campAIgn - AI Advertisement Generator and Campaign Manager

Diego Aguilar, Jonathan Warren, Daniel Rapuano

Senior Seminar Capstone - Final Report - CSC 400, Dr. Hossain

Southern Connecticut State University

30 April 2024

Table of Contents

1 – Problem Statement

1.1 – Objective and Deliverables

The objective of this project is to deliver a cloud-native AI-Driven Advertisement Campaign Generator and Management Platform for marketing and advertising professionals. This application will enable these content creators to create customizable, AI-generated advertising materials referencing user-input sources for credibility. We will leverage summarization, content generation, and image generation API's to create a final advertisement product and organize these into advertising *campaigns*. Users will be able to manage multiple simultaneous campaigns and finetune campaign parameters (demographic-relevant sources of credibility, advertiser point of view, and ultimate goal of advertising campaign) en route to creating publication-ready content. Once a campaign is ready for publication, our social media integration will enable users to disseminate their content directly into their favorite social media platform.

In order to organize and iteratively develop campaign materials, we will introduce *portfolio* level abstraction. This abstraction will enable users to create a portfolio per client (i.e. Apple), and then create several advertising campaigns for each of that client's product (i.e. iPhone, MacBook, iWatch). Finally, we will enable historical edits rendering of each campaign so that users and their stakeholders may visually navigate the iterative growth of their advertising products.

1.2 – Motivation

As the world of commerce moves away from billboards, magazines, and cliche slogans, the way businesses drive purchases is evolving to capture the savvier, more resourceful consumer of the internet-age. With endless information at their fingertips, the modern consumer requires personalized, data-driven advertising campaigns to influence purchasing decisions [1]. However, given the endless idiosyncrasies of each individual consumer, the work of generating personalized advertising campaigns cannot be performed manually - we need a new, automated approach.

AI-generated advertising campaigns solve the challenge of creating the vast amounts of personalized, engaging content required to drive purchasing decisions in today's e-commerce environment. Such campaigns accomplish this by automating consumer research ingestion, narrative design, and content generation, thus enabling businesses to appeal to ever-pickier consumers and maximize revenue.

1.3 – Significance

The significance of this application lies in its relevance to the modern e-commerce environment. By automating the creation of purchase-driving content, advertisers will be able to both increase volume and enhance personalization, thus accelerating the number of impactful consumer interactions required to drive a purchasing decision [3].

Moreover, our application will enable content creators to work on multiple simultaneous campaigns, thus increasing the agility with which campaigns can be orchestrated cohesively

together for maximum impact and revenue-generating potential. In short, users will be able to create a larger volume of higher-quality content with which to engage with their customer base.

2 – Related Work

2.1 – What alternative software/tools exist to meet the need you specified?

There are already a handful of APIs that are AI-equipped and able to separately manage individual services required to generate our advertising final product [3] - however, we did not come across a product that incorporates all of our listed services into one seamless application.

Most notably, OpenAI has its ChatGPT web interface and is able to provide summarization and content generation services. However, this service is exclusively for single-use prompts - the web interface does not provide any sort of advertising-centric user-interface or campaign/portfolio campaign management data storage or organizational layers. Most impactfully, any advertising campaign inputs and parameters are lost upon the conclusion of a session - this interface is not designed advertisement-generation purposes. Moreover, any sort of image generation engine is hosted on a separate interface and does not ingest historical campaign generation parameters that would finetune a final advertising product.

2.2 – Why is your project more suited to meet those needs?

Our project will provide all of the missing features outlined above - an advertisement generation interface, organizational layers, data/campaign parameter storage, and campaign management tools to streamline ad-generation process from inception to publication.

Our application will be tasked with merging each of the API's and their individual functions to create ready-to-publish advertisement content, as well as manage and keep track of each user's advertisement campaigns. Rather than build a project from scratch, our goal as a group is to expand upon relevant technology in today's landscape and create a new, fun, and valuable product/interface for our end users.

3 – Features

Through our AI Ad Campaign Generator and Platform, we want to create a fun, interactive interface for users to create effective advertising content, manage multiple simultaneous campaigns, and publish their advertisements directly to social media. The core features of the application include an AI Campaign Generator, a cloud-native Campaign Management Dashboard, and a Social Media Integration Interface.

3.1 – AI Advertisement Campaign Generator

The core deliverable of our application will be an AI-generated ad-campaign package comprising text and image files. We will integrate 3 APIs (AI Summarization, AI Content Generation and AI Image Generation) to provide these deliverables. This content will be customized via three user-input parameters:

- Point of View, "POV" - *what are you trying to accomplish with this ad?*
- Sources Cited - *which sources will your ad cite for credibility?*
- Relevant Imagery Description - *what sort of images will resonate with consumers?*

In order to produce the ad-campaign package, the system will perform the following tasks:

1) Feed user Sources Cited into the Summarization API to generate a summary for each source of information. Such sources can include scientific journals, websites, blog posts, news articles, or free text.
2) Feed source summaries and user POV into the AI Content Generator API to create the advertising text content.
3) Feed user Relevant Imagery Description into the AI Image Generation API to create corresponding imagery for the ad.

The user will be able to continuously fine tune the parameters until a campaign is deemed ready to export (text file & images zip) or publish to social media (via the social media integration interface).

3.2 – Campaign Management Platform

In addition to generating an advertising campaign, the user will have access to a campaign management platform. This platform will have the following features and services:

- Profile registration and user authentication.
- Landing page after login giving users an overview of the campaigns they are working on
- Rendering of local “feed” of their organization’s public campaigns for inspiration
- Create a new Portfolio for a new client, after which they can create a campaign for each of their client’s individual products
- Enable users to access and continue working on separate campaigns
- Store and render previous campaign states/versions
- Enable users to edit campaign parameters (point-of-view, sources cited, image prompts)

  to continue fine tuning their campaigns
- Access a campaign portfolio displaying headlines and corresponding visuals of all their current campaigns
- Enable users to export campaign package
- Enable users to publish campaign directly to social media
- Enable users to delete campaigns.

3.3 – Social Media Integration Interface

Once a user is ready to publish their campaign materials, they will be able to access a social media login plugin and posting interface integrated from Facebook through which they

will be able to post their text content and upload the campaign's corresponding imagery into their local account, ready to view by all of their followers/friends.

3.4 – Published, Live Web Application

Through the use of Google Cloud Protocol, the application is deployed live and is accessible 24/7. The application is configured with an 'app.yaml' file to instruct Google App Engine of key environment variables, such as the Python runtime version and API keys, and then the application files and directories are stored within two Google Storage Buckets, one for 'staging' and one for 'deployment'. Next, the application utilizes a Google CloudSQL instance as its associated database, which results in surprisingly fast client-server relations, since both are hosted within the Google Cloud Protocol ecosystem. Google App Engine is then used to deploy, configure, and manage versions and builds, and a Google SQL instance and Cloud Bucket serve to store user data and application files. We built and structured the application built to be easily scalable, both in feature expansion and availability, so as we provide new features within each Google App Engine version, the application will have 24/7 uptime and be pain-free to interact with.

3.5 – API Integrations

Through the use of various APIs and plugins, we provide our application with a multitude of quality-of-life features and enhancements. Primarily, the OpenAI API's and GPT-engines (gpt-4.0-turbo , gpt-3.5 , dall-e-3) provide us with content summarization from information

through specified sources (i.e. scientific journals, websites), text-based advertisement generation from source summaries, and generation of relevant ad imagery based on user provided information.

Various other APIs, such as Imgur, Gmail, GCP, and Facebook APIs are pulled together to provide a cohesive and positive user-experience. The Imgur API is implemented as a direct method to provide image permanence without needing to store raw image files into a database, subverting the temporary two-hour time limit of Dall-E generated images. The Gmail API allows for email notifications to be sent to users upon various action events, such as signing up for the application, changing their password, creating entities within the application (campaigns, portfolios), and editing entities.

The Google Cloud Protocol API allows for direct export of the application to GCP services, and allows for us to directly publish application versions directly to Google App Engine through the command-line interface, rather than having to manually create storage buckets and upload necessary files. The files are uploaded through the command-line, encrypted to ensure file security and safety, and then the build is containerized and deployed, re-routing HTTP traffic once the new version of the application is live.

3.6 – Miscellaneous Features

Aside from the aforementioned, more notable features, listed below will be additional features implemented throughout the development cycle. First, the Facebook API required the

implementation of numerous additional features in order to be considered for approval, notably a privacy policy, an “about” page, a data-collection policy, and the ability for users to delete all collected data from the application. More features included are as follows: Facebook Graph API/Javascript SDK Integration for social media content publication, searchbar in nav for enhanced user experience, 3-Tier abstraction - portfolio, campaign, and individual campaign iterations, homepage feed of public campaigns, user login page, campaign carousel, dynamic & recursive rendering of UI components using a tree data structure, tooltips included throughout the application for improved accessibility, campaign material export functionality through Imgur API and Facebook API, devs page, about page, privacy policy, delete user data policy, public/private optionality for user-generated content as a measure to ensure user privacy, and all-encompassing terms of service.

3.7 – Schematic Diagram

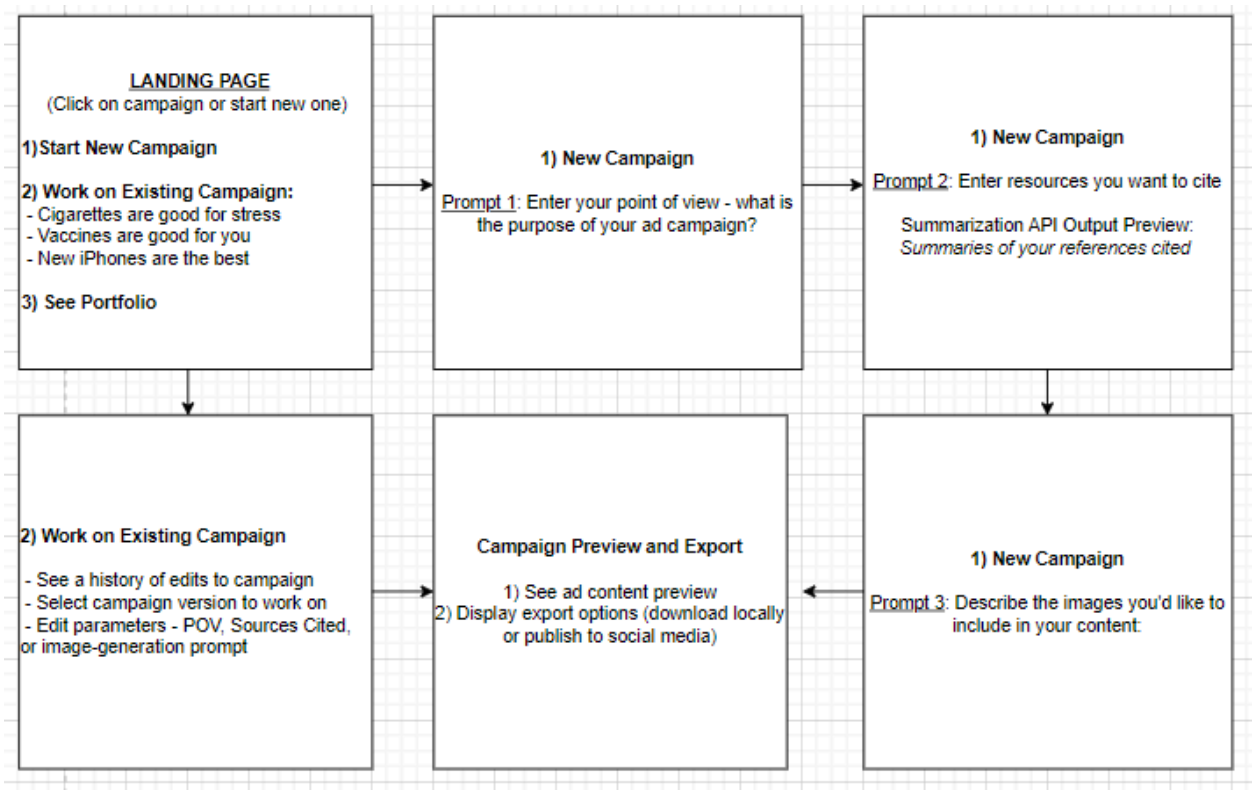

Figure 3.7.1 - the diagram above displays the workflow of the application including fundamental features including new campaign generation, campaign dashboard, and social media publication.

