Logo%20Main%20200

**Boston University**

**Electrical & Computer Engineering**

**EC464 Capstone Senior Design Project**

User's Manual

The Art of Valuation

Submitted to

Workshop Finance

brendan@workshopfinance.com

by

Team 30

DevUp

Team Members

Prithika Ganesh pvganesh@bu.edu

Ignacio Nunez Gomez inunezg@bu.edu

Suyash Bhatia suyashb@bu.edu

Zen Mae Lee leezm@bu.edu

Submitted: April 17th, 2023

#### Table of Contents

[**Executive Summary 2**](#_30j0zll)

[**1 Introduction 1**](#_1fob9te)

[**2 System Overview and Installation 2**](#_3znysh7)

[2.1 Overview Block Diagram 2](#_2et92p0)

[2.2 User Interface 3](#_tyjcwt)

[2.3 Physical Description 4](#_3dy6vkm)

[2.4 Installation, Setup, and Support 4](#_1t3h5sf)

[**3 Operation of the Project 6**](#_4d34og8)

[3.1 Operating Mode 1: Normal Operation 6](#_2s8eyo1)

[3.2 Operating Mode 2: Abnormal Operations](#_17dp8vu) 8

[3.3 Safety Issues](#_3rdcrjn) 9

[**4 Technical Background**](#_26in1rg) **10**

[**5 Relevant Engineering Standards**](#_lnxbz9) **11**

[**6 Cost Breakdown 1**](#_35nkun2)**3**

[**7 Appendices 1**](#_1ksv4uv)**4**

[7.1 Appendix A - Specifications 1](#_44sinio)4

[7.2 Appendix B – Team Information 1](#_2jxsxqh)5

# Executive Summary

Workshop Finance is a deal idea generation tool, allowing users to quickly build valuations. The application enables a user to evaluate a given company based on a variety of metrics, (e.g. similar companies, trading multiples, time periods) and arrive at a valuation for a given company. The app then performs the necessary calculations and presents results through a unique visualization called a ***Football Field***, a graphic which typically appears in investment banking pitch books. The mobile application allows users to create, refine, and view this football field directly through the app. Workshop Finance aims to enable deal idea generation, client communication (i.e. pitching to clients, initializing cases), and deal execution across the industry. It will become the primary medium through which users “tell the story” of a company’s valuation—the crux of any corporate finance pitch. This first-of-its-kind application will power the ***art of valuation*** through a combination of strict corporate finance practices and innovative user experience.

# Introduction

Generating relative financial valuation, i.e. valuing a company relative to other companies, is a laborious and time consuming task. While there are already tools and methods to value companies using a football field chart, none of them are fast, dynamic, and responsive. Such existing tools rarely evaluate both private and public companies. Financial corporations typically require valuations of both types of companies.

The core goal of Workshop Finance is to create a mobile application capable of generating a comprehensive corporate finance valuation. Workshop Finance aims to make football field generation easy and 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 valuations in a football field within minutes enables workers in the finance industry to make informed business decisions faster with baseline metrics.

The user begins by selecting 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. If the user decides to value a private company, the user will need to manually input its fundamental financial data. The app produces valuation models through both relative and intrinsic value methodology. The user will also be able to adjust the company’s value calculations and visual output by adjusting various financial variables—such as by adding companies from the specific industry to create a financial comparison (COMP) or by editing the index’s multiplier.

Workshop Finance holds dynamic and fast valuations at the top of its offerings. It will be the first of its kind application, determining valuations of public and private companies at an user’s fingertips.

