SRS Documentation v2.0

Software Requirements Specification Document for
Domain Pulse

Ctrl Alt Defeat

2023/05/23

Contents

1 Introduction

1.1 Overview

Introducing Domain Pulse, the ultimate sentiment analysis platform. With Domain Pulse, you can easily gauge the sentiment surrounding any domain. Whether it's a business, a person, or anything else, Domain Pulse gathers information from across the internet and analyzes what people are saying.

Domain Pulse presents the results in a visually stunning and easy-to-understand format. Our wide range of visualizations brings statistics to life, making it a breeze to grasp the online presence and sentiment for any domain. Take control of understanding public opinion like never before with Domain Pulse.

1.2 Objectives

The objectives of the Domain Pulse project are to develop a comprehensive web application that enables users to track and analyze data from multiple sources, perform sentiment analysis, and visualize statistics. The application aims to provide a user-centered design approach, ensuring usability, accessibility, and a clear and intuitive interface. The system will be built using a scalable and modifiable architecture, leveraging microservices to handle high traffic and enable easy modification and extension. Security will be a top priority, with encryption and access control measures in place to protect user data. The project also aims to achieve high performance through caching and database optimization techniques. Overall, the objective is to create a reliable and efficient platform that empowers users to gain valuable insights from data analysis.

2 User Characteristics

Domain Pulse caters to a diverse range of users with varying demographics, including more mature age groups, educational backgrounds, and professional roles such as business professionals, social media managers, researchers, and PR professionals.

These users are all unified by their personal and professional interests, with a keen interest in sentiment analysis, online presence monitoring, and understanding public perception. They value data-driven decision-making and insights to support their decision-making processes. Additionally, they have a strong interest in market research, branding, reputation management, and online sentiment analysis. With their analytical and research-oriented mindsets, these users are driven to monitor and manage online presence and sentiment, aligning their professional lifestyles with Domain Pulse.

Considering the technological proficiency of users, Domain Pulse accommodates individuals at different skill levels. Novice users, who possess basic technological skills, may require more guidance, while intermediate users have moderate experience and comfort using technology. Expert users, on the other hand, are technologically proficient and adapt quickly to new systems.

Furthermore, Domain Pulse ensures accessibility and usability for users with varying physical abilities, including those with different visual capabilities. It also takes into account the cognitive abilities of its users, recognizing that attention spans and memory capabilities vary among individuals.

In terms of prior knowledge and experience, users may have different levels of expertise in sentiment analysis and online presence monitoring, as well as varying degrees of familiarity with similar tools or platforms.

3 User Stories

3.1 First Iteration

Table 1: User Story: Add a domain

User Story	As a user/business manager, I want to add a domain (business, product, etc...) to my list of domains, so that I can view and track customers' sentiment of it.
Acceptance Criteria	Given that I provided the name of the domain I wish to track, When I click the 'add domain' button, Then the domain is added to my list.

Table 2: User Story: Remove a domain

User Story	As a user/business manager, I want to remove a domain (business, product, etc...) from my list of domains, so that I can remove unimportant or unneeded domains.
Acceptance Criteria	Given that I have selected the domain I wish to remove, When I click the 'remove domain' button, Then the domain is removed from my list.

Table 3: User Story: Selecting a website theme

User Story	As a user, I want to change the theme to light or dark mode for my personal preference, so that I can more enjoy my use of the web-app.
Acceptance Criteria	Given that I am within the web-app, When I click the 'Change Theme' button, Then the theme of the page is changed.

Table 4: User Story: Log out of an account

User Story	As a user, I want to log out of my account, so that I can log in on another or have more security of others not viewing my domains.
Acceptance Criteria	Given that I am currently signed into an account, When I click the 'Sign Out' button, Then I am signed out of my account and placed on the login page.

Table 5: User Story: Add a source

User Story	As a user/business manager, I want to add a source (Twitter, Instagram, etc...) for sentiment data to my list of sources, so that I can view and track customers' sentiment on said source.
Acceptance Criteria	Given that I provided the name/link to the source I wish to use, When I click the 'add source' button, Then the source is added to my list of sources.

Table 6: User Story: Select a domain

User Story	As a user/business manager, I want to select a domain (business, product, etc...) from my list of domains, so that I can view and track customers' sentiment regarding it and other pieces of meta-data regarding the sentiment.
Acceptance Criteria	Given that I have the domain I wish to view in my list of domains, When I click the domain's name in the list, Then the sources from where I pull data from are listed and the overall sentiment is displayed.

Table 7: User Story: Register an account

User Story	As a user, I want to create an account, so that I help my domains perform better by understanding if customers are satisfied by tracking customer sentiment.
Acceptance Criteria	Given that I have provided my email and password on the 'register' page, When I click the 'Register' button, Then my account is created and I am logged in.

Table 8: User Story: Update Password

User Story	As a user, I want to update my password, so that I can ensure the safety of my account or change it to one I shall remember.
Acceptance Criteria	Given that I am logged into an account, When I click the 'Update Password' button, Then I am prompted to verify by email if I want to update my password and enter my new password.

Table 9: User Story: Remove a source

User Story	As a user/business manager, I want to remove a source (Twitter, Instagram, etc...) for sentiment data from my list of sources, so as to remove an unhelpful or unwanted source of data.
Acceptance Criteria	Given that I have selected the source I wish to remove, When I click the 'remove source' button, Then the source is removed from my list of sources.

Table 10: User Story: Select a domain

User Story	As a user/business manager, I want to select a domain (business, product, etc...) from my list of domains, so that I can view and track customers' sentiment regarding it.
Acceptance Criteria	Given that I have the domain I wish to view in my list of domains, When I click the domain's name in the list, Then display the overall sentiment and list of sources selected for that domain.

