# Data Warehouse

Project Report

Submitted by Group # 6

Greg Petsul - 36258747

Michael Sheroubi - 21218169

Ahmed Fayed - 87874376

Wang Tianhao - 61903167

# Project Summary

> *This project is centred around building a Data Warehouse (DWH) to house checkouts and fundraiser related data for Trellis Social Enterprise Inc. This involves extracting Data from various data sources, storing the extracted data in a Data Warehouse designed by us, and creating an API to display analytics based on business use case questions required by our client, on their internal dashboard. We will be choosing which cloud DWH to implement, designing the DWH schema based on Trellis' various data sources and needs, finding and implementing an ETL tool to transfer the data from their sources into the DWH, and bringing it all together in an API so they can implement it in their dashboard.*

For the project, our team will design a schema for the DWH, then use a third-party service to host it. We will perform ETL (Data integration) to migrate their existing data into the DWH. We will design OLAP queries to answer business use case questions posed by our client. The results of these queries are accessible through an API that connects to the warehouse. This API will be used to display results on our client's internal dashboard.

## **Use Cases**

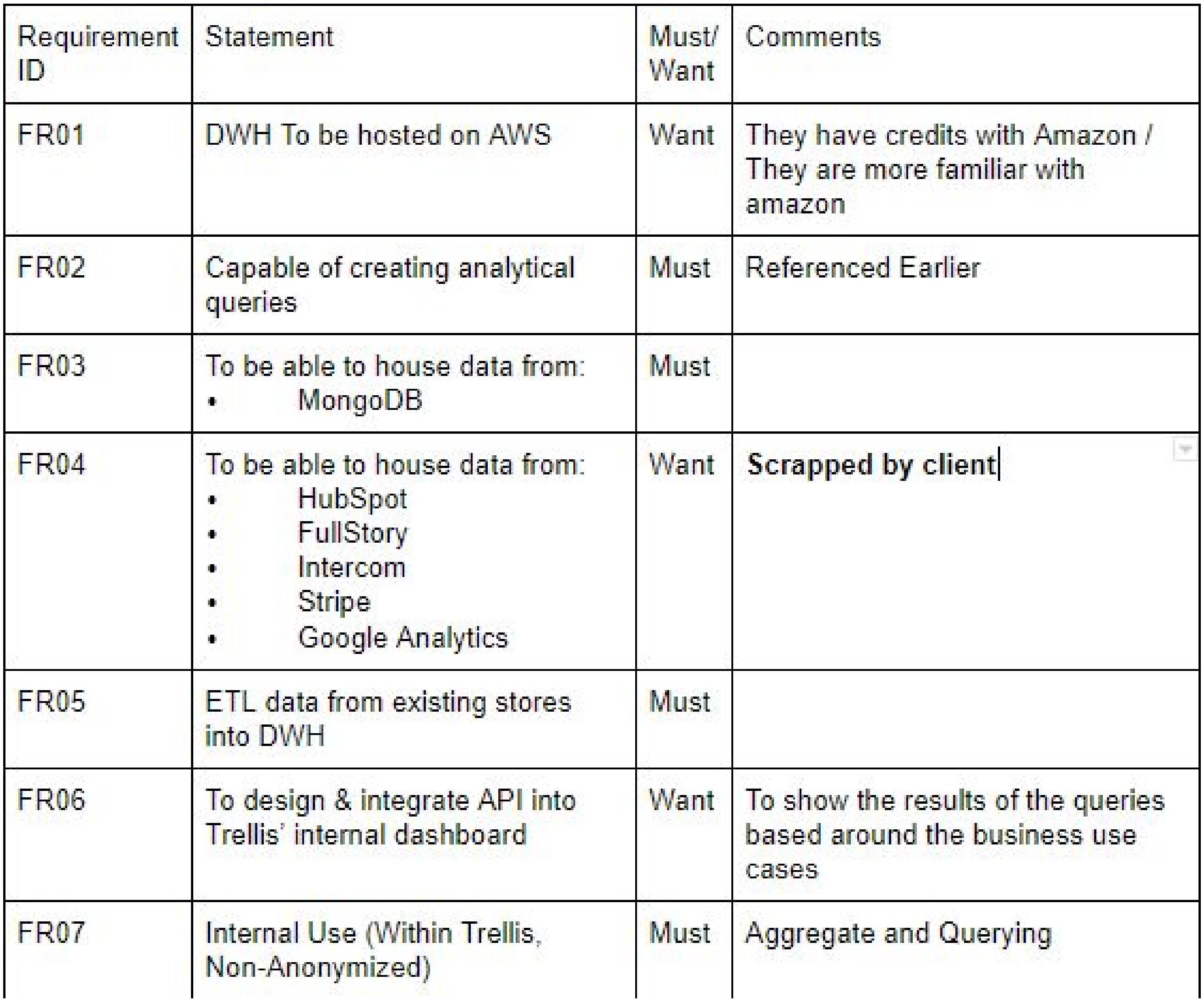
### *Sales*:

