UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Software Requirements Specification and Technology Neutral Process Design Mindmap PIM

Client: IMINISYS

Team: A-Cube-N

Dunkley, Nathan 14145759
Grobler, Arno 14011396
Lochner, Amy 14038600
Maree, Armand 12017800

Department of Computer Science, University of Pretoria

Contents

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 Personal Information Manager (PIM) platform with which they can interact to make use of the functionality from other platforms. A PIM as defined by *TechTerms* is "a software application that serves as a planner, notebook, and address book all in one. It can also include things like a calculator, clock , and photo album."

2 Vision

The vision of this project is to create a PIM which extracts data from various existing platforms such as Facebook, Gmail, Google Calendar, etc. The application will make use of various means to extract data, determine the general topic of the data then integrate this new information into their 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 specifically. 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 this project was elected for us to do, 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 current tasks that need to be completed. This

information will come from different sources and enables the user to see all relevant information without having to go to each source individually.

4 Important Terminology

- **MyBubbles** A potential 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** This term will be used interchangeable with PIM. It refers to the source where user data is retrieved from.

5 Architecture Requirements

5.1 Access Channel Requirements

5.1.1 Human Access Channels

Users will access the "Mind Mapped PIM" 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. When the user is on the website, they will have to log in, the option given to log in with their Facebook or Google accounts. 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

RESTful web services will as the system access channel for its lightweight, maintainability, and scalability. It will use URI's to allow easy access from the client side web service to server side.

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.

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 SVG or 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. The end goal is to let the system give benefit to it being graphic so the user can interpolate the data, and not make it more difficult.

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 or can only cater for a small amount of users before it starts slowing down. When, for example, a user clicks to expand a bubble the result should be almost instant and not break the graph or slow down the whole system because it is retrieving a few bubbles.

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

Implementation of reliability should be done in two parts, namely first prevention of faults, and secondly detection of faults.

5.2.4 Scalability

This system needs to be highly scalable both in terms in the non restriction on the amount of users that can use the system simultaneously and the amount of data stored of each user that needs to be both stored and displayed without hindering the rest of the system.

The amount of users using the system could be large at any time, and the use of the system by one user should not impact the use of the system by another user. The actual mind map should also not become slower or unresponsive if the amount of bubbles grow, which it will.

Polling of the respective social media and email should also be considered. The increase in the amount of threads used for this operation should not make the system unresponsive.

5.2.5 Flexibility

The system must be flexible in terms of plugablity of subsystems that are used namely our social media and emails. Since all of these system are different and use different API's and architectures. The system should be able to handle this, and any future systems that need to be added.

Some users will not have a Facebook account, for example, and thus the system should still be functional, irrespective if the user has a certain social media.

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 log in 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.

If a user prefers to not have certain features of social media, such as only look at posts and photos on Facebook but not post on their behalf, then the system should cater for that.

5.2.7 Auditability/Monitorability

Logs of all actions and settings of the user should be maintained. When the user logs back in at a different time, the graph that was displayed before should be displayed again and updated with new things that had happened while the user was offline. Settings should remain the same between sessions.

5.2.8 Integrability

The system should be integrable in terms of how well it integrates with other systems like Facebook and Google. This information should be easy to access and does not affect the functioning of the overall 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. **SMTP and IMAP:**
   Sometimes a user does not want to connect their PIMs for safety concerns and so the system cannot use the APIs required and thus will need to manually get the information by making use of IMAP to retrieve and sort though the users mailbox and SMPT to still have the option for the system to send emails on behalf of the user. This allows the system to still have functionality without any specified APIs for those users who do not wish to share that information or who do not own social media.

3. **REST:**
   As mentioned before, REST will be implemented. It is simple to implement and integrates well with systems already operational. REST integrates well with HTTP and follows a client-server model which is something our system utilizes. REST is vital to the use of APIs to gain access to information of social media of the user.

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 contacts, email, calender, hangouts/sms, Facebooks API (`https://developers.facebook.com/products`) for Facebook content and LinkedIn API(`https://developer.linkedin.com/docs/rest-api`) for LinkedIn related information.

