SRS Documentation v3.0

Software Requirements Specification Document for
Domain Pulse

Ctrl Alt Defeat

2023/07/31

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, franchise, a product, an event, or more. Domain Pulse gathers and analyzes sentiment data from various sources, ranging from social media to review platforms, with the support for providing your own data either via a live review process or a CSV.

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 a combination of powerful architectural patterns to achieve the quality requirements set out in the Architectural Design document and statement of quality requirements.

2 User Characteristics

Domain Pulse caters to a diverse range of users with varying demographics, ranging from high-school students to more mature age groups. Domain Pulse can be used by people with various educational backgrounds and professional roles. These may include business professionals, social media managers, researchers, and PR professionals - however other users may find value in using Domain Pulse by simply satisfying personal interests and exploration.

Despite the diverse background of our users, they 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 exploratory-oriented mindsets, these users are driven to monitor and/or manage online presence and sentiment, exploring their personal and professional interests with Domain Pulse.

Regarding 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.

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 (Youtube, Google Reviews, 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 and select the appropriate source type (providing the appropriate parameter data), 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: 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 13: User Story: Select a statistic

User Story	As a user/business manager, I want to select a statistic (sentiment or meta-data) 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 14: 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 15: 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.

Table 16: 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 or source I wish to see time series data on, When I click the navigate the graph pane, Then the page displays a graph of customer sentiment of the selected domain or source over a period of time.

Table 17: User Story: Report generation

User Story	As a user/PR manager, I want to generate a sharable PDF report that neatly summarizes the results of the sentiment analysis, so that I can present the results in a meeting or to an executive more formally.
Acceptance Criteria	Given that I have selected the domain I wish to generate a report for, When click the "Generate report" button, Then a QR code is displayed as well as a URL to view the PDF report.

Table 18: User Story: Live review

User Story	As a user/PR manager, I want to generate a QR code that enables anyone to make a review about any arbitrary thing, so that I may collect my own sentiment data in real time.
Acceptance Criteria	Given that I have selected the domain I wish to add a live review data source to, When click the "Add source" button and select live review, Then a QR code is displayed which acts as a link to the review form.

4 Functional Requirements

(grouped by subsystems)

4.1 Authentication

1. Registration
   (a) Can register using username and password
2. Login
   (a) Can login using username and password
3. Log out
   (a) A user has a means whereby they can log out of their account
4. Update password
   (a) A user may securely change their password
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 Youtube video) by selecting a source type and specifying a URL for that source

(b) Remove a data source

(c) The following sources must be available for analysis:

i. Youtube

ii. Google reviews

iii. Tripadvisor

iv. TrustPilot

v. Live review

- A user can toggle whether the review is enabled or disabled
- A user is presented with a QR code that can be shared - linking to a review form

vi. CSV upload

(d) Refresh the data for the whole domain or a singular source

(e) Edit a source's name

6. A user can delete domains (which should delete the sources within the domain too)

4.3 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) The most prevalent emotions

   (c) An overall sentiment score (from negative to positive)

   (d) An overall categorization within the following groups

   i. Very negative

   ii. Negative

   iii. Somewhat negative

   iv. Neutral

   v. Somewhat positive

   vi. Positive

   vii. Very positive

   viii. Undecided (if the sentiment cannot be confidently determined)

   (e) Toxicity score

3. Meta-data is displayed for either the entire domain or for each source
    (a) The meta-data to be returned is as follows
        i. How many pieces of data from each source were considered
        ii. An indication of the time-frame over which the data was produced

