



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Architectural and Functional
Requirements
Mindmap PIM
Client: IMINISYS

Team: A-Cube-N

Grobler, Arno Lochner, Amy Maree, Armand
14011396 14038600 12017800

Department of Computer Science, University of Pretoria

Contents

1	Introduction	1
2	Vision	1
3	Background	1
3.1	Future business/research opportunities	1
3.2	The Client's Problem	1
4	Important Terminology	2
5	Architecture Requirements	2
5.1	Access Channel Requirements	2
5.1.1	Human Access Channels	2
5.1.2	System Access Channels	2
5.2	Quality Requirements	2
5.2.1	Performance	3
5.2.2	Usability	3
5.2.3	Reliability	3
5.2.4	Scalability	3
5.2.5	Flexibility	3
5.2.6	Security	3
5.2.7	Auditability/Monitorability	4
5.2.8	Integrability	4
5.3	Integration Requirements	4
5.4	Architecture Constraints	5
6	Architecture Design	5
6.1	Infrastructure	5
6.2	Services	5
6.2.1	Polling Service	5
6.2.2	Processing Service	5
6.2.3	Business Service	6
6.2.4	Database Service	6
6.3	Tactics	6
6.3.1	Flexibility	6
6.3.2	Reliability	7
6.3.3	Security	8
6.3.4	Auditability	9
7	Database and Persistence	9
8	Process Specification	9
8.1	Server	10
8.2	Front End	10

9 Functional Requirements	10
9.1 Use Case Prioritisation	10
9.2 Use Case/Service Contracts	11
9.2.1 Sign up	11
9.2.2 Choose PIMs	12
9.2.3 Login to PIMs	13
9.2.4 Logout	15
9.2.5 Expand Bubble	15
9.2.6 Display Data	16
9.2.7 Remove Bubble	18
9.2.8 Deactivate Account	19
9.2.9 Specify the default depth factor	20
9.2.10 Specify the default branch factor	21
9.2.11 Reply to an email	22
9.2.12 Redirect to Gmail application from our application	23
9.2.13 Set quick access depth factor	23
9.2.14 Set quick access branch factor	24
9.2.15 Reset graph	24
9.2.16 Customise colour scheme	24
9.2.17 Help Centre	24
9.3 Required Functionality	25
10 Technologies	26
10.1 Framework	26
10.2 Web Container	26
10.3 Natural Language Processor	26
10.4 Build Tool	27
10.5 Communication	27
11 Initial Design	27
12 Open Issues	28

1 Introduction

People of this day and age often make use of many technologies and platforms for the purpose of staying in contact with people, sharing moments with friends, communicating with people and organising their day-to-day lives. Generally all the above tasks of a person in the 21st century are done on different platforms for example Facebook, Email and Google Calendar. This project serves to provide the people described above, with a single platform with which they can interact to see their data organised according to topic and make use of some functionality the mined platforms.

2 Vision

The vision of this project is to create an application which extracts data from various existing platforms such as Gmail and Facebook. The application will make use of a natural language processor to extract data, determine the general topic of the data then integrate this new information into the interactive mind map. The hope is that this will simplify the users life by only needing one application to 'monitor' all other platforms on which they might have an account and manage the information of those platforms from our application. The mind map will work in a similar fashion as our minds work. Where the exploration of one topic may lead to the exploration of another topic that is related to the user. Hence the "mind" map, it is a map of your mind visualized by the topics that come up in your life.

3 Background

We got elected for this project by including it as one of our top 3 choices for the projects that we wanted to do most. Once we were chosen for this project, we began to learn more about what the client expected from us.

3.1 Future business/research opportunities

The Mind Mapped PIM project provides an opportunity for us to create something that helps to simplify the user's life. It will do this by organizing all relevant information into one system. It can also help the user to plan out their daily and weekly schedules which can lead to reduced stress on the user.

3.2 The Client's Problem

The client wants to have a way of displaying all relevant information in one place in the form of a mind map. This will help to organize general life events such as meetings and. This information will come from different sources and enables the user to see all relevant information without having to go to each source individually. A key aspect is that Mind Map PIM attempts to do

is to dig through the clutter contained in other PIMs and only display things the user might want to see.

4 Important Terminology

- **Bubbles** Commercial name for the application.
- **Bubble** A single node in the mind map.
- **Bubble Map** The mind map that is presented to the user.
- **PIM** Personal Information Manager such as Google Calendar and social media platforms like Facebook.
- **Data Source** It refers to the source where user data is retrieved from.
- **Poller** The service which will constantly ask a data source for new information and then send new information to the processing service.

5 Architecture Requirements

5.1 Access Channel Requirements

5.1.1 Human Access Channels

Users will access this system through a website. It should be available to anyone wishing to sign up through any browser i.e. Chrome, Mozilla Firefox, Internet Explorer, Safari and Edge provided the user has a Gmail or Facebook account with which to sign up. It is our intention to ensure that the website will be under strict standards-compliance, that is, having the website comply with the World Wide Web Consortium (W3C) (W3C, 2016) so that the website will run on every browser exactly the same. The website should be mobile friendly as to be able to scale down to a phone screen size and still be usable and interactive. Offline capabilities for users who have already used the service before would be a nice-to-have feature.

5.1.2 System Access Channels

Websocket based web services will be used as the primary means of communication between the interface and the front end application. This is due to the bi-directional communication that it offers.

A technology known as RabbitMQ will be used to allow communication between our multiple back end applications such as the polling application, business logic application and the database application.

5.2 Quality Requirements

The following quality requirements have been placed in order of priority.

5.2.1 Performance

Performance has the highest priority since this system has to be real time. Any new information that is updated on the respective websites and email clients should, in the least amount of time possible, update the mind map too. Thus the backend processing and persistence of new information should happen rapidly.

Performance in terms of front-end should also be addressed. The system is a fully graphic website and thus should give the user the most fluid and smooth experience. To implement a fast interactive system technologies such as HTML canvas can be used to render the graph with smooth user interaction.

5.2.2 Usability

Usability is the second highest priority as the system is fully graphic in terms of what the end user will use. The system needs to have a very good User Experience (UX) and must be completely user friendly so it is not difficult to use.

5.2.3 Reliability

The system needs to be reliable, since it has to provide information in real time. This cannot happen if the system is constantly down. Thus hot swapping and hot reloading will be used to update the system without shutting down the service.

Reliability also refers to providing accuracy, and such needs to be implemented to provide the correct information to the right bubbles.

5.2.4 Scalability

This system needs to be highly scalable. The large number of users should not affect the system. Thus a queue messaging service (RabbitMQ) will be used to handle large number of requests. This will prevent backend processes responsible for data processing to be flooded with requests.

5.2.5 Flexibility

The system must be flexible in terms of plugability of polling services. A new polling service for a new PIM should be able to integrate with minor to no change of the rest of the system. The system also should be able to function with any combination of PIMs that a user selects.

5.2.6 Security

The system should be very secure since it is using private information that belongs solely to that user. Private information such as emails and photos

isn't something that users would like other people seeing. Thus a secure login process has to be implemented. Since Google and Facebook have both good log in systems, we will use that to log in to the system by using the OAUTH protocol.

5.2.7 Auditability/Monitorability

Aspect orientated programming will be used to maintain a log of the internal functioning of the system. Any requests made by users will thus also be recorded.

5.2.8 Integrability

The system should be developed in such a way that new polling services can easily integrate with the already implemented system.

5.3 Integration Requirements

1. HTTP:

This is the main protocol for all websites and will be the interface for the user to easily navigate the system.

2. IMAP:

In the case that a user uses an email service for which there is no API the system will make use of IMAP to send and receive emails.

3. REST:

REST will be implemented as the communication protocol between back-end services. It is simple to implement and integrates well with systems already operational. REST will be useful in this case since the system does not care from which poller information comes from.

4. TCP:

This protocol will be needed to establish network connections between the user's computers and the system servers. These data streams can then be exchanged between the connected machines. TCP will allow for error detection and rectification of data transmission. This will be done at a high level and would typically be handled by libraries or the operating system.

