Software Requirements and Design Document

For

Group 5

Version 2.0

Authors:
Darbi Bauer
Jacob Dienger
Aidan Ryan
Katie Skipper
Sarah Rosenfeld

1. Overview

We are creating a mobile application for stock recommendations. Based on price movements and news events, our app will recommend stocks and options from the NYSE and NASDAQ. We will provide any relevant information about price drops/increases. Each user may control which stocks they monitor, and will be given alerts based on “out-of-the-norm behavior” as opposed to predefined alerts. The app’s GUI is implemented in Java, using Android Studio. We will use Twitter to monitor news events and Google’s NLP engine to perform sentiment analysis. All backend computations will be performed using CUDA C++ for math, and Python for the Google NLP and Twitter API.

2. Functional Requirements

High Priority (1):

The app at the minimum requires data to be pulled from Yahoo’s Finance website, put into the backend server for calculations, the relevant data pushed into our database, and finally the data needs to be pulled from the database to the Android app. For pulling the data, we will create a script that pings Yahoo’s Finance website for each ‘stock’, ‘call’, and ‘put’. This data is immediately sent to our backend server, where calculations are performed to figure out Moving Average, Weighted Volatility, etc. The calculated data is sent to our database and sorted by stock ticker symbol; the database is hosted in Firebase. The database is connected to the Android app, and data for ‘stocks’, ‘calls’, and ‘puts’ are pulled directly from the database to be shown to the user as an option for whether or not to follow the specific stock.

Mid Priority (2):

The app also uses a Twitter feed specified by the user to get an overall sentiment score about topics that the user feels would affect stock prices. To do this we are calling a Twitter API to gather tweets from the specified source - e.g. the term puppies, sending the gathered tweets to Google’s Natural Language Processing API to get a sentiment score, the score is then sent to our backend server to be included in a calculation. This calculation gives an idea of how well a stock is performing. To pull tweets we must create a script that takes in a user specified term, it then pulls tweets containing the specified term, and sends them to another script dealing with Google’s NLP. The Google NLP takes in a file and sends it to a function that deals with Google’s NLP, and returns a sentiment score and magnitude back. These scores depend on what emotions are associated with the tweets that are pulled in. The scores are normalized and sent to the backend server, where they are incorporated into a stock’s ‘wellness score’.

Low Priority (3):

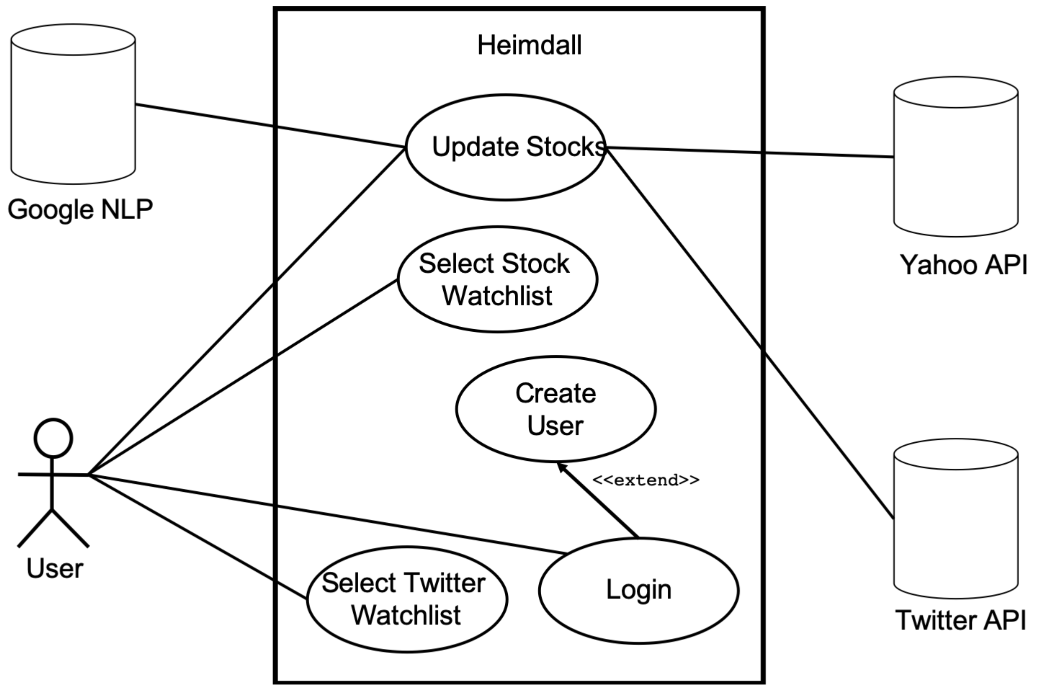
To allow users to save stocks to the home screen, we must implement a user login system which stores user information such as a username and password. The login functionality of the app uses Firebase Auth to create, store, or change user data. The app interfaces directly with Firebase Auth to resolve any user interaction. Firebase Auth will store username, password, and a list of the stocks and Twitter trends to follow. This is the lowest priority currently because we first need to make sure that there’s data to pull from the database.

