CSC 431

SAPP

System Architecture Specification (SAS)

Team #18

Daniel Portuondo	**Requirements Engineer, Analyst**
Ammar Jivraj	**Front-End Designer, Developer**
Alexandra Syunkova	**Project Manager, Quality Assurance**

Version History

Version	Date	Author(s)	Change Comments
0	02/23/21	Ammar Jivraj Daniel Portuondo Alexandra Syunkova	Functional/Non-functional requirements
0.1	03/09/21	Ammar Jivraj Daniel Portuondo Alexandra Syunkova	System Constraints Requirements Modeling Evolutionary Requirements
0.2	03/30/21	Ammar Jivraj Daniel Portuondo Alexandra Syunkova	System Architecture
0.3	05/3/21	Ammar Jivraj Daniel Portuondo Alexandra Syunkova	Update System Design

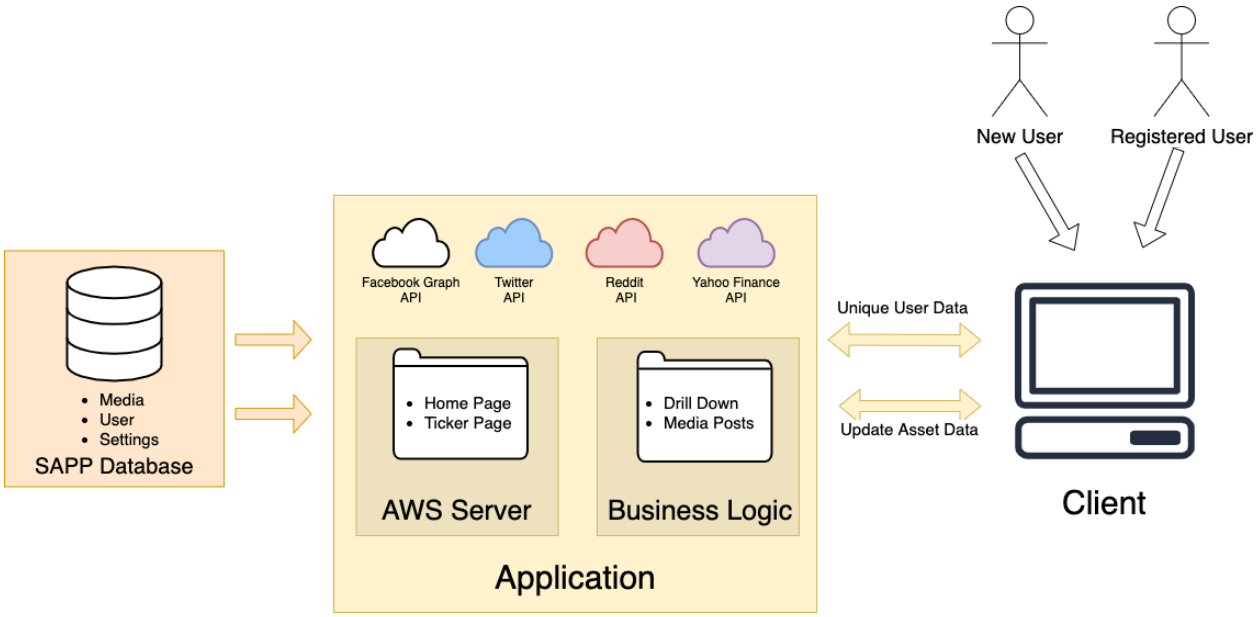
Table of Contents

1. System Analysis

1.1 System Overview

SAPP will implement a 3-tier architecture system. SAPP's own personal database containing user information, preferences, and historical data will occupy the database layer. The business logic, including customized homepages and performing searches, will be handled by the middle layer - the application. At this layer, resources and data from relevant API's (Twitter, Reddit, Facebook Graph, Yahoo Finance) will be pulled and applied to an in-house designed sentiment analysis algorithm using Python scripts and libraries. This layer will also hold the AWS server which contains all the necessary HTML, CSS, and Angular webfiles to provide the UI and UX for the user at the client layer. The client layer will send and receive requests to the layers below it, such as logins and sentiment score updates.

