UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

COS301 Capstone Project

Team Syntactic Sugar

Jargon Sentiment Analysis
Software Requirements Specification

Team Members
Graeme Coetzee
Christiaan Nel
Ethan Lindeman
Kevin Coetzee
Herbert Magaya

Client
Compiax

May 23, 2019

Contat email: syntacticsugar9@gmail.com

Contents

List of Figures

1 Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the Jargon Sentiment Analysis System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to the client, COMPIAX, for its approval.

1.2 Scope

The Jargon Sentiment Analysis system is a system enabling users to create projects specifying which topics to search for, and other additional configurations such as blacklisted or white-listed words. When users choose to run these projects, they start collecting information from a public source of opinion based data, such as Twitter. The system then analyzes this data, utilizing an Artificial Neural Network, classifying it into positive and negative sentiments. Users will then be able to interpret this result as a final opinion overview, as well as view more detailed information regarding the original data-set.

1.3 Definitions, Acronyms, and Abbreviations

NN - Artificial Neural Network, this refers to a processing framework that roughly resembles the workings of the human brain. It can utilize different different machine learning algorithms to process complex data inputs.[1] In the Jargon Sentiment Analysis system, this is what primarily will be used to analyse and classify sentiments.

2 Information Flow

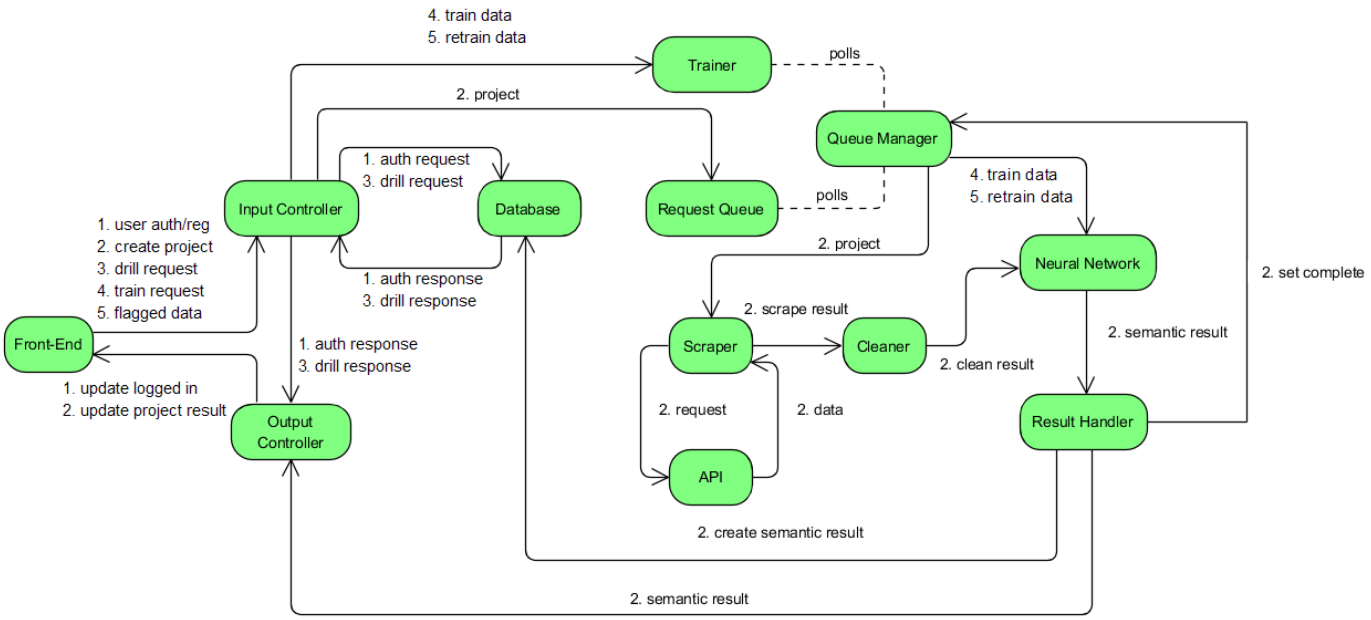

Figure 1: High-Level Information Flow Diagram

3 Domain Model

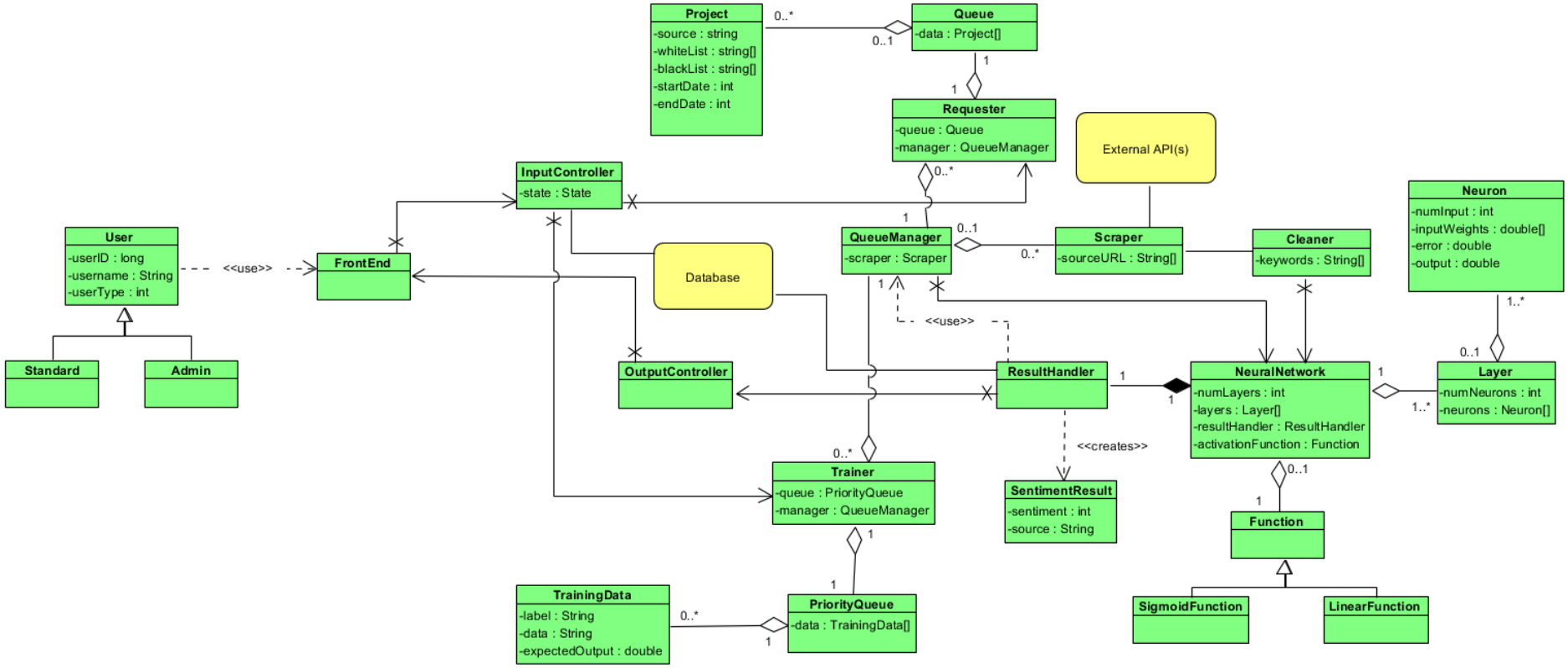

Figure 2: High-Level Domain Model

4 User Characteristics

- Researchers - This sentiment analysis system will at its core be a research tool for determining public opinion on different topics, and therefor the core users will be educated, professional researchers working for different companies and corporations, especially market researchers. These users will use the system to create projects and gauge opinions based on the topics and configurations chosen in these projects, or aid their management staff and higher-ups by setting up and running research projects, leaving the result interpretation up to their management staff.
- Management - Management or high-up personnel will use the end results the system produces as useful information in decision making on matters concerning brand management and product development. Their main use case for the system will be the interpretation of the system's result sets to make educated decisions.
- Administrators - Administrative users will include members of COMPIAX and the Syntactic Sugar team. These users are more knowledgeable on the topics of NNs, and will additionally make use of the system to retrain the NN with new or revised data.

