Software Design Document

Version 1.0

8 February 2022

Team Truthseeker

Garry Ancheta
Georgia Buchanan
Jaime Garcia Gomez
Kyler Carling

Project Sponsor	*Team Faculty Mentor*
NOBL Media – Jacob Bailly	Felicity H. Escarzaga

Accepted as baseline requirements for the project:

For the Client: _____

For the Team: _____

Table of Contents

1- INTRODUCTION 1

2 - IMPLEMENTATION OVERVIEW 3

3 - ARCHITECTURAL OVERVIEW 5

4 - MODULE AND INTERFACE DESCRIPTIONS 7

Section 4.1 - Web Application 7

Section 4.2 - API 14

5 - IMPLEMENTATION PLAN 17

6 - CONCLUSION 19

1 - Introduction

Today, misinformation is widespread on the internet. Different platforms intentionally spread misinformation to harm individuals and groups. The internet, however, is just another representation of many businesses through websites. For most businesses, websites are another source of income through the showing of advertisements. Demand Site Platforms (DSPs) are what allow placement of advertisements on websites and currently, DSPs deal with misinformation by blacklisting or demonetizing websites, but only do so when they are actively told to by the advertiser. Thus, an advertiser can be damaged by being associated with misinformation which is a waste of money since it might deter customers from supporting the advertiser when it is found that the advertiser appears to be supporting misinformation.

This is where the Misinformation and Credible News Analysis Tool comes in; the tool allows advertisers the choice not to support businesses that spread misinformation by rating how credible a page is on a website and deciding whether an advertisement should be placed on that page based on its credibility. Unlike the current process where businesses must tell DSPs which websites the advertisers' ads should not be placed on, NOBL Media has the capability to automate the process by allowing advertisers to simply tell NOBL Media what credibility rating a website should have for their advertisements to be placed on.

NOBL Media collects data over advertisement information for advertisers who choose to use NOBL Media's services. However, they do not have a way for their customers to visualize or obtain this data in any way. This is where Team Truthseeker steps in with two components: a web application and an Application Programming Interface (API). These two components allow NOBL Media to solve the problem with their business flow through the implementation of the following features:

1. The web application must be able to create and authenticate customer accounts to allow secure access to the customer's data using integrated technology Auth0.

2. The API must be able to handle user authentication requests. Once authenticated, the API must handle a request to be parsed into NOBL'S MySQL database.

3. The web application must be able to abstract JSON data coming from the NOBL Media MySQL database and represent these to customers through graphs and charts using technologies such as ECharts.

4. The API must be able to retrieve the JSON data from the NOBL Media MySQL database and return an HTTP 200 level response with JSON data to NOBL's web application or the client's own site.

5. The web application must allow customers to download customer ad data in a formatted file such as in a CSV or Excel file.

Once that information is collected, advertisers will be able to see how their advertisement is performing in terms of supporting misinformation and how much money the advertiser is losing due to misinformation through a user-interactive web application. Through the web application, users will be able to log in to see their respective campaign information, such as viewing different graphs and charts that depict how well their advertisement campaign is doing, how many non-credible sites it has avoided, and their overall NOBL score. Ultimately, the web application offers a way for clients to actually see that they receive more interaction with their advertisements when they are putting them on more ethical sites.

Chapter 2 - Implementation Overview

NOBL Media has an impressive AI that reads web pages and assesses how brand friendly they are and even holds records of it's analysis with metrics in their database. However, NOBL Media's Misinformation and Credible News Analysis Tool does not have a web application for NOBL's clients to interface with. In other words, NOBL client's data is all stored in NOBL servers and there is no way for clients to analyze and act on their very own campaign data. Team Truthseeker is developing the web application experience for NOBL's clients to interface with.

The Misinformation and Credible News Analysis Tools requires a safe and consistent user login system. The user logs in to their campaign with a username and password given to them by NOBL Media. Auth0 handles the user login by checking if the account exists and if the password matches the account. After Auth0 verifies the user's credentials the page takes the user to the Analysis Tool's campaign selection page. One user can have ad campaigns simultaneously. Once the user chooses a campaign, they will be navigated to that campaign's dashboard. Once in the Analysis Tool's dashboard, users will have access to campaign information, NOBL scores, as well as data visualization tools.