5. API:

Since the system will gather its information through the consented use of the users social media, a vital part of the system is to use the already well developed infrastructure created by PIMs and access the information through APIs or application programming interfaces. We will use APIs from Google (<https://developers.google.com/apis-explorer/#p/>) to access emails. Facebook's API (<https://developers.facebook.com/products>) for Facebook content.

5.4 Architecture Constraints

Architecture constraints include:

- **Programming language** We will be developing the system in Java using the Spring Boot Framework.
- **Operating system** The server will be run off a Linux (CentOS) machine.

Further there are no constraints and the architects have free roam to implement an architecture that suits their needs.

6 Architecture Design

6.1 Infrastructure

We will be using the micro services architecture. Figure 1 shows the design at the first level of granularity. The micro services approach will help us to develop different parts of the system concurrently with minimal dependency on other components because the individual services are loosely coupled. And secondly it will control and encapsulate the complexity of the system, thus a simpler development process. Testing each service is also easier which would result in a higher quality system.

The system is also well suited for this architecture since there are several modules that will have to interact with each other, but each serve a separate and specific purpose.

6.2 Services

6.2.1 Polling Service

Figure 2 shows the class diagram of the Gmail poller. A poller will either use long polling or polling with a sleep period afterwards (depending on what the API allows) to request new data from a specific PIM. Once it receives this information it will parse this information into a RawData object and add it into a queue which will be further processed by worker threads (see section 6.2.2).

6.2.2 Processing Service

Figure 3 shows the class diagram of the processing of raw data. Once raw data has been added to a queue by the poller in section 6.2.1, one of several worker threads will dequeue this raw data and pass the data to a Natural Language Processor (NLP). Once the NLP has finished processing the data the worker thread will parse this data into a ProcessedData object which will be sent to the database for persistence (see section 6.2.4).

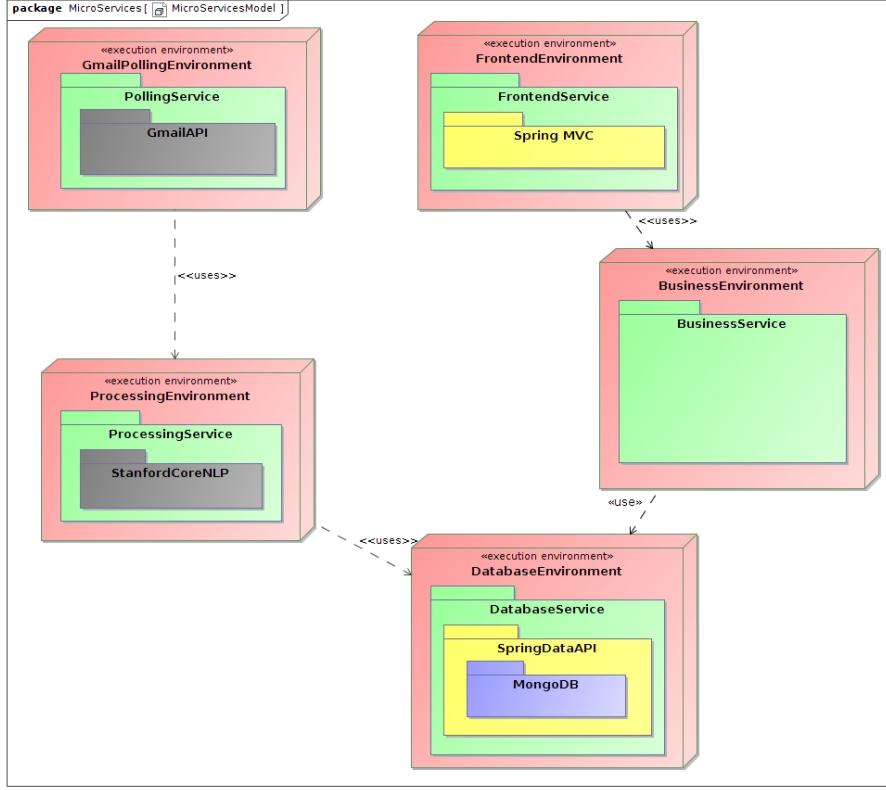


Figure 1: Micro Services System Architecture

6.2.3 Business Service

Figure 4 shows the class diagram of the business service. This service will start and stop polling services as users sign up/deactivate their accounts. Its purpose is basically to manage the other services based on user actions.

6.2.4 Database Service

Figure 5 shows the class diagram of the interface that will be used to communicate with the persistence API.

6.3 Tactics

6.3.1 Flexibility

- Hot reload and hot refresh.

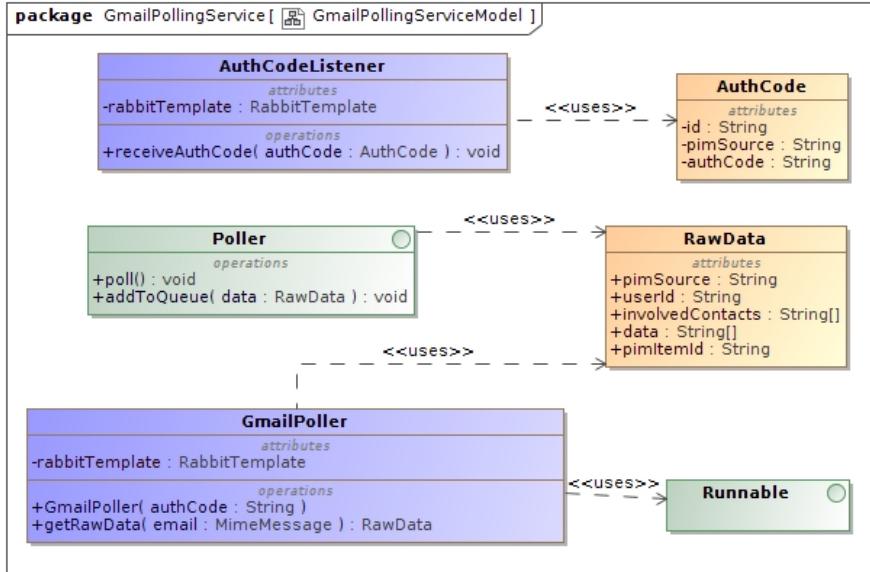


Figure 2: Gmail Polling service.

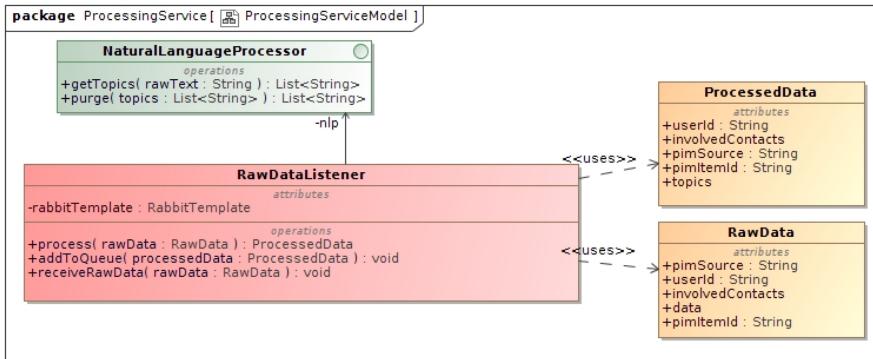


Figure 3: Processing service.

- Contract based programming, i.e. providing proper interfaces that allows future expansion.

6.3.2 Reliability

- Ability to roll back database changes should an error occur during the change.

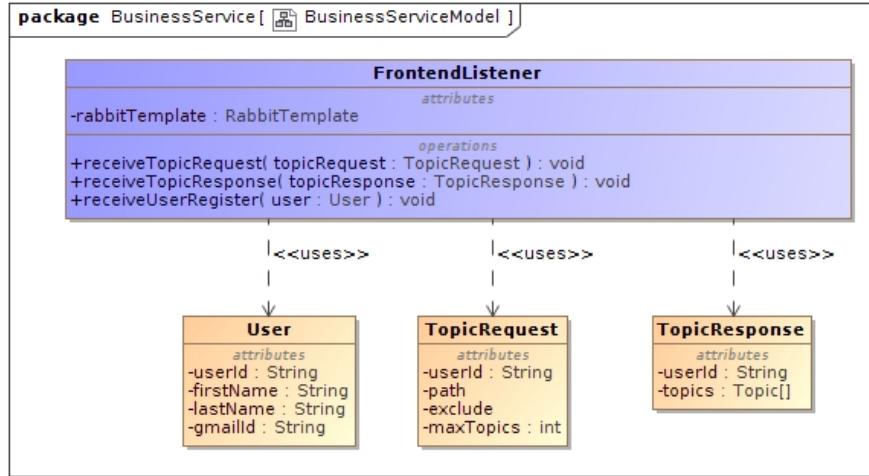


Figure 4: Business service.