We may refer to *researchers* and *management users* as *regular users*.

5 Architectural Design

The Jargon project shall be implemented using the **Microservices** architectural style. This is so that we can make the system as extendable and maintainable as possible for once it is in the hands of our client. Our client has also requested that we use microservices architecture. Microservices is also a leading architectural style in industry at the moment and has great capabilities. It allows us to be language and technology agnostic when developing different systems and also leads to a cleaner design overall as it forces us to clearly define our subsystems and functionality, thus it has great benefits towards development as a whole. It also benefits users of the system as it allows us to scale the system well and also to provide nearly zero downtime to the system as a whole by instantly restarting any services that go down, this leads to overall better service provision to users.

6 Functional Requirements

R1. Users should be able to interact with the system through a web interface frontend.
R2. Users should be able to register/delete their account.
  R2.1. Users should be able to edit/update their account details.
  R2.2. Regular users should be able to delete their own account.
  R2.3. Admin users should be able to delete any regular user's account.
R3. Users should be able to log in and out to the application.
R4. The system should restrict further application functionality to logged-in users.
R5. Users should be able to create multiple projects and configurations.
  R5.1. Users should be able to queue projects.
  R5.2. Configurations should allow the user to specify options to change the brand to search on.
  R5.3. Configurations should allow the user to specify options to determine when to start/stop streaming data and words that are whitelisted or blacklisted.

R6. Users should be able to compare completed projects.
R7. Users should be able to search for completed and in porgress projects.
R8. The system should be able to stream data (needed by a project) in real-time from a source such as Twitter.
R9. The system should be able to perform sentiment analysis on the data (of a project).
- **R9.1.** The sentiment analysis subsystem should give the sentiment result of a project a rating that can be either positive, negative or neutral.
- **R9.2.** The sentiment analysis subsystem should make use of a neural network to perform the sentiment analysis.
- **R9.3.** The sentiment analysis subsystem neural network model should be able to retrain itself.
- **R9.4.** The sentiment analysis subsystem neural network model should be able to be manually trained/trained by an admin user.
- **R9.5.** Admin users should be able to provide the sentiment analysis subsystem with training data, which the neural network will use to train itself.
- **R9.6.** Users should be able to provide the sentiment analysis subsystem with training data by means of flagging/modifying the results of a project.

R10. Each subsystem should be able to log its activities.
- **R10.1.** Regular users should be able to view logs of activities, directly related to their operations.
- **R10.2.** Admin users should be able to view activity logs of both the entire system in addition to those directly related to their operations.
- **R10.3.** Users should be able to view logs recorded between any valid given timeframe.
- **R10.4.** Users should be able to backup logs.
- **R10.5.** Users should be able to download log backup files.
  - **R10.5.1.** Users should be able to download log backup files to the device which they are interfacing through.
  - **R10.5.2.** Users should be able to download log backup files to a cloud storage service of their choosing.

7 Use Cases

- UC1.1 Register Account (Actor: User, Subsystem: Accounts)
- UC1.2 Login (Actor: User, Subsystem: Accounts)
- UC1.3 Logout (Actor: User, Subsystem: Accounts)
- UC1.4 Edit Account Details (Actor: User, Subsystem: Accounts)
- UC1.5 Remove User (Actor: Admin, Subsystem: Accounts)
- UC2.1 Train Neural Network (Actor: Admin, Subsystem: Neural Network)
- UC2.2 Analyse Data (Actor: System, Subsystem: Neural Network)
- UC2.3 Flag Items (Actor: User, Subsystem: Neural Network)
- UC2.4 Retrain Network (Actor: User, Subsystem: Neural Network)
- UC3.1 Create Project (Actor: User, Subsystem: Project)
- UC3.2 Edit Project (Actor: User, Subsystem: Project)
- UC3.3 Delete Project (Actor: User, Subsystem: Project)
- UC3.4 View Project Results (Actor: User, Subsystem: Project)
- UC3.5 Drill Down Results (Actor: User, Subsystem: Project)
- UC3.6 Compare Projects (Actor: User, Subsystem: Project)
- UC3.7 Search Project (Actor: User, Subsystem: Project)