# System Overview and Installation

## Overview Block Diagram

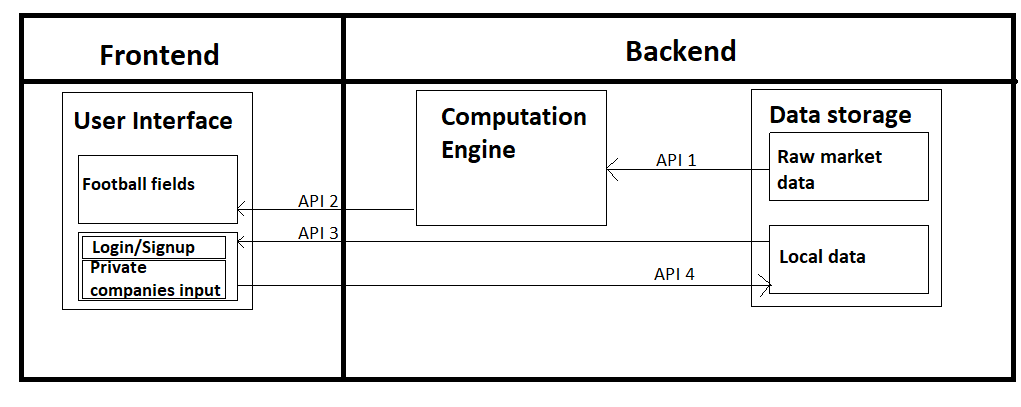


Figure 2.1: Overall software block diagram illustrating flow of data between Workshop Finance’s backend and frontend.

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 company’s 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.

## User Interface

The user interface is divided into the six main screens below. After the user creates an account and signs in, the ReactNative Stack Navigator routes the user to the home screen, where they can select one of three options: most recent Football Field, Coverage List, or User Profile, all of which are illustrated in Figure 2.3.