Using queries like the “time to first dollar”, a Trellis salesperson (SP) can use this information when pitching to new clients. The SP will log onto the Trellis Internal Dashboard and navigate to the tab containing the query results. There they will be able to see general query results and aggregates. They also have the option to find organization specific information.

### *Executives*:

Using queries like the “total lifetime value”, a Trellis Executive (TE) can use this information to get an up-to-date perspective on the performance of the fundraisers hosted on their platform. The TE will log onto the Trellis Internal Dashboard and navigate to the tab containing the query results. There they will be able to see general query results and aggregates. They also have the option to find organization and user specific information.

# Functional Requirements

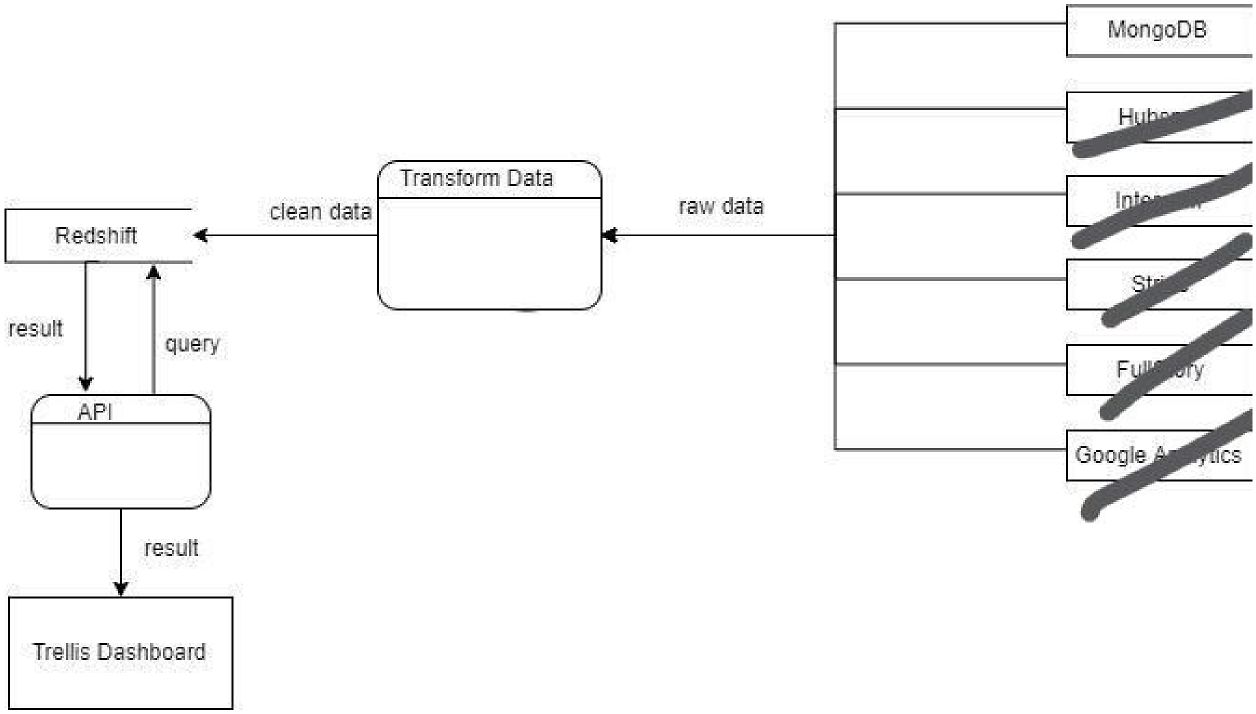


| Requirement ID | Statement | Must/ Want | Comments |
|---|---|---|---|
| FR01 | DWH To be hosted on AWS | Want | They have credits with Amazon / They are more familiar with amazon |
| FR02 | Capable of creating analytical queries | Must | Referenced Earlier |
