1/31/2019

Sam Judkis
Edmund Dea

CS 340 Final Project Specs

Description

Our goal is to create a website that displays stock market data. When a user logs in, the site will load the user's portfolio. The user's portfolio will contain one or more stock symbols, and they can enter stock purchase details. The website will display the price change and percentage price change for each stock in the user's portfolio.

Outline

I. Project Theme
   A. Stock website
      1. Front-End Design Inspiration - http://www.tradingview.com
      2. Real-Time Stock Quotes using API
         a) https://github.com/iexg/IEX-API/issues
         b) https://www.alphavantage.co/
         c) http://finance.google.com
         d) http://finance.yahoo.com
         e) https://stackoverflow.com/questions/10040954/alternative-to-google-finance-api

II. Querying Stock Data
   A. Use API to query stock data via JSON/etc for most recent price
   B. INSERT INTO <table> VALUES <data>
   C. Then the current logged in user can query the primary stock data table and display stock data.

III. Watchlist
   A. Display a list of stocks in a portfolio.
   B. User can
      1. ADD various stocks to add to their portfolio
      2. DELETE stocks from their portfolio
      3. SORT stock data by ticker/percentage change
      4. UPDATE the portfolio's stock tickers
      5. SEARCH for stocks within the portfolio by ticker ???
         a) Piazza -
            (1) "Every table should be used in at least one SELECT query. For the SELECT queries, it is fine to just display the content of the tables, but your website needs to also have the ability to search using text or filter using a dynamically populated list of properties. This search/filter functionality should be present for at least one entity. It is generally not appropriate to have only a single query that joins all tables and displays them."
            (2) Do they want us to implement a SELECT statement for each entity and/or do they want us to to add search/filter functionality associated with each table?

C. If a stock is removed from the stock exchange, then handle the case of removing the many-to-one relationship by settings all user portfolios containing that stock to NULL. Then, delete that stock from the appropriate table.
D. In a many-to-many relationship, to remove a relationship one would need to delete a row from a table.
E. Displays Data
    1. Stock ticker
    2. Current Price
    3. Percentage Change relative to the day's opening price

IV. Users
A. User can login
B. User's portfolio loads after logging in
C. (optional) Implement SSL connection

V. Optional Features
A. Displays stock data in a chart
    1. Shows OHLC (Open-High-Low-Close)
B. User can add purchase price and # of shared purchased. Then user can UPDATE their purchase price and # of shared purchased.

