runserver  
     
   you might get some migration error, but your server will run fine, also db.sqlite3 is created automatically

**Understanding File Structure**

**---> this is root level**

**---> this is inside level or project level**

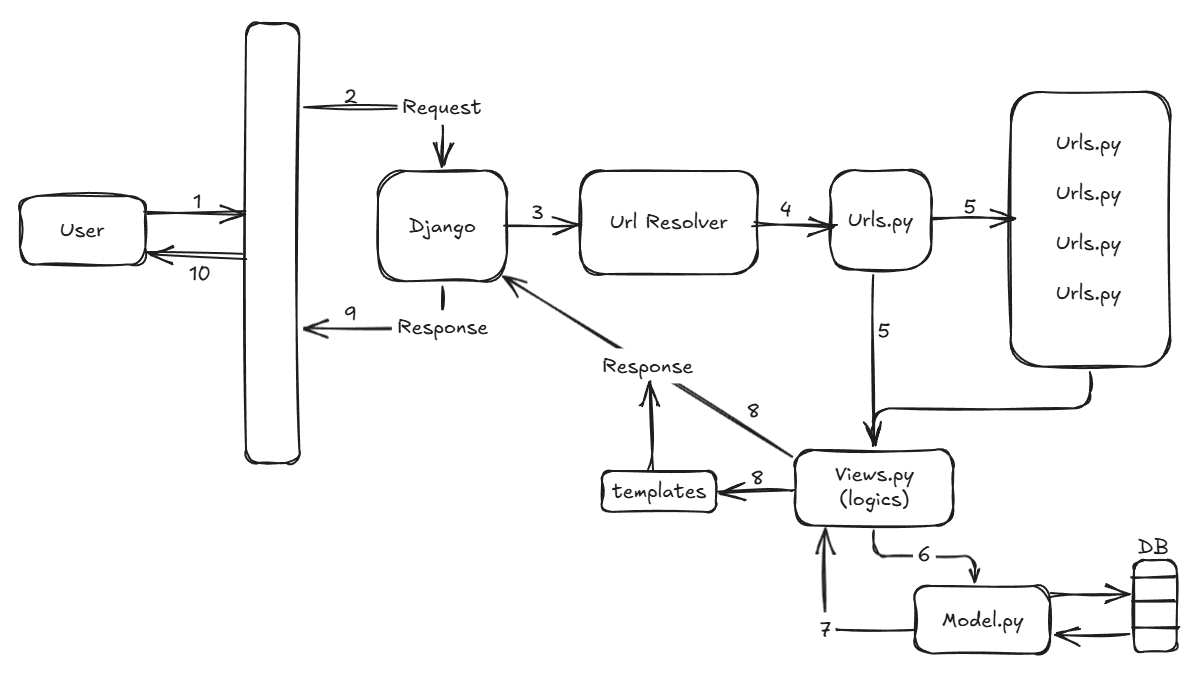
**---> this contain basic config**

**---> this contain routes**

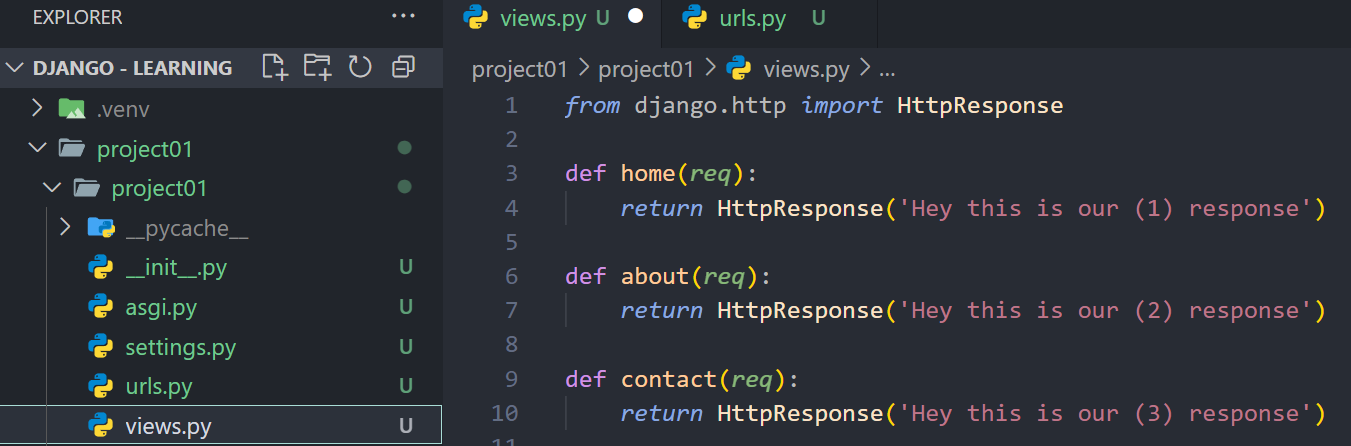
**---> (root lvl) contain database**

**---> (root lvl) contain start base function**

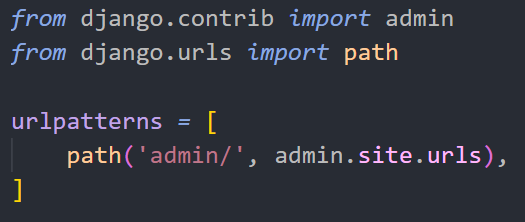
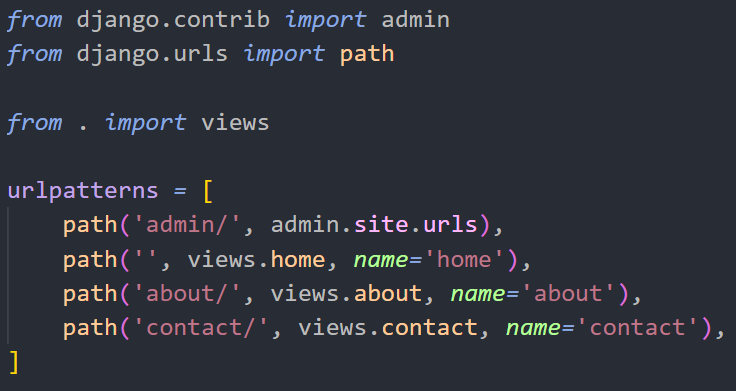
**How files Interact / workflow !!**



**Create & Setup views.py file**



Created a new views.py file and written some basic methods/functions in it   
we use ‘HttpResponse’ & ‘HttpRequest’ for req/res

Now Update the urls.py  
 

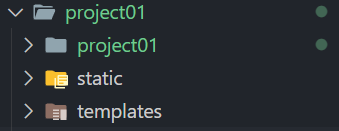
To

From

Now run the server and check the routes,

1. its easy first run you venv : ./venv/Script/activate
2. then go into your project root level
3. execute : python manage.py runserver
4. and that’s it

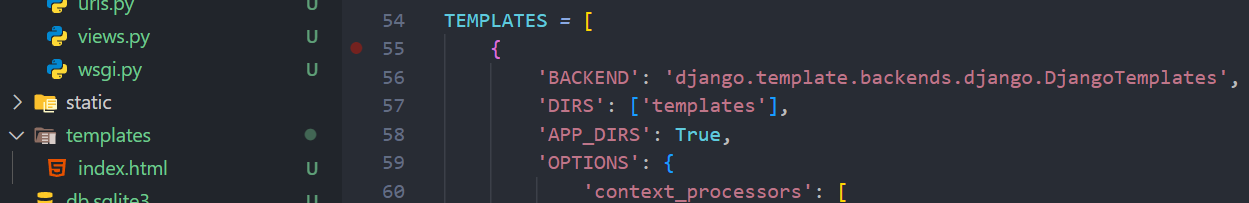
**Static & Templates (root level folder)**

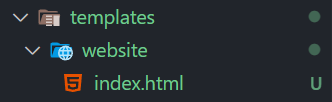
-> Static holds CSS and JavaScript files