3. Non-functional Requirements

- Unified Design -
    - Minimal number of fonts used
    - App format that follows common conventions (e.g. To reset a password, click the ‘Reset Password’ link because this is similar to other popular apps)

    - Page to page design is similar (e.g. Back button always in top left corner and leads to last screen)
    - Common color palette throughout app
        - To implement this, specify a list of 5 colors to use and where to use each color
- Stock analysis refresh rate should be at most 1 minute
- Securely store user password information
    - Since we are using Firebase Auth, this is taken care of automatically. Passwords not stored using plain text.

4. Use Case Diagram



Textual descriptions of use cases:

- *Update Stocks*
    1. **Unique name:** Update Stocks
    2. **Participating actors: User, Google NLP, Yahoo API, Twitter API**
    3. **Entry conditions: User** views watchlist in Android application
    4. **Exit conditions:** New stock information is updated and displayed in application
    5. **Flow of events:**
        - **User** views watchlist in Android application
        - All stock information is updated in database using information from the **Google NLP**, **Yahoo API**, and **Twitter API**
        - New information is displayed to **User** in application
    6. **Special requirements:** None

- *Login*
    1. **Unique name:** Login
    2. **Participating actors: User**
    3. **Entry conditions: User** selects login button on the welcome screen

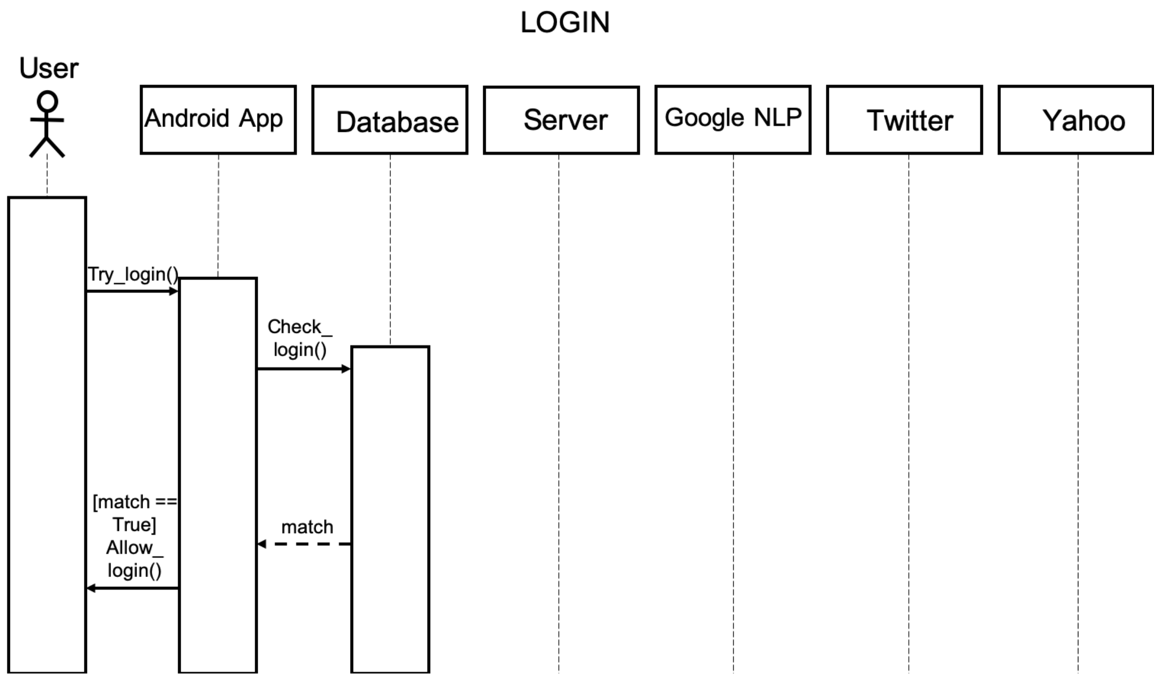
4. **Exit conditions: User** is either granted or denied login access
5. **Flow of events:**
    - **User** goes to login screen and types credentials
    - **User** presses the login button
    - Database is queried for matching login credentials
    - If matching credentials are found, **User** is allowed access; otherwise, access is denied
6. **Special requirements:** None