5.4 Architecture Constraints

Architecture constraints include:

- **Programming language** We will be developing the system in Java.
- **Operating system** The server will be run off a Linux 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 with 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.

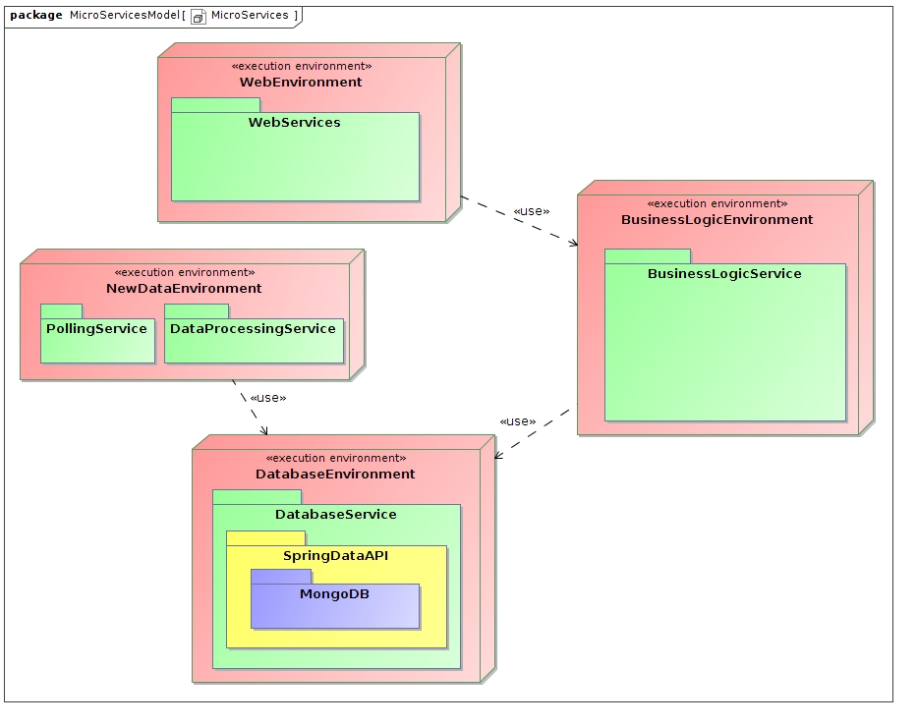

Figure 1: Layered System Architecture

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 Web Services

6.2.2 Data Polling Service

Figure 2 shows the class diagram of the pollers. A specific poller will use long polling 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 6.2.3).

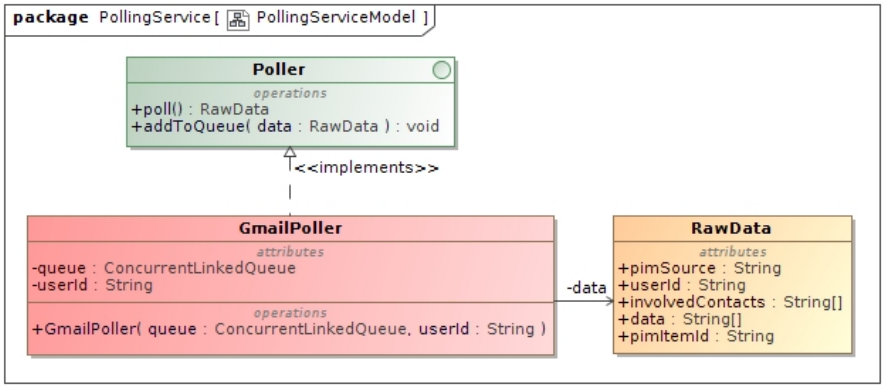

Figure 2: Data polling service.

6.2.3 Data Processing Service

Figure 3 shows the class diagram of the processing of new data. Once raw data has been added to a queue by the poller in section 6.2.2, 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 6.2.5).