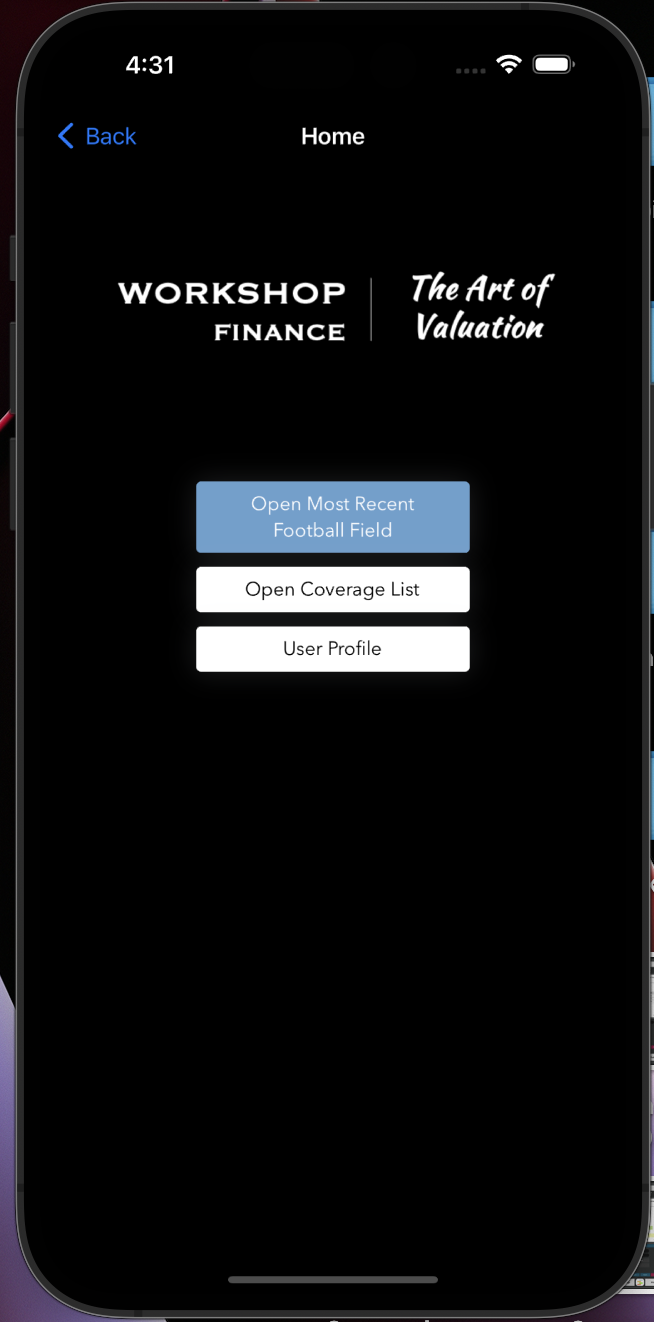
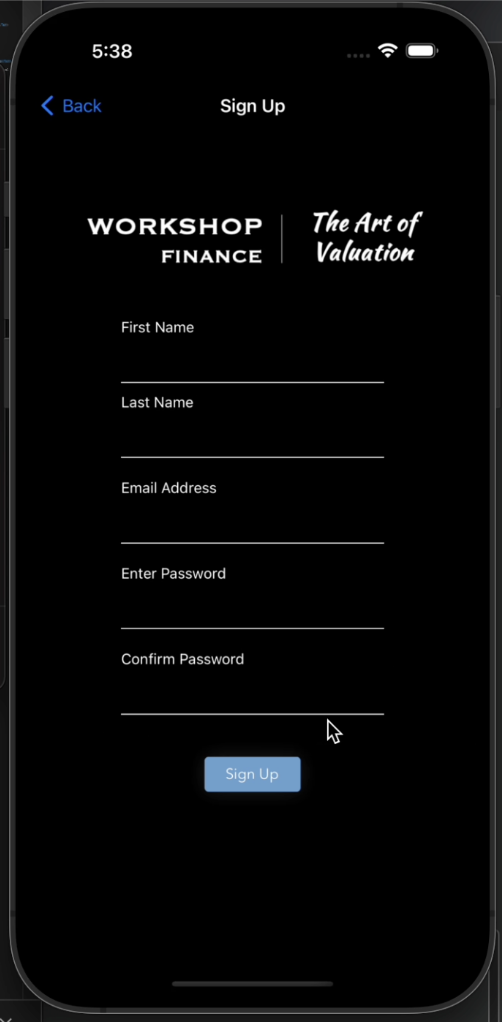
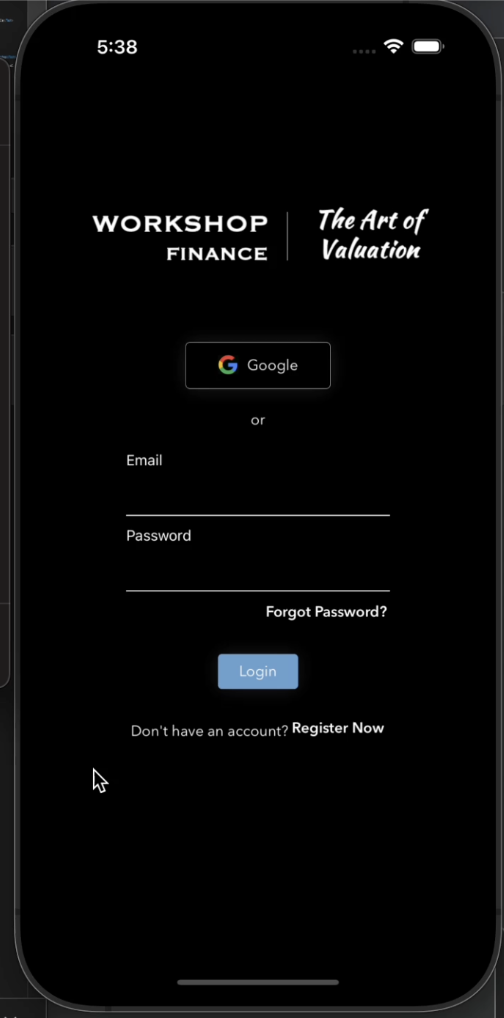


Figure 2.2: Screenshots of Sign In, Sign Up, and Home Screen screens.

