In [ ]:

```
from flask import Flask, request, render_template, redirect, url_for

################################          base          ##############################
import pymongo
import math
import random
import numpy as np

import threading
import time
import traceback

#在本地采用以下方式
client= pymongo.MongoClient("mongodb://localhost:27017/")

db = client["test"]
players = db['player']
items = db['item']
items_example = db['item_list']
storage = db['store']
player_list = []
it_idx = 0
it_list = []


item_rarity = ['white', 'green', 'blue', 'purple', 'gold']
item_info = []
item_info.append({'洛阳铲':5, '倚天剑':5, '屠龙宝刀':5, '玄铁重剑':5, '打狗棒':5, '秃子的红色拳套':5,
                  '厚土旗':4, '巨木旗':4, '洪水旗':4, '锐金旗':4, '烈火旗':4, '尚方宝剑':4, '黑卡':4, 'RTX 3090':4,
                  '手机':3, '笔记本电脑':3, '挖掘机':3, '照妖镜':3, '寻宝符':3, '兵家剑':3, '不知是谁的拂尘':3, '兵家矛':3, 'RTX 3060':3,
                  '空头支票':2, '玩具铲':2, '平底锅':2, '超市手推车':2, '蚌埠通行证':2, '军用水壶':2, '有灵气的茶壶':2, 'RTX 2060':2,
                  '某网站会员':1, '小区门卡':1, '卫龙':1, '可乐和曼妥思':1, '撬棍':1, '指南针':1, '乞丐给的葫芦':1, '秃子的拳套':1})
item_info.append({'降龙十八掌':5, '乾坤大挪移':5, '葵花宝典':5, '九阴真经':5, '易筋经':5,
                  '左右互搏':4, '黯然销魂掌':4, '辟邪剑法':4, '独孤九剑':4, '龙象般若功':4, '吸星大法':4, '打狗棒法':4, '大悲赋':4,
                  '玉女剑法':3, '凌波微步':3, '咏春':3, '蛤蟆功':3, '一阳指':3, '六脉神剑':3, '笑傲江湖':3, '化骨绵掌':3, '百家拳':3,
                  '闪电鞭':2, '军体拳':2, '七伤拳':2, '狮吼功':2, '日字冲拳':2, '螳螂拳':2, '形意拳':2, '蛇拳':2, '太极拳':2,
                  '偷袭老同志':1, '大学生军体拳':1, '三寸不烂':1, '龟息功':1, '鱼肺':1, '金城铁壁':1, '自在极意功':1, '枝折手':1})
item_type = ['equipment', 'ornament']
base_pro = [0.4, 0.3, 0.2, 0.08, 0.02]
luck_weight = [0, 0.05, 0.1, 0.25, 0.6]


class mythread(threading.Thread):
    def __init__(self,name, player, delay):
        threading.Thread.__init__(self)
        self.name = name
        self.player = player
        self.delay = delay
    def run(self):
        while 1:
```

```
            self.player.energy = 1
            time.sleep(self.delay)

#####################################            base_end            ##############################
##########

#####################################            player              ##############################
###########

class player(object):
    def __init__(self,id,name,wallet,ability,item_list,luck,wearing_equipment,wearing_ornament,e
nergy):
        self.id = id
        self.name = name
        self.wallet = wallet
        self.ability = ability
        self.item_list = item_list
        self.luck = luck
        self.wearing_equipment = wearing_equipment
        self.wearing_ornament = wearing_ornament
        self.energy = energy
        player_list.append(self)
        players.insert_one(self.__dict__)
        try:
            threadName = str(self.id)
            t = mythread(threadName, self, 5)
            t.start()
        except:
            traceback.print_exc()

    def python_to_mongo(self, pl):
        players.insert_one(pl.__dict__)

    def mongo_to_python(self, key):
        result = players.find(key)
        player_list = []
        for doc in result:
            temp = studentTable(doc['id'],doc['name'],doc['wallet'],doc['ability'],doc['item_lis
t'],
                                doc['item_list'],doc['luck'],doc['wearing_equipment'],doc['weari
ng_ornament'])
            player_list.append(temp)
        return player_list

    def work(self):
        salary = random.randint(self.ability, self.ability + self.luck)
        self.wallet += salary
        players.update_many({"id":self.id}, {"$set":{"wallet":self.wallet}})
        return salary

    def treasure_hunt(self):
        if self.energy < 1:
            return "fail"
        self.energy = 0
        if self.luck <= 18:
            #print("你没戴满饰品寻个🔨宝")
            val = 0
        else:
            val = max((math.log(random.randint(1, self.luck // 2),3) - 2) / 2 , 0)
        pro = np.array(base_pro)
        for i in range(0, 5):
```

```
            pro[i] = pro[i] * (1 + val * luck_weight[i])
        pro = (pro / (sum(pro)))
        for i in range(1, 5):
            pro[i] += pro[i - 1]
