BITBYBIT

BITBYBIT

COS301: SOFTWARE ENGINEERING

TWITTER SUMMARISER: SYSTEM REQUIREMENTS SPECIFICATION

By submitting this assignment I confirm that I have read and am aware of the University of Pretoria's policy on academic dishonesty and plagiarism and I declare that the work submitted in this assignment is my own as delimited by the mentioned policies. I explicitly declare that no parts of this assignment have been copied from current or previous students' work or any other sources (including the internet), whether copyrighted or not. I understand that I will be subjected to disciplinary actions should it be found that the work I submit here does not comply with the said policies.

Tlholo Koma	20575085
Isheanesu Dzingirai	20536951
Jonah Gasura	20592061
Gabriel Shoderu	16186258
Batsirai Dzimati	20456078

Contents

List of Figures

1 Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the Twitter Summariser. It will explain the purposes and feature of the system and what the system will do. This document is intended for the project owners, COS 301 lecturers and the developers of the system.

1.2 Scope

This software system will be a Twitter Summariser for a user who would like summaries of most talked-about topics on Twitter.The system will be designed to provide meaningful summaries from tweets of the user's choice in an article-like format to users by providing a tool to curate tweets and to generate such articles. By providing such a system, this system's users will get highlights of discussions on Twitter of their choice.

More specifically, this system is designed to allow users to search for topics using keywords to extract tweets to generate a report. The system will facilitate communication between other users.

1.3 Definitions, Acronyms, and Abbreviations

Creator Person who can generate, edit, share and publish reports.

Database Collection of information monitored by the system.

Twitter Online news and social networking site.

Reader Person who can read pulished reports only.

1.4 References

Paul Gil. (2019). What Exactly Is Twitter? And What Is "Tweeting"? Lifewire.
https://www.lifewire.com/what-exactly-is-twitter-2483331

1.5 Overview

In section 2 of this document, a class diagram of the this system will be provided. This diagram will illustrate the software solution by illustrating different entities of the system and the relationship between the entities of the system.

The third section of this document details the user characteristics of the system users.

The fourth section of this document is written primarily for the developers and describes in detail the system requirements of the product.

The fifth section of this documents specifies the quantity of each quality requirements for this system.

In section 6, a requirements vs subsystems matrix is provided to illustrate the allocation of each system requirement to a subsystem.

2 User Characteristics

The system features two categories of yours: a report creator and a report reader.

- Creator
  - A report creator is a user of the system who has signed up the software and sign's in to the system to use creator level privileges.
  - A report creator is able to make, publish, edit, manage and share reports
  - A report creator also has reader level privileges having the ability to read published reports
- Reader
  - A report reader need not sign-in nor sign-up to use their reader level privileges.
  - A report reader is able to read and share published reports.
  - A report reader does not have creator level privileges, they need to elevate there account to a creator to use creator level privileges.

3 User Stories

User Story 1

As a creator, I want to search a topic so that I receive a list of tweets about that topic and I am able to filter and sort this list.

User Story 2

As a creator, I want to generate a report based on a topic so that I receive a shareable link that is already saved to my profile containing a summary of the tweets on that topic in a form of a report.

User Story 3

As a creator, I want to view the list of tweets returned when generating a report so that I can organise and decorate the list of tweets.

User Story 4

As a creator, I want to edit a report I have generated so that I can add custom text and styles to the report, as well as add custom tweets.

User Story 5

As a creator, I want to publish a report I have generated so that it can be made public for other users to read.

User Story 6

As a creator, I want to manage the results sets returned by the Twitter API so that I can use the same result set to generate another report.

User Story 7

As a creator, I want to delete a report I have generated so that it doesn't exist anymore.

User Story 8 As a creator, I want to clone a report I have generated so that I can have a copy of this generated report.

User Story 9

As a creator, I want to share report I have generated with other creators so that they are able to edit and clone the report which I have shared with them.

User Story 10

As a creator, I want to schedule a periodic report for a specific keyword so that I don't have to manually generate a report for the same keyword every time I need to.

4 Functional Requirements

4.1 Requirements

FR1: Twitter Summariser must allow a creator to search for topic using a keyword or term input to generate report

FR2: Twitter Summariser must allow a creator to filter search results by time-range, non-replies and minimum number of likes

FR3: Twitter Summariser must allow a creator to sort search results by most likes, comments and retweets

FR4: Twitter Summariser must allow a creator to save generated report draft

FR5: Twitter Summariser must allow a creator to edit generated reports

- The creator must be able to add custom text to generated report.
- The creator must be able to choose custom styles for the generated report.

FR6: Twitter Summariser must allow a creator to publish generated reports

FR7: Twitter Summariser must allow a creator to share generated reports

- The creator must be able to share reports with other creators.
- The creator must be able to clone shared reports
- The creator must be able to edit shared reports

FR8: Twitter Summariser must allow a creator to clone shared reports

FR9: Twitter Summariser must allow a creator to edit shared reports

FR10: Twitter Summariser must allow a creator to schedule periodic reports for specific keywords

- The creator must be able to specify topic to generate periodic reports for.
- The creator must be able to choose when to generate periodic reports.
- The creator must be able to publish scheduled reports.

FR11: Twitter Summariser must allow a creator to manage previously created reports.

- The creator must be able to delete previously generated reports.
- The creator must be able to edit previously generated reports.
  - The creator must be able to add custom text to previously created report.
  - The creator must be able to add/change custom style to previously created report.

FR12: Twitter Summariser must allow a new creator to create an account

FR13: Twitter Summariser must allow an existing creator to log into their account

FR14: Twitter Summariser must allow a creator to explore published reports

FR15: Twitter Summariser must allow a reader to explore published reports

FR16: Twitter Summariser must allow a creator to generate a report from result sets

4.2 Subsystems

	Subsystem	Functional Requirement(s)
SS1	Collaboration	FR5; FR6; FR7
SS2	Management	FR2; FR3; FR8; FR9
SS3	Generation	FR1
SS4	Publication	FR4; FR12; FR13
SS5	Authentication	FR10; FR11

5 Quality Requirements

5.1 Performance

The responsiveness of the user interface must be almost immediate. An excellent response time for the service layer to deliver the report is expected.