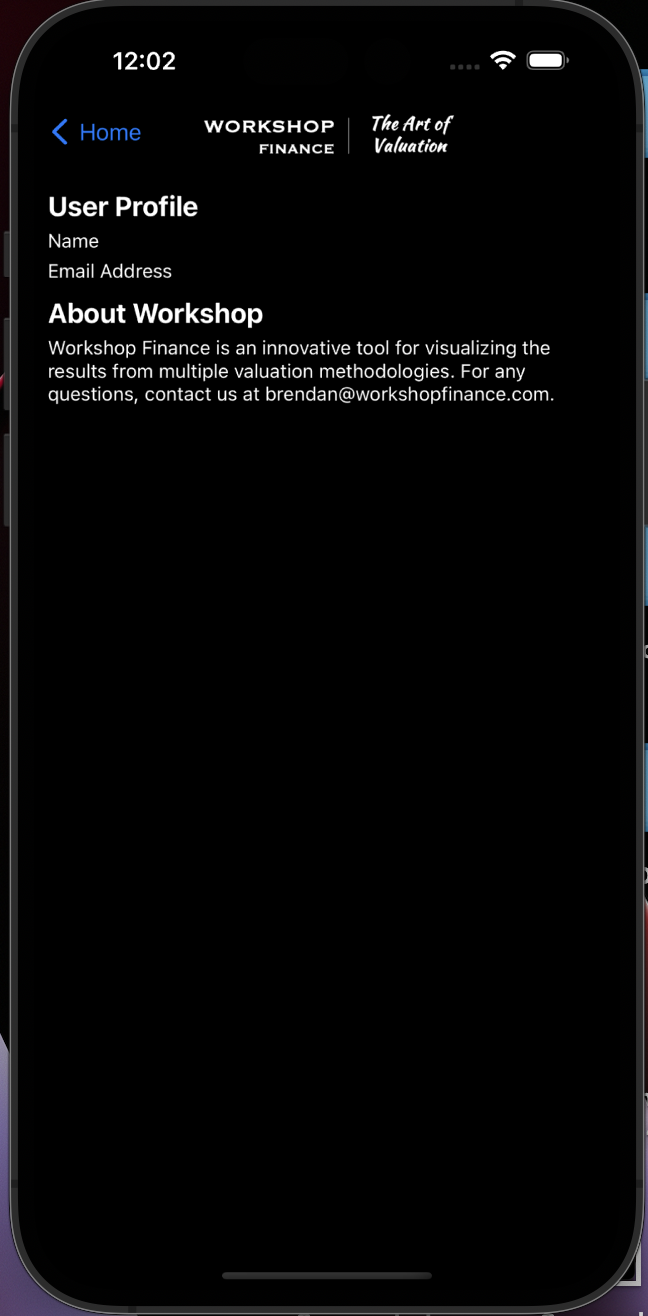
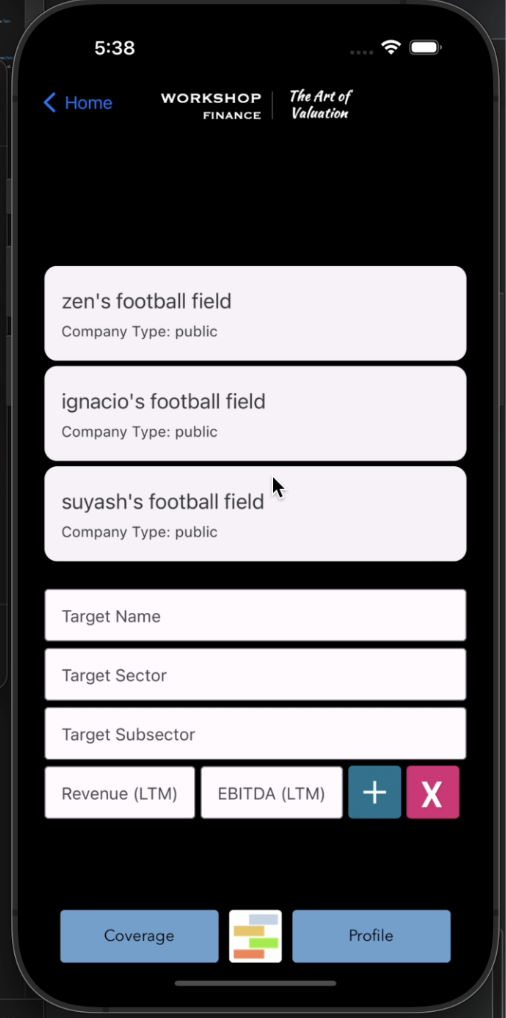
