CSCC43: Introduction to Databases

Project - Fall 2025: A Social Network for Stocks

In this project, you will develop an application from scratch that meets the specifications laid out here. This application combines requirements that are common to end-to-end database-backed applications with some elements of data mining and analysis using database engines.

The goal of this project is to get you to exercise your skills in data modeling, database design and optimization, and application implementation. This document forms a description of the domain you will model using the entity-relationship and relational approaches, which occurs before you write a line of SQL or other code.

This project is designed for academic purposes only to practice database and application development skills. No investment advice is offered in or should be inferred from any aspect of this project.

Overall Requirements

You are to develop a database-backed application that lets a user do the following:

- Register for an account and log in.
- Access their portfolios. A user may have one or more portfolios.
- Manage stock lists. A user can have zero or more stock lists.
- Track a portfolio of the user's stocks, including their current value and other aspects of their performance.
- Analyze and predict the performance of stocks, portfolios, and stock lists based on historical performance. You will have access to 5 years of historical daily stock data for this purpose. Plus you will add a facility to add new data and integrate this new data into analysis and prediction. This part of the project will give you the opportunity to play with simple data mining, and rather more sophisticated prediction techniques.
- Send requests to other users to become friends, view and accept friend requests, and view and remove existing friends.
- Share stock lists with friends to request reviews. Reviews on stock lists that are shared with friends can only be viewed by the sharer and the friend who made the review.
- Stock lists can also be made public. These lists can be reviewed by any user(s), and the reviews are also publicly visible.

Below we explain in detail each of these items, and give the concrete specifications for each.

Portfolios and Portfolio Management

A portfolio is a collection of investments that is intended to serve some purpose for the portfolio holder. For the context of this project, investments will consist purely of stocks and cash. We will also ignore the numerous details of real stock investment, such as stock dividends, taxes, margin, and trading costs. There are also many kinds of investments other than stocks that could be included in a portfolio.

While the stocks actually held by an individual investor certainly constitute a portfolio, portfolios are put together for other reasons too, for example to analyze how a particular collection of stocks has done in the past, to predict how well it may do in the future, or to evaluate how well an automated trading strategy might work for the portfolio.

An important investing problem is how to choose investments and their relative proportions such that the risk (and reward) of the portfolio as a whole is controlled. For example, a 20-something computer scientist may be willing to have a much riskier portfolio than a retired 70-something teacher.

The intuition behind designing a portfolio with a given “risk profile” is pretty simple. The amount of risk of an investment (a stock here) is basically the variance of the value of the investment, sometimes normalized to the average value of the investment (standard deviation over mean, or “coefficient of variation” (COV)). A high COV means the stock is “volatile” and thus riskier. Now, suppose you are trying to choose two stocks. You cannot only compute their individual variances, but also their covariation (and thus correlation). If the two stocks are positively correlated, then this means that if both of them are in your portfolio, the combination will be more volatile (higher risk). If they are negatively correlated, then the combination will be less volatile (lower risk). So, “portfolio optimization” is the process of choosing a collection of stocks such that their covariances/correlations with each other combine to give you the variance (risk) that you want while maximizing the likely return (reward). One simplification is to just consider the correlation of each stock with the market as a whole (this is called the “Beta coefficient”) in building a portfolio. The Beta of the whole portfolio can thus be made larger or smaller than the market as a whole by choosing the right stocks.

The devil in the details is that in order to build a portfolio like this, we would need to know the future values of volatility, covariance, Beta, etc, or of the stock prices themselves. We only have the past ones. So, a very important discipline is prediction, determining how a stock is likely to move in the future based on how it and all other stocks have moved in the past, as well as predicting what the future values of the other statistical measures will be. Since the statistics are almost certainly nonstationary, we will occasionally fail completely in predicting the future. The best we can do is muddle through, but there is a huge range of possibilities, and a big part of any serious trading enterprise is data mining historical data to develop better and better predictors.

Another important consideration is automation. We would like to have a computer program that continuously adapts the portfolio holdings in pursuit of maximizing return while controlling risk. These programs are called “trading strategies”, and another important goal of data mining of historical financial data is to find them.

Stocks

A share of stock represents a tiny bit of ownership of a company. Companies pay dividends (cash money) on their outstanding shares. The value of a stock is essentially the (discounted) sum of all of the future dividends it will pay. Since no one knows what that is, markets try to estimate it by buying and selling. The price at which a stock is sold (the “strike price”) is an estimate of its value—the seller thinks the stock's value is less than the strike price, while the buyer thinks it's more. Notice that a “price event” happens on every sale, and there may be 1000s of sales per day of a stock. If you look at the “stock price” in some typical free web stock quoting service, you're seeing an average of these sales over some interval of time, or, in some cases, the most recent sale.

For the purpose of this project, we will consider only information about the “day range” of a stock. In particular, for each day and stock, we have:

- Symbol: the alphabetic string representing the company, e.g., AAPL represents Apple Inc.
- Timestamp: date for which the stock price was recorded
- Open: the strike price of the first trade of the day
- High: the highest strike price during the day
- Low: the lowest strike price during the day
- Close: the strike price of the last trade of the day
- Volume: the total number of shares traded during the day

We will later provide 5 years of such historical data for the companies tracked by S&P 500.

Managing Portfolios

A portfolio consists of the following. You will need to figure out what parts of this need to live in the database and what parts can be generated on the fly.

- Cash account – money that the owner can withdraw or use to buy stocks. If they sell a stock the money it brings in goes here. The owner can also deposit more money from cash accounts in other portfolios or external bank accounts.
- A list of stock holdings. A stock holding consists of a stock symbol and the number of shares of the stock that the owner has. We ignore stock lots here since we are ignoring taxes.