The dashboard is how users will interface with the Analysis Tool and all of their campaign data. NodeJS serves as the Analysis Tool's front end platform that supports javascript for the website. NodeJS is meant to build input/output, server side applications. Express.js is a framework built on NodeJS that is meant to build web applications because it provides routing and middleware services. Routing means selecting paths for web traffic through multiple networks (GET, POST, PUT, and DELETE HTTP requests). Middleware receives Request and Response objects. Gatsby is a React-based Framework that creates static web pages during build time on the user's machine. NodeJS, Express.js and Gatsby.js are the basic building blocks for the web application. The final piece for the front end of the web app's visualization is provided by Apache ECharts. Apache

ECharts is a visualization library that produces sophisticated graphs and charts. The visualization of data is important to the web application because it gives NOBL Media's clients insight on trends and outliers. NOBL's clients will be able to act quickly on trends seen on graphs to improve their ad campaigns.

The web application requires a method for gathering information to populate the skeleton of the dashboard. The REST API for the web application is the necessary middle man between the web application and the servers holding all the campaign information. The web application is using REST API to do the database queries for the web application to be retrieved from NOBL Media's servers. The web application's REST API retrieves JSON data from NOBL's servers to populate the NOBL Scores for NOBL clients' campaigns as well as provide the data for the app's data visualization.

NOBL Media's client campaign information is all stored in a MySQL server. The rest of the technologies are meant to be compatible with the NOBL server to populate the web application with database information, since MySQL is an environmental requirement.

Chapter 3 - Architectural Overview

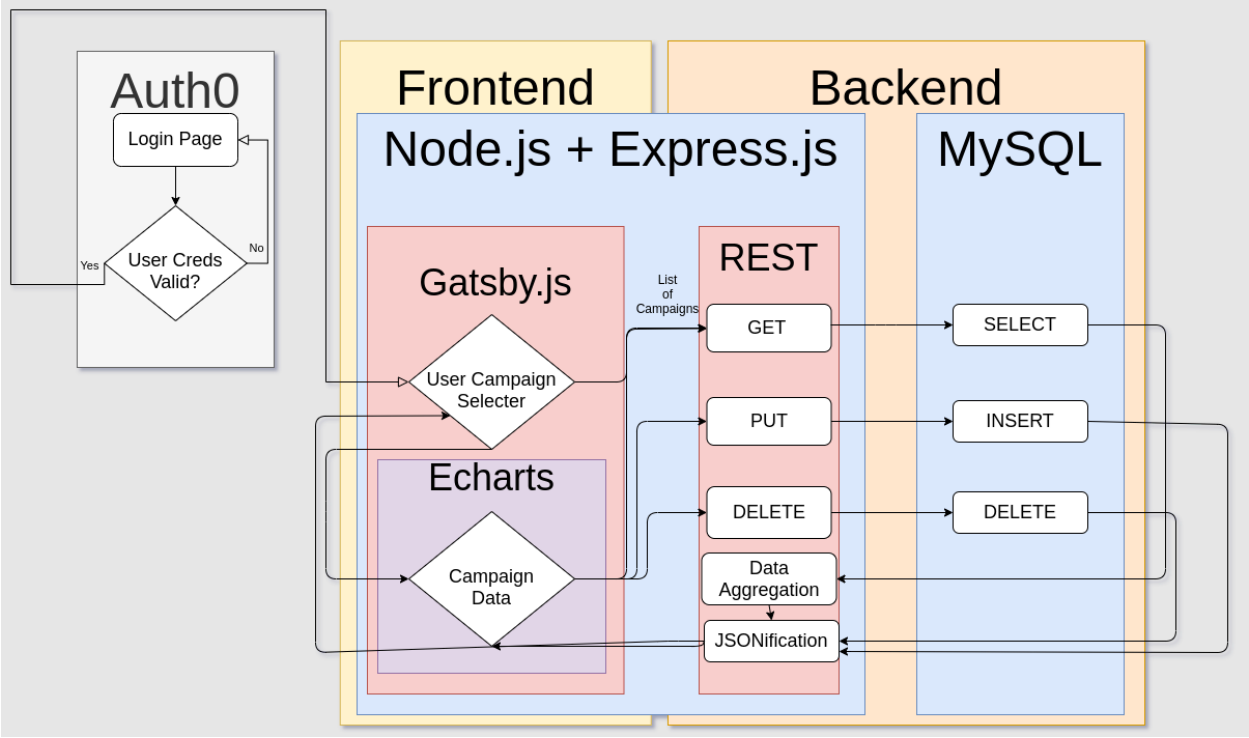

Figure 3.1 - Overview of the NOBL System

The system is divided into three parts: Auth0, a third-party component that handles authentication shown in Figure 3.1 in an off-white eggshell color. Note the absence of an account creation mechanism because all account creations will take place via direct communication with NOBL.

After authenticating via Auth0, the user will reach the front-end web application (colored tan-ish yellow in figure 3.1) powered by Gatsby.js, the framework that is responsible for quickly rendering the static elements of the webpage while ECharts is used to display the graphical depictions of NOBL's data. The two technologies work in tandem to produce the form and functionality of the web dashboard frontend and are expected to be the main interface to NOBL's data that NOBL's customers make use of.