| FR03 | To be able to house data from:<br>• MongoDB | Must | |
| FR04 | To be able to house data from:<br>• HubSpot<br>• FullStory<br>• Intercom<br>• Stripe<br>• Google Analytics | Want | **Scrapped by client** |
| FR05 | ETL data from existing stores into DWH | Must | |
| FR06 | To design & integrate API into Trellis' internal dashboard | Want | To show the results of the queries based around the business use cases |
| FR07 | Internal Use (Within Trellis, Non-Anonymized) | Must | Aggregate and Querying |

- Total lifetime value (LTV) by header type question not answered
- Total lifetime value (LTV) by champion question not answered

# System Architecture

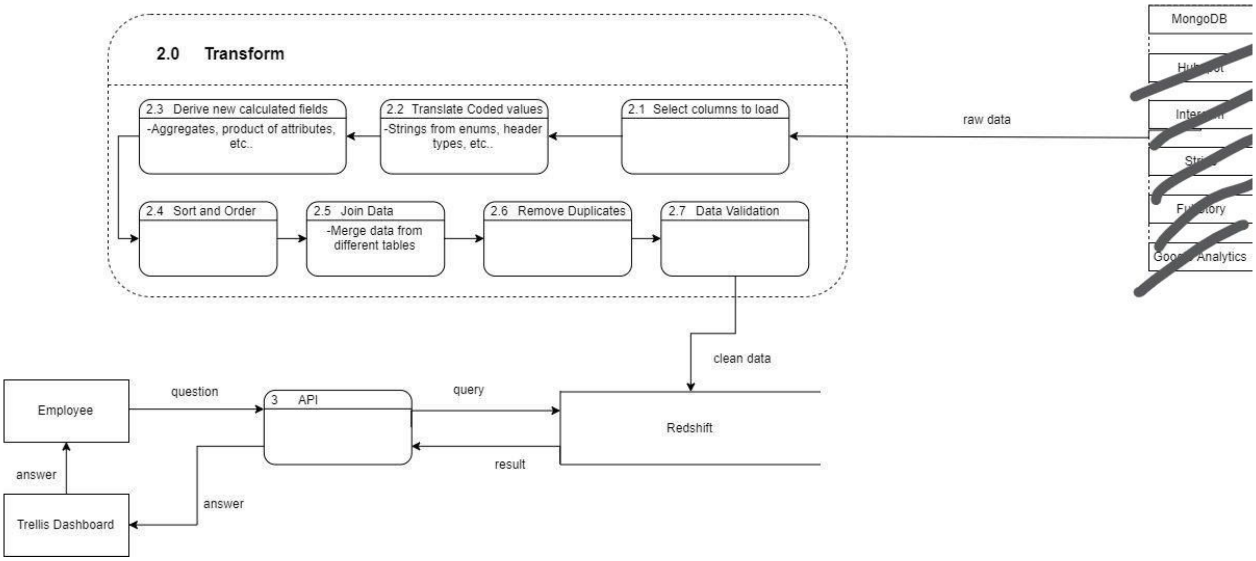
## DFD Level **0**



Notes:

- The data-flow focuses on how the third-party data reaches the Trellis dashboard. This system does not interact with systems other than the dashboard and the listed third-party applications.
- Per the client’s instructions, the list of data sources has been trimmed down to only include their data in MongoDB

## DFD Level 1



Notes:

- The API will allow the trellis dashboard to connect to the Redshift cluster hosting the data warehouse.