Database Outline

Entities

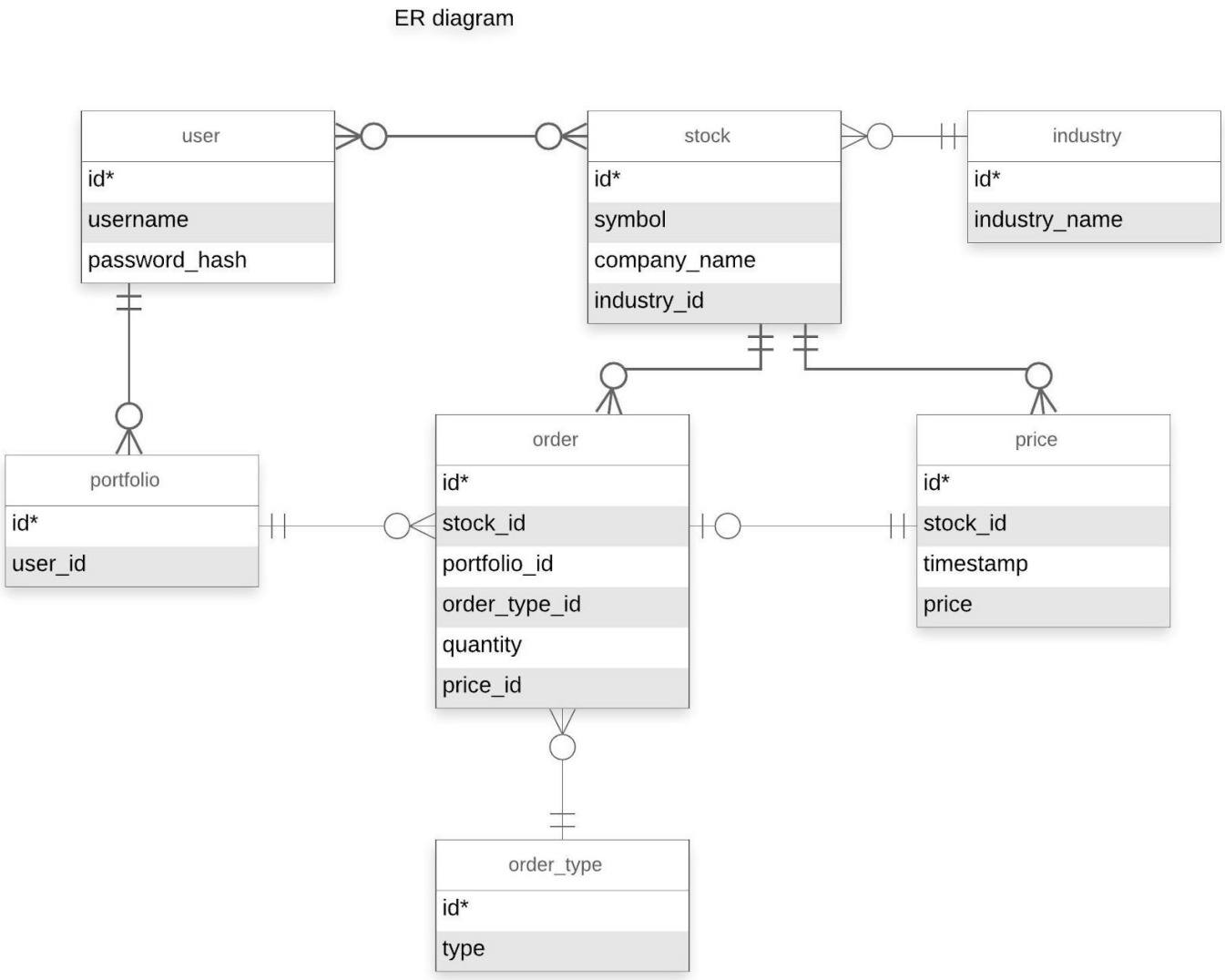
- User
    - Id (primary key) auto incremented
    - Username (varchar, 100)
    - Password_hash (varchar, 255) (optional, nice-to-have feature)
- Stock
    - Id (primary key) auto incremented
    - Symbol (varchar, 4)
    - Company name (varchar, 100)
    - Industry_id (int, foreign key)
- Price
    - Id (Primary key) auto incremented
    - Stock_id (int, foreign key)
    - Date/Time (datetime)
    - Price (dec)
- Portfolio
    - Id (primary key) auto incremented
    - User_id (int, foreign key)
- Order
    - Id (primary key) auto incremented
    - Stock_id (int, foreign key)
    - Portfolio_id (int, foreign key)
    - Order_type_id (int, foreign key)

    - quantity (int)
    - Price_id (int, foreign key)
- Order_type
    - Id (primary key) auto incremented
    - Type (varchar, 15)
- Industry
    - Id (primary key) auto incremented
    - Industry_name (varchar, 25)