5.2 Modifiability

The system must be developed in a way that it can be expandable to add more features to the system.

5.3 Security

The system must protect the sensitive data of a system's users. The system must maintain the integrity of the system's users data. All the system's users data must be made available for legitimate use. The system must ensure the user's information is not visible to outside parties. The system must ensure the user's credentials are not visible to outside parties and any developer who has access to the system's data.

5.4 Availability

A requirement of the system will and must be functional and available for service at all times. The system must be up for about 99% of the time.

5.5 Usability

The user interface of the software must be easy to use.It will be usable with no need of tutorial, but users will be guided through tool tips.

5.6 Reliability

The system must ensure the integrity of the data the system provides.

6 Use Cases

UC1: Register/Login The Creator enters authentication details to enter web app.

Before this use case can be initiated, the creator has already accessed the Twitter Summariser Website.

1. If a user does not have a Twitter Summariser account, then the user is prompt to create one.
   (a) The system will register the new user as a creator.
   (b) The system presents the user with the login option.
   (c) The new creator is prompted to enter credentials.
   (d) The system provides the creator access to use the system.
2. If a creator has a Twitter Summariser account, then the creator is prompted to enter credentials.
   (a) The system provides the creator access to use the system.

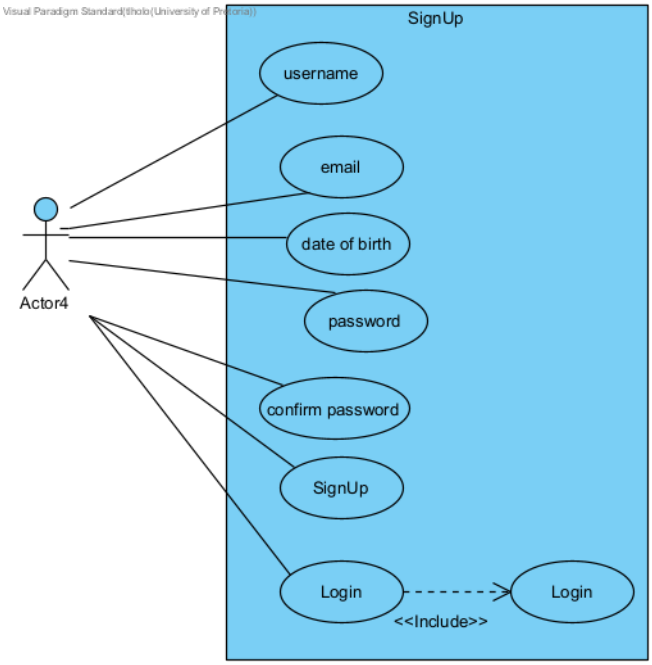

Figure 1: Use case diagram for sign up

UC2: Search for topic The Creators enters keyword to generate report.

Before this use case can be initiated, the creator has already connected to the Twitter Summariser Website.

1. The creator enters keyword or topic to search.
2. The system presents a choice for entering keywords or topics.
3. The creator can filter the search by time range, minimum number of likes and number of tweets to return.

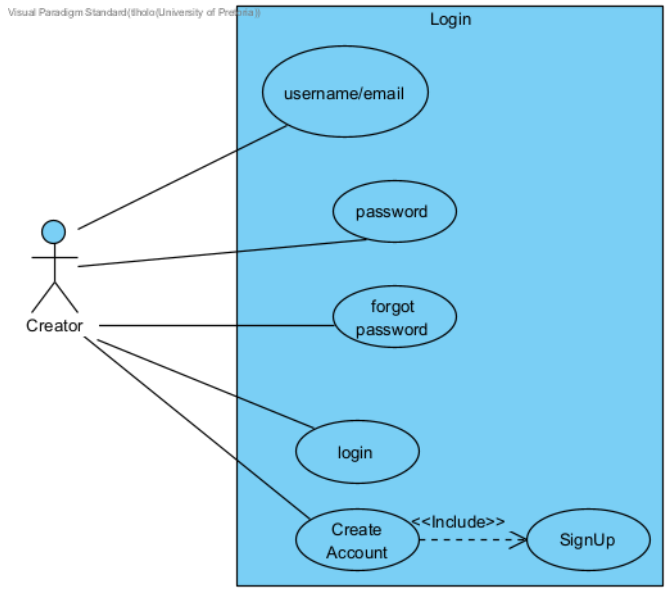

Figure 2: Use case diagram for login

4. The system presents a choice of filters(time range, minimum number of likes and number of tweets to return).
5. The creator can sort the search results by most replies and most retweets.
6. The system presents a choice of sorting by most replies and most retweets.

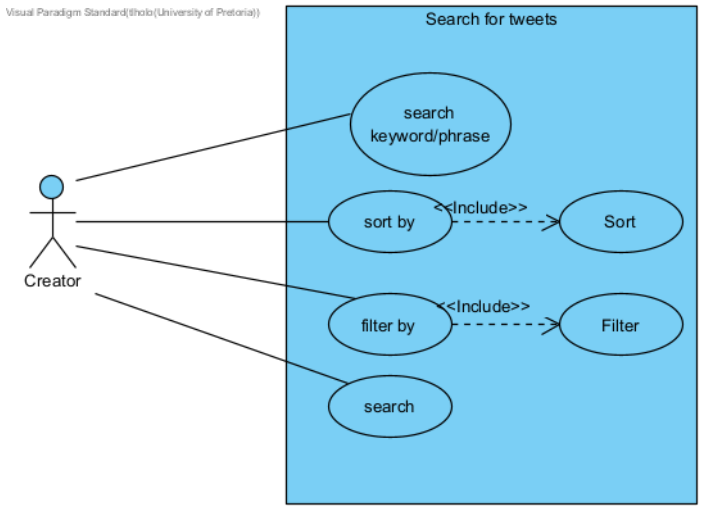

Figure 3: Use case diagram for search

UC3: Sort searched tweets The creator selects a sort option from a drop down list to sort the results of the search by.

1. The creator expands the sort by list.
2. The system presents a choice to sort by likes, comments or retweets.
3. The creator can choose what to sort by from the provided sort options.

4. The system sorts the searched tweets by option selected by creator.
5. The system returns sorted tweets to the creator.