Table 11: User Story: Log into an account

User Story	As a user, I want to log into my account, so that I help my domains perform better by understanding if customers are satisfied by tracking customer sentiment.
Acceptance Criteria	Given that I am not currently signed into an account, on the 'log-in' page and have my account details entered, When I click the 'Log In' button, Then I am logged into my account that stores my previously created domains and sources.

Table 12: User Story: Forgot Password

User Story	As a user, I want to update my password, so that I can change it to one I shall remember and access my account.
Acceptance Criteria	Given that I am on the log-in screen, When I click the 'Forgot Password' button, Then I am prompted to verify by email if I want to update my password and enter my new password.

Table 13: User Story: Select a source

User Story	As a user/business manager, I want to select a source (Twitter, Facebook, etc...) from my list of sources for a domain, so that I can view and track customers' sentiment regarding my domain within the source.
Acceptance Criteria	Given that I have selected a domain and have provided sources for said domain, When I click the source in the list, Then the overall sentiment specific to the source is displayed.

Table 14: User Story: Select a statistic

User Story	As a user/business manager, I want to select a statistic (sentiment or metadata) from all available statistics, so that I can gain a better insight into how that statistic compares to other statistics and how it affects the overall sentiment.
Acceptance Criteria	Given that sentiment analysis has been performed, When I click on a specific statistic, Then a visualization of the statistic is displayed.

Table 15: User Story: View source data

User Story	As a user/business manager, I want to be able to see examples of data that was retrieved from my sources, so that I can confirm that the correct source was specified and correctly retrieved.
Acceptance Criteria	Given that sentiment analysis has been performed, When I am viewing the source of a domain, Then the raw source data is also displayed.

Table 16: User Story: View source data sentiments

User Story	As a user/business manager, I want to be able to see what the application predicts people think based on what they have said, so that I can confirm the validity of the application's predictions and trust the system.
Acceptance Criteria	Given that sentiment analysis has been performed, When I am viewing the examples raw source data of a domain, Then the predicted sentiment is displayed along with it.

3.2 Later Iterations

Table 17: User Story: View Time Series data

User Story	As a user/business manager, I want to view the time series data of a domain's sentiment from customers, so that I can understand when customers most enjoyed or disliked my product.
Acceptance Criteria	Given that I have selected the domain I wish to see time series data on, When I click the 'Time Series' button, Then the page displays a graph of customer sentiment of the selected domain over a period of time.

4 Functional Requirements

(grouped by subsystems)

4.1 Authentication

1. Registration
   (a) Can register using username and password
   (b) Can register using Google account
2. Login
   (a) Can login using username and password
   (b) Can login using Google account
3. Log out
   (a) A user has a means whereby they can log out of their account
4. Update password
   (a) If a user has forgotten their password, they may securely reset it
5. Remove account
   (a) A user can delete their account

4.2 Domain management

1. A user may create and domains with custom names, the domain acts as a 'folder' for a number of sources of data
2. A user may add a description for a domain
3. A user may add an image or select an icon to represent the domain
4. A user may remove a domain
5. Within a domain, the following operations can be performed
   (a) Add a data source (ex: Comments on a specified Instagram account) by selecting a source type and specifying additional parameters relevant for that source
   (b) Remove a data source
   (c) Refresh the data for the whole domain or a singular source
   (d) Edit a source type or source URL
   (e) Optional: Add, edit, and remove groups of keywords to track
6. A user can delete domains (deleting the sources within the domain too)

4.3 Data Visualization and Statistics

1. A user can view a select sample of data that was retrieved from their specified sources
2. A user can view all the statistics (derived from sentiment and meta data) for different sources contained within the domain or all the combined sources
3. A user can view data visualizations for sentiment data, meta-data, and optionally: Time-series data
    - Sentiment Analysis
        (a) Pie chart - Ratios of sentiment
        (b) Gauge chart - Objectivity-Subjectivity Score / Overall Sentiment Score
    - Meta-data
        (a) Bar graph - Data from each source / Data retrieval metrics for all sources
        (b) Timeline - Timeframe of data production for all sources
    - Optional: Time-series data

4.4 Sentiment Analysis

1. Sentiment analysis can be performed within any of the following groupings
    (a) For the domain as a whole (this includes all data across all specified sources)
    (b) Per data source (this considers all the data retrieved from one specific source)
2. The results of sentiment analysis on a grouping are returned as follows
    (a) The ratios of positive, negative, and neutral sentiment
    (b) An objectivity-subjectivity score
    (c) An overall sentiment score (from negative to positive)
    (d) An overall categorization within the following groups
        i. Very negative
        ii. Negative
        iii. Negative-to-Neutral
        iv. Neutral
        v. Neutral-to-Positive
        vi. Positive
        vii. Very positive
3. Meta-data is returned whenever a user performs sentiment analysis on an entire domain

(a) The meta-data to be returned is as follows

i. Which analysis sources were consulted (ex: Twitter, Instagram, etc.)
ii. How many pieces of data from each source were considered
iii. An indication of the timeframe over which the data was produced
iv. Whether new data needed to be retrieved from the web
v. Display metrics pertaining to how quickly data was retrieved
vi. Optional: Based on the number of and type of sources consulted, provide the user an estimate of how good a source the data is for sentiment analysis

4. Optional: Time-series data

(a) Time-series data is returned whenever a user performs sentiment analysis
(b) The following time-series data will be returned

i. The change in sentiment score (negative to positive) over a period of time.
ii. A prediction of the future trend of the sentiment of the domain