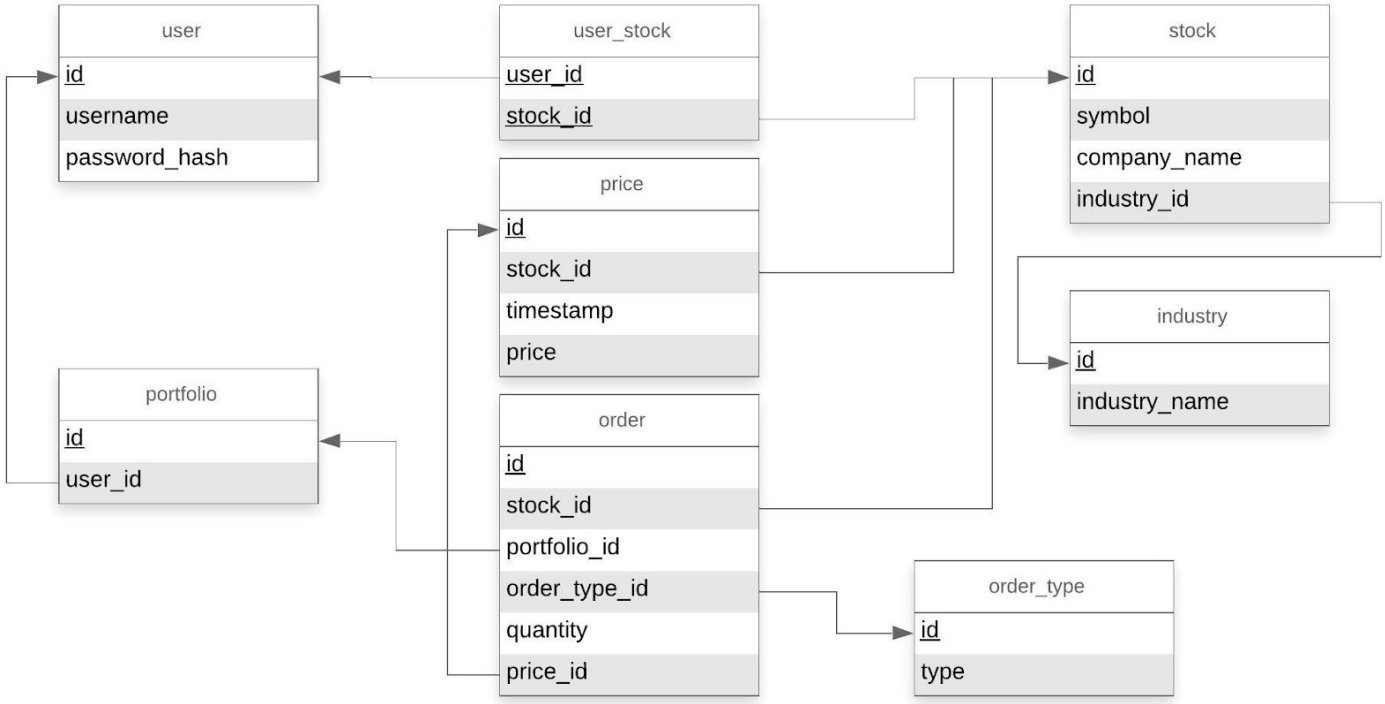
Relationships

- User and stock (many to many)
    - A user can follow 0 or more stocks
    - Stocks can be followed by 0 or more users
- User and portfolio (one to many)
    - A User can be related to 0 or more portfolios
    - A portfolio can be related to exactly one user
- Stock and Price (one to many)
    - A Price is related to exactly 1 Stock
    - A Stock is related to 0 or more Prices
- Order and Stock (one to many)
    - An order is related to exactly 1 Stock
    - A Stock is related to 0 or more orders
- Stock and industry (one to many)
    - A stock is related to exactly one industry
    - An industry is related to 0 or more stocks
- Portfolio and Order (one to many)
    - A portfolio is related to one 0 or more orders
    - An order is related to exactly 1 portfolio
- Order and order_type (one to many)
    - An order is related to exactly one order_type
    - An order_type is related to 0 or more orders
- Order and price
    - An order is related to exactly one price
    - A price can be related to 0 or more orders. This allows for the rare circumstance when 2 users order the same stock at exactly the same time. In this case, both orders will reference the same stock price id.
        - Note: If 2 or more orders are processed simultaneously, then these orders will query for the latest stock price during order execution. The price query function will first check if a current price already exists and, if not, will lock a mutex, query the API for the latest price, and insert the latest stock price into the Price table. Subsequent calls to the price query function will wait on a locked mutex, check if a current price exists, and then reference the same price id without querying the API for the latest price.