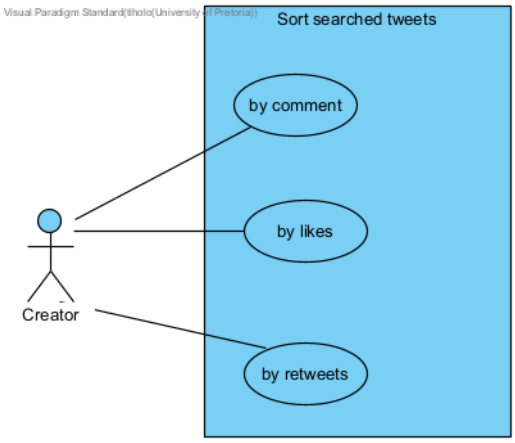

Figure 4: Use case diagram for sort

UC4: Filter searched tweets The creator selects a filter option from a drop down list to sort the results of the search by.

1. The creator expands the filter by list.
2. The system presents a choice to filter by verified users, date range or non-replies.
3. The creator can choose what to filter by from the provided filter options.
4. The system filters the searched tweets by option selected by creator.
5. The system returns filtered tweets to the creator.

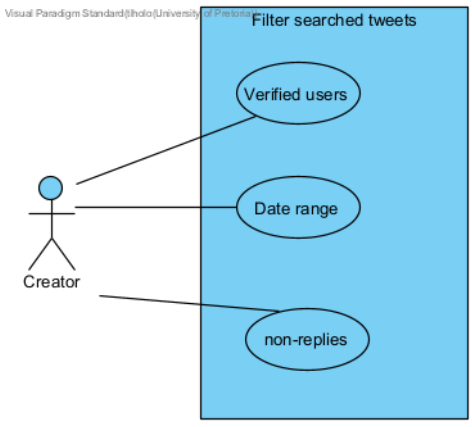

Figure 5: Use case diagram for filter

UC5: Generate report from keyword search The Creator pushes button to generate report.

Before this use case can be initiated, the creator has already connected to the Twitter Summariser Website.

1. The Creator selects to *Generate Report*.
2. The system returns a report in the form of a hyperlink.

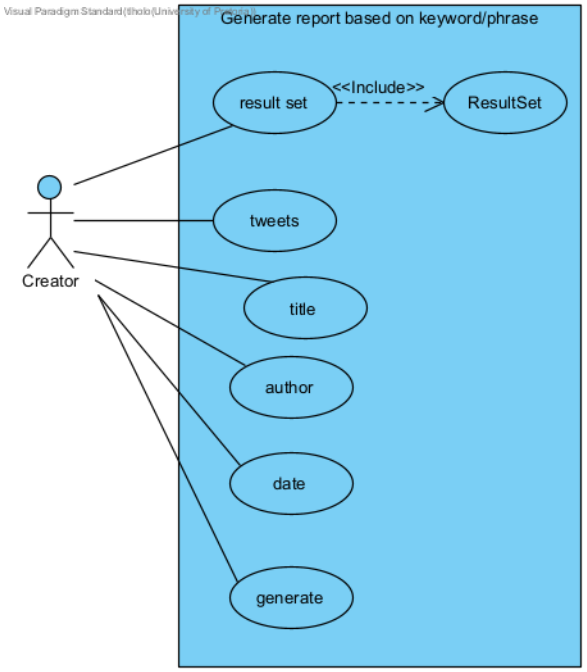

Figure 6: Use case diagram for generate report from keyword search

UC6: View result sets The Creator can view result sets (history of the search) on previous searches

1. The creator navigates to History.
2. The system presents previously searched keywords/phrases.
3. The creator chooses a result set.
4. The system presents the list of tweets from the selected result set.

UC7: Generate report from result sets The Creator generates a report from a result set after viewing the result set.

1. The creator navigates to History.
2. The system presents previously searched keywords/phrases.
3. The creator chooses a result set.
4. The system presents the list of tweets from the selected result set.
5. The creator selects to *Generate Report*.
6. The system returns a report in the form of a hyperlink.

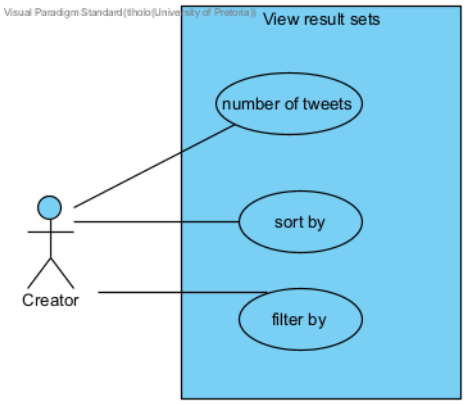

Figure 7: Use case diagram for view result sets

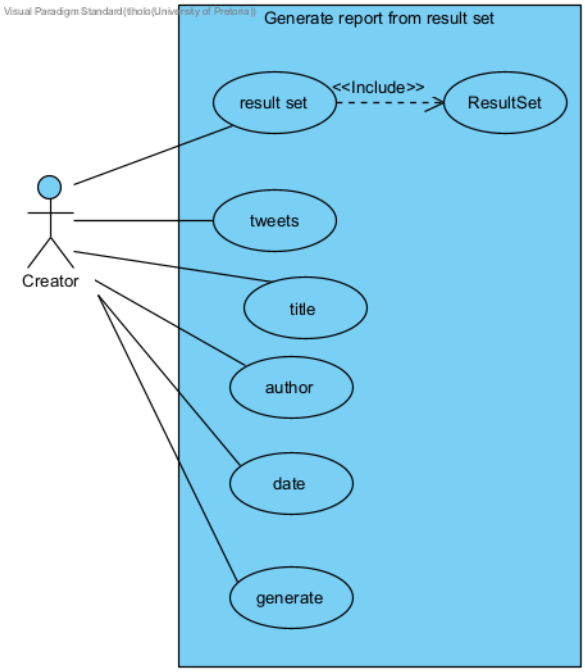

Figure 8: Use case diagram for generate report from result sets

UC8: View draft report

1. The creator navigates to Drafts.
2. The system presents draft reports.
3. The creator chooses a draft report.
4. The system presents the draft report.