In order to populate the web application with graphical data elements, data is queried from a MySQL database. The REST API translates HTTP requests into equivalent SQL queries, queries NOBL's backend MySQL database (denoted in darker-tan in Figure 3.1), aggregates the data, and returns the result in JSON notation for ECharts to process into graphical elements. The API's responsibility is not limited to the web application and may be used outside of it for organizations that wish to process the aggregated data themselves for their own reports or to display on their own websites.

This interaction between the web application and NOBL's backend MySQL database via a REST API forms the core general loop of information flow within the project. The specific flow of information will be determined by the action the user takes when using the website. For example deleting a record in the database skips the aggregation step because there is no data to aggregate. Instead the index of the deleted record is simply returned.

Another thing to note about figure 3.1 is that it does not depict the aforementioned technical users who may make use of direct access to the API and bypass the web dashboard. They can be thought of as existing in the frontend area given that they will still need to authenticate via Auth0 and query via the REST API effectively opting for a command line interface to the data as opposed to a graphical user interface.

Chapter 4 - Module and Interface Description

There are a variety of components to this architecture design. It serves as an overview to a deeper description of software components. Each component will be described in this chapter.

Section 4.1 - Web Application

Three components are needed for the web application: a login page, a campaign selection page, and the dashboard page. The components will provide a familiar interface to NOBL client's. Aesthetics aside, the web application's functionality will allow NOBL clients to analyze and interact with their campaign data securely and effortlessly.

Login Page

The login page is the user's first module from the Misinformation and Credible News Analysis Tool they will interface with. NOBL Media provides clients with a username and password to login for their businesses account. Once the user enters their credentials, Auth0 will check if the user exists and if the passwords match. For security purposes, the login page will not tell users if the email is incorrect. The login page will present the user with a message saying the credentials were typed in wrong. If the login credentials are correct, the user will be sent to the campaign selector page.

Figure 4.1 shows the current team design for the login page. The user is prompted for their email address (username) and for their password. Should users find that they have forgotten their passwords, they will be able to reset their password by clicking the "Forgot Password?" button that is at the bottom in between the password input form and the continue button.

