Matthew Peden

CS396

November 28th, 2022

**Project Report - Phase 3**

**Project Description**

The first phase of this project involved creating a forum website that allows users to post discussion boards and reply to other users. Also in phase 1 was the file management system, where users can upload numerous files, such as pdf or png files. In phase 2 I created the financial system, which consists of an interface that allows users to view and purchase stocks, properties, and other products. Banks and bonds were also incorporated. Additionally, in phase 2, I created a query system that allows users to search for specific information, such as user transactions by date, all assets belonging to an agent, or loan ratings for a bank. In phase 3, I did more work with the financial system. First, I added a “user expenditures” feature where users can record their expenses for a specific timeframe. Within the query system, you can also view expenditures for a user in a specific timeframe, and if their expenses are more than their income, a flag/warning message is displayed. Data visualization is also utilized on this same screen, with a pie chart, a bar chart, and a graph of the simple moving average being displayed. Data visualization for stock data is also used. Additionally, I created a function that automatically scrapes stock data from Yahoo Finance for the current day and inserts it into the database.

**Technical Details**

The database framework being used for this project is MySQL. I was previously using MySQLite, which is default in Django, but I decided to switch to MySQL in phase 2 of this project. MySQL is an open source database framework that uses relations and it requires a server connection and a database administration tool to manage everything, such as MySQLWorkbench. The database administration tool that I have been using for this project is DBeaver. The languages used in this project are Python, CSS, HTML, and a small amount of JavaScript. For the client-side design of the website, Bootstrap is used which is an open-source CSS framework that provides a way to design a website easily and efficiently. This entire project was completed using Django, which is a Python-based web framework that uses models, templates, views, and forms, among other things. The Django admin system is also incorporated in this project to provide admin users with a system to manage the website, and the Django Debug toolbar is implemented to provide helpful statistics and information.

Packages/modules used in this project are as follows. First, crispy forms is used. Crispy forms provides an additional filter and tag (|crispy and {% crispy %}, respectively) that allows you to easily and smoothly control the rendering behavior of Django forms. Widget tweaks is also used. Widget tweaks is a module that allows you to “tweak” the rendering of form fields inside templates. Mysqlclient is also used, which is what connects a MySQL database to a Django project. Pandas, an open source data analysis tool, is also used with data visualization and stock data scraping. Another package used for data visualization is NumPy, which provides many useful mathematical functions. Finally, also being used in data visualization, is matplotlib, which is a Python plotting library.

**Notable Features**

One notable feature I would like to emphasize is my automatic scraping of stock data from Yahoo Finance. I was very excited when I finally got this working, and it is good to know that the stock information will always be up to date. To scrape this data, I used a function from the Pandas library to dynamically read a csv file that contains stock data directly from Yahoo Finance, created variables from the value returned from the function that reads the csv file, and inserted those variables into the database.

Another notable feature I want to emphasize is my data visualization using the matplotlib plotting library and the Pandas library. I was able to incorporate 3 charts for both stock data and user expenditures: a bar chart, pie chart, and a simple moving average chart. Matplotlib is very easy to use and understand, and there are numerous charts available to create with it. Paired with Pandas to create a tabular data structure, these charts are very efficient. The rendered charts can be saved into a directory and referenced later in an html img tag to be displayed, which is the route that I took.

**Functions Implemented**

In phase 3 I created several new functions as well as updated functions that I created in phase 1 and 2. Some of the new and more notable functions are listed below.

* list\_expenditures: This function uses POST data to list all expenses for a user in a given time frame and creates multiple charts that are displayed for data visualization. These charts are the pie chart, bar chart, and a chart that depicts the simple moving average.
* scrape\_stocks(): This function scrapes stock data from Yahoo Finance and inserts it into the database.

Shown below is the code for each function listed above.

def list\_expenditures(*request*):

    template = 'financial\_system/list\_user\_expenditures.html'

    if *request*.method == 'POST':

        user = User.objects.get(*name*=*request*.POST['user'])

        start\_date = *request*.POST['start\_date\_year'] + '-' + *request*.POST['start\_date\_month'] + '-' + *request*.POST['start\_date\_day']

        end\_date = *request*.POST['end\_date\_year'] + '-' + *request*.POST['end\_date\_month'] + '-' + *request*.POST['end\_date\_day']

        exp\_data = UserExpenditures.objects.filter(*user\_id*=user.id, *start\_date\_\_gte*=start\_date, *end\_date\_\_lte*=end\_date)

        stock\_transactions = StockTransaction.objects.filter(*user\_id*=user.id, *transaction\_type\_id*=1)

        prop\_transactions = PropertyTransaction.objects.filter(*user\_id*=user.id, *transaction\_type\_id*=1)

        oth\_transactions = OtherTransaction.objects.filter(*user\_id*=user.id, *transaction\_type\_id*=1)

        total = 0

        stock\_expenses = 0

        for entry in stock\_transactions:

            stock\_expenses = stock\_expenses + (entry.shares \* entry.price)

        prop\_expenses = 0

        for entry in prop\_transactions:

            prop\_expenses = prop\_expenses + (entry.price)

        oth\_expenses = 0

        for entry in oth\_transactions:

            oth\_expenses = oth\_expenses + (entry.amount \* entry.price)

