Carter Williams

11/27/2022

Dr. Zhonghang Xia

CS 396

Phase Three Report

**Summary**

Phase three of the term project is the final phase of the term project. This phase, I added onto work mostly done in phase two by adding three key features. Stock data is updated every Friday at 11:59 P.M. and saved to the database; this includes data about the stock’s price each day of the week. Stock data is also graphed over a given period with a 50-day simple moving average; the data from the data range is also used in a multiple linear regression model to predict the closing price of a stock based on three variables. Finally, expense tracking is also added with features like pie charts, bar charts, warnings when the expenses exceed the income, and a prediction of monthly/yearly expenses based on income.

**Description**

Phase three of the term project added financial tools and functions to the existing website that allows users to perform many more actions. Before discussing the new features, the requirements from the rubric include:

1. The user can post a memo/plan about income, cost, budget, and family members can comment on the post by the reply function. The receipts and documents can be uploaded to the website.
2. Record the user’s expenditures on food, health, entertainment, etc, into the system, and the user can search the record for a specific time.
3. Use a pie chart to display the user’s expenditures with the corresponding percentages.
4. Periodically update financial data in the system. For example, stock market data are supposed to be updated on a daily base.
5. Use bar charts to show the user expenditures, incomes, and other financial data on a month/year base.
6. Show a red flag (some signals) when the user’s expenditure exceeds his/her income.
7. Plot the simple moving average (SMA) curve for the expenditure and close stock price for a given period.
8. Use a linear model (e.g., linear regression model) to predict the future (tomorrow/next month/year) expenditure and close stock price.

This section will briefly describe how the project meets these requirements using a black-box approach. In other words, this section will describe these new features with little regard to the technical details. The implementation details will be discussed in the next section.

Requirement 1 was met in phase one of the project; users make a post with a title, body, and multiple files; these files can be a pdf, image, zip folder, etc. Any other user can make a comment on the post while viewing it.

Requirement 2 required completely new functionality to be added to the project by creating a new part of the database to handle keeping track of expenses. Users can add new expenses by specifying the date, amount, category, and an optional, short description. These expenses are then displayed in a table, and the table data can be filtered by the year, month, or day.

Requirement 3 builds off requirement 2, and simply requires a pie chart to be built based on the user’s expense data. Expense data is used to create a pie chart that is then saved as a ‘.png,’ and this image is displayed to the user with percentages for each piece.

Requirement 4 was partially implemented in phase two by building the database to handle this data. Every Friday at 11:59 P.M., the system grabs financial data for stocks currently in the database from the past week and uses it to store in the database.

Requirement 5 was met by creating a bar graph that displays the user’s specified income, total expenses for the time frame specified, and total value for currently held assets. If the time frame is a year, the yearly income and expenses are shown; if the time frame is a month, the monthly income and expenses are shown. The total asset value does not change based on the timeframe specified.

Requirement 6 was met by creating a check on the page that shows expenses. The check adds all expenses from the current month and checks it against the user’s monthly income. If the expenses are higher than the income, a warning message is shown on the expenses page that shows the current month’s expenses and income.

Requirement 7 was met in two different places. First, a graph plots the simple moving average for expenses in a specified time frame. The simple moving average windows range from 50 to 2 depending on the number of datapoints in the dataset. Secondly, the close stock price of a specified stock is plotted on graph along with a 50-day simple moving average. The data’s date range is specified by the user.

Requirement 8 was the hardest to meet but was accomplished in two parts. Firstly, a user’s expenditures can be predicted for a month or year based on all user data. Income and expense data is gathered from all users in the database (there are too little users, so data was fabricated to implement this feature), and the income is used as the independent variable in a simple linear regression model. The model is developed from the user’s data and used to create a linear line for predicting a user’s expenses based on their income. This model is then used to predict the user’s expenses for the month or year based on their income. Secondly, a multiple linear regression model is used to predict the close stock price based on the open, low, and high for a day. The training data is grabbed from a time period specified by the user, and the user can specify the three independent variables to predict the closing price.

As seen from the previous paragraphs, all requirements were met for phase three. The next section will discuss more technical details regarding the implementation of the requirements.

**Technical Details**

This section will discuss the technical and implementation details of the requirements for this phase. The details of requirement 1 will not be discussed because it was discussed in phase one of the project. This section will be split into two sub-sections: one will discuss the technical details relating to user expenditures and the other will discuss the technical details of stock information.

**User Expenditure Technical Details**

The first major requirement to add actions related to expenditures is to add support for storing expenditures in the system. Two tables were added to the database, expense and expense category. The expense category simply holds the name for the category an expense came from (i.e. health, transportation, food, etc.). Expense stores the cost, date, and description of the expense along with foreign keys for the user and category. Now, the system just needed a way to gather and put the data into the database. This was accomplished by creating an HTML form that passes the information necessary to an expense to a function that creates a new row in the database for the expense. All the expenses are then displayed in a table, and the table data can be filtered by the year, month, and/or day.

