# 期末大作业报告

## 在线股票模拟交易系统

张浩威 1600012720

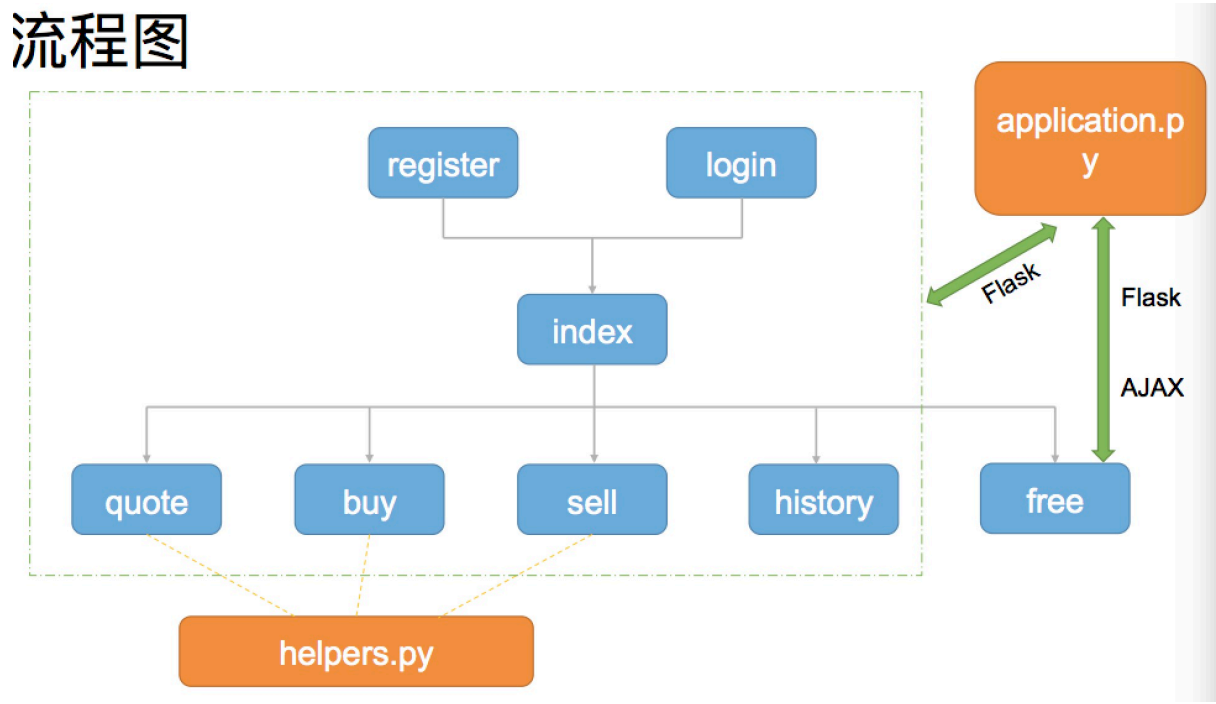
信息科学技术学院16级本

2018.6

# 基本功能&操作流程

一、网页设计

主要通过html5语言编辑页面内容，以提供的代码为框架，进行了完善丰富。




二、注册

用户点击register进入注册界面，填写用户名和两次密码即可完成注册。

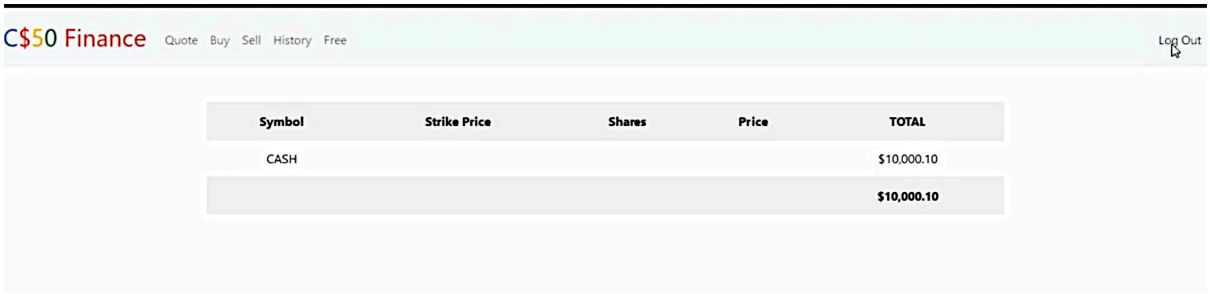

注册时对用户输入的约束：新注册的用户名必须尚未被使用、设置的密码不得少于6位、两次输入的密码必须一致。若条件不满足会进入apology页面，并显示相应

的错误原因。这部分主要用到了数据库的查询、存储的知识，我们是通过学习并使用Sqlite3来是实现这一部分的。另外为了防止用户利用密码攻击数据库，我们使用了md5对密码进行加密。