        total = total + stock\_expenses + prop\_expenses + oth\_expenses

        food\_total = 0

        health\_total = 0

        entertainment\_total = 0

        vehicle\_fuel\_total = 0

        children\_total = 0

        travel\_total = 0

        other\_total = 0

        entry\_total = 0

        dates = []

        entry\_totals = []

        for entry in exp\_data:

            entry\_total = entry.food + entry.health + entry.entertainment + entry.vehicle\_fuel + entry.children + entry.travel + entry.other

            total = total + entry.food

            total = total + entry.health

            total = total + entry.entertainment

            total = total + entry.vehicle\_fuel

            total = total + entry.children

            total = total + entry.travel

            total = total + entry.other

            food\_total = food\_total + entry.food

            health\_total = health\_total + entry.health

            entertainment\_total = entertainment\_total + entry.entertainment

            vehicle\_fuel\_total = vehicle\_fuel\_total + entry.vehicle\_fuel

            children\_total = children\_total + entry.children

            travel\_total = travel\_total + entry.travel

            other\_total = other\_total + entry.other

            dates.append(entry.start\_date)

            entry\_totals.append(entry\_total)

        labels = [

            'Food',

            'Health',

            'Entertainment'

        ]

        sizes = [

            food\_total,

            health\_total,

            entertainment\_total

        ]

        if(vehicle\_fuel\_total > 0):

            labels.append('Vehicle Fuel')

            sizes.append(vehicle\_fuel\_total)

        if(children\_total > 0):

            labels.append('Children')

            sizes.append(children\_total)

        if(travel\_total > 0):

            labels.append('Travel')

            sizes.append(travel\_total)

        if(other\_total > 0):

            labels.append('Other General')

            sizes.append(other\_total)

        if(stock\_expenses > 0):

            labels.append('Stocks')

            sizes.append(stock\_expenses)

        if(prop\_expenses > 0):

            labels.append('Properties')

            sizes.append(prop\_expenses)

        if(oth\_expenses > 0):

            labels.append('Other Products')

            sizes.append(oth\_expenses)

        fig1, ax1 = plt.subplots()

        ax1.pie(sizes, *explode*=None, *labels*=labels, *autopct*='%1.1f%%', *shadow*=True, *startangle*=90)

        ax1.axis('equal')

        plt.savefig('boards/static/boards/images/pie\_chart.png', *dpi*=100)

        product = {

            'date' : dates,

            'spending' : entry\_totals

        }

        df = pd.DataFrame(product)

        df.head()

        plt.figure(*figsize*=[10,5])

        plt.grid(True)

        plt.plot(df['date'],df['spending'],*label*='SMA')

        plt.legend(*loc*=2)

        plt.xlabel('Date')

        plt.ylabel('Expenditure ($)')

        plt.savefig('boards/static/boards/images/exp\_sma.png', *dpi*=100)

        fig = plt.figure(*figsize*=(10,5))

        plt.bar(dates, entry\_totals, *color*='blue', *width*=5)

        plt.xlabel('Date')

        plt.ylabel('Expenditure ($)')

        plt.savefig('boards/static/boards/images/bar\_chart.png')

        income = f'{user.monthly\_income:,}'

        total\_formatted = f'{total:,}'

        context={

            "exp\_data":exp\_data,

            "user":user,

            "income":income,

            "total":total,

            "total\_formatted":total\_formatted,

            "stock\_transactions":stock\_transactions,

            "prop\_transactions":prop\_transactions,

            "oth\_transactions":oth\_transactions,

            "dates":dates,

            "entry\_totals":entry\_totals

        }

        return render(*request*, template, context)

def scrape\_stocks():

    tickers = {

        "AAPL": 2,

        "MSFT": 3,

        "CSCO": 4,

        "META": 5,

        "AMZN": 6,

        "TSLA": 7,

        "NFLX": 8,

        "GOOGL": 9,

        "HOOD": 10

    }

    for key, value in tickers.items():

        ticker = key

        id = value

        period1 = int(time.mktime(datetime.datetime(2022, 11, 25, 23, 59).timetuple()))

        period2 = int(time.mktime(datetime.datetime.now().timetuple()))

        interval = '1d'

        query\_string = f'https://query1.finance.yahoo.com/v7/finance/download/{ticker}?period1={period1}&period2={period2}&interval={interval}&events=history&includeAdjustedClose=true'

        data = pd.read\_csv(query\_string)

        date = data['Date'].tolist()[0]

        open = data['Open'].tolist()[0]

        high = data['High'].tolist()[0]

        low = data['Low'].tolist()[0]

        close = data['Close'].tolist()[0]

        volume = data['Volume'].tolist()[0]

        stock\_object = Stock.objects.get(*id*=id)

        stock\_check = StockData.objects.filter(*date*=date, *stock\_id*=id).exists()

        if stock\_check == False:

            s = StockData(*open\_price*=open, *close\_price*=close, *high\_price*=high, *low\_price*=low, *volume*=volume, *date*=date, *stock*=stock\_object)

            s.save()

**Discussion**

Now that I am finished with this project, I can say that it has certainly been a success. I learned quite a lot about both Python/Django and the creation of a website in general during my work on this project, and I am very excited to work on other projects that use similar frameworks and logic in the future, as I had a lot of fun creating this website. I’m also taking both my Senior Project class and Mobile App Development next semester, so hopefully I will be able to apply some things that I have learned here in this class with my work on this project.