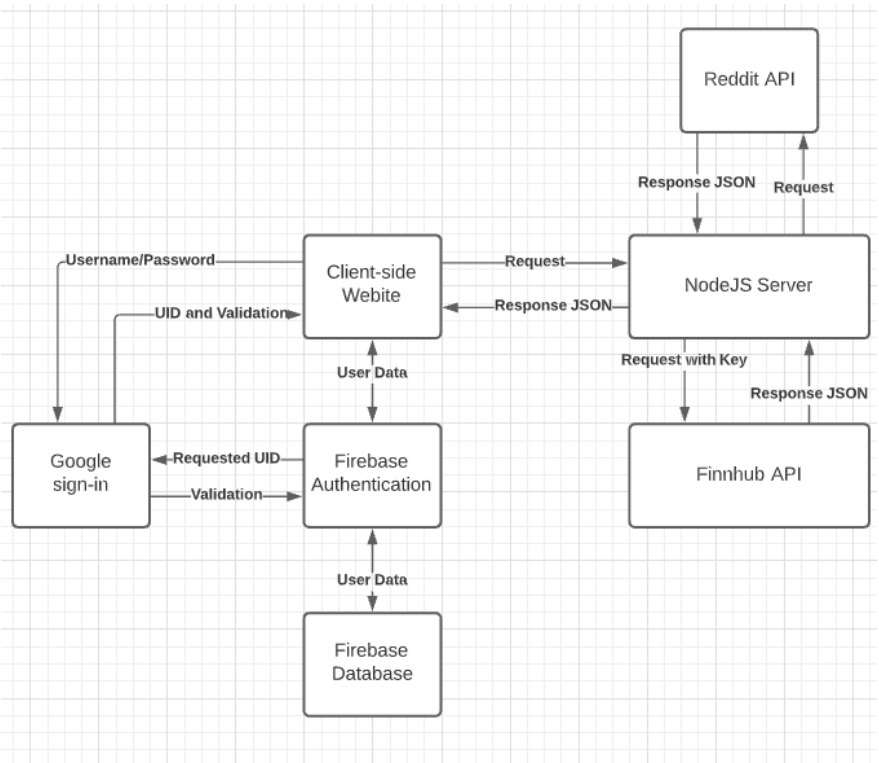
Revised List of Features (Ranked from highest to lowest priority)

- Create an account
    - Users can log in using Google accounts which already provides security.
    - Firebase for database and account management and authentication.
- Visualizing stock data
    - Type into a search bar and the ticker will lead you to a graph and other scrapable, useful data such as volume, market cap, 52 week high and low.
    - Finnhub.io
- Interacting stock data
    - Users can simulate buying and selling of stocks, which will add and remove these stocks from a virtual portfolio.
- Create and view watchlist
    - Users will see a feed of stock data upon logging in, and this data can be customized to display specific stocks.
- View top 10 Reddit feed
    - Will show the title of the post and the image preview. Clicking will open the reddit page in a new tab.

Architecture Diagram:



Front End design:

- https://github.com/CSCI-3308-CU-Boulder/3308SP21_024_5/tree/main/Milestone%20Submissions/Milestone%202/WireFrames
- Wireframes are at this link.

Web Service Design

- We used Finnhub.io for stock data.
    - According to the website, finnhub achieves: Real-Time RESTful APIs and Websocket for Stocks, Currencies, and Cryptos.
    - Specifically I requested close prices denoted in the API as “c” - a List of close prices for returned candles. Which I have written in Finnub.js file along with it’s output for this and all other possible requests.


```
{  // Sample response

{"c":[1.1127,1.10701,1.10655,1.10489,1.10146,1.10325,1.10078,1.10054,1.10194,
1.10479,1.10683,1.10776,1.10716,1.10579,1.10162,1.10067,1.1021,1.09983,1.1005
9,1.10159,1.10784],

"h":[1.11749,1.11397,1.10923,1.10912,1.10549,1.10425,1.10379,1.10198,1.10271,
1.10564,1.10894,1.10834,1.10808,1.10965,1.10863,1.10316,1.1025,1.10234,1.1017
5,1.10278,1.10889],

"l":[1.11242,1.10631,1.10638,1.10353,1.10146,1.10152,1.10019,1.09948,1.09885,
1.1014,1.10457,1.10621,1.10524,1.10516,1.10138,1.10029,1.10057,1.09918,1.0995
8,1.09806,1.10023],

"o":[1.11674,1.1127,1.10737,1.10655,1.10483,1.10215,1.10325,1.10081,1.10013,1
.10199,1.10477,1.10705,1.10766,1.10713,1.10566,1.10156,1.10114,1.10202,1.0999
4,1.10051,1.10228],

"s":"ok","t":[1572818400,1572904800,1572991200,1573077600,1573164000,15734232
00,1573509600,1573596000,1573682400,1573768800,1574028000,1574114400,15742008
00,1574287200,1574373600,1574632800,1574719200,1574805600,1574892000,15749784
00,1575237600],

"v":[46494,60926,46621,69028,46822,31994,44108,46795,44539,36153,39113,34391,
47213,44356,49441,40379,40476,40904,23875,36521,47505]}

}
```