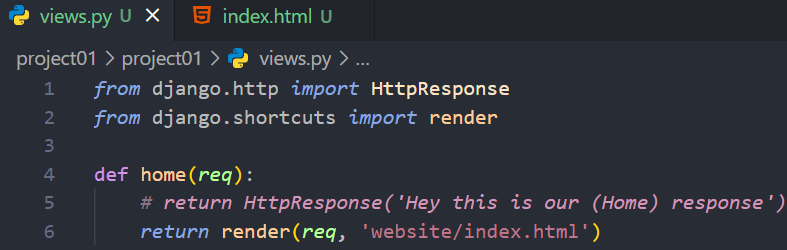
-> templates hold html files

Now, we also need to tell the settings about templates and static files  
if we don’t do this then we face a “TemplateDoesNotExist” Error



We included in the DIRS about template location,  


Lets move index to more inner folder,  
-> this is a good structure to follow at industry level

Now we again go template error, so we need to update path at render,  


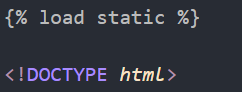
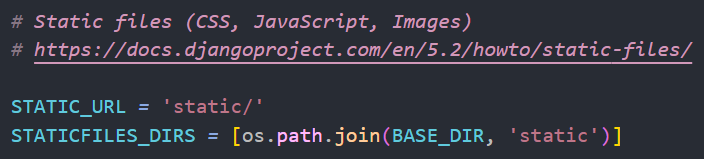
**Template Engine**

Now we want to insert the css into our index.html,

Can we do it like below,  


Nothing will happen, **So how to achieve it ?**

We need to inject the css file into html,

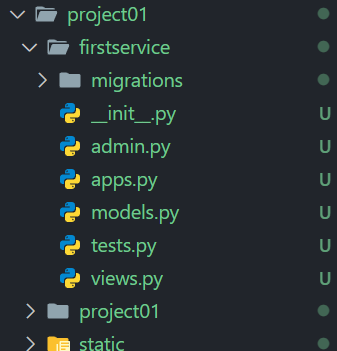
1. we need to load the static file  
   
2. now inject the static style.css file  
   
3. most important, **update setting**
   1. import os first
   2. then add staticfile\_dirs to it  
      

**Jinja – a template language** ([link](https://docs.chaicode.com/youtube/chai-aur-django/jinja-templates/) refer for common syntax docs)

**App in project**

They are standalone application and are part of the whole website, treat them as microservices

