SRS Documentation v1.0

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

2.1 Demographics

- **Age**: Users of varying age groups, depending on their professional roles and interests.
- **Gender**: Users of all genders.
- **Education Level**: Users with diverse educational backgrounds.
- **Occupation**: Business professionals, social media managers, researchers, PR professionals, etc.

2.2 Psychographics

- **Attitudes**: Users interested in sentiment analysis, monitoring online presence, and understanding public perception.
- **Values**: Users who value data-driven decision-making and insights for decision support.
- **Interests**: Users interested in market research, branding, reputation management, and online sentiment analysis.
- **Lifestyles**: Users with professional roles that involve monitoring and managing online presence and sentiment.
- **Personality Traits**: Users with analytical and research-oriented mindsets.

2.3 Technological Proficiency

- **Novice Users**: Users with basic technological skills who may require more guidance.
- **Intermediate Users**: Users with moderate experience and comfort using technology.
- **Expert Users**: Users who are technologically proficient and can quickly adapt to new systems.

2.4 Physical Abilities

- **Vision**: Users with varying visual abilities.
- **Other Physical Abilities**: Consideration for accessibility and usability for all users.

2.5 Cognitive Abilities

- **Attention**: Users with different levels of attention spans.
- **Memory**: Users with varying memory capabilities.

2.6 Prior Knowledge and Experience

- Users with different levels of knowledge and experience in sentiment analysis and online presence monitoring.
- Familiarity with Similar Tools or Platforms

2.7 Goals and Tasks

- Monitoring and analyzing sentiment and online presence of specific domains.
- Gathering insights for decision-making, market research, or branding purposes.
- Tracking public perception, PR campaign impact, or personal brand sentiment.
- Supporting research and analysis with data on sentiment trends and patterns.

2.8 Emotional Factors

- Users with preferences for user interfaces and interactions that evoke positive emotions.
- Designing a user experience that is intuitive, engaging, and delightful.

3 User Stories

4 Functional Requirements

4.1 Authentication

- Registration
  - Can register using username and password
  - Can register using Google account
- Login
  - Can login using username and password
  - Can login using Google account
- Log out
  - A user has a means whereby they can log out of their account
- Update password
  - If a user has forgotten their password, they may securely reset it
- Remove account
  - A user can delete their account

4.2 Domain management

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

4.3 Data Visualization and Statistics

- A user can view a select sample of data that was retrieved from their specified sources
- 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
- A user can view data visualizations for sentiment data, meta-data, and optionally: Time-series data

4.4 Sentiment Analysis

- Sentiment analysis can be performed within any of the following groupings
  - For the domain as a whole (this includes all data across all specified sources)
  - Per data source (this considers all the data retrieved from one specific source)
- The results of sentiment analysis on a grouping are returned as follows
  - The ratios of positive, negative, and neutral sentiment
  - An objectivity-subjectivity score
  - An overall sentiment score (from negative to positive)
  - An overall categorization within the following groups
    - Very negative
    - Negative
    - Negative-to-Neutral
    - Neutral
    - Neutral-to-Positive
    - Positive
    - Very positive
- Meta-data is returned whenever a user performs sentiment analysis on an entire domain
  - The meta-data to be returned is as follows
    - Which analysis sources were consulted (ex: Twitter, Instagram, etc.)
    - How many pieces of data from each source were considered
    - An indication of the timeframe over which the data was produced
    - Whether new data needed to be retrieved from the web
    - Display metrics pertaining to how quickly data was retrieved
    - 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
- Optional: Time-series data