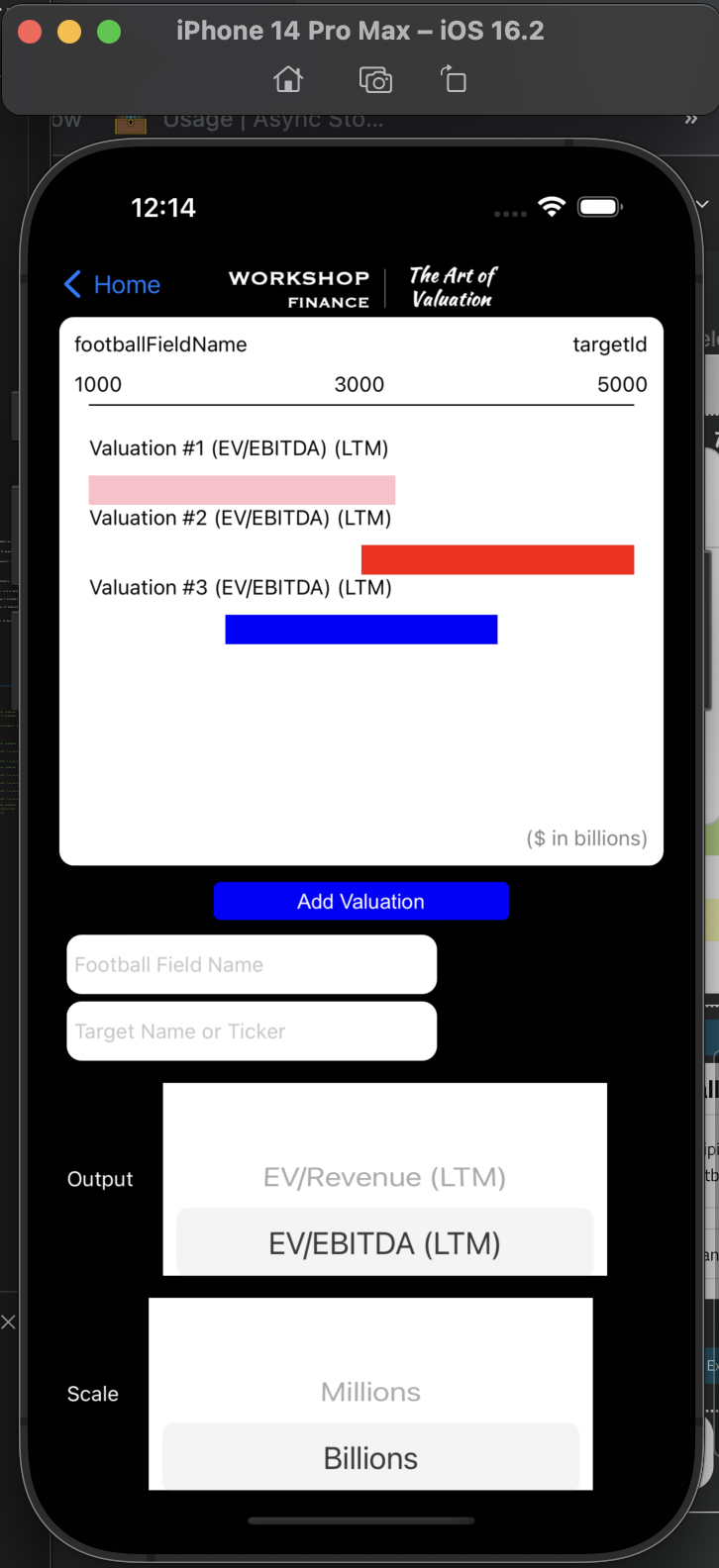