1.2 System Diagram



1.3 Actor Identification

1.3.1 Primary Actors

1.3.1.1 New User

The New User is an external user without an account in the system. This actor uses the system to create an account (thus becoming a Registered User) and/or browse the pages accessible to users who are not logged in. As they will be the recipients of the system's services and responsible for triggering use cases, they serve as one of the primary actors.

1.3.1.2 Registered User

A Registered User is a user with an already existing account in the system. This actor uses the system to log in and then browse pages accessible to logged in users. Registered users will also be responsible for triggering use-cases and will be able to take advantage of more of the system's services than new users, making them another primary actor.

1.3.2 Secondary Actors

1.3.2.1 Social Media API

Social media API's such as the Reddit, Twitter, and Facebook Graph API serve as secondary actors because they support a new or registered user's request for sentiment analysis data. SAPP requires the support of these API's to complete critical use cases such as Pie Chart Drill Down.

1.3.2.2 Stock Data API

The Yahoo Finance API serves as a secondary actor by supporting users will up to date price information with the ticker page's price chart as well as the customized homepage.

1.3.2.3 AWS

The AWS server is a secondary actor due to its ability to store all web files required for the user to interact with SAPP. It is critical for providing a comprehensive way to manipulate, requesting, and protecting the back end

1.4 Design Rationale

1.4.1 Architectural Style

The 3-tier architecture style will be implemented due to its popularity and efficiency in web and database applications. SAPP will require several intensive yet necessary operations to provide quality services to its users. It will utilize a regularly updated stream of data from social media APIs for hundreds of thousands of posts regarding hundreds of tickers and cryptocurrencies, which must be consistently fed through the filter of the sentiment analysis algorithm. It must also be responsible for pulling real-time price data for price history charts. Due to the size of this responsibility, it seems most effective to consolidate the majority of them into a single layer, the application layer, as they all fall under the category of application logic. To impose all of these operations onto the database would likely be too computationally expensive and ultimately affect performance. Therefore, this layer can serve as the middleman and medium of communication between the client and the database, existing as the source for API's, web files, and the sentiment analysis algorithm while remaining logically consistent and conceptually understandable. Additionally, it supports distributing the load of requests from one layer to another, for client requests that do not need to reach the database and vice versa. This ultimately allows for a faster application, which is critical for the sheer volume of data SAPP must handle at any given moment.

1.4.2 Design Patterns

1.4.2.1 Facade

The facade design pattern is applicable to our system because it helps maintain a relatively simple user interface without allowing the user to access or alter the underlying functionality. A ticker info page provides detailed information about a ticker. The maintenance of every individual page requires a complex underlying system that communicates with a social media API, analyzes social media posts, and calculates the sentiment score for the ticker. However, the user can only view the information that is presented and whatever additional information they request to see, with limitations.

1.4.2.2 Decorator

We intend to apply a decorator design pattern to the users' notification settings and ticker pinning system. The notification system must allow the registered user to customize the frequency and type of notifications sent to them. Moreover, registered users must be able to

choose which tickers to add to their watchlist. The decorator pattern applies here as the user needs to be able to add on new functions without altering the already existing ones.