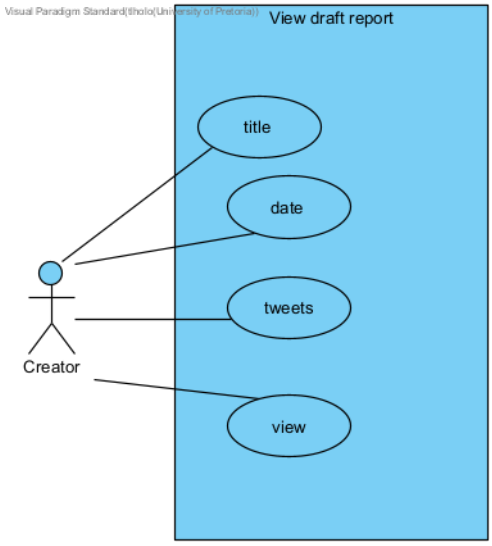

Figure 9: Use case diagram for view draft report

UC10: Share generated report
The Creator shares report with other creators.
Before this use case can be initiated, the creator has already connected to the Twitter Summariser Website.

1. The creator selects a saved report.
2. The system presents the selected report.
3. The creator chooses to share report.
4. The system presents an input to enter username of creator to share to.
5. The system notifies the other creator about shared report.
6. The system sends shared report link to other creator.

UC11: Edit shared report The other creators edits shared reports.
Before this use case can be initiated, the creator has already connected to the Twitter Summariser Website.

1. The creator selects *Shared reports*
2. The system provides the creator with shared reports.

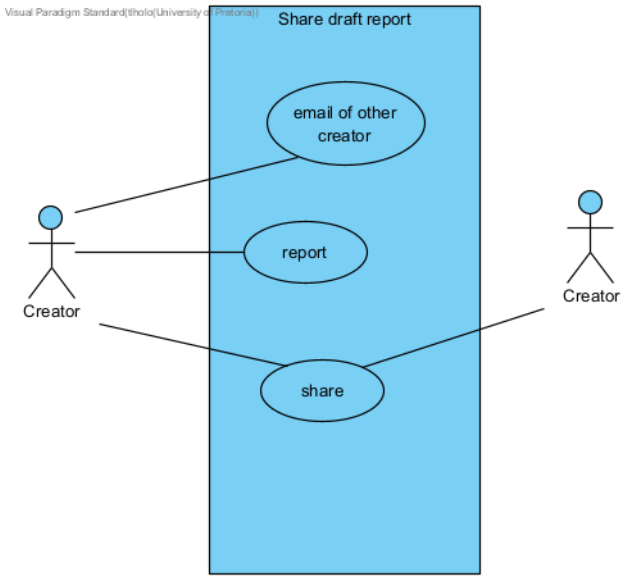

Figure 10: Use case diagram for share generated report

3. The creator selects a shared report.
4. The system presents the creator with the selected report.
5. The system presents the creator a choice to edit shared report.
6. The creator is given the ability to edit the shared report.

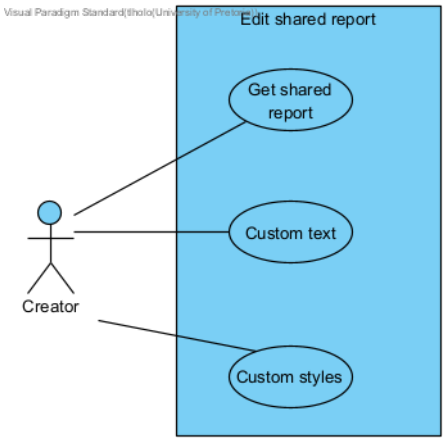

Figure 11: Use case diagram for edit shared report

UC12: Clone shared report The other creators can clone shared reports.

Before this use case can be initiated, the creator has already connected to the Twitter Summariser Website.

1. The creator selects *Shared reports*
2. The system provides the creator with shared reports.
3. The creator selects a shared report.

4. The system presents the creator with the selected report.
5. The system presents the creator a choice to clone shared report.
6. The system provides the creator with a copy of shared report.

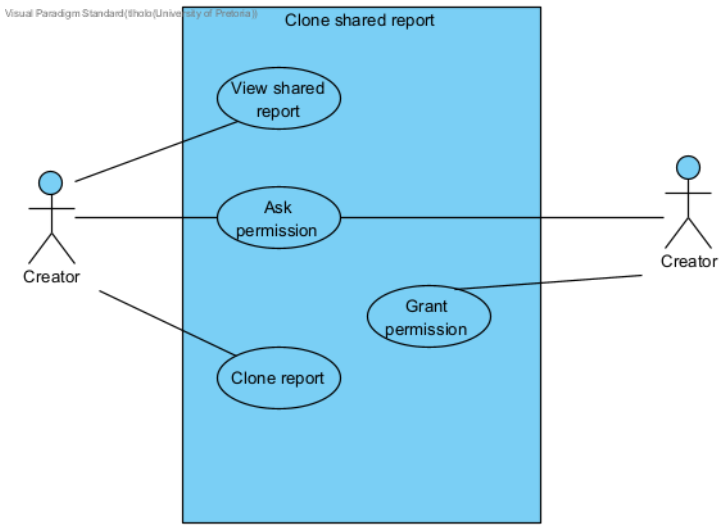

Figure 12: Overall Use case diagram for clone shared report

UC13: Manage reports The Creator can edit and delete previously generated reports.

Before this use case can be initiated, the creator has already connected to the Twitter Summariser Website.

1. The creator selects *Drafts*
2. The system presents the creator with generated reports.
3. The system presents the creator a choice of deleting or editing a report.
4. The creator chooses to delete or edit.
5. The system links to the Twitter Summariser database.
6. If the creator is deleting a report, the system removes report from Twitter Summariser database; else if the creator is editing a report, the system returns the report for editing.

UC14: Schedule reports The Creator can schedule automatic generated reports for specific keywords.

Before this use case can be initiated, the creator has already connected to the Twitter Summariser Website.

1. The creator enters the Twitter Summariser Website.
2. The system presents the creator with interactive reports generated periodically based on specific keywords.
3. The system presents the creator a choice to publish report.