Figure 2.3: Screenshots of Football Field, Coverage, and Profile screens.

## Physical Description

Since our application is exclusively for iOS, the hardware used to run Workshop Finance is the user’s iPhone or a device, such as a laptop, where an iPhone emulator such as XCode can be run. A device, such as a computer or laptop, is also required to run the backend local server for our application.

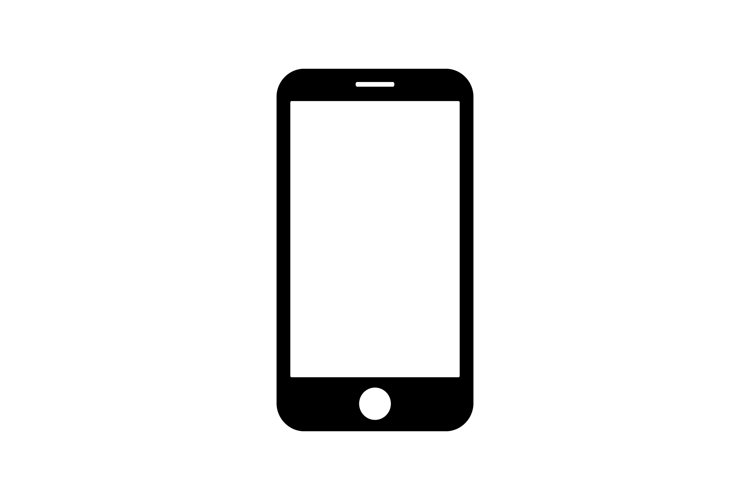


Figure 2.4: Required hardware includes computer to run backend server and iPhone for frontend interface.

## Installation, Setup, and Support

Users are able to use our application in one of two ways: downloading Expo Go and scanning the Metro Bundler QR code to run the application locally on their mobile phones, or pulling the code from GitHub and using npm combined with an iPhone emulator. For the former method, the user downloads the Expo Go application from the App Store and scans the QR code generated from Metro Bundler below via Expo Go.

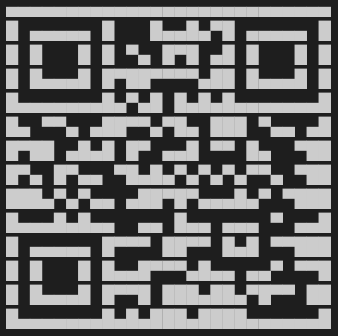


Figure 2.5: QR Code generated from Metro Bundler.

For the latter method, the user downloads the code from GitHub in whichever way is preferable i.e. downloading as a ZIP file or cloning the repository into their local repository. The user would then do the following to run the application on an iPhone emulator on their device:

1. Navigate into the *frontend* directory
2. Run ‘npm start’ → ’i’ for iOS
3. Navigate into the *backend* directory
4. Run the routes.py file (i.e. python routes.py) in a new terminal to initiate running the backend

# Operation of the Project

## Operating Mode 1: Normal Operation

Upon app installation (run via QR Code on ExpoGo or run using npm on emulator), the user will open the app to find a login screen. If the user has used the app before and has an account with Workshop Finance, they can login with their credentials by filling in their email and password and clicking the “Login” button. If this is the first time a user has ever used the app, they must create a new account by pressing the “Register now” button.

If a user is creating an account for the first time, they will be taken to the “SignUp” page, where they will be instructed to enter their First Name, Last Name, email address, as well as create and confirm a password. Their password must be between 8-16 characters, contain one lowercase letter, one uppercase letter, and one number.

After signing into the app, a user will be taken to the home page, where they can navigate to either the Coverage, Football Field, or User Profile pages.

The Coverage page displays a list of every football field graphic created by the user, as listed by the user-defined football field’s name. A user can open these pre-existing football fields by simply clicking on them. The Coverage page also contains visual prompts for a user to create new “targets”' and subsequent football fields through simple entry prompts. Through this, the user will be able to create a new target or select an existing target for their subsequent football field. To create a new target, a user can select a public company or manually enter the data for a private company. Once a target is selected, a user will be sent to the Football Field page.



The “Football Field” page’s primary function is to display the currently selected football field graph and allow users to edit said football field. At the Football Field page, the user will be able to add various valuations to the Football Field by clicking the “Add Valuation” button. The user must then select the valuation type, the metric, the stat, the spread and the desired color. In doing so, and finalizing the selection by hitting the + button, the app generates a bar for the football field graph. These bars can then be re-edited by clicking on the desired bar.



Lastly, the User Profile page displays the user’s account information, ie Name and email address, as well as a blurb about Workshop Finance.

The user can use the top tabs to navigate between these pages.

## Operating Mode 2: Abnormal Operations

The first anticipated abnormal state is if the user is trying to sign in with an invalid password or email address. If the user enters an email address and password combination that is not saved to our “user” database, an alert will pop up indicating that the username or password are invalid or not found in our database. This will prompt the user to re-enter their username and password.

The next abnormal use accounted for in the Sign In/Sign Up page is for if the user forgets their account’s password. If the user forgets their password, they can click on the “Forgot Password” button. There, the user will be prompted to enter their email address. If the email address entered is valid and exists in our database, a password will be emailed to that email address.