There are three charts that are displayed to the user relating to their expenses: a pie chart split into categories, a bar chart that shows the values of income, expenses, and assets, and a line chart that shows the simple moving average for expenses. These three charts use expense data from the database and matplotlib to create them. Matplotlib is a Python library for creating charts, and after all the data is added to the chart, they are temporarily written to a folder as a ‘.png’ to be displayed to the user. These charts update based upon the dates entered by the user as well. For example, if the user wants only expenses from October of 2022, the charts will only show data from October of 2022.

Finally, a simple linear regression model is used to predict a user’s expenses for a month or year. The model uses the income as the independent variable and the expenses as the dependent variable. Training data is supposed to be gathered from other users in the database, but there are not enough users to accomplish this currently, so sample data is passed to the linear model. This linear model is created with the scikit learn Python library and predicts the user’s expenses for the month or year by feeding their income into the model. A graph is displayed with the training data, and the user is shown their predicted expenses next to it.

**Stock Features Technical Details**

The second major part of this phase required implementing many features centered around tracking and analyzing stock information. No new tables had to be added to the database, this is because the previous phase already had the tables built to handle updating stock pricing information. This leads to requirement 4, when the Django server starts up, a task is created for every Friday at 11:59 P.M., and this task simply goes through all the stocks in the database and updates the pricing and volume information from the previous week.

The second major requirement centered around stock information is displaying a graph for a stock’s closing price over a given period. This feature requires the user to input the ticker and the date range. The system uses many packages, like yfinance to gather the pricing information and pandas to store the information, to gather the data. This data is then used to plot the price over the period using matplotlib. A 50-day simple moving average curve is also plotted, and this is made possible by gathering data from the previous 50 trading days. If data was only gathered from the time frame specified, the first 49 days of the stock’s price would not have a 50-day simple moving average.

Finally, the last feature builds off the previous one by taking the data gathered from the previous operation and building a multiple linear regression model. There are three independent variables: the open, low, and high price for the day; these are then used to predict the dependent variable, the closing price. The user can then input the independent variables into the system in order to get a predicted closing price. This model is not plotted onto a graph because there are not enough axes to support three independent and one dependent variable. This information can be used by the user to predict the closing price for a day, if they think the stock has already achieved its open, low, and high for the day.

**Summary of Achievement**

Overall, the final system implemented provides a lot of functionality that is useful to many users. The system allows users to make forum posts, replies, and store files; this functionality alone acts as a simplified social media site or forum board. A very useful feature of the system is the functionality of tracking information related to a user’s finances, with some analysis provided. The system tracks assets, expenses, incomes, and miscellaneous stock information, and that alone is of great use. Graphs and charts are displayed to the user, so they can analyze their expenses and stock pricing data. Linear regression models are also used in order to predict a user’s expenses and a stock’s closing price. Of course, these predictions should be taken with a grain of salt because they do not take all or most factors into account.

The main achievement of this project is likely to whole financial management system. Allowing users to add various assets and track their expenses is very useful. Not only does the system store this data, but it also analyzes it in order to provide a wider range of information to the user. For example, the system breaks down the user’s expenses, and this information allows them to discover if they spend too much in one area. Another example includes the stock price chart and prediction tool. These two features can provide the user with more insight into a stock before they buy or sell it.

Not only is the system beneficial to users, but it was also beneficial to me as a developer. This project covered a large area of programming and software engineering. I learned a great deal about software architecture, server-client systems, web-development, database interaction, package management, and many other useful concepts. In the end, this project is close to being a complete website with many useful features, but it also helped me learn more about software development.

**Discussion**

So far, I have not discovered any problems or faults with the system, but I have only conducted system tests. One conceptual problem I would like to discuss is the prediction of the closing stock price based on the three independent variables of open, low, and high price. I am unsure if I implemented the model correctly. I have not taken linear algebra yet, and the idea of linear regression slightly confused me. The implementation details of it are discussed in the technical details section, so I will not go over it again, but I think I implemented it correctly.

In terms of work, I would like to highlight, I implemented all the requirements throughout each phase of the project, and the system works with a somewhat friendly user interface. If I could go back, I would use bootstrap for making the UI because it would look nicer and be easier to create web pages with.

Future work for this project could take many directions. If I viewed this as needing to meet the requirements of a commercial system, I would have to refactor most or all of the codebase. I have learned a lot about software engineering this semester, and at the beginning, when this project started, I was mostly clueless about topics like OOP design patterns and system architecture. Also, adding new features to the system will be very hard and time consuming, unless the codebase is cleaned up. After the codebase is in a more manageable state, I would redesign the UI to use bootstrap with more JavaScript. This would create a more interactive, friendlier, and better-looking user interface. Finally, I would add more features to specifically the financial management side of things. Many useful and neat features could be added like automatic expense tracking (by adding consistent monthly charges like a Netflix subscription), dividend tracking, and better bond information. These are just a few examples, but, in reality, the number of potential new features is endless.

Overall, I am very proud of this project, and I believe it taught me a great deal about creating software systems. I made many mistakes relating to the design and architecture of the system, but these mistakes helped me learn better techniques to writing cleaner code.