1.4.2.3 Observer

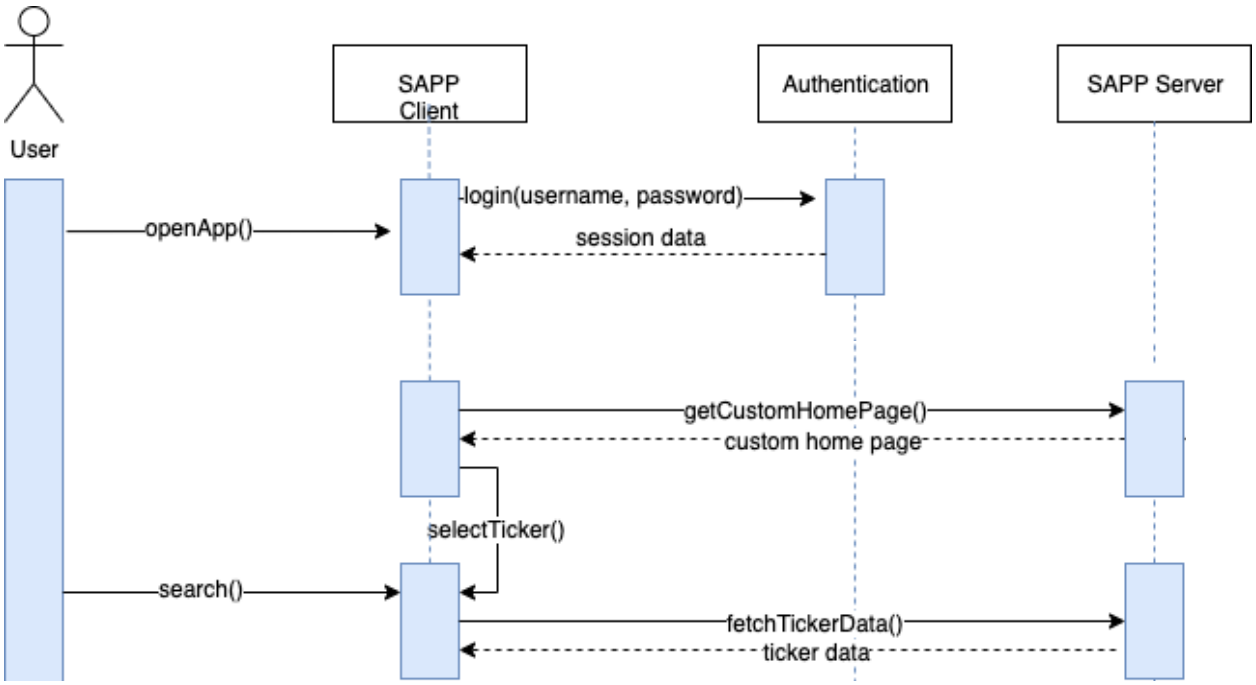
The observer design pattern is needed to automatically update the statistics of each ticker in response to changes to stock data and social media data linked to it. When the database of social media posts - the subject - changes state, which it is expected to do at least once every 30 seconds as new posts appear or the statistics of some posts change, its dependents are updated automatically (meaning the sentiment score, pie chart, pie chart drill down, are updated). This way, the tickers become observers of the social media and stock data APIs; this pattern is then convenient because it permits adding new tickers or removing unsuccessful ones with less hassle.

1.4.3 Framework

The MEAN framework will be used primarily due to familiarity with Angular, Node.js, and Javascript. The MEAN framework's greatest strength is its ability to be written in one language, Javascript, which allows for faster development of the final product. Because our team is familiar with Javascript, this will increase development efficiency further. To date, cryptocurrency relevance is at an all time high with Bitcoin continuing to surge and the bursts of purchases of GameStop stock from r/WallStreetBets tread on. Therefore, to improve the application's potential for success, it is best to get it deployed quickly to leverage the overwhelming popularity its subject matter is currently experiencing.