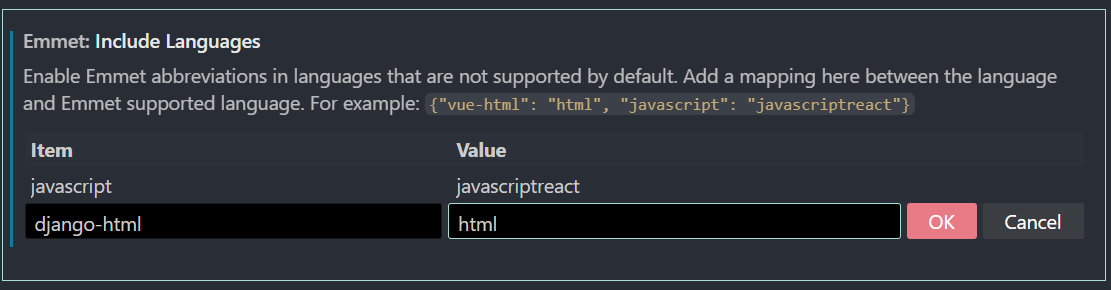
To create a app first go on to root level : python manage.py startapp <app-folder-name>  

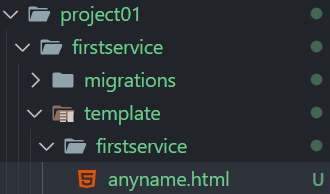

Now, we can able to see a new folder  
  
  
we notice we don’t have urls.py file here, so create it and copy-paste the content of urls.py from ‘*project01* ’  


Now, we have to make ‘aware’ our project that we created a new app

So go into setting.py and update the “INSTALLED\_APPS”  
-> we added our <app-file-name> in the last

-> there are some pre-installed app are their, they are known as “***cooked app***”

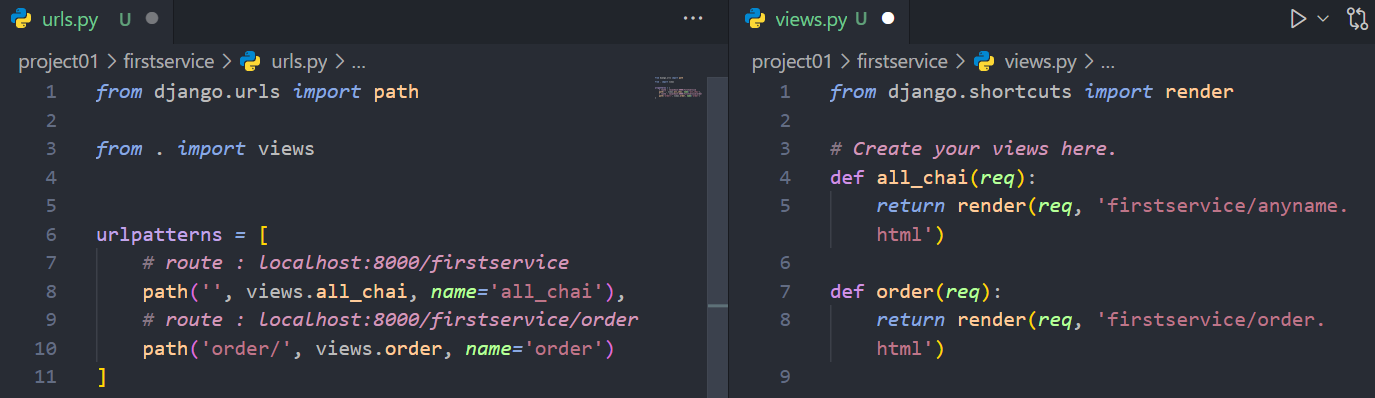
Note: we can’t use emmet in Django app html’s, so do a vscode setting to use emmet :  


Now Industry standard uses separate template folder for each app, we can use template at root level also but industry standard ya know it so,  
  
  
  
  
and yes we have to have same name as app-name

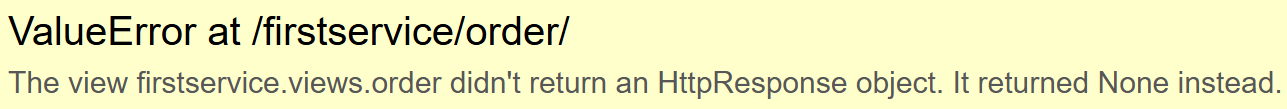
Now we need to connect urls.py to urls.py [ check diagram ]

We have to ***transfer the control*** from main urls.py to app urls.py, how ?  
Update the urls.py in main project01  
  
  
  
  
--> this is what we added

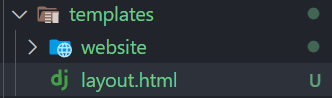
Below is how we created urls & views in the ‘firstservice’



\*note : if you write **template** instead of **templates** you will get ‘templateNotFoundError’

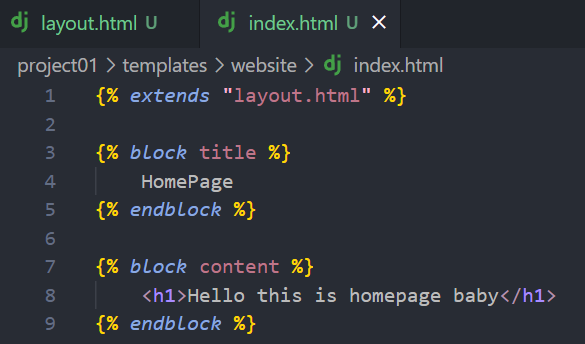
  
above error or tupple object error indicates you didn’t used render() while returning in views.py