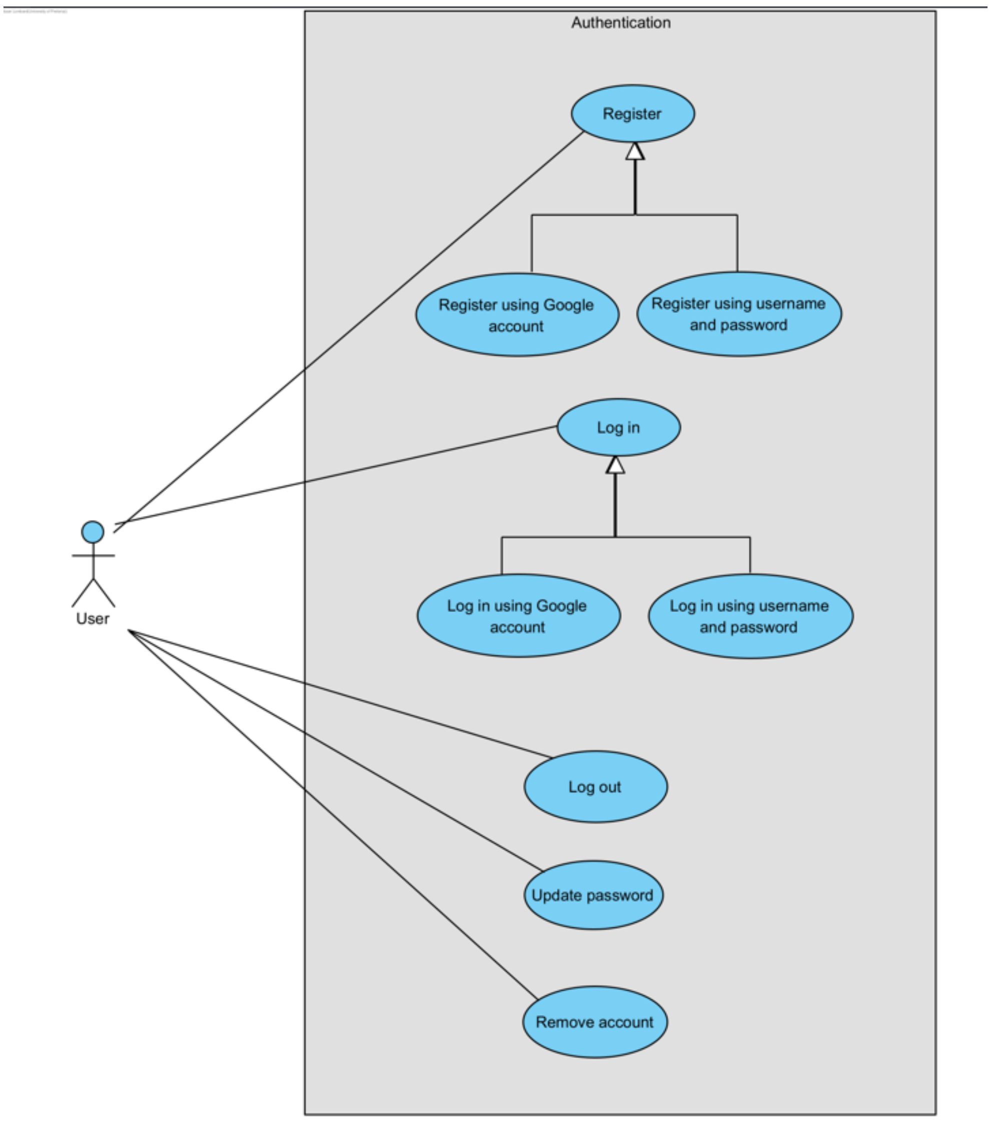
  - Time-series data is returned whenever a user performs sentiment analysis
  - The following time-series data will be returned
    * The change in sentiment score (negative to positive) over a period of time.
    * A prediction of the future trend of the sentiment of the domain

