Big Data Analytics and Tactical Decision Making

A Major Qualifying Project Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the degree of Bachelor of Science.

Submitted by:

Andrew Aberdale: A. Aberdale

Juan Carlos Chávez Guerrero: Juan CCG.

Zhi Hui: [illegible]

Juan Pablo de Lima: [illegible]

Benjamin Sarkis: [illegible]

Rekik Tafesse: [illegible]

Date: January 21, 2017

Project Sponsors
Vestigo Ventures
Cogo Labs

WPI Faculty Advisors
Professor Jon Abraham
Professor Michael Ciaraldi
Professor Reinhold Ludwig
Professor Kevin Sweeney

Abstract

This project focused on utilizing in-house proprietary systems to analyze big data and perform educated predictive decision-making for investment and financial solutions. Using JavaScript, Python, and SQL our team automated the information gathering and data compilation of the target companies selected by Vestigo Ventures. Our team then analyzed the presently available data to make tactical and financial recommendations to the Vestigo Ventures executives. We then ended the project by developing a probabilistic device fingerprinting algorithm for Cogo Labs.

Table of Contents

Acknowledgements

We would like to express our appreciation for the contributions from the following people:

Ian Sheridan for drafting our onboarding documents, and for the time he dedicated to us when we required his assistance.

Mike Nugent for his involvement with both the graduate and undergraduate team, for organizing our paperwork and meetings, and for the time he spent with us to ensure a high level of quality in our projects.

Todd Federman for his online IT help regarding Cogo’s proprietary Python API and server implementation, and for his suggestion to use Flask as our system backbone.

Matt Wiens for the time he spent with us designing our device fingerprinting algorithm.

John Roland for introducing our team to Vestigo’s and Cogo’s operations and leadership, for the history lesson on Vestigo and Cogo, and for his concise explanation of Cogo’s proprietary database managing interface.

Rob Fisher for his help with Cogo’s proprietary database management system (DBMS) , and for his software and SQL expertise regarding many of our projects.

Ken Koldewyn for his help with Docker, server technology, and Linux.

We would like to thank our sponsors, Vestigo Ventures and Cogo Labs and all their employees for working alongside them, and for attaining our personal goals for this MQP.

The WPI graduate team: Alex Shoop, Khasan Dymov, Madhuri Surve, Zoey Chen for giving us clear objectives and what deliverables satisfy these objectives, for the days they spent with us solving technical issues, and for the time they spent working with Professor Abraham on device fingerprinting to review our concept.

Professor Suzanne Mello-Stark for her security expertise. This was instrumental to solving integrity issues regarding our online form.

Professor Kevin Sweeney for making this opportunity possible, for his leadership role, and for his role as an advisor.

Finally, Professors Michael Ciaraldi, Jon Abraham, and Reinhold Ludwig for making this an educational and fulfilling project, for outlining expectations clearly, and for their feedback.

Executive Summary

Two primary companies spearheaded the overall structure of our project. The main company was Vestigo Ventures. They acted as our sponsor and provided the bulk of the layout for our project. Cogo Labs was closely associated with Vestigo Ventures and had us work on an analysis of their databases. These analyses provided the information necessary to make predictive tactical decisions for future investments. In addition to our undergraduate team, there was a WPI graduate team that worked in tandem. Vestigo had four aims for this project, one of which was designed to aid Cogo Labs:

1. To analyze Vestigo's past and current deals and to present "lessons learned" from said analysis
2. To improve Vestigo's investment process
3. To gather business experience for the undergraduate team
4. To give Cogo new ways to analyze their big data to develop new capabilities in B2B signal search.

The WPI graduate team realized these aims had overlapping objectives. They mitigated this issue by separating Vestigo's aims into four subprojects:

1. Venture Decisions Made Over Time
2. Deal Analysis and Industry Engagement
3. Signal Search and Market Landscape Analysis
4. Device Fingerprinting

The graduate report associated with this paper describes the rationale of these delineated goals.

We worked with the graduate team to establish and review the goals of each project. Once established, we assisted the graduate team in defining the solutions and creating action items.

The first project, labeled "Venture Decisions Made Over Time" (hereafter known as "Project 1", or "Data Mining"), addressed the first goal. We broke Project 1 into two tasks. The first task was compiling investment and company information into a "Deal Flow" spreadsheet. This task involved a Deal Flow mailbox where Vestigo forwarded all relevant information on investment opportunities. It warranted manual entry into a small component of Vestigo's data. With the graduate team's help, our Deal Flow spreadsheet took two weeks to finish.

The second task was to research and implement a procedure that automates this process. It made the methods of the first task quicker and more scalable. Our sponsors recommended the creation of an online form at the outset. Through our research we discovered that the React.js framework addressed Vestigo's expectations for the form. Using the React framework we created the front end to this online form and sent entered information to Cogo's database via a Python server. We automated the process using a combination of said server and several SQL queries. The queries executed within Cogo's database. They pulled and recorded "confidence" levels for each potential investment from Cogo's Apollo algorithm. This was intended to help Vestigo make informed executive decisions on whether to invest in an opportunity or not.

We amplified the original functionality of our form by adding security and integrating the system with a customer relationship manager (CRM) platform. About a month into our project, one of our academic advisors raised a security concern. Thus, we revisited the objectives of the second task to implement security on the base iteration of our form. We drafted a STRIDE analysis for our security and brought it to our sponsors as outlined in Appendix H. They approved the plan and asked that we enact it. The plan included encryption, authentication, data validation, and administrator rights. Nearing the end of our development in Project 1, our sponsors asked that we research methods to connect our system into their CRM platform, Close.io. Vestigo regularly used Close.io in their deal-making. We incurred a minimal time cost when we integrated Close.io to our existing system. This was because Close.io had a Python API that we utilized to interact with our Python server. Therefore, no time was spent researching a compatible API. By the end of the development of our first project, we had a functioning frontend and backend with security measures implemented, and a system that updates information in Close.io with our results.

The second project, labeled "Deal Analysis and Industry Engagement" (hereafter known as "Project 2" or "Deal Analysis") provided value to the first three goals. Our sponsors wanted us to analyze their previous deals to evaluate the efficacy of their decision making process. After developing the backend of our first project, where we gathered information from Apollo, we were able to derive a set of measurements for our Deal Flow spreadsheet. We used the measurements to evaluate Vestigo's profile and present our findings. It was also requested by our sponsors and graduate team that we evaluate these companies using a 4-point scoring system as laid out in Appendix D. In the 4-point scoring system, we evaluated a company based on its team, technology, market size, and valuation. After presenting our findings and our score evaluations, Vestigo used our data the following week during a meeting with InTeaHouse, a global investment platform, and Ascensus, a technical based asset manager [10, 12]. During these meetings Vestigo and their guest used our data for several business purposes. In the meeting with InTeaHouse, Vestigo used the data to select 12 potential companies to go on an expo across China to help their business expand to the other side of the world. In the meeting with Ascensus, Vestigo used the data to advertise the categorization of their decision making process.

For our third project, “Signal Search and Market Landscape Analysis” (hereafter known as “Project 3” or “Signal Search”), our sponsors requested a system to analyze a company from a scientific evaluation, and an updated Apollo algorithm to increase its sensitivity to B2B signals. We recommended that Vestigo continue the process outlined in our second project, Deal Analysis and Industry Engagement. We recognized that gathering information, labeling the company with a market segment, and labeling the company with an industry could be done automatically; leaving only the 4-point scoring to be done manually. However, complications with the project arose from interfacing with the Apollo algorithm. With information provided from Rob Fisher, we realized the algorithm’s current issue arises from an enlarged and outdated dataset that skews the results. The issue is correctable, but we projected that it required enough man hours to warrant a separate team.

The resources available to the MQP team upon initiation of the third project were not enough to create a workable deliverable while maintaining quality in the other projects. We brought this concern to our sponsor’s attention and it was their recommendation to put the project on hold until all other projects were completed. Because we wanted a high quality in all projects we began, this project was never brought to development. As such, our concept design stands in hopes that this project will be undertaken by a new team at a later date.

For our fourth and final project, Device Fingerprinting (also known as “Project 4”), the task was to design a new algorithm for Cogo Labs’ fingerprinting database. The database contained 3.64 trillion entries with an attribute representing a website cookie. Out of the 3.64 trillion cookie entries in Cogo’s database, only 30% have been assigned to users. The primary objective was for us to improve Cogo’s match rate from 30% to 45%. A secondary objective was to make this algorithm compatible with Safari browsers, as their cookies had an accelerated expiration rate. To begin this project, we met for a briefing from Matt Wiens, our device fingerprinting contact. Next, we met with the graduate team and listened to their interpretation and understanding of the project. After meeting with the graduates, we moved onto developing the concept.

Cogo sought the ability to take any two cookies in the database and calculate the probability they are related to the same user. We noted multiple complications with this method. First, it would require an excessive amount of storage. Second, the varying time complexity of each entry, and the size of the dataset resulted in a prohibitively long process time. These obstacles made it impossible to get quick results.

We then designed a new concept, which would take the most likely probabilities and store them. The revised design also went a step further by dissociating two entries from each other, and instead connected them via the concept of a Home IP. This concept allowed the grouping of a number of entries underneath an IP address they were most likely to correspond to. We then presented this idea to Matt Wiens, concluding that our end product would yield a list of users with their most likely associated Home IP, and a list of cookie entries with their most likely Home IP. By selecting a profile based off the Home IP, Cogo could accurately identify a cookie-user relationship by multiplying the two relative probabilities together. Matt Wiens agreed, and our development began.

Throughout the course of the project, very little changed. The concept of our "coefficients" was introduced to represent the accuracy of identifying a user based on User Agent, Cookie ID, and IP address. There was also the introduction of a cookie dependent and cookie independent Home IP probability calculation, so as to address an indexing issue in our calculations. After designing our concept, we developed a small scale model to deal with a single cookie. This led to the discovery that the Cogo database is partitioned to increase process efficiency. We decided to alter our concept to take advantage of partitioning, otherwise the algorithm yielded a time complexity of ~3 hours per cookie. This would have resulted in ~1 million years total process time. We finished by altering our concept accordingly, presenting to Matt Wiens, and maintaining our small scale model implementation for the benefit of Cogo's engineers.

1. Introduction

This report focuses on the accomplishments and findings of the Major Qualifying Project ("MQP") team at the Vestigo Ventures ("Vestigo") site in the Wall Street/Fintech project center. Vestigo is a venture capital ("VC") group which seeks out meaningful investments in financial technology ("fintech"). Through their partnership with a previous investment/startup Cogo Labs ("Cogo"), Vestigo uses Cogo's 5 petabytes (5120 terabytes in binary) of data to analyze which startup companies are most likely to go viral and become successful. This analysis is based on the startup company websites and other internet traffic pertaining to each startup. With this information, Vestigo is able to predict in a startup's early stages where it is going, and can make an accurate investment decision. Our MQP team's involvement with Vestigo covered all steps in the investment process from information gathering, data processing, information analysis, and decision making; all while being supported by in-depth data science research. To better organize this broad scope, the overall project was divided into the following 4 projects: Vestigo needed our MQP team to explore its venture decisions measured over time (Project 1), deal analysis & industry engagement (Project 2), and signal search and market analysis (Project 3); while Cogo needed a new device fingerprinting algorithm (Device Fingerprinting Project). Our MQP team worked on these projects under the guidance of a team of WPI graduates .

This paper begins with a background on our sponsors and a brief description of the sponsors' goals for the projects. Then we describe how the team organized and managed itself towards completing the projects. Next we go into each of the projects we worked on. They are sectioned in the paper based on the order we started them. For each project, we expand upon the background, methodology, results and analysis, and recommendations. They act as subsections within each project. The background sections describe the reasoning and value behind each major component of the MQP. In the methods sections, we talk about the decisions and steps taken. The modular nature of the first project means that Project 1's method section has subsections of its own. Each method subsection represents a different component of the overall system within the first project. No other methods section contain subsections. A results section follows the methods section of each project. We then devote a section to a discussion of future work and further development that we recommended to take place moving forward. Project 2 does not have a recommendation because the deliverable of the project itself was giving a recommendation. Finally, the paper wraps up with the project in the context of the world at large and the departing thoughts of the team members involved.

Throughout the paper are located both superscripted numbers and numbers within brackets. The superscripted numbers reference a work cited by the corresponding number in the Bibliography. Bracketed numbers indicate details on a design decision we abandoned and why we abandoned said decisions. Each abandoned decision is referenced next to the decision we took in the paper.

2. Sponsor Background

2.1 Vestigo Ventures

Vestigo Ventures is a Venture Capital group based in Cambridge, Massachusetts, which invested in early-stage financial technology companies, ranging from the Seed (first stage investment)[33] to the “A” Series round (optimization stage of investment)[34]. Vestigo fulfills a niche for early stage fintech firms in the Boston area. The company was founded in May 2015 by investor Ian Sheridan [1]. Since its founding, they have had investor and advisor Mark Casady as their Advisory Board Chairman and General Partner [2]. The most recent addition to this team was Mike Nugent, who occupies the Managing Director role [3]. David Blundin, founder and Chairman of Cogo since 2006, fulfills the role as a General Partner of Vestigo [4]. These four individuals compose the senior staff of Vestigo who, along with a twelve-member advisory board, evaluate the most promising fintech companies and innovations.

During the past three years, Vestigo invested in three fintech startups. The first one was LifeYield, a company focused on software-driven financial advice and portfolio optimization [30]. The second one was NetCapital, an open-market that connected entrepreneurs and investors [31]. The last one was Vestmark, which provides real-time technologies as financial advisory solutions [32]. In the following years Vestigo expected to evaluate several fintech startups to expand their portfolio. To assist in the process, Vestigo provides financial backing to Cogo in return for access to Cogo’s multi-petabyte database. From this data, Vestigo reinforces their deal making process, as data provides objectivity in a field dominated by emotional decisions [5].

As an early VC, Vestigo received over 270 fintech startup investment presentations since March 29th, 2016 and continues to receive around three to five investment deals per week. Their clientele derived from entrepreneurs across North America seeking early stage investment. Any startups Vestigo is interested in go through Vestigo’s due diligence. Due diligence is a rigorous process VC firms carry out to determine whether or not they will invest in a company [6]. Vestigo would first look at potential company’s key VC metrics such as the company’s fundraising goal, funding round, and valuation. Then Vestigo would look beyond the numbers and evaluate the company across 4-points: its team, technology, market size, and valuation. In addition, Vestigo uses Cogo Labs’ proprietary Apollo database as a key factor in decision making. Apollo is a database that complies clickstreams and using Cogo’s proprietary algorithm, determines a “confidence” for a domain that it believes will go viral. Vestigo leverages this data to understand the web traffic a prospective investment generates.

With three to five companies approaching Vestigo on a weekly basis, keeping track of investment opportunities could be difficult. It was important to have a central repository of the companies and relevant information not only for making the due diligence process more efficient, but also for evaluating Vestigo's portfolio and industry engagement. The algorithm Cogo set up for Apollo's signal search sometimes missed an occasional viral website. Under Vestigo we worked on three projects:

Project 1: Venture Decisions Made Over Time

Project 2: Deal Analysis and Industry Engagement

Project 3: Signal Search and Market Landscape Analysis

The overarching goal of these projects was to analyze Vestigo's past and present deals by mining data, evaluate Vestigo's industry engagement and possibly work on improving Apollo's signal search. This would provide Vestigo with the proper data to make their predictive tactical decisions for future investments. The majority of Project 1 involved technical development after we realized a shortcoming in Vestigo's data collection process, while Projects 2 and 3 focused on more analytical work to evaluate Vestigo's portfolio and improve Apollo's signal search.

2.2 Cogo Labs

Cogo Labs is a company located in Cambridge, Massachusetts that specializes in big data analytics. Through analyzing digital traffic, Cogo could identify thriving markets, products, and companies. Cogo monetized this data and profited by: aiding their in-house startup companies with valuable market insight, pointing Vestigo to high potential companies for investment, and selling valuable data to consumer groups. Cogo has an extensive team that specialized in a wide variety of fields, especially business and technology, to effectively manage their data. The current Chief Executive Officer of Cogo is Mira Wilczek, a 2004 MIT graduate. She began as an Entrepreneur in Residence from 2013 to 2016 [14]. She started her work with Cogo labs in 2013. Cogo's current Vice President is Robert Fisher, a business and economics graduate from North Carolina State University [15]. In terms of Cogo's business operations, John McGeachie is the Chief Strategy Officer. He was instrumental for Evernote's success having created their Evernote Business division, which employed more than 60 people worldwide. Mr. McGeachie had a background in both finance and technology, having graduated from Dartmouth College with a Computer Science degree.

The model for Cogo derives from TripAdvisor’s search engine functionality [16]. Their most concrete application of this mastery is through their Apollo algorithm and interface. While mostly proprietary, Apollo tracks the web presence of numerous website domain names. Apollo labels the value of the presence as “confidence”. Not all websites may be recognized by Apollo, as the presence is based off visits from a random sample. If not enough of that random sample has visited the website, Apollo will not record it. Through gathering clickstream data, Cogo incubates new startup ideas.

3. Organization and Management

To maintain organization of the multiple projects, the graduate team configured GitHub as a repository for all our code and as a progress tracker for each project. All important members of Cogo, Vestigo, the graduate team, and our MQP team had access to this repository. After our graduates organized our action items, we had to develop our timeline. To do this, our MQP team created a Gantt chart. The Gantt chart as shown in Appendix B sorts our action items by overarching objectives, resources, and dependencies. With guidance from the Gantt chart, we were able to complete our objectives while allocating time for our other projects. Any time one or more team members did not complete an objective in the expected time, the Gantt chart would reflect those changes across all other dependent objectives. This meant if any unexpected circumstances arose, the Gantt Chart software readjust the timeline for our remaining work.

To further maintain order within each project, our managerial structure took the initial shape of the graduates leading while our MQP team followed. We then evolved it a step further where each MQP team member would take charge of multiple action items, and become the designated "specialist" or leader for that topic. It was then their responsibility to ensure the task got completed, with or without aid from the team; depending on their preference. If progress was made for an objective, the leading team member would update the Git to reflect the news. Likewise, to keep the team on pace, any time an objective was completed, Andrew would update the Git and the Gantt chart, and kept the team updated with changes relevant to their development or the projects.

When approaching each project our process would start with reading any materials we had received from Vestigo or the graduate team. We would then contact and meet with the graduate team to discuss objectives, lay out action items, and set up a follow up meeting. We would then move into development. This phase always started with concept design, and a project plan. Once we had developed both, we would contact the graduates and important parties from Vestigo or Cogo. After confirming with said parties, we would then begin development.