3.8 – ER Diagram

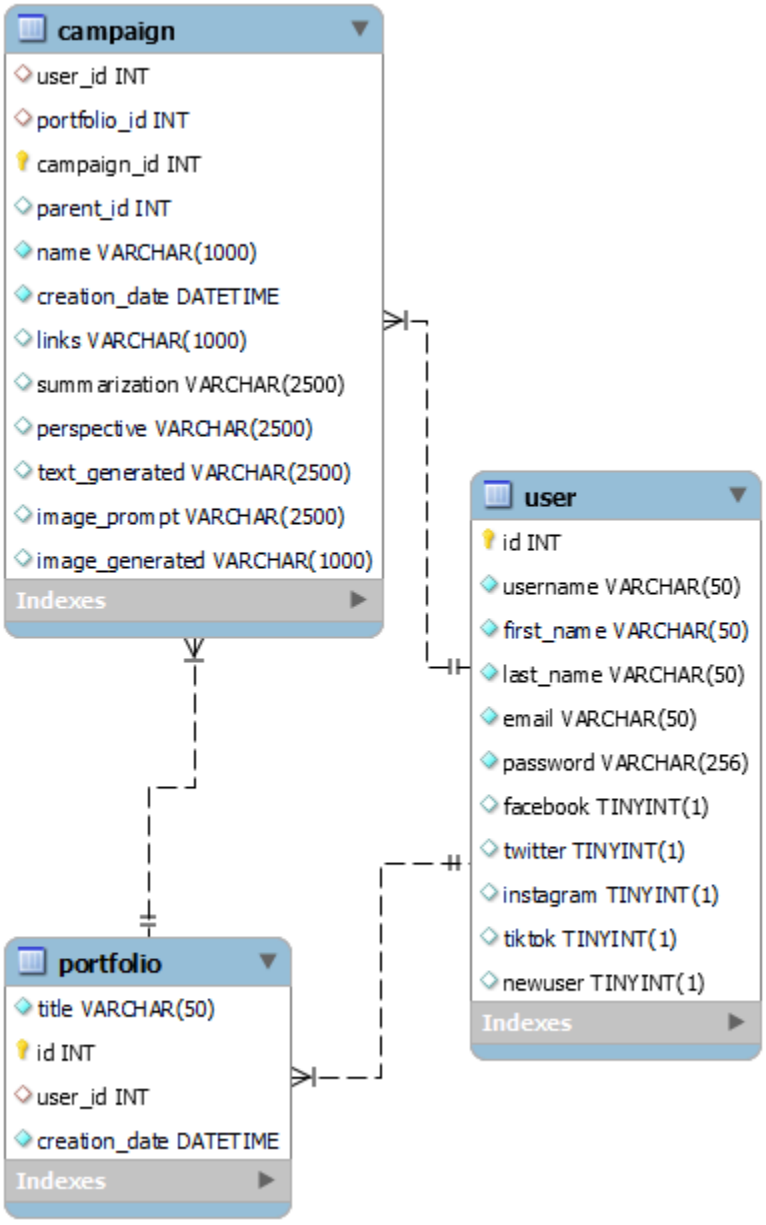

Figure 3.8.1 - the diagram above displays our entity relationship breakdown, outlining each object's variables and their interactions. As can be seen, both ***user*** *and* ***portfolio*** *have one-to-many relationships with a* ***campaign*** *object, and a* ***user*** *has a one-to-many relationship with a* ***portfolio*** *object..*

3.9 – Class Diagram

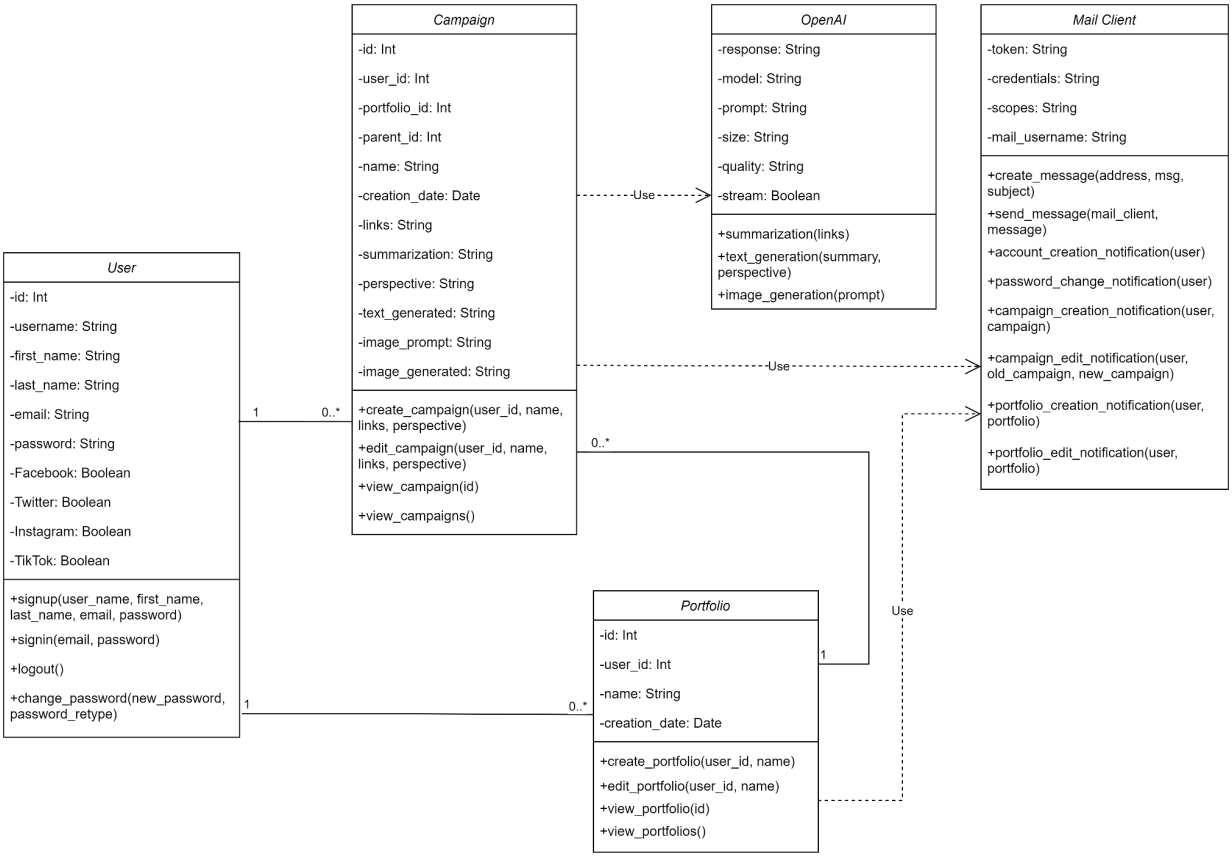

Figure 3.9.1 - the diagram above displays our uml class diagram, outlining each class's components, functions, and their interactions. As seen above, both ***user*** *and* ***portfolio*** *have one-to-many relationships with a* ***campaign*** *object, and a* ***user*** *has a one-to-many relationship with a* ***portfolio*** *object. Both* ***campaign*** *and* ***portfolio*** *have dependencies to the* ***OpenAI*** *and* ***Mail Client*** *classes.*

3.10 – User Interface Diagrams

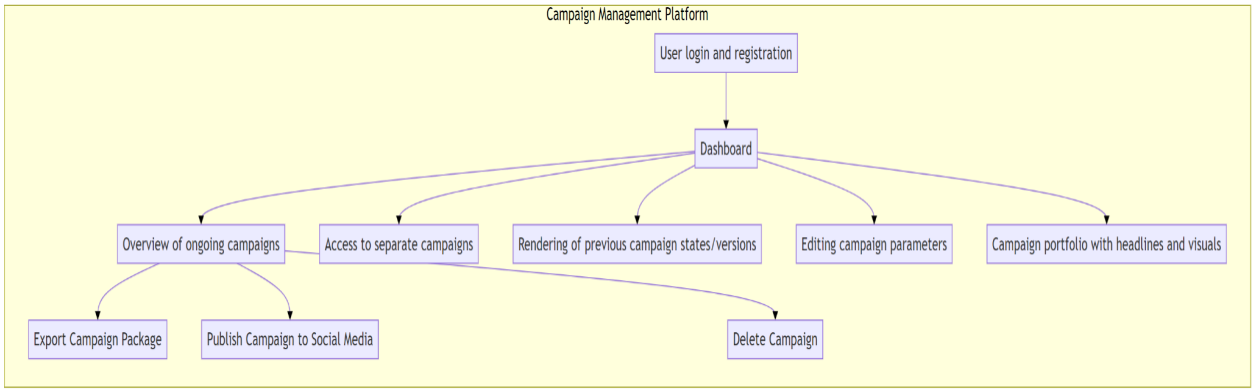

Figure 3.10.1 - the diagram above displays the User Interface Diagram for the Campaign Management workflow in the application. From the dashboard, the user will be able to manage multiple simultaneous campaigns and access social media posting capabilities.

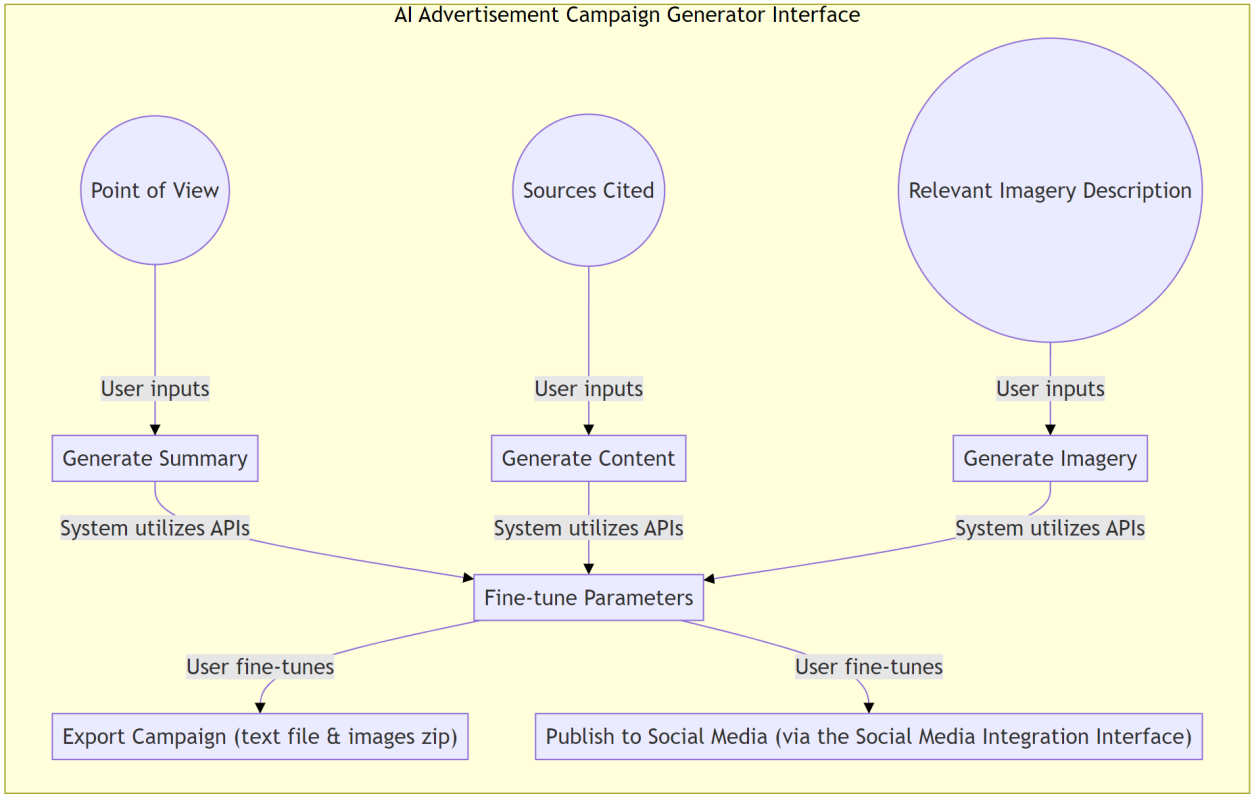

Figure 3.10.2 - the diagram above displays the User Interface Diagram for the Campaign Generation Interface, whereby an user will detail sources to be cited, perspective, and a general direction for the image to accompany the advertisement.

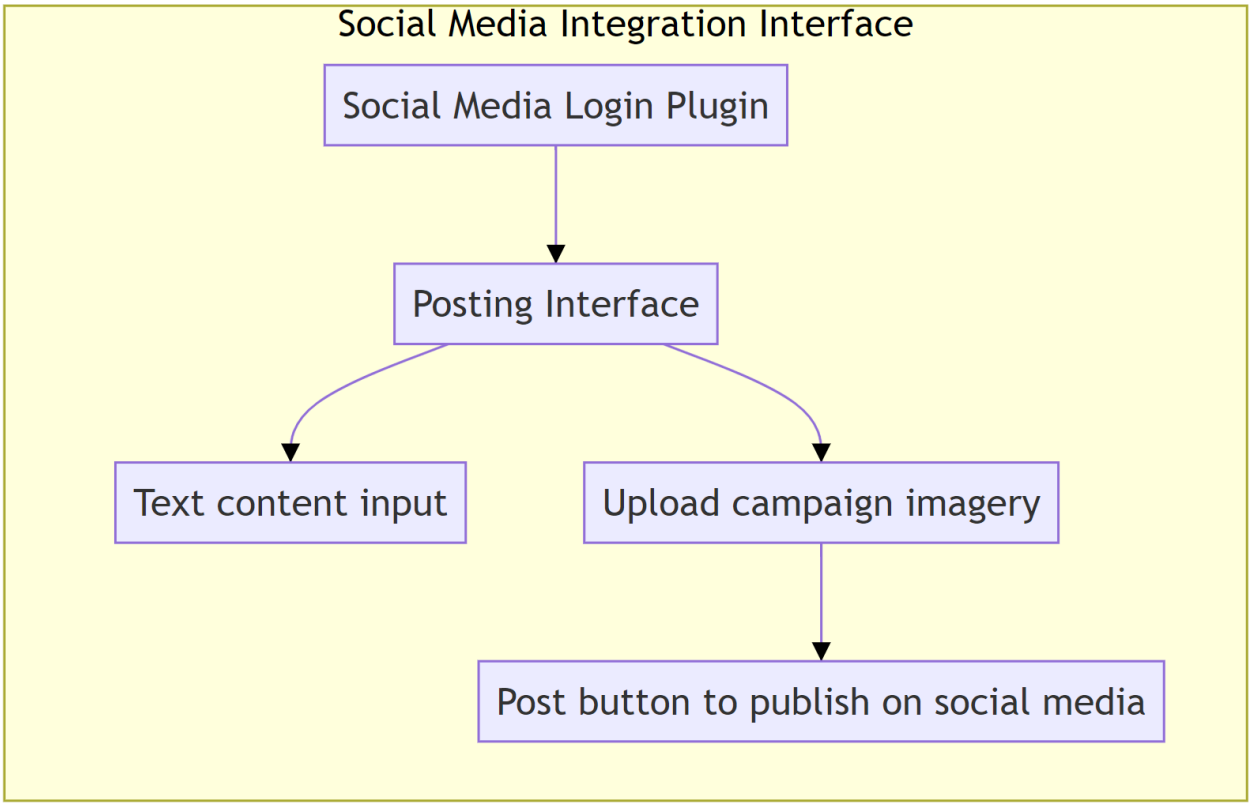

Figure 3.10.3 - the diagram above displays the User Interface Diagram for the Social Media Integration Interface, whereby an user will be able to publish campaign materials for swift dissemination.