ER Diagram


ER diagram
user
id*
username
password_hash
stock
id*
symbol
company_name
industry_id
industry
id*
industry_name
portfolio
id*
user_id
order
id*
stock_id
portfolio_id
order_type_id
quantity
price_id
price
id*
stock_id
timestamp
price
order_type
id*
type

Schema



Feedback by the peer reviewer

1) Amy Stockinger

> Hi Ed,
>
> It looks like we have similar project ideas! That's exciting, to see how someone else approaches certain things. At the same time, I hope I'm not biased by my own project. First off, I love that you made your outline complete with notes, options and considerations to be made based on time resources. I also like that you included links to inspiration and APIs.
>
> I noticed you grayed out some potential modifications, so I have to say that I think it would be good to at least add something like a "portfolio" or "holdings" entity to associate a user with their own stocks. I'm also intrigued that you made a price entity and a purchase entity. Is there a reason that price is a separate entity from stock? Price seems more like an attribute of stock. Unless maybe you planned to use that for the ticker, in which case, should percentage change be an attribute? Industry could also potentially be an entity.
>
> Overall I appreciate that your outline conveys all the thought processes you have for your project. It seems like you're trying to keep it expectations realistic, which is understandable.

2) Nathan Seabourn

> Hi Edmund.
>
> Your outline, ERD, and Schema look great. I don't see any errors. The only change I would make is to have a one-to-many relationship between price and purchase. This would get rid of "stock_id", "purchase_price", and "purchase_date" from the purchase table and add

"price_id". Since the price table has all of that information, there's no need for it to be in the purchase table. It would also make the price table more accurate for that stock since every time a stock is purchased, a new row in the price table would be made for it.

3) Evan Linepensel

Hi Samuel. Thanks for posting your project for review. I like the project idea to display stock market data for users' stocks. The outline for the project is well organized and I like the provided links to references. Overall, it seemed clear what you wanted to implement in the different functions of your database.

I also like the use of APIs to draw in data for some of your tables. This seems like it would make it useful in the "real world" and not just for this class project.

Under your entities, you have an entity Purchase. What if a users wants to sell instead? It appears your greyed out section may cover purchasing and selling. For simplicity, it may not be necessary to do both within the scope of this class project, but something to consider.

It seems like the Stock and Price data will be updated regularly via API, so the data in those tables would be only snapshots in time. You may want to clarify if that is only user initiated or if the system is updating it regularly regardless of user intervention.

Your ERD and Schema look straightforward and are easy to follow. I also like the optional security features of password hash and SSL. Cybersecurity is always a good consideration.

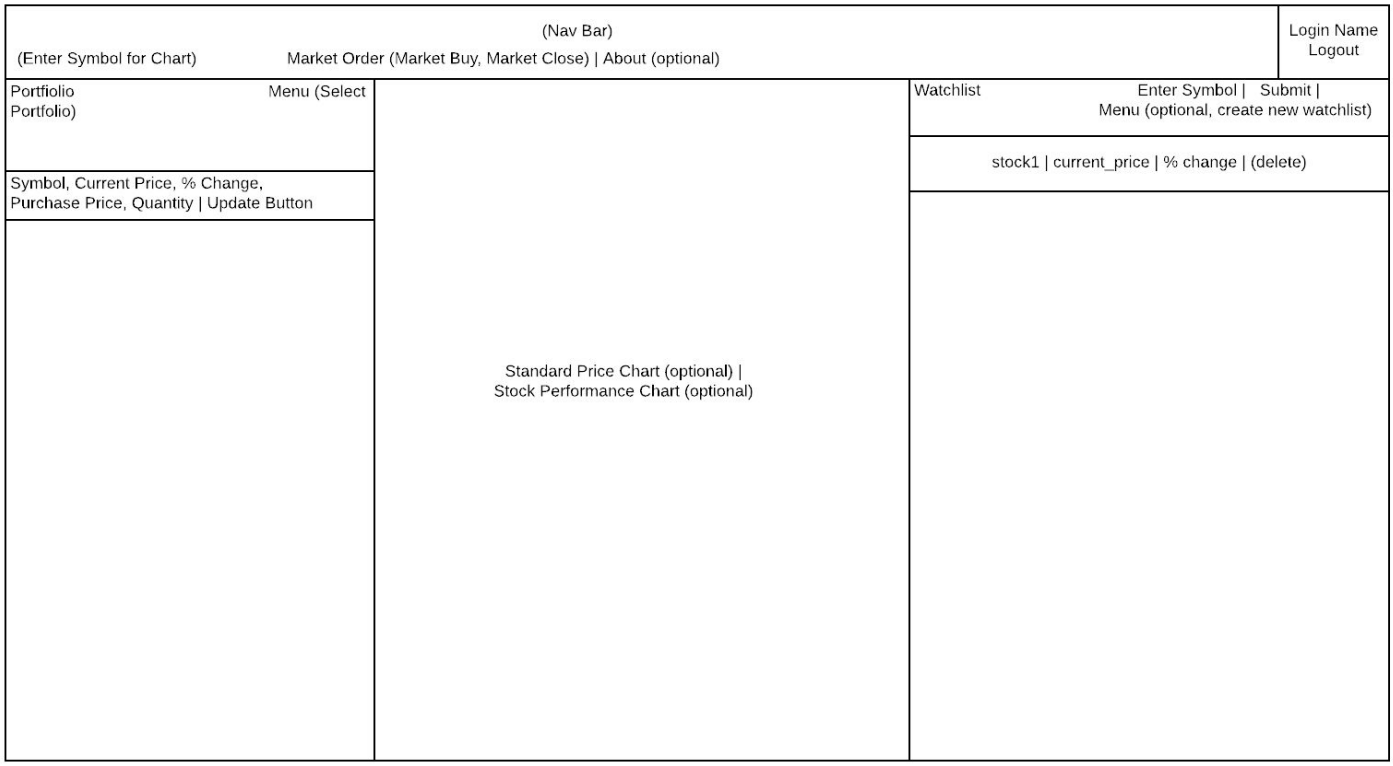
Overall, your documentation looks great. Best of luck on your project.