During development we broke the solution down into smaller parts and would run constant tests to ensure quality. If each test came back with positive results, we would continue development on the next part and present the most recent development to the graduates. During any moment if we had a difficult question that would change the format of our deliverable, we would pose the question to the affected parties at Vestigo and Cogo. The objective of that step was to customize our deliverable to their needs. Finally, after we had developed a functioning deliverable we confirmed with the graduates, then the sponsors that it was satisfactory to the objectives. Any additional time we had, we spent testing and reconfiguring, while simultaneously adjusting our concept/deliverable to better fit our sponsor's need. As the base iteration of our projects neared completion, our sponsors asked us to undertake additions to these projects. Our sponsors understood the organization burden of scope creep, and thus referred to them as "bonus" additions. The particular phenomenon of extra additions occurred primarily in the first project. After all objectives and "bonuses" were completed, we would measure our success by matching up our product to our initially defined deliverable, what our sponsors wanted, and the overall functionality.

4. Project 1: Venture Decisions Measured Over Time

4.1 Background

This project was the first step towards evaluating Vestigo's portfolio and decisions over time. Through this project, our sponsors intended for us to familiarize ourselves with the internal tools they used and get a look into their due diligence process. Vestigo received investment proposals from 270+ companies when we began. We needed to gather information on these 270+ companies to have a basis for analyzing Vestigo's deals and decisions. Upon compiling this information, we performed analysis to see how the companies have been doing. These analyses determined whether Vestigo made the right decision to pass or invest. In addition, we offered recommendations for the investments that were still open. Open in this context means still under consideration for a deal with Vestigo.

The graduate students, whose role was managerial and advisory, laid down the groundwork for this project by starting a spreadsheet (the Deal Flow spreadsheet), that contained 271 companies from Vestigo's past and present opportunities. The Deal Flow spreadsheet included key VC metrics Vestigo uses to evaluate the companies. Some of the VC metrics include the companies' fundraising goal, funding round and their valuation. We, the undergraduate team, continued the work towards filling the spreadsheet. There were two ways to find the information needed to fill the spreadsheet: using the Deal Flow mailbox and using online resources.

The Deal Flow mailbox is a mailbox Vestigo uses to keep track of its incoming deals. Conversations Vestigo had with possible investment opportunities were forwarded to this shared mailbox. A mail system worked well for personal conversation, crucial in the VC world. However, it was inefficient going through emails to dig out VC metrics for each company. In the VC world, the relative efficiency of a firm in selecting and monitoring investments gave it comparative advantage over other firms [7]. The Deal Flow mailbox system was an inefficient way of keeping track and evaluating these incoming investment opportunities. After sifting through thousands of emails, we found that there were still gaps in the data. If Vestigo maintained this manual system with no additional methods, they would remain at risk of facing challenges moving forward because this system is not scalable.

The second way we gathered information was by utilizing online resources. We used resources such as Crunchbase, FormDs, and Pitchbook to find information on the list of companies. Early on in the project, we found that there was a general lack of information out there that described the key VC metrics for the companies. Finding funding information in particular became a real challenge. We also had cases where the information available would return conflicting results. These results would come directly from sources identified by Cogo as the standard for investment data retrieval. For instance, FormDs reported that Riskalyze raised $50 million [8] while Crunchbase reports that the company raised $23.5 million [9]. A reason for this could be the way Crunchbase and FormDs collect information. Crunchbase is a crowd-sourced platform, meaning that both people and companies enter data on themselves and others. FormDs is a database of companies that have filed a Form D. If the companies happen to not file a Form D or update information in Crunchbase, inconsistencies such as the one mentioned will arise. The general lack of consistent data made filling the Deal Flow spreadsheet not a challenging task, but an impossible one without having direct contact with the company.

Our two methods of gathering information were insufficient. After using the Deal Flow mailbox and online resources, we found that 244 companies out of 271 (approximately 90%) of the companies were still missing some sort of information. Having necessary information available and an efficient procedure in place is important for the success of a venture capital firm [7]. The obstacles we faced when trying to gather information resulted in recognizing the need for a better, more efficient platform.

The graduate team proposed creating a form Vestigo could use to collect information. The form would be used to fill in any gaps that were in the Deal Flow spreadsheet. In addition, it would also be used when companies approach Vestigo in the future. We along with the sponsors outlined the following requirements for the form:

1. Have new and existing companies be able to add and modify their information.
2. Prevent users from updating information that is not theirs.
3. Have a central database where the information is stored.
4. Have the form attached to a URL so Vestigo has the ability to integrate it to their website and be able to share a link with prospective companies.
5. Utilize information that can be leveraged from Cogo Labs' Apollo database.
6. Possibly integrate the automated system with Close.io.
7. Accommodate the needs of North American startups.

A successful implementation of this form would result in a streamlined and efficient way of compiling information. The form would be instrumental in filling in the gaps in the Deal Flow spreadsheet we were unable to fill. It would also facilitate the due diligence process for Vestigo by offering a centralized location with necessary information on investment opportunities.

4.2 Methodology

Upon realizing the shortcomings with Vestigo's data collection process, we along with the graduate students and the sponsors decided to create an online form with design requirements described in section 4.1. This project required the creation of multiple modules. For the components, we used JavaScript, React.js, Firebase, SQL, Python, Flask, Cogo's proprietary database, and Cogo's proprietary Python API.

We started by uploading the Deal Flow spreadsheet we compiled with the graduate students to a table in Cogo's databases. Once we had the information in a centralized location, we worked on an SQL query that would pull confidence levels and detection dates for the companies from Apollo. After accomplishing that, we moved on to automating the process of adding and modifying companies through an online form, and integrating the system with the CRM Vestigo uses, Close.io.

4.2.1 SQL Query to Extract Information from Apollo

After we completed compiling information on the companies, we ported them into a table we created in Cogo's databases and started developing an SQL query (update_from_apollo) within Cogo's database. The update_from_apollo query gathered the earliest and latest domain tracking detection dates within Cogo's proprietary database managing interface and their corresponding confidence level (which is the output of Cogo's analysis algorithms). This was taken from a table in Cogo's proprietary database called "apollo_histories" and used to update the relative attributes in the table we created.

The Deal Flow database (wpi_vestigo_deals) has 25 columns. Our SQL query updates the following five columns: in_apollo, initial_detected_date, initial_confidence, latest_detected_date and latest_confidence. In addition to updating these columns, it performs simple arithmetic on the values that were mined and updates three additional columns: total_days, total_confidence_growth and confidence_growth_per_day. The particulars of the arithmetic used are found in the methods of Project 2. All of this information for the companies is mined from Apollo using the company's website.

In our SQL query we chose to use a WITH statement to temporarily define tables in the query for the parameters we were going to pull from the Apollo database (Appendix E [2]). Following the WITH statement was an UPDATE statement that updates all the columns in the Deal Flow database using the information from Apollo and performs simple arithmetic to update the remaining columns. The SQL query can be found in Appendix C.

With the backend SQL query working, we worked on automating the process of updating the Deal Flow database by creating a form companies seeking investment can fill out. To automate this process the system required multiple components of design: additional SQL queries, a Python Flask server and a fillable form with a JavaScript backend. These components are explained in further detail in the following sections. The overall flow of data is as follows (Figure 1):

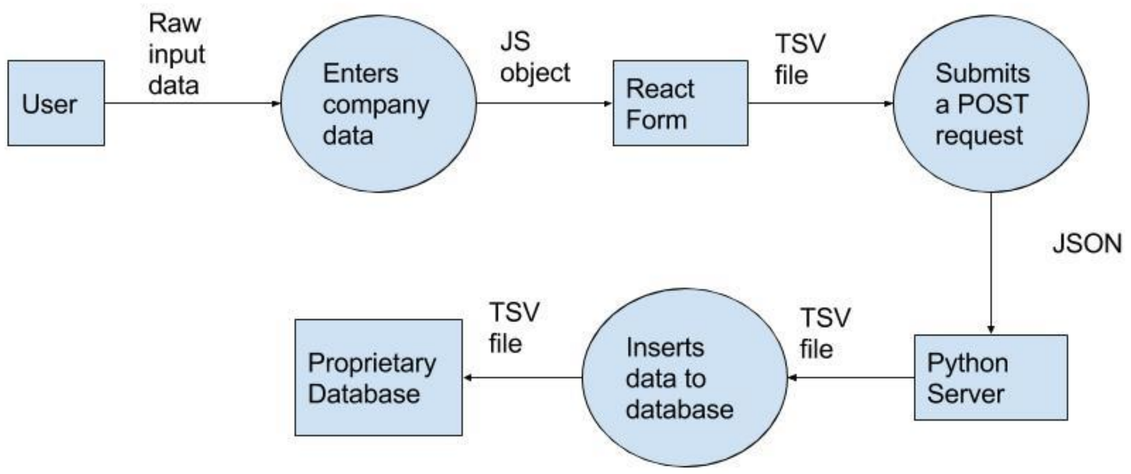


Figure 1. Diagram describing flow of data

4.2.2 Additional SQL Queries

In addition to the update_from_apollo query described in section 4.2.1, the design of an automated system required three additional queries: update_existing_deals, insert_new_deals and clear_survey_entries.

Early on, we found that when we upload a .tsv file, the datatype of the columns is overwritten in the database. The update_from_apollo query required certain columns to be a specific datatype to conduct algebraic operations. Because of this, we chose to create another table that would act like a buffer (wpi_survey_entries). We would upload the .tsv file to the wpi_survey_entries table. Then, using the update_existing_deals and insert_new_deals queries, we casted the data types and moved the entry over to wpi_vestigo_deals.

We decided to have two separate queries, one that updated existing entries and one that inserted new entries (Appendix E [3]). The update_existing_deals query went through the wpi_vestigo_deals table and if the company is already in the table, the query updated information for the company. The insert_new_deals query added new queries to the table. This query checked that a company is not already in the database and inserted a new entry.

Table 1: An example of duplicate queries on basis of company name and domain.

	company	domain	round	contact_email
Entry A	Some Company	someurl.com	Angel	
Entry B	Some Company	someurl.com		contact@email.com

In Table 1, entries A and B would be treated as duplicates because they have the same company name and website. Using a query to delete either of these entries would result in losing either the 'round' or 'contact_email' information. By creating two separate queries that will update or insert deals to the table, we avoid creating duplicates altogether and prevent data loss. Once the entry is properly inserted or updated in the wpi_vestigo_deals table, the clear_survey_entries query will delete everything in the wpi_survey_entries table.

4.2.3 Python Flask Server

For the backend design of this system, we utilized Cogo's proprietary Python API they use to interface with their databases. The API provided ways to perform a variety of things including creating or editing queries, uploading .tsv files to tables, and queueing queries. We familiarized ourselves with this API by creating a short script that uploads a locally saved .tsv file and runs the update_from_apollo query.

The next step was to link the submissions from the form to this Python script. To make the Python script accessible, we decided to use Flask, a "microframework" written in Python [18]. The Python Flask sets up a web application, and the Python script is triggered when a request is sent to the URL. By turning our existing Python script into a Flask application, we were able to navigate to a URL and trigger the .tsv upload and query execution. Upon completing the query executions, the Flask server also utilized the Close.io Python API to update a company's information on the Close.io platform. Close.io is a customer relationship management system where our sponsors track their email communications with interested investors. Our sponsors use Close.io more regularly than accessing the Deal Flow database, so integrating the automated form with Close.io became a good addition to the overall system.

During this process, email notifications were sent using Flask-mail, an extension used to send SMTP email messages from a Flask application. Four different email notifications occurred during the execution of the form (Appendix F). The first one was a message containing an automatically generated passcode, which was sent when a user signed up. The second and third notifications occurred as a confirmation of form submission; one of the emails got sent to the user submitting the form and the other one to Vestigo's Deal Flow mailbox. The last notification occurred when a user submitted an update, which got sent directly to Vestigo's Deal Flow email as well.

Finally, to deploy the Flask and the form, we used Docker. Docker is a software technology that provided an additional layer of abstraction and automation of operating-system-level virtualization on Windows and Linux [19]. We set up two different containers, for our two applications, the Flask and the form.

4.2.4 Form with React.js

Prior to coding the form, the graduate team presented the idea of two forms. This followed the principle of separating the case for adding a new entry and overwriting an old entry. While this adhered to good coding principles, security flaws create a danger with this approach. More forms mean more public points of entry into the proprietary database, thus creating more risk for Vestigo. Hence our team decided to compose both functions into one form. It would go against separation of concerns, but resulted in increased security (Appendix E [4]).

We drafted our own form to overcome the partition between Cogo and the outside world. The prospect of a form application developed through the use of the React.js framework hosted in concert with the Flask server proved desirable to all parties. The sponsor could realize their vision for the look and feel of the form, and we could manually script it to facilitate integration into the overall system.

When a user clicks on the link to the form from anywhere, they are prompted with a login screen (Figure 2). Should the user encounter the form for the first time, they can click on the sign up button to send a randomly generated password to their email (Appendix E [5]). With the password, the user then logs into the form and begins inputting data.

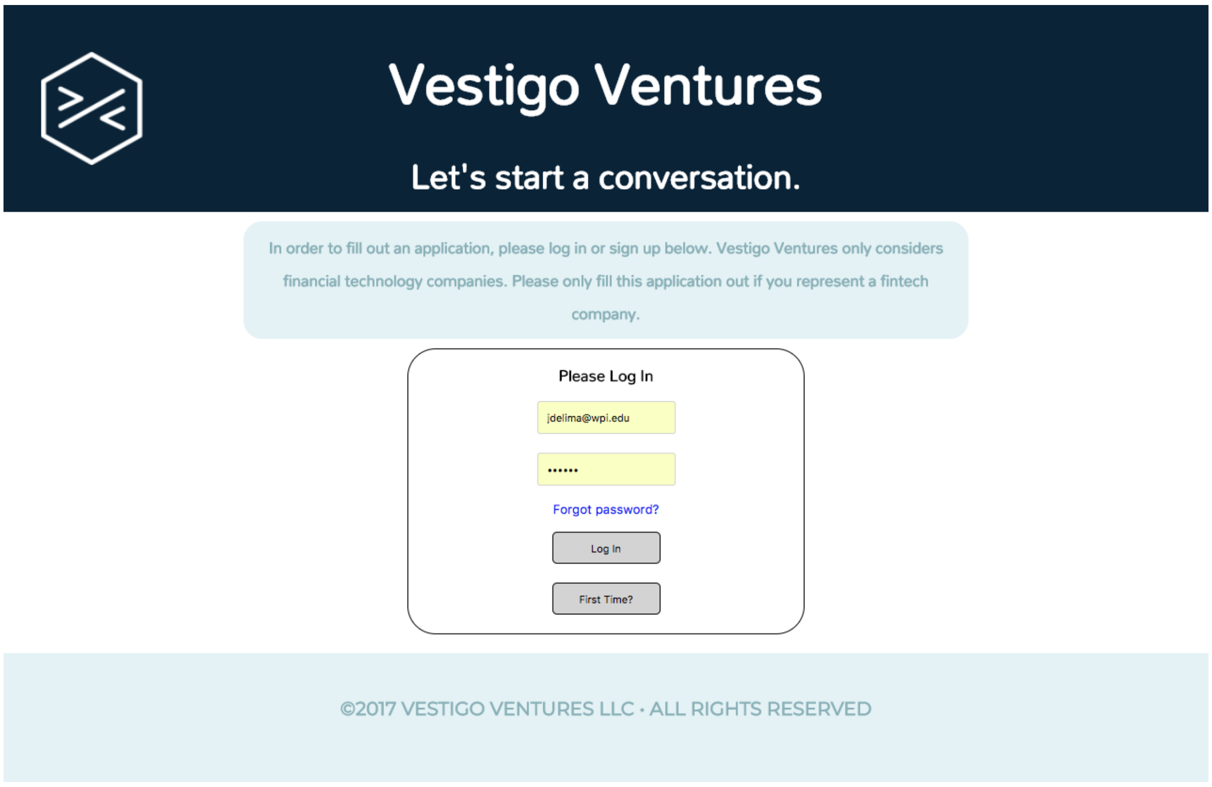
The form itself included two tabs: tell us your story and give us an update. On the first tab, there are seven required fields (Figure 3). The user cannot submit until all the fields have valid inputs. For example, fields that require email or money inputs required a certain format to be considered acceptable. When the form detects that the client meets all requirements for submission, the submit button turns green. Submission begins the information transfer from the form to the Python Flask server.

The second tab gave the user the option to 'give an update', which is intended for use when the company is already in contact with Vestigo (Figure 4). The layout for this option is different: it contains three required fields and an optional 'file upload' to send a file to Vestigo's Deal Flow email with the update information. Just like in the first tab, each field checks for valid input. An admin account exists to be used by specific members of Vestigo Ventures. This admin account has its own set of credentials, and views the form in a different way. While logged in with the admin account, the only required fields are company name and domain. Further, this account has permissions to modify or update any company's information.

For transferring the form information, the form API generates three different requests though the XMLHttpRequest format. The first one packages all information from the form in a .tsv file. The .tsv file is encapsulated into a JSON file (Appendix E [6]), and sent through a post request to the Flask server's root route. Upon receiving the JSON file, the server obtains it within the payload and writes it to a temporary .tsv file. Several functions from Cogo's proprietary API begin to execute.

The first function uploads the .tsv file information into the database. The next four functions call four SQL queries and print out their execution for transparency. Upon conclusion of these queries, the server deletes the .tsv file and returned "Done". With an automated data pipeline, the need for manually obtaining and uploading data becomes obsolete.

The second request gathers basic user information from the state of the form API, and sends it using the POST method to the Flask server. Instead of sending the information to the '/' route, it sends information to the '/confirmation' route. Then, the server script utilizes Flask-mail to send a confirmation email to the user and Vestigo's Deal Flow mailbox with the purpose of letting them know that the form information was uploaded and processed correctly (Appendix E [7]).



Vestigo Ventures

Let's start a conversation.

In order to fill out an application, please log in or sign up below. Vestigo Ventures only considers financial technology companies. Please only fill this application out if you represent a fintech company.

