Workshop Finance: The Art of Valuation

Prithika Ganesh, Ignacio Nunez Gomez, Suyash Bhatia, and Zen Mae Lee

Abstract—Workshop Finance is a deal idea generation tool, allowing users to quickly build valuations. The application provides a recommendation engine leveraging a broad set of third-party fundamental data to identify the best levers (e.g. similar companies, trading multiples, time periods) for a given company. Workshop then performs the necessary calculations and presents results through a visualization called a football field, a graphic which typically appears in investment banking pitch books. The mobile application allows users to view, refine, and share this football field directly through the app. Workshop will enable deal idea generation, client communication (i.e. pitching to clients, initializing cases), and deal execution. Workshop Finance will become the primary medium through which users "tell the story" of a company's valuation—the key point in any corporate finance discussion. This first-of-its-kind application will power the art of valuation through a combination of strict corporate finance theory and innovative user experience.

Index Terms—Company Valuation, Market Data, Football Fields.

1 Introduction

WHILE there are already tools and methods to value companies using a football field chart, none of them are fast, dynamic, and responsive. Additionally, such tools rarely evaluate both private and public companies, yet financial corporations typically require valuations of both types of companies. The core of Workshop Finance is to create a mobile application for comprehensive corporate finance valuation. Workshop Finance aims to make generating football fields more efficient by providing initial valuations at the consumer's fingertips while modernizing the old practice of using numerous Excel sheets and math formulas.

The application's ability to calculate and deliver comparable valuations in a football field within minutes enables workers in the financial sector to make informed business decisions much faster with baseline metrics. Within the application, the user will select the target company to value. If the company is public, the application will automatically fetch its public financial data from an API using customer relationship management integration. This API will provide all of the public market data needed to precisely calculate valuations. If the user decides to value a private company, the user will need to manually input its fundamental financial data.

Once the desired company to value is chosen, the app produces valuation models through both relative and intrinsic value methodology. The user will then be able to adjust the company's value calculations and visual output by adjusting financial variables, such as by adding several companies from the specific industry to create a financial comparison (COMP) or by editing the index's multiplier. It is a priority that the valuation is dynamic and fast. The user will be able to share this visualization on social media platforms, send it via email, or export and save it as a JPEG document. Workshop Finance will be the first of its kind, determining how much a company is worth today, or at some point in the past.

2 Concept Development

2.1 Engineering Understanding of Client's Problem

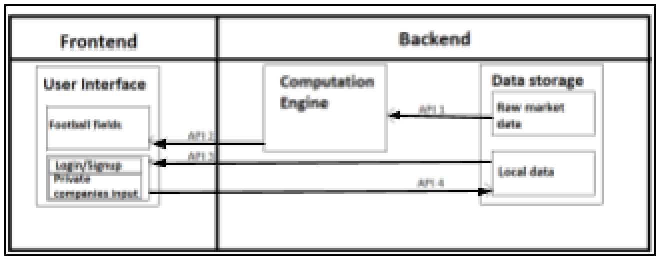
The client's fundamental problem is their need to create a convenient way for users to draw football fields in a short span of time, such that a user would be able to analyze these visualizations on the go. Workshop Finance has been the client's passion project for several years to date, and they have created a web-based prototype which accurately models the necessary financial calculations for company valuations. To achieve convenience of the application, the team needs to create a mobile application accessible to both iOS and Android users which would be able to perform all the required functions modeled by the client's web application prototype. From an application development and software engineering perspective, the team needed a cloud-hosting software, APIs, and to develop a fully functional frontend with a logical user flow (UX) from the ground up based on wireframes provided by the client. Users would then be able to wirelessly create company valuations on their mobile phones.

2.2 Conceptual Approach

Because a web application did not fulfill the client's requirements, the team's initial and sole conceptual approach was to develop an iOS and Android mobile application by using ReactNative as the primary frontend framework as it caters to both softwares. A comprehensive list of the engineering requirements for Workshop Finance including metrics for ideal local data storage and stretch goal features is listed below under the Appendix in Section 7.1.