The next anticipated abnormal state is in the SignUp page, where a user registers an account for the first time. Here, if the user enters an email address without a “@” or a domain (ex, .com or .net), they will be prompted with an “invalid email address message”. Similarly, if the user enters an invalid password, meaning that the 2 passwords do not match or if they enter a password that is not 8-16 characters long, does not contain at least one lowercase letter, one uppercase or one number, they will be prompted by a series of alerts indicating which incorrect problem they face. These alerts will be prompted for each invalid entry until the user enters an appropriate email address and corresponding passwords.

The next anticipated abnormal state is when our user attempts to delete any created fixture in the app. On the off chance that the user accidentally clicked the various “delete” buttons in the app, the app prompts the user through an alert, asking the user to confirm the deletion and reminding the user that this action cannot be undone. This applies to objects such as Football Fields or Valutions/Comps. The user gets an alert asking for them to confirm that they want to permanently delete the said object, in case the user accidentally pressed the button before.

Similarly, another abnormal case is when a user is editing a Football Field with active valuations, and attempts to change the Target of this Football Field. In doing so, the entire basis of the current valuations change, which may not be what the user intends. Therefore, to prevent any unintended changes, the user is prompted with an alert, asking if the user would like to save this target change as a new football field or continue this edit on the existing football field. If the user chooses to start this as a new football field, the metrics are cloned and a new football field is created with this new target value. If the user chooses to replace the current football field, the target is simply changed. Or a user may choose to cancel the action, upon which no changes are made to the existing football field.

We will also deal with the abnormal state that the user enters a ticker or company name that does not exist into the Target or Valuation pages. Here an alert with an error message that says “No Public Companies for Ticker Symbol or Company Name Found”.

## Safety Issues

There are very limited safety issues with our app. The main concern with our app are the implications of security. If our data is not securely stored, a hacker may infiltrate our system. In doing so, they would have access to our users’ private data (their names, email addresses, and passwords) as well as any private company information they have stored on the app, such as their company’s earnings and revenue values. These issues have attempted to be addressed through practicing certain engineering standards, which will be further expanded upon in section 5. Additionally, our app is the starting product for the Workshop Finance start up, and therefore lacks a greater dependent ecosystem of subsequent products and dependencies (ie, no other products are meant to solely rely on our app). And as the Workshop Finance app is geared towards individual users, as opposed to direct integration into a larger commercial atmosphere, this risk is also limited from the end users’ standpoint. Therefore, if the app fails or completely falls out of use, the worst consequences is that the end user will not be able to access their readily made football fields, and will instead have to go back to manually creating these graphs through spreadsheets until the issue is resolved.

# Technical Background

Our team’s technical approach for Workshop Finance was to develop an iOS mobile application. The core components of the application are a cloud-based hosting software, APIs, and to develop a fully functional frontend, using ReactNative as the primary framework, 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.

The technical background for the frontend is fundamentally ReactNative components, such as buttons, functions with prop arguments pass through to implement hide/show logic, and the StackNavigator to route the user through different screens. One key aspect of using ReactNative is importing the appropriate libraries for each component. Additionally, the style components need to be dynamically scaled using the Dimensions.get(‘window).height and Dimensions.get(‘window).width to be compatible with different iPhone models, such as the iPhone 14 and iPhone SE.

Our backend involves developing a REST API using Python and Flask. The API will retrieve stock market data from IEX Cloud, a financial data provider that offers real-time and historical data for stocks, ETFs, mutual funds, and more. Flask is a popular Python web framework that is used for building web applications and APIs. It is lightweight, flexible, and easy to use, making it an ideal choice for small to medium-sized projects. Flask also supports various extensions, such as Flask-RESTful, which makes it easy to build RESTful APIs.

IEX Cloud provides financial data through an API, allowing developers to access a wide range of financial data, including real-time and historical market data, news, and more. The data is stored on IEX Cloud's servers and can be accessed through API calls. In our project, we will use Python to retrieve stock market data, such as the latest stock prices, historical stock prices, and market statistics. We will then expose this data through a REST API, allowing users to access the data in a standard and structured way.