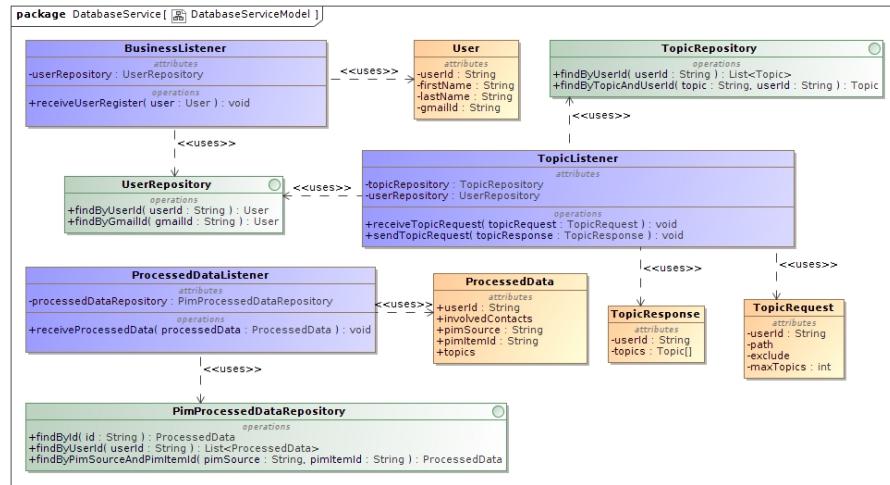


Figure 5: Database service.

- Ability to restart services that have crashed due to some error that occurred.

6.3.3 Security

- Authentication will be required for a user to use the system.

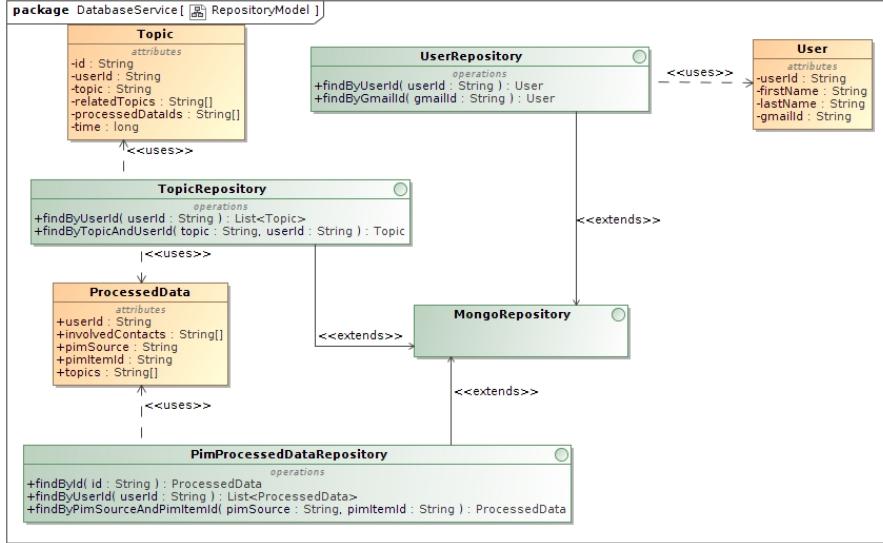


Figure 6: Repository Model.

- Data confidentiality will be achieved by using secure communication (HTTPS) and by applying salted hashing algorithms to keep passwords safe. Encryption will also be used on data that needs to be retrieved but has to be kept secure.

6.3.4 Auditability

Auditability will be achieved by logging all events that occur on the server. This will ensure first of all, accountability and secondly it can assist in fault and error tracking later on.

7 Database and Persistence

We plan on using MongoDB as our database. This choice was made on the basis that a NoSQL database's document structure would fit our application's needs better than a standard relational database. MongoDB is also the perfect tool to use when data mining and high write loads are brought into play.

8 Process Specification

In this section we will address the manner in which the system will be implemented on a very abstract and overview manner.

8.1 Server

In the back end, the system would have a polling service that polls the PIMs for new information about the users. If a polling thread receives new data from the platform, they add this new data to a queue to be processed. Several worker threads dequeue data from the queue and uses natural language processing to determine the topic(s) contained in the data. This data is then added to a database and the corresponding topic(s) as well. Note that since hashtags are used so frequently, any hashtag in the text should be added as a topic of its own. When a user requests their bubble map, the system responds with all relevant topics of their life based on frequency and recency. The topics thus have a weight related to them that is calculated by the frequency and the temporal component. The weight is used to decide which topics should be selected. These topics are then returned to the user in the form of JSON objects. Once a user expands one of the bubbles a request is sent to the server to retrieve all the relevant topic corresponding to the topic to the user. These new topics is then added to the bubble map in the form of bubbles. Relevant topics are found on the basis of your own events and the events of your friends.

8.2 Front End

The user starts out with a root bubble that has relevant topics connected to it including a people bubble. All these bubbles can be expanded and reveal information related to that topic. The "people" bubble shows contacts you recently communicated with or might want to communicate with now. Expanding one of these contacts will show the different platforms you can interact with this contact, including viewing a profile and sending a message/email. Other topics will expand into related topics and those can be expanded again and so forth. Upon expanding a topic a user can then interact with a specific occurrence in that topic (comment/reply) in a side panel that will be presented on the right of the screen.

9 Functional Requirements

The following sections contain a brief overview of some of the functionality that will be provided by our system. The sections will become more detailed as development progresses.

9.1 Use Case Prioritisation

Critical (See figure ??)

- Sign up
- Choose PIMs
- Login to PIMs

- Logout
- Expand bubble
- Display Data
- Remove bubble

Important

- Deactivate Account
- Reply to an email
- Gain access to Gmail via our application
- Set default depth factor
- Set default branch factor

Nice-to-Have

- Set quick access depth factor
- Set quick access branch factor
- Reset graph
- Customise colour scheme of interface
- Help centre

9.2 Use Case/Service Contracts

9.2.1 Sign up

Description: A user is required to sign up on our system. This entails using one's Google or Facebook account.

Prioritisation: Critical

Pre-conditions

- A user must have a Google or Facebook account.
- The user must enter the correct information in order for the validation to be successful.

Post-conditions

- The user will have access to the functionality provided by our system.
- The user may set their preferences as they wish.
- The user may add or remove PIMs.
- The user may log out of the system when they wish to.
- The user may deactivate/delete their account should they so wish.