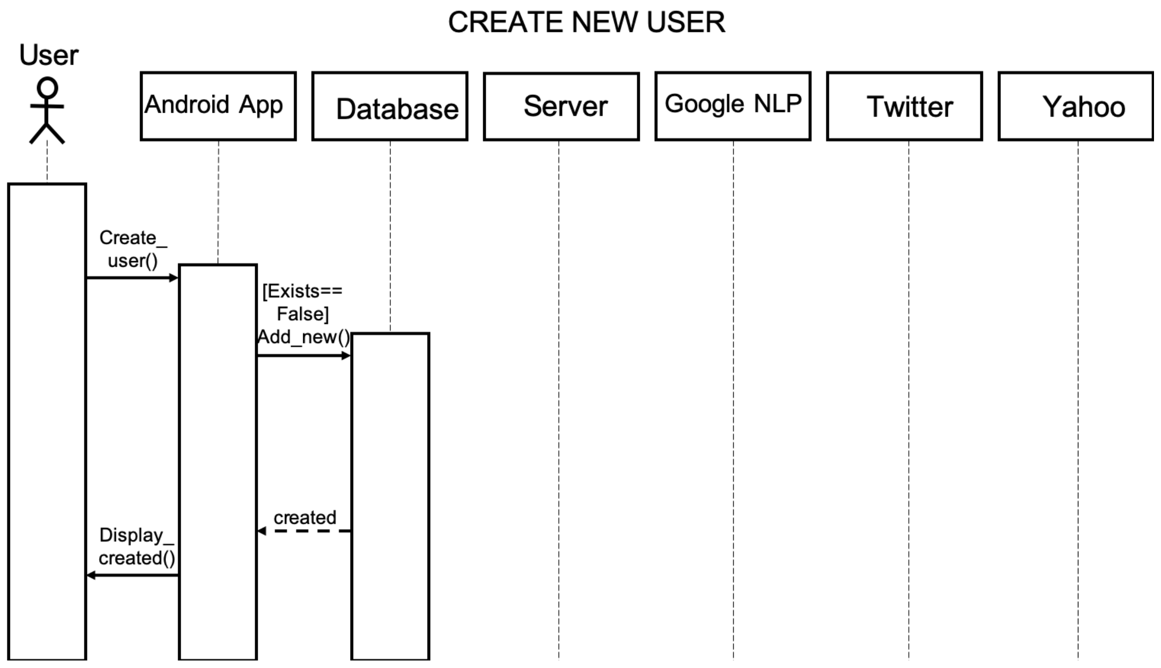
- *Create User*
    1. **Unique name:** Create User
    2. **Participating actors: User**
    3. **Entry conditions: User** selects new account button on the login screen
    4. **Exit conditions:** Either a new **User** is created, or permission is denied
    5. **Flow of events:**
        - **User** goes to login screen and selects create new account
        - **User** inputs information to create new account
        - Database is queried for matching login credentials
        - If no matching emails are found, a new **User** is created and a confirmation email is sent; otherwise, access is denied
    6. **Special requirements:** None

- *Select Stock Watchlist*
    1. **Unique name:** Select Stock Watchlist
    2. **Participating actors: User**
    3. **Entry conditions: User** goes to stock list in application and selects add option
    4. **Exit conditions:** New company is added to **User's** stock list
    5. **Flow of events:**
        - **User** goes to stock list in application and selects add option
        - **User's** existing list is queried to check for matching stocks
        - If no matching stock is found, data is pulled from **Yahoo API** and added into the database containing the **User's** stock watchlist
    6. **Special requirements:** None