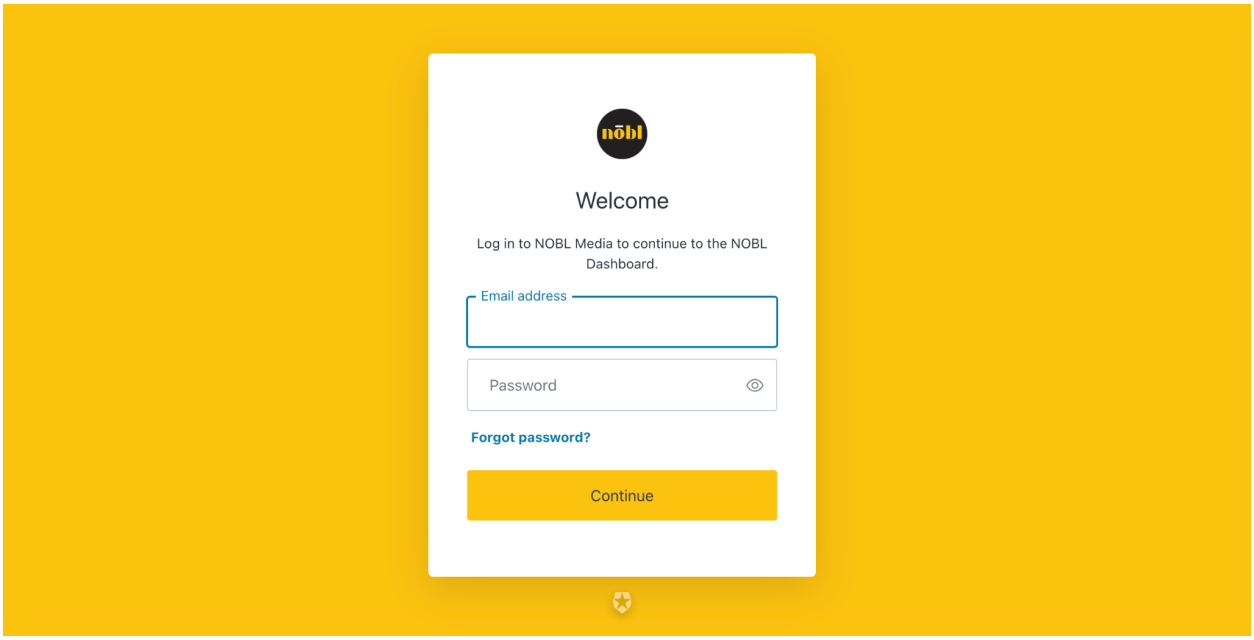

Figure 4.1 - Current implementation of the Login Page

Figure 4.2 shows a state diagram of the user process of logging in. It starts with the user entering their username and password. Should the user forget their password, they will have to click on the "Forgot password?" button and will be then sent a link to reset their password to their email. Once they have clicked the link and reset their password, they will then have to re-enter their username and password. Should the user enter their correct email address and password, authentication will proceed and check if the email address is found and the password for that specific email address matches the one inputted by the user. If both are found to be true, then the user is authenticated and can proceed to the campaign selector page. Should the user fail to authenticate themselves, they will be prompted with a notification that either their email address or password were wrong and will have to re-enter them once more.

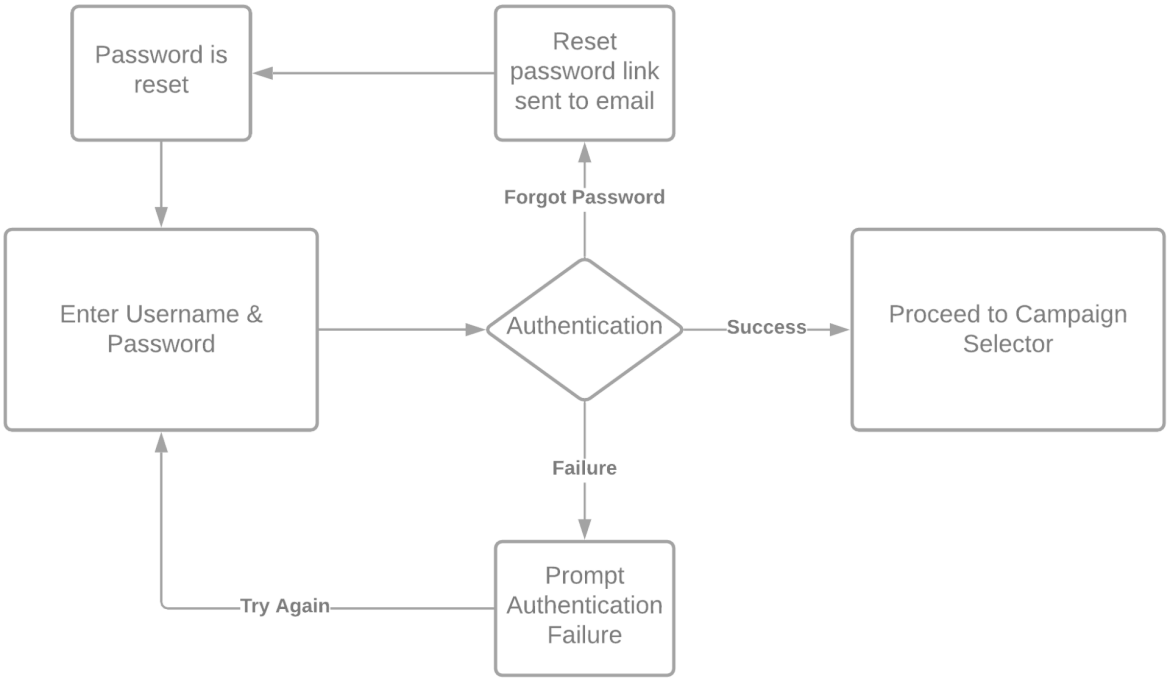

Figure 4.2 - Process diagram for the login page

Campaign Selector

NOBL Media clients can have multiple ad campaigns per account. At this point in the login process, a user will have to pick a single campaign to proceed to the user dashboard.

Referring to Figure 4.3, the header section of the page is where the "Organization Name" and the greeting of the day is located. Both the greeting of the day and where it says "Organization Name" is dynamically displayed, pulling data from the NOBL database to retrieve the user's first name and last name and the organization that the user is under. Additionally, the menu below the header section is also dynamically created and will only be fully initialized once the web application has retrieved all the campaigns that the user is associated with. Additionally, Figure 4.3 also shows the "Confirm" button, grayed out. This is controlled by the web application so that the button cannot be clicked by the user before clicking on a campaign. Once a campaign is selected, which the user will know since it will be highlighted on the menu, the confirm button will brighten and turn blue making it known that it can be clicked.