三、登录登出

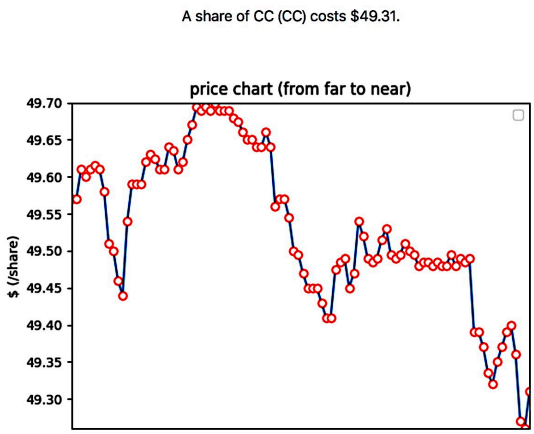
flask run之后可以直接进入登录界面，用户输入已有用户名和密码，若正确则进入个人账户页面，否则进入apology页面报错。每次成功进入后系统会自动更新股票信息和用户持有的财富值。账户页面index会显示已经购买的股票相关信息（股票代号、成交价、现价、持有数、余额、总钱数等）。


登录后，如果想退出，点击右上角的 log out 即可。

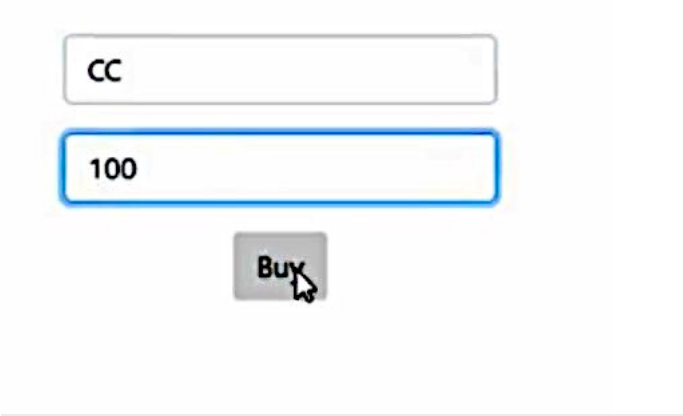



四、股票查询、买卖

用户可以点击quote查询股票，如果代号正确则进入quoted页面显示股票现价、价格走势折线图，这是通过AlphaVantage提供的API接口以及matplotlib库实现的。



用户可以在buy页面输入对应symbol买入股票，如果现金不足则报错。也可以在sell界面卖出，如果用户现有股票数不足也会报错。如果用户全部卖出，则主界面index将不再显示这支股票。数据库中相应记录也被修改或删除。





| Symbol | Strike Price | Shares | Price | TOTAL |
|---|---|---|---|---|
| CC | CC | 100 | $49.31 | $4,931.00 |
| CASH | | | | $5,069.00 |
| | | | | $10,000.00 |

五、交易记录

用户的每次交易都会被记录下来存入数据库，用户可以随时点击history查询交易记录。


| Symbol | Shares | Price | Transacted |
|---|---|---|---|
| CC | 100 | $49.31 | 2018-06-02 06:03:36 |
| CC | -1 | $49.31 | 2018-06-02 06:04:06 |
| CC | -99 | $49.31 | 2018-06-02 06:04:15 |

# 功能实现说明

下面将以buy的实现为例，详细说明相应功能是如何实现的。 其中db指的是保存用户的数据库，db2指的是保存信息的数据库。

1、识别出用户点击了“buy”按钮，则进入buy.html页面。

2、用request.form.get获取用户输入的symbol，通过给定的API接口查询相应的股票信息，如果股票不存在则报错“Invalid Symbol”。

3、判断用户输入的股票数目是否为正整数，否则报错。

4、查询数据库中user表，判断当前用户持有的现金cash是否足够支付股票，如不足则报”Not enough money”

5、如果资金充足，则修改数据库history表，增加交易记录。也即在db2中的插入字段。并更新user表中的现金cash，减少股票数乘以股票实时价格。查询用户目前持有的股票，如果用户已经持有这支股票，则更改数据库中表portfolio股票数目；否则增加一条记录。