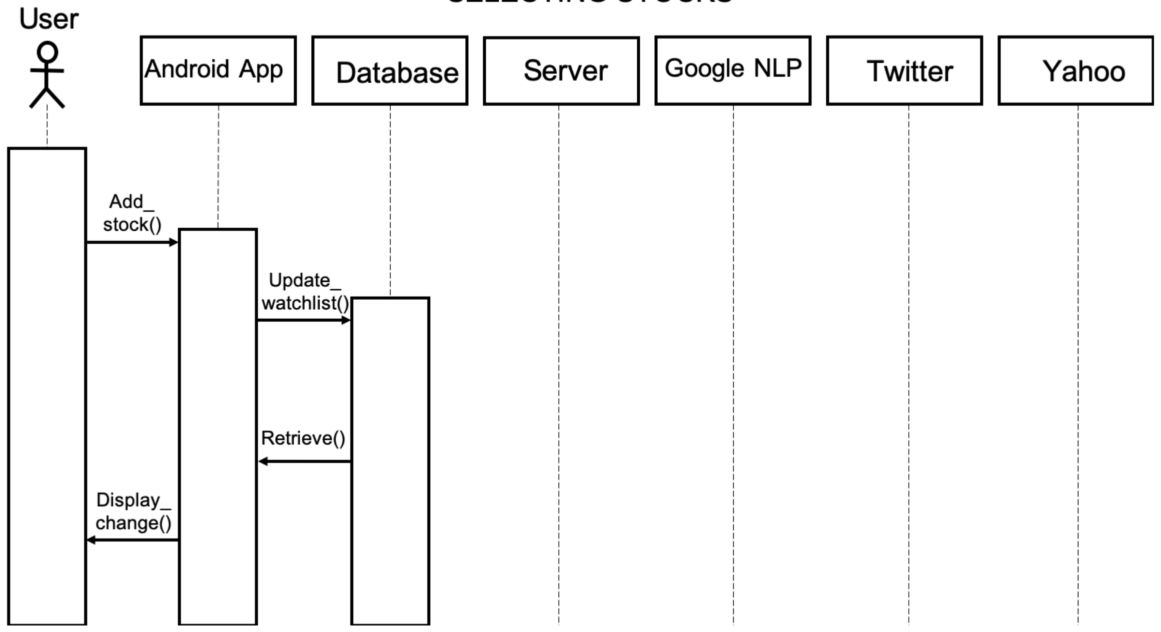
- *Select Twitter Watchlist*
    1. **Unique name:** Select Twitter Watchlist
    2. **Participating actors: User**
    3. **Entry conditions: User** goes to twitter watchlist in application and selects add option
    4. **Exit conditions:** New search term is added to **User's** twitter watchlist
    5. **Flow of events:**
        - **User** goes to twitter watchlist in application and selects add option
        - **User's** existing list is queried to check for matching twitter terms
        - If no matching term is found, data is pulled from **Twitter API** and added into the database containing the **User's** Twitter watchlist
    6. **Special requirements:** None