4.4 Data Visualization and Statistics

1. A user can view a sample of data that was retrieved from their specified sources
2. A user can view all the visualizations (both sentiment and meta data) for different sources contained within the domain or all the sources combined
3. A user can view data visualizations for sentiment data, meta-data, and time-series data
    - Sentiment Analysis
        (a) Pie chart - Ratios of positive, negative, and neutral sentiment
        (b) Bar Graph - Emotions
        (c) Gauge chart - An overall sentiment score (from negative to positive) with an overall categorization within the following groups
        (d) Pie Chart - Toxicity
    - Sentiment Analysis Time Series
        (a) Stacked, cumulative area line chart showing sentiment ratios over time as an exponential moving average
        (b) Line chart showing overall sentiment score over time as an exponential moving average
        (c) Overlay-ed line charts showing each emotion score over time as an exponential moving average
            – A user may toggle emotions to be displayed or not
        (d) Bar graph showing the number of toxic comments on each day, overlay-ed with overall sentiment over time line chart
            – A user may toggle the overall sentiment line and/or bars to be displayed or not
    - Meta-data
        (a) Bar graph - How many pieces of data from each source were considered from each sources
        (b) Timeline - Line graph indicating the number of sentiments expressed over time
4. A user may download visualization data as a CSV for further use

4.5 User Profiles

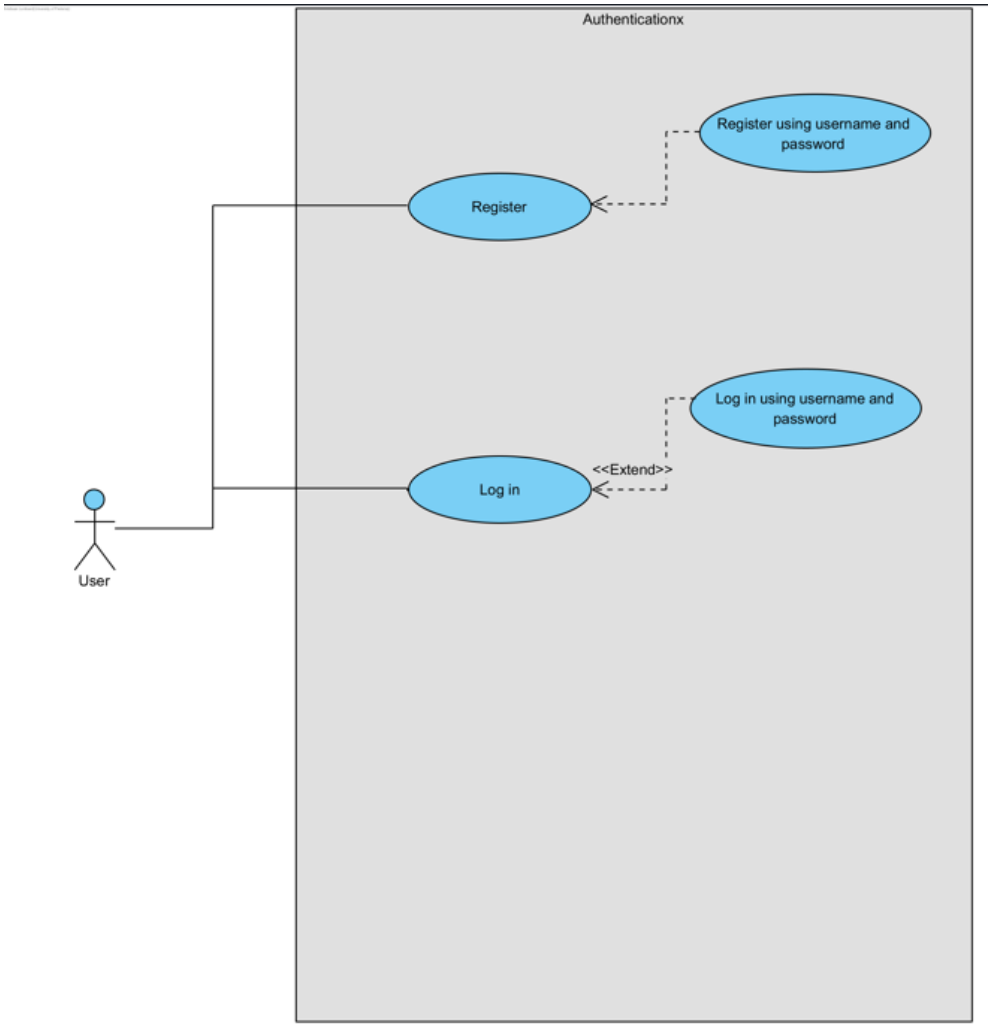
1. Personalization and preferences
    (a) User can specify either dark mode or light mode