- Reddit API - The request from reddit is pretty straightforward. You request a subreddit and the amount of post you want and the data is returned as JSON. Then parse the JSON to add the links to the page.

Response Attributes:

```
o = List of open prices for returned candles.

h = List of high prices for returned candles.

L = List of low prices for returned candles.

c = List of close prices for returned candles.

v = List of volume data for returned candles.

t = List of timestamp for returned candles.

s = Status of the response. This field can either be ok or no_data.
```

Database design

Our application will use Firebase for its database. This database will hold user data, user authentication, and Google's account services. The data stored in our database will be in JSON format, with each user having a JSON object stored under their UID. We chose to use Firebase instead of SQL or a similar database for several reasons; primarily because it is automatically hosted from Google's servers, saving us time, and because the JSON format it stores is easily accessible in JavaScript and NodeJS.

```
// each user is stored as an object under their UID, which is
// generated by the firebase authentication library,
// as an automatic property of each user
"UID": {
  displayName: "John Smith",            // string
  pfp: "google's_pfp_website/user.png", // string
  ownedStocks: [                        // array
    {
      symbol: "GME",                    // string
      shares: 0.5,                      // float
      timestampBought: 1615334670131    // long
    },
    {                                   // every instance of a "stock" item
      symbol: "GOOGL",                  // has the same format
      shares: 0.1,
      timestampBought: 1615334755989
    }
  ],
  currentCompetitions: {
    "Competition UID": {
      ownedStocks: [                    // more instances of "stock" items
        {                               // stored only for competitions
          symbol: "GME",
          shares: 0.5,
          timestampBought: 1615334670131
        },
        {
          symbol: "GOOGL",
          shares: 0.1,
          timestampBought: 1615334755989
        }
      ]
    }
  }
}


// The competitions' UID is also generated with firebase
// using the ref.push() function
"Competition UID": {
  displayName: "Default Competition Name", // string
  players: [
    {                                     // The player object here is a
                                          // reference to the player, not a
                                          // duplicate of their entire object

      UID: "player UID",                  // string
      balance: 0                          // float
                                          // balance is calculated from the
                                          // currentCompetitions[UID] object,
                                          // and reported here
    }
  ],
  timestampStart: 1615334670131,          // long
  timestampEnd: 1615334755989             // long
}
```

Individual Contributions

- Avery worked on getting the Finnhub API working - We are now able to retrieve data from the API and it is functional on our website. He is now working on integrating this data with the time series graphs. https://github.com/CSCI-3308-CU-Boulder/3308SP21_024_5/tree/main/Code
- John worked on setting up, launching, and integrating the Firebase database into the front-end website
- Thomas - The technical things I have worked on so far are the navbar and the reddit post implementation. I have also taken notes for the weekly meetings and have been keeping up with the project management side of things on Jira. Assisting team members where I can.
    - https://github.com/CSCI-3308-CU-Boulder/3308SP21_024_5/commit/88490298f40a868a2cfc6aca5f5d613b51383682
- Xiang worked on the game page that allows users to stimulate buy and sell stock

Challenges

- Identify at least 3 challenges and/or risks to your project at this point.
    - Avery had a problem getting Finnhub to work, using the example requests from their online tester so I asked Cory for help and we got it resolved.
    - Implementing all features to the full extent before the deadline
    - Hosted Nodejs server being implemented in time.
- What is your backup/risk mitigation plan in case you are unable to resolve the challenge(s)?
    - Replacing with other mature APIs that provide same feature
    - If needed implement the bare bones for the features to work. Worstb case scenario drop the feature.
    - Backup plan would be to run things client side.