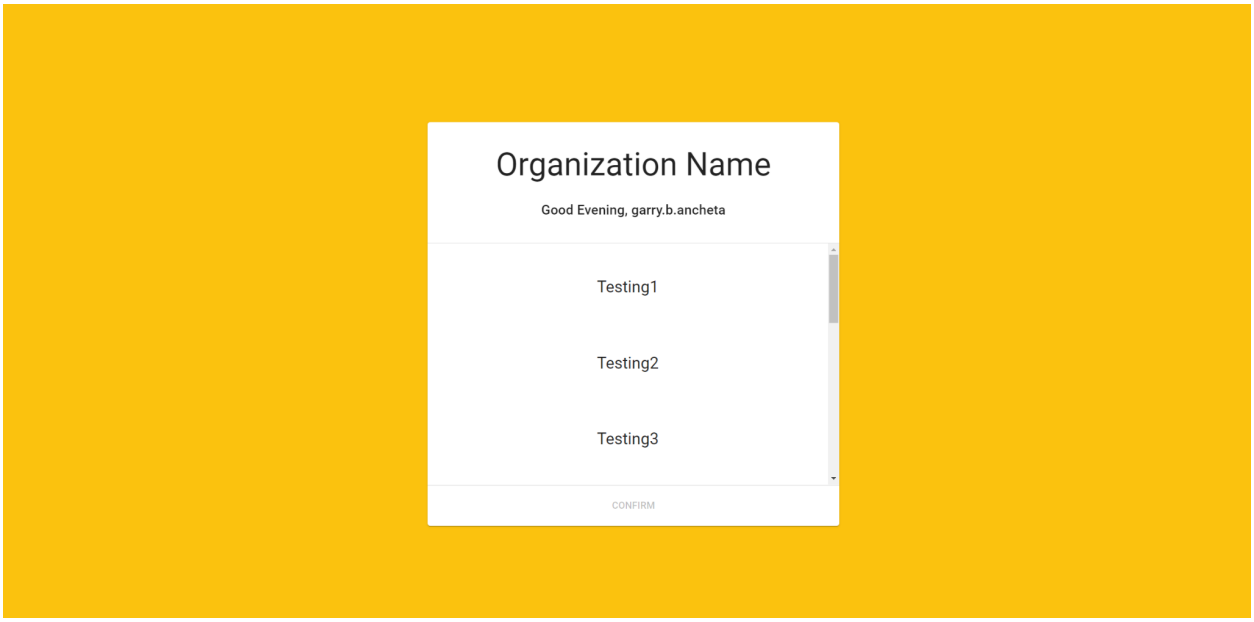

Figure 4.3 - Current Implementation of the Campaign Selector Page

Figure 4.4 shows the basic workflow of the campaign selector page. Once the user has completed the authentication process referred to in Figure 4.2, as soon as the user is logged in, the web application will start the process at the beginning of Figure 4.4. In the underlying processes of the web application, the campaign selector page will connect to the API, which will then go to the REST API route for campaign data retrieval. The route will then attempt a query to the database to retrieve all campaign data that the user is associated with. This is where there are three possible results: an error, no campaigns found, or campaigns were found. If there is an error, this will be sent back to the web application and the user will be prompted with an error message. If there are no campaigns found, the menu section will prompt that no campaigns have been found for the user and should contact a NOBL administrator. If campaigns are found, the API will then send the information to the web application. The web application will then display the campaigns, allowing the user to pick one. Once the user has picked a campaign and has pressed the confirm button, they will be sent to the campaign dashboard.