        p_val = np.random.rand()
        pro = pro.tolist()
        pro.append(p_val)
        pro = np.array(pro)
        pro = np.sort(pro)
        pro = pro.tolist()
        r = pro.index(p_val)
        it_list = list(items.find({"rarity":item_rarity[r], "status":""}))
        if not it_list:
            return None
        it = (random.sample(it_list, 1))[0]
        it['belonging'] = self.name
        it['status'] = "in_inventory"
        self.item_list.append(it)
        if len(self.item_list) > 10:
            self.remove_item()
        items.delete_one({"id":it['id']})
        players.update_many({"id":self.id}, {"$set":{"item_list":self.item_list}})
        return it

    def wear_equipment(self, it):
        if self.wearing_equipment != None:
            self.wearing_equipment['status'] = "in_inventory"
        self.ability = it['ability']
        self.wearing_equipment = it
        it['status'] = "wearing"
        players.update_many({"id":self.id}, {"$set":{"wearing_equipment":self.wearing_equipment
}})
        players.update_many({"id":self.id}, {"$set":{"ability":self.ability}})

    def wear_ornament(self, it, pos):
        if self.wearing_ornament[pos] != None:
            self.wearing_ornament[pos]['status'] = "in_inventory"
        self.wearing_ornament[pos] = it
        self.luck = 0
        for i in range(len(self.wearing_ornament)):
            if(self.wearing_ornament[i] == None):
                continue
            self.luck += self.wearing_ornament[i]['ability']
        it['status'] = "wearing"
        players.update_many({"id":self.id}, {"$set":{"wearing_ornament":self.wearing_ornament}})
        players.update_many({"id":self.id}, {"$set":{"luck":self.luck}})

    def consign(self, it, price):
        if it == self.wearing_equipment or it in self.wearing_ornament:
            print("你是要脱衣服吗")
            return "Error"
        it['price'] = price
        it['status'] = "on_block"
        storage.insert_one(it)
        players.update_many({"id":self.id}, {"$set":{"item_list":self.item_list}})
        players.update_many({"id":self.id}, {"$set":{"wallet":self.wallet}})
        return "OK"

    def purchase(self, it):
        global pls
        if it['belonging'] == self.name:
```

```
            it['status'] = "in_inventory"
            it['price'] = 0
            for i in self.item_list:
                if i['id'] == it['id']:
                    i['status'] = "in_inventory"
                    i['price'] = 0
                    break
            players.update_many({"id":self.id}, {"$set":{"item_list":self.item_list}})
            storage.delete_one({"id":it['id']})
            return self.name, self
        if self.wallet <= it['price']:
            return "Error", None
        self.wallet -= it['price']

        print(it['belonging'])
        seller = pls[it['belonging']]
        seller.wallet += it['price']
        for i in seller.item_list:
            if i['id'] == it['id']:
                (seller.item_list).remove(i)
                break

        it['belonging'] = self.name
        it['status'] = "in_inventory"
        self.item_list.append(it)

        print(seller.item_list)

        if len(self.item_list) > 10:
            self.remove_item()
        storage.delete_one({"id":it['id']})
        players.update_many({"id":seller.id}, {"$set":{"wallet":seller.wallet}})
        players.update_many({"id":seller.id}, {"$set":{"item_list":seller.item_list}})
        players.update_many({"id":self.id}, {"$set":{"wallet":self.wallet}})
        players.update_many({"id":self.id}, {"$set":{"item_list":self.item_list}})
        return seller.name, seller

    def remove_item(self):
        temp = self.item_list[0]
        for it in self.item_list:
            if it['ability'] < temp['ability'] and it['status'] == "in_inventory":
                temp = it
        self.item_list.remove(temp)
        return temp


########################################      player_end      ##############################


########################################    item      ############################