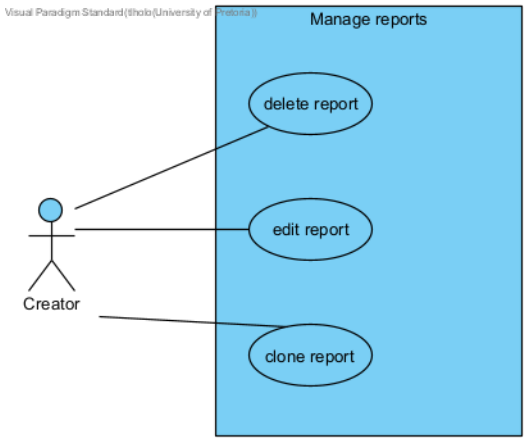

Figure 13: Use case diagram for manage reports

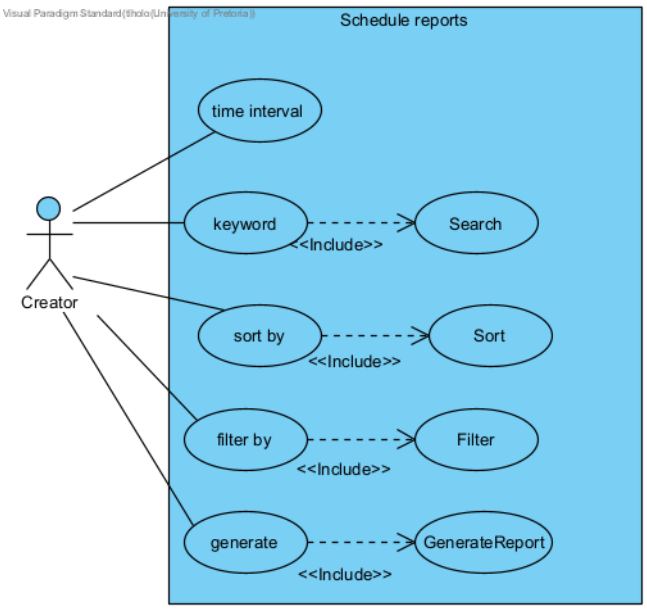

Figure 14: Use case diagram for schedule reports

UC15: Publish generated reports The Creator can publish generated reports.

Before this use case can be initiated, the creator has already connected to the Twitter Summariser Website.

1. The creator selects *Drafts*
2. The system presents the creator with generated reports.
3. The creator selects a report
4. The system presents the creator with the selected report.
5. The system presents a choice of publishing a report.
6. The creator chooses to publish report.
7. The system makes the report public to the Twitter Summariser community.

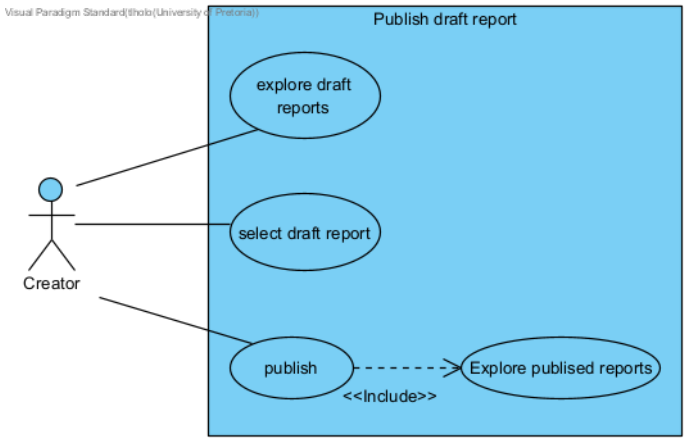

Figure 15: Use case diagram for publish draft report

UC16: Explore published reports The Creator can explore published reports.

Before this use case can be initiated, the creator has already connected to the Twitter Summariser Website.

1. The creator selects *Explore*
2. The system returns published reports

UC17: Install WebApp The Creator can install the Twitter Summariser web app.

1. The creator opens web app in browser.
2. The creator clicks the install button in address bar of the browser.
3. The system installs the web app as an app.
4. The system creates a desktop icon of the app.

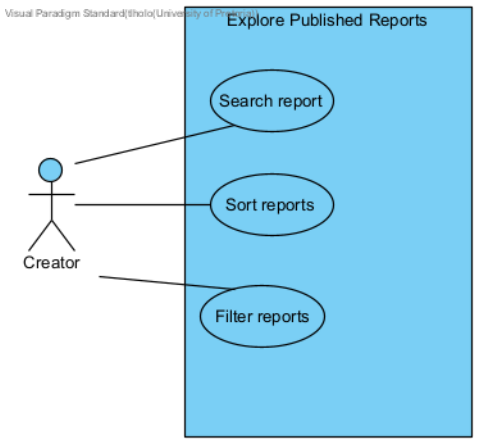

Figure 16: Use case diagram for explore published reports

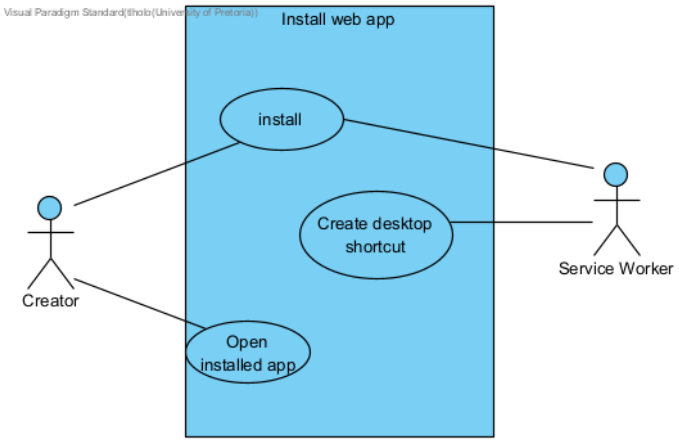

Figure 17: Use case diagram for install web app

7 Trace-ability Matrix

		Subsystem				
		Collaboration	Management	Generation	Publication	Authentication
Requirement	Priority					
FR01	1			X		
FR02	3		X			
FR03	4		X			
FR04	3				X	
FR05	3	X				
FR06	3	X				
FR07	4	X				
FR08	5		X			
FR09	3		X			
FR10	2					X
FR11	2					X
FR12	3				X	
FR13	3				X	
QR5.1	-		X	X		
QR5.2	-	X		X		
QR5.3	-		X			X
QR5.4	-		X		X	
QR5.5	-	X	X		X	
QR5.6	-	X				