4 – Work Plan

4.1 – Technology Stack

Python, Flask, and Django were the core of the application, and the backend forms, routes, models, and functions were implemented in Python. Flask routing handles user interaction, and handles routing logic for the application. Django provides HTML structural foundations, and works in tandem with Python and Flask to render user-content.

HTML, CSS, JavaScript, and Bootstrap were the core of the front-end, user experience component for the project. Accumulated user feedback led to changes in layout, structure, and user accessibility through CSS styling and Bootstrap component baselines. JavaScript was used to implement front-end logic such as asynchronous rendering of forms and AI-generated token streams of text and images, and enabling tooltips and popups.

Google Cloud Protocol (GCP), in tandem with CloudSQL and Google App Engine was utilized to link the application to a global, cloud-hosted database and publish as a live, cloud-hosted application. Live at https://ai-enhanced-advertisements.ue.r.appspot.com/

4.2 – Describe your plan steps/procedures to complete this project.

On the backend, we will need to have a working login manager, alongside a database to keep track of users and their advertisement campaign parameters. In order to generate campaign content we will need to establish access to three separate APIs, take the user's input and call each API using the provided input, and then output and display deliverables generated by the three APIs, compiling an interactive gallery to display the user's campaign portfolio, saving the deliverables/parameters to the database, and allowing the user to download the text/images or access them via the application at any time. For publication, the application will initially provide integration to Facebook for quick dissemination of the advertisement content. Finally, the application needs to be deployed on Google Cloud Platform.

4.3 – What skills, tools, and technologies would be needed to develop this project? And of those, which would you need to become more familiar with?

The most prominent features of our proposed project are separate APIs, each tasked with the responsibility of summarizing articles from provided links, providing a POV (point-of-view) for the summaries to be expressed from, and generating images relevant to the user's ad campaign. Each of the APIs will be implemented using Python. As a group, we will be tasked with molding the three separate APIs into one cohesive, visually appealing UI in which the user will be able to create, access, and modify their ad campaigns in one centralized location. We will also be implementing various routes, models, and a coinciding database to store users, provide functionality for ad campaign generation, and store each resulting ad campaign alongside its linked user for the ability to refer back to and either access, modify, or delete the campaign altogether. Finally, we will need to become more familiar with Google Cloud Platform's App Engine to deploy our project on the cloud.

5 – Implementation Details

5.1 - Implementation Procedure

Our initial design intent was to create an application that would bridge the gap between newly emerging AI generative technology, and the average person who may not understand its utility. Our goal meant developing an easy to use application that allowed the general public to

create and publish customizable advertisements, predicated on pre-existing generative technology.

Our development began in earnest by determining the core functionality. Not only should users be able to quickly and intuitively create advertisements, but be able to store and edit these ads. Someone who may need to generate ads for several products, and require multiple delineations of ads respectively needs to have access to functions that allow them to organizationally store, track, re-create, and re-personalize their marketing materials.

We created the skeleton of our application in the final months of our previous fall semester. This provided significant advantage upon the start of the course. From the beginning on the current semester we initially tackled the backend functionality. That is, our focus lied on ensuring the core features of our application functioned. The key feature, that which defined our application, was the integration of the OpenAI LLM APIs. Once integrating the API's, and our generations being successful, our focus shifted toward being able to save creations in the database (SQL), and then rendering a general UI for testing purposes.

We began implementing the majority of the backend functionality, to ensure our application in its current state meets most of our expectations as far as usability. Users are able to login, produce advertisement materials, and organize multiple campaigns into a folder/portfolio structure. The most key feature to be implemented is social media integration in order to export advertisements to a preferred social platform. Aside from this functionality, another key focus at this stage in development was building/refining the UI. This entailed

pouring through front-end resources and documentation in order to find components and styles that facilitate an organized and intuitive user experience, while also being visually appealing.

We have worked on creating a consistent experience across all pages to make navigation as accessible as possible. We also fully implemented Portfolio-level abstraction for campaign and portfolio management, so users can now refine campaign input parameters and output content directly on initial campaign generation page, and organize campaigns by folder/portfolio, providing additional organizational abstraction.

We also implemented various other essential API's, such as Imgur API to retain image permanence from OpenAI-generated images, because images generated by OpenAI LLMs initially had 2-hr expiration date, as well as various Facebook data-compliance functionalities, including delete user data requests, a privacy policy statement, and begun testing our application with the Facebook Graph API in order to obtain API privileges. As part of the Facebook Graph API application process, we had to expedite our timeline to deploy on GCP so we have a high-availability website for Facebook compliance review and approval. As part of the Facebook review process, we completed unit, integration, and overall user-experience testing as Facebook policy denies incomplete apps.

Additionally, in order to increase intuitiveness, we have tried to be mindful of pre-existing UX schemas that make it simple to find particular information or use particular functions. This means having an organizational structure that is widely used across most applications users would be familiar with, which mitigates any sort of "learning curve" for using

our software. We underwent a complete UI/UX overhaul through styling all CSS components, establishing a consistent theme through UX, creating interactive JavaScript functions including load-icons/display, implementing blur styling and interactive card pop-out, improving accessibility to routes and features within the application, and optimizing user experience throughout application interaction.

While implementing the UI/UX overhaul, we reached out to current users, as well as friends, family, and colleagues, to provide feedback regarding UI optimization. User-provided feedback was the driving factor for many of the UI design decisions, and led to additional features such as providing tooltips to the user for better understanding of the application, implementing asynchronous form rendering to prevent unnecessary page routing and redirects, and providing users with multiple ways to access application features, such as signing in/signing out, adding/deleting campaigns and portfolios, changing user password, and deleting user data. Text and cards throughout the application were also configured to be higher-contrast, so it would be easier to read and therefore more accessible to application users.

5.2 - Tools & Technologies

Various tools and technologies were used throughout the development process, and each aided in final user experience in one way or another.

Python, Flask, and Django were the core of the application, and the backend forms, routes, models, and functions were implemented in Python. Flask routing handles user

interaction, and handles routing logic for the application. Django provides HTML structural foundations, and works in tandem with Python and Flask to render user-content.

HTML, CSS, JavaScript, and Bootstrap were the core of the front-end, user experience component for the project. Accumulated user feedback led to changes in layout, structure, and user accessibility through CSS styling and Bootstrap component baselines. JavaScript was used to implement front-end logic such as asynchronous rendering of forms and AI-generated token streams of text and images, and enabling tooltips and popups.

Google Cloud Protocol (GCP), in tandem with CloudSQL and Google App Engine was utilized to link the application to a global, cloud-hosted database and publish as a live, cloud-hosted application. Live at https://ai-enhanced-advertisements.ue.r.appspot.com/

5.3 - Implementation Timeline

December 1-15: Researching and deciding the best APIs for our proposed project.
December 15-22: Division of labor, assigning each group member their project role.
December 22-January 5: Holiday Recess
January 5-19: Develop Framework for project (linked DB, functional routes/models).
January 19-February 9: Acquire API keys, implement APIs into project, test API functionality.
February 9-March 1: Deadline for working ad campaign generation, focus shifts to UI/UX.
March 1-29: Implement visually appealing, functional UI to enhance user experience.
March 29-April 19: Upload app to GCP, ensure a seamless transition from local machine to cloud-based web app.
April 19-April 23rd: Implement social media publication compliance and functionalityt (ex. ‘Uploading each ad campaign as a social media post to Facebook’)

6 – Test Strategy

6.1 - Unit Test Plan

We will ensure that each module or block of code functions as expected in isolation via the following approach:

Plan:

- **Campaign Generation Module:**
    - Test the functionality of AI Summarization API integration.
    - Verify the accuracy of campaign parameter handling functions.
    - Validate the creation of advertising text content based on user input and API responses.
- **Campaign Management Module:**
    - Test user authentication and profile registration.
    - Validate campaign editing and deletion functionalities.
    - Verify the retrieval and rendering of previous campaign states/versions.
- **Social Media Integration Module:**
    - Test Graph API interface functionality.
    - Validate the posting interface integration with Facebook.
    - Ensure successful publication of campaign materials to Facebook.

Execution:

- **Campaign Generation Module:**

    AI Summarization API Integration:

- Verify that the AI Summarization API is successfully integrated into the campaign generation module.
- Test various scenarios to ensure accurate summarization of campaign parameters.
- Validate the integration by comparing the AI-generated summaries with expected results.

Campaign Parameter Handling Functions:

- Test different input scenarios to verify the accuracy of campaign parameter handling functions.
- Ensure that the module handles edge cases and invalid inputs appropriately.

Creation of Advertising Text Content:

- Test the functionality of creating advertising text content based on user input and API responses.
- Verify that the generated content aligns with user preferences and API-generated summaries.

- **Campaign Management Module:**

User Authentication and Profile Registration:

- Test user authentication process to ensure secure access to the system.
- Validate profile registration functionality by registering new users and verifying their data storage.

Campaign Editing and Deletion:

- Test the ability to edit and delete campaigns.

- Ensure that changes are reflected accurately in the database and UI.

Retrieval and Rendering of Previous Campaign States/Versions:

- Verify the retrieval of previous campaign states or versions.
- Test the rendering of historical data to ensure consistency and accuracy.

- **Social Media Integration Module:**

Facebook Login Plugin Functionality:

- Test the functionality of facebook login via Graph API interface.
- Ensure seamless authentication with Facebook Graph API.

Posting Interface Integration:

- Validate the integration of posting interface with Instagram or Facebook.
- Test posting various types of campaign materials (text, images, videos) to ensure compatibility.

Publication of Campaign Materials:

- Test the publication process to Facebook.
- Verify that campaign materials are successfully published and accessible to the target audience.

Outcome:

- The Unit Test Plan was executed systematically, covering each module comprehensively.
- Issues or bugs encountered during testing were documented and discussed as a team for resolution.
- Test cases that passed validation confirmed the robustness and reliability of the system's functionalities.

- Any discrepancies between expected and actual outcomes were noted for further investigation and refinement of the system.
- Overall, successful execution of the Unit Test Plan ensures the quality and functionality of the campaign management platform across its various modules.

6.2 - Integration Test Plan

The following is our strategy to validate the interactions between different modules or components of the system.

Plan:

Overview

- Test the end-to-end flow from user input through AI processing to campaign generation.
- Validate data consistency between the user interface and the backend database.
- Verify seamless integration between the Campaign Management Platform and Social Media Integration Interface.

Integration between Campaign Generation and Campaign Management:

- Test the flow of campaign parameters from the generation interface to the management platform.
- Validate the storage and retrieval of campaign data in the database.

Integration between Campaign Management and Social Media Integration:

- Test the integration of social media login credentials with the campaign management platform.

- Validate the seamless posting of campaign materials to social media platforms.

End-to-End Integration:

- Test the complete flow from campaign generation to publication, ensuring data consistency and integrity throughout.

Execution:

Integration between Campaign Generation and Campaign Management:

- Input: Enter campaign parameters in the generation interface.
- Action: Trigger campaign generation and observe the flow of parameters to the management platform.
- Validate: Check the storage and retrieval of campaign data in the backend database.
- Outcome: Ensure that campaign parameters are accurately transferred and stored in the database without loss or corruption.

Integration between Campaign Management and Social Media Integration:

- Input: Connect facebook login credentials in the campaign management platform.
- Action: Attempt to post campaign materials to social media platforms.
- Validate: Verify that the posting process is seamless without errors.
- Outcome: Confirm that campaign materials are successfully posted on Facebook without any discrepancies.

End-to-End Integration:

- Input: Initiate a complete campaign generation process.
- Action: Generate a campaign, manage it through the platform, and attempt publication.

- Validate: Monitor the flow from generation to publication, ensuring data consistency and integrity.
- Outcome: Ensure that the entire process, from campaign generation to publication, is smooth, with no data inconsistency or integrity issues.

Outcome:

Integration between Campaign Generation and Campaign Management:

- Result: The flow of campaign parameters from generation to management is successful.
- Outcome: Campaign data is accurately stored and retrieved from the backend database, ensuring data integrity.

Integration between Campaign Management and Social Media Integration:

- Result: Facebook login credentials are integrated with the campaign management platform.
- Outcome: Campaign materials are posted seamlessly to Facebook platforms without errors or interruptions.

End-to-End Integration:

- Result: The complete flow from campaign generation to publication is tested.
- Outcome: Data consistency and integrity are maintained throughout the process, ensuring a smooth end-to-end experience for users.

6.3 - System Test Plan

Plan:

- Verify all features and functionalities as described in the requirements.
- Test user authentication, campaign creation, editing, deletion, and social media publication.

Usability Testing

- Assess user interface intuitiveness and ease of navigation.
- Test user interactions with various input types for campaign generation and management.
- Simulate multiple users concurrently accessing and managing campaigns.

Performance Testing

- Evaluate system response times for generating campaigns, accessing the management platform, and publishing to social media.
- Evaluate the application's ability to recover gracefully from database failures or API outages.
- Test system scalability under different load conditions.

Security Testing

- Ensure data protection measures, including encryption of sensitive information and secure communication protocols.
- Test authentication mechanisms to prevent unauthorized access.

Recovery Testing

- Test the system's ability to recover from failures or errors gracefully.
- Validate data backup and recovery procedures to ensure minimal data loss.

Potential Tools and Technologies Required:

- Python testing frameworks like unittest, pytest for unit testing.
- Integration testing tools like Postman for API testing.
- Load testing tools like JMeter for performance testing.
- Security testing tools like OWASP ZAP for vulnerability scanning.

Execution:

Functional Testing:

- Input: Review requirements and test cases for each feature and functionality.
- Action: Perform authentication, campaign creation, editing, deletion, and social media publication.
- Validate: Ensure that each feature behaves as expected according to the requirements.
- Outcome: Confirm that all features and functionalities are functioning correctly without any deviations from the requirements.