4.5 User Profiles

1. The domains a user wants to track are stored, this includes:

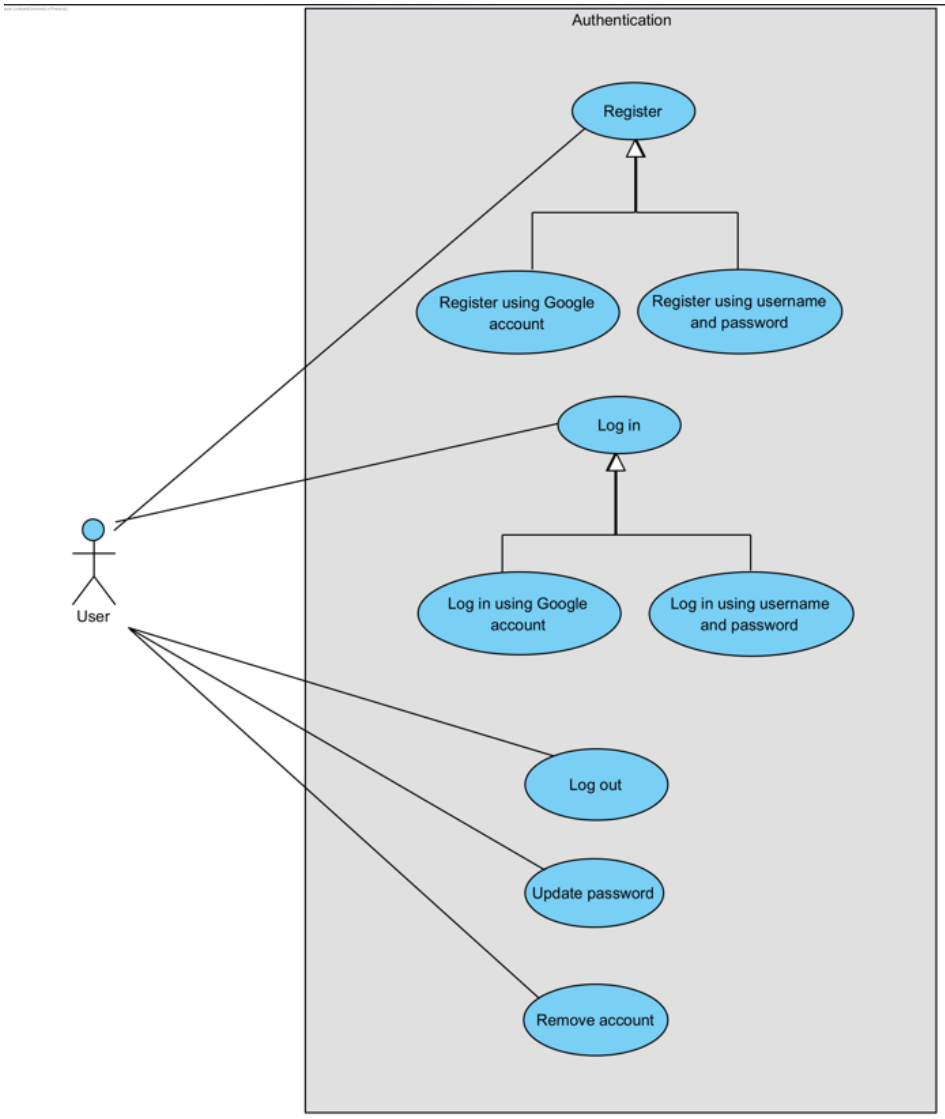
(a) The sources for the domain
(b) Optional: The keywords to specifically monitor

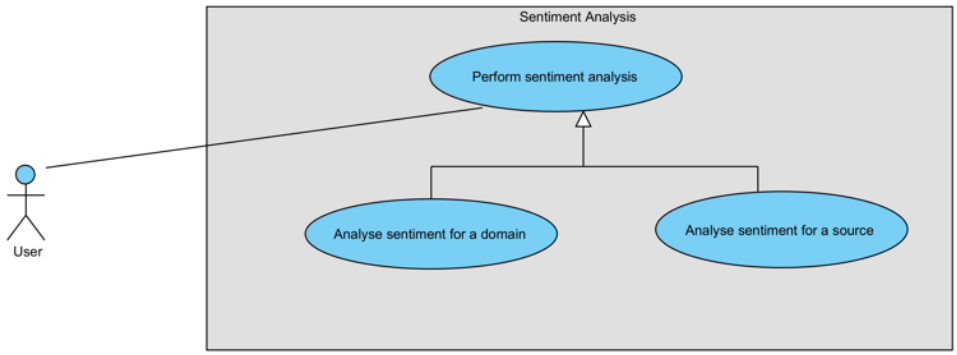
2. Personalization and preferences

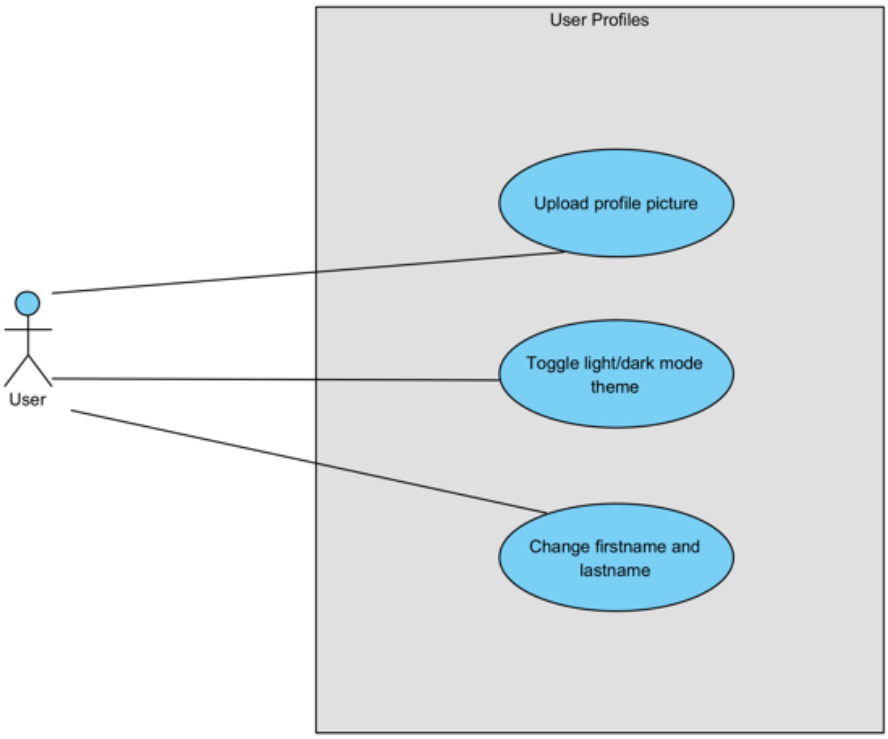
(a) User can specify either dark mode or light mode
(b) User can upload a profile image
(c) User can change their first name and last name

