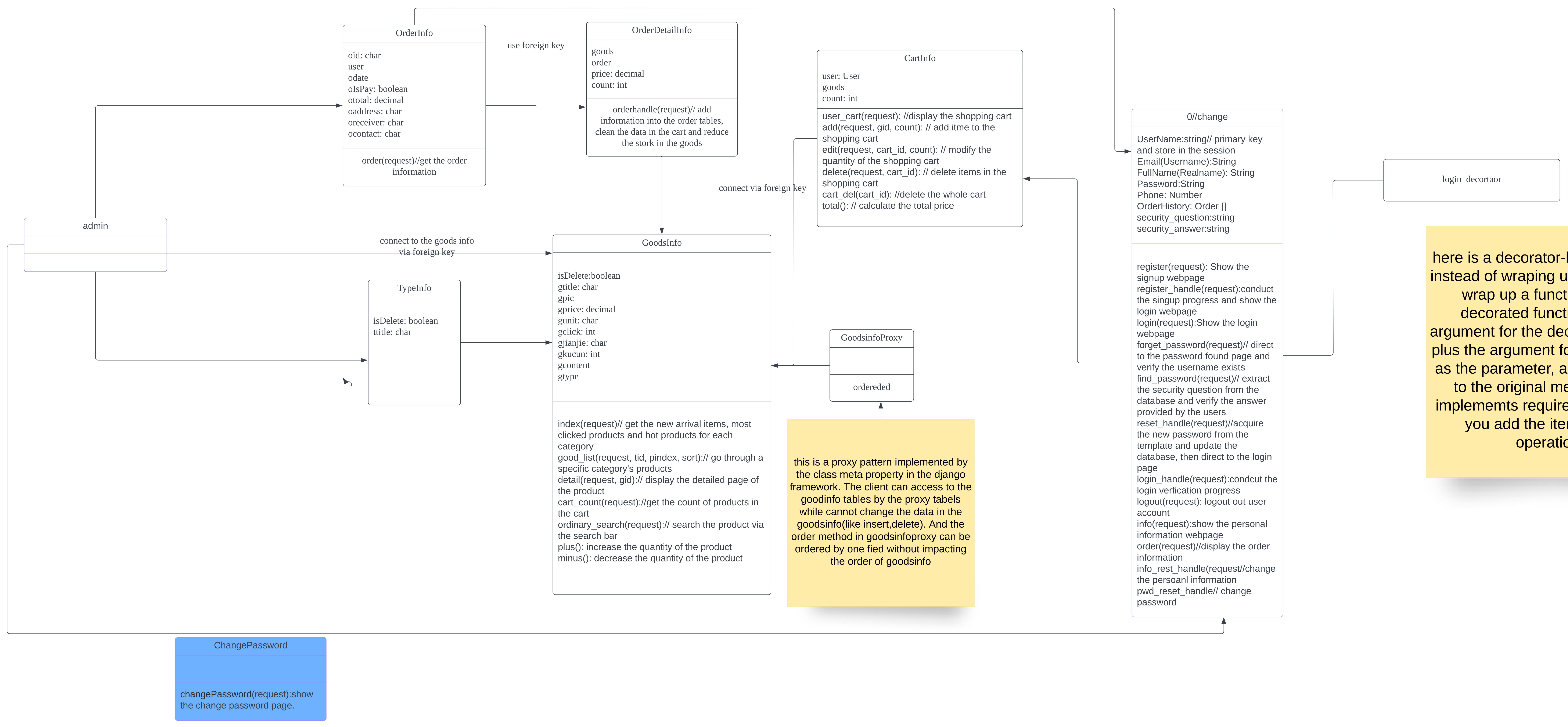
after our development, we just found that Django is not OO framework, instead, it's a functional framework. The reason is that a lot of objects like request, it's instantiated by the framewok automatically, all the request, information just encapsulated in a dict-like format in the request, and it just required the function to accept the request, pull out sessons, cookies, and other information from the request, and then pull put the database from the ORM and encapsulate the information into the context, then render to the template and display to the users. There is no place for the client to instantiate the class by hand, the only one need to be instantiated is the obejct of ORM when insterting a new row. so it's hard for us to implement design pattern. However, we try to understand the patteren that built in the django

For MVC(MTV) pattern, please refer the readme file in our repo

The Django views themselves are a kind of command pattern, since the client send the request from the front end and the server(invoker) passes it to the route(urls, command), the route will let the specific function(receiver) in the views to process the request,, the template don't know who will process the request, it just know the command. it's also can be think as 'broker' like patterm, since the subject sent the request to the broker, and the broker redict the request to the observer(view).

the admin is not a real class, but it canbe think as a class, it just manage the data in the tables via built-in API. we just need to set and decide which tables that admin can access and what fields it can edit



OrderInfo
oid: char
user
odate
oIsPay: boolean
ototal: decimal
oaddress: char
oreceiver: char
ocontact: char
order(request)//get the order information
use foreign key
OrderDetailInfo
goods
order
price: decimal
count: int
orderhandle(request)// add information into the order tables, clean the data in the cart and reduce the stork in the goods
CartInfo
user: User
goods
count: int
user_cart(request): //display the shopping cart
add(request, gid, count): // add itme to the shopping cart
edit(request, cart_id, count): // modify the quantity of the shopping cart
delete(request, cart_id): // delete items in the shopping cart
cart_del(cart_id): //delete the whole cart
total(): // calculate the total price
connect via foreign key
0//change
UserName:string// primary key and store in the session
Email(Username):String
FullName(Realname): String
Password:String
Phone: Number
OrderHistory: Order []
security_question:string
security_answer:string
register(request): Show the signup webpage
register_handle(request):conduct the singup progress and show the login webpage
login(request):Show the login webpage
forget_password(request)// direct to the password found page and verify the username exists
find_password(request)// extract the security question from the database and verify the answer provided by the users
reset_handle(request)//acquire the new password from the template and update the database, then direct to the login page
login_handle(request):condcut the login verfication progress
logout(request): logout out user account
info(request):show the personal information webpage
order(request)//display the order information
info_rest_handle(request//change the persoanl information
pwd_reset_handle// change password
login_decortaor
admin
connect to the goods info via foreign key
TypeInfo
isDelete: boolean
ttitle: char
GoodsInfo
isDelete:boolean
gtitle: char
gpic
gprice: decimal
gunit: char
gclick: int
gjianjie: char
gkucun: int
gcontent
gtype
index(request)// get the new arrival items, most clicked products and hot products for each category
good_list(request, tid, pindex, sort):// go through a specific category's products
detail(request, gid):// display the detailed page of the product
cart_count(request)://get the count of products in the cart
ordinary_search(request):// search the product via the search bar
plus(): increase the quantity of the product
minus(): decrease the quantity of the product
GoodsinfoProxy
ordereded
ChangePassword
changePassword(request):show the change password page.

this is a proxy pattern implemented by the class meta property in the django framework. The client can access to the goodinfo tables by the proxy tabels while cannot change the data in the goodsinfo(like insert,delete). And the order method in goodsinfoproxy can be ordered by one fied without impacting the order of goodsinfo

here is a decorator-like pattern, but instead of wraping up a class, it just wrap up a function, pass a decorated function and the argument for the decorated function plus the argument for the decorator as the parameter, add the function to the original method. which implememts required log in before you add the items or othe operations