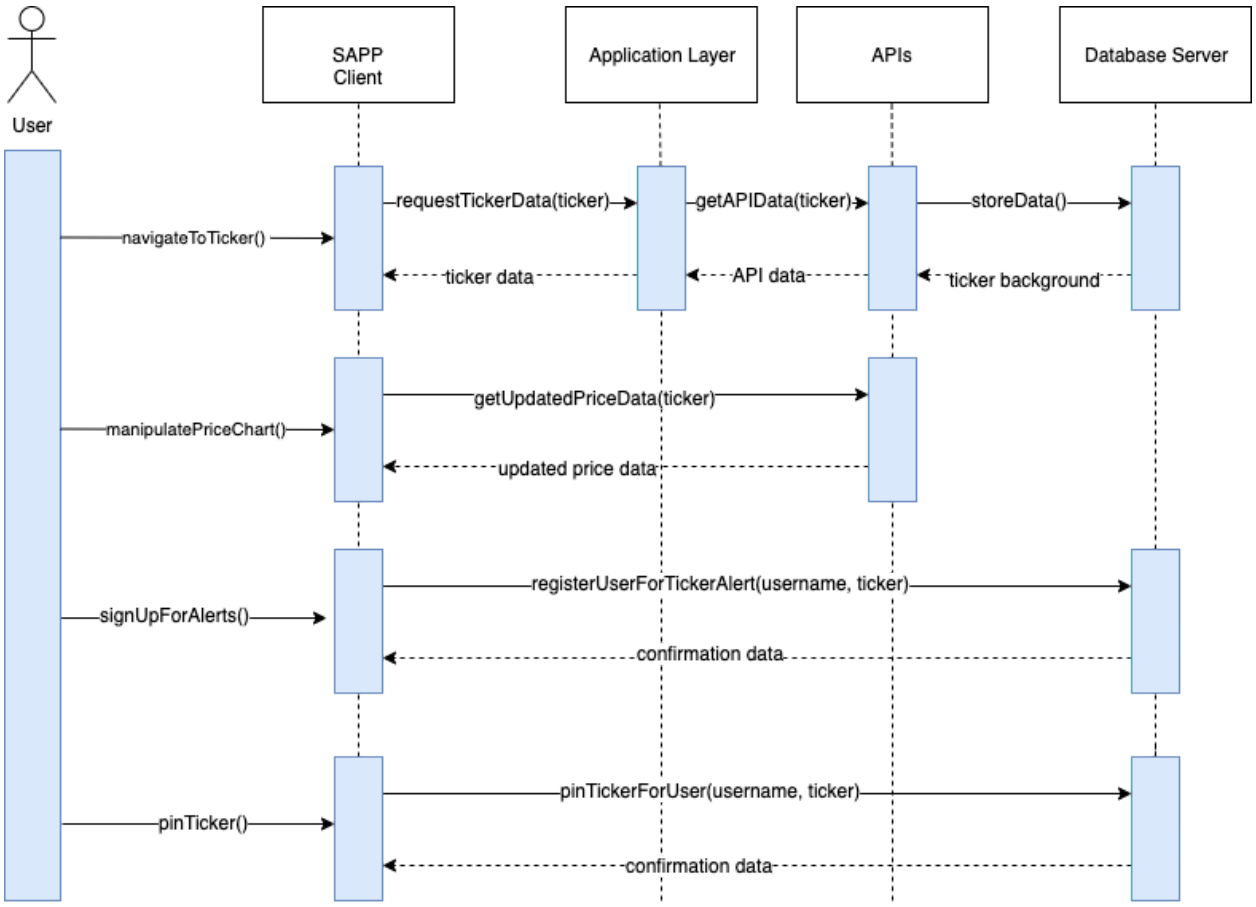
2. Functional Design

2.1 General Design



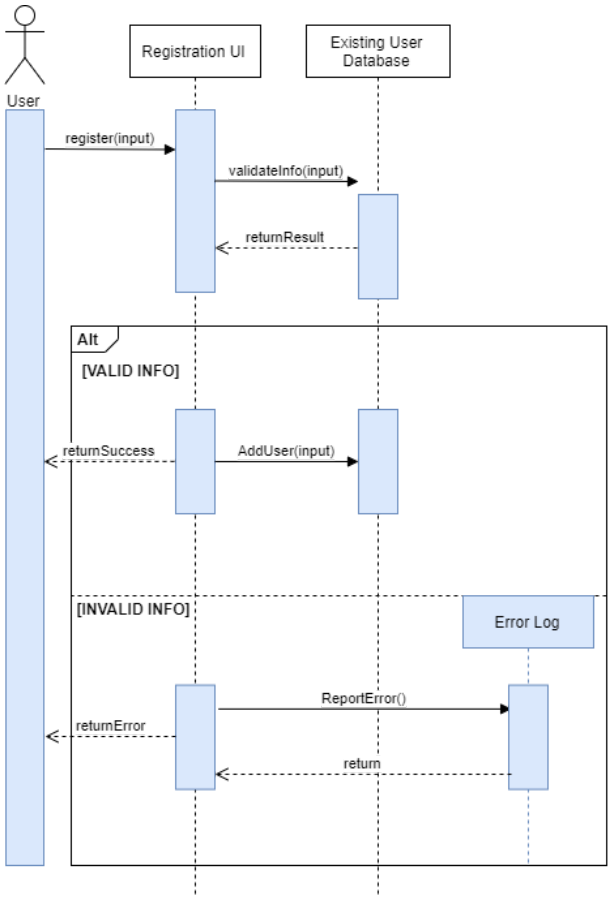
2.2 Ticker Page Sequence Diagram

Included functional requirements: Manipulate Price Chart, Sign Up for Email Alerts, Pin Ticker