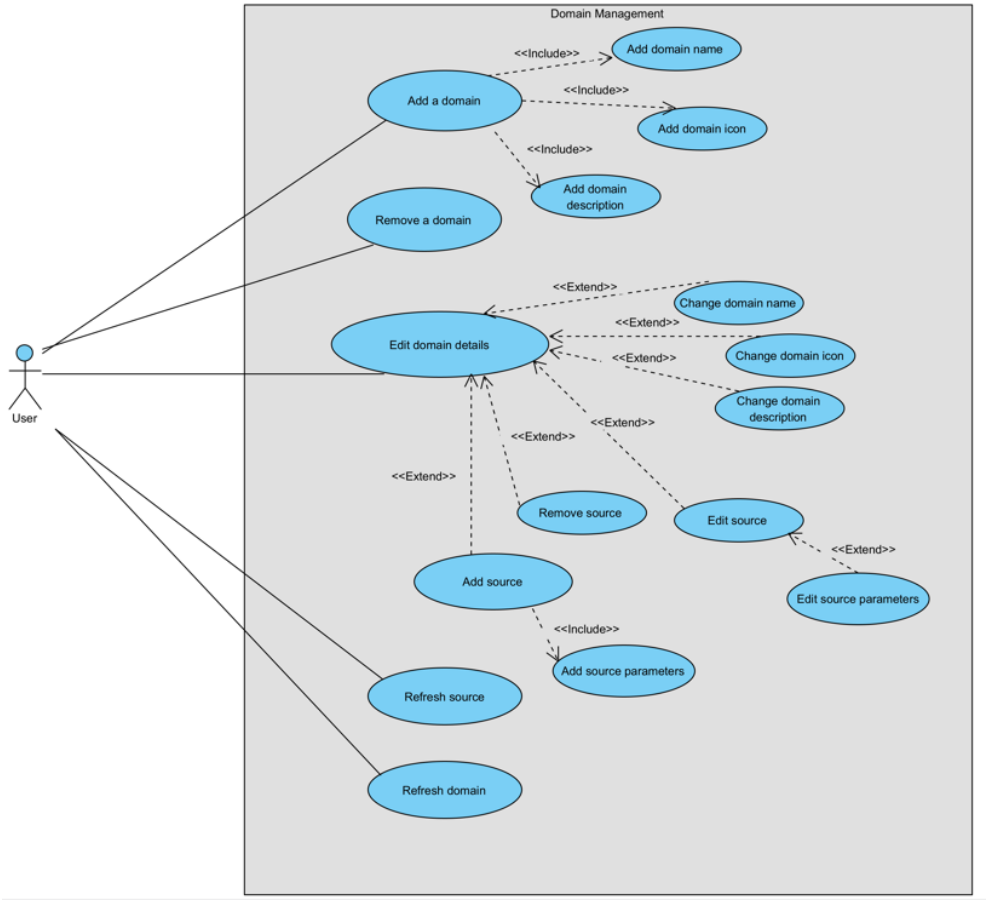
4.6 Requirements

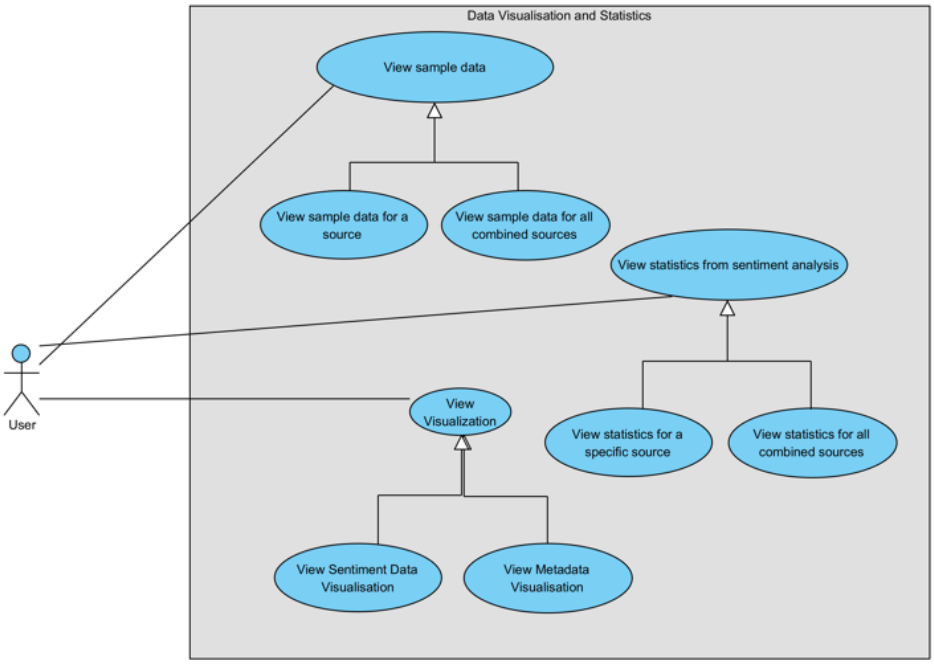
4.7 Use Case Diagrams






User Profiles
Upload profile picture
Toggle light/dark mode theme
Change firstname and lastname
User


