Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).

Our original proposal included a Logins table, which tracked whenever the user would login or logout, but the Logins table was unapproved because of its telemetry-like structure. We moved forward with the HistoricalStocks table, keeping track of the closing price of all the companies available in our database since 2010. The original plan was to grab information about a company (sector, company name, industry, market cap) by using an API in real time, but, instead, we decided to limit the number of stocks known to the user and keep track of the information in the Companies table. The companies table is made up of only companies in the S&P 500, and as HistoricalStocks is linked to Companies, the close prices are of those same companies. This limit of utilizable stocks is to further simplify the experience for the user.

Discuss what you think your application achieved or failed to achieve regarding its usefulness.

Our application has achieved its purpose in being a simple tool to help users learn about investing. There is no real risk involved, allowing users to try as many different methods as they want through multiple portfolios that initiate with $100,000, allowing lots of leeway, and they can be as risky as they want. The stock prices are updated in real time accounting for the fluctuations in the market. There are a limited number of stocks to choose from, as not to confuse the user with too many options. We have also added a collaboration feature where users can collaborate together on portfolios. The application achieved its purpose of being a learning tool for investing in stocks.

A failure we have encountered is that when using the API, it returns ‘NaN’ when requesting the current price at times. The page will load, but the values will show ‘NaN.’ This can usually be resolved by refreshing the page after a short amount of time to display the correct values. Another is that we were unable to properly host the program on GCP to allow use of the program away from the local machine.

Discuss if you changed the schema or source of the data for your application.

Initially, we had a watchlist as a weak table but in reality, after some advice, we realized it was actually more of a relationship. We included a companies table to make this possible, while also using the table to store information about the companies. The attributes include the company name, sector, industry, market cap, revenue growth, and where the company is established. We also added the historical stocks table to be able to gather more information about those companies as that table consists of the everyday close prices since 2010. These were from a database from kaggle.

Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?

The change for Watchlist from a weak table to a junction table made more sense after the addition of the Companies table. Since we are now holding company info in the database itself, Watchlist should be connected to that because there is a clear many to many relationship between Portfolios and Companies. HistoricalStocks being a weak table was a necessity, since its information is related to a company, as well as a specified date. This allows us to keep company information in the database instead of having to request it using an API, where requests could potentially fail.

Discuss what functionalities you added or removed. Why?

We added a stats page which allows the user to search for companies by name, allowing users to learn the different stocks they have access to. On the stats page we also included sections to search for portfolios heavy in a certain sector (chosen from a dropdown menu), as well as best performer stocks within a certain time frame specified by the user (start and end dates). Finally, the stats page also provides a section listing off the top high market cap investors (users). We primarily included this page to allow the user to get a better idea of the market across time as well as be able to look at the API-provided data through various filters to get a better image of it. Since the program

does allow collaboration, it is a way for users to be let known about others who are doing well.

Explain how you think your advanced database programs complement your application.

Most of our advanced queries are used to display information on our Stats page. This page uses the “top 5 most valuable stocks” query to show the top most valuable stocks and the users who own them. It also uses a version of the “list of covid best performers” query, but with expanded functionality to allow the user to choose the date range they would like to observe. Our “tech-heavy portfolios” query is also implemented, and expanded upon on our Stats page and now allows the user to see portfolios that are heavy in a sector of their choice (choosing from a dropdown menu). Finally, our last complex query was implemented in the watchlist on each of the users portfolios; It allows the user to see the historically lowest and highest price of the stock they are observing.

Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.

Sulabh: I had an issue connecting the database from my local computer to the MySQL instance in Google Cloud Platform. One of the large parts of the problem was that I didn’t know I had to whitelist my IP address. An issue that followed was finding the correct IP address as I attempted to use my private IP address of my laptop but that is not the correct one. The IP address you want to add to the connections of the MySQL instance on Google Cloud Platform is the public IP address which can be found by a simple google search to https://whatismyipaddress.com/. The other part was using the correct name. I had initially used the instance name, when I should have used the name of the actual database where all the tables were stored.

Nikhil:There was some difficulty in trying to pass arguments to the html templates from the flask app file. It was simply because I didn't know how to do that. Eventually, through some googling, I found when rendering templates, I just need to add the arguments as well, making sure the parameters I'm assigning the arguments to are spelled the exact same in the template file. As an example, when passing any arguments, simply add it to the render_template() function as such "return render_template('transaction.html', transactions=transaction_data, portfolio=portfolio_type)" where 'transactions' and 'portfolio' is used in the template 'transaction.html.' A companion issue was how to use the variable in the html file. The solution is to place the variable in double curly brackets (e.g. {{portfolio}}).

Zack: The yfinance API gave us an unexpected issue when trying to display the current information of a stock. Sometimes, it won't output correctly and returns a NaN. This issue forced us to rework our code to account for this random occurrence.

Emilia: I had an issue when trying to send a specified error message from a stored procedure or trigger in mySQL so that it can be received and interpreted on the Python side accordingly. I resolved this issue by finding that I could use SIGNAL SQLSTATE to send a custom error message.

Are there other things that changed comparing the final application with the original proposal?

We mostly kept to the functionality and general plan that we had in our original proposal. The main changes that occurred were not including a table to track login history, as well as adding our Companies and HistoricalStocks tables which are linked to Portfolios via the Watchlist junction table (Companies links to Portfolios).

Describe future work that you think, other than the interface, that the application can improve on.

One option would be to implement the functionality to sell a subset of the number of shares of a particular stock in a user portfolio (as opposed to the total amount initially purchased). This would allow for a more realistic market simulation allowing users to sell part of their share. Another improvement would be the addition of graphs and charts of stock behaviors through time, giving a visual representation and also making it potentially interactive. We could use the information stored in the Historical Data table to plot the graphs.

Describe the final division of labor and how well you managed teamwork.

Division of labor from the project proposal:
Zack Edds: Backend development and API implementation. Developed the API logic for handling stock information such as company's stock price, the percent change and dollar change in the value of their positions and implementing it into the frontend.

Emilia Michalek: Backend and Database Management; in charge of the creation of the schema, writing transactions and triggers used to satisfy stage 4 requirements.

Sulabh Patel: Linked mySQL backend to the frontend Python and html files. Backend and Database Management; in charge of the creation of the schema, including stored procedures and triggers.

Nikhil Thomas: Front-end and UI Design Focus on user interface design and implementation, including front-end development of user account management interface and personalized dashboard, portfolios, and stats page.