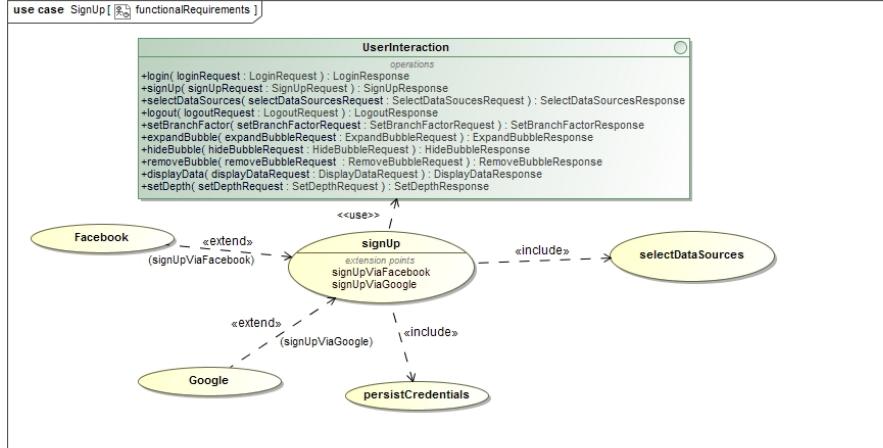


Figure 7: Use case diagram for Signing up

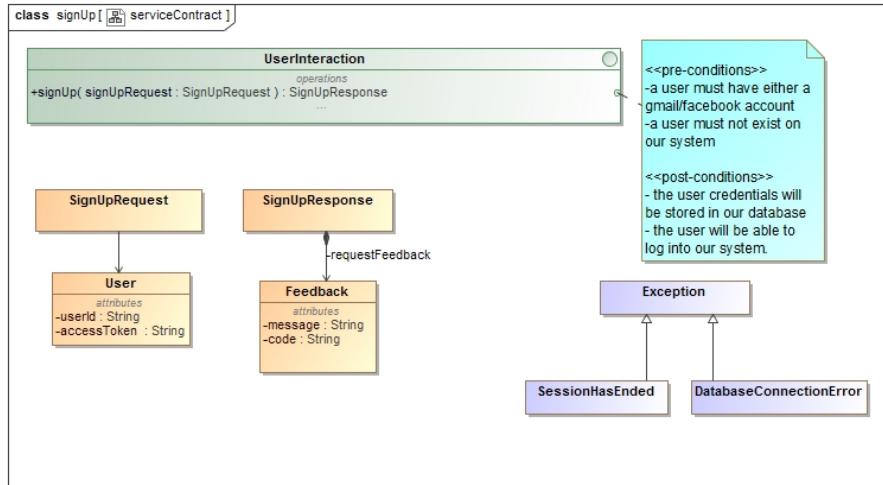


Figure 8: Service contract diagram for Signing up

9.2.2 Choose PIMs

Description: A user is required to select the various PIMs they wish to be used when extracting data to build their bubble map.

Prioritisation: Critical

Pre-conditions

- A user must be registered.

- A user must be logged in to the system.
- The user must have an active and valid account for that specific PIM.

Post-conditions

- The user may add or remove PIMs.
- The selected PIMs will be mined for data.
- The unselected PIMs will not be mined for data.

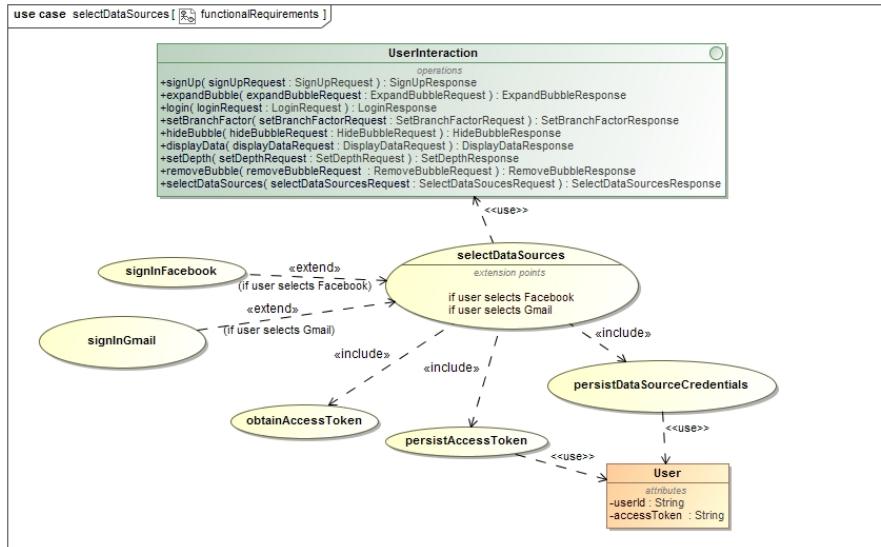


Figure 9: Use case diagram for selecting data sources

9.2.3 Login to PIMs

Description: A user will login with their Google account. This will be done by using the OAUTH protocol which is compatible with the Google and Facebook API.

Prioritisation: Critical

Pre-conditions

- A user must be logged in to the system.
- A user must have selected at least one PIM.
- A user must have an active and valid account for that PIM.

Post-conditions

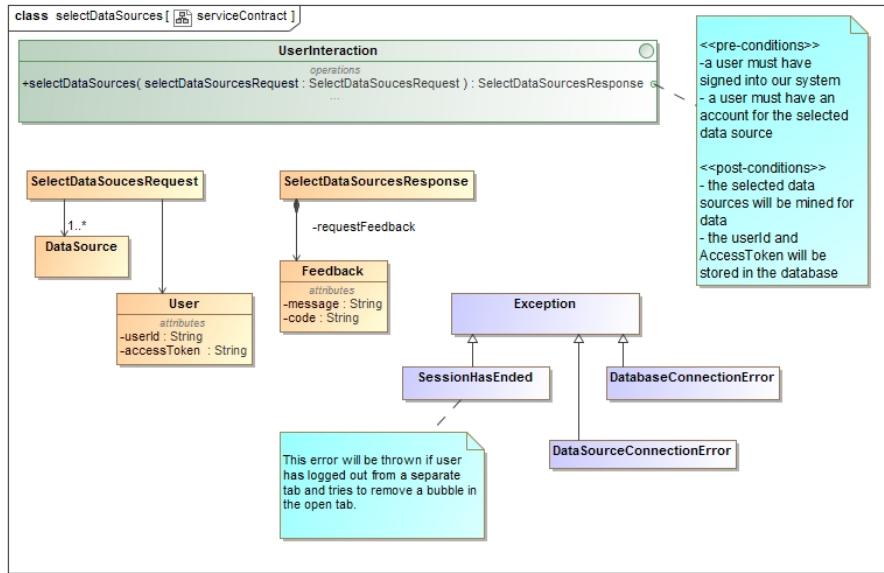


Figure 10: Service contract diagram for selecting data sources

- The PIMs will be mined for data.
- The login credentials for that PIM for a specific user will be stored in our database.

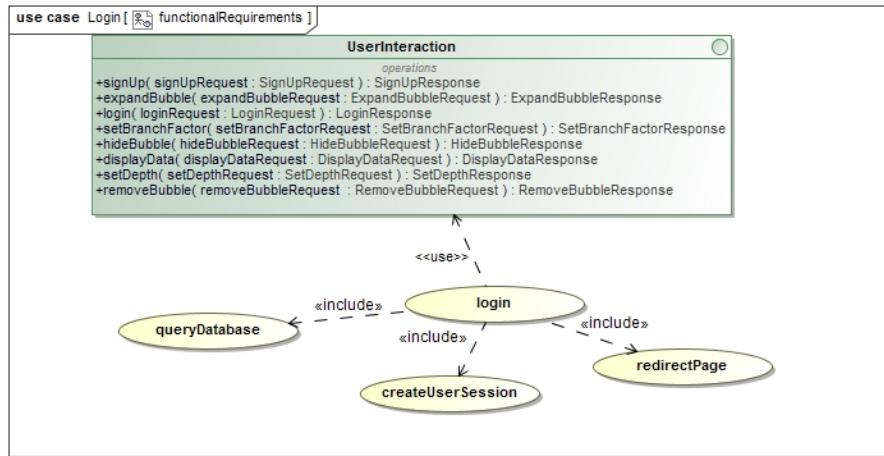


Figure 11: Use case diagram for Logging in

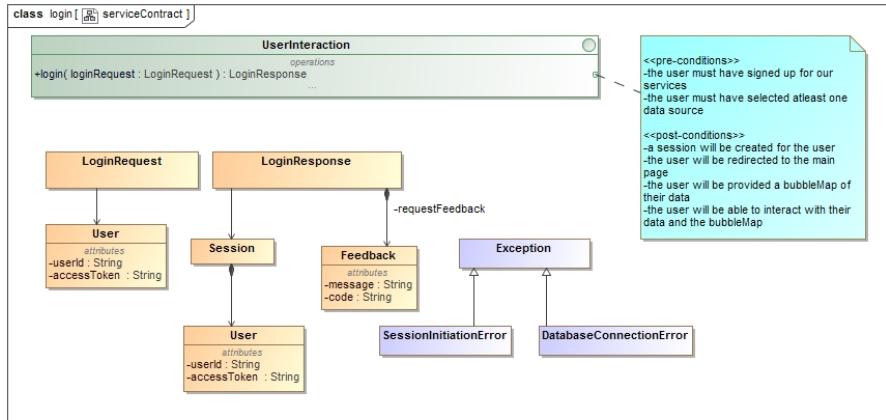


Figure 12: Service contract diagram for Logging in

9.2.4 Logout

Description: A user can choose to log out of the system in a secure manner.