Domain Management
Add domain name
<<Include>>
Add a domain
<<Include>>
Add domain icon
<<Include>>
Add domain description
Remove a domain
<<Extend>>
Change domain name
Edit domain details
<<Extend>>
<<Extend>>
Change domain icon
Change domain description
<<Extend>>
<<Extend>>
<<Extend>>
Remove source
Edit source
<<Extend>>
Add source
Edit source parameters
<<Include>>
Add source parameters
Refresh source
Refresh domain
User


Data Visualisation and Statistics
View sample data
View sample data for a source
View sample data for all combined sources
View statistics from sentiment analysis
User
View Visualization
View statistics for a specific source
View statistics for all combined sources
View Sentiment Data Visualisation
View Metadata Visualisation

5 Service Contract

5.1 Provided Software

As per the agreement between the client and development team, the software "Domain Pulse: A Sentiment Analysis Platform" shall be developed and deployed within the given timeframe. The system will allow users to register, create domains, add sources for the domains, and view aggregated and analyzed sentiment data from the sources.

5.2 Technology Stack

The system will be developed using Django as the backend web framework and Angular for the frontend UI. MongoDB, a NoSQL database, will be used for sentiment data storage, while PostgreSQL will be employed for user data storage. The system will be deployed and hosted on a client-provided virtual machine.

5.3 Project Management

The project will be managed using the Agile methodology, with weekly meetings between team members for effective communication and task coordination. GitHub Project Boards will be used to track tasks and progress. Biweekly meetings with the client will be conducted to provide progress reports and sprint reviews.

5.4 Module Agreement

The developed system will consist of external services and libraries within the codebase, comprising no more than 15

5.5 Timeline

The project will be completed within the provided timeframes of the COS 301 Capstone Project, and all required information and progress updates will be provided during the respective demos.

5.6 Security

Encryption will be implemented on all endpoints within the project to ensure secure data transmission between the server and the client. The virtual machine hosting the system will be secured and protected using measures such as firewalls. POST requests will be preferred over GET requests for enhanced data security.

5.7 Law and User Privacy

The software will comply with South African regulations, including laws such as the Protection of Personal Information Act (POPIA). User privacy and data will be protected and secured as required by the regulations.

6 Contract Design

6.1 Profile

changeMode

Precondition: *mode: modeEnum*
Postcondition: Object of type 'Profile' corresponding to the user who made the method call containing the updated 'mode' is persistently stored and returned.

updateProfilePicture

Precondition: *pictureURL: string*
Postcondition: Object of type 'Profile' corresponding to the user who made the method call containing the updated 'profileIcon' is persistently stored and returned.

6.2 User

Authentication User Functions will be controlled by Django Authentication System

deleteProfile

Precondition: *userID: string*
Postcondition: A boolean is returned with true meaning 'success' and false meaning 'failure'

createProfile

Precondition: *profileIcon: string, mode: modeEnum*
Postcondition: Object of type 'Profile' containing the specified 'profileIcon', 'mode' and an empty array of strings (string[]) called domainID is persistently stored and returned.

getDomain

Precondition: *domainID: string*
Postcondition: The corresponding Domain Object is returned.

6.3 Domain

createDomain

Precondition: *name: string*
Postcondition: An Object of type Domain is returned and persistently stored containing the provided name, an empty description, icon and sources.

getDomains

Precondition: *userID: string*
Postcondition: An array of Domain objects is returned containing all domains and their relevant data.

editDomainDescription

Precondition: *description: string*
Postcondition: An Object of type Domain corresponding to the edited domain is returned and persistently stored, with the updated description value.

editDomainIcon

Precondition: *icon: string*
Postcondition: An Object of type Domain corresponding to the edited domain is returned and persistently stored with the updated icon value.

addSourceToDomain

Precondition: *newSource: Source*
Postcondition: An Object of type Domain corresponding to the edited domain is returned and persistently stored with the updated Sources array containing newSource.

deleteDomain

Precondition: *domainID: string*
Postcondition: A boolean is returned with true meaning 'success' and false meaning 'failure'.

6.4 Source

getSource

Precondition: *sourceID: string*
PostCondition: An Object of type Source with the passed in platform and query string value is returned.

createSource

Precondition: *platform: PlatformEnum, queryString: string*
PostCondition: An Object of type Source with the passed in platform and query string value is returned and persistently stored.

deleteSource

PreCondition: *sourceID: string*
PostCondition: A boolean is returned with true meaning 'success' and false meaning 'failure'.

editSource

Precondition: *queryString: string*
PostCondition: An Object of type Source corresponding to the edited source is returned and persistently stored with the queryString value.

6.5 AnalysisEngine

analyseData

Precondition: *domain: Domain*
Postcondition: An array of SentimentMetrics objects corresponding to the stored sentiment records relating to the domain.

analyseData

Precondition: *source: Source*
Postcondition: An array of SentimentMetrics objects corresponding to the stored sentiment records relating to the specific source of the domain.