**Django extension for keyword references**-> install the first one which came on typing Django XD

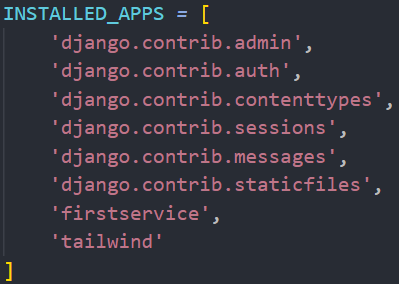
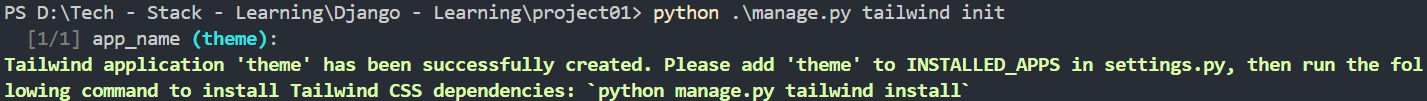
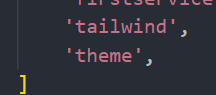
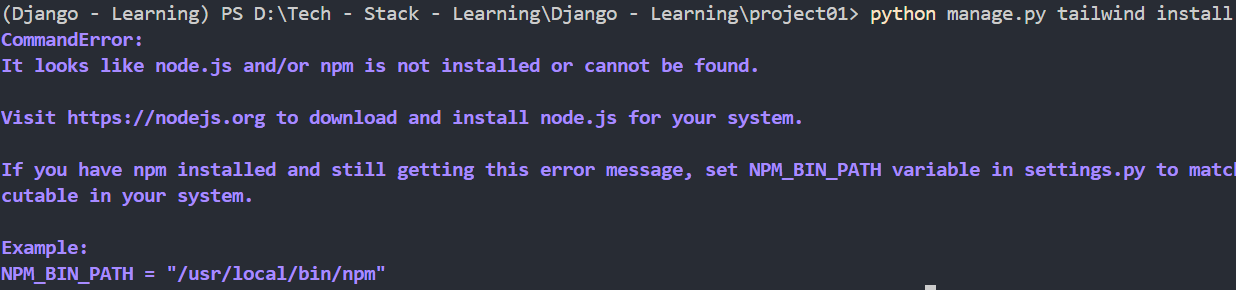
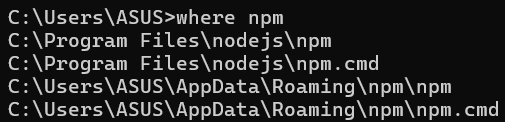
**Basic layout.html file**-> find it at root level

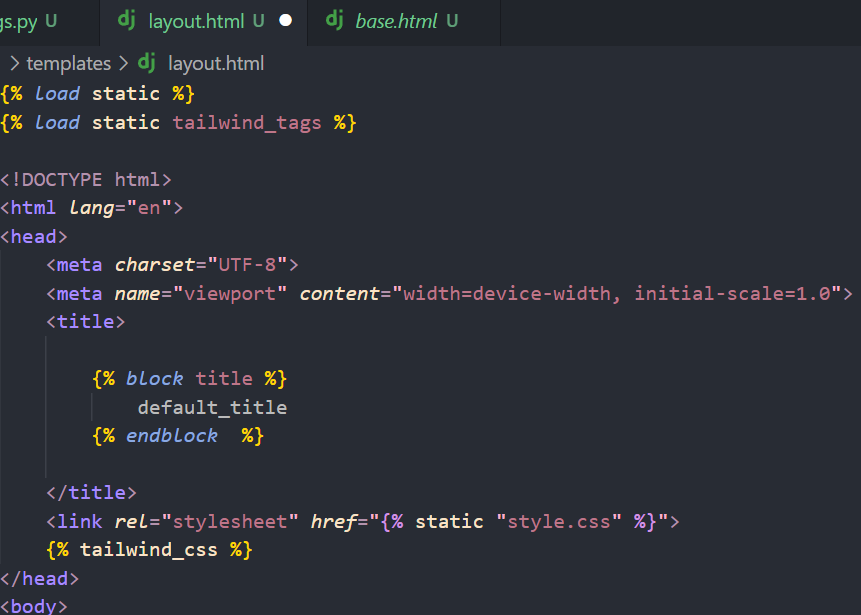
We have created it because we don’t want to provided repetitive code