Prioritisation: Critical

Pre-conditions

- A user must be registered on the system and logged in to the system.

Post-conditions

- The user will not have access to the main functionality provided by the system.
- The user will need to log in again to access their bubble map.

9.2.5 Expand Bubble

Description: A user can choose to expand a bubble to reveal more topics related to the bubble.

Prioritisation: Critical

Pre-conditions

- A user must be registered and logged in on the system.
- A user must have selected one or more PIMs to be mined.

Post-conditions

- The system will expand the bubble Map and introduce new topics.

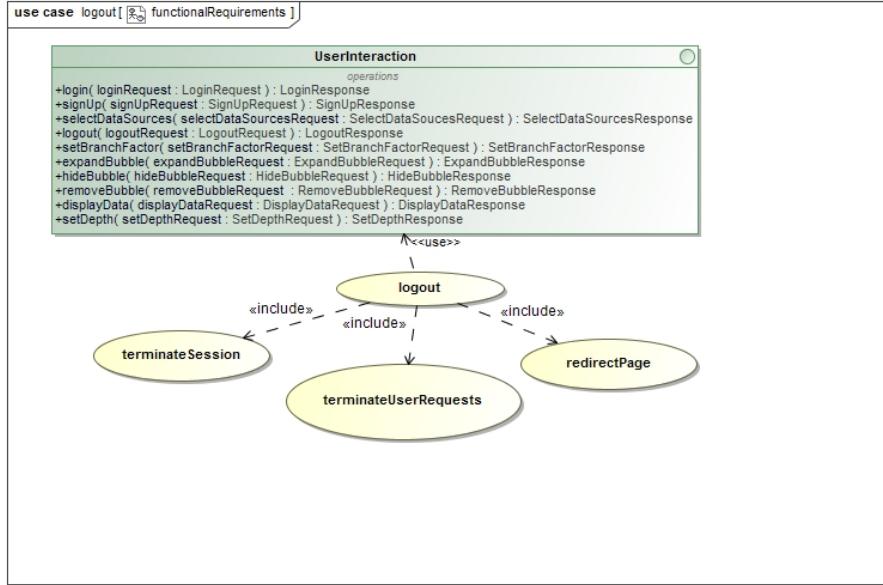


Figure 13: Use case diagram for Logging out

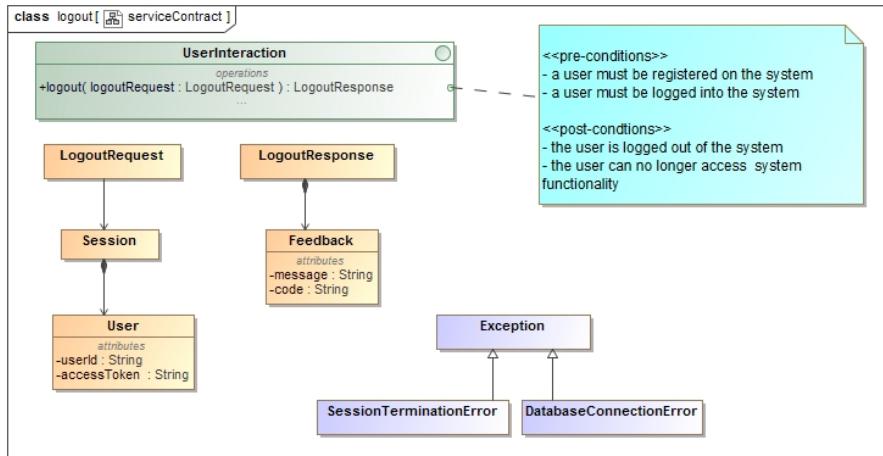


Figure 14: Service contract diagram for Logging out

9.2.6 Display Data

Description: A user can choose to view a bubble to reveal specific data and functionality surrounding a chosen bubble.

Prioritisation: Critical

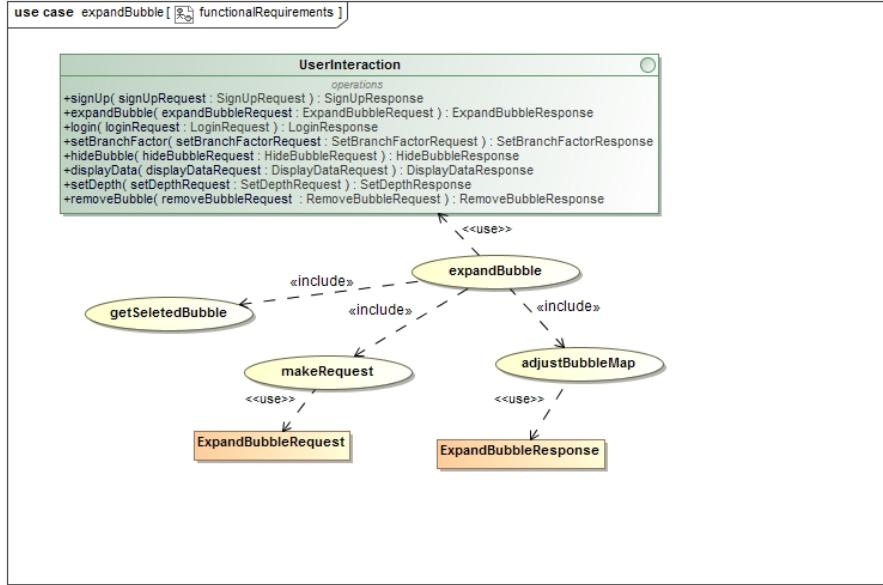


Figure 15: Use case diagram for Expanding a Bubble

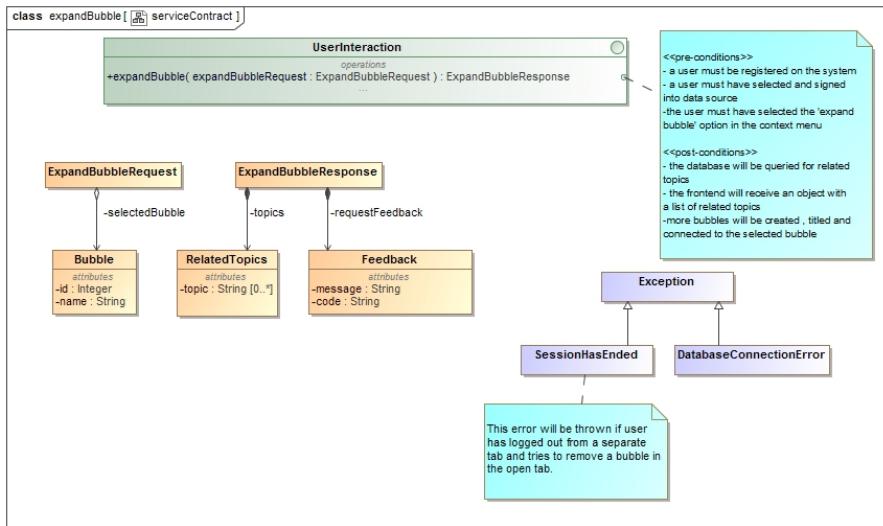


Figure 16: Service contract diagram for Expanding a Bubble

Pre-conditions

- A user must be registered and logged in on the system.

- A user must have selected PIMs to be mined.

Post-conditions

- The system will reveal specific data regarding this topic in a side panel.
- The system will provide certain functionality for certain types of data e.g. a Facebook post will have the functionality to be like, commented on or shared.