6.6 DataWarehouse

getSentimentData

Precondition: *domainID: string*
Postcondition: An array of sentimentRecord objects is returned containing the records (comments, posts, etc.) pertaining to all sources of the domain provided.

fetchNewData

Precondition: *source: Source*
Postcondition: An array of sentimentRecord objects is persistently stored containing the records (comments, posts, etc.) pertaining to the specific source provided.

refreshSource

Precondition: *source: Source*
PostCondition: An array of sentimentRecord objects is returned and persistently stored containing the records (comments, posts, etc.) pertaining to the specific source provided.

getMetaData

Precondition: *source: Source*
PostCondition: An object is returned containing the number of pieces of data used in the source's data collection and the time taken for said data to be retrieved.

refreshAllSources

Precondition: *domain: Domain*
Postcondition: An array of sentimentRecords objects is returned and persistently stored containing the records (comments, posts, etc.) pertaining to all sources of the domain provided.

6.7 Data Dashboard

displayMetrics

Precondition: *domain: Domain*
Postcondition: The relevant data pertaining to the domain's sentiment metrics shall be displayed on the web page.

displayGraphs

Precondition: *domain: Domain*
Postcondition: The relevant graphs data pertaining to the domain's sentiment metrics shall be displayed on the web page.

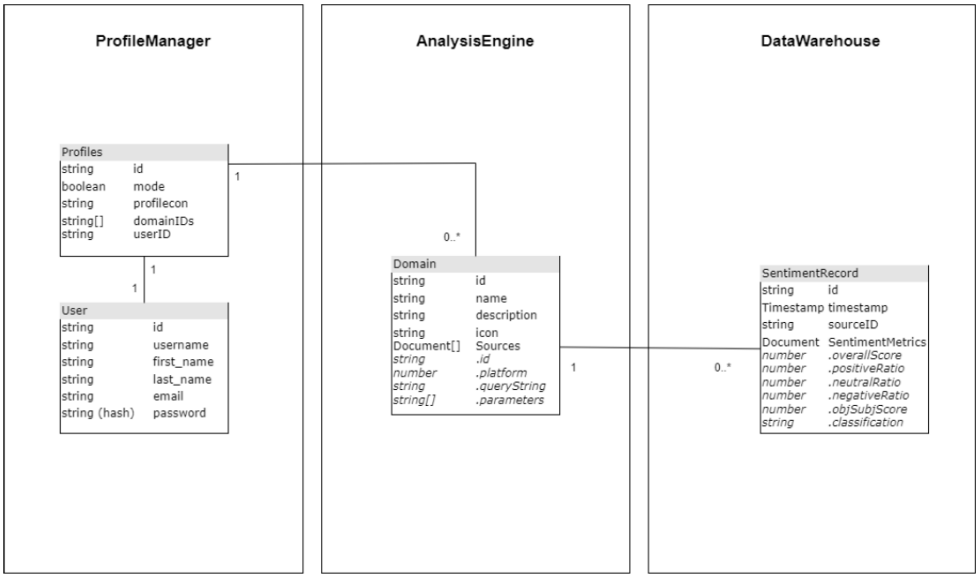
displayMetrics

Precondition: *source: Source*
Postcondition: The relevant data pertaining to the domain's specific source sentiment metrics shall be displayed on the web page.

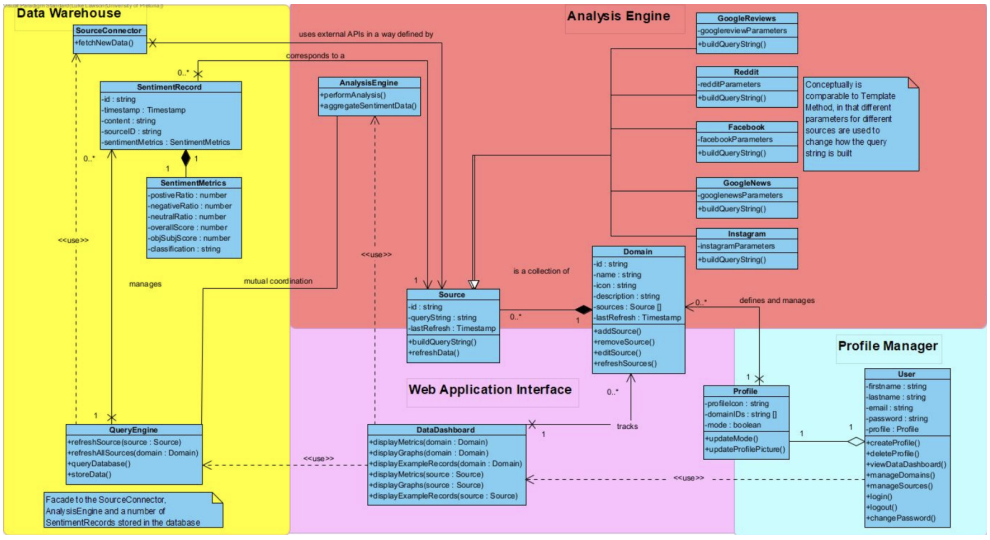
displayGraphs

Precondition: *source: Source*
Postcondition: The relevant graphs pertaining to the domain's specific source sentiment metrics shall be displayed on the web page.

7 Database Design



8 Class Diagram



Design Patterns: While the nature of the architecture does not lend itself to the Object-Oriented paradigm, two design patterns have been identified (and noted within the class diagram) - namely:

Facade: The *QueryEngine* effectively acts as a *Facade* to the *SourceConnector*, the *Analysis Engine*, and conceptually, to the database. This ensures that the *QueryEngine* acts as a uniform interface to the frontend web-app (as well as the rest of the application in general) for the retrieval of data. That is, the rest of the system need not interact with the complicated underlying systems, but rather can communicate uniformly with the *QueryEngine* (which abstracts out the complicated query logic involved with retrieving data from the database, from external APIs, or sending data for sentiment analysis in the *AnalysisEngine*).