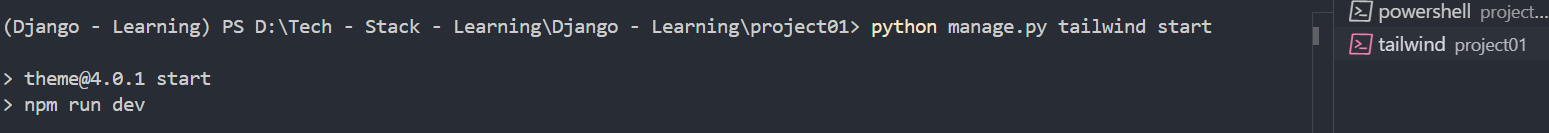
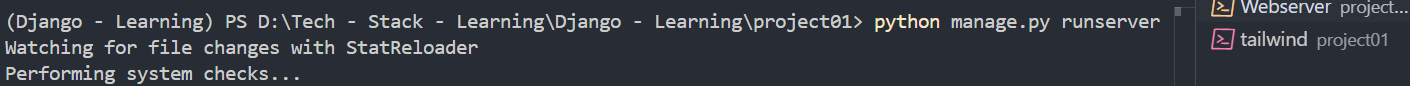
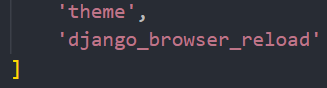
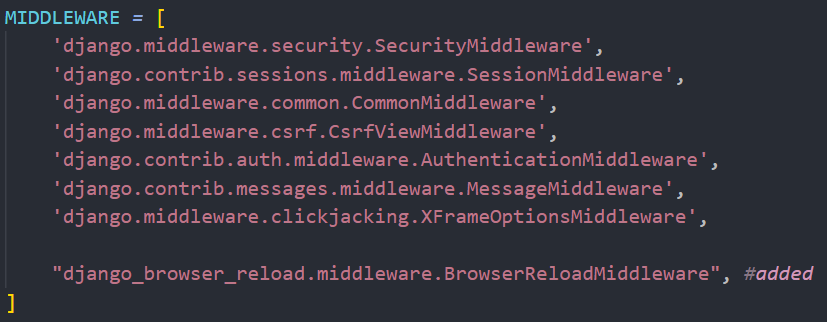
Now instead of writing big code in index.html,  


We can replace it with,  


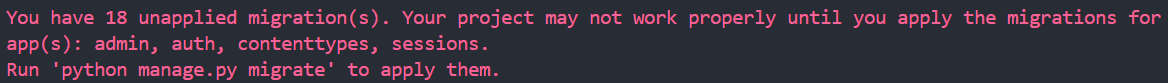
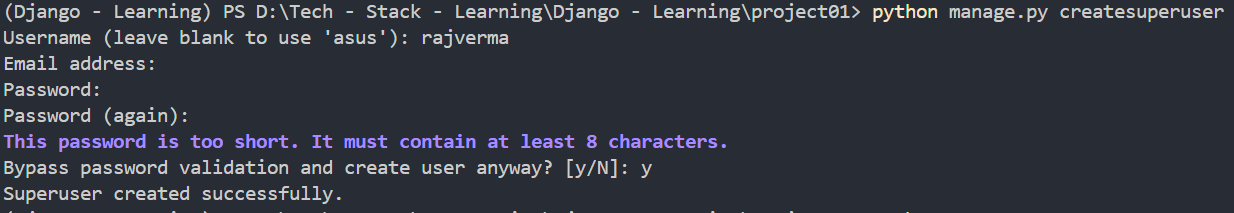
**Installing TailwindCSS**

1. add tailwindcss package : uv add django-tailwind
2. for auto/hot reloading when you write tailwind : uv add 'django-tailwind[reload]'
3. add ‘tailwind’ in app in setting  
   
4. init tailwind : python .\manage.py tailwind init  
   [ if you get cookiecutter error then install cookiecutter ]  
   
5. update setting > installed app  
   
6. now create this two things below installed app:  
   
7. now install the tailwind in manage.py: python manage.py tailwind install  
     
   if you are getting above error don’t worry, set npm path location in settings.py  
     
   No sure the path, go into global cmd and do, [ where npm ]  
     
   add the **npm.cmd** path on **program files**
8. update your layout.html of root lvl template



1. we nee to use to terminal one for tailwind another for website server  
     
   [ we rename & change terminal name & color, do it by right click on it ]  
   
2. update settings.py > installed app again with,  
   
3. Update the middleware in setttings.py  
     
   this is the **first** time we are adding something to middleware, don’t need to learn it as when you use some package this things mentioned their you just need to copy paste the content
4. Finally just update the urls.py on root project level  
   
5. Restart your servers to see the final changes

**Admin Panel**

1. Firstly the error we are seeing uptill now we have to remove it,  
     
   run : python manage.py migrate
2. Now create a super user : python manage.py createsuperuser  
   
3. Now go onto locahost:8000/admin page to access admin page

Note : to change password for a user : python manage.py changepassword <user\_name>

**Interaction with database**

> let’s create a basic model in an app i.e. firstservice > models.py

*from* django.db *import* models

*from* django.utils *import* timezone

*# Create your models here.*

class chaiVariety(models.Model):

    CHAI\_TYPES\_CHOICE *=* [

        ('ML', 'masala')

        ('GR', 'ginger')

        ('KL', 'KIVI')

        ('EL', 'Elaichi')

    ]

    name *=* models.CharField(max\_length*=100*)

    image *=* models.models.ImageField(upload\_to*=*'chais/')

    date\_added *=* models.DateTimeField(default *=* timezone.now())

    type *=* models.CharField(max\_length*=2*, choices*=*CHAI\_TYPES\_CHOICE)