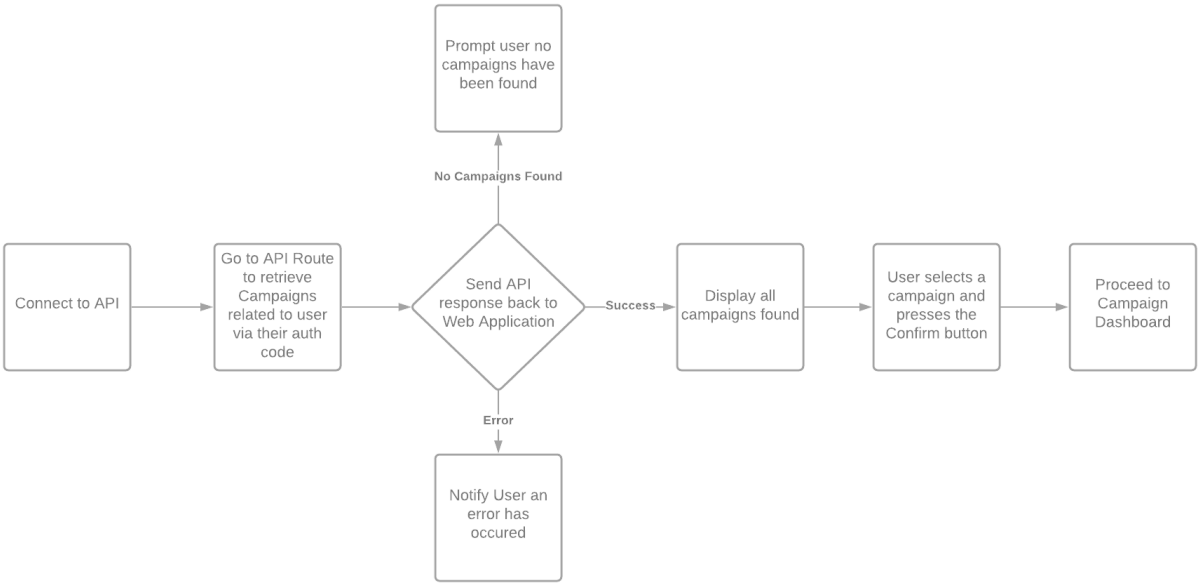

Figure 4.4 - User process diagram of the Campaign Selector page

Campaign Dashboard

After logging in and choosing an ad campaign users will finally arrive at the campaign dashboard page. This is the main page users will spend their time on.

The dashboard contains tabs for different campaign information. The main tab at the top allows the user to switch between their account's ad campaigns. As seen on Figure 4.5 there are additional tabs showing different views of the campaign data. Figure 4.6 shows a close up view of the tabs.

- Overview
    - A summary of all campaign data. Provides a quick look at how the campaign is performing at this time.
- Toxicity
    - Overall toxicity score of web pages hosting ads.
- Topics
    - Allows users to pick the parameters of data they want to be shown from the web server.
- Campaign Data
    - Allows users to retrieve specific campaign data in the form of a CSV file or an excel spreadsheet.

The right side of the dashboard (where the "GraphDemo" placeholder currently is in Figure 4.5) is where the data visualization will go. The user will be able to modify the data with the subject tabs. There will also be options for the type of graph the user will want to use. Users will have the ability to isolate data in a set timeframe.

The Download now button (seen on the bottom left of Figure 4.5) will allow users to download the current report on the screen. There will also be an option for users to get the report in the form of a CSV or Excel file.