Usability Testing:

- Input: Access the user interface and navigation elements.
- Action: Interact with the interface to assess intuitiveness and ease of navigation, from portfolio creation to campaign edits and publication.
- Validate: Test various input types for campaign generation and management.
- Outcome: Evaluate user interactions and determine if the interface is user-friendly and intuitive, especially under scenarios involving campaign creation and management by multiple users simultaneously.

Performance Testing:

- Input: Simulate typical user actions, such as generating campaigns, accessing the management platform, and publishing to social media.
- Action: Measure system response times under different load conditions.
- Validate: Evaluate the application's ability to handle database failures or API outages.
- Outcome: Assess system scalability and performance under varying loads, ensuring acceptable response times and graceful recovery from failures.

Security Testing:

- Input: Review security measures implemented in the system, including data encryption and authentication mechanisms.
- Action: Test data protection measures and authentication mechanisms, introducing common security threats such as SQL injection.
- Validate: Ensure secure communication protocols and prevent unauthorized access.
- Outcome: Confirm that sensitive information is encrypted, and the system is resilient against security threats, maintaining the integrity and confidentiality of user data.

Recovery Testing:

- Input: Introduce failures or errors to the system.
- Action: Trigger failures and errors to test the system's recovery mechanisms, such as not saving a campaign after campaign generation to assess system’s behavior.
- Validate: Verify data backup and recovery procedures to minimize data loss.

- Outcome: Ensure that the system can recover gracefully from failures, maintaining data integrity and minimizing disruptions to user activities.

Outcome:

Functional Testing:

- Result: All features and functionalities meet the requirements.
- Outcome: The system behaves as expected, fulfilling user needs and expectations.

Usability Testing:

- Result: The user interface is intuitive, and navigation is smooth.
- Outcome: Users find it easy to interact with the system, even when managing campaigns concurrently.

Performance Testing:

- Result: System response times meet acceptable benchmarks under varying loads including multiple simultaneous users accessing cloud hosted URL.
- Outcome: The system demonstrates scalability and resilience, maintaining performance during peak usage and recovering gracefully from failures.

Security Testing:

- Result: Data protection measures and authentication mechanisms are effective.
- Outcome: The system ensures the security and privacy of user data, preventing unauthorized access and maintaining confidentiality.

Recovery Testing:

- Result: The system recovers smoothly from failures with minimal data loss.

- Outcome: Data backup and recovery procedures are reliable, ensuring continuity of operations in the event of failures or errors.

7 – Discussion

7.1 Unresolved Issues and Limitations

The key issues and limitations we experienced during development of this application were social media integration and a more realized user interface.

7.1.1 - Social Media Integration

A primary goal of our application was not only for users to be able to create personalized advertisements, but to share them on social media platforms. Naturally, if an ad is created and cannot be seen by others, there is no value. This, however, proved substantially more difficult than expected. Due to a lack of experience and understanding of how necessary social media API's work in practice, we did not foresee how arduous it would be to create this functionality. In order to use respective API's (i.e. Facebook, Instagram, Twitter), strict requirements needed to be met. Such requirements included having a documented business and paying excessive amounts of money. Twitter specifically was $100 per month which we were not prepared to pay given this was a school project and not a commercially deployed application. Should we find a way to monetize our application, publication to Twitter would be feasible. In regard to Facebook and Instagram, our issue lay in the fact that you needed to have an established business LLC and

articles of incorporation. Given the time limitations of our project we could not reasonably create an LLC to procure this API integration.

7.1.2 - Application Deployment

In order to gain access to the Facebook Graph API, it was necessary to have the application deployed and live 24/7. In previous courses, we learned how to deploy applications using GCP on a Google Compute Engine instance (VM), and then SSH into the client and deploy the code onto it, however this solution would not meet the expectations of the Facebook API criteria. Thus, we began research into other means of deployment, and came across the Google App Engine.

The Google App Engine provides an easily scalable, and (initially) free method of application deployment, which would resolve the majority of our issues. Google App Engine provided many pros compared to a Compute Engine instance, namely it was free, it routed traffic through an HTTPS-secure webpage, and it maintained 24/7 uptime without needing to directly interact with the virtual host machine through SSH. With that being said, implementing Google App Engine was much easier said than done.

Hours were spent referencing Google App Engine deployment documentation, and results were ambiguous at best, and contradictory at worst. Eventually, the necessary app.yaml file was created, the Google Cloud package was installed locally, and deploying the application was as simple as a “gcloud app deploy” within the terminal.

Numerous redeployments were necessary to handle runtime configuration issues within the app.yaml, as well as manually modifying the app “staging” bucket to include a /tmp directory to write new credentials to the API tokens, however in due time, the application was live and the re-deployment cycle was as easy as the initial “gcloud app deploy”.

7.1.3 - Asynchronous Render of AI-Generated Tokens

Another major issue we encountered was how our text generated upon creating a campaign. Our goal was to show a progressive rendering of text, similar to how ChatGPT functions. That is, we wanted the text created to render letter by letter and not all at once, simulating the experience of an AI sentience responding to your requests. This proved difficult within App Engine deployment.

While local machines will open the event stream, render the text token-by-token, and then subsequently close the stream, Google App Engine prevents rendering the event stream until the stream is closed, so all text and images render at the same time in a block once the event stream is closed. This was discovered by Jonathan through monitoring the browser’s developer tools and watching the flow of network connections rendered while the page was active. Once all tokens were in queue, and the “end-of-stream” message was sent to the server, only then did the text and image properly render on the page. While the text was properly being sent from the server and the API to the client, this did not convey the “excitement” users experience when initially working with LLM’s and seeing text be generated token-by-token.

Clearly, App Engine protocol would prevent us from rendering content token-by-token, as we had initially hoped. However, Jonathan came up with the ingenious workaround of bypassing the initial token render and instead waiting for the event stream to close, then once the stream was closed, use asynchronous JavaScript Timeout callback function to manually render the text character-by-character, which would be called once the stream was closed. This solution proved to be effective, and provided users with the experience of "live" generation, ensuring the "excitement factor" remained within the application, albeit leading to a longer wait time in rendering materials as we wait for the token stream to conclude.

7.1.4 - Image Regeneration

Once text was properly rendering token-by-token, and initial generation was functional, the next step was to expand functionality to provide re-generations and allow the user to directly alter the generations. While text regeneration was straightforward, simply taking feedback and making another call to the OpenAI API, image regeneration provided a much more nuanced approach.

Dall-E 3 did not allow for image manipulation, and Dall-E 2 allows for a simple image masking function, however this did not meet our needs. The user would need to provide a .png mask to apply onto the image as an overlay, which would ultimately render the text feedback useless and require the user to do extra, unnecessary work when trying to create an advertisement. It was clear that neither of these models would allow for the image regeneration functionality we had hoped for.

Jonathan proceeded to research different OpenAI models, and managed to find GPT-4.0, the upgraded version of GPT-3.5-turbo, the base text-generation model, which conveniently allowed for computer vision functionality. The final implementation consisted of the generated image being fed into GPT-4.0 and described in text, then concatenating the AI image description with the user-provided feedback as a new prompt to Dall-E 3, thus generating a new image.

7.1.5 - UI Optimization Through User Feedback

We did not have enough time to fully realize our UI. While it is serviceable, there is substantial room for improvement. We want our application to easy to use for relevant commercial demographics. This primarily means small business owners. Considering the vast majority of people are not familiar with AI generation, let alone the application of such technology, means it is our duty to make the most intuitive application possible.

Going forward, we would like to update the UI extensively through the use of a front-end framework, such as React JS. React, in tandem with cutting-edge CSS libraries such as TailwindCSS, ShadCN, and AceternityUI. The front-end development space has expanded rapidly, and through the use of modern libraries and frameworks our application can benefit greatly in terms of UI/UX.

7.2 - Future Direction and Further Discussion

We would like to gestate on our work in the coming weeks and determine what could be added (or removed) from our application to achieve the most optimal design and performance.

All of us unanimously desire to continue work in our designated roles on this assignment as we have become dramatically invested in its outcome.

The main goal would be to make our application commercially viable. This means integrating the social media API's primarily. As previously stated, our application lacks much value without the ability to publish user-created ads for others to see. In order to achieve this, we need to find some form of monetization. This can perhaps be based in a subscription model, or possibly through ads within the pages. Monetization would not only cover the cost of the social media API's, but more importantly, the OpenAI API calls every time a user generates content. Each generation costs us approximately 5-7 cents per call. Scaled up for commercial use, this would not be sustainable, especially considering users are expected to create several generations per product.

Another important goal is to solidify a more user friendly UI. While we maintain that our UI is serviceable, it is insufficient. Aspects such as feedback, contrast, and tutorials should all be further implemented to create as natural of an application as possible. For example, card components within our pages blend in with the background, we acknowledge this and will work to create a contrast between foreground and background elements. Other pages, particularly the account page and generation page require a cleaner formatting. We believe if these aspects of UI are further developed, the overall user experience is elevated.

Tutorials and tooltips need more representation as well. From our rudimentary user testing, we found that non-technical demographics struggled to understand what/ how the

application did/ worked. While we did implement some basic tooltip icons for further explanation on particular functions on a given page, we would ideally like to implement a guided tutorial for first time users. This would entail the application recognizing a newly created account, and walking through the ad generation/management process step by step in an intuitive way to ensure users know precisely what they are doing. Creating a more accessible application would mean a larger user-base. As previously stated, if we intend to make this application commercially viable, it is necessary to make the application as easy to use as possible, especially considering our target demographic is most likely not advanced in technological literacy.

8 – Contributions Breakdown

Jonathan Warren - *Lead Engineer*

As the Lead Engineer and Architect on the project, Jonathan's contributions spanned:

- Overall system architecture design, development, and implementation.
- Database design & backend application logic.
- Google Cloud Platform deployment & management.
- UI Component design and development - feeds, carousels, and recursive campaign edit history rendering
- Computer vision implementation using DALL-E 3 for image regeneration with user feedback.
- IMGUR, Gmail, & Facebook Graph API Integrations.

Diego Aguilar - *Project Manager/Backend Developer*

As Project Manager & Backend Developer, Diego's contributions were:

- Delivered direction toward intermittent goals, prioritizing timelines, deadlines, and sprint deliverables.
- Proof of concept ideation, OpenAI design, logic, integration, and development.
- System architecture, direction of production, database design, and SCRUM lead.
- Create, edit, and regenerate campaign routes.
- Graph API integration and Facebook publication compliance including Terms of Use, Privacy Policy, and user data deletion routes.

Daniel Rapuano - *Frontend Developer / UX Designer*

As Front-End UI/UX Developer and QA, Dan's contributions were:

- Created and conceptualized user-facing elements.
- Established focus on user accessibility tooltips and experience
- Conducted user-testing and experience feedback loops with with real-world demographics.
- Delivered front-end designs, themes and components.
- Homepage design, About, and Developers routes.
- Wrote literature and documentation deliverables.

Responsibility Allocation Chart

Figure 8.1 - The diagram below, in RACI Matrix form, displays the responsibility breakdown across main deliverables for the project:

Task Description	[R]esponsible	[A]ccountable	[C]onsulted	Status
Task Name				
Proof of Concept	Diego/Jonathan	All	Daniel	
Base Architecture	Jonathan/Diego	Jonathan	Daniel	
Navigation UI	Daniel/Jonathan	Jonathan	Diego	
OpenAI API Integration	Diego/Jonathan	Diego	Daniel	
DALL-E3 Computer Vision	Jonathan	Jonathan	Diego/Daniel	
Initial Routes & Testing	Jonathan/Diego	Jonathan/Diego	Daniel	
Database Design	Jonathan/Diego	Diego	Daniel	
Database Implementation	Jonathan	Diego/Daniel	Daniel	
UI Design	Jonathan/Daniel	Diego	Diego	
UI Component Implementation	Jonathan/Daniel	Daniel	Diego	
Advanced Routing	Jonathan	Diego	Daniel	
Google Cloud Platform Deployment	Jonathan	Jonathan	Diego/Dan	
Facebook Graph API Compliance	Diego	Diego	Jonathan/Daniel	
Facebook Graph API Implementation	Diego/Jonathan	Diego	Daniel	
User Acceptance Testing	Daniel	Daniel	Diego/Jonathan	
User Experience - Tooltips	Daniel	Daniel	Jonathan/Diego	
IMGUR/GMAIL API Integration	Jonathan	Jonathan	Diego/Daniel	

9 – Conclusion

Without a shadow of a doubt, we are beyond grateful for the opportunity to work on this project throughout the Spring semester, and are undoubtedly very proud of the final result of our application. The AI-Enhanced Advertisement Campaign Generator provides both a useful tool and a fun and interactive user experience to those looking to take their advertisement campaigns to the next level.

The experience gained and the lessons learned throughout the development lifecycle have proved invaluable, and will most certainly translate into further career prospects. While on the topic of further career prospects, various group members have used this exact project to impress Fortune-500 companies and propel through interviews, and the project has been widely acclaimed by fellow classmates, department professors, even potential investors, discussing the potential to transform the application from an education experience into a full-fledged startup!

10 – References

[1]https://www.forbes.com/sites/square/2020/11/18/from-brick-and-mortar-to-click-and-order-transitioning-to-ecommerce/?sh=253702da5067

[2]https://searchengineland.com/linkedin-ai-powered-ad-creation-tool-campaign-manager-432849

[3] https://simplified.com/blog/design/ai-ad-generator/

[1] All commerce is moving online. consumers are smarter than ever before, more resourceful, and are able to find better deals and increased convenience via shopping online. Ads that are deployable on the Internet will be better received by an online shopping crowd, which will soon be the majority (if it is not the majority already).

[2] Other companies are already experimenting with AI-generated ads, with a prominent example being released in early October 2023 by Linkedin. By having a centralized platform

able to replicate Linkedin's efforts on a larger, more generalized scale, our product will be open and applicable to a much larger audience base.

[3] The domain of AI for advertisement generation is rapidly growing and expanding, becoming mainstream at a record-breaking pace. Being able to break into this field before other platforms are able to establish a strong footing in the field will prove both optimal and profitable in due time.