**Note :** to handle images we need to install a plugin called as “pillow”  
e.g. uv add Pillow

Now update update some setting add media paths,  


Update urls.py in root level,

*from* django.contrib *import* admin

*from* django.urls *import* path, include

*from* django.conf *import* settings                *#added*

*from* django.conf.urls.static *import* static    *#added*

*from* . *import* views

urlpatterns *=* [

    path('admin/', admin.site.urls),

    path('', views.home, *name=*'home'),

    path('about/', views.about, *name=*'about'),

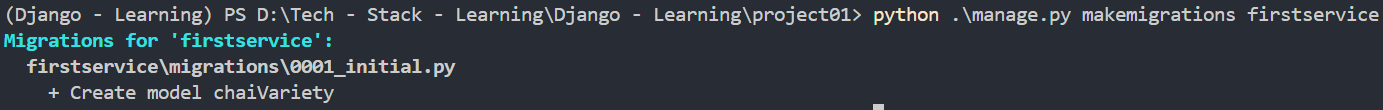
    path('contact/', views.contact, *name=*'contact'),

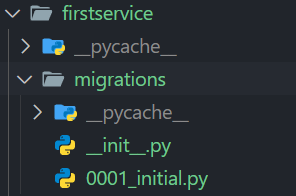
    path('firstservice/', include('firstservice.urls')),

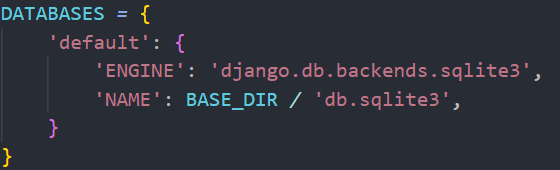
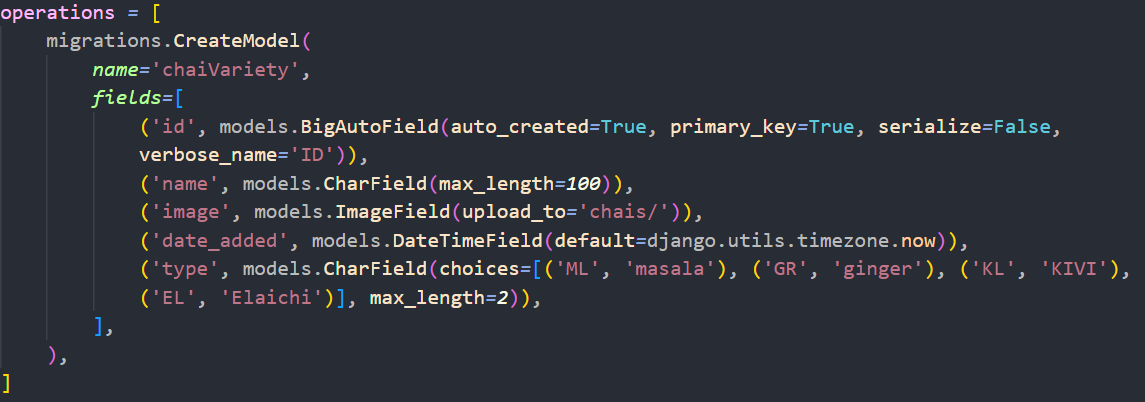
    path("\_\_reload\_\_/", include("django\_browser\_reload.urls")), *# try to put it in last*

] *+* static(settings.MEDIA\_URL, document\_root *=* settings.MEDIA\_ROOT) *#added*

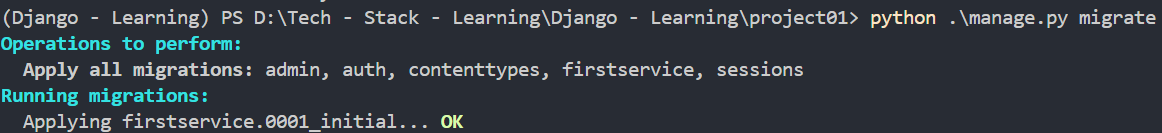
**Creation of migration after building model**

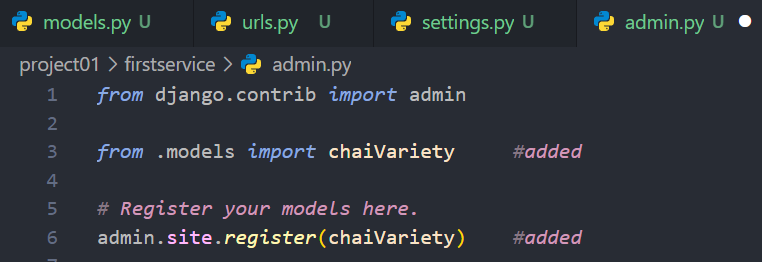
1. First create the migration of your app in project  