6、返回index页面。

（附：buy函数部分代码截图）

```
def buy(): #购买模块
    if request.method == "GET":
        return render_template("buy.html")
    else:
        stock = lookup(request.form.get("symbol")) #查看是否有这个股票
        if not stock:
            return apology("Invalid symbol")
        try:
            shares = int(request.form.get("shares")) #查询输入的是否是一个正整数
            if shares < 0:
                return apology("You should input a positive integer")
        except: #如果连整数都不是发现异常
            return apology("You should input an integer")
        #上面这段和sell是一模一样的
        money = 0  #得到当前所拥有的钱
        data = db.execute("select id, cash from users")
        for i in data:
            if (i[0]==session["user_id"]):
                money = i[1]

        if float(money) < stock["price"] * shares: #如果钱已经不够了
            return apology("Not enough money")
        #买入成功
        db2.execute('''insert into tbhis
                    (id,symbol,shares,price)
                    values({},'{}',{},'{}')'''.format(session["user_id"],request.form.get("symbol"),int(request.form.get("shares")),usd(stock["price"])))
        db2.commit() #更新历史信息

        sum = stock["price"] * float(shares)
        db.execute('''update users set cash = cash - '''+str(sum)+''' where id = '''+str(session["user_id"]))
        db.commit() #更新剩余钱数

        data = db2.execute("select id, symbol, shares from tbhave") #更新还剩多少股票
        FLAG = 0 #FLAG表示这个股票是否早已在数据库中
        for i in data:
            if i[0]==session["user_id"] and i[1] == request.form.get("symbol"):
                FLAG = 1
                db2.execute("update tbhave set shares = shares + "+str(request.form.get("shares"))+ " where id = "+str(session["user_id"])+ " and symbol = '"+ request.form.get("symbol") + "'")
                db2.commit()

        if FLAG == 0: #发现没有
            db2.execute('''insert into tbhave
                    (id,symbol,shares,name)
                    values({},'{}',{},'{}')'''.format(session["user_id"],request.form.get("symbol"),int(request.form.get("shares")),stock["name"]))
        db2.commit()
        return redirect(url_for("index"))
```

# 亮点功能

一、动态广告

用户可以通关观看广告赚取0.1美金，为系统增加了趣味性，丰富了页面内容。

Watch this vedio, get free dollars!

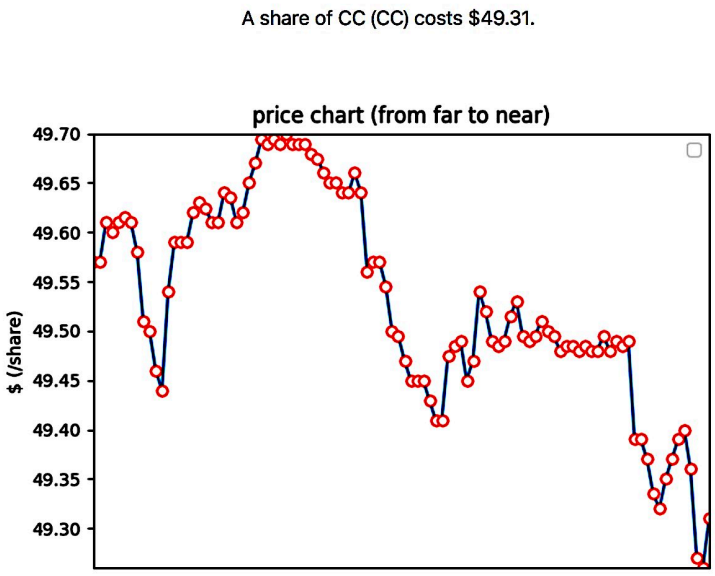

我们在static的文件夹中事先放入了ad.mp4文件，在html中将文字与视频展现出来。

之后我们利用jquery中的ajax来进行客户端与服务器之间的通信。当监听器检测到视频结束时，通过判断t的值来得知用户是否是第一次看完视频，如果条件成立，则向服务器端发送要增加用户金钱的信息。在服务器中接受到这个信息时，再更新数据库。

```
{% block main %}
    <div style="font-size:40px">Watch this vedio, get free dollars! </div> <br> <br>
     <video id="media" src="static/ad.mp4" controls width="480px" heigt="360px"></video>
    <script>
    var t = 1
    var md=document.getElementsByTagName("video")[0];
    md.addEventListener("ended",function(){
        if (t===1)
        {
            alert("You got one free dollar! Try to watch it again!");
            $.ajax({
                        type:"POST",
                        url:"/add",
                        success:function(data){}
            })
        }else
            alert("I am just kidding!");
        t += 1;
    })
    </script>
{% endblock %}
```