11 – Appendix

10.1 - Runtime instructions

The application is currently live (https://ai-enhanced-advertisements.ue.r.appspot.com/), where everyone interested is free to create an account and begin creating portfolios and campaigns! For those looking to learn more about the development process and what went into the application, the codebase is also available on GitHub

(https://github.com/JPWarr7/AI-Campaign-Manager).

To run the application on a local machine, you would need to acquire API keys to Facebook, OpenAI, and Imgur, install the required dependencies in the 'requirements.txt' file into a local virtual environment, and then type 'flask run' in the terminal.

The code is broken down between routes, forms, templates, database, and static folders. The *routes* folder contains the core logic of the application. The *forms* folder contains Python forms for data retrieval and storage. The *database* folder contains the database initiation and logic routes. The *templates* and *static* folders contain HTML, CSS, Bootstrap, and Javascript components for UI rendering.

10.2 - Snapshots

10.2.1 - Project File Structure

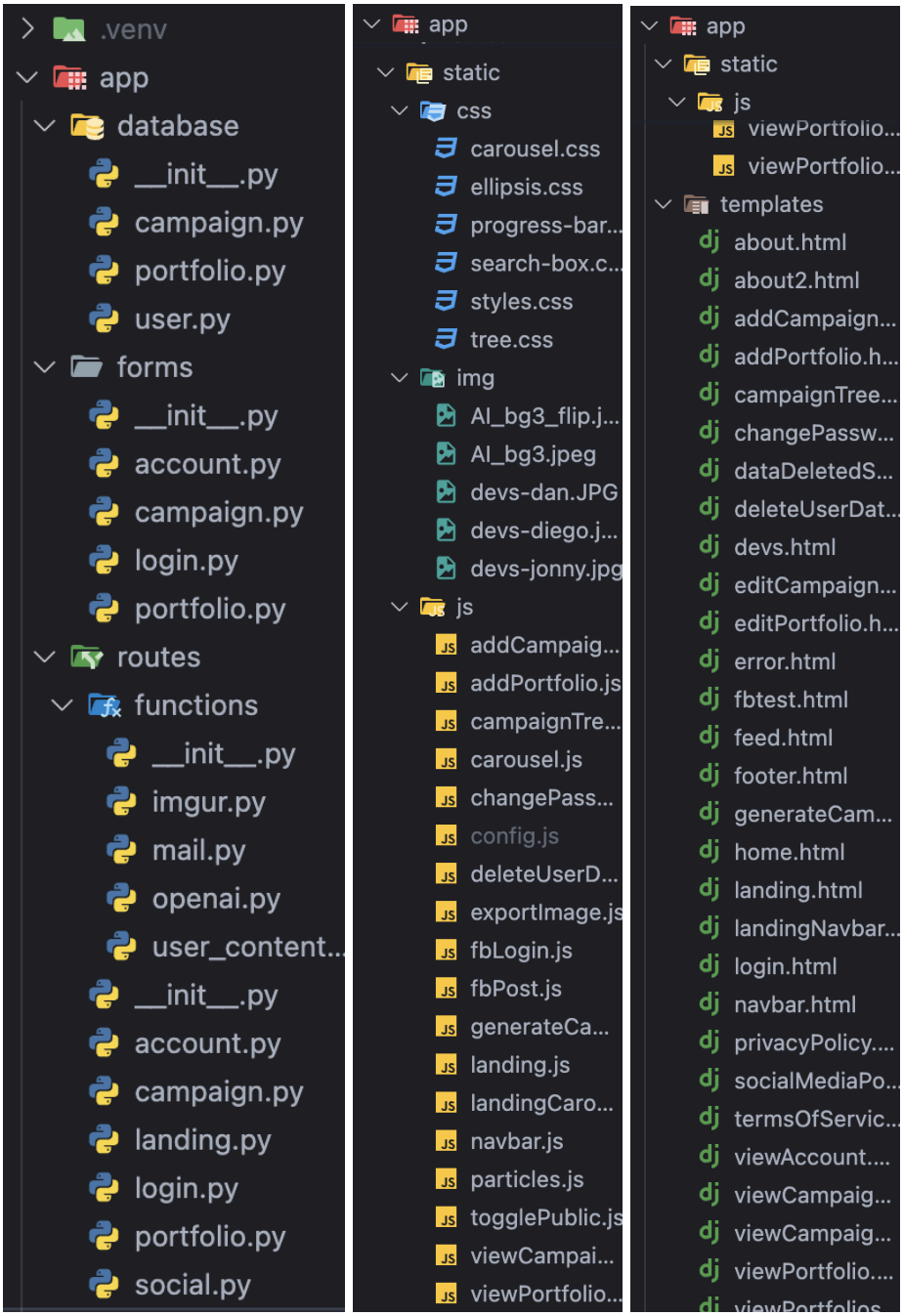

Figure 10.2.1.1 - the diagram above displays the application file structure.

10.2.2 - Functional Code Snippets

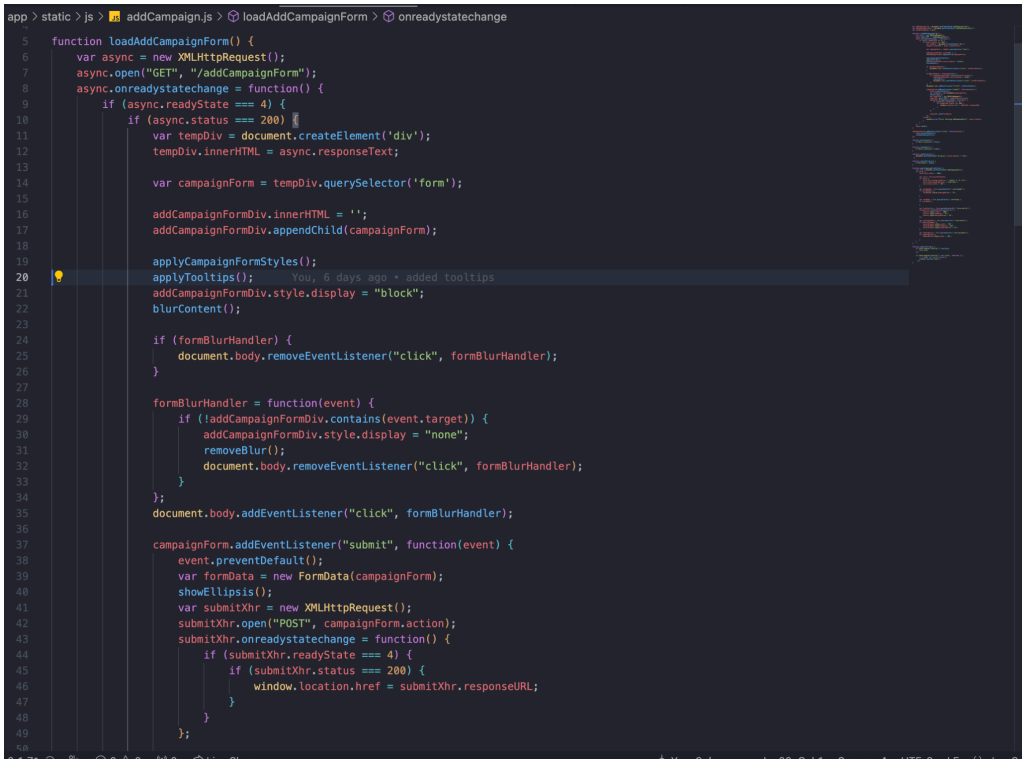

Figure 10.2.2.2 - the snippet above displays the logic for asynchronously rendering the addCampaign Form.

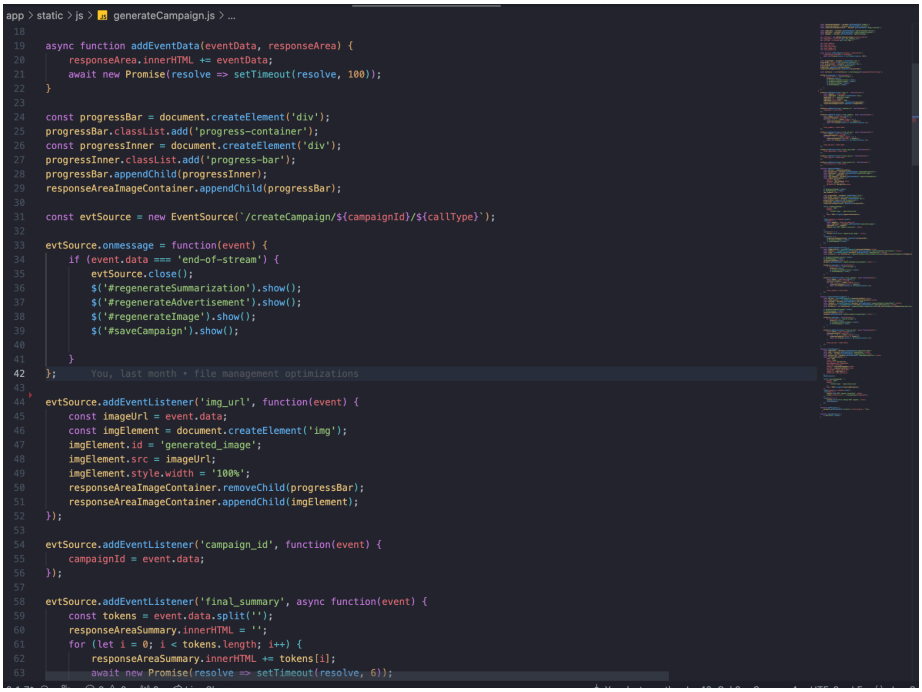

Figure 10.2.2.3 - the snippet above displays the logic for asynchronously rendering the event-stream generated by OpenAI API LLM's..

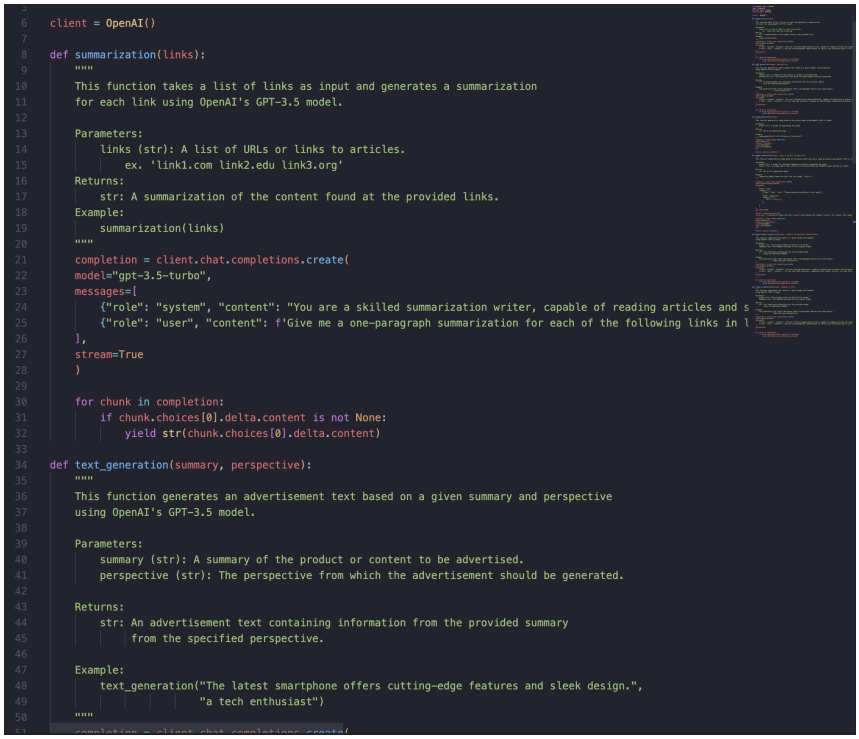

Figure 10.2.2.4 - the snippet above displays the logic for making calls to the OpenAI API, used as driver functions.

app > static > js > fbPost.js > ...

You, 6 days ago | 1 author (You)

```
import config from './config.js';

function shareToFacebook(imageUrl) {
    var campaignText = document.querySelector('.ad-text').textContent.trim();
    FB.init({
        appId: config.facebookAppID,
        version: config.facebookAPIVersion
    });

    FB.ui({
        method: 'share',
        href: imageUrl,
    }, function(response) {
        if (response && !response.error_message) {
            console.log('Post shared successfully:', response);
        } else {
            console.error('Error sharing post:', response.error_message);
        }
    });
}

document.getElementById("shareButton").addEventListener("click", function() {
    var imageUrl = document.getElementById("campaignImage").src;

    shareToFacebook(imageUrl);
});

function copyToClipboard(text) {

    var tempInput = document.createElement("input");
    tempInput.value = text;
    document.body.appendChild(tempInput);
    tempInput.select();
    document.execCommand("copy");
    document.body.removeChild(tempInput);

    var popup = document.createElement("div");
    popup.textContent = "Text copied to clipboard!";
    popup.style.position = "fixed";
    popup.style.bottom = "20px";
    popup.style.left = "50%";
    popup.style.transform = "translateX(-50%)";
    popup.style.padding = "10px 20px";
    popup.style.background = "#333";
    popup.style.color = "#fff";
```

Figure 10.2.2.5 - the snippet above displays the logic for handling Facebook API functionality, and includes the export to Facebook functionality as well as copying the advertisement text to the user's clipboard.