Template Method: Conceptually, the *Template Method* design pattern is applied as follows: When a user defines a source, the source is categorized based on its type (e.g., Facebook, Reddit, etc). Each of these types of sources has different APIs with different parameters (and thus have different query strings). While the parent *Source* class specifies the operation to build a query string, the individual, specific type of source defines exactly how that string may be built. i.e., The *Source* is effectively a template for the construction of external API query strings, while the derivatives of the *Source* (e.g., Reddit, Facebook, etc.) define how that template is "filled in" to construct the query string.

9 Architectural Requirements

9.1 Quality Requirements

- Security
- Usability
- Accessibility
- Scalability
- Availability
- Modifiability
- Performance

9.2 Architectural Patterns and Tactics

- Below we discuss which architectural patterns and tactics we will use to meet the quality requirements. The patterns and tactics are in bold.

Security

- Authentication system of Django that encrypts data created and sent pertaining to user data (email, password, etc.).
- Use POST if possible as opposed to GET.
- Our virtual machine on which we deploy shall have extensive firewalls set up so as to prevent foreign attacks of our system.
- Access to the virtual machine can only be performed by members of our group using SSH and by logging into the machine with private details.

Usability

- User-Centred Design Approach
  - Consider the end-user throughout the design process and design the system accordingly.
  - Make the terminology easy to understand but still meaningful, considering users with no technical knowledge about NLP (Natural Language Processing).
  - Examples of end-users: R&D specialists, social media managers, project leaders, executives, consultants.
- Clear and Intuitive Interface
  - Reduce clutter on the dashboard.
  - Ensure that the meaning and purpose of actions is clear through the use of descriptive and minimalistic icons.

  - Provide user feedback as they navigate through the application.
  - Utilize a user workflow of top-to-bottom, left-to-right navigation, ensuring that the process of completing steps feels natural and ordered.

- Usability Testing
  - Test the system with representative users.
  - Collect and implement feedback.

Accessibility

- Implement a dark mode to cater for visually impaired and cognitive disabilities by providing a simple, free-from-distraction, high contrast user interface.

Scalability

- By employing elements of the microservices architecture, we intend to improve the application's ability to cope with high amounts of traffic from multiple concurrent users.
- Microservices allow for service isolation. Since services will be in isolation of each other, we may prevent issues in one service from impacting others. Ex: should one service experience high traffic, it can be scaled independently without affecting other services. Furthermore since this bottleneck will be isolated to an individually deployed system, it will impact other services to a minimal degree.

Availability

- By leveraging aspects of the microservices architecture, we can improve the availability of the application.
- The microservices architecture promotes fault isolation, meaning issues in one service are less likely to propagate to other services. This increases the overall resilience of the system.
- Consequently, this will improve the overall availability of the system since even if a single deployed system fails, the other deployed units are still able to function (to an extent).

Modifiability

- Furthermore, use of the microservices architecture will improve the ease with which us as developers can modify or extend the existing system.
- By promoting separation of concerns (architectural tactic) it is easier for
- For example: since the service that performs sentiment analysis on the data is deployed independently of the service responsible for fetching and aggregating data from external APIs, it is easy to extend the types of sources a user may consult for analysis, without making any changes to the analysis engine itself.

Similarly, changes to user profiles are totally irrelevant to the other services of the system

- Consequently, easy modifiability of the system is accomplished

Performance

- By employing the architectural tactic of caching can we improve the performance of the system. Caching is accomplished as follows:
  - Once sentiment analysis is performed on data, those computed sentiment metrics are stored along with the actual data
  - Consequently, the next time the user wants to perform analysis on the data, data for which sentiment metrics have already been computed do not need to be computed again, since they have been 'cached' by the database
  - While this is not true caching (since the sentiment metrics once computed are permanently persisted) it does ensure that sentiment analysis overhead is drastically reduced since analysis will never be performed more than once for the same piece of data
- Furthermore, by using the architectural tactic of database optimization, we can improve performance of the system by reducing the execution time of database queries. Database optimization is achieved via the following means
  - By tightly grouping data that is highly related/dependent, we design the database in such a way that (slow) compound queries are not necessary. For example: a NoSQL document-based database is used to store the domains a user manages. Since the purpose of these domains is effectively to act as collections of sources, it is much more efficient to make the list of sources in a domain a list attribute on the domain stored in the database itself, rather than have a separate collection or database for the sources. Hence, whenever a domain is retrieved, so are the sources it contains, without the overhead of writing another query or a complex join to fetch the corresponding sources
  - Playing into the advantages of the microservices architecture, we are able to leverage different database technologies for different deployed systems depending on what is more fit for purpose. For example, in our context, persisting user profiles and authentication data is much more naturally and efficiently done by the use of a SQL database, as opposed to using a NoSQL database (which the other deployed systems make use of). Subsequently, we may achieve faster database queries on user profile and authentication related data, improving performance

9.3 Constraints

Technical Constraints

- We are required to make use of a NoSQL document-based database in our application (upon specification from Southern Cross Solutions), however, upon request and their advice (given our architecture), it is acceptable to use a SQL database for microservices where we deem it necessary.
- All technology used for development must be free and open source (upon specification from Southern Cross Solutions).

Performance Constraints

- There are no specific, measurable metrics of performance that the application is required to conform to; however, the following guidelines must be adhered to:
  - The user must not wait for an inordinate amount of time for the data to be refreshed, including fetching new data from external APIs and performing analysis on that data.
  - Viewing metrics for different sources and domains should feel seamless, with minimal waiting time involved in fetching the data.
  - The web app needs to feel user-friendly, easy-to-use, and responsive.