3 System Description

Fig 1: Block diagram of Workshop Finance App system



The overall project system is divided into two main parts: a Javascript user-interface frontend and a multi-block backend. The backend has two main components: the computation engine developed in Python, and the SQL data storage hosted in IEX Cloud. The data storage is divided into two parts: raw market data and local data on the application. To establish communication between these different blocks, we will use CRM integration with APIs. The APIs which connect the data storage with the rest of the application (API 1, API 3 and API 4), will be provided by IEX Cloud. The API that connects the computation engine to the user interface (API 2) will be generated using Flask, a Python framework which was used to develop the computation engine.

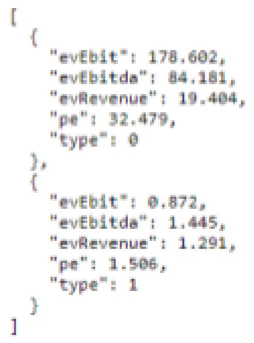
In order for the user interface to fulfill its main purpose of visualizing the football fields to value a specific target, the frontend first needs to obtain its calculation data. However, the financial metrics needed to generate those valuations (such as the PE ratios or the EV/EBITDA metric) are calculated in the computation engine. Therefore, once these are calculated, the user interface obtains them via API 2. At the same time, for the computation engine to be able to make those calculations, it needs access to market data to obtain the companies' raw financial data. For example, if the computation engine needs to calculate the PE ratio of a company, it first has to obtain the price and earnings data of that company. The computation engine will obtain that data from IEX Cloud using API 1.

Additionally, the app needs direct communication between the user interface and the application database. For example, when a new user is created in the sign up page, they will be directly added to the app's database. When a user tries to log in, we will need to verify whether they are already in the app's database or not. If that user already has an account, they will need to access their football fields, with their respective valuations, set of comparables and targets. The information about each user's football field has to be stored in the app's database, and provided back to the user interface whenever the user needs it.

Moreover, the application will have the ability to set a private company as a football field's target. In that case, the user will have to introduce the necessary financial data manually, since IEX Cloud does not have the ability to provide private enterprises' information. Therefore, if the user introduces a private's company data, the application will store it in the local database. This data will be private, therefore, the user will be the only one able to access the company they manually introduced. All of this communication between the user interface and the local database will be handled by API 3 and API 4. Whenever the user introduces data into the database, API 3 is called, and whenever he obtains information from it, API 4 is used.

To summarize, API 1 (provided by IEX Cloud) will transfer raw market data, such as enterprise value or enterprise earnings from IEX Cloud to the computation engine. API 2 (generated with Flask framework) will transfer calculated financial metrics such as P/E or EV/EBITDA from the computation engine to the frontend. It will have a structure similar to the following:

Fig 2: API 2 JSON output



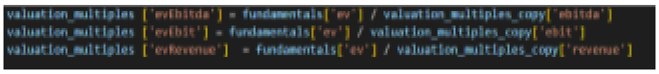
```
[
  {
    "evEbit": 178.602,
    "evEbitda": 84.181,
    "evRevenue": 19.404,
    "pe": 32.479,
    "type": 0
  },
  {
    "evEbit": 0.872,
    "evEbitda": 1.445,
    "evRevenue": 1.291,
    "pe": 1.506,
    "type": 1
  }
]
```

API 3 and API 4 (provided by IEX Cloud) will bidirectionally transfer information about user's authentication and authorization, football fields and valuations, and private data information, between the user interface and the app's local database.

For an individual to use the app, they first have to log in or to sign up. They will select the football field they want to visualize or generate a new one. If the user generates a new one, they will have to decide the football field features such as what type of valuation they want to make, the date in which they want to value, and the football field's target or the set of comparables. Then, the frontend will request the needed values to the computation engine to be able to display this information in the football field chart.

The computation engine provides the calculations layer. It gets the market data and generates some calculations. Some of the basic ones are shown in the code below:

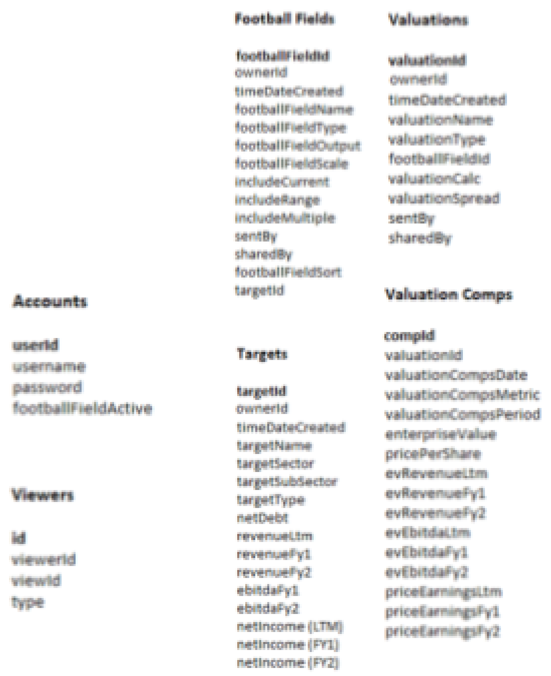
Fig 3: Valuation calculations



```
valuation_multiples ['evEbitda'] = fundamentals['ev'] / valuation_multiples_copy['ebitda']
valuation_multiples ['evEbit'] = fundamentals['ev'] / valuation_multiples_copy['ebit']
valuation_multiples ['evRevenue']  = fundamentals['ev'] / valuation_multiples_copy['revenue']
```

These calculations are sent to the frontend. Moreover, we have also mentioned the two parts that the data storage contains: the raw market data and the application's local data storage. The local database has six tables, with the following structure:

Fig 4: App's database tables



Football Fields

footballFieldId
ownerId
timeDateCreated
footballFieldName
footballFieldType
footballFieldOutput
footballFieldScale
includeCurrent
includeRange
includeMultiple
sentBy
sharedBy
footballFieldSort
targetId

Valuations

valuationId
ownerId
timeDateCreated
valuationName
valuationType
footballFieldId
valuationCalc
valuationSpread
sentBy
sharedBy

Accounts

userId
username
password
footballFieldActive

Targets

targetId
ownerId
timeDateCreated
targetName
targetSector
targetSubSector
targetType
netDebt
revenueLtm
revenueFy1
revenueFy2
ebitdaFy1
ebitdaFy2
netIncome (LTM)
netIncome (FY1)
netIncome (FY2)

Valuation Comps

compId
valuationId
valuationCompsDate
valuationCompsMetric
valuationCompsPeriod
enterpriseValue
pricePerShare
evRevenueLtm
evRevenueFy1
evRevenueFy2
evEbitdaLtm
evEbitdaFy1
evEbitdaFy2
priceEarningsLtm
priceEarningsFy1
priceEarningsFy2

Viewers

id
viewerId
viewId
type

The 'Accounts' table stores personal data about the users. The 'Football Fields', 'Valuations' and 'Valuation Comps' are tables which contain the necessary information for a user to visualize their valuations in the football field chart. The 'Targets' table contains financial information about the private companies users introduce manually. Additionally, Workshop Finance users will have the ability of sending and sharing football fields and valuations to other users. If a football field owner sends their football field to another user, this one will be able to edit it as if it belonged to them. The football field will automatically be added to their list of football fields. However, if an owner shares their football field with a viewer, the viewer will only be able to see (not edit) it. Therefore, we will have a table of 'Viewers'. This way, a user whose ID appears in the table 'Viewers' under the column 'viewerId', will not be able to edit the football field/valuation with ID 'viewId'. The variable 'type' will determine if the object the user is not able to edit is a football field or a valuation.

4 FIRST SEMESTER PROGRESS

The first semester was centered around the planning and design of the application. The team had to make design choices catering to the functionality of a financial application. This semester began with fulfilling many of these design choices, including adhering to the client's design vision and the server side of the application.

Chronologically, the team's first few meetings with the client were centered around absorbing the task at hand and understanding the minutest of requirements of the application. The team discussed what financial functions the application must perform, the models or calculations necessary to achieve specified functions, and how to display the output of the financial functions or calculations performed. A few other questions the team needed to answer were: What are the features and functions of the application? How should the output of these functions be displayed?

An important aspect of the team's first semester progress was logistical planning. The team planned their goals for every sprint over the next few months and set individual deliverables. The team discussed important bi-weekly goals needed to achieve until the due date of the final product, with the exception of some flexible goals.

The team also discussed manpower management, such as dividing work amongst team members on the basis of prior experience and knowledge. Each member was assigned work with some degree of familiarity. Based on strategic planning the team split their workforce into two groups: the frontend developers and the backend developers. Lastly, the team discussed best practices of agile development and sprint planning.