Please Log In

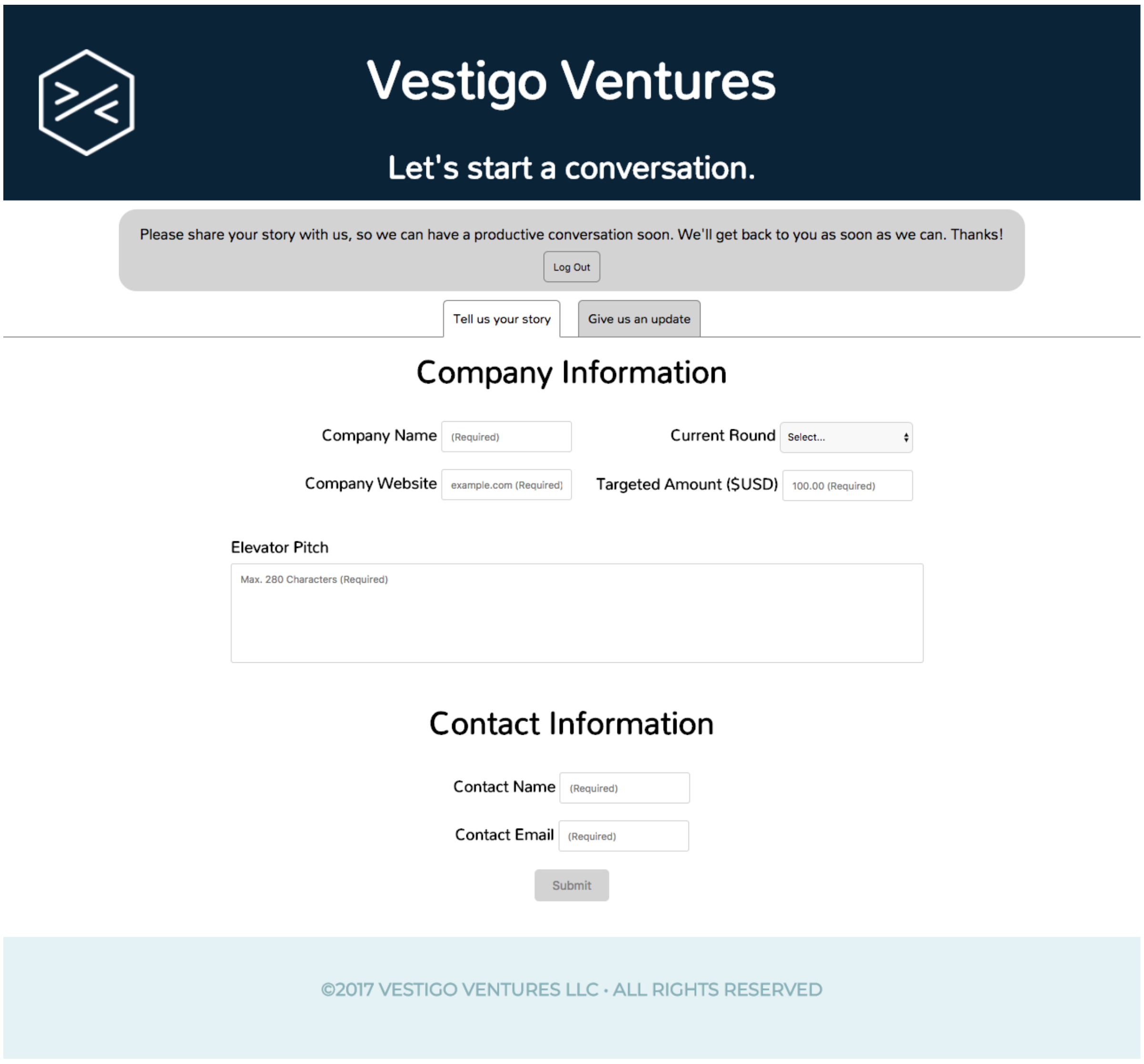
jdelima@wpi.edu

••••••

©2017 VESTIGO VENTURES LLC • ALL RIGHTS RESERVED

Figure 2. The login screen of the form.

The third request is similar to the second one. It takes all of the information filled into the form from the update tab and sends it through a POST request to the './update' app route. Then, just like in the second request, it sends an email using the Flask-mail extension. Additionally, the update tab has the option to include a file. If the user decides to upload a file to support their message, it will be sent through the request and included in the email as an attachment.


Vestigo Ventures

Let's start a conversation.

Please share your story with us, so we can have a productive conversation soon. We'll get back to you as soon as we can. Thanks!

Log Out

Tell us your story | Give us an update

Company Information

Company Name (Required)

Current Round Select...

Company Website example.com (Required)

Targeted Amount ($USD) 100.00 (Required)

Elevator Pitch

Max. 280 Characters (Required)

Contact Information

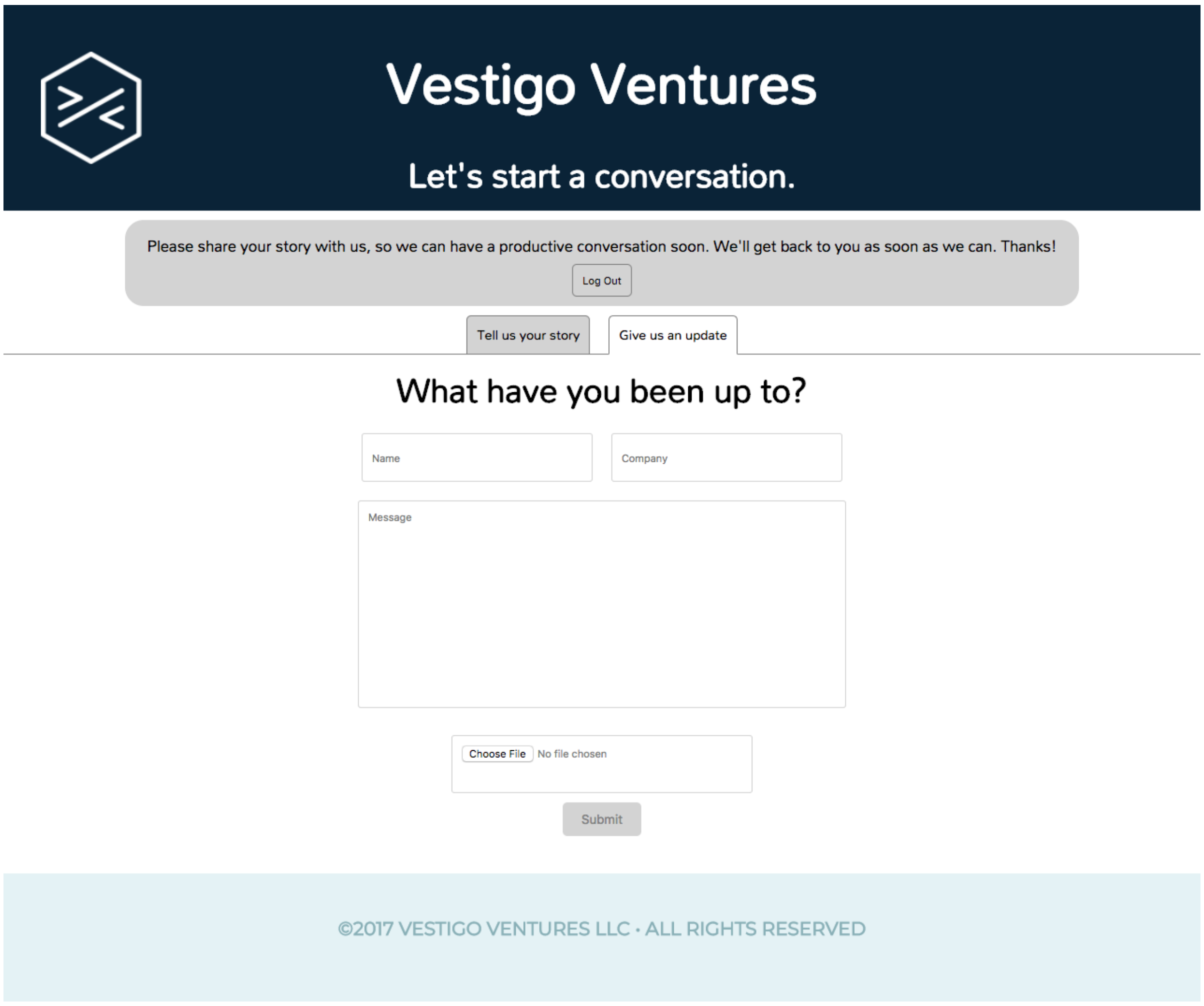
Contact Name (Required)

Contact Email (Required)

Submit

©2017 VESTIGO VENTURES LLC • ALL RIGHTS RESERVED

Figure 3. The main page of the form.


Vestigo Ventures

Let's start a conversation.

Please share your story with us, so we can have a productive conversation soon. We'll get back to you as soon as we can. Thanks!

What have you been up to?

©2017 VESTIGO VENTURES LLC • ALL RIGHTS RESERVED

Figure 4. The update page of the form

4.3 Results and Analysis

Project 1 involved data mining and developing an automated system. When researching the companies in the Deal Flow table, we found most companies lacked information. This is one of the issues the automated form helped mitigate. The final system pipelined data from the form to Close.io. It offered prospective companies a secure way of giving their information to Vestigo. In addition, it used the confidence levels generated by Apollo to aid Vestigo with its decision making.

When developing the automated system, there were a variety of challenges. We had to develop on several platforms at once. The system's components had to communicate with one another and rely on the available resources, i.e. proprietary tools and the engineers on site. We found the limitations on certain platforms impeded the process on the other platforms. Proprietary tools and libraries held functionality that worked around said limitations.

A modular design accommodated the various independent components of the form system. The structure of the system eased test-driven development. We ran manual tests on each part and made changes without affecting the operation of other components. Should a future group change our system, they can refactor with ease one part at a time. Whenever we made changes to the system, we experienced little difficulty due to the modular structure.

We ran a full system test once we confirmed the workability of each part. We drafted several common use cases for a client's interaction with the form. These tests realized the use cases as well as potential malicious input. At the conclusion of testing we confirmed with our sponsor that our pipeline worked as intended.

4.4 Recommendations

We recommend the assignment of one specialist to support the project 1 system on behalf of Vestigo. This person would need access to Cogo's proprietary database managing interface, have a good understanding of Python and SQL, and understand how Vestigo wants to interact with their clients. Either Cogo or Vestigo may assign one of their employees to the task.

5. Project 2: Deal Analysis and Industry Engagement

5.1 Background

Vestigo's due diligence involves evaluating a company beyond its financial numbers. Vestigo looks into a company's team, technology, market size and valuation before moving forward with the conversation. The goals of this project was to perform analysis on Vestigo's past and present deals, to gain an understanding of Vestigo's industry engagement, and to get exposure to the business world.

To evaluate the portfolio of a venture capital firm, one of the items that can be done is evaluating the deals they passed on. Vestigo appreciates insight on deals that they passed on that could have provided value. Another portion of evaluating a VC firm's portfolio involves looking at its industry engagement. For VC firms, having a specific industry focus reduces their due diligence time to investigate potential investments. In addition, it increases the firm's ability to leverage their industry expertise and network. We conducted research on all the companies in the Deal Flow spreadsheet to assess Vestigo's industry engagement. We did this by classifying the companies into one of four fund investment segments (Figure 5): market structures, operation solutions, worksite management and personal wealth.

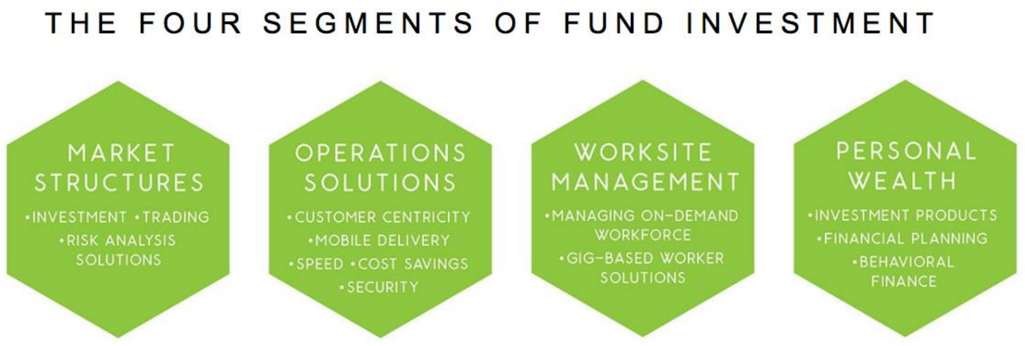


Figure 5. The four segments of fund investment

The basis of successful VC deals derives from the quality of conversations with potential clients. Our sponsors welcomed us to four of these meetings as a means of understanding the practical aspects of venture deal making. The four companies we met with were Ascensus, Horace Mann, Virtual Cove, and InTeahouse. All four acted in a different field of fintech. Ascensus provides technological solutions to asset managers [10]. Horace Mann provides insurance to non-college educators [11]. Virtual Cove uses virtual reality software to visualize data [12]. InTeahouse is a global investment platform [13]. Beyond the VC context, the goal of Vestigo is to establish long term relationships with all their clients including these four.

5.2 Methodology

To conduct analysis on the deals Vestigo had received we used the queries we built in Project 1 to pull the confidence levels from Apollo. As previously discussed in section 3.2.1, the update_from_apollo query did simple arithmetic to calculate: total_days, total_confidence_growth and confidence_growth_per_day.

- total_days is the difference between the initial and latest detected date
- total_confidence_growth is the difference between a company's initial and latest confidence
- confidence_growth_per_day is the total_confidence_growth divided by the time interval between the initial detected date and the current date.

This was the most accurate depiction of confidence rate (Appendix E [8]).

Using the values generated by the query, we performed analysis on 20 companies that saw the highest confidence_growth_per_day and whose current status was marked as open to investment by Vestigo. After we had narrowed the selection down to 20 companies, we did more specific research and used a 4-point scoring method evaluating each company on its team, technology, market size, and valuation. We used information and data found on Crunchbase, FormDs and other sources to do the research for each top 20 company and give them score 1 to 5 for these four fields.

To rate the company's team and technology, we used a metric Vestigo provided. However, in order to rate the company's valuation and market size we had to come up with our own metric. Valuation is how much a company is worth. For some of the companies in our list, this information was available but for others it wasn't. So, we based our valuation ratings on the funding data we could find for the company. We believed the amount of funding that a company has raised can represent the confidence of other investors indirectly answering how much the company is worth.

Scoring the market size was tricky. Market size represented the number of individuals in a certain market who are potential buyer or sellers of a certain product/service. Determining the company's market size based on the information we have available to us was challenging. We did some research and came up with this formula:

$$market\ volume\ =\ number\ of\ target\ customers\ *\ penetration\ rate$$

where the number of target customers refers to the total portion of the market the service can reach. For instance, something like life insurance would have a high number of target customers because the insurance industry is quite large. We gave the companies a 1-5 rating for their 'number of target customers'. To determine a company's penetration rate, we looked at what proportion of the customers the company could get within the industry. When rating the penetration rate of the company, we looked at the novelty of the idea and the amount of competition that exists in the industry. A company that is very novel and has little competition would receive a high penetration rate. We gave companies a rating of 1, 3 or 5 for their penetration rate. Using the equation stated above, we had a market volume scale that ranged from 1-25. Using this score we determined the market size rating on a scale of 1-5. After we finished the four-point scoring on team, technology, valuation and market size, we picked three companies with the highest score for our recommendations to Vestigo. The more detailed rubric we used for the four-point scoring can be found in Appendix D.

To analyze Vestigo's industry engagement, we classified all the companies in the Deal Flow table into one of the four segments Vestigo invests in. The four segments are: market structures, operation solutions, worksite management, and personal wealth. Market structures deals with trading analysis, risk analysis, algorithmic trading and blockchain. Operation solutions include internal tools/software, frontend/backend and technical stakes. Worksite management encompasses employer benefits from a high-level perspective, i.e. from the perspective of the business. Financial services related to the individual, such as a personal financial advisor, fall under the personal wealth category. We assigned each company to exactly one segment. Each assignment required research evaluated alongside the four-point scoring rubric.

We concluded our Deal Analysis by presenting our findings and results to Mike Nugent and Ian Sheridan.

5.3 Results and Analysis

Project 2 involved the due diligence process of Vestigo's deal making. We classified all 271 companies in the Deal Flow database into four categories: Market Structures (MS), Operation Solutions (OS), Worksite Management (WM) and Personal Wealth (PW) as described in section 3.3. Two super-categories, B2B and B2C further defined Vestigo's investment goals. Ninety companies belonged to OS, 79 of them belonged to MS, 62 companies belonged to PW, 18 companies belong to WM, and 21 companies lacked enough information for categorization. The OS and MS companies best performed the functions of the B2B super-category, while the WM and PW companies fit the B2C super-category. In other words, other businesses act as the customers of OS and MS businesses, while WM and PW services have individual persons as clientele. This process followed the guidelines of the graduate team's recommendation.

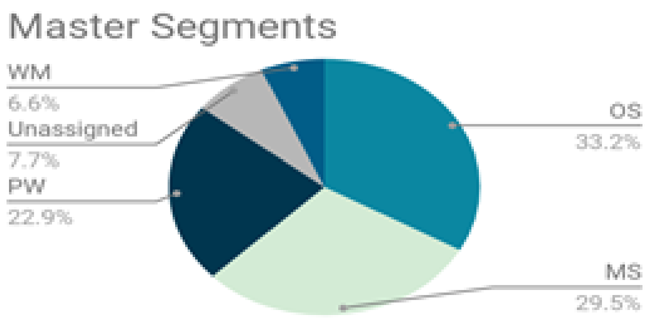


Figure 6. Market Segments Distribution

For all companies in the Deal Flow spreadsheet, there are three statuses: passed, open, or invested. The spreadsheet displays three companies as invested and 58 as passed. Passed means that Vestigo passed on the opportunity without investing in the company. Open means they are open to the possibility of investment but remain unsure about commitment. We ran the update_from_apollo query described in section 3.2.1 and found that Apollo detected 67 of the 271 companies in our table. For those 67, we recorded their confidence level and confidence growth rate fields. We found that 46 of the 67 had a positive confidence growth and 5 companies saw a decline in their confidence. The remaining 16 companies had no change in their confidence level because Apollo only detected the company's domain once.

Apollo detected around 25% of companies. This was significantly lower than the sample size needed for accurate measurement. Given our population of 271, a desired confidence level of 99%, and a margin of error of 1%, the calculated sample should be 267 [24]. While a value 67 companies missed the desired margin of 267 companies by a value of 200, it gave us a way of narrowing down the list of companies based on the web traffic they generate. Furthermore, filtering the companies using the 4-point scoring added dimension to the way we were analyzing the companies. An analysis based on team, technology, and market size added breadth to our confidence analysis. We presented the top 20 deals Vestigo passed on and the top 20 that are still in the open category. In the interests of Vestigo, we focused on the five companies that are still open and had the highest scores after the 4-point scoring. The sample of the Deal Flow table is confidential and is not in this paper.