8 Class Diagram

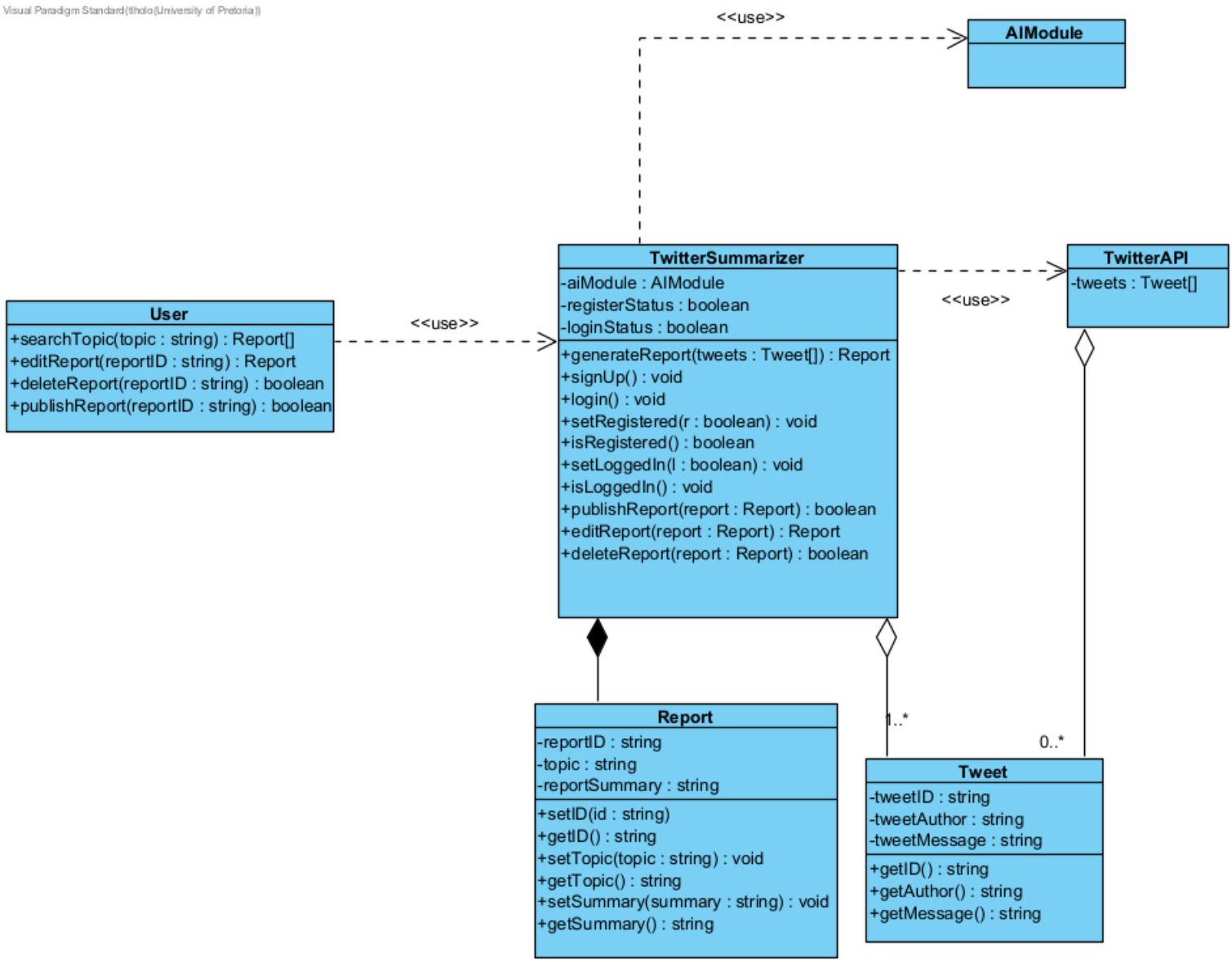

Figure 18: Twitter-Summarise Overall Class Diagram

9 Architectural Structure

9.1 Architectural design strategy

The main strategy implemented for the development of the Twitter Summariser architecture is the design based on quality requirements strategy, whereby the initial step is the identification of the system quality (non-functional) requirements and then design the architecture based on these established requirements.

9.2 Architectural quality requirements

The following quality requirements exist in order from highest to lowest priority.

- **QR1: Reliability**
  - Reliability is the most important quality requirement and is focused on how well the system maintains a certain level of quality over time and thus is the consistency of the subsystems within the system.
  - Reliability is crucial for the following subsystems:
    - Keyword Search:
      - A logged in user should always be allowed to search for a particular keyword on the home page
      - Tweets must always be returned for every keyword for which there are tweets that are linked to that particular keyword
      - Filters and sort criteria applied should always be correctly applied in the results returned from the search
    - Report Generation:
      - If tweets are successfully returned for a particular keyword, then a report should always be returned when the generate report action is triggered
    - Managing reports:
      - A user should always have access to reports they have created
      - A user should always have access to reports that have been shared to them
      - Any modifications made to a report should always be reflected when these changes are saved.
      - Any deleted reports should no longer be accessible once a user has deleted them
    - Sharing reports:
      - A report shared to a user needs to be in the same state that it was shared in when that user receives that report
- **QR2: Performance**
  - Performance is significant to ensure the system's ability to meet certain timing requirements.
  - Performance is crucial for the following subsystems:
    - Keyword Search:

        - Tweets should be correctly returned relatively quickly when a user searches for a particular keyword
    - Report Generation:
        - Once “generate report” is triggered, if tweets exist, the report must be returned in a reasonable amount of time
    - Exploring Reports:
        - Published reports should be returned to each user’s explore page in a reasonable amount of time
    - Sharing reports:
        - Sharing a report should be reflected in a reasonable amount of time.