# Technical Specifications

## Amazon RedShift

Tellis has experience using AWS and has a credit to go towards amazon services, making RedShift as our Data Warehouse of choice. Keeping Trellis in a familiar ecosystem, having the reliability of Amazon's cloud services, and having a Data Warehouse that fulfills their needs while at a reasonable, pay-for-what-you-use, price point is why we chose RedShift over Data Warehouse service like Google BigQuery and Snowflake . BigQuery is a less cost effective option if the company does not plan to use it frequently and does not require many large queries. RedShift, on the other hand, has a predictable hourly rate or an option to pay upfront for 1 or 3 years (at a deep discount) at a much lower and predictable rate than BigQuery can offer. There is a further breakdown of RedShift, Snowflake, BigQuery, as well as Microsoft Azure SQL Data Warehouse that we brought to our client to help make our decision which can be found in the pages following this section.

## Typescript

Typescript was outlined by our client to be a requirement for our project as it is what they are familiar with. Typescript is a strongly type language that, when compiled, gets translated into javascript, essentially making it a more familiar-feeling, object-oriented style language with some additional error checking over Javascript. Java is a language with similar benefits as Typescript has over Javascript. We are also familiar with Java, however, Typescript can be used for both back-end server side programming as well as having the benefit of Javascript for any front-end client side programming we need. Being able to use one language for our project that the developers at Trellis are used to using outweigh the benefits of Java.

## Node.js

Node.js is a JavaScript runtime environment that allows JavaScript code to be executed outside of a web browser. This allows for server-side scripting that you would not be able to do with JavaScript. Installing Node.js also includes npm (node package manager) which was used to install dependencies and other node packages like Nest.js.

## Nest.js

Nest.js was also a requirement from our client for our project. Nest.js is an angular styled wrapper for Node.js and Express.js which enables us to create server-side applications using typescript. Nest.js is an opinionated framework, that is, it will help guide the structure of our project, which should help our project be more maintainable. This is as opposed to an un-opinionated framework, like Express.js. This should hopefully allow us to build a more cohesive project, while still allowing us to use the features of Express.js. Nest.js implements the functionality of a typical Node.js framework, however, Nest.js introduces tools and structure to better organize our project over plain old Node.js. This should allow our code to be a bit more readable and maintainable when we hand our project off to Trellis.

## Angular, Angular Materials and ngx-charts

Angular, Angular Materials, and ngx-charts were used, together, for the front-end component of our project as recommended by our client. Angular is a component-based architecture that allows for reusability and better readability. In angular you build components with Typescript, which is the language we've chosen, and create the templates in html. Angular Materials allows for clean form building, dialog popups, auto-complete (with some extra work) and many more. Angular Materials is similar to bootstrap but is integrated seamlessly with Angular. Angular materials allowed us built, clean looking, dynamic front-end. Ngx-charts is a Data visualization tool created for Angular. Ngx-charts allowed us to create easy-to-use, aesthetically pleasing charts, using data directly from our API.

## Python (MongoDB to Redshift ETL)

Using the pymongo library to connect to MongoDB and psycopg2 to connect to Redshift, one script was used to connect to both simultaneously. For every collection, the script iterates through every document and appends any attributes that are missing from the Redshift schema matching that same collection. After the redshift schema is defined to include every possible attribute in the mongo collection, one collection at a time, the script extracts every document from the collection and loads it into the redshift cluster. While this is being done, the script verifies and matches data types.

## Stitch Data (Scrapped)

To extract the data from Trellis' various data sources and integrate it into Amazon RedShift, we chose to go with Stitch Data. Stitch Data is an ETL tool that will help us Extract, Transform, and load our client's data sources into our data warehouse. Stitch supports all of the data sources required by Trellis as well as RedShift. Informatica, another ETL tool, has less transparent pricing and requires a locked in annual contract. While another ETL tool, AWS Glue, has a less predictable, hourly, pricing and would require us to orchestrate Glue to work for our needs where Stitch Data supports what we need out of the box.