6. Project 3: Market Landscape Analysis

6.1 Background

Cogo’s current signal search algorithm has been invaluable for B2B communication. However, for this project Vestigo wanted a tangible and applicable algorithm to aid in fintech investment decisions. As such, this project came in two parts; improving the Apollo algorithm’s signal search parameters for Rob Fisher to review, and performing a market B2B landscape analysis for Vestigo based on the results of the new signal search algorithm. As mentioned in section 2.1, Vestigo uses Apollo as a key factor in decision making but misses the occasional viral website. The risk of missing viral websites means that if an investment opportunity grew substantially in a short amount of time, then Vestigo missed a chance to capitalize on its growth. Therefore, improving upon Apollo’s signal search would be useful in ensuring an earlier detection for virality.

6.2 Methodology

For this project we discovered two key objectives; improving the Apollo algorithm’s signal search parameters for Rob Fisher to review, and performing a market landscape analysis for Vestigo based on the results of the new signal search algorithm. After finishing the Deal Analysis Project ,our team understood that the market landscape analysis could be completed for all Apollo entries; provided they were indexed by the industry and market segment of best fit. Without sorting through the database by hand, our team had determined the most efficient way to accomplish this was creating an automated sequence for parsing and search terms that would take place immediately after the signal search. We concluded this code would act on a table in our database containing Pitchbook data that Cogo had previously purchased; otherwise, it would run on a domain’s website as a “catch-all” and look for particular keywords. After we had devised this concept, we had suspected the level of complexity for implementation would interfere with our other projects. Knowing this, we brought the issue to Mike Nugent and Rob Fisher. Mike agreed with our suspicion and made the executive decision to postpone the project until all other projects (Venture Decisions Measured Over Time, Deal Analysis Project , Device Fingerprinting) were fully developed. This postponement includes the possibility passing off this project to another group.

6.3 Results and Analysis

Our core deliverable, the basic concept outlined in section 6.2, satisfied our sponsor's expectations. After presenting our project, our sponsors noted that we had found a "failure mode" of the company. Only one scientist at Vestigo understood the entirety of the Apollo algorithm and could make updates to it. We derived no further deliverable from this project.

6.4 Recommendations

We had one recommendation for Vestigo Ventures and one more recommendation for Cogo labs. Vestigo Ventures should give resources for one or more specialists to learn and master the Apollo algorithm. Cogo Labs assign a research team to follow up on this project. A workable alternative to a research team possibility is another WPI MQP team. Before pursuing this project, the aims should be reevaluated by Vestigo and Cogo to ensure they are up-to-date. The assigned team can decide if our findings remain problematic.

7. Project 4: Device Fingerprinting

7.1 Background

Cogo Labs had 3 petabytes of consumer profile data consisting of ~800 million profiles. Big data can be evaluated over three dimensions known as the 3Vs: volume, variety and velocity [29]. According to a Cogo employee, Cogo obtained nearly 5 billion rows of data daily from a random population. This grouping covered 80% of the United States populace, therefore in addition to volume and velocity, the data Cogo compiled guaranteed variety.

By using this database and an extensive cookie match network, Cogo could pinpoint an exact user 30% of the time, given that the user goes through any of the websites monitored by Cogo or its partners. Cogo used emd5 hashes in their process of matching cookies with users. An emd5 hash derived from the MD5 hashing algorithm [17]. However, the emd5 algorithm hashes email addresses in particular. This is done before the data touches Cogo’s network as a means to maintain the privacy of the user and for ethical concerns. Cogo does not want Personally Identifying Information (“PII”) to touch their network. PII included a non-hashed email, phone number, date of birth, and social security. In the case that an emd5 could not be traced to a cookie, other browsing signatures facilitate detection. Such signatures included user agent (the physical device used), IP address, and browser version. The objectives of this project were to increase Cogo’s match rate from 30% to 45%, and to make the system compatible with Safari browsers.

The concept of device fingerprinting originated a little over ten years ago [20]. General implementation began seven years ago [21]. Given the novel concept of fingerprinting, our group began the project reading through several research papers pertaining to browser fingerprinting. Much of the following material derived from the methods outlined in a Microsoft research paper [22]. We were tasked with designing a new algorithm for Cogo to build off of. After weeks of development, we decided to build a Python program to implement and exemplify the algorithm.

7.2 Methodology

Our first step in making the fingerprinting probabilistic model was to design a concept that would tie a user to a particular cookie. We were informed by our Cogo Labs mentor for this project, Matt Wiens, that only approximately 30% of the database has a user tied to a cookie (stored in the scratch.matches table of the Matches database), hence we need to connect the other 70% of the data in scratch.ad_requests table to a valid user. To bridge the gap between users and entries, we noted that all entries have the same three fields in the database; apxlv_id (cookie id), user_agent (device information), and ip (Internet Protocol Address). We used a combination of these fields to determine the probability of an entry matching a user. Particularly, we developed the concept of Home IP addresses. We based our Home IP concept on the frequency of IP addresses. The most frequently appearing IP address for a particular cookie would likely be the Home IP address of this cookie(Appendix E [9]). We could then use this Home IP as a best guess as the home location of the user it's associated to.

In many of the following formulas, there are three probability coefficients; C, U, and I. They represent the accuracy of using cookie, user agent, and IP address to identify a unique user, respectively. As we explored multiple possibilities for setting our coefficient, we have listed three options for the choice of these coefficients. All three of these options have been thoroughly tested, and our findings have been placed into section 4.4 of this paper.

First, by assuming all users could be accurately identified by cookie, user agent, and IP lead us to temporarily setting all coefficients to 1. In which, we could take a more scientific approach. However, we understood our calculations would be less accurate to what they were meant to represent. This was used strictly for testing and proof of concept until the graduate team had time to work out the coefficients with us.

Second, we used the numbers coming from the Microsoft research paper we found [23]. The authors of this paper applied their algorithm to calculate the percentage of fingerprints that correspond to a host. The result is C = 0.8235, U = 0.6201. In order to calculate the IP coefficient, we reverse-engineered the calculation of UA-IP combined coefficient in conjunction with the user agent coefficient, which equals to I = 0.4899.

Third, we decided to use a similar method from the one explained in the Microsoft paper[14] to calculate our own coefficients. This method was designed by the graduate team, while the detailed calculation was done by our undergraduate team. We used the total counts for cookie, IP and user agent divided by the total count of entries in scratch.matches table.

To start our calculation process, we noted that there are two datatables within the Matches database: scratch.matches and scratch.ad_requests. Scratch.matches contains the fields emd5 (a hashed email address for security), apxlv_id , user agent, ip, and numerous additional fields that we did not use. Additionally, scratch.ad_requests contains the same fields except emd5. We gathered all emd5s, ips, and user agents from each entry in the scratch.matches table and scratch.ad_requests table for one unique cookie and inserted them into a table named scratch.wpi_device_fingerprinting (a.k.a “ALPHA”) to calculate our Home IP, which is simply the most frequent IP address amongst all entries of the same cookie.

When gathering all this information and inserting it into the ALPHA table, we found the execution time of the queries for a single cookie to be around of three hours. After some research on the database's structure using the “explain” SQL command, we figured out that Cogo’s database was partitioned (or subdivided) by the date the entry was created. Additionally, the scratch.ad_requests table was partitioned by cookie suffix. The partitioning of this data allows for improved performance when executing queries on these tables [23]. Hence we decided to execute our queries between 2 dates, while also sticking to a particular cookie ID suffix. Taking advantage of the database’s partitioning, we reduced the execution time of queries to an average of 3 minutes for testing purposes only.

We then counted the number of times the Home IP and IP are the same within the same entry, multiplied by our coefficients, then divided that by the total number of entries within ALPHA(Appendix E [10]).

$$\text{cookie\_dep\_hip\_prob} = \frac{(\text{Frequency count where the specific IP=HomeIP}) \times C^2 \times \mathrm{I}}{\text{Total count of Cookie}}$$

This calculation became the cookie_dep_hip_prob (or cookie dependent Home IP probability), signifying the probability a Home IP was accurate while indexing by the cookie (Appendix E [11]). This was done so Cogo could see the probability of a Home IP belonging to a user with respect to the entirety of the Home IP profile where all entries have the same cookie (i.e. the same user). After calculating, we moved the results from ALPHA to a theoretical BETA table. At the time of the paper’s writing, all aspects of the BETA table were based off projection as opposed to empirical analysis. As such, no post-process storage is required. For implementation of 2 or more cookies, all Home IP’s would need to be calculated before calculating probability. In this instance, the BETA table would need to be initialized for post-process storage of a set of entries for one unique cookie at a time. After moving our data from ALPHA to BETA, we deleted all entries in ALPHA to prevent duplication. To end our first phase, we repeat this process for every unique cookie in the database. This resulted in every cookie being mapped to a Home IP address(Appendix E [12]).

In the second phase, after all entries have been assigned a Home IP, we copied all entries with a unique Home IP from BETA back to ALPHA. Within ALPHA we counted the number of times the Home IP and ip fields are the same for an entry, then multiplied by our coefficients, and divided that by the total number of entries in ALPHA.

$$\text{cookie\_ind\_hip\_prob} = \frac{(\text{Frequency count where the specific IP=HomeIP}) \times \text{C} \times \text{I}}{\text{Total count of Home IP}}$$

This calculation became the cookie_ind_hip_prob (or cookie independent Home IP probability), signifying the probability the Home IP is accurate regardless of indexing by the cookie, much like the cookie_dep_hip_prob. The utility here mimicked the utility of the cookie_dep_hip_prob as well: so that Cogo can see the probability of the Home IP belonging to a user with respect to the entirety of the Home IP profile and all the relevant entries. We started to calculate the end deliverable: the cookie_prob. This signified the probability that the cookie belonged to the Home IP (i.e. user). We did this by taking each entry in ALPHA, and retrieving how frequently the entry's cookie, user agent, and/or IP show up in the rest of the table. Then for the individual frequencies of cookie, user agent(Appendix E [13]), and ip we completed the following formula for our first entry (Appendix E [14]).

$$\text{Cookie\_Prob} = 1 - (1 - X)(1 - Y)(1 - Z)$$

Where:
X = C * Cookie_Frequency
Y = U * UA_Frequency
Z = I * IP_Frequency

We took our final calculation and inserted it into our first entry cookie_prob field(Appendix E [15][16]). Then we iterated through our list of entries in ALPHA, performing the same steps, from cookie_dep_hip_prob to cooie_prob, for each one (Appendix E [17]). After calculating, we removed all entries with that Home IP (partition) from our BETA table, and inserted all entries from ALPHA into BETA to replace the partition. Finally, we deleted all entries in ALPHA to optimize space for future space usage. We then did these calculations for all unique Home IP's within BETA. Our ideal deliverable was all entries within BETA having a Home IP, and a cookie_prob. This datatable will act as an archive on all entries and users, and should pre-process the information for quick acquisition.

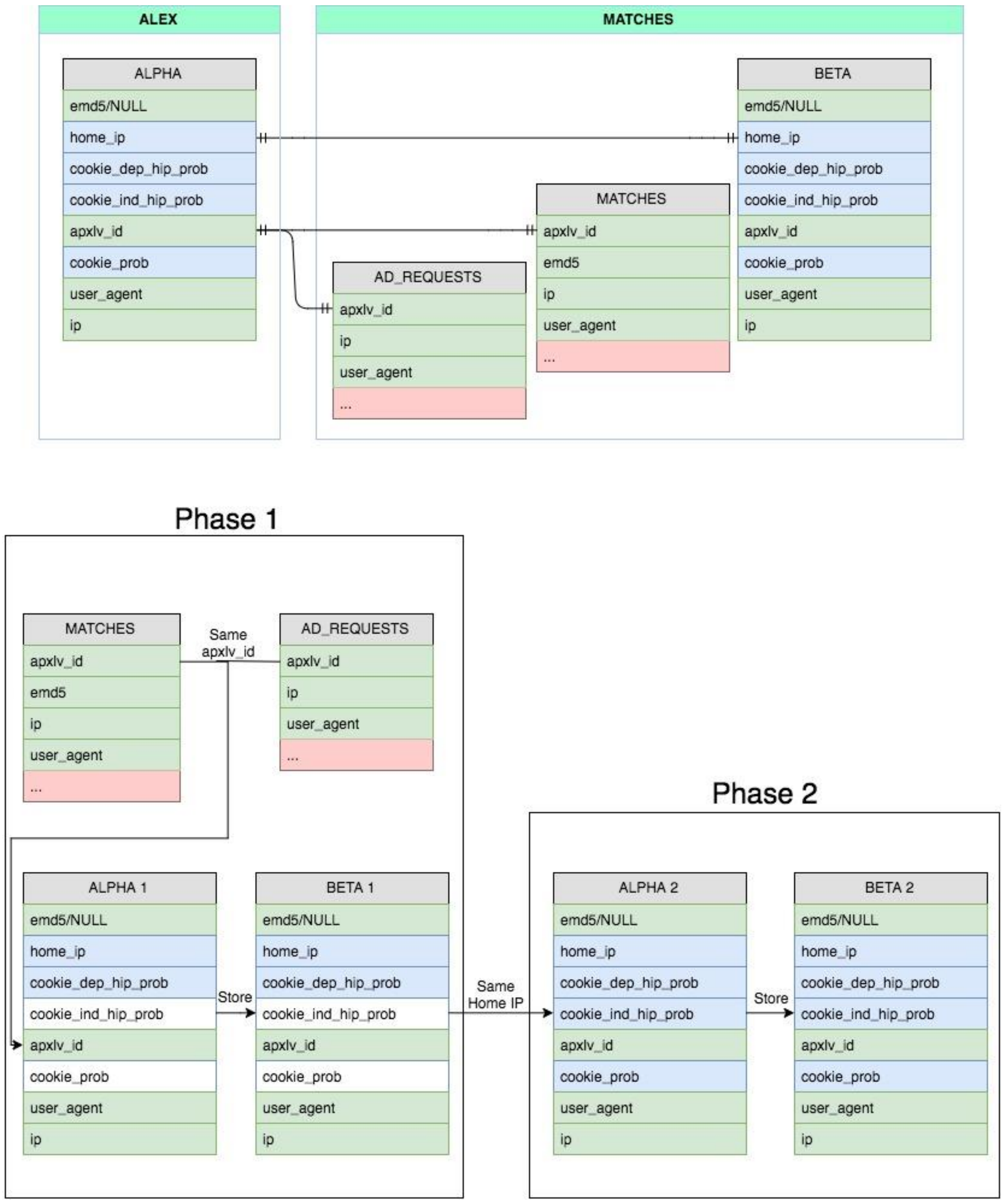
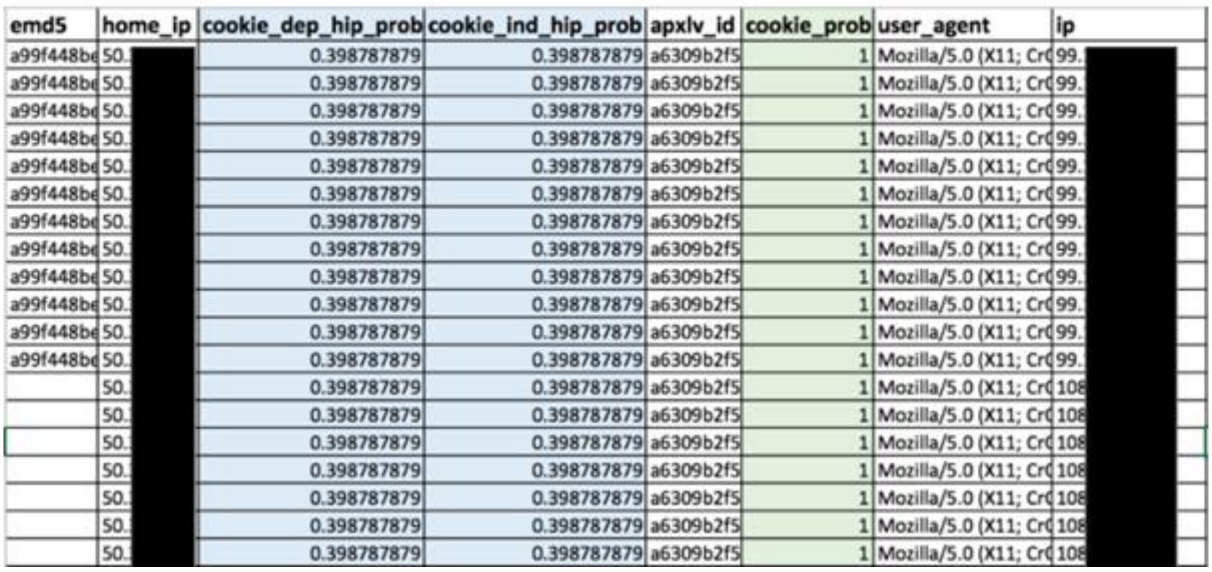


Figure 7. Device Fingerprinting Process

7.3 Results and Analysis

The underlying structure of the ad_requests and matches tables exclusively allowed removing and adding by partitions, not by entries. The data scientists at Cogo Labs designed this database for data browsing, not data manipulation. Under this constraint we implement the fingerprinting algorithm to work only for one cookie, given that we could neither store nor update individual entries inside this database. In addition, the execution time for our

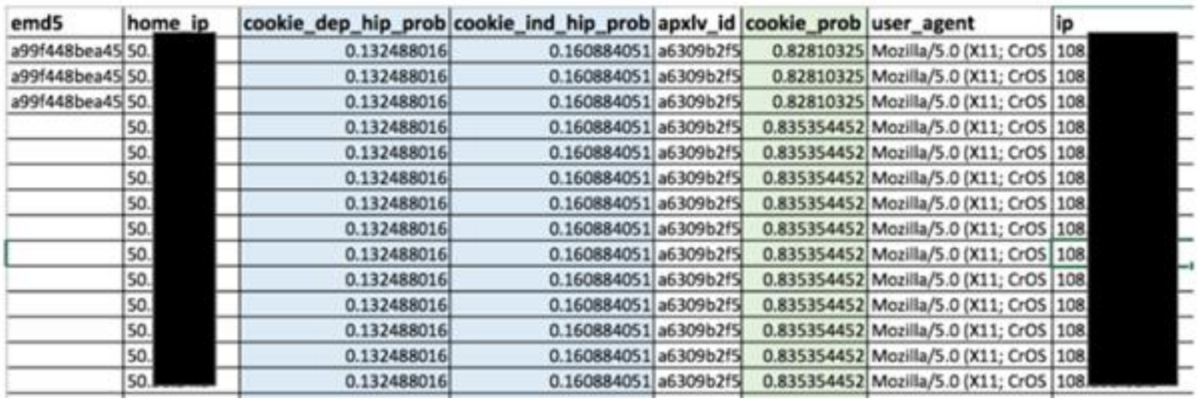
algorithm on one cookie took three minutes. An application of the algorithm to every cookie in the database would not return a result in a reasonable time for Cogo Labs.
Executing our Python implementation of the algorithm for our test cookie "a6309b2f5240ace44dcc8df60d56cf19" took 4 hours to perform all calculations. Figure 8 shows the results of these calculations. The execution time varies from cookie to cookie, thus 4 hours is not indicative of every cookie. For this test run, we set all accuracy coefficients to be 1. The emd5 field showed whether the entry comes from the matches table or ad_requests table. If the emd5 field was null, this shows that the entry belonged to the ad_requests table. If the emd5 field held a non-null value, it belonged in the matches table. The Home IP for all entries shown in Figure 8 are the same because we joined entries from both tables on the Home IP field. Cookie_dep_hip_prob and cookie_ind_hip_prob show identical results because the coefficients for both calculations are equal to 1. Cookie_prob will always contain the value 1 because the sample size of one cookie equals our population of one cookie. From a mathematical perspective, this makes the product of the cookie probability formula equal to 0, in turn making cookie probability equal to 1.



emd5	home_ip	cookie_dep_hip_prob	cookie_ind_hip_prob	apxlv_id	cookie_prob	user_agent	ip
a99f448b	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	99.[redacted]
a99f448b	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	99.[redacted]
a99f448b	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	99.[redacted]
a99f448b	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	99.[redacted]
a99f448b	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	99.[redacted]
a99f448b	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	99.[redacted]
a99f448b	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	99.[redacted]
a99f448b	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	99.[redacted]
a99f448b	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	99.[redacted]
a99f448b	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	99.[redacted]
a99f448b	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	99.[redacted]
a99f448b	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	99.[redacted]
	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	108[redacted]
	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	108[redacted]
	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	108[redacted]
	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	108[redacted]
	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	108[redacted]
	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	108[redacted]
	50.[redacted]	0.398787879	0.398787879	a6309b2f5	1	Mozilla/5.0 (X11; Cr	108[redacted]

Figure 8. Device fingerprinting implementation results with coefficients equal to 1.