REST (Representational State Transfer) is a popular architectural style for building APIs. REST APIs are designed to be simple, scalable, and easy to consume. They use HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources and use URLs to identify and locate resources. Flask-RESTful makes it easy to build RESTful APIs in Flask by providing a set of classes and methods for defining resources, handling requests, and returning responses.

# Relevant Engineering Standards

Over the course of development, testing, and finalization of our project, we have followed good coding standards, including: compliance with industry standards (e.g., ISO), consistent code quality—no matter who writes the code, software security from the start, and reduced development costs and accelerated time to market. The methodology in following ISO is described below in further detail. Our team maintained consistent code quality by maintaining a neat and readable standard of code and providing comments explaining major functions wherever necessary. We also maintained security by protecting our application keys, maintaining a private github, maintaining a secure login for our database system, and creating a secure login for our app’s users.

Our project is compliant with the ISO 12207 software life cycle processes and Agile development process. Agile development adheres to periodic stages of iterative development. Our project has a well-documented history or iterative stage-centric development. Similarly, the ISO 12207 standard of development dictates that a project's methods, activities, and tasks should be selected based upon a project’s described scope. Additionally, a record of who is accountable for each process, activity, or task should be kept. To comply with ISO 12207, an organization must identify what procedures, actions, and tasks are necessary to be performed in order to meet the standard's minimal requirements. Our group has maintained a list of requirements based on deadlines and priority levels.

Our project is also compliant with ISO 29119 Software Testing standards. The ISO 29119 Software Testing focuses on concepts and definitions, test processes, test documentation, and test techniques. ISO 29119 “concepts and definitions” section facilitates the use of the other parts of the standard by introducing the vocabulary on which the standard is built and provides examples of its application in practice. This provides definitions, a description of the concepts of software testing, and ways to apply these definitions and concepts to the other parts of the standard. Our project does that by defining key financial terms like EBITDA, Revenue, and Football Field, and technical terms like API, prior to testing. The Test Processes section of ISO 29119 defines a generic test process model for software testing that defines the software testing processes at the organizational level, test management (project) level, and dynamic test process levels. The processes defined in this standard can be used in conjunction with different software development lifecycle models. Meanwhile, the third section of the ISO 29119 defined how the above tests and their subsequent results should be documented. The fourth section of the ISO 29119, Test techniques, provides standard definitions of software test design techniques (also known as test case design techniques) and corresponding coverage measures that can be used during the test design and implementation processes defined in Part 2. The standard's test design techniques are categorized into three main categories: specification, structure, and experience-based test design techniques.

Our project also attempts to follow ISO 27001 Compliance for Information Security. ISO 27001 compliance first requires the identification, classification, and labeling of significant information assets. It is necessary to define the user groups who have access to different pieces of information. As dictated by ISO 27001, mechanisms for data access and protection must be developed. Our project has done that by defining a users database and using a user’s email and password combination as a key to other user data, such as privately entered company metrics and private user-generated Football Fields.

# 

# Cost Breakdown

| Project Costs for Production of Beta Version (Next Unit after Prototype) | | | | |
| --- | --- | --- | --- | --- |
| Item | Unit cost per month | Total Number of Months used | Description | Overall Cost Per Month |
| 1 | $50 | 6 | IEXCloud Databse | $300 |
| Total Cost (Beta Version) | | | | $300 |

**IEXCloud:** $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.

Our client’s plan for deployment follows a peer-to-peer model where individuals in his network will use the application. Our client does not currently have any plans to sell the application or deploy it on Amazon web services for use by the general public. Hence, no money will be spent on application deployment. However, IEXCloud will continue to charge an overall subscription fee of $50 per month to maintain the database.

# Appendices

## Appendix A - Specifications

| **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. | 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. | 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. |
| 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. |

## Appendix B – Team Information

| **Name** | **Degree** | **Graduation Year** |
| --- | --- | --- |
| Ignacio Nunez Gomez | Computer Engineering  Business Administration | 2023 |
| Suyash Bhatia | Computer Engineering,  Machine Learning Concentration | 2023 |
| Prithika Ganesh | Computer Engineering,  Machine Learning Concentration  Innovation & Entrepreneurship Minor | 2023 |
| Zen Mae Lee | Computer Engineering | 2024 |