6.2.4 Business Logic Service

6.2.5 Database Service

Figure 4 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 deployment

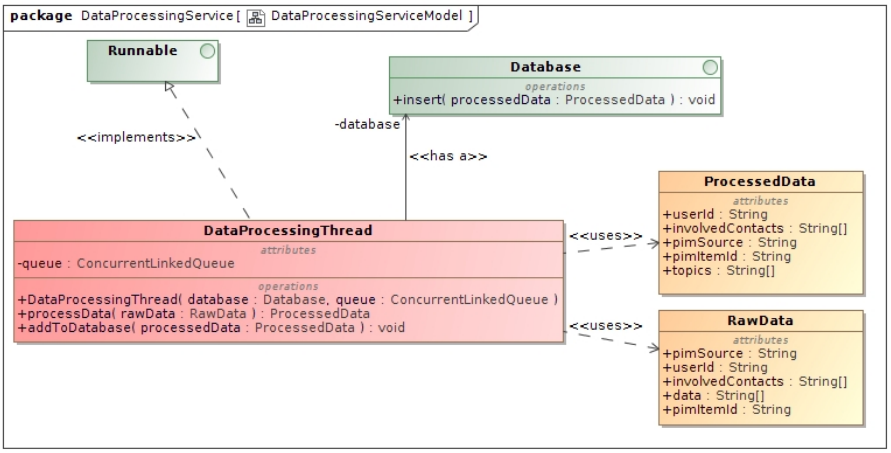

Figure 3: Data processing service.

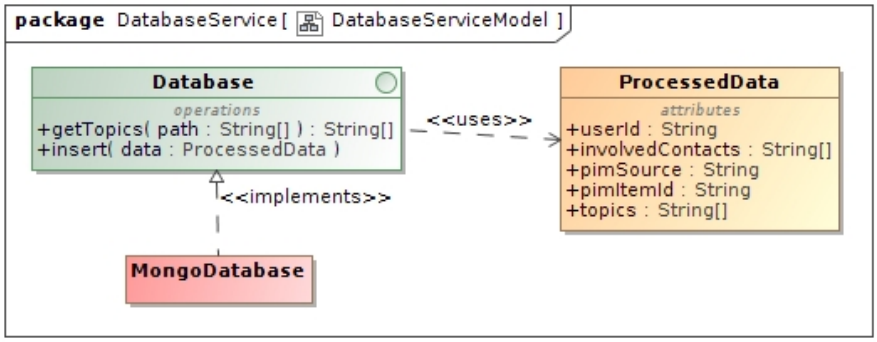

Figure 4: Database service.

- 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.
- 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.
- 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 graph 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 bought 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) of 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 temperal 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 5)

- Sign up
- Choose PIMs
- Login to PIMs
- Logout
- Delete Account

Important (See figure 6)

- Expand bubble
- View bubble
- Hide bubble
- Minimize bubble

Nice-to-Have (See figure 7)

- Minimize bubble
- Customize layout
- Specify the initial depth
- Specify the branch factor
- Specify filter