Apart from spending a significant amount of time on design choice, planning, and understanding the application, the team accomplished a significant amount in terms of programming the application. The specifics of the backend and frontend development progress is as follows:

4.1 Frontend

The approach to developing the frontend has changed drastically from the initial stages of development through trial and error this semester—, for example, initially using a total of eighteen different screens with ReactNative's stack navigator and imitating the client's basic wireframe designs from scratch, to now only using a total of seven screens and generating ReactNative code using a plug-in extension on Figma based on the client's updated wireframes.

Over the course of the semester, the frontend team was able to create a basic app framework, consisting of several different app screens, including: a sign in/sign up page, coverage page, a football field page, and an about page. Additionally, the front end team was able to set up text input for the user interface. Utilizing the text input feature, the frontend team was able to set up a sign in/sign up process for the mobile application. The frontend team also gained intimate knowledge about the interworking of ReactNative as they experimented with instantiating different methods as functions or class objects, discovered limitations of instantiating hooks, and attempted to fetch data from a JSON file. Ultimately, the frontend team laid the groundwork and testing capabilities for a functional app to work on both Android and iOS systems, testing the code on both a XCode iOS emulator and an AndroidStudios emulator.

Given the team's inexperience in frontend development, the client was able to connect them with an iOS/Android developer who provided helpful pointers on using ReactNative, given it was not their specialty language. The knowledge gained during this semester will allow the team to work on the frontend more seamlessly over the break and during next semester.

4.2 Backend

The team has successfully designed the databases for the application. After discussion and thought given to the functional capabilities and requirements, the team was able to create a database design for the application, with a total of six SQL data tables ranging from user profiles to company data. After designing the database, the team was able to successfully set it up in IEX Cloud. The setup required specification of each data type in each data table. Lastly, the team was able to use an IEX Cloud API to modify the datatables.

For the computation engine, the team made a design choice of employing Python along with its Numpy library. The team decided to run the computation engine solely on the backend. A significant progress was made on programming the computation engine. The team was able to create a script which would take three inputs: the ticker symbol of a target company, the ticker symbols of all the companies in the basket of comparable companies, and lastly a valuation method. Given these inputs the script was successfully able to return a valuation for the target company.

The team was also able to create sign in functionality for the application. The "sign in script" would require the user to input a username and a password. It would then be able to validate if a user exists in the app's database. Lastly, it would cross check the password against the app's database after the user is validated. This process was made using IEX Cloud's database API features.

The team was successfully able to complete sign up functionality on the backend, and to facilitate account creation for new users of the application. The script was able to take a user's first name, last name, email, and password to create an account, using IEX Cloud's API features.

5 Technical Plan

To complete the overall application, the team has developed a technical plan and schedule consisting of the following eight stages. These stages are also visualized in the Gantt chart in the appendix (Fig 8).

5.1 Phase I: Core Individual Feature Completion 12/10/2022 - 12/24/2022

Task 1. Integrating File Saving

The frontend of the app needs a system to save football field diagrams to their account, to later be reopened. This required file saving must be local data stored in the app. *Lead*: Prithika, *Assisting*: Zen and Ignacio.

Task 2: User Login

The team needs to set up a google-authentication for the app, as well as internal registration and sign-in systems. This involves integrating a text input and display system for the user interface, and appropriate prompts when unmatching data is received. *Lead*: Zen, *Assisting*: Suyash.

Task 3: User Data Authentication and Fetching

The backend must be able to verify that a username entered is a valid username with login credentials, and must be able to call up associated user data. *Lead*: Suyash, *Assisting*: Ignacio.

5.2 Phase II: Full Front & Backend Integration 12/25/2022 - 1/2/2023

Task 4: API Integration

Integrating the IEX Cloud data and API into the frontend of the app for username authentication (taking in text input from log-in), and displaying verified results. *Lead*: Suyash, *Assisting*: Zen.

Task 5: Calculation Engine Integration

The resultant data of the financial calculations must be displayed on the app. This involves integrating the calculation engine to feed its output into the football field data display system. *Lead*: Prithika, *Assisting*: Ignacio.

5.3 Phase III: Recommendation Engine 1/3/2023 - 2/10/2023

Task 6: Researching Clustering Algorithms

The overall recommendation engine will suggest "comparison" (COMPs) and other evaluation techniques based on a user's previous preferences and market data for the company. To do so, we must research different ways of clustering data points and other machine learning techniques to group data for recommendation, and then pick top 2 algorithms to implement for further testing. *Lead*: Suyash, *Assisting*: Prithika.