A new folder name “migration” will appear in your app,  


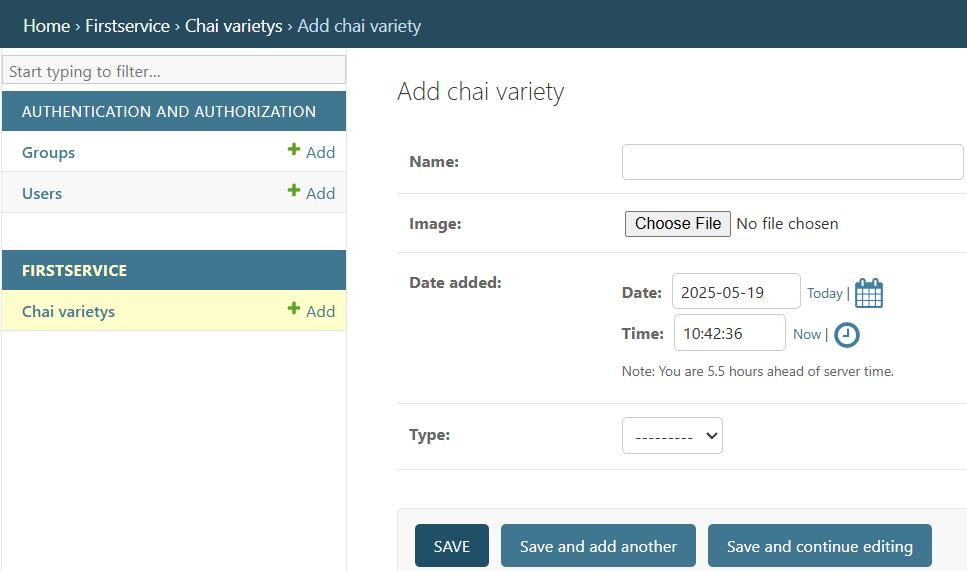
Now this 0001\_inital.py will contain the code for sql database fields,  
default database is sql as we saw in settings  
  
Now if we change this database do we need to change the model code also ?  
> the answer is **No**.  
Django uses ORM to talk with databases, and write code for specific database itself when we use migration  
for e.g. code in 0001\_inital.py contain is,  
  
it is the code for creating table and adding fields in table in sqlite

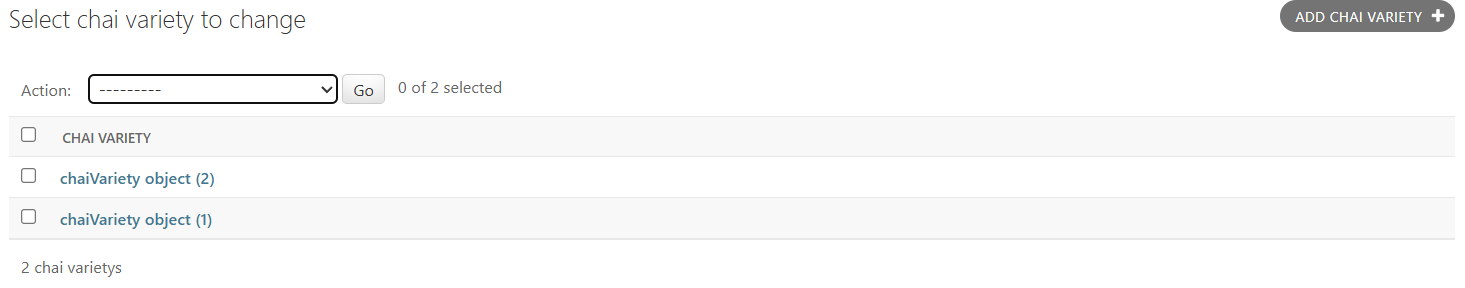
2. Now initiate your normal migration command