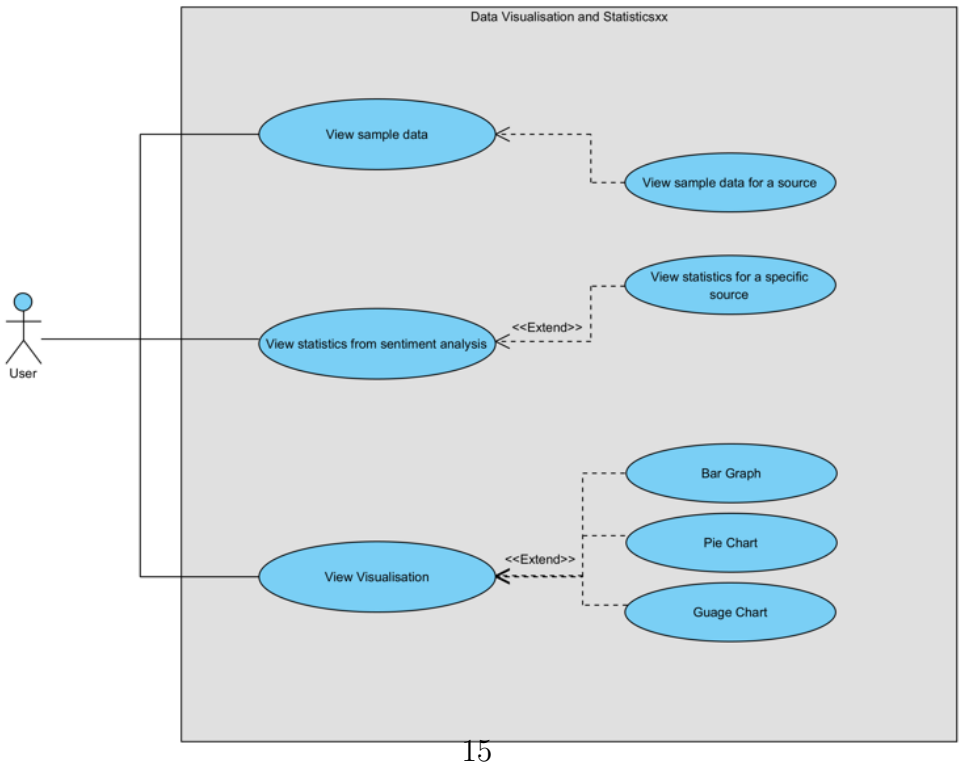
(b) User can upload a profile image

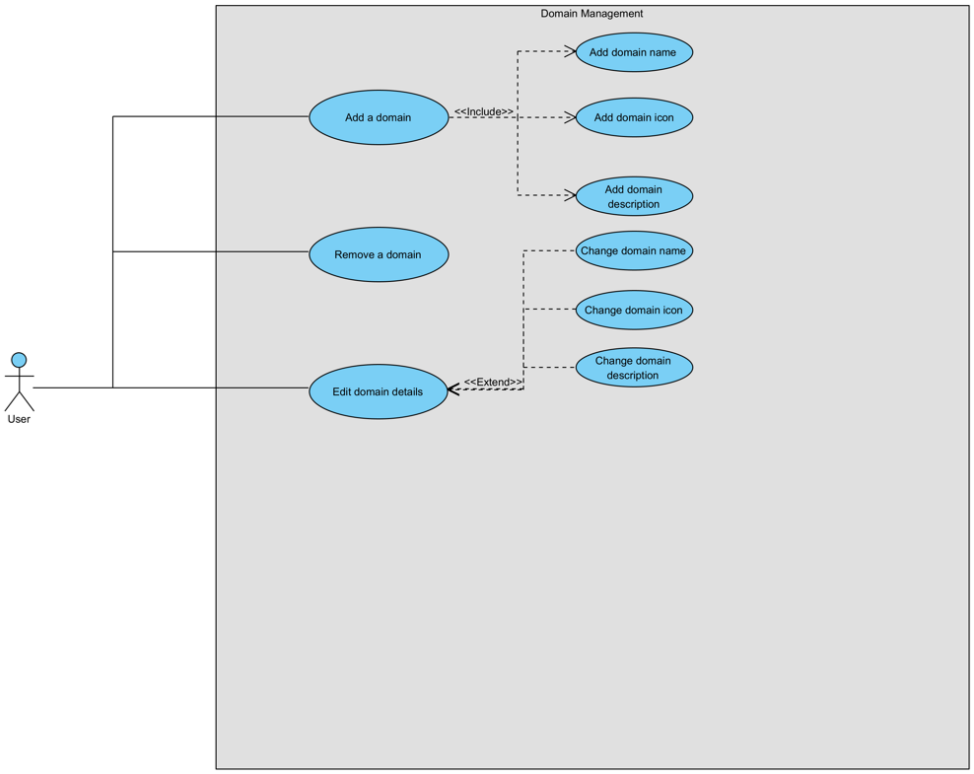
(c) User can change their first name and last name