5. Class Diagram and/or Sequence Diagrams

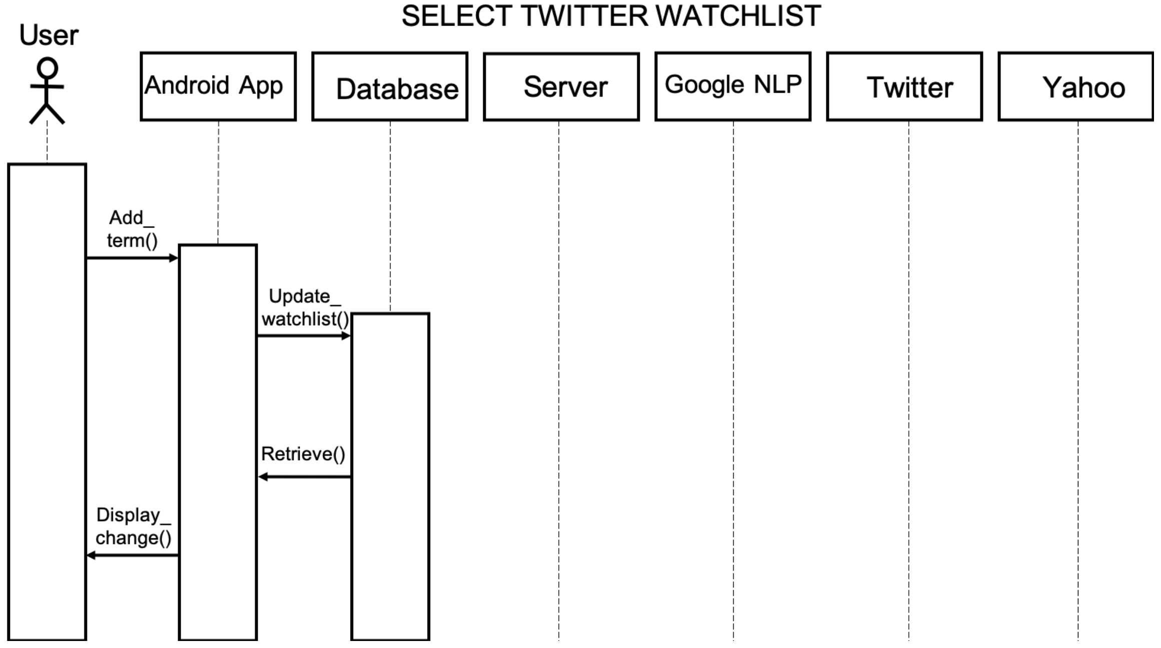


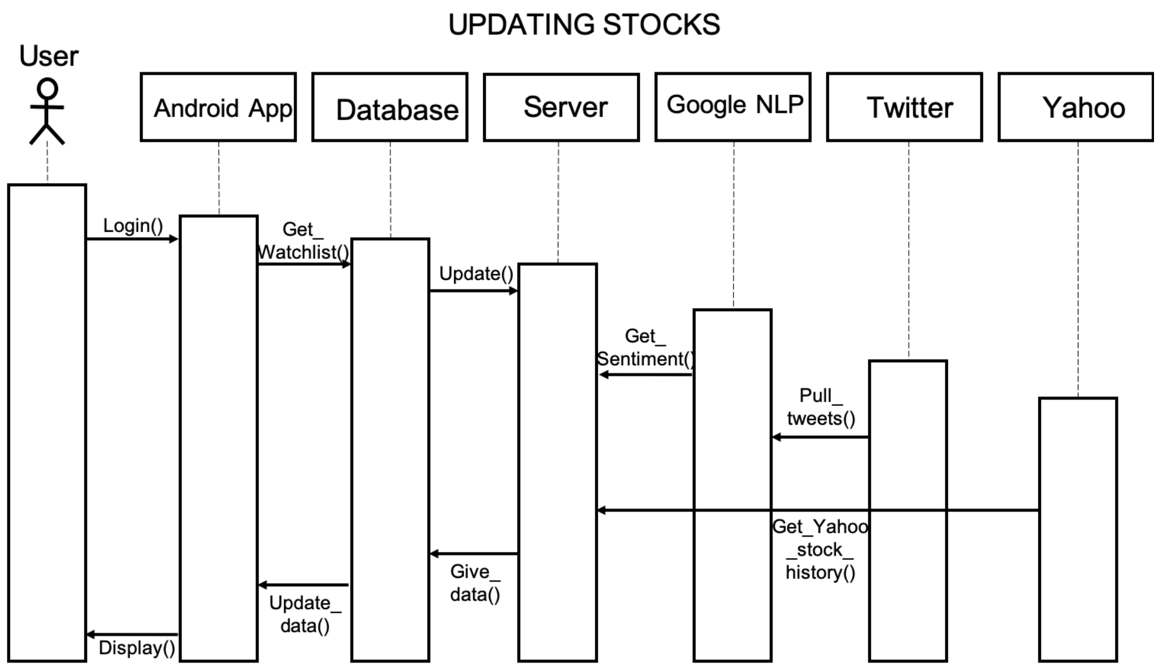


SELECTING STOCKS



SELECT TWITTER WATCHLIST





6. Operating Environment

Our users will interact with the Android app interface, which uses data from our database. The database must pull the stock analyses from our backend server, which contains the actual data that we want to show users. Our backend server is using mathematical formulas to rate stock performance based on stock history. Stock performance ratings are also based on Twitter feed sentiment analysis, which is pulled from a program that will also be running on the backend server.

Twitter -> Google NLP -> Backend Server
Yahoo -> Backend Server : Mathematical Formulas[takes in data from Yahoo and NLP] -> Database
Database <-> Android App

7. Assumptions and Dependencies

- Yahoo’s Stock API properly updating and allowing data to be pulled
- Twitter for Python library pulling tweets
- Google’s NLP processing our data
- Our backend server processing the stock data that we push, and returning the analyzed data back
- The UI may look different on other phones than what we test on, which could be an issue if an element doesn’t fit on the screen