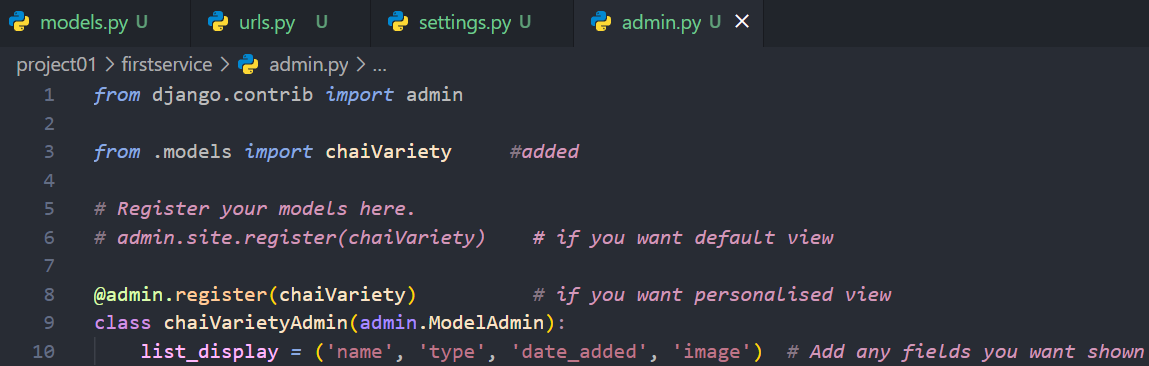


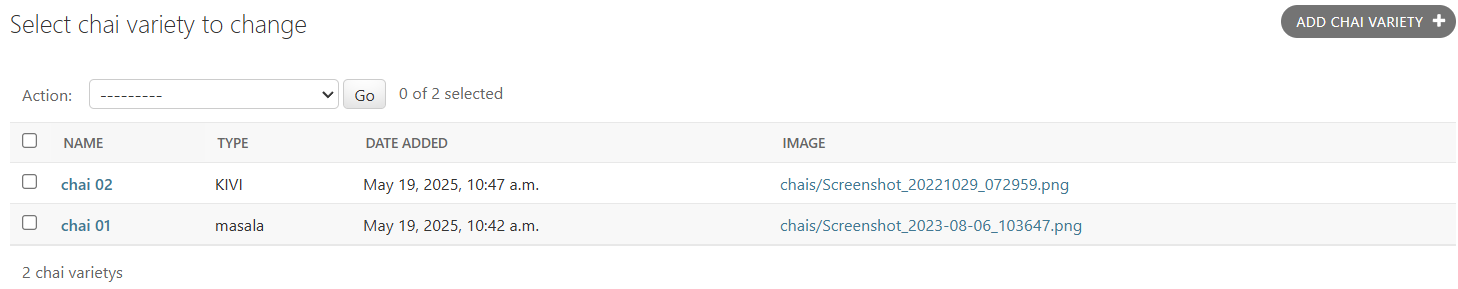
3. Register your model in admin.py  


**Configuring Admin Page**

Below is how our detail filling form looks like,  


Now lets add some chai,  
  
so this is how our objects looks like in admin panel, **but** we want a more clear look about object name and want to add more fields while viewing it, SO ADD BELOW:



So we used decorator and created a class in above format, result looks like :  


**Access data from model > views > anyname.html**

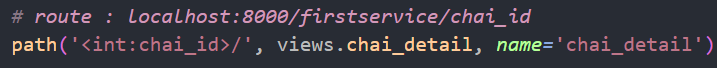
We passed the data to the render component through below technique,



We used key : value pair to pass the data, ‘data’ i.e. key can be of any name

  
we updated our anyname.html file in the firstservice app

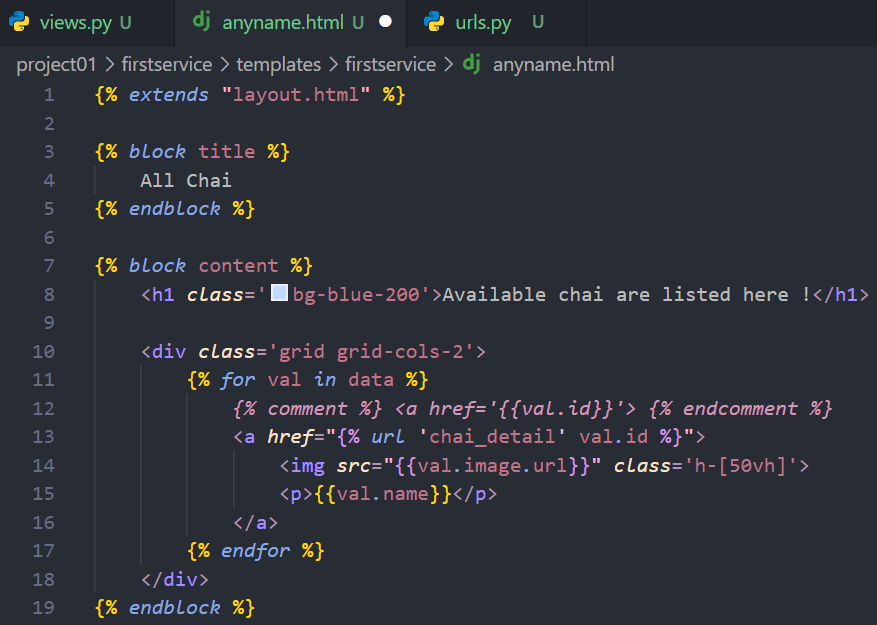
**How to have a Dynamic link**

**First update urls.py  
  
<> : using diamond bracket we can create the dynamic link**

Note: if you update your model, let say you forgot to add description field,  
then you have to do migrations again,  
1. python manage.py makemigrations <app\_name>  
2. python manage.oy migrate

Now update the anyname.html, because we want to pass the object id

> We created a page where user can create on image and go onto next Page to see that specific object/image detail



If we directly used {{ val.id }} then it would be good isin’t ?  
-> it will work in some case but,

**1. {{ val.id }}**

* **It simply outputs the value of val.id (which is likely an integer or string).**
* **The result may not form a valid URL unless val.id is already a full or relative URL.**

**2. {% url 'chai\_detail' chai.id %}**

* **It generates a URL for the view named 'chai\_detail', passing chai.id as an argument. Used for routing.**

Now create a html page where you want to redirect the request

In our case we create : ‘chaiDetail.html’

{% *extends* "layout.html" %}

{% *block* title %}

    Chai Detail

{% *endblock* %}

{% *block* content %}

    <h1 *class*='bg-blue-200'>Current chai created date & type are</h1>

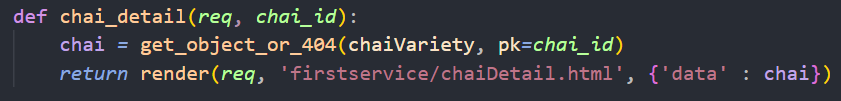
    <p>

        {{data*.*date\_added}} - {{data*.*type}}

    </p>

{% *endblock* %}

Now update your firstservice > views.url

  
we are passing the chai\_id in this object to search for the object in whole database setting id as primary key to find

**How to register a class Old vs New**

Old: define class first then register it , i.e. register it in the end

admin.site.*register*(chaiVariety,chaiVarietyAdmin)

New:

@admin.register(chaiVariety)          *# New Method to register using decorators*

class chaiVarietyAdmin(admin.ModelAdmin):