Task 7: Implementing Algo 1

This is the implementation of the python code for the first machine learning algorithm to recommend specific evaluations/COMPs for the users. *Lead*: Suyash, *Assisting*: Ignacio.

Task 8: Implementing Algo 2

This is the implementation of the python code for the second machine learning algorithm to recommend specific evaluations/COMPs for the users. *Lead*: Prithika, *Assisting*: Ignacio.

Task 9: Testing Algorithm Accuracy

To determine which algorithm is better suited for our app, a series of tests and metrics need to be designed, tested, and measured to compare both algorithms. These quantified test results allow for best algorithm selection. *Lead*: Ignacio, *Assisting*: Suyash.

Task 10: Integrating Algorithm into Recommendation Engine

A recommendation engine needs to be created with the selected algorithm. *Lead*: Suyash, *Assisting*: Ignacio.

Task 11: Displaying Recommendations on UI

The user interface needs to be updated to reflect the ability to suggest/recommend valuation components as provided by the recommendation engine, and finally integrated with the recommendation engine. *Lead*: Zen, *Assisting*: Prithika.

5.4 Phase IV: Finalization and Optimization 2/11/2023 - 2/28/2023

Task 12: Backend Search and Fetching Optimization

This is a finishing touch of further optimizing the API calls and recommendation engines to be as fast and efficient as possible and adhere to industry standards. *Lead*: Ignacio, *Assisting*: Suyash.

Task 13: UI Beautification

This task involves updating graphics, text, color and other aspects to look polished and seamless. *Lead*: Zen, *Assisting*: Prithika.

Task 14: Smoothing Out UX

This involves testing the user experience making edits to create a more intuitive and seamless user experience. *Lead*: Prithika, *Assisting*: Zen.

Task 15: File Sharing and Exportation

This involves exporting the football field result as a pdf or picture format (jpg/png) file that can be shared via email or text. And this involves allowing other users of the app to access a current football field, which would mean one needs to upload football field data to the app's database under the user's account, and then share accessibility with that information with another user. Frontend needs to be updated to display this capability with appropriate buttons and display. *Lead*: Ignacio, *Assisting*: Zen.

5.5 Phase V: Functional Testing and Debugging Buffer 3/1/2023 - 3/31/2023

Task 16: Debugging Buffer

This is a period of time that acts as a time buffer in case any issues arise in the previous processes. If there is an "emergency task" of a problem arising that cannot be solved within the allotted time for previous phases. *General Lead*: Team Lead of the Period, *Problem Lead*: Whoever's area of expertise this problem falls under, in accordance to the Team Contract, *Assisting*: Appropriate other team members.

Task 17: Functional Testing

This is a test plan and execution for testing each aspect of the application, including execution on both iOS and Android, testing UI error-catching, backend data fetching, and recommendation engine capabilities. *Lead*: Ignacio for backend and Zen for frontend, *Assisting*: All team members.

Task 18: Server Hosting Research

This is a period of time dedicated to researching hosting support with Amazon Web Services (AWS) and developing an integration plan. This may involve discussions with our client about what kind of support they would prefer. *Lead*: Ignacio, *Assisting*: Suyash.

Task 19: Host Integration

Conversion of local servers to be run on AWS. Updating local app connections to work with new AWS links. *Lead*: Suyash, *Assisting*: Ignacio.

Task 20: Finalization and Sign-Off

Ensuring all final edits of the app are made and any fixes are finalized. All members approve of the final product. *Lead*: Team Lead of the Period, *Assisting*: All team members.

5.6 Phase VI: App Hosting/Customer Integration 4/1/2023- 4/30/2023

Task 21: Handing Final Product Over to Client

Presenting the final product to the client, receiving feedback, and making subsequent minor edits. *Lead*: Team Lead of the Period, *Assisting*: All team members.

Task 22: Debugging Buffer

This is a few days that act as a time buffer in case any issues arise in the client integration process. If there is an "emergency task" of a problem arising that cannot be solved within the allotted time. *General Lead*: Team Lead of the Period, *Problem Lead*: Whoever's area of expertise the problem falls under, in accordance to the Team Contract, *Assisting*: Appropriate other team members.

5.7 Phase VII: Presentation and Report Prep 5/1/2023 - 5/4/2023

Task 23: Finalize Reports

This is time to prepare our final reports and other technical documents before ECE Day. *Lead*: Team Lead of the Period, *Assisting*: All team members.