Actions based on feedback

1) We added a Portfolio table. This allows the user to track one or more groups of stocks, as well as allows a stock to be included in more than 1 portfolio.
2) We replaced the Purchase table with an Order and OrderType table to account for different types of orders, including buying and selling market orders.
3) We separated the industry attribute of a stock into its own entity
4) We replaced the purchase_price in the Purchase table with price_id in the Order table. Rather than storing the price when a stock order is executed in the Purchase table, the stock price and its associated timestamp is stored in the Price table. This change satisfies the 3NF requirement.
5) Amy Stockinger suggested possibly storing the Percentage Change in price relative to the market opening price. We decided not to add the Percentage Change attribute because this type of value should be calculated real-time based on the current stock price.
6) Amy Stockinger asked why we chose to have a Price as its own table. We wanted the ability to add charts that display price over time. Therefore, we needed a table that stored history for Stock Prices and date/timestamp. The goal is to query real-time stock prices, store it in the database, and display charts. Optionally, this would also enable us to add stock indicators, like RSI, MACD, slow/fast stochastic oscillators, and possibly more.
7) Updated ER Diagram above
8) Updated Schema above

Upgrades to the Draft version

1) We created rough template of the front-end.



2) We considered adding Open-High-Low-Close price data to the Price table. Recording OHLC price data is necessary if we want to display stock charts. We considered removing fp_price.price and replacing it with new columns for Open, High, Low, Close. However, we decided that the OHLC price data should be in a separate table with Open, High, Low, Close columns that are foreign keys pointing to fp_price.id. We will evaluate implementing this upgrade in later revisions of our project.
3) In the initial version of our webpage, we will not include stock charts. Stock charts are being worked on in a separate page. Once we are done, we will upgrade the main page by adding a new column that display stock charts, as shown in the rough draft above.
4) We renamed the Industry table to a Sector table since our initial intention was to identify is stock sector. Upon further investigation, we found out that there are 11 major stock sectors and each of these sectors are associated with many industries. For example, Google is associated with the Technology sector.
5) We added a `name` column to the fp_portfolio table. This will allow the user to identify their Portfolio as text with the `name` column. For example, a user could label one portfolio as '401K' and another portfolio as `Brokerage Account`.
6) We wrote a small app in C# to parse stock data and generate a SQL script to input stock price data into the fp_price table.