class item(object):

    def __init__(self,id,name,type,ability,price,rarity,belonging,status):
        self.id = id
        self.name = name
        self.type = type
        self.ability = ability
        self.price = price
        self.rarity = rarity
```

```
        self.belonging = belonging
        self.status = status
    def python_to_mongo(it):
        items.insert_one(it.__dict__)
    def mongo_to_python(key):
        result = items.find(key)
        item_list = []
        for doc in result:
            temp = studentTable(doc['id'],doc['name'],doc['type'],doc['ablility'],
                                doc['price'],doc['rarity'],doc['belonging'],doc['status'])
            item_list.append(temp)
        return item_list

############################     end_of_item     ########################


##########################      generate items      ###################

items_per_day = 150
items_in_inventory = 25
def generate_treasure(n):
    global item_info
    global item_type
    global item_rarity
    global it_idx
    global it_list
    for i in range(n // 2):
        a = random.sample(item_info[0].keys(), 1)
        r = item_info[0][a[0]] - 1
        it = item(it_idx, a[0], 'equipment', random.randint(3 ** (r + 2), 3 ** (r + 3)), 0, item
_rarity[r], "","")
        it_idx += 1
        it_list.append(it)
        item.python_to_mongo(it)
    for i in range(n // 2):
        a = random.sample(item_info[1].keys(), 1)
        r = item_info[1][a[0]] - 1
        it = item(it_idx, a[0], 'ornament', random.randint(3 ** (r + 2), 3 ** (r + 3)), 0, item_
rarity[r], "","")
        it_idx += 1
        it_list.append(it)
        item.python_to_mongo(it)


def generate_storage(n):
    global S
    global storage
    global item_info
    global item_type
    global item_rarity
    global it_idx
    global it_list
    for i in range(n):
        t = random.randint(0,1)
        a = random.sample(item_info[t].keys(), 1)
        r = item_info[t][a[0]] - 1
        it = item(it_idx, a[0], item_type[t], random.randint(3 ** (r + 2), 3 ** (r + 3)),
              random.randint((2 ** r) * 10 * 3 ** (r + 2), (2 ** r) * 10 * 3 ** (r + 3)), item_r
arity[r], "Store","on_block")
        it_list.append(it)
        it_idx += 1
```

```
        S.item_list.append(it.__dict__)
        storage.insert_one(it.__dict__)
        items.insert_one(it.__dict__)



############################          end of generation          ##########################

##### local msg  ########

p = player(-1,"",0,0,[],0,None,[None, None],0)
user = ""
Id = 0
price = 0
S = player(1,"Store",10**6,0,[],0,None,[],0)
chk = None
s_name = None
s_rarity = None
s_type = None

pls = {}
Ids = {}
prs = {}

def init_search():
    global chk
    global s_name
    global s_rarity
    chk = None
    s_name = None
    s_rarity = None
    s_type = None

###############


app = Flask(__name__)

@app.route("/")
def hello_world():
    return render_template('test.html')

@app.route("/success/<name>")
def success(name):
    global storage
    global items_in_inventory
    global Idx
    global items
    global items_per_day
    global it_list
    if pls == {}:
        it_list = []
        storage.delete_many({})
        generate_storage(items_in_inventory)
        items.delete_many({})
        generate_treasure(items_per_day)
        Idx = 0
    if not (name in pls.keys()):
        pl = player(len(player_list),name,0,0,[],0,None,[None, None],0)
        pls[name] = pl
```

```
    return render_template('homepage.html', result = pls[name].__dict__)



@app.route('/login',methods = ['POST', 'GET'])
def login():

    if request.method == 'POST':
        user = request.form['nm']
    else:
        user = request.args.get('nm')
    return redirect(url_for('success',name = user))

##############      equipment     ##############

@app.route('/w_equipment', methods = ['POST', 'GET'])
def wear_equipment():
    if request.method == 'POST':
        user = request.form['nm']
    else:
        user = request.args.get('nm')