Task 24: Finalize Presentation

This is time to prepare our presentation and pitch for ECE Day. Powerpoint presentation must be finished, and the QR code to download and display the app must be available and working. *Lead*: Team Lead of the Period, *Assisting*: All team members.

5.8 Phase VIII: ECE Day 5/5/2023

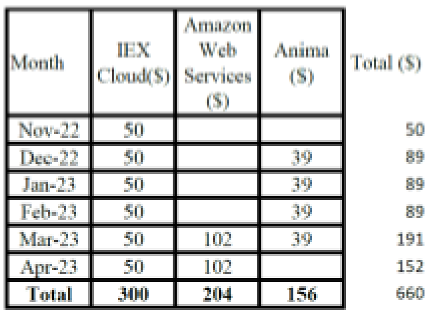
Task 25: Present Final Product

This is the final presentation and display of our work and product to the general ECE staff and judges in a symposium-style presentation. The team will all be answering questions about the app and working together to engage the audience. *Lead*: Team Lead of the Period, *Assisting*: All team members.

6 Budget Estimate

The following table states our estimated budget starting November 2022 to April 2023.

Fig 5: Cost table



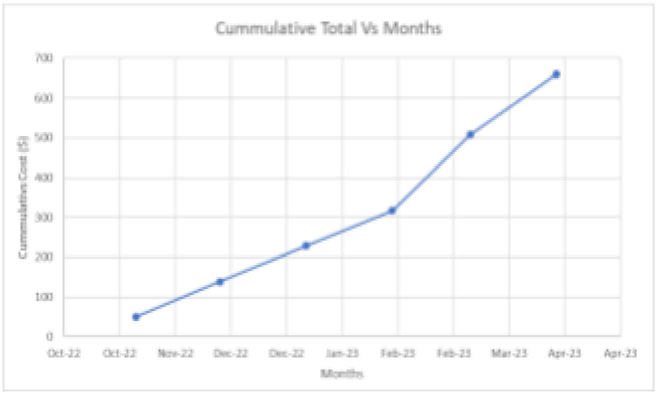
Month	IEX Cloud($)	Amazon Web Services ($)	Anima ($)	Total ($)
Nov-22	50			50
Dec-22	50		39	89
Jan-23	50		39	89
Feb-23	50		39	89
Mar-23	50	102	39	191
Apr-23	50	102		152
Total	**300**	**204**	**156**	660

IEX Cloud: $50 for IEX cloud is a subscription cost. IEX cloud is our financial data API provider as well as a cloud storage provider. IEX cloud gives real time access to market data as well as customized database storage on the cloud.

Anima: $39 for Anima is a subscription cost. Service to convert Figma wireframes to ReactNative code. Figma wireframes are used for UI/UX designing. Anima converts these UX designs to react native code which facilitates seamless frontend development.

AWS (Amazon Web Services): $102 subscription cost for a high-volume production app. Amazon web services provides hosting services for mobile application applications. Hosting an application on a cloud platform enables millions of users to be able to access the application worldwide.