二、股价趋势图

在查询股票实时价格的同时，我们的网页会自行显示股价波动情况，帮助用户更好地决策。




我们利用matplotlib来实现图表功能。首先我们在helpers.py增加“pic”这个list，用来返回最近100秒该股票的价格。使用matplotlib利用pic里存储的信息来得到整张图表。如果我们直接更新服务器中的图表，在html界面直接刷新可能无法得到最新的图表，这是由于浏览器的缓存造成的。这里我们采取的解决方案是给所有图表一个不同的名称，在代码中使用变量num来实现。采取这样的方法，浏览器每次想得到一张图表时，都不再使用之前的缓存，问题得以解决。

```
x=[]
y=[]
for i in range(0,100):
    x.append(i)
    y.append(info["pic"][99-i])
frame = plt.gca()
frame.axes.get_xaxis().set_visible(False)
plt.plot(x, y, marker='o', mec='r', mfc='w')
plt.legend()
plt.margins(0)
plt.subplots_adjust(bottom=0.15)
plt.ylabel("$ (/share)") #Y轴标签
plt.title("price chart (from far to near)") #标题
global num
num += 1
plt.savefig("static/test"+str(num)+".png")
plt.close('all')
```

# 问题和解决

一、debug问题

在编写代码的过程中，我们主要遇到了两类问题：其一是编译错误，对于这类错误，我们首先查看编译器的提示，查看代码是否有语法、逻辑错误，或者上网查询相关错误及解决方案。其二是网页显示没有达到预期效果，这类错误我们在html中点击F12进入开发者模式，然后在console中设置段点，逐段执行，查看相关变量的变化来找到错误原因，也就可以找出解决方案。

二、图表更新问题

在制作图表的过程中，即使我们在服务器上更新了图片，刷新网页仍然无法查看到最新的图片。这是浏览器中的缓存引起的。解决方案是对我们每个图片设置一个新的标号，这样就不会引起cookie带来的问题了。

三、交互问题

在广告界面中，用户观看完广告后会有0.1$的奖励金额。如果我们采取常规的方式，使用服务器跳转到新的界面，用户体验将会大大降低。一个好的解决方案是采取异步交互，也就是在不必采用链接跳转，在同一界面就能完成信息的交互。我们采用jquery中的ajax来实现这一部分，最终的结果可以看出用户体验是非常友好的，达到了我们的预期要求。

# 未来展望

尽管我们在finance.cs50.net的基础上，增加了新的功能，但想要真正将模拟炒股系统投入市场还面临着巨大挑战。

一、当有多个用户端与服务器连接时，常常会出现堵塞的情况。例如服务器在等待用户1发出的请求，而此时用户2正在发送请求却等不到服务器的相应，就会造成拥堵。一个好的解决方法是将服务器采用多线程的方式实现。

二、在这个简易炒股系统中，我们无法得知所有股票的状况，只能通过输入股票名得到它最近100秒发生的信息。以用户体验的角度来看是极其不友好的，一个解决方案是在quote界面新增一个历史股票按钮，用户可以快捷地查询它曾经查询过或者买过的股票信息。

三、在真实炒股环境中，用户往往面临选择的困境。为了方便用户观察所有股票，在升级版的模拟炒股系统中，可以加入对比功能。解决方案是参照iphone官网，选择多只股票，在一张图表中显示它们的价格趋势。

四、股票走势预测。这是一个很热门的话题。一个简单的想法是利用机器学习得到一个损失尽可能小的函数去拟合股票的走势，但事实上这一方法的正确率非常低，其原因是因为股票的价格受到很多因素的影响，简单采用最近一段时间股票价格的变化去预测未来是一件不靠谱的事情。这一方面我们暂时还没有特别完善的解决方案，需要进一步探讨与尝试。

由于时间较为紧张，学业过于繁忙，对于这些挑战我们也只能暂时提出一个解决思路，日后我们将逐步实现挑战前三项功能，并尝试研究挑战四，期待将这一款模拟炒股系统真正投入市场的那一天。

# 总结

本次大作业，我使用了cs50提供的环境，使用python作为后端，灵活调用已知的包和API接口，使用了SQL语言进行数据库操作，使用html5作为前端编写网页。更加深入地理解编程的逻辑过程，了解了以上几方面的具体知识，总体能力得到了很大的提高。

我将所有代码与大报告在Github上进行了开源。注释丰富易于理解，希望能带给以后的同学们启发与帮助，也欢迎大家学习交流。

Github开源网址：https://github.com/alpq654321/cs50-finance