    return redirect(url_for('equipment',name = user))

@app.route('/equipment/<name>')
def equipment(name):
    global pls
    p = pls[name]
    return render_template('equipment.html', result = p.__dict__)

@app.route('/wear', methods = ['POST', 'GET'])
def wear_item():
    global Ids
    global user
    if request.method == 'POST':
        Id = request.form['iid']
        user = request.form['nm']
    else:
        Id = request.args.get('iid')
        user = request.args.get('nm')
    Ids[user] = int(Id)
    return redirect(url_for('wear',name = user))

@app.route('/wear_id/<name>')
def wear(name):
    global pls
    global Ids
    p = pls[name]
    for i in p.item_list:
        if i['id'] == Ids[name]:
            it = i
            break
    p.wear_equipment(it)
    pls[name] = p
    return render_template('wear_success.html', result = it)

###############      end of equipment      ##################

################        treasure hunt         ###################

@app.route('/treasure_hunt', methods = ['POST', 'GET'])
def treasure_hunt():
```

```
    if request.method == 'POST':
        user = request.form['nm']
    else:
        user = request.args.get('nm')
    return redirect(url_for('treasure',name = user))

@app.route('/treasure/<name>')
def treasure(name):
    global pls
    global it_list
    p = pls[name]
    it = p.treasure_hunt()
    if it == "fail":
        return render_template('treasure_fail.html', pl = pls[name])
    pls[name] = p
    it_list[int(it['id'])].belonging = name

    return render_template('outcome.html', result = it)

###############        end of treasure     ############


#################       store      ##############

@app.route('/store', methods = ['POST', 'GET'])
def store():
    global chk
    global s_name
    global s_rarity
    global s_type
    if request.method == 'POST':
        user = request.form['nm']
        chk = request.form['afd']
        s_name = request.form['it_name']
        s_rarity = request.form['it_rarity']
        s_type = request.form['it_type']
    else:
        user = request.args.get('nm')
        chk = request.args.get('afd')
        s_name = request.args.get('it_name')
        s_rarity = request.args.get('it_rarity')
        s_type = request.args.get('it_type')
    return redirect(url_for('store_inventory',name = user))

@app.route('/storage/<name>')
def store_inventory(name):
    global storage
    global chk
    global s_name
    global s_rarity
    global item_rarity
    global s_type
    global pls
    p = pls[name]
    query = {}
    if chk == "yes":
        query['price'] = {"$lte":p.wallet}
    if s_name != None and s_name != '':
        query['name'] = s_name
    if s_rarity != None and s_rarity != '':
        query['rarity'] = item_rarity[int(s_rarity) - 1]
```

```
    if s_type != "both" and s_type != None:
        query['type'] = s_type
    result = storage.find(query)
    init_search()
    s = []
    for doc in result:
        temp = item(doc['id'],doc['name'],doc['type'],doc['ability'],
                            doc['price'],doc['rarity'],doc['belonging'],doc['status'])
        s.append(temp.__dict__)
    return render_template('store.html', result = s, w = p)

@app.route('/purchase_item', methods = ['POST', 'GET'])
def purchase_item():
    global Ids
    if request.method == 'POST':
        Id = request.form['pid']
        user = request.form['nm']
    else:
        Id = request.args.get('pid')
        user = request.args.get('nm')
    Ids[user] = int(Id)
    return redirect(url_for('purchase_it', name = user))

@app.route('/purchase/<name>')
def purchase_it(name):
    global pls
    global Ids
    global it_list
    p = pls[name]
    idx = Ids[name]
    print(idx)
    print(it_list[idx].__dict__)
    it = it_list[idx].__dict__
    y, seller = p.purchase(it)
    if y == "Error":
        return render_template('purchase_fail.html', result = it, pl = p)
    else:
        pls[name] = p
        pls[y] = seller
        if name == seller.name:
            it_list[idx].status = "in_inventory"