The owner of a portfolio should be able to do the following:

- Deposit and withdraw cash from the cash account.
- Record stocks bought and sold, i.e., changing the cash and stock holdings in the portfolio.
- Record new daily stock information (like the above), beyond that in the historic data we give you. If you want to be fancy, you can pull this information from your favorite Internet stock quoting service.
- Integrate stock information from the historical data (which is read-only) and from the new daily stock information to provide a unified view of stock information. That is, the view of information about a stock should be the union of the views in the historic data and in the new stock data that the user or script enters.
- Examine the portfolio. The portfolio display should show the total amount of cash, and the holdings of each of the stocks. It should probably also show the information in the following item:
- Get the estimated present market value of the portfolio, the individual stock holdings, and the cash account. We will consider the present market value of a stock to be based on the last close price we have available.
- Get statistics of the portfolio. Based on historical data, we should be presented with information about the volatility and correlation of the stocks in the portfolio. Note that these can take some time to compute, so they should be cached.
- Get the historical prices of the individual stock holdings. It should be possible to click on any holding and get a table and plot of its past performance.
- Get the future price predictions for individual stock holdings. It should be possible to select any holding and generate a plot of its likely future performance, based on past performance.

The last three items are explained in greater detail as follows.

Portfolio Statistics

It should be possible to display the following information about the portfolio, all of which should be computed from the data in the database.

- The coefficient of variation and the Beta of each stock.
- The covariance/correlation matrix of the stocks in the portfolio.

It should be possible to compute these values over any interval of time. We expect you to do the computation within the database, not just by pulling the data down to the front-end.

Historical Prices of Stocks

From the display of all the stocks in the portfolio, it should be possible to select any stock to generate a graph of the stock's past value (the close price) for some interval of time. It should be possible to set the interval (typical intervals for daily information include a week, a month, a quarter, a year, and five years). Recall that you need to integrate both historic and current stock price data.

Future Prices of Stocks

This is the most open-ended part of the project. You should create (or use) a tool that predicts the future value (close price) of a stock from its past. In the portfolio view, the user can then select a stock and see a graph showing your predictions, for some user-selected interval of time into the future.

Note that this is a databases course, not a course on machine learning or statistics. *You won't be graded on your prediction performance*. We want you to have the experience of integrating this kind of tool with a database. If you think a bit about prediction in addition, that's nice, but not essential.

Social Networking

The key functionality of social networks is to connect people. In this project, we provide a social platform for users who own portfolios to exchange opinions on stocks and their combinations with friends. To that end, this social network consists of the following components.

Managing Friends

A user can request to be friends with any other user in the system. Once a user logs in, they should be able to view existing friends, incoming friend requests, as well as outgoing requests that have been sent but haven't been accepted. They can process a request from another user by either accepting or rejecting it. A friendship is mutual—if a request is not approved, the two users are not friends. No duplicate requests should be sent, and a rejected request can be re-sent again five minutes later from the rejection. Users can delete friends, and the five-minute limit is also applied to requests from deleted friends.

A user's friend list and incoming and outgoing requests cannot be accessed by other users.

Sharing Stock Lists

A stock list consists of stocks and their shares, i.e., each element in the list is a pair of a stock symbol and the number of shares of that stock. Each stock list has a creator, the user who created the list. Different users can create stock lists that consist of the same stocks and shares. Unlike a portfolio, which can be accessed by the owner only, a stock list can be shared with other users by its creator and even be made public, i.e., shared with all the current and future users in the system. When a user logs into the system, they should be able to view all stock lists accessible to them, tagged by the proper visibility category (i.e., whether the stock list is “private” to the owner themselves, “shared” with a friend, or “public” to all platform users). The user can delete stock lists that were created by them.

Portfolio statistics are extended to stock lists: coefficient of variation and the Beta of each stock, and the covariance/correlation matrix of the stocks in the list. For any stock list accessible to a user, i.e., one created by or shared with them, or a public one, the user should be able to access its statistics.

Reviewing

A user can write reviews for stock lists. A review is essentially a block of text expressing a user’s opinions on a stock list. We can assume that a review never exceeds four thousand characters. A user can write at most one review for a stock list that is accessible to them, although a review can be edited by the reviewer after it is written. A review for a non-public stock list can be viewed only by the reviewer and its creator. All the reviews of a public stock list can be viewed by any user in the system.

A review can be deleted by the reviewer or the stock list creator. If a stock list is deleted, all the associated reviews should be deleted.

Implementation

You’re allowed to pick any programming language, e.g., Java and C++, to implement the application. More instructions and preparation for implementing the project will be provided later in the term.

Extra Credit

You can extend your application with the following tasks to earn extra credit. Each of these tasks is worth up to 5% of the project grade.

- Optimized interface. We don’t require a GUI for the application, i.e., all the functions described in this document can be implemented on the command line. A team can gain extra credit by making the interface more intuitive to use and optimizing the visualization. In the demonstration, we will ask if you want to request extra credit for the application interface, and if so, in the report you should explain your efforts as an appendix of the report.
- Optimized query performance. We only evaluate the correctness and completeness of the application functions (we do check anomalies in your database design though). A team can optimize the performance of these functions (i.e., execution time) by various database techniques, e.g., indexing, caching, and more advanced application abstraction and schema design, to make the application faster and earn extra credit. To request extra points, a team should explain what optimization techniques they adopted and how effectively they improved the application performance in the report, also as an appendix.