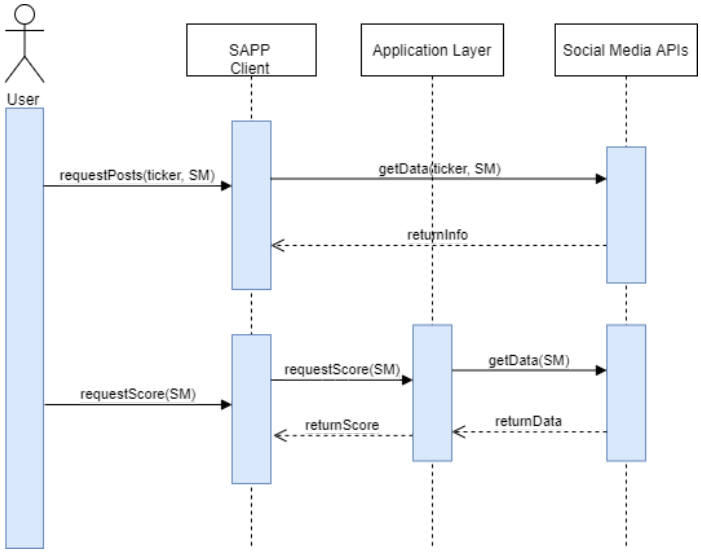
2.3 Account Creation Sequence

Included Functional Requirements: Login, Register



2.4 Pie Chart Drill Down Sequence

Included Functional Requirements: Pie Chart Drill Down, Social Media Post Exploration