Security Constraints

- User credentials must be stored with utmost care, ensuring that sensitive user information cannot be leaked and that another user may not gain unauthorized access to a user's account.
- The SSH keys provided to the team by Southern Cross Solutions cannot, under any circumstance, be leaked publicly and must only be sent across a secure platform.

Regulatory Constraints

- Need to conform to the Terms of Service as specified by each external API we make use of.
- Need to ensure compliance with POPIA if we were to associate content with the person who posted it.
- Need to adhere to Southern Cross Solutions' company policy regarding keeping SSH keys to virtual machines secure.

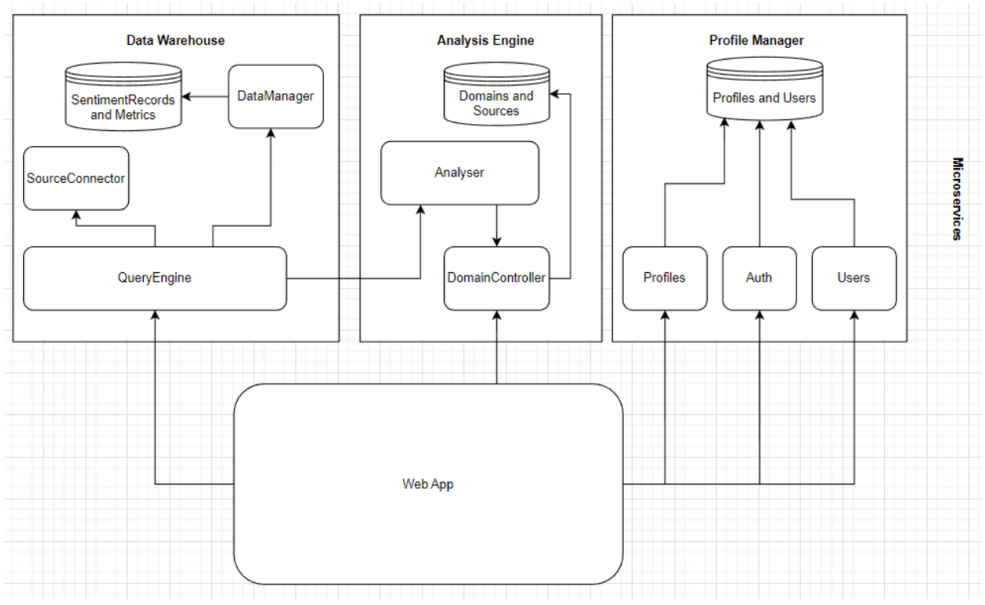
Cost Constraints

- No budget allocated for the project as of yet.
- Potentially a $50 credit to make use of AWS services.
- Southern Cross Solutions have not confirmed whether they will supply a budget to make use of paid external APIs (such as that for Twitter) - this is an issue that will be addressed further along in development.

Time Constraints

- Need to have the final product delivered before the final demonstration and Project Day (approximately 6 months).
- Need to conform to milestone deadlines as set out by the requirements for demos 1 to 4.

Architectural Diagram



10 Technology Requirements

10.1 Development Environment

For development, our team members shall be developing within VS Code on Linux to ensure consistency within the produced code and testing of our software. Having all members using VS Code also allows for the use of software such as Live Share for collaborative development to improve efficiency within development.

10.2 Version Control

Git will be used for version control, using a clear and simple branching strategy that allows members to easily work on distinct components of our system while also allowing the reverting to previous versions as a 'fail-safe'. Meanwhile, GitHub will be used to host the Git repository.

10.3 Programming Languages

Within our team, many (if not all) members have a high proficiency in coding in Python. Python is considered one of the best programming languages for Machine Learning and data analysis, which are aspects on which our system will heavily rely. Therefore, Python is our programming language of choice.

10.4 Frameworks and Libraries

We have decided to use Django as our web framework for the creation of our app due to its simplicity, efficiency, and secure data transmission capabilities. Within our Django project, we will be utilizing Python's proficiency in data analysis and machine learning by using the powerful Vader and Grafana libraries, which allow us to perform sentiment analysis on text and visualize our data, respectively. For the front-end development, we will be using Angular due to its versatility and ability to create high-quality user interfaces.

10.5 Database Systems

For our database systems, we will be using MongoDB as a document-based NoSQL database for data collection of comments, posts, and other content related to user data. Additionally, PostgreSQL will be used as an SQL database for storing user profiles and authentication data for ease of access and querying.

10.6 Testing and Quality Assurance Tools

We will perform various forms of testing to ensure the quality of our software. For front-end testing, Cypress will be used, and for unit testing, we will utilize the built-in Unittest library of Python, which is recommended for Django.

10.7 Deployment and Infrastructure

For deployment, our clients have provided a Linux-based virtual machine where we can deploy our software. We will use one virtual machine for the testing environment and another for the production environment to ensure a separation of concerns. Additionally, a domain may be acquired for ease of customer access to the software.

10.8 Collaboration and Communication Tools

To ensure effective communication within our team, we have a private WhatsApp group for important announcements, a private Discord server for more specific announcements and discussions, and a GitHub Project Board to track work progress. For communication with our clients, we have set up a Slack workspace for quick communication.

10.9 Security and Encryption

Our system is secured using technologies such as Django's authentication system, which encrypts user data. We prioritize the use of POST over GET for more secure data transmission. Our virtual machine is protected by extensive firewalls to prevent foreign attacks, and access is restricted to authorized members of our team using SSH and private login details.

10.10 Continuous Integration and Deployment Tools

We will utilize CI/CD technologies such as Codecov and GitHub Actions to ensure thorough testing and checking before deployment. GitHub Actions will automate the building, testing, and deployment processes, while Codecov will provide insights into test results and failures. We have been provided with separate production and testing environments to deploy and check the application's functionality.