- **QR3: Security**
  - Security is significant to ensure confidentiality, integrity and availability throughout different aspects of the Twitter Summariser system.
  - Security is crucial for the following subsystems:
    - Authentication:
        - The system should only allow registered users to log in to use the system – if unregistered, the user must first sign up and then log in to access the system. This ensures only authorised access to the Twitter Summariser system
    - Report Generation:
        - A user should be the only one with access to the draft report they generate unless they share it with other users and only then may those other people access these drafts
    - Managing reports:
        - The Owner of a report should be the only user with full access rights to that report – viewing, editing, publishing, “unpublishing” and deleting
        - The Editors of a report should only be allowed to view and edit that particular report
        - The Viewers of a report should only be allowed to view that particular report
    - Sharing reports:
        - The owner of a report should be the only one allowed to share a report that they have created that is not published
        - The owner of a report is the only one able to enable someone else to be an editor for that same report by sharing it to them for “Viewing Editing”
        - A viewer of a report shared to them may only view that report

- **QR4: Usability**
  - Usability is focused on user support and how easy it is for a user to perform certain tasks.
  - Usability is crucial for the following subsystems:
    - Keyword Search:
        - A user should not struggle to enter a keyword to search for tweets
        - A user should easily be able to apply filtering and sorting for the tweets returned for the keyword they search for

        - The user is limited in the sorting and filters the apply to prevent them from applying anything that does not exist and thus preventing faults
        - A user must be informed when any kind of loading is taking place to fetch the results for the search to prevent uncertainty
    - Managing reports:
        - A user should be able to easily edit documents they are authorised to edit
        - A user should be able to easily delete reports they are authorised to delete
    - Exploring reports:
        - A user should be able to easily explore published reports

- **QR5: Availability**
    - Availability is significant to ensure the system is always ready to perform particular tasks when it is required. This requirement is closely related to the Reliability requirement.
    - Availability is crucial for the following subsystems:
        - Authentication:
            - If a user is registered and they enter valid credentials when logging in, the system should always allow them to access the system
            - A valid user should never be denied access into the system when their credentials have been used to sign up
        - Exploring reports:
            - Publicly available should always be available for all Twitter Summariser users to explore

- **QR6: Modifiability**
    - Modifiability is significant to ensure the adaptability of the system and to minimize the risk of making changes to and within the system.
    - Modifiability is crucial for the following subsystems:
        - Keyword Search:
            - The subsystem should allow for changes to easily be made to the sort and filter options without negatively impacting any other elements of the system

9.3 Architectural styles

The system makes use of the following two (2) main architectural styles:

- **AS1: Layered Architecture**
  - The layered architecture consists of the following layers in the context of the Twitter Summariser:
    - Presentation Layer:
      - The Presentation layer mainly consists of the system's user interface that a user directly interacts with.
    - Business Layer:
      - The presentation layer consists of the components mainly involved in the main processing operations that take place in the system. These components include:
      - **Twitter API:** This is the interface provided by the Twitter application to access real Twitter data.
      - **Semantic Analysis Module:** This is the component responsible for assisting with semantic analysis operations when report text is automatically generated by the system
      - **Serverless Computing Service:** This is the service responsible for using the relevant modules and services to implement certain functionality within the system.
      - **Twitter Summariser API:** This is the main system API that handles user requests and triggers the necessary functions to be executed based on those requests
    - Database Layer:
      - The database layer mainly consists of the main system database where all the relevant data the system will need over time is stored and as well as a ”spare” backup database with the same data.

- **AS2: Client-Server Architecture**
  - The client-server architecture consists of the following layers in the context of the Twitter Summariser:
    - Client Side:
      - The client side of the system consists of the frontend element of the system that deals with direct user interaction.
    - Server Side
      - The server side of the system consists of the backend element of the system that deals with the main system processing and data access.

9.4 Architectural patterns

The system makes use of the following main architectural patterns to implement the chosen architectural styles:

- **AP1: Model View ViewModel (MVVM)**
  - The MVVM pattern is a specific type of the Model View Controller (MVC) pattern whereby the entry point for the system is the View unlike traditional MVC whereby the entry point of the system is the controller. MVVM clearly separates the frontend and backend of the system and allows them to interact with one another through the link known as the ViewModel. This pattern consists of the following layers in the context of the Twitter Summariser:
    - Model:
      - The database layer of the architecture is implemented as the model of the system in MVVM which consists of the database that holds all the system data.
    - View
      - The presentation layer of the architecture is implemented as the view of the system in MVVM which consists of the system user interface that the client directly interacts with.
    - ViewModel
      - The business layer of the architecture is implemented as the ViewModel of the system in MVVM which consists of the system's API that manages interactions with the relative services and modules as well as the Twitter API.
- **AP2: Active Redundancy**
  Within the database Layer of the system, the system implements the active redundancy pattern whereby there is an "active" backup of the original system database. This pattern is essential to achieve the most important quality requirement of the system - reliability - which is achieved by ensuring that in the event that some data cannot be retrieved from the main database, the backup will replace that entry and the system can still access the relevant data with minimum delay
- **AP3: Interceptor Validator**
  Within the Business Layer of the system, the system implements the interceptor pattern whereby there exists a "Validator" that cross checks and evaluates all data coming into the system. This pattern is essential to improve security of the system as data is not immediately stored before proper validation and in the case that there are attempts to make unauthorised access to the system this validator can "intercept" and prevent this action. The Twitter Summariser API is the validator for this system.