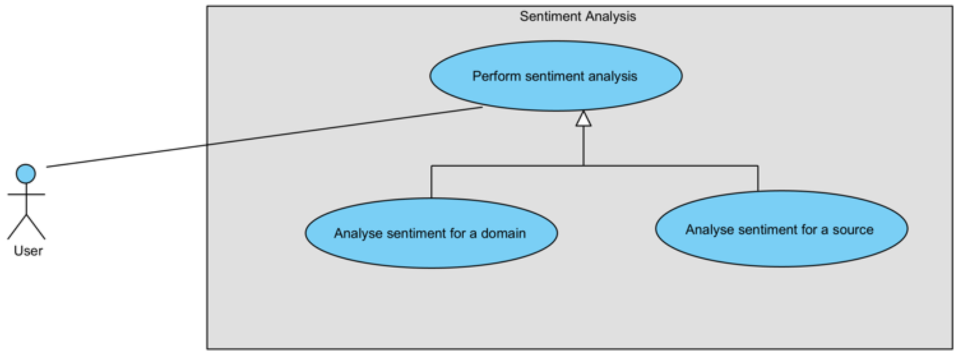
4.5 User Profiles

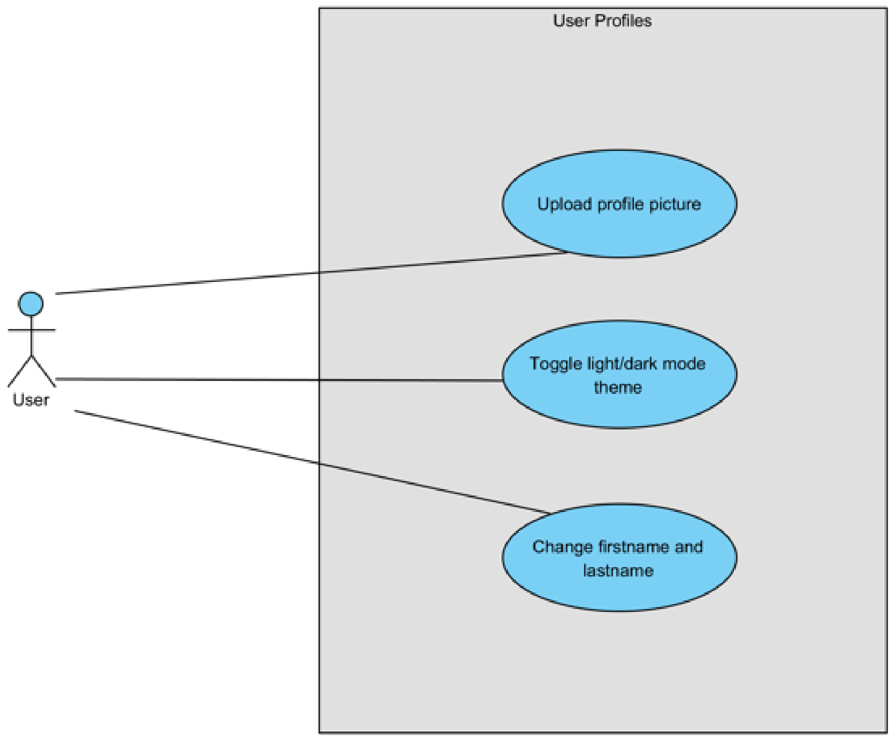
- The domains a user wants to track are stored, this includes:
  - The sources for the domain
  - Optional: The keywords to specifically monitor
- Personalization and preferences
  - User can specify either dark mode or light mode
  - User can upload a profile image
  - User can change their first name and last name