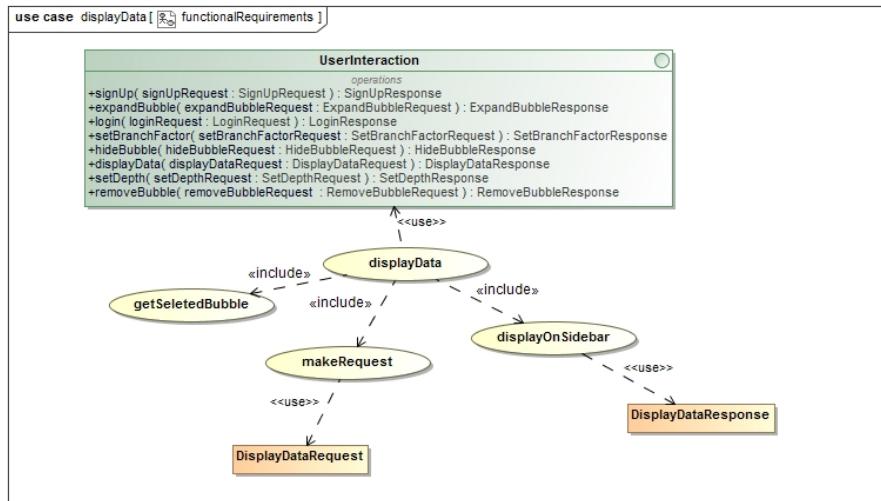


Figure 17: Use case diagram for Displaying data

9.2.7 Remove Bubble

Description: A user can choose to remove a bubble which will remove all bubbles related to it.

Prioritisation:Critical

Pre-conditions

- A user must be registered and logged in on the system.
- A user must have selected PIMs to be mined.

Post-conditions

- The system will remove the specified bubble.
- The system will remove all bubbles in the sub branch of the selected bubble.

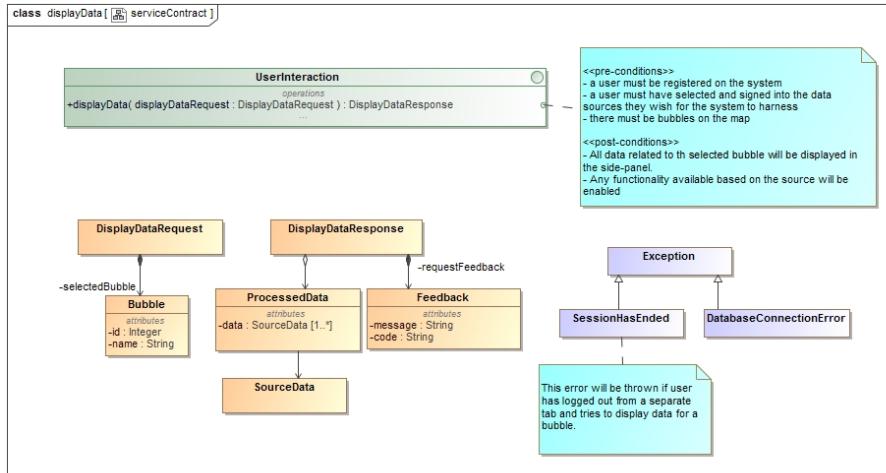


Figure 18: Service contract diagram for Displaying data

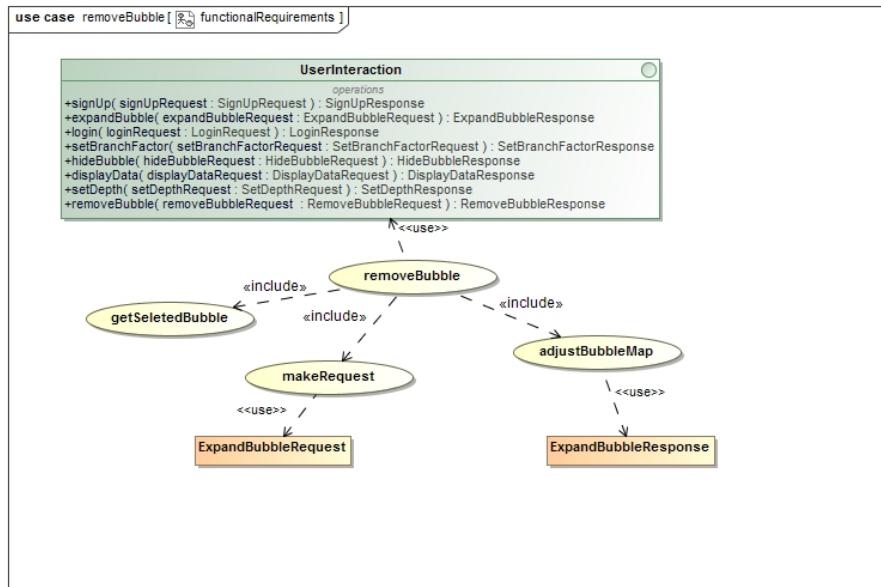


Figure 19: Use case diagram for removing a bubble

9.2.8 Deactivate Account

Description: A user can choose to permanently delete their account or deactivate it.

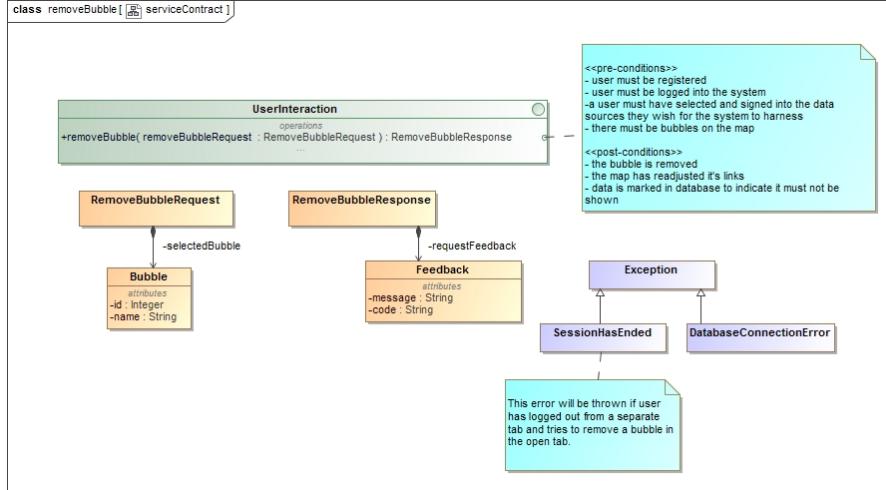


Figure 20: Service contract diagram for removing a bubble

Prioritisation: Important Pre-conditions

- A user must be registered on the system.

Post-conditions

- The user will not have access to the main functionality provided by the system.
- The user will need to sign up again to access the system functionality.

9.2.9 Specify the default depth factor

Description: A user can choose the number of levels the bubble map should expand when they open the map for the first time during the current session.

Prioritisation: Important Pre-conditions

- A user must be registered and logged in on the system.
- A user must have selected PIMs to be mined.

Post-conditions

- The user's bubble map will modify itself to adapt to the user's specifications.