9.5 Architectural design

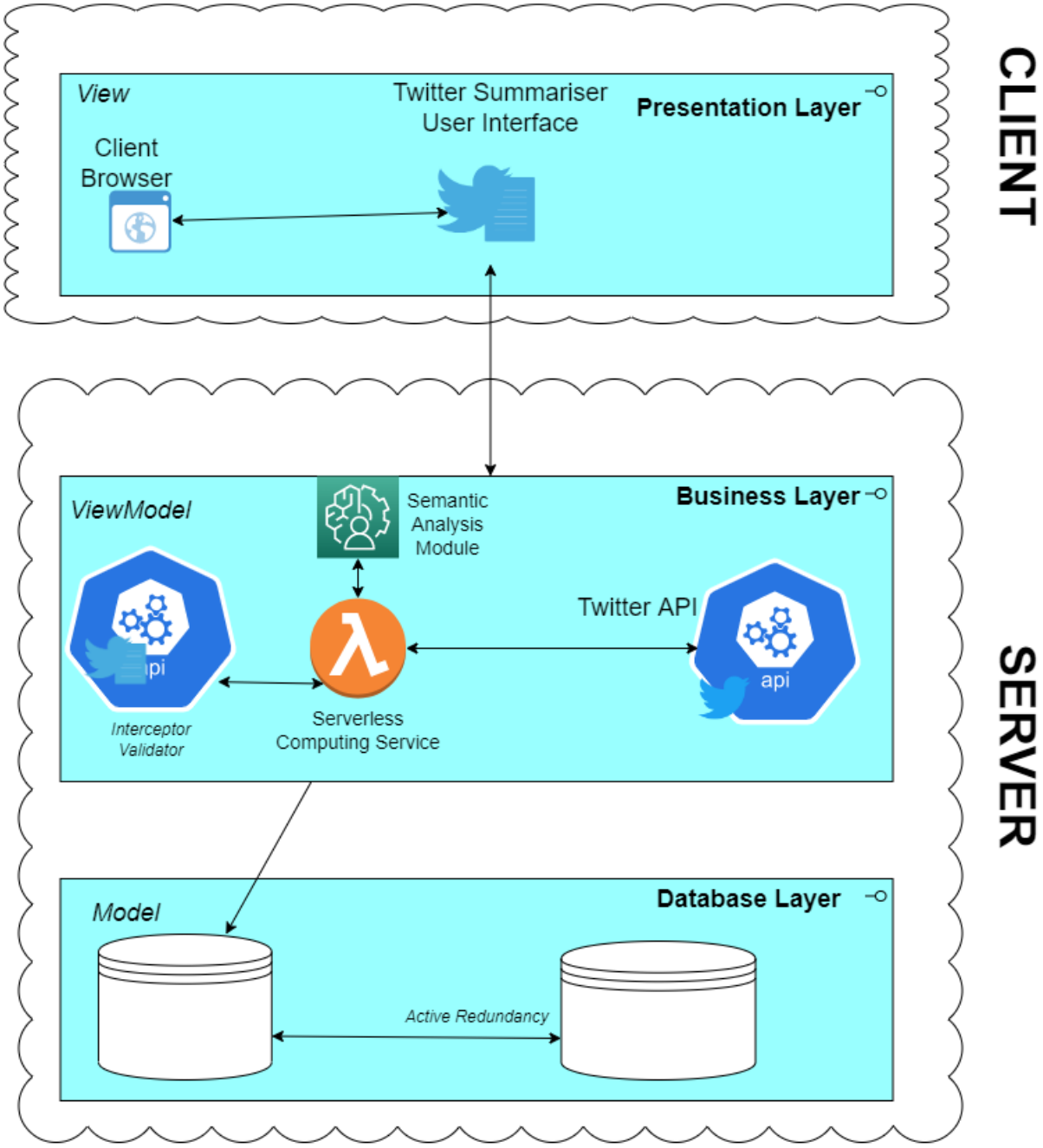

Figure 19: Twitter-Summarise Architectural Pattern

9.6 Architectural constraints

The main system architecture constraints include:

- User requests are restricted to components within the architecture
- System requests and responses are restricted to components within the architecture
- The client and server components of the Twitter Summariser must not negatively impact one another when modifications are made to the respectively
- The client perspective should be completely unaware of the logical components on the server side.

10 Technology Choice

10.1 Backend

The backend will comprise of functions written in Nodejs and TypeScript, which will then be run on AWS Lambda. AWS Lambda is the perfect tool to use as it allows the project to meet the requirements of being server-less, while also being well scalable enough to allow for potentially many requests and processing of data at a time. Node.js, with TypeScript, will make API calls to the Twitter API, and pre-process the data from the response before it goes to the NLP libraries/module for analysing, after which the results obtained will be further processed to allow for the report creation to be sent to the front-end. Twitter API, and pre-process the data from the response before it goes to the NLP libraries/module for analysing, after which the results obtained will be further processed to allow for the report creation to be sent to the front-end.

Pro:

- Scalable

Con:

- free tier is limited

10.2 AI Module

The optional AI module which is separate from the backend will be coded in Python (using AWS Comprehend as the cloud-based ML service). Python was chosen over JavaScript because in the system design it was stated that the AI module should be written in python. AWS Comprehend will be used along with Python instead of TensorFlow and PyTorch because it has a powerful machine learning model that can be used without any machine learning experience. This will be beneficial as most of the team members have zero to little experience in machine learning and natural language processing algorithms. Since the duration of the project is small, it is better we spent more time developing the Twitter Summarizer and ensuring it meets all the specified requirements than spending most of the limited amount of time we have developing and training models. Using AWS Comprehend will greatly assist in completing our system optional requirement

10.3 Frontend

The available technology choices for the user interface implementation were React and Vue. The chosen technology for the frontend interface of the system was React as it has the following advantages over Vue:

Pros:

- The current Twitter user interface is built on React and thus supports this system as it is closely related to Twitter and thus contributes to the system's consistency.
- React is highly compatible with Node.js and the backend of the system will mainly be implemented using Node.js and it is essential that the front-end implementation is compatible to the backend.

- React was developed before Vue and thus there exist more resources for research purposes when implementing the system.

Cons:

- React uses a syntax extension known as JSX that limits development progress.
- The technology is constantly changing and thus can become a barrier over time.

React implements the View and ViewModel aspects of the Model View ViewModel architecture which is the main architectural patter used in the system's architecture.

10.4 Database

The technology used for database is AWS DynamoDB. AWS DynamoDB is a NoSQL database service offered by Amazon that supports key-value and document data structure.

Pros

- DynamoDB is scaleable.
- DynamoDB ensures data security as it backs up data.

Cons

- DynamoDB does not have no triggers.
- The concept of foreign keys does not exist in DynamoDB.
- DynamoDB does not have server-side scripts.

Because the architecture includes Active Redundancy, DynamoDB is a suitable database service that can create a separate copy of the database automatically to achieve reliability.