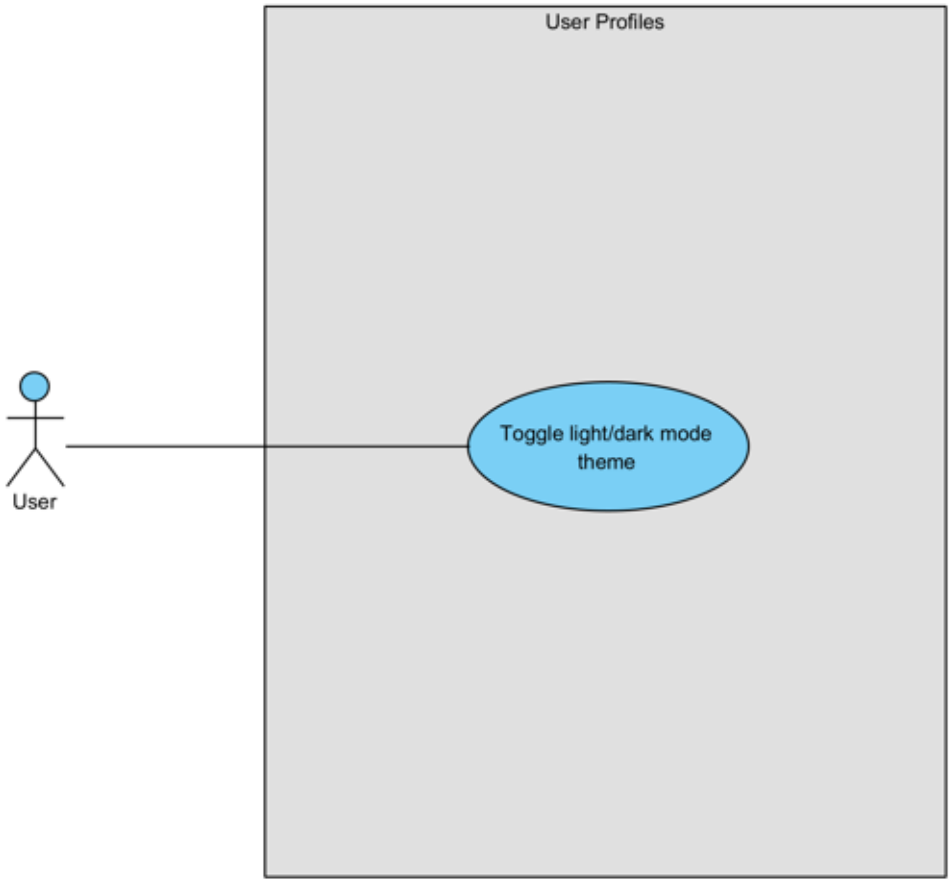
5 Use Case Diagrams

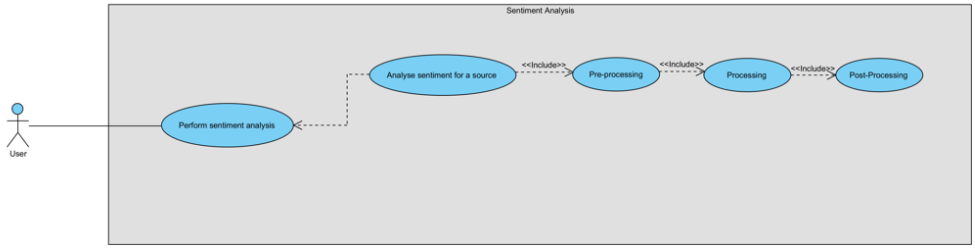
5.1 Use Case Diagrams Component-By-Component Design






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


User Profiles
User
Toggle light/dark mode theme


Sentiment Analysis
Analyse sentiment for a source
<<Include>>
Pre-processing
<<Include>>
Processing
<<Include>>
Post-Processing
Perform sentiment analysis
User

6 Service Contract

6.1 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 time-frame. The system will allow users to register, create domains, add sources for the domains, and view aggregated and analyzed sentiment data from the sources.

6.2 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.

6.3 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.

6.4 Module Agreement

The developed system will consist of external services and libraries within the codebase, comprising no more than 15% of the total developed codebase. The team will ensure compliance with The University of Pretoria's Plagiarism and Code of Conduct policies, ensuring the originality and ownership of the produced code.