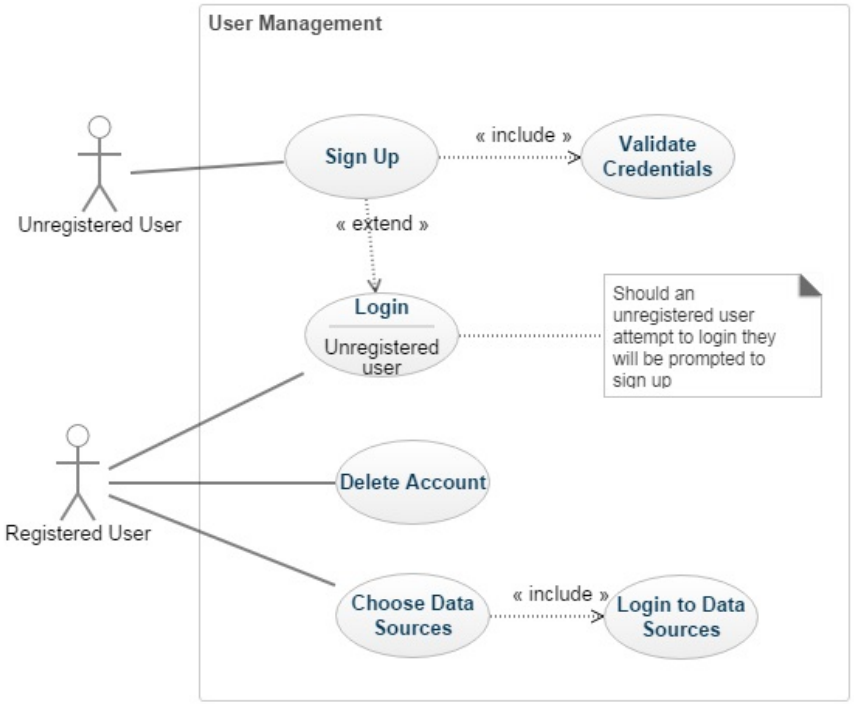

Figure 5: Use Case: User Management

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.

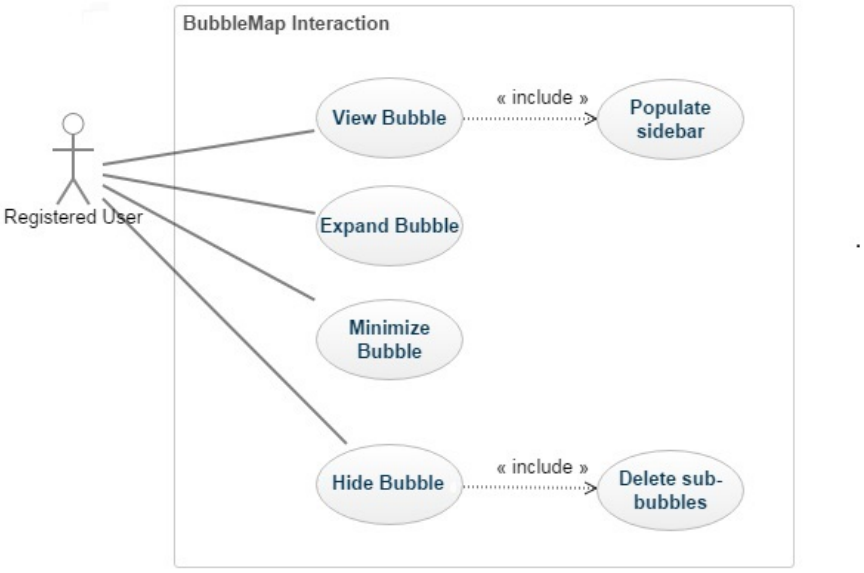

Figure 6: Use Case: Bubble Map Interaction Functionality

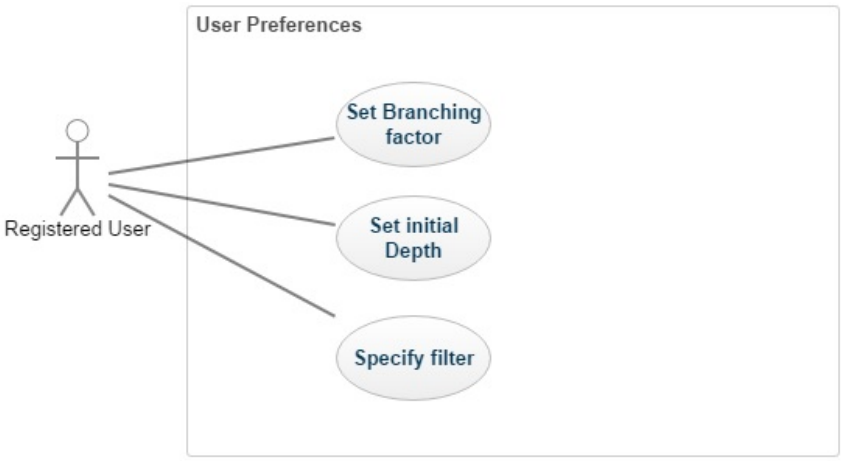

Figure 7: Use Case: User Preferences

- The user may log out of the system when they wish to.