10.2.3 - Google Cloud Protocol Configuration

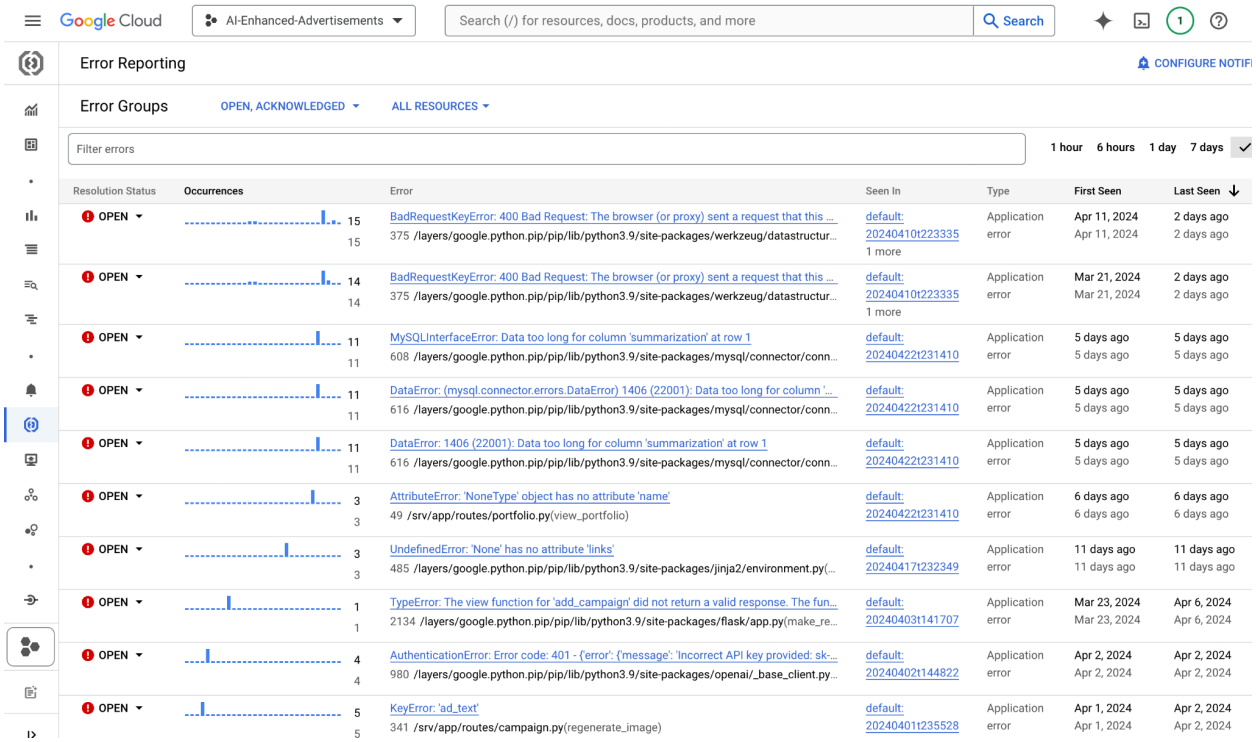

Figure 10.2.3.1 - the graph above displays the Google Error Reporting dashboard, used to debug App Engine Builds and Versions when they come across runtime errors.

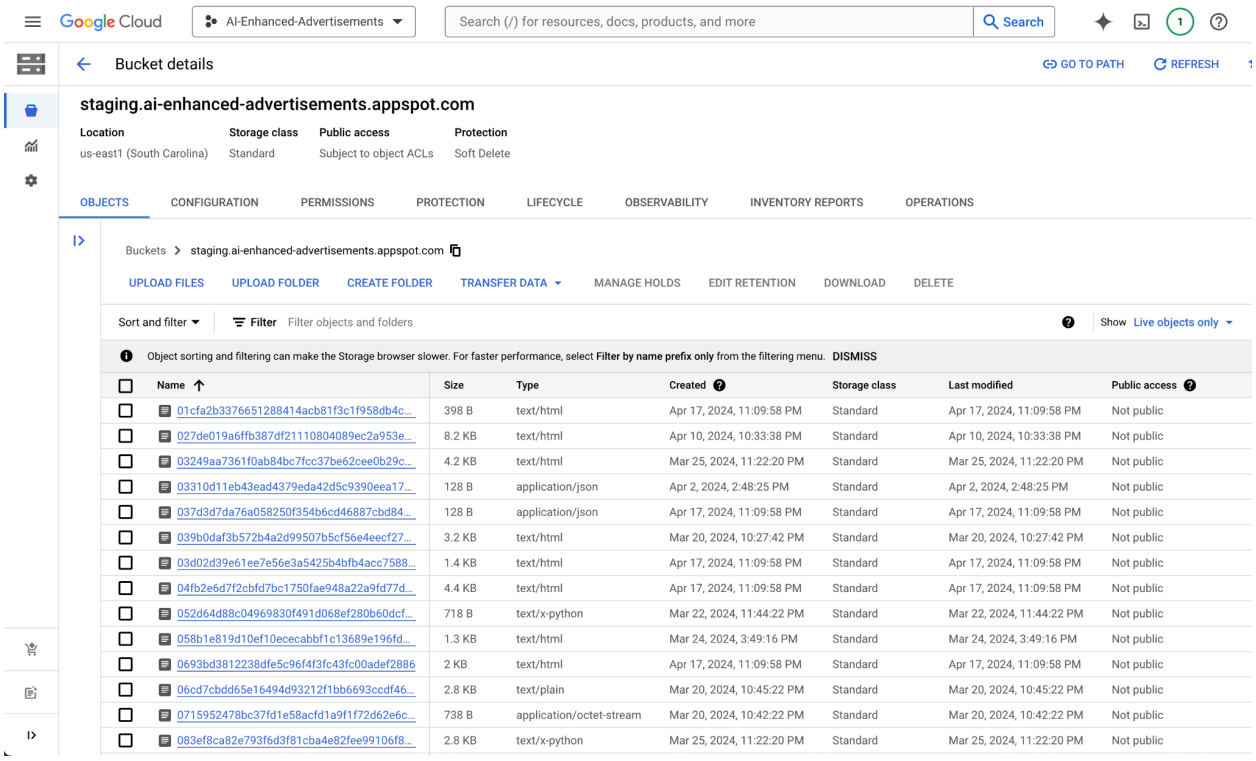

Figure 10.2.3.2 - the table above displays the "staging" bucket for the application, where files not used by the production App version are stored and referenced when configuring

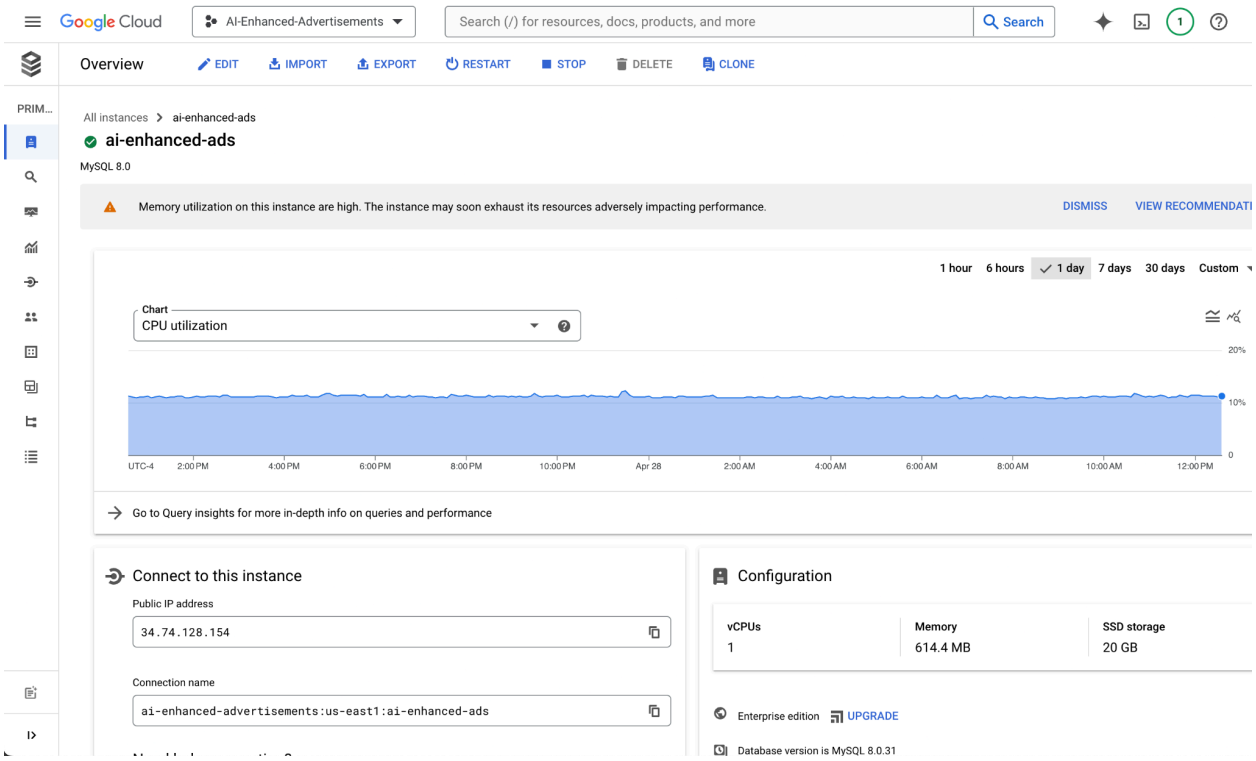

Figure 10.2.3.3 - the dashboard above represents the Google Cloud SQL instance, used to host the database for the application.

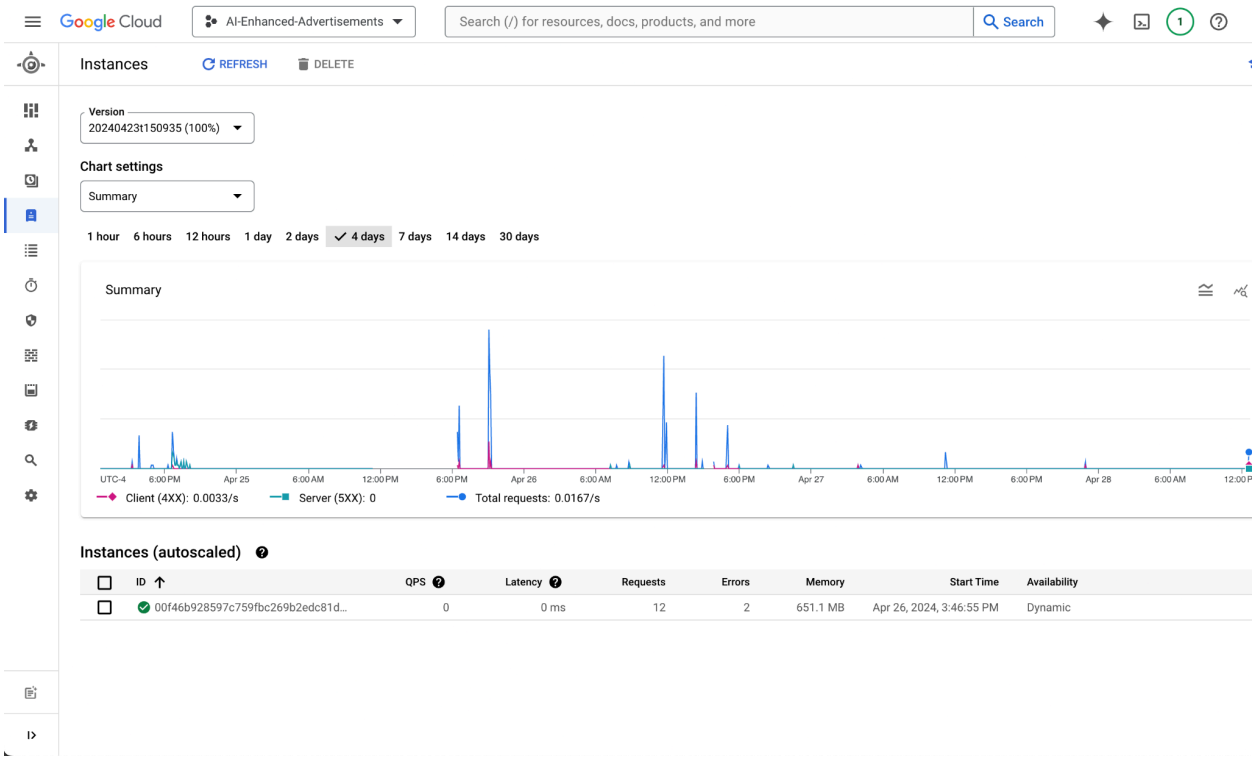

Figure 10.2.3.4 - the dashboard above represents the Google App Engine instance, used to deploy and manage different Application versions and builds, as well as allocate HTTP traffic to a particular version.

10.2.4 - Live Demo

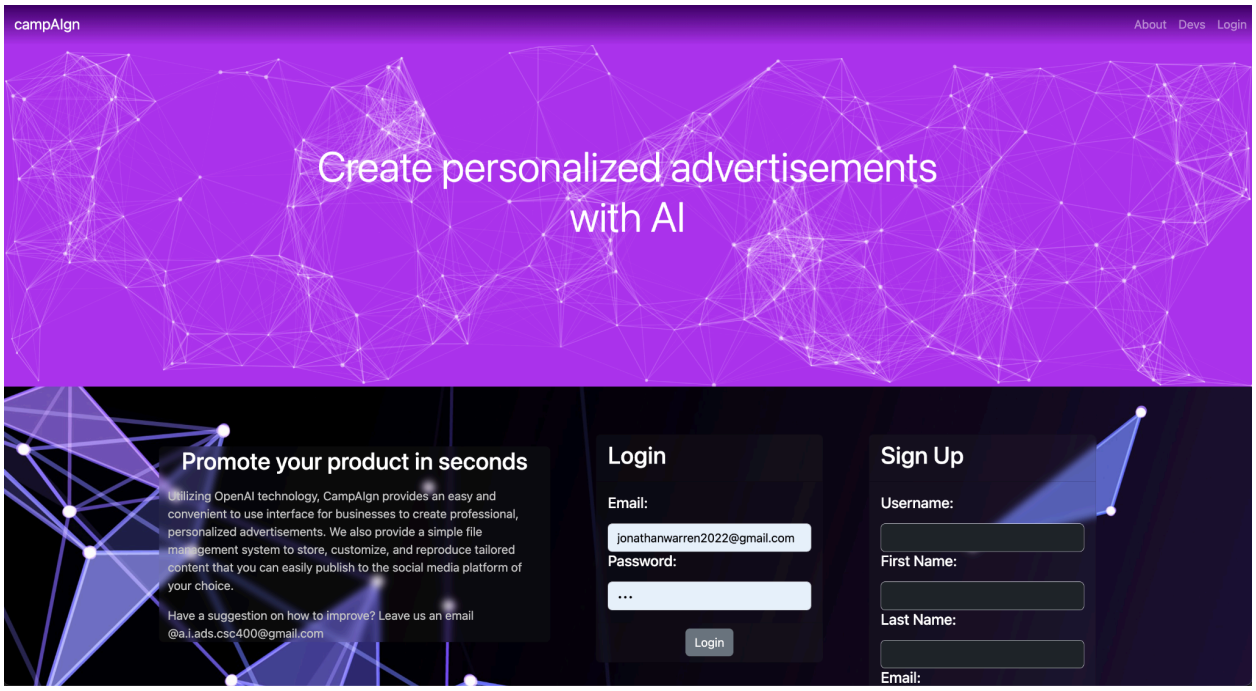

Figure 10.2.4.1 - the snapshot above displays our application landing page, from which an user can log in or access our public pages including our About, Privacy Policy, Terms of Use, and Developer pages.