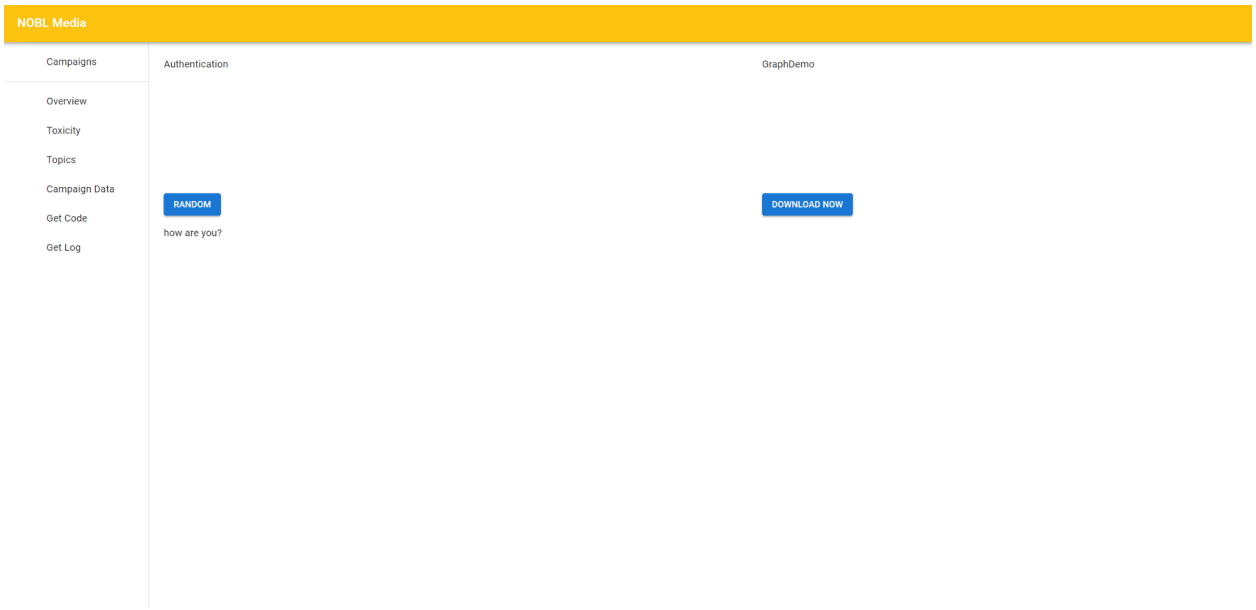

Figure 4.5 - Current implementation of the Dashboard Page

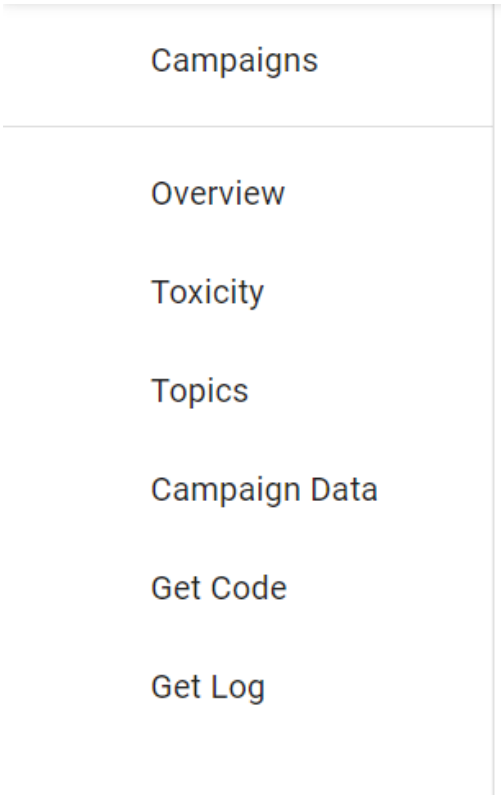

Figure 4.6 - Current implementation of the Dashboard Sidebar

Section 4.2 - API

The back end of this web application consists mainly of two components: the API and NOBL's database. The web application's API is written as a REST API, which is a common API architectural style using HTTP requests. The API itself will be the middleman between the web application and NOBL's database. Additionally, the API can also become the middleman between the NOBL customers directly and the NOBL database.

Pertaining to the API, its functionalities are not yet set in stone as the team discusses the different needs of NOBL customers as well as the web application's needs. However, there are several functions that are currently defined within the API to retrieve specific information that the web application needs.

Client Functions

These are functions pertaining to the retrieval, updating, and creation of NOBL client data in the database.

1. Create
    a. Inserts a new client into the NOBL database
    b. Success - Returns an Insert ID
    c. Failure - Returns an error
2. Find Specific Client
    a. Retrieves a specific client based on the Client ID
    b. Success - Returns the an object containing Client ID, Client Name, Due Interval Hours
    c. Failure - Returns an Error
3. Retrieve All Clients
    a. Retrieves all clients in the database
    b. Success - Returns an object containing Client ID, Client Name, Due Interval Hours
    c. Failure - Returns an Error

4. Update Specific Client
    a. Changes specific data fields for a specific client
    b. Success - Returns nothing
    c. Failure - Returns an Error

Datafile Functions

These are functions pertaining to the table that has a one-to-many relationship to the actual page data that has been scanned and rated by NOBL.

1. Create
    a. Inserts a new datafile into the NOBL database
    b. Success - Returns an Insert ID
    c. Failure - Returns an error
2. Find Specific Datafile
    a. Retrieves a specific client based on the datafile ID
    b. Success - Returns the an object all data from the specific datafile
    c. Failure - Returns an Error
3. Retrieve All Datafiles
    a. Retrieves all clients in the database
    b. Success - Returns an object containing all datafiles in the database and their corresponding data
    c. Failure - Returns an Error
4. Update Specific Datafile
    a. Updates a specific data field(s) for a specific datafile
    b. Success - Returns Nothing
    c. Failure - Returns an Error

Page Functions

1. Create
    a. Inserts a new page into the NOBL database
    b. Success - Returns an Insert ID
    c. Failure - Returns an error
2. Find Specific Page
    a. Retrieves a specific page based on the page ID
    b. Success - Returns the an object all data from the specific datafile
    c. Failure - Returns an Error
3. Retrieve All Pages
    a. Retrieves all clients in the database
    b. Success - Returns an object containing all pages in the database and their corresponding data
    c. Failure - Returns an Error
4. Update Specific Page
    a. Updates a specific data field(s) for a specific page
    b. Success - Returns Nothing
    c. Failure - Returns an Error

Chapter 5 - Implementation Plan

With respect to the modules planned for the project, the implementation needs to flow in a manner that allows progressive implementation of the project functionalities from the ground up. This means that the implementation must flow how the user accesses the web application. Figure 5.1 is the simplification of the user's workflow. As a result, the basic modules of the project are:

1. Login Page
2. Campaign Selector
3. Campaign Dashboard

Figure 5.1

The basic foundation of the codebase was implemented starting in November, in preparation for the tech demo. Using the tech demo, the team was able to implement the login page using Auth0 without any problems. Up until the start of the semester, the team was trying to configure basic API functions that allowed the web application to pull data from the NOBL database. For security purposes, the database for containing user information is separate from the NOBL SQL database so that if user information were to be hacked, the sensitive information about ad campaigns that are tied to the user will not be vulnerable.