3. Structural Design

3.1 Components

-User

-Home Page

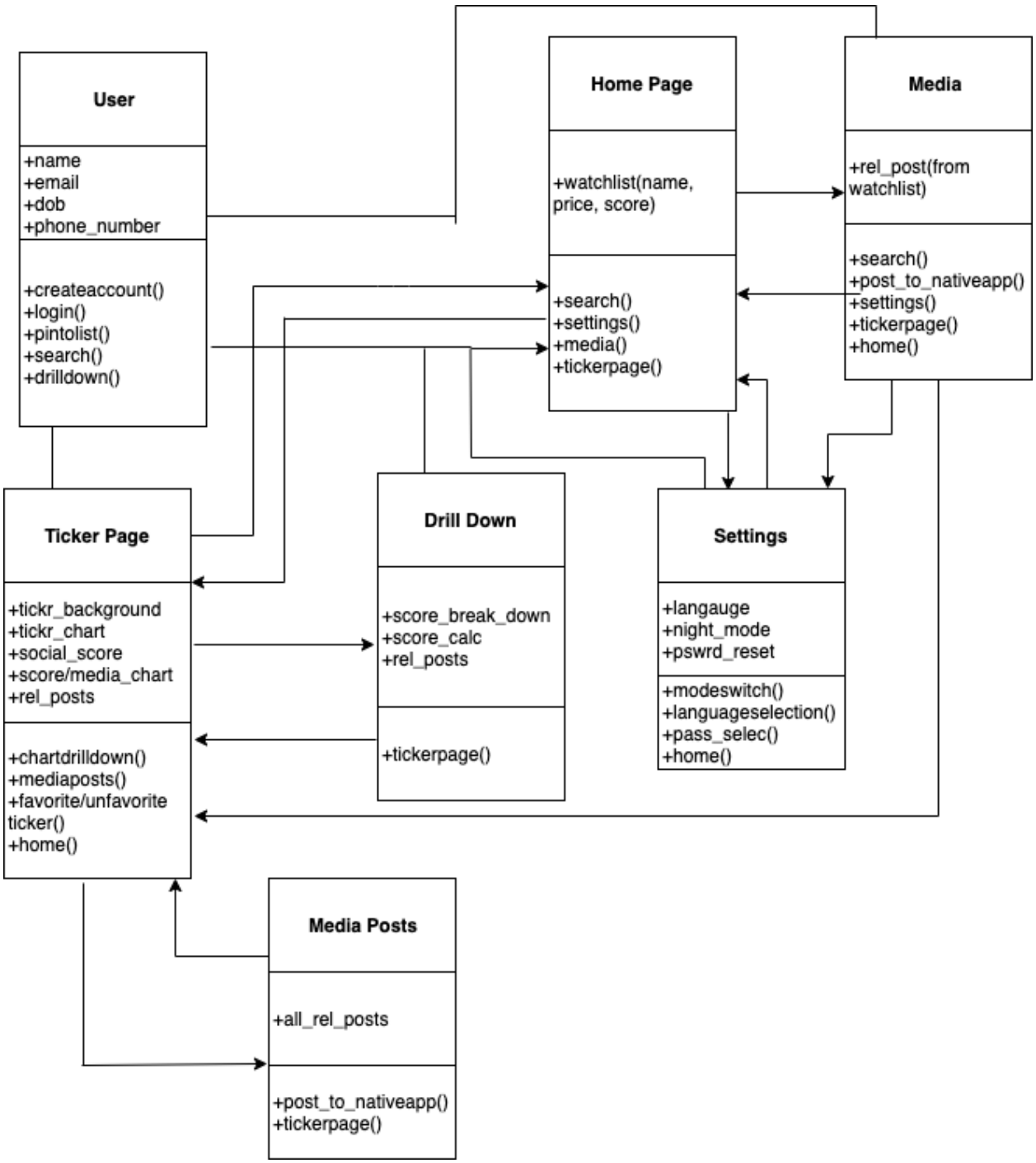
    -Mobile and Web format

-Ticker Page

-Drill Down

-Settings

3.2 Diagram


User
+name
+email
+dob
+phone_number
+createaccount()
+login()
+pintolist()
+search()
+drilldown()
Home Page
+watchlist(name, price, score)
+search()
+settings()
+media()
+tickerpage()
Media
+rel_post(from watchlist)
+search()
+post_to_nativeapp()
+settings()
+tickerpage()
+home()
Ticker Page
+tickr_background
+tickr_chart
+social_score
+score/media_chart
+rel_posts
+chartdrilldown()
+mediaposts()
+favorite/unfavorite ticker()
+home()
Drill Down
+score_break_down
+score_calc
+rel_posts
+tickerpage()
Settings
+langauge
+night_mode
+pswrd_reset
+modeswitch()
+languageselection()
+pass_selec()
+home()
Media Posts
+all_rel_posts
+post_to_nativeapp()
+tickerpage()