Fig 6: Graph of total spending over time



7 Attachments

7.1 Appendix 1 – Engineering Requirements Chart

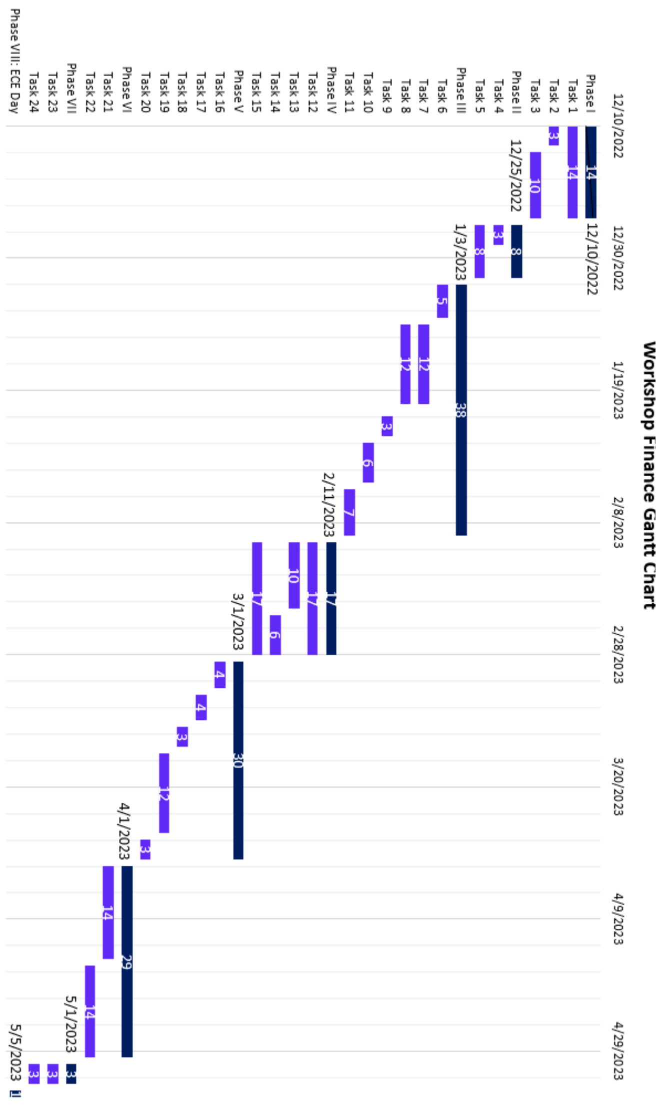
Fig 7: Feature, Objectives, Functions, Constraints chart of engineering requirements

Feature	Objectives	Functions	Constraints
User Experience	For the app to run smoothly and reliably produce visuals and results.	Efficiently generate view of valuations with quick data fetching facilitated by an optimized backend. Ideal page load time of <3 seconds [5].	Team's limited knowledge on UX development.
User Interface	Enable users to control the application and view the football field graphics.	Include basic navigation features and a holistic range of levers for the user to customize their valuations; functional scope of application to have ~70 elements e.g. "Add New Valuation" and "Change Target".	Team's limited knowledge on UI development.
Local Data Storage	To preserve user search and valuation history to be retrieved when the user returns to application after use.	Store user data locally on the device to provide users with an instant startup experience. Productivity apps on iOS typically take up about ~400MB of space e.g. Excel [6].	Limited to device storage application is operating on.
Computation Engine	Reduce workload on the frontend to reduce storage load on the device.	Define the frontend payload to perform actions such as users changing their valuation criteria.	Running several calculations can be computationally intensive.
API Integration	To minimize use of local storage and allow users to efficiently access a wide range of updated market data for valuations.	Implement API optimization techniques such as caching and ensuring sufficient cloud infrastructure.	Limited by the abilities of the API used and the amount of cloud storage available.
Identity Service	To have safe, synchronous multi-user access for customers.	Authenticate user profiles for each user logging into the application and keep user information confidential.	Team's limited knowledge on cybersecurity; may encounter data leaks.
Market Database	To reliably and efficiently fetch data from the market database (i.e. based on the date the user chooses).	Employ a range of third-party databases to be accessed through APIs with a daily refresh rate.	Limited by financial resources and the magnitude of data needed.
Recommendation Engine*	To provide a personalized user experience and buffer as a user searches for a company to value.	Using machine learning models such as clustering on a range of user inputs over time.	Limited by the capabilities of the ML model chosen to deploy.
Cloud-based Hosting	Scalable access to computing resources for the functioning of the application.	Employing advanced, reliable cloud services such as IEX.	Limited by financial resources.

Note: Project stretch goals are marked with an asterisk (*).

7.2 Appendix 2 – Gantt Chart

Fig 8: Detailed task Gantt Chart for project overview


Workshop Finance Gantt Chart
Phase I
Task 1
Task 2
Task 3
Phase II
Task 4
Task 5
Phase III
Task 6
Task 7
Task 8
Task 9
Task 10
Task 11
Phase IV
Task 12
Task 13
Task 14
Task 15
Phase V
Task 16
Task 17
Task 18
Task 19
Task 20
Phase VI
Task 21
Task 22
Phase VII
Task 23
Task 24
Phase VIII: ECE Day
12/10/2022
12/30/2022
1/19/2023
2/8/2023
2/28/2023
3/20/2023
4/9/2023
4/29/2023
12/10/2022
12/25/2022
1/3/2023
2/11/2023
3/1/2023
4/1/2023
5/1/2023
5/5/2023