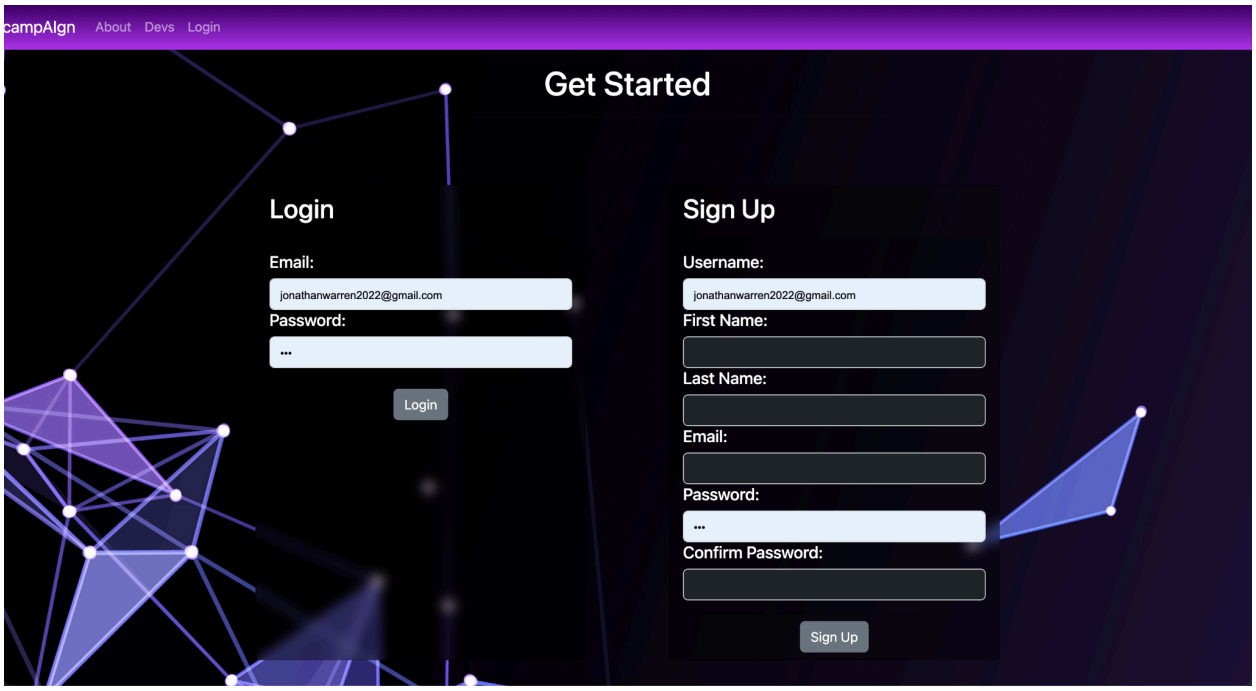

Figure 10.2.4.2 - the snapshot above displays our application login and registration. Here new users can create an account or login to a pre-existing account.

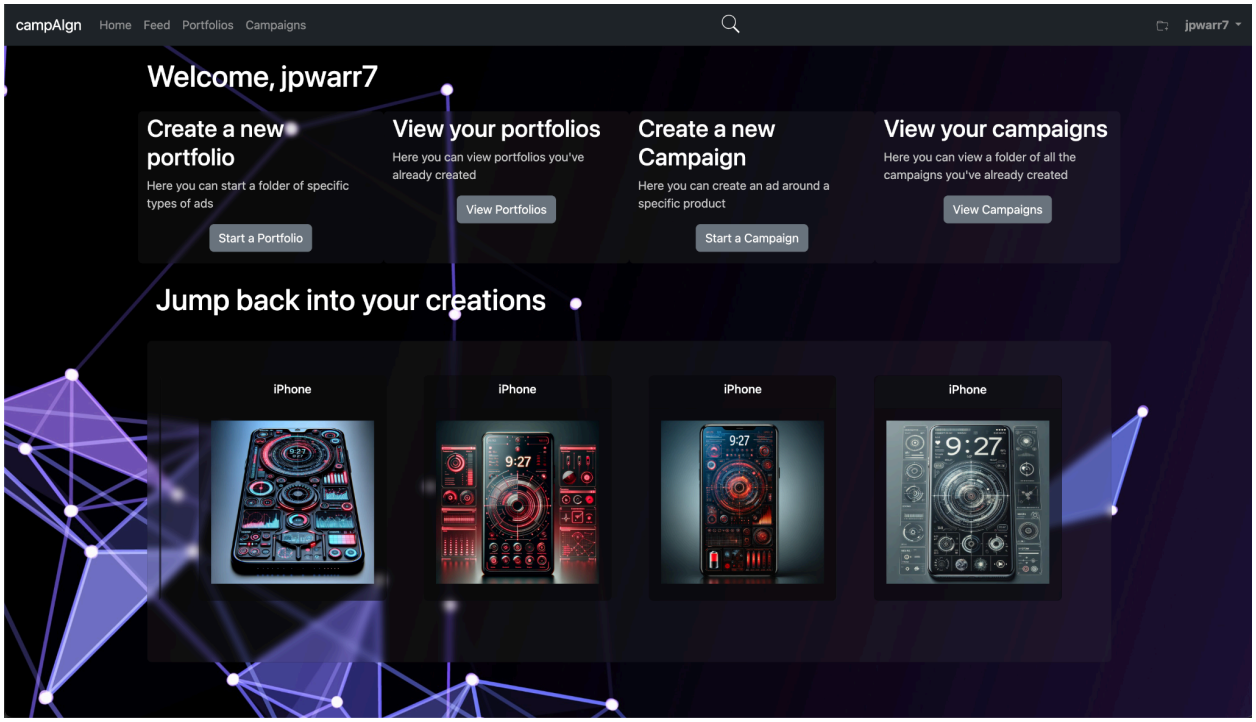

Figure 10.4.2.3 - the snapshot above displays our post-login home page page, from which an user can view/edit their portfolios, campaigns, and create new portfolios or campaigns. The page also renders a "quick view" carousel of the campaigns they have worked on in the past.

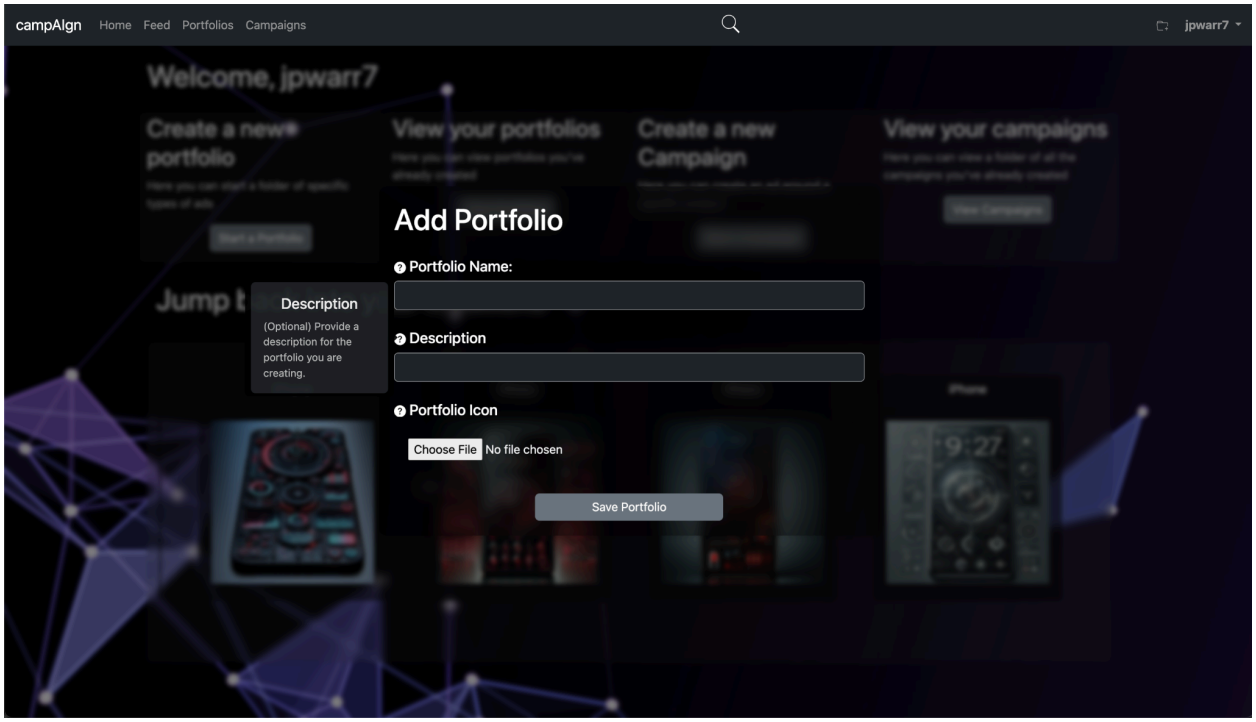

Figure 10.4.2.4 - This snapshot illustrates the add-portfolio form. This is a requirement before a campaign can be created. This feature creates a level of abstraction that organizes your ad creations into categories.

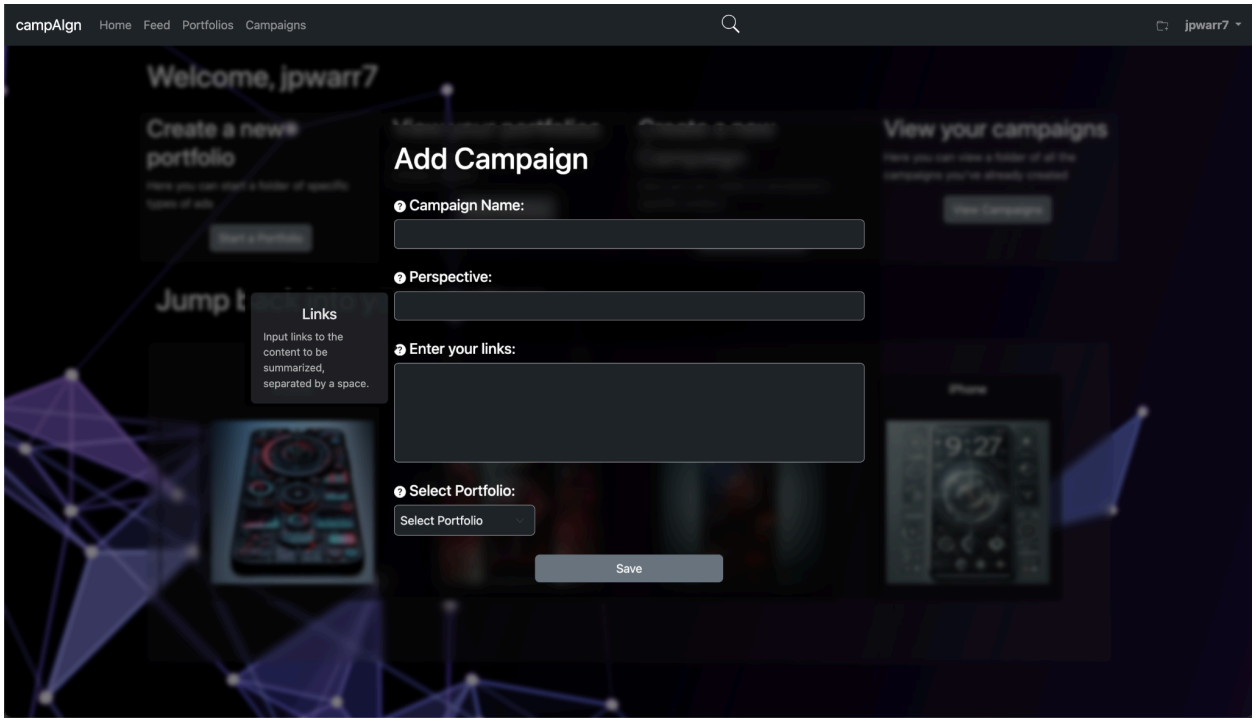

Figure 10.4.2.5 - This snapshot illustrates the add-campaign form. This is where you actually create your advertisement. The parameters for generation take in a POV (who the ad sounds like) as well as resource links to generate the summarization the text generation is based on.