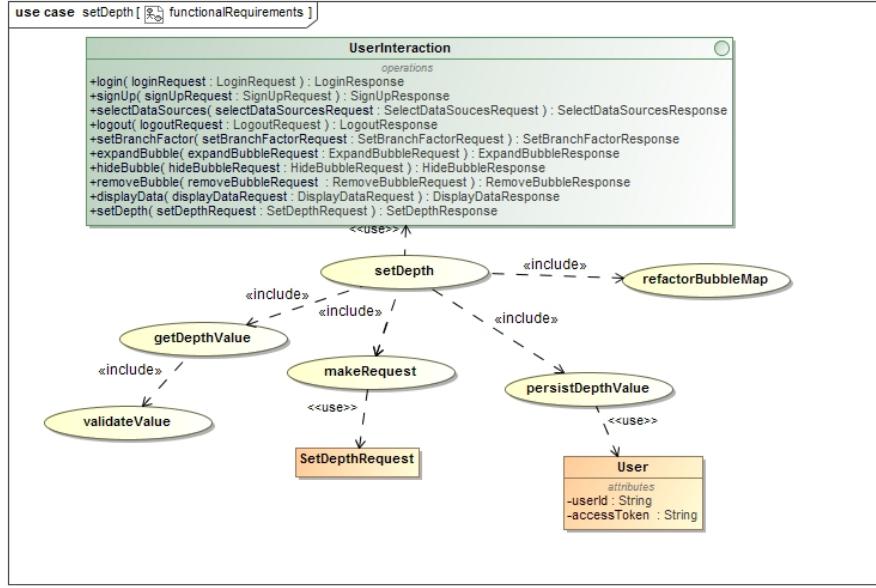


Figure 21: Use case diagram for setting default depth

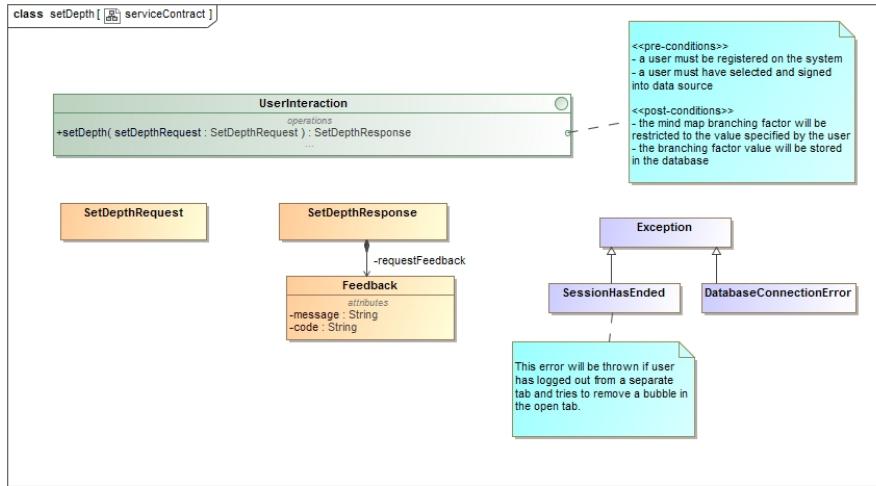


Figure 22: Service contract diagram for setting default depth

9.2.10 Specify the default branch factor

Description: A user can choose specify how many branches the bubble map will initially have.

Prioritisation: Important

Pre-conditions

- A user must be registered and logged in on the system.
- A user must have selected PIMs to be mined.

Post-conditions

- The user's bubble map will modify itself to adapt to the user's specifications.

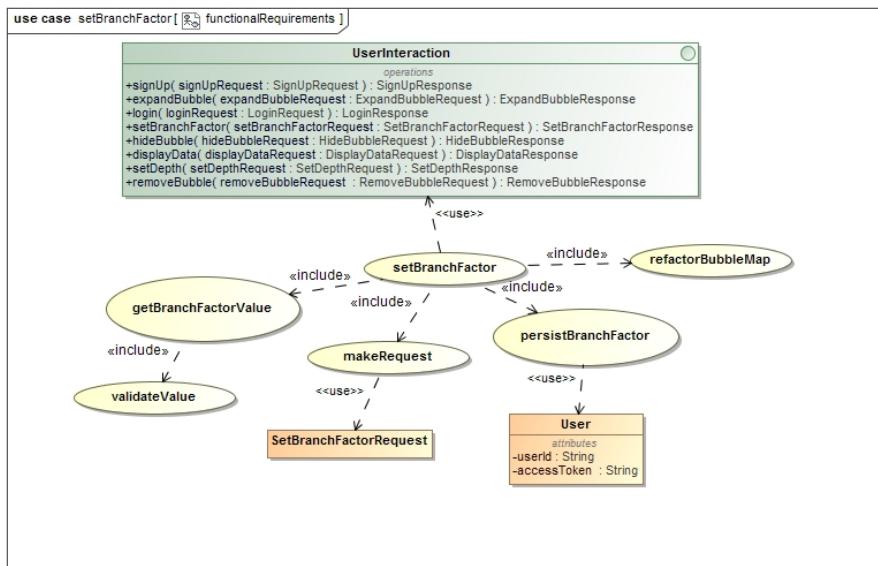


Figure 23: Use case diagram for setting default branch factor

9.2.11 Reply to an email

Description: A user can respond to an email directly from our application

Prioritisation: Important

Pre-conditions

- A user must be registered and logged in on the system.
- A user must have selected an email source.

Post-conditions

- The user will be able to reply to a message that has been displayed by our system

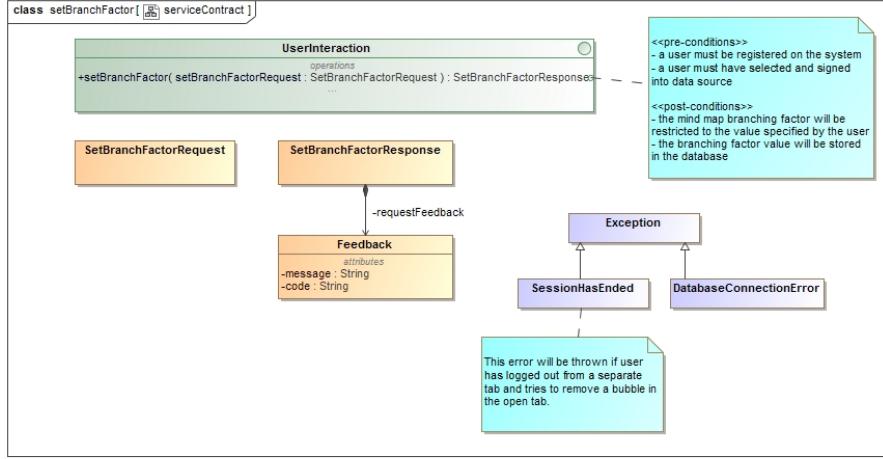


Figure 24: Service contract diagram for setting default branch factor

9.2.12 Redirect to Gmail application from our application

Description: A user can respond to an email directly from our application

Prioritisation: Important

Pre-conditions

- A user must be registered and logged in on the system.
- A user must have selected an email source.

Post-conditions

- The user will be able to navigate to Gmail from our application

9.2.13 Set quick access depth factor

Description: Manipulate the BubbleMap quickly but not affect its state permanently

Prioritisation: Nice-to-Have

Pre-conditions

- A user must be registered and logged in on the system.

Post-conditions

- The user will be able to change BubbleMap depth factor for instant manipulation

9.2.14 Set quick access branch factor

Description: Manipulate the BubbleMap quickly but not affect its state permanently

Prioritisation: Nice-to-Have

Pre-conditions

- A user must be registered and logged in on the system.

Post-conditions

- The user will be able to change BubbleMap branch factor for instant manipulation

9.2.15 Reset graph

Description: Reset the graph to the state it was in when it loaded that session

Prioritisation: Nice-to-Have

Pre-conditions

- A user must be registered and logged in on the system.

Post-conditions

- The user will be able to revert all changes to the BubbleMap settings

9.2.16 Customise colour scheme

Description: Change the colour scheme of the interface

Prioritisation: Nice-to-Have

Pre-conditions

- A user must be registered and logged in on the system.

Post-conditions

- The user will be able to change the BubbleMap background colour
- The user will be able to change the Navigation bar colour
- The user will be able to change the sidepanel colour

9.2.17 Help Centre

Description: A means of communication between users and the developers

Prioritisation: Nice-to-Have

Pre-conditions

- A user must be registered and logged in on the system.

Post-conditions

- The user will be able to contact the developers and report issues or ask for assistance if the Help/FAQ page did not suffice

9.3 Required Functionality

Functionality that should be included in the system includes:

- A login/sign up system.
- Adding various PIMs to your account.
- Interaction with PIMs directly from the bubble map itself (commenting, emailing, etc.).
- Expanding bubbles in order to find articles related to that bubble.
- Rapidly integrate new information into the system.
- Logs will be used to track any unhandled exceptions or problems.