6.5 Timeline

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

6.6 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.

6.7 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.

7 Contract Design

7.1 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.

7.2 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.

7.3 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'.

7.4 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.

7.5 AnalysisEngine

analyzeData

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

analyzeData

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

7.6 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.

7.7 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

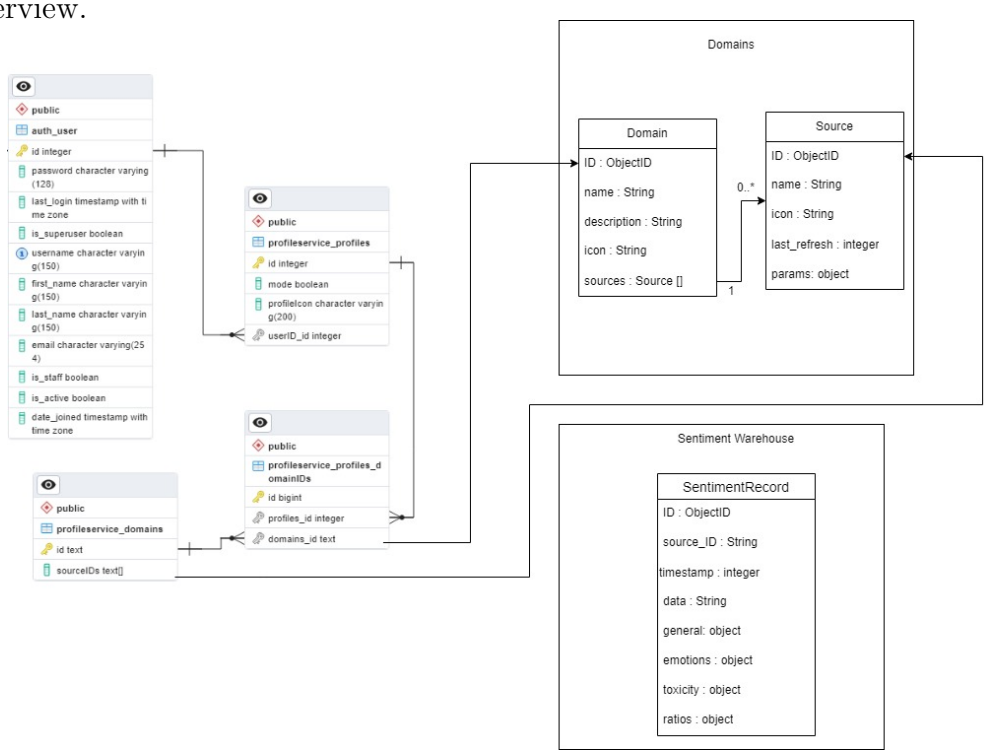
Precondition: *domain: Domain*
Postcondition: The relevant graphs data pertaining to the domain's sentiment metrics shall be displayed on the web page.