            return render_template('reclaim_success.html',result = it, pl = p)
        else:
            return render_template('purchase_success.html', result = it, pl = p)

####################   end of store     ##################


#################  work   ###################

@app.route('/work', methods = ['POST', 'GET'])
def work_prepare():
    if request.method == 'POST':
        user = request.form['nm']
    else:
        user = request.args.get('nm')
    return redirect(url_for('working', name = user))

@app.route('/working/<name>')
def working(name):
    global pls
```

```
    p = pls[name]
    money = p.work()
    pls[name] = p
    return render_template('work.html', result = money, pl = p.__dict__)


################   end of work   #############


##################  ornament    ##################

@app.route('/w_ornament', methods = ['POST', 'GET'])
def w_ornament():
    if request.method == 'POST':
        user = request.form['nm']
    else:
        user = request.args.get('nm')
    return redirect(url_for('ornament', name = user))

@app.route('/ornament/<name>')
def ornament(name):
    global pls
    p = pls[name]
    return render_template('ornament.html', result = p.__dict__)

@app.route('/wear1', methods = ['POST', 'GET'])
def w1():
    global Ids
    if request.method == 'POST':
        Id = request.form['oid']
        user = request.form['nm']
    else:
        Id = request.args.get('oid')
        user = request.args.get('nm')
    Ids[user] = int(Id)
    return redirect(url_for('wear1',name = user))


@app.route('/wear2', methods = ['POST', 'GET'])
def w2():
    global Ids
    if request.method == 'POST':
        Id = request.form['oid']
        user = request.form['nm']
    else:
        Id = request.args.get('oid')
        user = request.args.get('nm')
    Ids[user] = int(Id)
    return redirect(url_for('wear2',name = user))

@app.route('/wear1_id/<name>')
def wear1(name):
    global pls
    global Ids
    p = pls[name]
    idx = Ids[name]
    for i in p.item_list:
        if i['id'] == idx:
            it = i
            break
    p.wear_ornament(it, 0)
```

```
    pls[name] = p
    return render_template('wear1_success.html', result = it)

@app.route('/wear2_id/<name>')
def wear2(name):
    global pls
    global Ids
    p = pls[name]
    idx = Ids[name]
    for i in p.item_list:
        if i['id'] == idx:
            it = i
            break
    p.wear_ornament(it, 0)
    pls[name] = p
    return render_template('wear2_success.html', result = it)

#################    end of ornament    ####################


#################     consignment     ################

@app.route('/consign', methods = ['POST', 'GET'])
def consignment():
    if request.method == 'POST':
        user = request.form['nm']
    else:
        user = request.args.get('nm')
    return redirect(url_for('consign', name = user))

@app.route('/consignment/<name>')
def consign(name):
    global pls
    p = pls[name]
    return render_template('consign.html', result = p.item_list, pl = p)

@app.route('/consign_item', methods = ['POST', 'GET'])
def consign_id():
    global Ids
    global prs
    if request.method == 'POST':
        Id = request.form['cid']
        price = request.form['price']
        user = request.form['nm']
    else:
        price = request.args.grt('price')
        Id = request.args.get('cid')
        user = request.args.get('nm')
    Ids[user] = int(Id)
    prs[user] = int(price)
    return redirect(url_for('consign_item', name = user))

@app.route('/consign_complete/<name>', methods = ['POST', 'GET'])
def consign_item(name):
    global prs
    global pls
    global Ids
    global it_list
    price = int(prs[name])
    p = pls[name]
    idx = Ids[name]
```

```
    for i in p.item_list:
        if i['id'] == idx:
            it = i
            break
    p.consign(it, price)
    it_list[idx].status = "on_block"
    it_list[idx].price = price
    pls[name] = p
    return render_template('consign_success.html', result = it)

############       end of consign      ##########################
```