- The user may deactivate/delete their account should they so wish.

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.

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

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

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 Delete Account

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

9.2.6 Expand Bubble

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

9.2.7 View Bubble

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

9.2.8 Hide Bubble

Description: A user can choose to hide a bubble which will hide all bubbles related to it.
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 system will hide the specified bubble.
- The system will hide all bubbles in the sub branch of the selected bubble.

9.2.9 Minimize Bubble

Description: A user can choose to minimize a bubble which will shrink the bubble map and essentially hide all sub-bubbles.
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 system will hide the sub-bubbles of a selected bubble.
- The bubble will provide the functionality to be expanded again.

9.2.10 Specify the initial depth

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: Nice-to-Have
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.

9.2.11 Specify the branch factor

Description: A user can choose specify how many branches the bubble map will initially have.
Prioritisation: Nice-to-Have
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.

9.2.12 Specify a filter

Description: A user can choose specify a filter to use on the bubble map.
Prioritisation: Nice-to-Have
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.

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

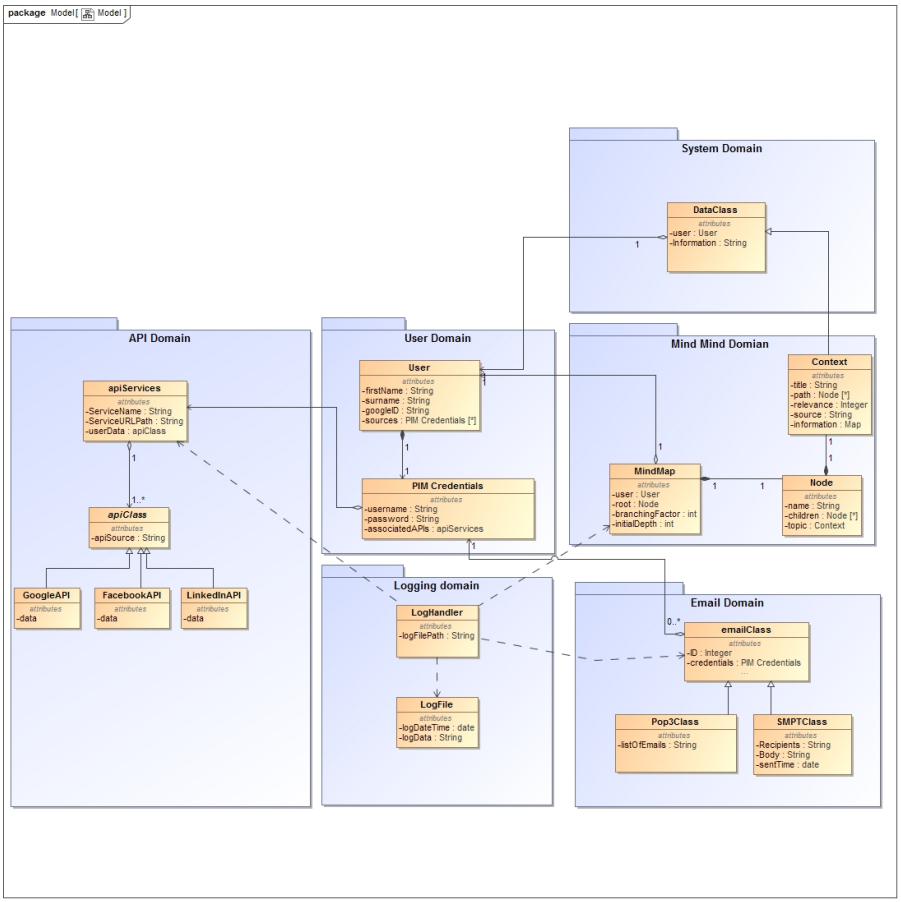

Figure 8: Class Diagram: Domain Model

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

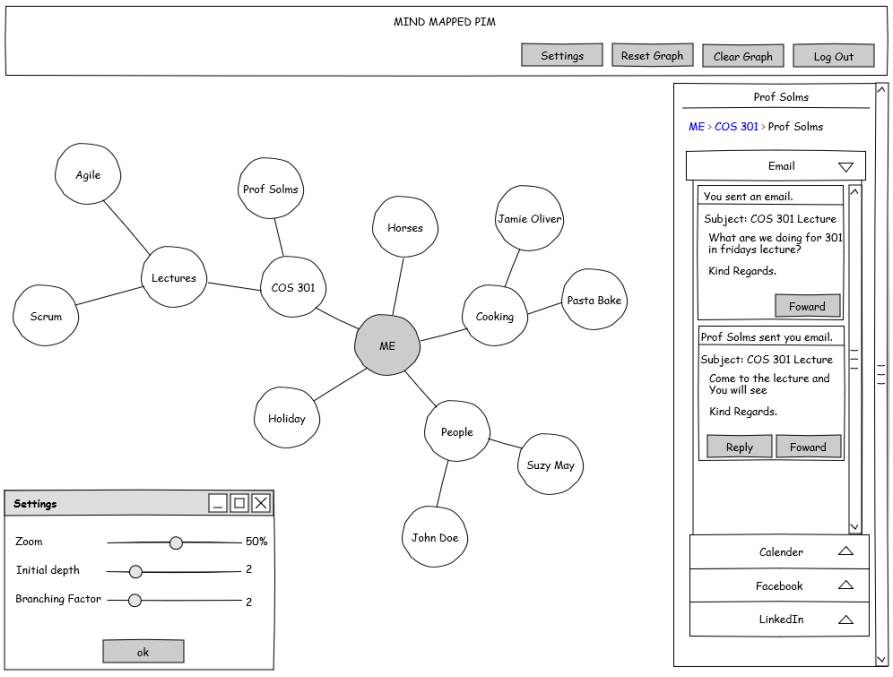

Figure 9: Main Interface- Email Expanded

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.

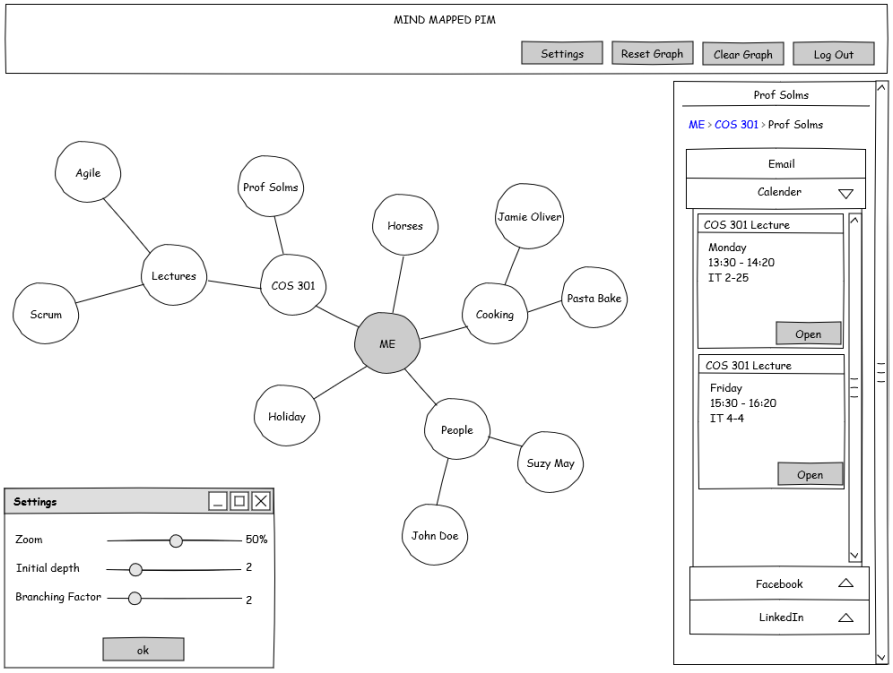

Figure 10: Main Interface- Calender Expanded

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?