displayMetrics

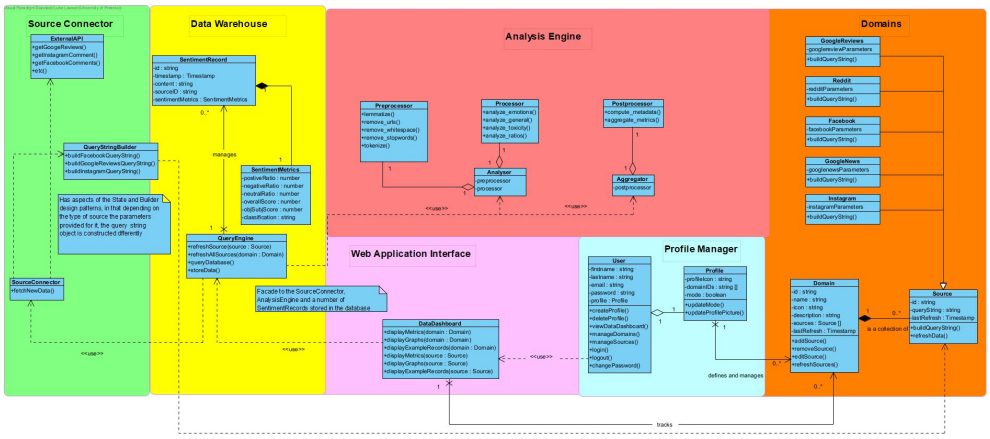
Precondition: *source: Source*
Postcondition: The relevant data pertaining to the domain's specific source sentiment metrics shall be displayed on the web page.

8 Database Design and Data Models

The following is an outline of the structure of the data to be persisted and relationships between data entities. The system makes use of a combination of relational and document-based database technologies, hence the below diagram provides an overview.



9 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.

10 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 services where we deem it necessary.
- All technology used for development must be free and open source (upon specification from Southern Cross Solutions).

Performance Constraints

- Generally speaking, the following guidelines must be adhered to. Specific quantification of these requirements can be found in our statement of quality requirements and Testing Manual:
  - 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

- Budget allocated for Project Day materials as well as an amount of no more than R300 towards making use of 3rd party web-scraping services.

Time Constraints

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

11 Technology Requirements

Below the technology requirements of the system are presented. All the mentioned technologies (with the exception of Azure and Outscraper) are free and open source. The justification of technology choices can be found in the Architectural Design Document.

11.1 Development Environment

For development, our team members shall be developing within VS Code in an Ubuntu Linux environment 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.

11.2 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.

11.3 Programming Languages

Within our team, 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 backend programming language of choice - along with the appropriate libraries for NLP, importing models, and other utility tasks. Frontend development will make use of standard languages: Typescript, CSS and HTML.

11.4 Frameworks

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 a variety of powerful libraries for sentiment analysis and Natural Language Processing. For the front-end development, we will be using Angular due to its versatility and ability to create high-quality user interfaces.

11.5 Libraries

A number of libraries will be employed throughout the application for various purposes, these are as follows:

11.5.1 Frontend (Angular)

- Apex Charts is a Javascript library that will be used to plot visuals and graphs. This library is able to easily construct attractive graphs with minimal code,

and thus we will be making use of this graphing library rather than implementing our own graphing functions.

- NGXS state management is used to predictably manage the state of our app's user interface. In modern applications, the use of state is is considered industry standard and hence we will make use of this library.