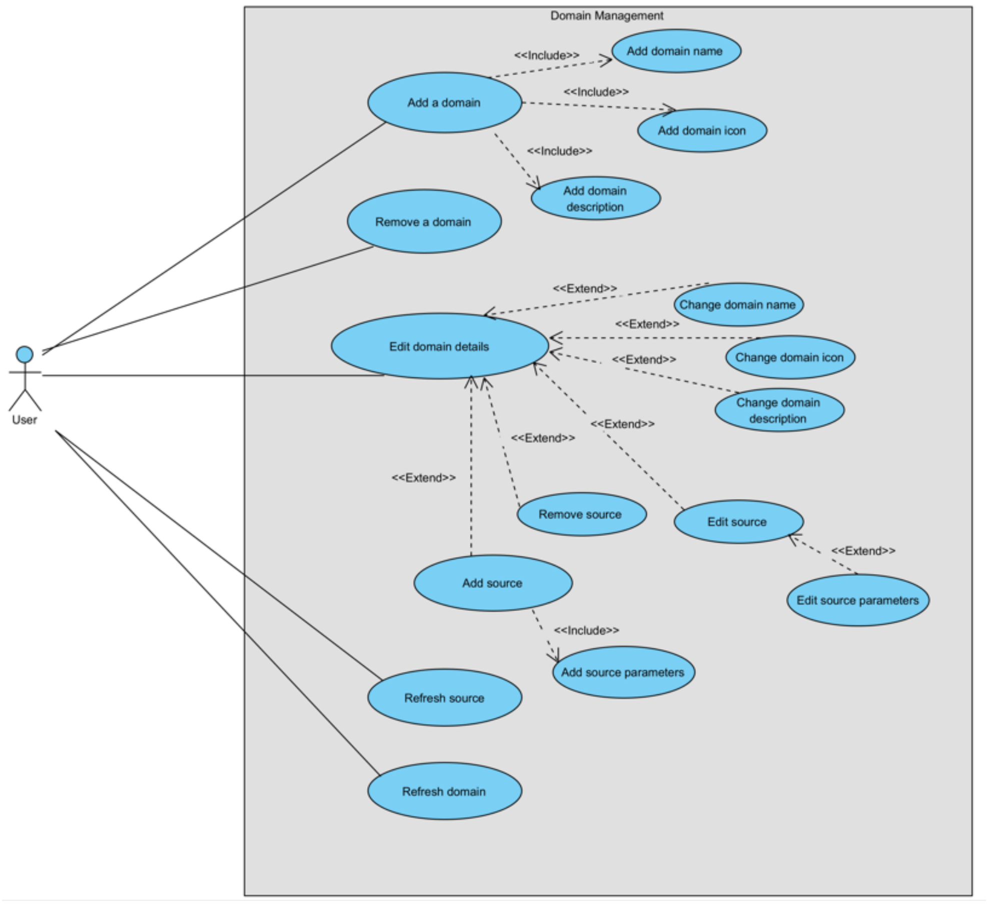
4.6 Requirements

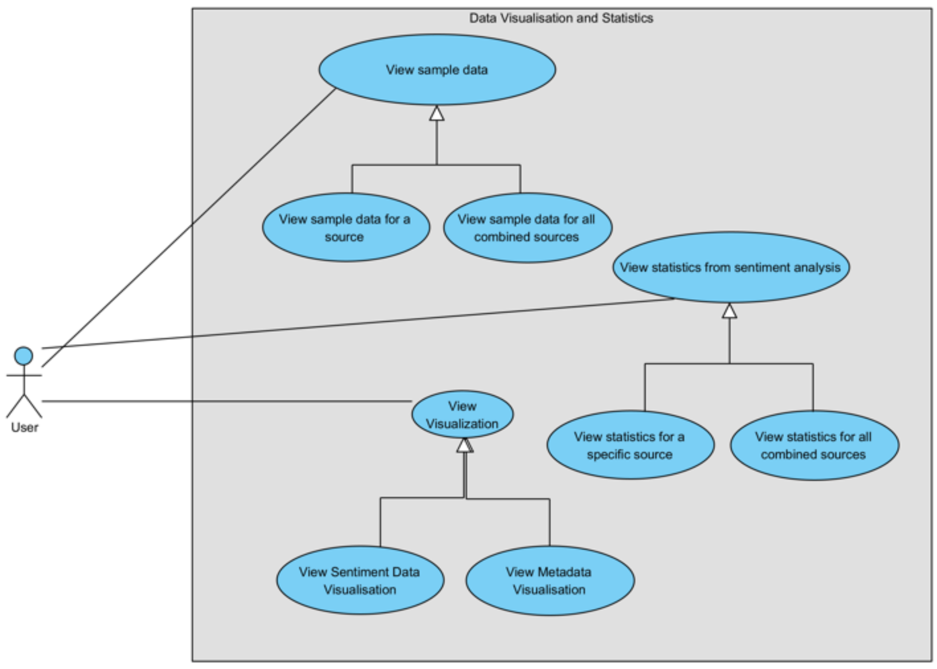
4.7 Use Case Diagrams


Authentication
Register
Register using Google account
Register using username and password
Log in
Log in using Google account
Log in using username and password
Log out
Update password
Remove account
User