The results shown above were different from our hypothesis. Cookie probabilities should never be 1 given the impossibility of a certain match between a cookie and a user. The values of the cookie_dep_hip_prob and cookie_ind_hip_prob fields between each record should be different. The reason behind these abnormalities derived from the values of our coefficients. We decided to try with a different set of coefficients as a possible remedy. We used probability coefficients calculated from the Microsoft paper [13] for a second test run. We set C=0.8235, U=0.6201, I=0.4899. Figure 9 displays the results of the second test run. Cookie_dep_hip_prob and cookie_ind_hip_prob have different values and cookie_prob does not keep the value of 1. These results reflect our hypothesis.



emd5	home_ip	cookie_dep_hip_prob	cookie_ind_hip_prob	apxlv_id	cookie_prob	user_agent	ip
a99f448bea45	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.82810325	Mozilla/5.0 (X11; CrOS	108[redacted]
a99f448bea45	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.82810325	Mozilla/5.0 (X11; CrOS	108[redacted]
a99f448bea45	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.82810325	Mozilla/5.0 (X11; CrOS	108[redacted]
	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.835354452	Mozilla/5.0 (X11; CrOS	108[redacted]
	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.835354452	Mozilla/5.0 (X11; CrOS	108[redacted]
	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.835354452	Mozilla/5.0 (X11; CrOS	108[redacted]
	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.835354452	Mozilla/5.0 (X11; CrOS	108[redacted]
	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.835354452	Mozilla/5.0 (X11; CrOS	108[redacted]
	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.835354452	Mozilla/5.0 (X11; CrOS	108[redacted]
	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.835354452	Mozilla/5.0 (X11; CrOS	108[redacted]
	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.835354452	Mozilla/5.0 (X11; CrOS	108[redacted]
	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.835354452	Mozilla/5.0 (X11; CrOS	108[redacted]
	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.835354452	Mozilla/5.0 (X11; CrOS	108[redacted]
	50.[redacted]	0.132488016	0.160884051	a6309b2f5	0.835354452	Mozilla/5.0 (X11; CrOS	108[redacted]

Figure 9. Device fingerprinting implementation results using Microsoft paper coefficients.

7.4 Recommendations

We suggested three recommendations for Cogo Labs. We first recommended that Cogo Labs revised the table partition in the Matches database. Particularly, Cogo Labs should have partitioned the matches table by cookie suffix in conjunction with the date partition. This partition structure would mimic the partition structure in ad_requests. The addition of these partitions would improve the execution time of the Python implementation down to a projected 5 minutes. Next, we recommended that Cogo Labs considers calculating their own accuracy coefficients used in the Home IP probability and cookie probability calculations. The Microsoft Research coefficients may not be entirely correct for Cogo Labs' case. Microsoft evaluated their coefficients based on around 100 million entries in their Outlook Database. Cogo Labs had trillions of entries of more detailed data, hence the need to tailor these coefficients for Cogo Labs' situation. Finally, we recommended a full implementation the fingerprinting algorithm, as shown in Appendix F. The Python implementation made by the MQP team works around the current partition issue in the database. Thus, it worked for one cookie and not all cookies in tandem. Fully implementing the algorithm would successfully perform the calculations for all cookies in the database.

8. Concluding Remarks

Cogo Labs and Vestigo used their data in a niche model applied at only a handful of companies. The employees of both companies eased a rapid work pipeline that capitalized on the strengths of each team member. Such a frictionless, unstructured environment proved beneficial to the working environment. This project displayed a full picture of the investment market in practical application. Furthermore, Cogo's integration of big data with investment served as a microcosm for the multifaceted future of modern fintech. With big data analytics just emerging in today's world as a cutting-edge technology, it had piqued our interest into learning more about the topic. To that end, two of our team members will pursue Data Science in an academic context. As of the writing of this paper Andrew Aberdale was applying for an MS in Data Science, and Benjamin Sarkis was applying for a Data Science minor to append to his BS in Computer Science. A parallel phenomenon occurred with our sponsor in the late 1980's. Ian Sheridan shared with us during this project how he used to be a "gopher" in the New York Stock Exchange, and how he was there for the first 10,000 automated transactions on the Exchange floor. He reflected on that experience as bringing perspective to what he had done, and what Cogo Labs had done. It is a hope of the undergraduate team members that the work of this project will become common place in the future.

Bibliography

All bibliographic citations are based off the IEEE standard.

(1) Ian Sheridan. “Ian Sheridan.” Internet: https://www.linkedin.com/in/iansheridan1/, 2017. [Nov. 6, 2017]

(2) Mark Casady. “Mark Casady.” Internet: https://www.linkedin.com/in/mark-casady-5a662b8/, 2017. [Nov. 7, 2017]

(3) Michael Nugent. “Michael Nugent.” Internet: https://www.linkedin.com/in/minugent/, 2017. [Nov. 6, 2017]

(4) David Blundin. “David Blundin.” Internet: https://www.linkedin.com/in/david-blundin-11220a/, 2017. [Nov. 6, 2017]

(5) Mark Casady, Spoken Word

(6) “The due diligence process in venture capital,” *MaRS*, 06-Dec-2013. [Online]. Available: https://www.marsdd.com/mars-library/the-due-diligence-process-in-venture-capital/. [15-Dec-2017].

(7) R. Amit, J. Brander and C. Zott, "Why do venture capital firms exist? theory and canadian evidence," *Journal of Business Venturing,* vol. 13, *(6),* pp. 441-466, 1998.

(8) “Riskalyze Inc.,” *Riskalyze Inc. - Other Technology- Lee FormDs.com - SEC filings of fundraisings and investments in hedge funds, private equity firms, startups, and growing companies*. Internet: http://www.formds.com/issuers/riskalyze-inc, 2017. [Dec. 1, 2017]

(9) Crunchbase, “Riskalyze.” Internet: https://www.crunchbase.com/organization/riskalyze. [Dec. 1, 2017]

(10) Ascensus. “The Largest Independent Retirement Savings and College Savings Provider.” Internet: https://www2.ascensus.com/. 2017. [Dec. 12, 2017].

(11) Horace Mann. “In service to educators since 1945.” Internet: https://www.horacemann.com/. 2017. [Dec. 12, 2017].

(12) InTeahouse. “A Global Network of Innovators and International Investors.” Internet: https://inteahouse.com/. 2017. [Dec. 12, 2017].

(13) VirtualCove. “Immersive Data Visualization.” Internet: http://www.virtualcove.com/. 2017. [Dec. 12, 2017].

(14) Mira Wilczek. “Mira Wilczek.” Internet:
https://www.linkedin.com/in/mira-wilczek-6558052/, 2017. [Dec. 5, 2017]

(15) Robert Fisher. “Robert Fisher.” Internet:
https://www.linkedin.com/in/robert-fisher-49463316/, 2017. [Dec. 5, 2017]

(16) Dennis Keohane. “CarGurus follows the plan...the TripAdvisor plan.” Internet:
https://utterlybiased.com/2017/04/07/cargurus-follows-the-plan-the-tripadvisor-plan/, April 7, 2017. [Nov. 15, 2017]

(17) R. Rivest, “The MD5 Message-Digest Algorithm,” Internet:
https://www.ietf.org/rfc/rfc1321.txt, April 1992. [Nov. 15, 2017]

(18) Armin Ronacher, “Flask, web development, one drop at a time.” Internet:
http://flask.pocoo.org/, 2017. [Oct 30, 2017].

(19) Docker Docks. “Select a Storage Driver.” Internet:
https://docs.docker.com/engine/userguide/storagedriver/selectadriver/. 2017. [Dec. 12, 2017].

(20) Tadayoshi Kohno, Andre Broido, K. C. Claffy. “Remote physical device fingerprinting.” *IEEE Transactions on Dependable and Secure Computing*, vol. 2, pp. 93-108, June 20, 2005. http://ieeexplore.ieee.org/abstract/document/1453529/?reload=true

(21) P. Eckersley, "How Unique Is Your Browser?" in Proceedings of the 10th Privacy Enhancing Technologies Symposium (PETS), 2010.
https://panopticlick.eff.org/static/browser-uniqueness.pdf

(22) Ting-Fang Yen, Yinglian Xie, Fang Yu, Roger Peng Yu, Martin Abadi. “Host Fingerprinting and Tracking on the Web: Privacy and Security Implications.”
Microsoft,
February 2012.
https://www.microsoft.com/en-us/research/wp-content/uploads/2012/02/ndss2012.pdf

(23) Jaumin Ajdari, Nehat Mustafa, Xhemal Zenuni, Bujar Raufi, Florije Ismaili. "Impact of table partitioning on the query execution performance." *International Journal of Computer Science Issues*, Vol. 13, pp. 52 - 58, July 2016. [Dec. 4, 2017]
http://ijcsi.org/articles/Impact-of-table-partitioning-on-the-query-execution-performance.php

(24) Survey Monkey, "Sample Size Calculator." Internet: https://www.surveymonkey.com/mp/sample-size-calculator/, 2017. [Dec. 4, 2017].

(25) Investopedia. "Venture Captial." Internet: http://www.investopedia.com/terms/v/venturecapital.asp, 2017. [Oct. 28, 2017].

(26) Investopedia. "Fintech." Internet: http://www.investopedia.com/terms/f/fintech.asp, 2017. [Oct. 28, 2017].

(27) Wendy W. Moe, Peter S. Fader. "Capturing Evolving Visit Behavior In Clickstream Data." *Journal of Interactive Marketing*, vol. 18, pp. 5 - 19, Winter 2004. http://mmdcommunications.com/pdf/Monitoring%20web%20traffic.pdf

(28) Don. "RE: Question on security issues of when2meet's password system." Personal E-mail (Dec. 4, 2017).

(29) Russom, P. (2011). Big Data Analytics. TDWI Best Practices Report, pp.6-9 https://vivomente.com/wp-content/uploads/2016/04/big-data-analytics-white-paper.pdf [Jan. 17, 2018].

(30) LifeYield. "Make more. Keep more." Internet: https://www.lifeyield.com/main/, 2017. [Jan. 17, 2018]

(31) Netcapital. "We connect entrepeneurs and investors to help businesses grow." Internet: https://netcapital.com/, July 7, 2017. [Jan. 17, 2018]

(32) Vestmark. "Unified Wealth Management." Internet: http://www.vestmark.com/, 2018. [Jan. 17, 2018]

(33) Investopedia, “The Stages in Venture Capital Investing,” Investopedia. https://www.investopedia.com/exam-guide/cfa-level-1/alternative-investments/venture-capital-investing-stages.asp [Jan. 17, 2018]

(34) S. Delventhal, “Series A, B, C Funding: What It All Means and How It Works,” Investopedia. https://www.investopedia.com/articles/personal-finance/102015/series-b-c-fun ding-what-it-all-means-and-how-it-works.asp [Jan. 17, 2018]

Appendices

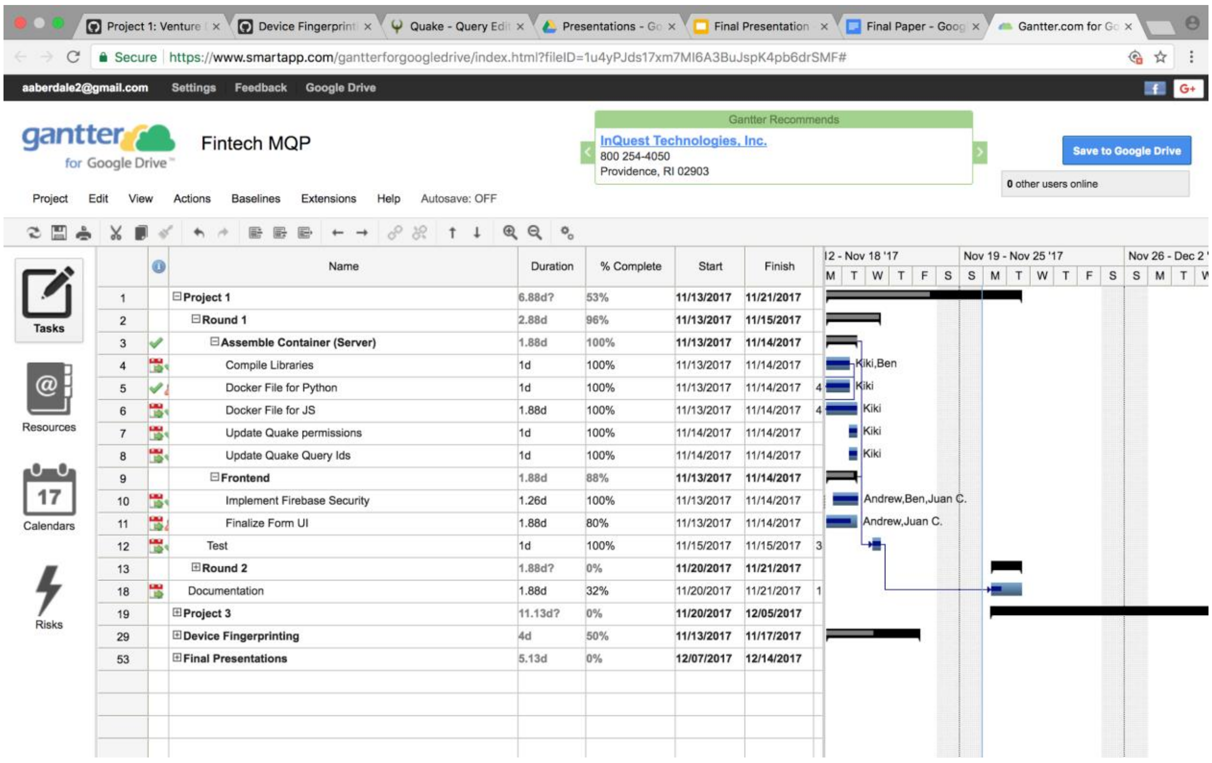
Appendix A: Terminology

Venture Capital ("VC"): "Venture Capital is financing that investors provide to startup companies and small businesses that are believed to have long term growth potential." [25]

Financial Technology ("Fintech"): "Fintech is a portmanteau of financial technology that describes an emerging financial services sector in the 21st century. Originally, the term applied to technology applied to the backend of established consumer and trade financial institutions. Since the end of the first decade of the 21st century, the term has expanded to include any technological innovation in the financial sector, including innovations in financial literacy and education, retail banking, investment and even cryptocurrencies like bitcoin." [26]

Clickstream: "A series of mouse clicks made while using a website or in linking to multiple websites." [27]