Referring to Figure 5.1, the team has completed the interface for the campaign selector and have currently completed integrating the API into AWS Amplify. The team had to integrate the API into AWS Amplify so that it is able to access the

database without any security issues. Additionally, the team has started creating the Campaign Dashboard ahead of time as well. During the window for the Campaign Dashboard, the team will also start integrating the API into the dashboard as well as granulating the API. Granulating the API means that the API itself will be able to retrieve specific data from different data fields instead of only being able to retrieve groups of data programmed by the team. As the team implements the web application and the API, the team will also be considering the aesthetics of the web application so that the project will be presentable to the client. The team is separated into two subgroups, the front-end team and the back-end team.

After spring break, the team intends to debug and test the application. Specifically, the team intends to use unit testing to make sure all functions used in the web application and API outputs the correct results. Additionally, the team will test all interface functionality to make sure they are working properly. Once debugging and testing is complete, the team will then refactor the code to make it as readable as possible as well as expandable by the NOBL team.

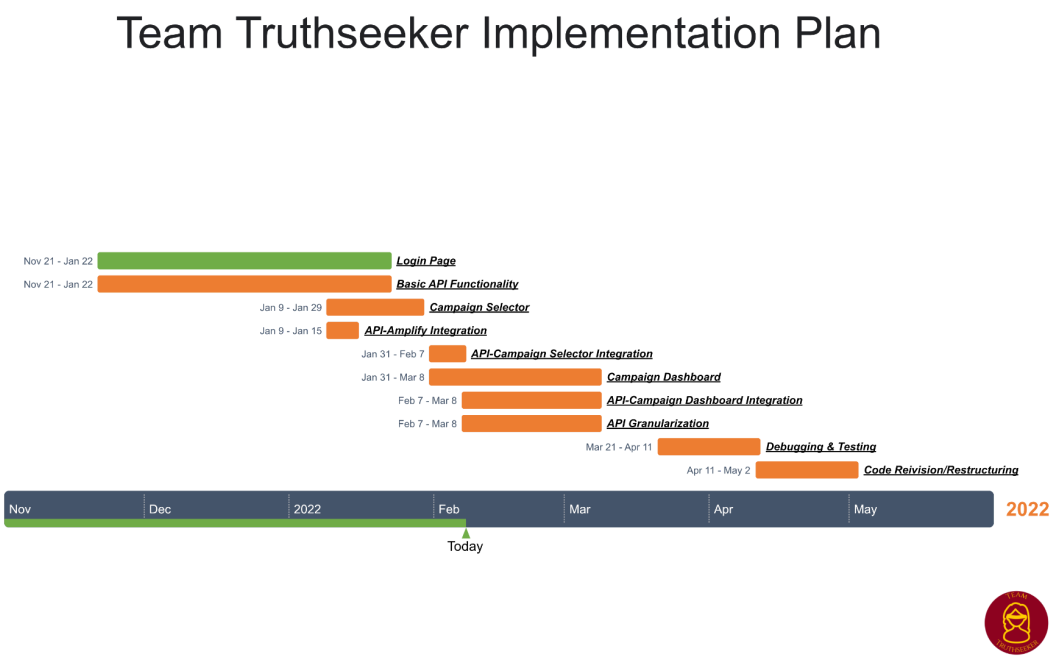

Figure 5.2 - Implementation timeline for the project

Chapter 6 - Conclusion

Due to the lack of proof that can be shown to NOBL Media customers, NOBL Media cannot truly show that their service has any benefits. This produces the problem that the project is intended to solve. The project has two components, the web application and the API. These two components work together to achieve the solution to NOBL Media's problem: show what NOBL's service actually provides.

The solution to NOBL Media's problem requires that the web application be the interface between the NOBL Media customer and NOBL Media while the API acts as the connection between the web application and the NOBL Database. The web application intends to show NOBL Media customers statistics of how NOBL Media's service is affecting their ad campaign by way of graphs and charts. The API is the connection point which transfers data from the NOBL Database to the web application for rendering. Thus, the two components, working together, will be able to solve the problem.

Currently, the team is in the middle of implementing the web application and the API. Specifically, the team has completed the preliminary design of the Campaign Selector and will now be approved by our client. Furthermore, the team is now in the process of completing the first iteration of the dashboard. Additionally, the back-end team is starting on granulating the API, and are in the process of tying the API with the front-end. The team is extremely satisfied with the progress being made and cannot wait to present the final results to our client by the second week of March!