Sentiment Analysis
Perform sentiment analysis
Analyse sentiment for a domain
Analyse sentiment for a source
User


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


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


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

6 Contract Design

7 Database Design

7.1 ProfileManager

7.2 User

- `username` : string
- `first_name` : string
- `last_name` : string
- `email` : string
- `password` : string
- `userID` : string

7.3 Profile

- `profileIcon` : string
- `mode` : `ModeEnum`
- `domainIDs` : string []

7.4 ModeEnum

- `LIGHT`
- `DARK`

7.5 DataWarehouse

7.6 SentimentRecord

- `timestamp` : `Timestamp`
- `content` : string
- `sentimentMetrics` : `SentimentMetrics`
- `sourceID` : string

7.7 SentimentMetrics

- `overallScore` : number
- `positiveRatio` : number
- `neutralRatio` : number
- `negativeRatio` : number
- `objSubjScore` : number
- `classification` : `SentimentClassificationEnum`

7.8 SentimentClassificationEnum

- `VERY_NEGATIVE`
- `NEGATIVE`
- `SOMEWHAT_NEGATIVE`
- `NEUTRAL`
- `SOMEWHAT_POSITIVE`
- `POSITIVE`
- `VERY_POSITIVE`

7.9 AnalysisEngine

7.10 Domain

- `Name` : string
- `sources` : `Source` []
- `description` : string
- `icon` : string

7.11 Source

- `platform` : `PlatformEnum`
- `queryString` : string

7.12 PlatformEnum

- `TWITTER`
- `FACEBOOK`
- `INSTAGRAM`
- `REDDIT`
- `GOOGLE_REVIEWS`
- `GOOGLE_NEWS`

8 Class Diagram

9 Architectural Requirements

Quality Requirements

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

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

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.

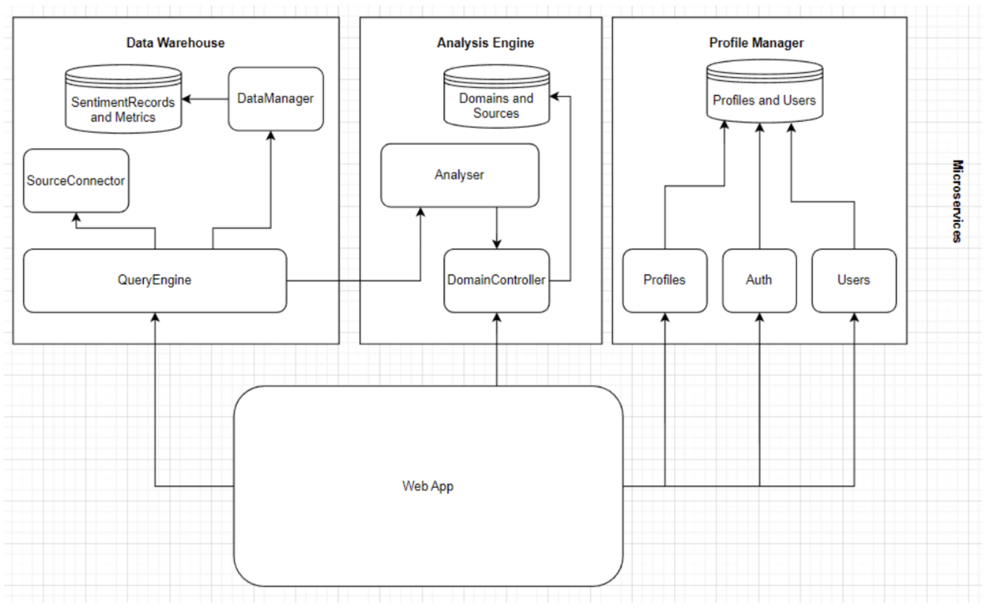
Cost Constraints

- 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 Quality Requirements

11 Architectural Patterns and Tactics

12 Design Patterns

13 Constraints

14 Technology Requirements