9.4 Domain Model

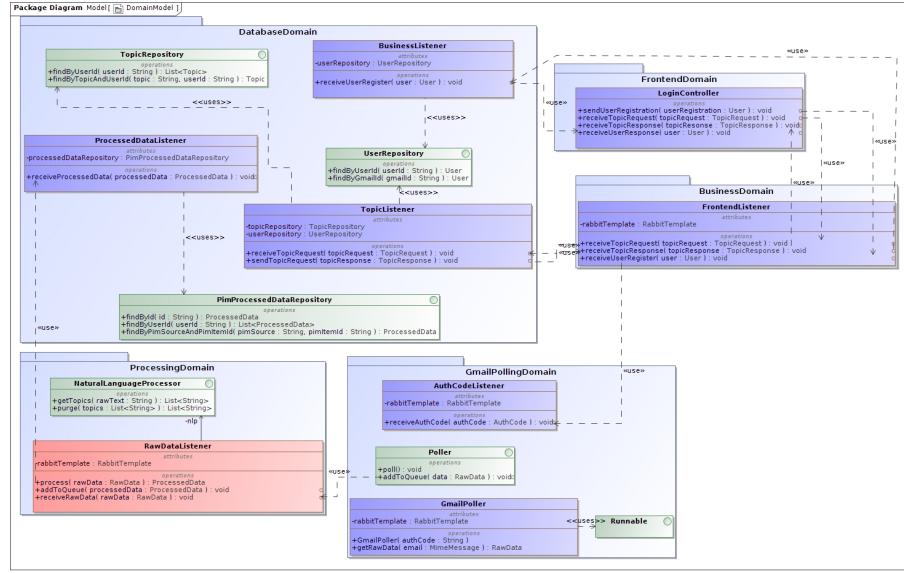


Figure 25: Class Diagram: Domain Model

The mind map for a user will portray information from PIMs requested by the user. The mind map will consist of a root node containing child nodes of relevant information (topics). Each topic will point to a PIM (Facebook, Google etc.) and will contain a list of information relative to the topic.

10 Technologies

10.1 Framework

The framework we decided to use is Spring Boot. This was chosen on the fact that Spring is a lot more lightweight than JavaEE. Spring also uses the convention-over-configuration idea that simplifies the learning experience.

Spring is a well known framework with a large community of support. It is highly scalable and works towards being stateless. Spring also integrates well with technologies like Tomcat and build tools like Gradle. Spring Boot was chosen over Spring MVC for its simplicity.

Research on Google's framework "Google Guice" was also done as an alternative, and while it is a good architecture in terms of what it does and how

it makes it simpler for the programmer by moving away from XML, it is still a relatively new framework and does not have much support yet. Spring has been tried and tested and the fault tolerance on Spring is the deciding factor over Google Guice.

10.2 Web Container

The web container we have chosen to use is Tomcat. Since, as mentioned before, we have chosen to use Spring as our framework Tomcat would be the obvious choice, given it has good integrability with Spring.

Jetty was also considered but due to the amount of documentation and experience that Tomcat has over Jetty, the choice was made to rather go with Tomcat. Benchmark tests were also run with Tomcat, Jetty, Glassfish and it was shown that Tomcat was the better contender.

We have decided to move away EJB containers and thus JavaEE containers to use the much lighter web container server in response to the dynamics of our particular system which required a fast, light footprint system to run our framework on.

10.3 Natural Language Processor

Due to the complexities involved in getting Google's Parsey McParseFace set up, the Stanford CoreNLP will be used to analyse the syntax and underlying topics of sentences.

10.4 Build Tool

Gradle was chosen as the build tool due to its simplicity and simplistic integrability with Spring. It also doesn't use the XML approach that Maven uses but rather uses a more natural programming style that is recognizable.

10.5 Communication

WebSockets will be used to have a persistent client to server communication that will create a low latency, real time interaction, a requirement for our system. Some of the advantages of WebSockets is its high scalability, high performance and protocol layering.

A hypothesis with WebSockets initially was that not all browsers could support this technology. This was however proven incorrect as tests ran on all browsers showed that every browser except Opera mini (A mobile client for Opera) supported WebSockets. <http://caniuse.com/#feat=websockets>

11 Initial Design

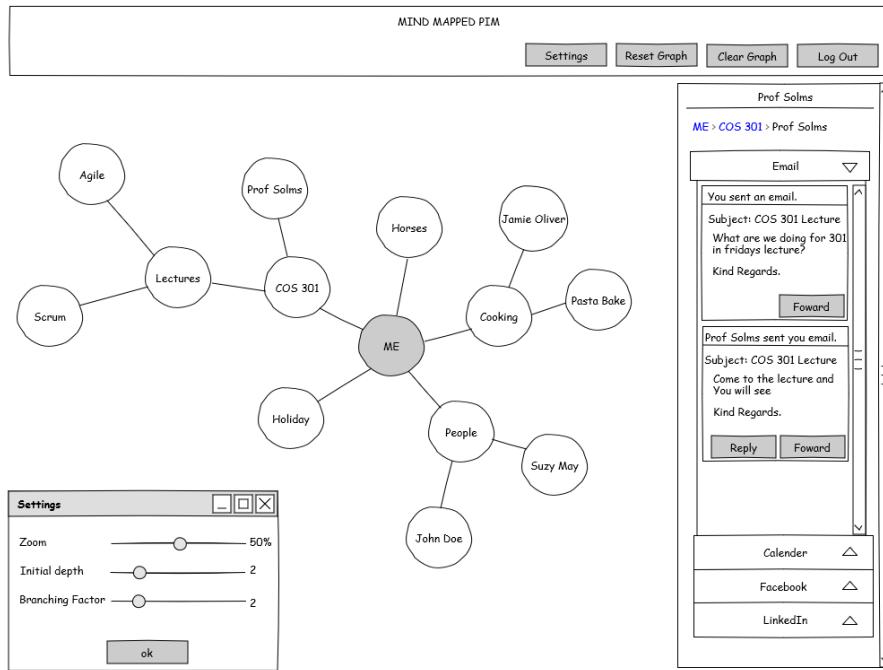


Figure 26: Main Interface- Email Expanded

An initial Idea for the main mind map interface, drawn with wire frames, with a sidebar that shown the currently active node "Prof Solms" with relevant information on why and what information is relevant. In this figure the Emails column is expanded, in the next figure the Calender will be expanded.

12 Open Issues

- What we are going to use for our physical server?
- How often should the mind map refresh?
- How much data mining should be done before the information is displayed and the mining starts being done behind the scenes?
- Should removing bubbles remove data from the database or not?
- Is a privacy policy required?

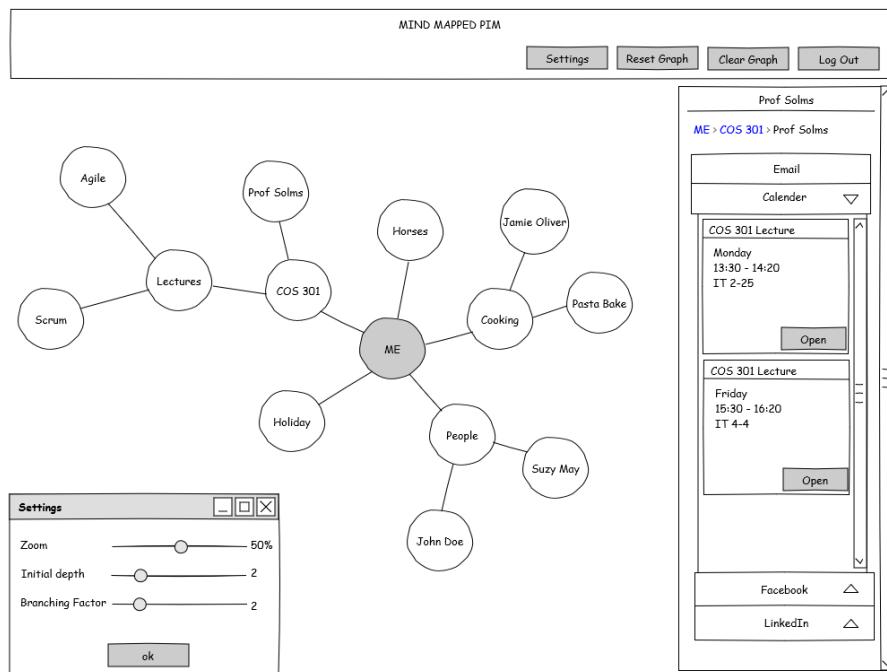


Figure 27: Main Interface- Calender Expanded