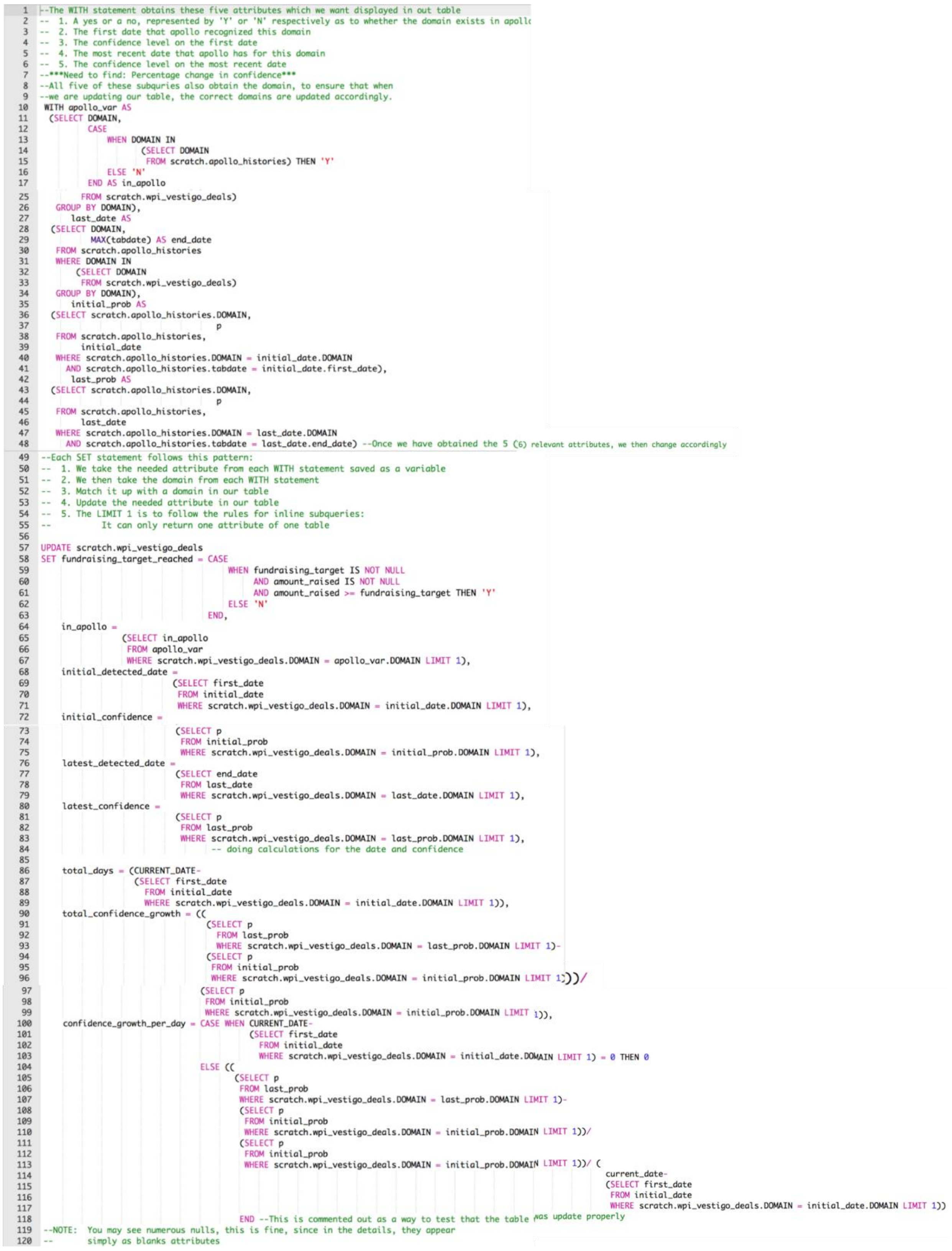
Appendix B: Gantt


Fintech MQP

		Name	Duration	% Complete	Start	Finish
1		⊟Project 1	6.88d?	53%	11/13/2017	11/21/2017
2		⊟Round 1	2.88d	96%	11/13/2017	11/15/2017
3	✓	⊟Assemble Container (Server)	1.88d	100%	11/13/2017	11/14/2017
4		Compile Libraries	1d	100%	11/13/2017	11/14/2017
5	✓	Docker File for Python	1d	100%	11/13/2017	11/14/2017
6		Docker File for JS	1.88d	100%	11/13/2017	11/14/2017
7		Update Quake permissions	1d	100%	11/14/2017	11/14/2017
8		Update Quake Query Ids	1d	100%	11/14/2017	11/14/2017
9		⊟Frontend	1.88d	88%	11/13/2017	11/14/2017
10		Implement Firebase Security	1.26d	100%	11/13/2017	11/14/2017
11		Finalize Form UI	1.88d	80%	11/13/2017	11/14/2017
12		Test	1d	100%	11/15/2017	11/15/2017
13		⊞Round 2	1.88d?	0%	11/20/2017	11/21/2017
18		Documentation	1.88d	32%	11/20/2017	11/21/2017
19		⊞Project 3	11.13d?	0%	11/20/2017	12/05/2017
29		⊞Device Fingerprinting	4d	50%	11/13/2017	11/17/2017
53		⊞Final Presentations	5.13d	0%	12/07/2017	12/14/2017

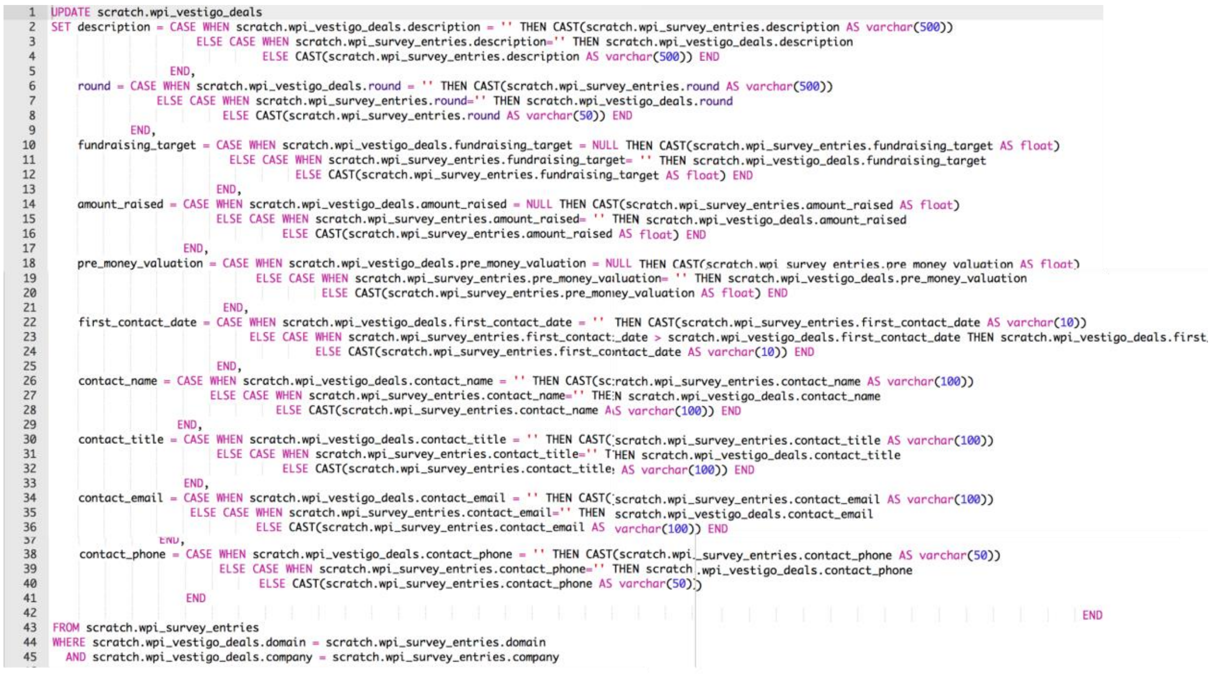
Appendix C: SQL queries

1. wpi_update_from_apollo query



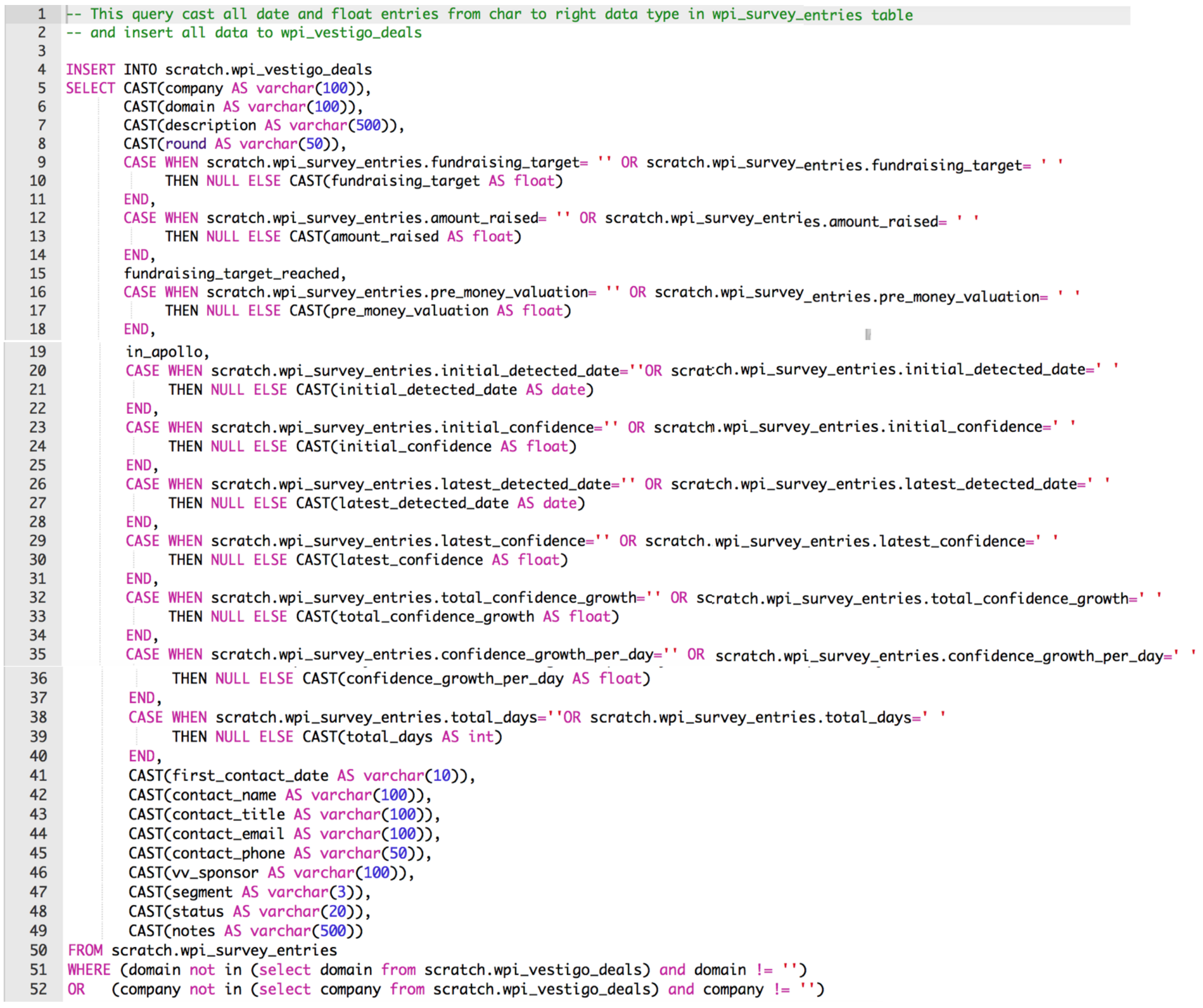
```sql
--The WITH statement obtains these five attributes which we want displayed in out table
--  1. A yes or a no, represented by 'Y' or 'N' respectively as to whether the domain exists in apoll
--  2. The first date that apollo recognized this domain
--  3. The confidence level on the first date
--  4. The most recent date that apollo has for this domain
--  5. The confidence level on the most recent date
--***Need to find: Percentage change in confidence***
--All five of these subquries also obtain the domain, to ensure that when
--we are updating our table, the correct domains are updated accordingly.
 WITH apollo_var AS
  (SELECT DOMAIN,
          CASE
              WHEN DOMAIN IN
                     (SELECT DOMAIN
                      FROM scratch.apollo_histories) THEN 'Y'
              ELSE 'N'
          END AS in_apollo
         FROM scratch.wpi_vestigo_deals)
   GROUP BY DOMAIN),
      last_date AS
   (SELECT DOMAIN,
           MAX(tabdate) AS end_date
    FROM scratch.apollo_histories
    WHERE DOMAIN IN
        (SELECT DOMAIN
         FROM scratch.wpi_vestigo_deals)
    GROUP BY DOMAIN),
      initial_prob AS
   (SELECT scratch.apollo_histories.DOMAIN,
                                    p
    FROM scratch.apollo_histories,
         initial_date
    WHERE scratch.apollo_histories.DOMAIN = initial_date.DOMAIN
      AND scratch.apollo_histories.tabdate = initial_date.first_date),
      last_prob AS
   (SELECT scratch.apollo_histories.DOMAIN,
                                    p
    FROM scratch.apollo_histories,
         last_date
    WHERE scratch.apollo_histories.DOMAIN = last_date.DOMAIN
      AND scratch.apollo_histories.tabdate = last_date.end_date) --Once we have obtained the 5 (6) relevant attributes, we then change accordingly
--Each SET statement follows this pattern:
--  1. We take the needed attribute from each WITH statement saved as a variable
--  2. We then take the domain from each WITH statement
--  3. Match it up with a domain in our table
--  4. Update the needed attribute in our table
--  5. The LIMIT 1 is to follow the rules for inline subqueries:
--          It can only return one attribute of one table

UPDATE scratch.wpi_vestigo_deals
SET fundraising_target_reached = CASE
                                     WHEN fundraising_target IS NOT NULL
                                          AND amount_raised IS NOT NULL
                                          AND amount_raised >= fundraising_target THEN 'Y'
                                     ELSE 'N'
                                 END,
    in_apollo =
               (SELECT in_apollo
                FROM apollo_var
                WHERE scratch.wpi_vestigo_deals.DOMAIN = apollo_var.DOMAIN LIMIT 1),
    initial_detected_date =
                          (SELECT first_date
                           FROM initial_date
                           WHERE scratch.wpi_vestigo_deals.DOMAIN = initial_date.DOMAIN LIMIT 1),
    initial_confidence =
                          (SELECT p
                           FROM initial_prob
                           WHERE scratch.wpi_vestigo_deals.DOMAIN = initial_prob.DOMAIN LIMIT 1),
    latest_detected_date =
                          (SELECT end_date
                           FROM last_date
                           WHERE scratch.wpi_vestigo_deals.DOMAIN = last_date.DOMAIN LIMIT 1),
    latest_confidence =
                          (SELECT p
                           FROM last_prob
                           WHERE scratch.wpi_vestigo_deals.DOMAIN = last_prob.DOMAIN LIMIT 1),
                                -- doing calculations for the date and confidence

    total_days = (CURRENT_DATE-
                   (SELECT first_date
                     FROM initial_date
                     WHERE scratch.wpi_vestigo_deals.DOMAIN = initial_date.DOMAIN LIMIT 1)),
    total_confidence_growth = ((
                                (SELECT p
                                  FROM last_prob
                                  WHERE scratch.wpi_vestigo_deals.DOMAIN = last_prob.DOMAIN LIMIT 1)-
                                (SELECT p
                                 FROM initial_prob
                                 WHERE scratch.wpi_vestigo_deals.DOMAIN = initial_prob.DOMAIN LIMIT 1))/
                               (SELECT p
                                FROM initial_prob
                                WHERE scratch.wpi_vestigo_deals.DOMAIN = initial_prob.DOMAIN LIMIT 1)),
    confidence_growth_per_day = CASE WHEN CURRENT_DATE-
                                         (SELECT first_date
                                           FROM initial_date
                                           WHERE scratch.wpi_vestigo_deals.DOMAIN = initial_date.DOMAIN LIMIT 1) = 0 THEN 0
                               ELSE ((
                                     (SELECT p
                                      FROM last_prob
                                      WHERE scratch.wpi_vestigo_deals.DOMAIN = last_prob.DOMAIN LIMIT 1)-
                                      (SELECT p
                                       FROM initial_prob
                                       WHERE scratch.wpi_vestigo_deals.DOMAIN = initial_prob.DOMAIN LIMIT 1))/
                                      (SELECT p
                                       FROM initial_prob
                                       WHERE scratch.wpi_vestigo_deals.DOMAIN = initial_prob.DOMAIN LIMIT 1))/ (
                                                                                                              current_date-
                                                                                                              (SELECT first_date
                                                                                                               FROM initial_date
                                                                                                               WHERE scratch.wpi_vestigo_deals.DOMAIN = initial_date.DOMAIN LIMIT 1))
                                      END --This is commented out as a way to test that the table was update properly
--NOTE:  You may see numerous nulls, this is fine, since in the details, they appear
--       simply as blanks attributes
```