11.5.2 Backend (Django)

- Python's 'NLTK' (Natural Language ToolKit) will be used to aid in the handling and preprocessing of textual data. This library will be employed for it's performance, convenience, and power regarding NLP.

- 'vadersentiment' is a Python library that implements the VADER sentiment analysis model which is used as part of the sentiment analysis process in our application. This library is used as it would be infeasible, too costly, and too time-consuming to implement VADER or a similar lexicon-based sentiment analysis tool ourselves.

- The 'transformers' and 'torch' (Pytorch) libraries are required to import, construct, and implement the DistilBERT models that are used by our application. It would not be feasible to build neural network models of this complexity and scale ourselves. Furthermore, the need the expense associated with collecting sufficiently substantial and general training data, in addition to the computational expense associated with training a state of the art sentiment analysis mode, is outside the scale and scope of our project. Hence, libraries are used to implement this functionality.

- 'djangorestframework-simplejwt' is a Django that allows us to generate secure JWTs - this is needed for the authorization and security aspect of our project and hence we are inclined to make use of a secure and extensively tested library for this purpose.

- 'matplotlib' is a Python library used for plotting graphs in Python. Creating our own graph plotting library is outside the scope of our project, and thus a graphing library is far more suitable.

- 'numpy' is a Python library that enable performance and powerful operations on large sets of data - this library is used to improve the slickness with which data is handled and does not implement major functionality.

- 'pdfkit' is used for rendering HTML documents as PDFs. HTML to pdf conversion from first principles is a tedious task that falls outsides the project scope - hence we use this library to convert the HTML pages we construct into PDF reports.

- 'bleach' is a highly tested and trusted library for input sanitization in Python - since this is a critical security function, we prefer to trust this functionality to a library rather than implement our own sanitization techniques.

- The following libraries are required for simple utilities that would be grossly impractical to implement ourselves from first principles:

- 'mock' - to mock out functions during unit testing
- 'coverage' - to track code coverage
- 'django-cors-headers'- for constructing cors headers
- 'pymongo' - for connecting to a MongoDB instance from Python code
- 'python-dotenv' - accessing .env file data
- 'outscraper'- a library for integration with Outscraper's API (see 'Web-scraping' below)
- 'pytz' - Used for performing timezone conversions
- 'psycopg2-binary' - general utilities for connecting to a Postgres database
- 'azure-storage-blob' for connecting to Azure storage
- 'shortuuid' - generating unique ID codes for documents
- 'unidecode' - for decoding unsupported ASCII characters

11.6 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.

11.7 Testing and Quality Assurance Tools

We will perform various forms of automated testing to ensure the quality of our software. For integration and E2E testing, Cypress will be used, and for unit testing, we will utilize the built-in Unittest library of Django Python, which is recommended for Django.

11.8 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 (accessible with different domain names) to ensure a separation of concerns. Additionally, a domain may be acquired for ease of customer access to the software.

11.9 Machine Learning and AI

We will make use of a variety of open source, freely available, pre-trained sentiment analysis models. The BERT-style models are to be sourced from Hugging Face, while VADER is already available through the "vader" Python library.

11.10 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 private Signal group for sharing sensitive information such as secret keys, and a GitHub Project Board to track work progress. For communication with our clients, we have set up a Slack workspace for quick communication.

11.11 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.

11.12 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.

11.13 Hosting and Deployment

Our clients, Southern Cross Solutions, have provisioned a virtual machine for us to host our application. Furthermore, we will be making use of Microsoft Azure cloud services for resource hosting and backup application hosting.

11.14 Web Scraping

In order to comply with the terms of service of the platforms from which Domain Pulse collects data, we will make use of Outscraper, a third party web scraping tool, to perform scraping on various review platforms such as Google Reviews and Tripadvisor. Furthermore, implementing a performant and secure webscraper that does not violate the terms of service of any online platform is a massive task that falls outside the scope of Domain Pulse - motivating the use of Outscraper. To retrieve data from Youtube, we will make use of the freely available Youtube Data v3 API, which allows us to retrieve comments on videos.