**Memo**

To: Professor Pisano

From: Prithika Ganesh [pvganesh@bu.edu](mailto:pvganesh@bu.edu), Ignacio Nunez Gomez [inunezg@bu.edu](mailto:inunezg@bu.edu), Suyash Bhatia [suyashb@bu.edu](mailto:suyashb@bu.edu), Zen Mae Lee leezm@bu.edu

Team: Team 30, DevUp

Date: 03/17/23

Subject: The Art of Valuation: Final Testing Report

1. **Summary of Equipment and Setup**

1.1 Required Materials

1.2 Set Up

1.3 Testing Procedure

**2.0 Measurements Taken**

2.1 Measurable Criteria

* 1. Measurements Taken Table Results

**3.0 Conclusion**

**1. Summary of Equipment and Setup**

**1.1 Required Materials**

Software:

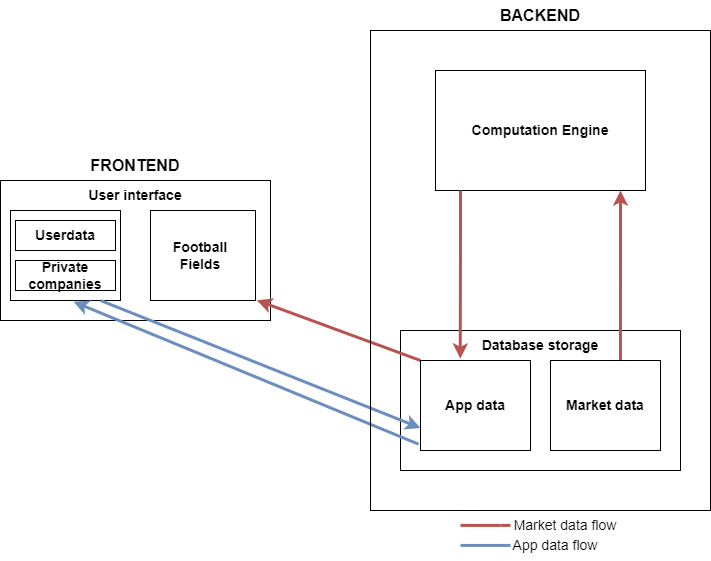
* Emulators: XCode
* Languages: ReactNative and Python 3
* Frameworks: Flask (Python)
* Data APIs and API keys: IEX Cloud Apperate

Hardware:

* Personal computing device i.e. laptop, PC etc.

**1.2 Set Up**

The setup requires running the frontend and backend code on an iOS emulator. The Visual Studio Code IDE will be used as our main platform.



*Figure 1: Illustration of Data Flow from Backend to Frontend*

**1.3 Testing Procedure**

1. For the frontend, open Visual Studios and navigate into the *workshopfinance* directory.
2. Change directory to the frontend (i.e. cd frontend) and run the XCode emulator by typing ‘npm start’ → ‘i’ for iOS.
3. Change directory to the backend (i.e. cd backend) and run the routes.py file (i.e. python routes.py) in a new terminal to initiate running the backend.
4. Open the IEX Cloud database to demonstrate that our database and tables are designed and functional.
5. App will be open to the Sign In/Sign Up page. Create an account for the user by clicking “Register Now”. Fill in first name, last name, email address, password, and confirm password. Hit submit to create account.
6. Navigate to the Sign In page. Fill out just created account information. Hit submit.
7. Upon submission, you will arrive at the home screen with “Coverage” and “Profile” options. Click on “Coverage” to navigate to the “Coverage” page.
8. Create a Public Target.
9. Add a Football Field to that Target.
10. Create and add one valuations to the Football Field.
11. Add individual Comps to this valuation. Select different companies and adjust multiples to create different Comps. (Refresh login to see updates.)
12. Show that they have been added to the database.
13. Repeat steps 11 and 12 as many times as desired.
14. Show the latest football field to see the complete football field.
15. Navigate to the Coverage page to see a list of Football Fields.
16. Navigate to the Profile page to show user information.

**2. Measurements Taken**

| **Action** | **Category** | **Y/N?** |
| --- | --- | --- |
| Flask working properly (routes.py) running smoothly | Backend | Y |
| Open IEX Cloud database UI to display updates and design | Backend | Y |
| IEX Cloud database functional:  Adding rows to tables using REST API | Backend | Y |
| Football Field name updated automatically in database | Backend | Y |
| Application runs smoothly on iOS emulator | Frontend | Y |
| Successfully creates new user account | Frontend | Y |
| Can successfully sign in with new user account | Frontend | Y |
| Creates a valuation | Frontend | Y |
| Visual bars are adjusted for multiples selected | Frontend | Y |
| Creates a successful “comp” from multiple valuations | Frontend | Y |
| Valuation charts drawn with data from database | Frontend | Y |
| Coverage page lists different Football Fields created | Frontend | Y |
| User profile page shows user information | Frontend | Y |
| **Result** | | 100% |

**3. Conclusion**

Given the team’s progress with integrating the backend and frontend for our second prototype testing, our main goal was to be able to successfully create valuations with comps for the football field and create new targets on the coverage page (where new football fields could subsequently be generated).

For the backend, we were able to fetch data based on tickers keyed in by the user into text input boxes. Once this input was provided to the backend, we were able to produce respective valuations for the data entered by the user. The user was given the liberty to choose the multiple valuation methods. The user was also able to choose the basket of comps by entering the respective ticker symbols for the same.

Apart from the app functionality, we achieved success with smooth functioning of the backend. We successfully served the backend data locally with the help of flask. Lastly, we were able to achieve a significant feat in terms of database work done. We were able to add, delete and update relevant rows in our database as required. This was integral for the functioning of our application as well as successfully achieving the goals set by us.

For the frontend, our emulator ran smoothly during the testing session. The frontend is now able to accommodate adding comps to individual valuations. A request is sent from the frontend to the backend to fetch data based on the inputted comp ticker, which the computation engine will use to calculate a valuation based on the target. Our frontend also is able to successfully manage passwords for new, registering users by ensuring they are between 8-16 characters long and ensuring that emails are entered in a valid format.

The criteria used to measure the final prototype essentially captures the application’s ability to create a basic valuation and create and store football fields which the user can revisit again after. Over the next few weeks leading up to ECE Day, the team will be focusing on brushing up the UI and finalizing the backend to make the application’s functionality smoother.