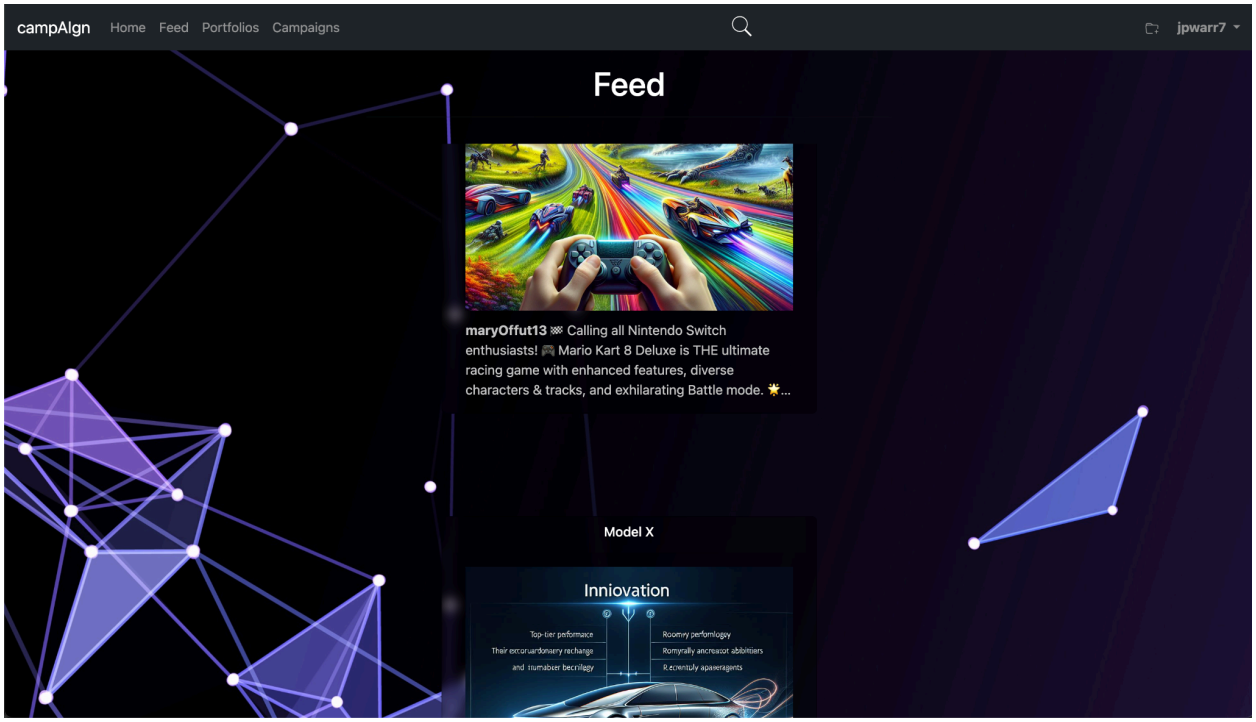

Figure 10.4.2.6 - the snapshot above displays our post-login user feed, where users can view all Public campaigns produced by their organization as well as their own. If the campaign selected belongs to the user, they will be able to immediately edit it. If the campaign does not belong to the user, they will access a view-only version of the campaign.

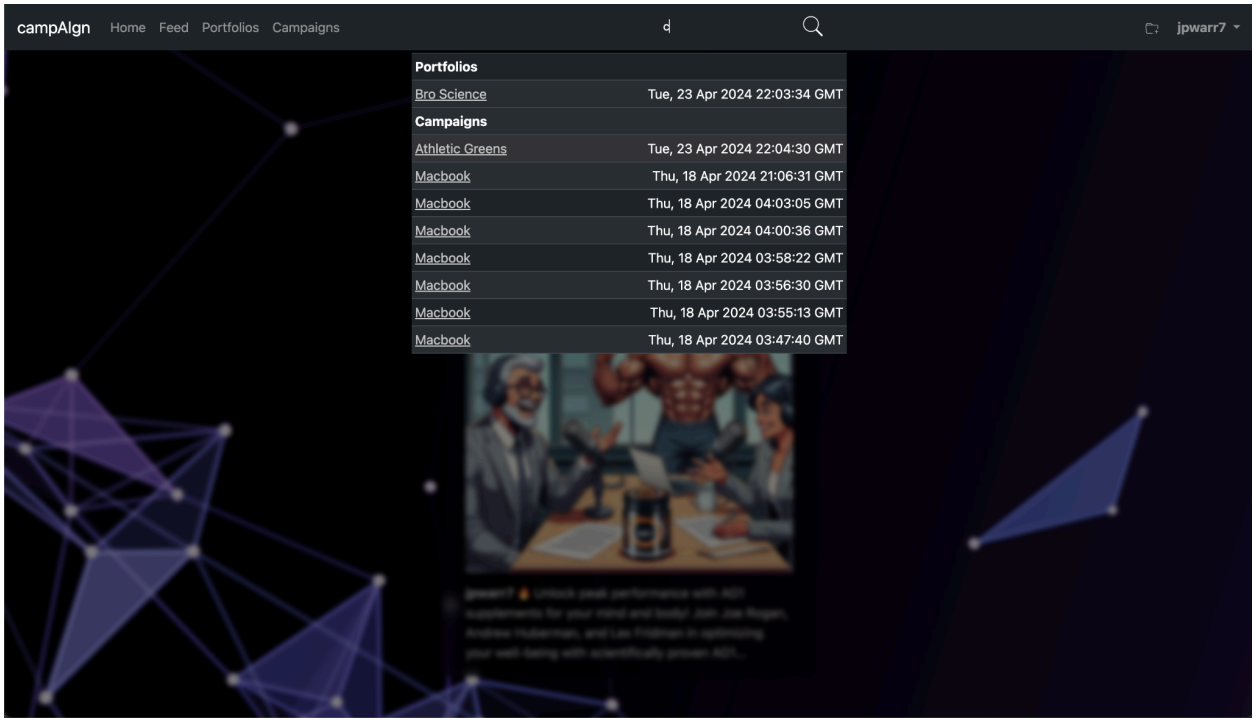

Figure 10.4.2.7 - the snapshot above displays our campaign search feature bar feature, whereby users can immediately search through their portfolio/campaigns by name..

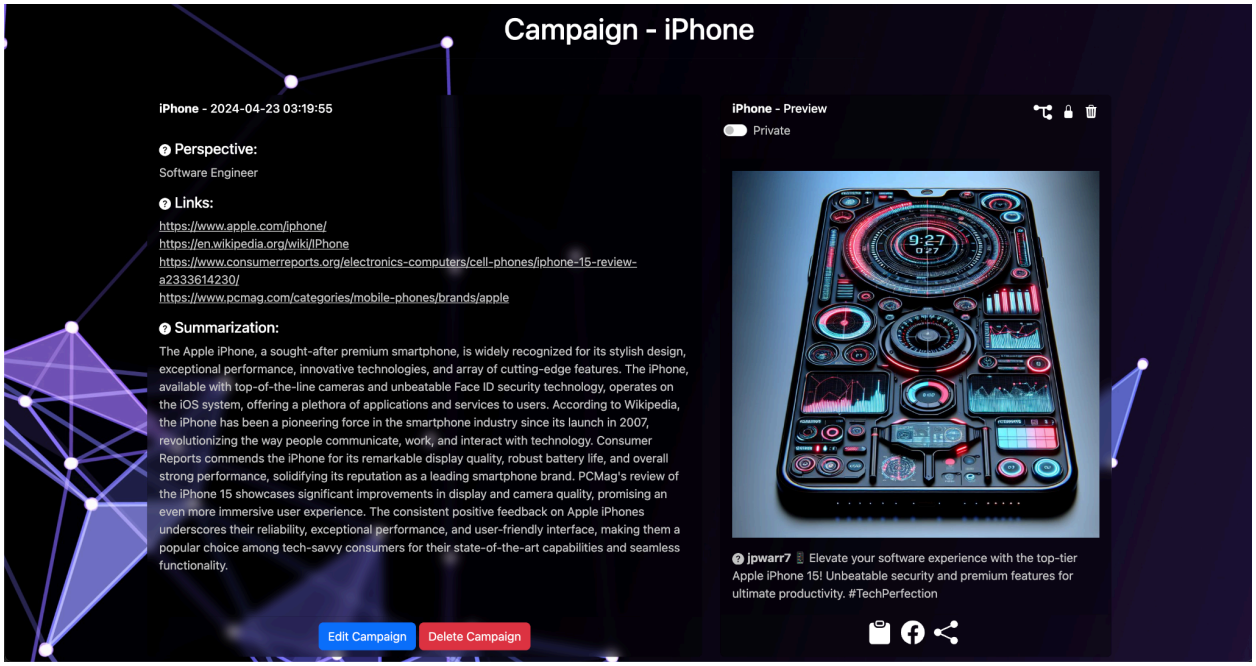

Figure 10.4.2.8 - Here we can see the generative product. This includes the summarization of links, the text generation, and the image generation. Functions to edit or delete are readily available, as are icons for social media exportation.

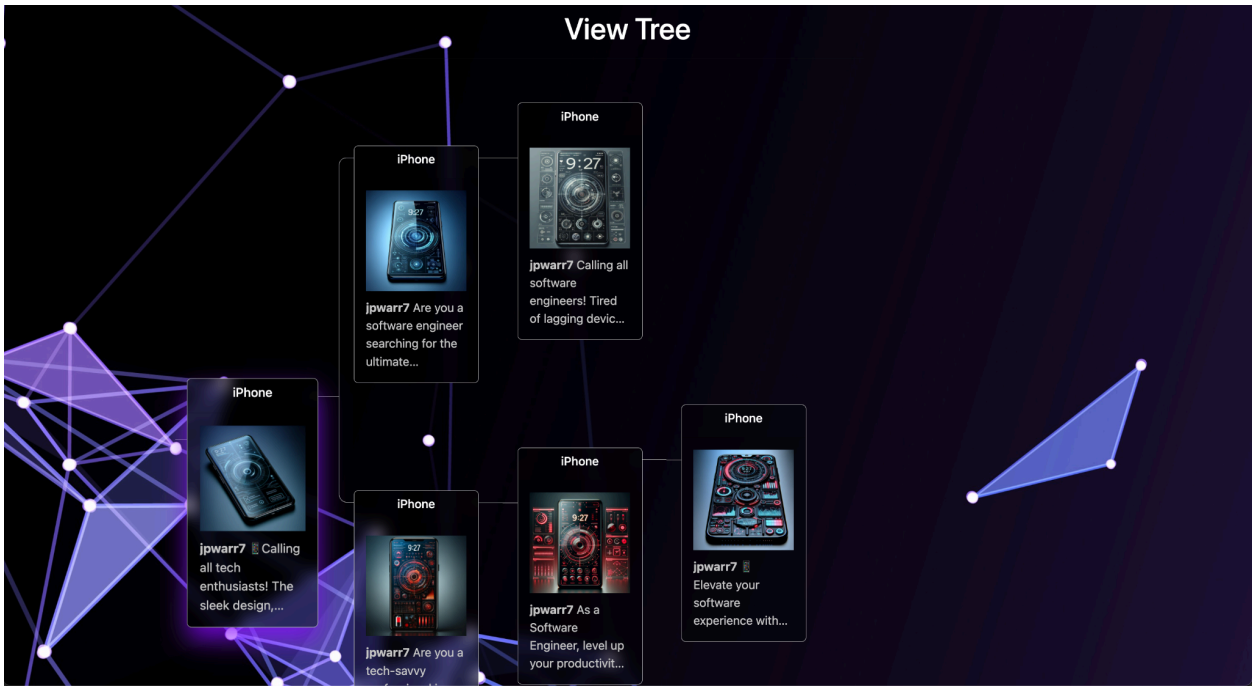

Figure 10.4.2.9 - the snapshot above displays the campaign edit history tree-view, where users can visually track the growth and iterative development stages of their campaign advertising product.

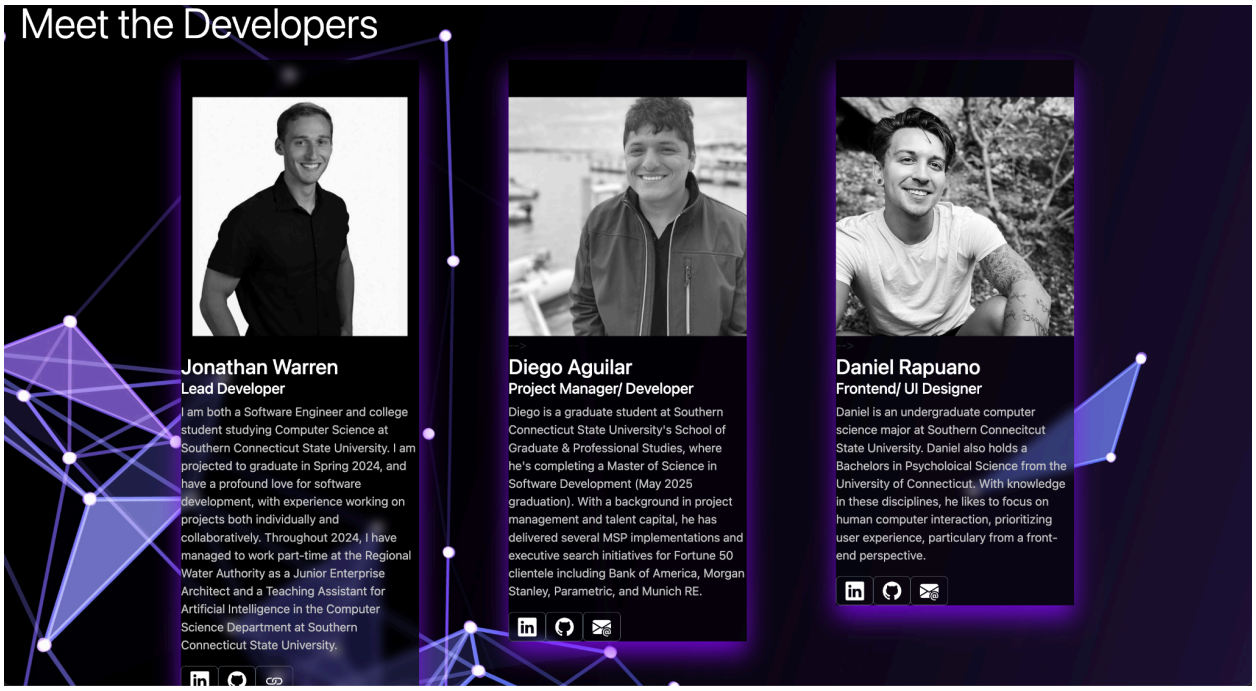

Figure 10.4.2.10 - the snapshot above displays our Developers page used for quick contact, feedback, and networking purposes.