2. wpi_update_existing_deals query



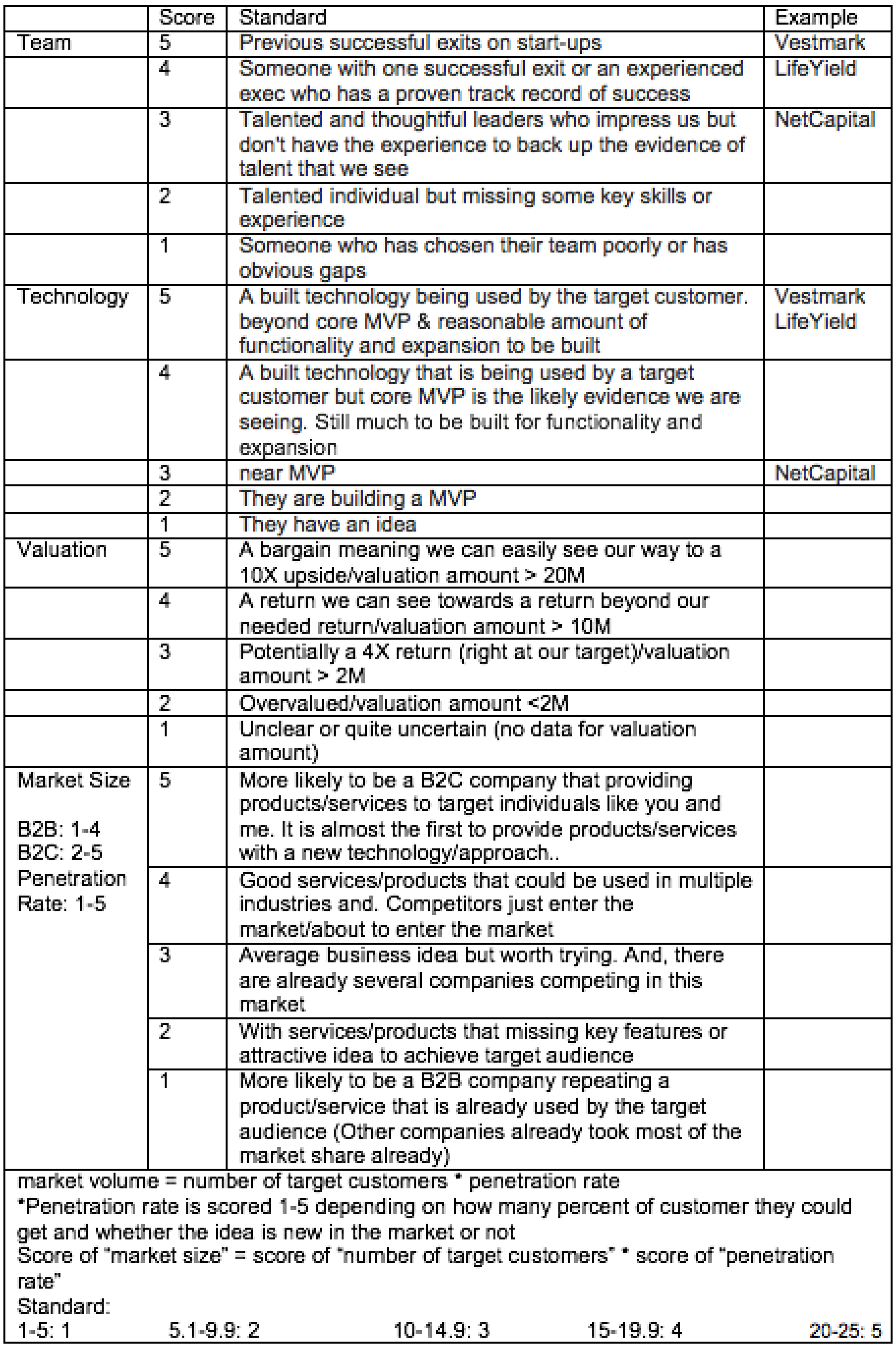
```
UPDATE scratch.wpi_vestigo_deals
SET description = CASE WHEN scratch.wpi_vestigo_deals.description = '' THEN CAST(scratch.wpi_survey_entries.description AS varchar(500))
                      ELSE CASE WHEN scratch.wpi_survey_entries.description='' THEN scratch.wpi_vestigo_deals.description
                                ELSE CAST(scratch.wpi_survey_entries.description AS varchar(500)) END
                  END,
    round = CASE WHEN scratch.wpi_vestigo_deals.round = '' THEN CAST(scratch.wpi_survey_entries.round AS varchar(500))
                ELSE CASE WHEN scratch.wpi_survey_entries.round='' THEN scratch.wpi_vestigo_deals.round
                          ELSE CAST(scratch.wpi_survey_entries.round AS varchar(50)) END
            END,
    fundraising_target = CASE WHEN scratch.wpi_vestigo_deals.fundraising_target = NULL THEN CAST(scratch.wpi_survey_entries.fundraising_target AS float)
                           ELSE CASE WHEN scratch.wpi_survey_entries.fundraising_target= '' THEN scratch.wpi_vestigo_deals.fundraising_target
                                    ELSE CAST(scratch.wpi_survey_entries.fundraising_target AS float) END
                         END,
    amount_raised = CASE WHEN scratch.wpi_vestigo_deals.amount_raised = NULL THEN CAST(scratch.wpi_survey_entries.amount_raised AS float)
                         ELSE CASE WHEN scratch.wpi_survey_entries.amount_raised= '' THEN scratch.wpi_vestigo_deals.amount_raised
                                   ELSE CAST(scratch.wpi_survey_entries.amount_raised AS float) END
                    END,
    pre_money_valuation = CASE WHEN scratch.wpi_vestigo_deals.pre_money_valuation = NULL THEN CAST(scratch.wpi_survey_entries.pre_money_valuation AS float)
                               ELSE CASE WHEN scratch.wpi_survey_entries.pre_money_valuation= '' THEN scratch.wpi_vestigo_deals.pre_money_valuation
                                        ELSE CAST(scratch.wpi_survey_entries.pre_money_valuation AS float) END
                          END,
    first_contact_date = CASE WHEN scratch.wpi_vestigo_deals.first_contact_date = ''  THEN CAST(scratch.wpi_survey_entries.first_contact_date AS varchar(10))
                              ELSE CASE WHEN scratch.wpi_survey_entries.first_contact_date > scratch.wpi_vestigo_deals.first_contact_date THEN scratch.wpi_vestigo_deals.first
                                       ELSE CAST(scratch.wpi_survey_entries.first_contact_date AS varchar(10)) END
                         END,
    contact_name = CASE WHEN scratch.wpi_vestigo_deals.contact_name = '' THEN CAST(scratch.wpi_survey_entries.contact_name AS varchar(100))
                        ELSE CASE WHEN scratch.wpi_survey_entries.contact_name='' THEN scratch.wpi_vestigo_deals.contact_name
                                  ELSE CAST(scratch.wpi_survey_entries.contact_name AS varchar(100)) END
                   END,
    contact_title = CASE WHEN scratch.wpi_vestigo_deals.contact_title = '' THEN CAST(scratch.wpi_survey_entries.contact_title AS varchar(100))
                         ELSE CASE WHEN scratch.wpi_survey_entries.contact_title='' THEN scratch.wpi_vestigo_deals.contact_title
                                   ELSE CAST(scratch.wpi_survey_entries.contact_title AS varchar(100)) END
                    END,
    contact_email = CASE WHEN scratch.wpi_vestigo_deals.contact_email = '' THEN CAST(scratch.wpi_survey_entries.contact_email AS varchar(100))
                     ELSE CASE WHEN scratch.wpi_survey_entries.contact_email='' THEN scratch.wpi_vestigo_deals.contact_email
                               ELSE CAST(scratch.wpi_survey_entries.contact_email AS varchar(100)) END
                END,
    contact_phone = CASE WHEN scratch.wpi_vestigo_deals.contact_phone = '' THEN CAST(scratch.wpi_survey_entries.contact_phone AS varchar(50))
                         ELSE CASE WHEN scratch.wpi_survey_entries.contact_phone='' THEN scratch.wpi_vestigo_deals.contact_phone
                               ELSE CAST(scratch.wpi_survey_entries.contact_phone AS varchar(50))
                    END
                                                                                                                                                           END
FROM scratch.wpi_survey_entries
WHERE scratch.wpi_vestigo_deals.domain = scratch.wpi_survey_entries.domain
  AND scratch.wpi_vestigo_deals.company = scratch.wpi_survey_entries.company
```

3. wpi_insert_new_deals query



```sql
-- This query cast all date and float entries from char to right data type in wpi_survey_entries table
-- and insert all data to wpi_vestigo_deals

INSERT INTO scratch.wpi_vestigo_deals
SELECT CAST(company AS varchar(100)),
       CAST(domain AS varchar(100)),
       CAST(description AS varchar(500)),
       CAST(round AS varchar(50)),
       CASE WHEN scratch.wpi_survey_entries.fundraising_target= '' OR scratch.wpi_survey_entries.fundraising_target= ' '
            THEN NULL ELSE CAST(fundraising_target AS float)
       END,
       CASE WHEN scratch.wpi_survey_entries.amount_raised= '' OR scratch.wpi_survey_entries.amount_raised= ' '
            THEN NULL ELSE CAST(amount_raised AS float)
       END,
       fundraising_target_reached,
       CASE WHEN scratch.wpi_survey_entries.pre_money_valuation= '' OR scratch.wpi_survey_entries.pre_money_valuation= ' '
            THEN NULL ELSE CAST(pre_money_valuation AS float)
       END,
       in_apollo,
       CASE WHEN scratch.wpi_survey_entries.initial_detected_date=''OR scratch.wpi_survey_entries.initial_detected_date=' '
            THEN NULL ELSE CAST(initial_detected_date AS date)
       END,
       CASE WHEN scratch.wpi_survey_entries.initial_confidence='' OR scratch.wpi_survey_entries.initial_confidence=' '
            THEN NULL ELSE CAST(initial_confidence AS float)
       END,
       CASE WHEN scratch.wpi_survey_entries.latest_detected_date='' OR scratch.wpi_survey_entries.latest_detected_date=' '
            THEN NULL ELSE CAST(latest_detected_date AS date)
       END,
       CASE WHEN scratch.wpi_survey_entries.latest_confidence='' OR scratch.wpi_survey_entries.latest_confidence=' '
            THEN NULL ELSE CAST(latest_confidence AS float)
       END,
       CASE WHEN scratch.wpi_survey_entries.total_confidence_growth='' OR scratch.wpi_survey_entries.total_confidence_growth=' '
            THEN NULL ELSE CAST(total_confidence_growth AS float)
       END,
       CASE WHEN scratch.wpi_survey_entries.confidence_growth_per_day='' OR scratch.wpi_survey_entries.confidence_growth_per_day=' '
            THEN NULL ELSE CAST(confidence_growth_per_day AS float)
       END,
       CASE WHEN scratch.wpi_survey_entries.total_days=''OR scratch.wpi_survey_entries.total_days=' '
            THEN NULL ELSE CAST(total_days AS int)
       END,
       CAST(first_contact_date AS varchar(10)),
       CAST(contact_name AS varchar(100)),
       CAST(contact_title AS varchar(100)),
       CAST(contact_email AS varchar(100)),
       CAST(contact_phone AS varchar(50)),
       CAST(vv_sponsor AS varchar(100)),
       CAST(segment AS varchar(3)),
       CAST(status AS varchar(20)),
       CAST(notes AS varchar(500))
FROM scratch.wpi_survey_entries
WHERE (domain not in (select domain from scratch.wpi_vestigo_deals) and domain != '')
OR    (company not in (select company from scratch.wpi_vestigo_deals) and company != '')
```

Appendix D: 4-point Scoring System



	Score	Standard	Example
Team	5	Previous successful exits on start-ups	Vestmark
	4	Someone with one successful exit or an experienced exec who has a proven track record of success	LifeYield
	3	Talented and thoughtful leaders who impress us but don't have the experience to back up the evidence of talent that we see	NetCapital
	2	Talented individual but missing some key skills or experience	
	1	Someone who has chosen their team poorly or has obvious gaps	
Technology	5	A built technology being used by the target customer. beyond core MVP & reasonable amount of functionality and expansion to be built	Vestmark LifeYield
	4	A built technology that is being used by a target customer but core MVP is the likely evidence we are seeing. Still much to be built for functionality and expansion	
	3	near MVP	NetCapital
	2	They are building a MVP	
	1	They have an idea	
Valuation	5	A bargain meaning we can easily see our way to a 10X upside/valuation amount > 20M	
	4	A return we can see towards a return beyond our needed return/valuation amount > 10M	
	3	Potentially a 4X return (right at our target)/valuation amount > 2M	
	2	Overvalued/valuation amount <2M	
	1	Unclear or quite uncertain (no data for valuation amount)	
Market Size B2B: 1-4 B2C: 2-5 Penetration Rate: 1-5	5	More likely to be a B2C company that providing products/services to target individuals like you and me. It is almost the first to provide products/services with a new technology/approach..	
	4	Good services/products that could be used in multiple industries and. Competitors just enter the market/about to enter the market	
	3	Average business idea but worth trying. And, there are already several companies competing in this market	
	2	With services/products that missing key features or attractive idea to achieve target audience	
	1	More likely to be a B2B company repeating a product/service that is already used by the target audience (Other companies already took most of the market share already)	

market volume = number of target customers * penetration rate
*Penetration rate is scored 1-5 depending on how many percent of customer they could get and whether the idea is new in the market or not
Score of "market size" = score of "number of target customers" * score of "penetration rate"
Standard:
1-5: 1 5.1-9.9: 2 10-14.9: 3 15-19.9: 4 20-25: 5

Appendix E: Rejected Decisions

Throughout the course of the project, we tried and discussed many options that were ultimately scuttled. Each subsection follows the format:

[X] Rejected Decision Name:

Description

The “[X]” is referenced in the paper next to the decision we ultimately took.

----

[1] Date form:

We tried using the format “mm/dd/yyyy” initially.

[2] JOIN statements in update query:

An initial iteration of this query involved the use of SQL JOIN statements. JOIN statements could not do calculations in the order we wanted them to. WITH statements allowed for this flexibility.

[3] Dealing with new companies and existing companies in the update query:

We considered creating an entry for each submission and have a query that would act as a garbage collector to remove ‘duplicate’ entries. A duplicate entry would be one that has the same company name and domain (Table 1). However, we found that this design could potentially result in data loss if the two entries have the same company and domain name, but otherwise differ.

Another option that was considered was coding in a condition that would check if the entry already existed in the table before adding it in and update accordingly. However, this proved to be inefficient as well as difficult to design and did not integrate effectively into the architecture.

[4] Form Decision:

Our form underwent two key iterations over the course of the project. The sponsors and the graduate team gave the suggestion of Google Forms, which offers various script APIs that facilitate data conversion, form behavior, and security. The added benefit of an excel sheet tied to all form responses allows access to the functionality associated with Google Sheets. We wrote a script that automatically creates an updated .tsv file in the form’s directory whenever the sheet changed from a new form entry. From the JavaScript backend (Google Apps Script for this specific case), we attempted to send a post request to the Flask server we set up locally. However, through several trials, we discovered a partition between the form API and Flask that prevented Flask from receiving the post request. The script was running on Google’s server; a post to a “localhost” was directed to a host on the Google server instead of our local Flask server.

The first attempt at surmounting the partition began with moving the Python server to a different platform. Google Cloud Platform would host the server and thus integrate with the google scripts associated with the form. Flask could have an address reachable from the form. This method would work if we only used public libraries. Through further investigation, the proprietary library for the database we accessed could not be used outside of Cogo's interior network. The library required authentication to Cogo's network for proper use. Furthermore, the triggers we had set up in Google scripts could not be refactored to the python server; direct communication between Flask and the form could not occur. The library's reliance on Cogo eliminated the option of having the Python Flask hosted on the Google Cloud Platform. We decided to move away from Google APIs and the Google platform to create a form from scratch.

[5] Password/Hashing decision:

Earlier iterations of the password security option involved hashing. Our first hashing idea was inspired from the method employed by when2meet. However, much of when2meet's security lies in their unique hashing for each survey. Since we have one url that Vestigo wants to use, this security aspect is not available to us. The other aspect of the when2meet system is trust, which is not strong enough security in high-stakes environment Venture capitalism.[28]

The second hashing idea was one random hash per survey. It was the strongest security option, yet it would have required heavy refactoring before adding it. Our table would require an extra field as well as implementing hash generation functionality in JavaScript. The amount of security behind Cogo's network rendered this option not productive.

Finally, our sponsor offered to send out surveys in controlled batches. Within each batch, we could try hashing each survey. While this would require only as many hashes as there are surveys per batch, it would create an unneeded burden on the sponsors. They would have to manually search through each batch and a specific hash for a particular survey.

[6] On coding the .TSV into JSON:

The original iteration of passing a .TSV with encoded JSON involved simply passing the raw .TSV. However, there was no reliable content-type or mime-type for .TSVs, so we used JSON, as there are several standards associated with it.

[7] Functionality for sending emails:

The question of how we would implement email functionality required both solidifying which libraries we would use and where in the entire system the email would be sent from. We tried two options, both inside the frontend of the form. The first of these was react-html-email for structuring and sending emails. We valued the formatting feature of this module from node-package-modules, but it would require restructuring the whole application. A proper react-email could not be sent unless we had meteor.js integrated into the React.js application from the beginning.

After abandoning react-html-email, we attempted the use of nodemailer. Nodemailer worked very well with react and gmail, but could only be applied on the backend of a React app. Since our backend was in Python, nodemailer was immediately discarded. From here, we looked at backend python libraries and chose the decision specified in the paper.

[8] Confidence Growth Decision:

For the time interval when calculating confidence_growth_per_day, we first considered using the time interval from the first day the domain was detected by Apollo to the latest detection date. However, we noticed that some of the domains were not been detected for one year ago or only been detected for a few days, which made the confidence growth rate either outdated or abnormally high. In addition, this particular kind of data does not tell the full story because apollo does not guarantee picking the initial start date of the company, only the time when it is visible. We decided that this did not accurately represent the company's growth so we adjusted our confidence_growth_per_day calculation as depicted in the paper.

[9] Develop Home IP only based on emd5:

Since the Home IP can be a unique identifier for a user, we first developed based on emd5. But only 30% of our data in scratch.matches table have emd5 so that using emd5 will shrink our results. Therefore we decided to use apxlv_id which both of our tables have to make Home IP more completely covered the whole database.

[10] The order of calculating the cookie_dep_hip_prob

When calculating the cookie_dep_hip_prob, we first wanted to do it in our second phase. But it ends up with the same result if we calculate in first phase. Also, doing it in phase two will have more complicated calculation and take more time.

[11]Device Fingerprinting Indexing:

We considered indexing by a combination of cookie, user agent, and ip as opposed to just cookie. However, we then realized it would make our Home IP calculations more exclusive, which in turn would yield skewed calculations. We decided to just index on cookie to prevent damage to our calculations. However, should Cogo invest more time in this, it may be worthwhile to reconsider this option.

[12] Device Fingerprinting deliverable:
At the beginning of the device fingerprinting project, we were given a introduction packet by Mike Nugent and the grads. This packet outlined our projects and the expected deliverables. The primary deliverable for device fingerprinting was expected to be a system take could take two cookies and find the likelihood that they are related to each other. At that time we understood the process time would take a while, so we foresaw that the project would require preprocessing and storage. We ran the calculations for a choose function of pairs for 3.64 trillion (1*10^12) entries and found that to store all our calculated entries, it would take about 6.6 septillion (1*10^24) entries worth of additional space.

[13] User Agent String analysis by parts

Analyzing the user agent string by parts would allow us to know exactly how similar 2 user agent strings are. However, we cannot calculate the accuracy coefficients for each of the user agents string parts. Hence, we rejected this design and compared user agent strings as a whole, given that we had an accuracy coefficient for the whole user agent string.

[14] Device Fingerprinting final calculation:
After we had completed most of our algorithm, we focused on a long standing issue. Our system would take the highest probability, as opposed to the most frequent. We addressed this by minorly altering the code from a "MAX" SQL function to one based off of "COUNT(apxlv_id)"/"COUNT(*)". This focused the system to look for the most frequent probability, which is a more accurate indicator than looking for the maximum probability.