- UC4.1 Create Log (Actor: System, Subsystem: Logs)
- UC4.2 View Logs (Actor: Admin, Subsystem: Logs)
- UC5.1 Submit Error Report (Actor: User, Subsystem: Support)
- UC5.2 Resolve Error Report (Actor: Admin, Subsystem: Support)
- UC5.3 Submit Bug Report (Actor: User, Subsystem: Support)

8 Subsystems

- Project subsystem
- Accounts subsystem
- Logs subsystem
- Neural Network subsystem
- Support subsystem

Figure 3: Use case diagrams

9 Quality Requirements

Performance

Q1 System response time should be limited to within 3 seconds of a user action. Frequent response delays will be logged and any discrepancies in response time will be analysed and fixed.

Security

Q2 The system will be implemented with access control. Users can only view the projects of other users, only admins have full read/write access. This will be done through the use of third party software, additionally, users may submit error reports in the case of any unexpected behaviour.

Q3 Each user will have a maximum of three failed login attempts before their account is temporarily suspended. More than 10 failed attempts will result in a frozen account.

Safety

Q4 Logs need to be kept in order to keep track of system actions performed in the case of a system breach. Data will be periodically backed up to prevent data loss.

Availability

Q5 The neural network needs to have a downtime of less than 1%. Downtime will be tracked and if it's above 1% then optimization measures will be put into place.

Q6 The system downtime is measured and the number of failures per month will be tracked. These will be monitored daily and we aim to have less than 0.1% risk of downtime.

Q7 We aim to have no downtime when adding a new server to the system as well as when deploying new code. This shall be done using Kubernetes and a continuous deployment tool.

Reliability

Q8 The neural network must be trained so as not to produce too many incorrect results. Users will have the ability to flag incorrect results in order to retrain the network. A count of these incorrect results will be kept in order for the system admin to notice any discrepancies in the network. We aim to have a failure rate of less than 3%.

Maintainability

Q9 The system must run over multiple servers to prevent one central point of failure. Server failures should be monitored and audit logs reviewed in the case of a failure. We aim to have a failure rate of lower than 1%.

Q10 The codebase will be modularized to further allow for easier extension and maintenance. Bug reports may be submitted by users and we aim to have fewer than 1 bug report per month.

Usability

Q11 The system UI needs to be designed taking UX into account. User experience surveys shall be issued to the users in order to determine the overall experience. We aim to have 95% user satisfaction.

Q12 The system needs to be intuitive to use and self-explaining. Usability tests shall be issued to users in order to determine where the system needs improvement. We aim to have 100% task completion in the usability tests.

Scalability

Q13 The system needs to be able to run multiple Twitter listeners in parallel for different projects. We aim to have zero bottlenecking into the neural network. This shall be done using Kubernetes and Docker.

10 Trace-ability Matrix

	Project subsystem	Accounts subsystem	Logs subsystem	Neural Network subsystem	Support subsystem
R.1.					X
R.2.		X			
R.2.1.		X			
R.2.2.		X			
R.2.3.		X			
R.3.		X			
R.4.		X			
R.5.	X				
R.5.1.	X				
R.5.2.	X				
R.5.3.	X				
R.6.	X				X
R.7.	X				X
R.8.				X	
R.9.				X	
R.9.1.				X	
R.9.2.				X	
R.9.3.				X	
R.9.4.		X		X	
R.9.5.		X		X	
R.9.6.				X	
R.10.			X		
R.10.1.		X	X		
R.10.2.		X	X		
R.10.3.			X		
R.10.4.			X		
R.10.5.			X		
R.10.5.1.			X		
R.10.5.2.			X		

Table 1: Traceability matrix mapping subsystems to functional requirements

	Project subsystem	Accounts subsystem	Logs subsystem	Neural Network subsystem	Support subsystem
Q1					X
Q2	X	X			X
Q3		X			
Q4			X		
Q5				X	
Q6			X		
Q7			X		X
Q8				X	
Q9			X		X
Q10			X		
Q11			X		X
Q12					X
Q13				X	

Table 2: Traceability matrix mapping subsystems to quality requirements