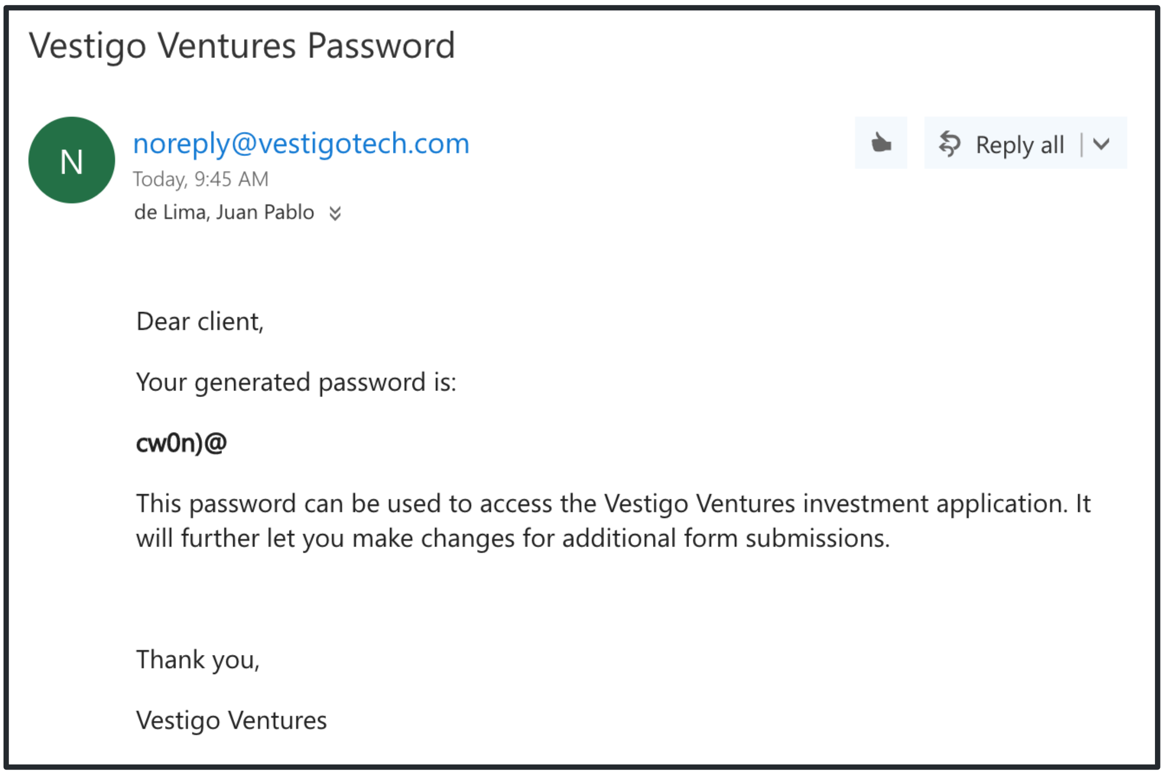
[15] Rename cookie_prob to home_ip_cookie_match_prob
The graduate team suggested renaming cookie_prob field but the name they suggested is far too long and complex. It is unnecessary and over-descriptive.

[16] Multiply cookie_prob by the hip_prob

Because the cookie_prob is based on the concept of Home IP, we should consider the accuracy of the Home IP. We provided Cogo two frame of reference for the Home IP probability and developed cookie_prob independent to the Home IP probability. So Cogo Labs can choose which frame they want to refer.

[17] Calculate cookie_prob for every entry
Calculating cookie probability for every entry can take a lot of time to do. Some of the entries in the database can have duplicates, so we decided to keep track of the results for each entry and refer to that result when the algorithm finds a duplicate entry.

Appendix F: Email Notifications


Vestigo Ventures Password

noreply@vestigotech.com
Today, 9:45 AM
de Lima, Juan Pablo

Dear client,

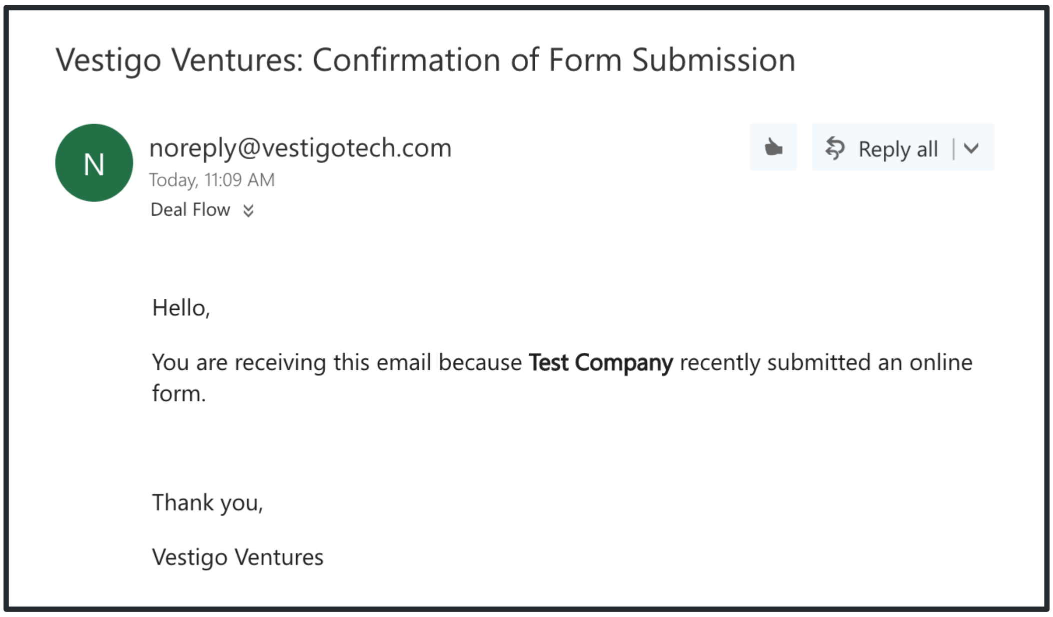
Your generated password is:

cw0n)@

This password can be used to access the Vestigo Ventures investment application. It will further let you make changes for additional form submissions.

Thank you,

Vestigo Ventures


Vestigo Ventures: Confirmation of Form Submission

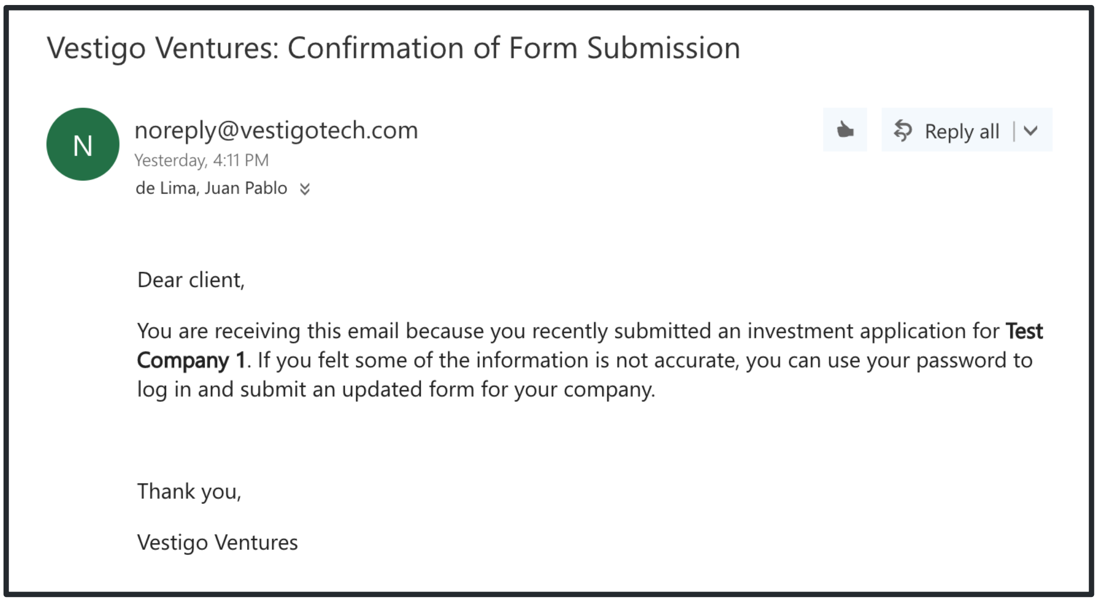
noreply@vestigotech.com
Today, 11:09 AM
Deal Flow

Hello,

You are receiving this email because **Test Company** recently submitted an online form.

Thank you,

Vestigo Ventures



Vestigo Ventures: Confirmation of Form Submission

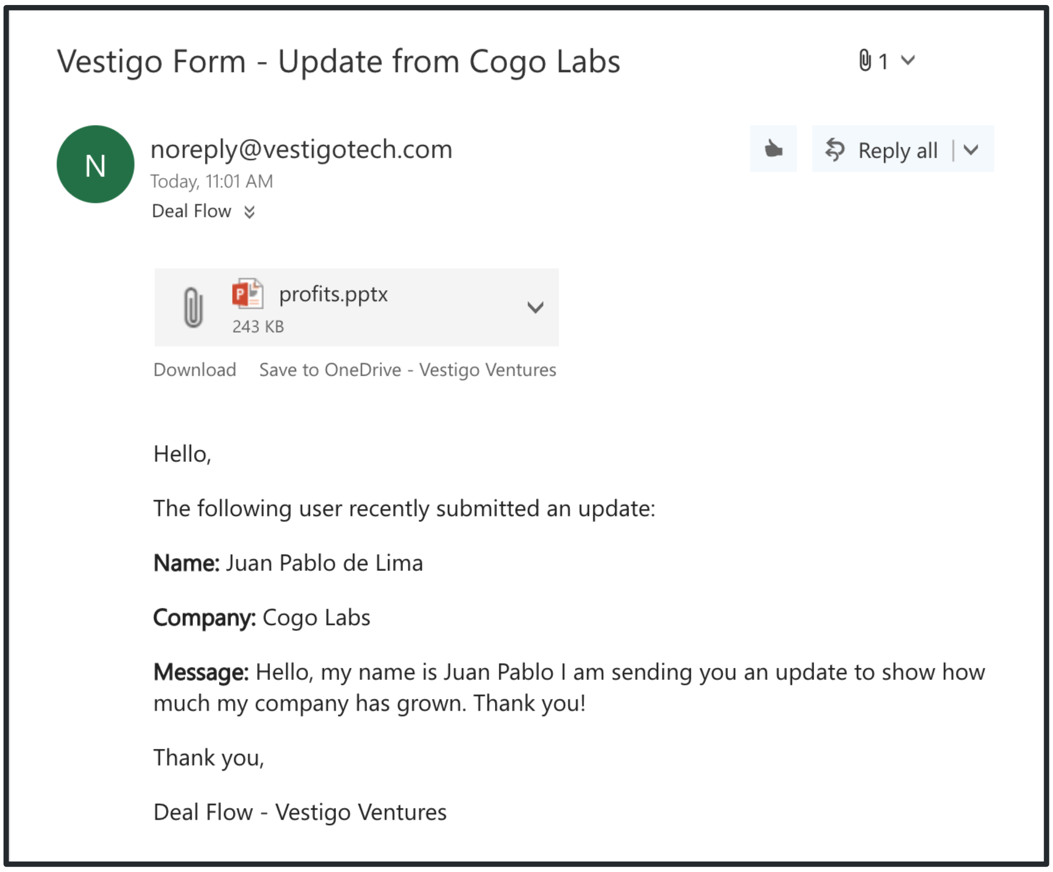
noreply@vestigotech.com
Yesterday, 4:11 PM
de Lima, Juan Pablo

Dear client,

You are receiving this email because you recently submitted an investment application for **Test Company 1**. If you felt some of the information is not accurate, you can use your password to log in and submit an updated form for your company.

Thank you,

Vestigo Ventures



Vestigo Form - Update from Cogo Labs

noreply@vestigotech.com
Today, 11:01 AM
Deal Flow

Hello,

The following user recently submitted an update:

Name: Juan Pablo de Lima

Company: Cogo Labs

Message: Hello, my name is Juan Pablo I am sending you an update to show how much my company has grown. Thank you!

Thank you,

Deal Flow - Vestigo Ventures

Appendix G: Device Fingerprinting Algorithm

Constants

We are assuming, we can always accurately identify someone based on the cookie. Therefore coefficients for calculations are below. These are in place incase Cogo decides there is a different associated percentage chance with accurately identifying a user based on a specific field. These variables allow for easy change.

Assumed accuracy (Currently in use):

C=1

U=1

I=1

Microsoft Paper (Not in use):

C=0.8235

U=0.6201

I=0.4899

STEP 1: Determine the HOME_IP and COOKIE_DEP_HIP_PROB

INPUT TABLES: match.matches, match.ad_requests

1)

a) Gather ALL rows from match.matches and match.ad_requests WHERE cookie = cookie.

i) Eg. cookieA has 20 rows from match.matches, and cookieA has 10 rows from match.ad_requests, then in total we'll take 30 rows.

ii) Eg. cookieB does not exist in match.matches, but cookieB has 40 rows from match.ad_requests. Then in total we will take in 40 rows.

b) Store these rows into the Alex DB table, scratch.wpi_device_fingerprinting [call this ALPHA].

2) Within ALPHA, calculate home_ip field. We define:

home_ip as "the most frequent IP address for a given cookie."

3) Within ALPHA calculate cookie_dep_hip_prob. Formula is the following:

$$\text{cookie\_dep\_hip\_prob} = \frac{(\text{Frequency count where the specific IP=HomeIP}) \times C^2 \times \text{I}}{\text{Total count of Cookie}}$$

4) Append all entries from ALPHA to BETA. (BETA will be partitioned by HOME_IP or some derivative of it)
5) Delete all entries in ALPHA
6) Do this repeatedly for each cookie instance from match.matches and match.ad_requests

STEP 2: Calculate COOKIE_IND_HIP_PROB

INPUT TABLE: BETA

1) Gather all rows from BETA table for a given home_ip, say Home_IP = 1. Say for eg within our BETA table we have 30 rows of cookieA with Home_IP=1 and 20 rows of cookieB with Home_IP=1.
a) Place these (eg. 50) rows into ALPHA.
b) Delete these rows from BETA (that means “delete this partition”).
2) Within ALPHA, calculate cookie_ind_hip_prob field. Formula is the following:

$$\text{cookie\_ind\_hip\_prob} = \frac{\text{(Frequency count where the specific IP=HomeIP)} \times \text{C} \times \text{I}}{\text{Total count of Home IP}}$$

STEP 3: Calculate COOKIE_PROB

1) Still within ALPHA, get the frequency count of Cookie, UA, and IP across the “home-ip rows”.

$$\text{Cookie\_Frequency} = \frac{\text{Frequency count where the specific Cookie=Cookie}}{\text{Total count of HomeIP}}$$

$$\text{UA\_Frequency} = \frac{\text{Frequency count where the specific UA=UA}}{\text{Total count of HomeIP}}$$

$$\text{IP\_Frequency} = \frac{\text{Frequency count where the specific IP=IP}}{\text{Total count of HomeIP}}$$

2) Calculate the “probability coefficients in home-ip world”
a) Formula to use:
i) X = C * Cookie_Frequency
ii) Y = U * UA_Frequency
iii) Z = I * IP_Frequency
3) Finally, calculate cookie_prob:

$$\text{Cookie\_Prob} = 1 - (1 - X)(1 - Y)(1 - Z)$$

4) Append all entries from ALPHA to BETA.
5) Delete all entries in ALPHA

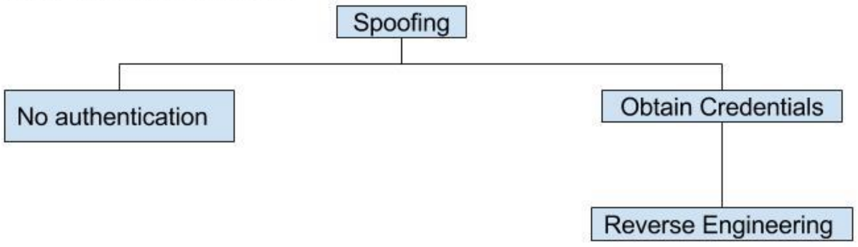
Appendix H: STRIDE Analysis

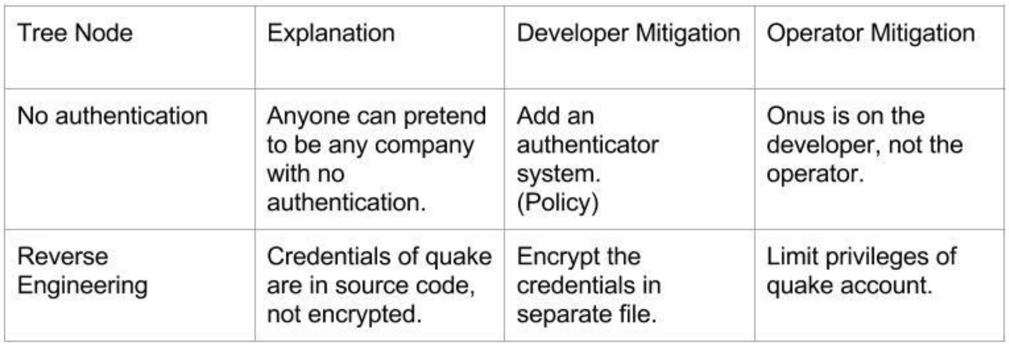
Summary of Threats

Security Threat	Explanation	Developer Mitigation	Operator Mitigation
No authentication	Anyone can pretend to be any company with no authentication.	Add an authenticator system. (Policy decision)	Onus is on the developer, not the operator.
Reverse Engineering	Credentials of quake are in source code, not encrypted.	Encrypt the credentials in separate file.	Limit privileges of quake account.
Several overwrite submissions.	If an attacker figures out the name of a submitted company, they can continually update contact info, and lead to some fake route.	Ensure that all submissions are securely logged a unique password or hashkey.	Send out controlled batches of surveys.

STRIDE (Spoofing) Tree:

Threat: Spoofing the Form and the Flask server

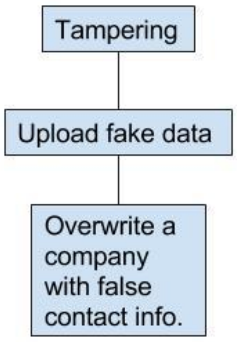




Tree Node	Explanation	Developer Mitigation	Operator Mitigation
No authentication	Anyone can pretend to be any company with no authentication.	Add an authenticator system. (Policy)	Onus is on the developer, not the operator.
Reverse Engineering	Credentials of quake are in source code, not encrypted.	Encrypt the credentials in separate file.	Limit privileges of quake account.

STRIDE (Tampering) Tree:

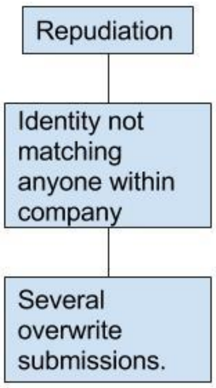
Threat: In a spoofing context, tamper with data in proprietary database.



Tree Node	Explanation	Developer Mitigation	Operator Mitigation
Overwrite a company with false contact info.	Can replace contact data of a company with different values.	Maintain some sort of unique hashing preventing a different person from making such changes.	Onus is on the developer, not the operator.

STRIDE (Repudiation) Tree:

Threat: Possibility of not confirming authenticity of identity.


Repudiation
Identity not matching anyone within company
Several overwrite submissions.

Tree Node	Explanation	Developer Mitigation	Operator Mitigation
Several overwrite submissions.	If an attacker figures out the name of a submitted company, they can continually update contact info, and lead to some fake route.	Ensure that all submissions are securely logged a unique password or hashkey.	Send out controlled batches of surveys.