Vrije Universiteit Brussel

Faculty of Science
Department of Computer Science
and Applied Computer Science

Web Engineering: Date4Life

Web Information System 2014-2015

Lars Van Holsbeeke (100755, lavholsb@vub.ac.be, WISE)
Tim Witters (93716, twitters@vub.ac.be, WISE)

Professor: Sven Casteleyn
Assistant: Pejman Sajjadi

June 2015

Contents

1 Introduction

This document is the deliverable concerning the third assignment of the course Web Engineering. It gives the reader a complete overview of the work accomplished and how the Date4Life application is internally organized and implemented. The mission statement of the website and the audience modeling of the users with their functional requirements will be given first.

Later the layout of all different conceptual designs including the CTT's, Domain Model and IFML models will be treated. A short reflection on certain choices we made and some problems we faced together with a conclusion finalizes this report.

An online version of the application can be found at the following address ():

`http://date4life.timwitters.me/Date4Life`

An available test user is available with the following credentials: **Ted** (Username) / **test** (password).

2 Mission Statement

We want to become a well-known, trustworthy website on which anyone can find the love of his/hers life in a comfortable atmosphere. Everything in the system should be specifically fine tuned to promote new relations in all possible ways. This will be achieved by providing the (registered) user:

- A friendly to use interface allowing him/her to make a personal profile and providing him/her solutions to execute advanced searches for other users. Such an advanced search should be able to find potential dates based on (but not limited to): marital status, amount of children, eye color, hair style, clothing style, nationality, occupation, language knowledge, degrees, favorite food/drink/-book/movie, etc.
- A user should also be able to spread out a message to the public by posting public messages on his/hers own wall and also be able to reply on them.
- When two particular potential 'lovers' found each other, they should be able to communicate in a private, discrete manner. This should be accomplished using an in-website messaging system (analog to private messaging on (for example) Facebook). As such kind of messages are usually the seeds of a new relationship, reading and replying should be as easy as pie.
- If a user is eager to get in touch with another user in the system, but is too feared to send a private message, he/she can get attention from his/her (hopefully) future beloved in a more discrete way by liking him/her using a single button on his/her profile. From then on, the user is notified of any profile updates of his/hers beloved. The beloved one will also receive a notification of the 'like' event. He/She can then do two things: contact the liker or remove him/her from his/hers personal 'liked users' list.
- When our lovers eventually come to 'a point of no return', they can decide to offer each other a range of gifts. The system should, for example, provide the user an interface to: send his/hers beloved some flowers, a bottle of wine, a kiss, his/hers heartbeat, ...
- If a user does not know where to start looking for a potential partner, he/she should be able to let the system suggest other users, based only on gender and age.
- Being popular can also have its disadvantages such as for example annoying users, spam users, stalking users, ... To prevent this kind of practices, each user should be able to block and report any other user.

To increase network effects of the system and attract more users, non-registered users should also be able to search user in the system without having to pass the registration procedure. However, to trigger them to make an account, search possibilities as well as results should be limited.

3 Audience Modeling

3.1 Audience Class hierarchy and Characterization

The goal of this application is to provide a useful platform for people, to interact with each other using different means of communication. To reach this goal the system recognizes three different types of people. We separate them into the following types: Non-registered Guests, Registered Users and Administrators. We explain each class in the next section, including the user characterization.

3.1.1 Non-Registered Guest (G)

Users who are visiting the website, without being logged in, should be offered a limited amount of functionality like explained under the functional requirements 3.2. These tasks mostly consist of simple viewing of some of the basic information of already registered users in the system.

It is important to remember to provided a unambiguous way to register for a new account, making the sign-up process as easy as possible. The user characterization for this user is as follows.

Type User	Visitor
Knowledge	Low
Use Frequency	Low
User Skills	Unknown
Motivation	Possible motivations but not limited to: • Registration • Searching for users • Loging in

3.1.2 Regular Registered User (R)

Once a user is registered he will belong to this group of the regular members. This user is the main type of our system and should have access to most of the functionality. Therefore we should focus our efforts on providing a great user experience for this user group.

As this will be a big group in our application, following characteristics can be expected:

Type User	Standard Member
Knowledge	Medium
Use Frequency	High
User Skills	Average
Motivation	Possible motivations but not limited to: • Finding people sharing the same interest. • Chatting with friends on the platform. • Update personal and edit profile information.

3.1.3 Administrator (A)

Certain user profiles can be upgraded to an Administrator. These have the ability to manage all the information of any other user in the system. They will manage all disputes on the site and have full access to every account.

Type User	Administrator
Knowledge	High
Use Frequency	Medium
User Skills	High
Motivation	Includes but not limited to: • Settle disputes between users • (Un)Block users for improper conduct • Edit user information

3.2 Information and functional requirements

In this section we will describe all the functional requirements that will be implemented by our application. Each of them will be targeted at a specific type of user of the platform.

User Registration

ID	FR1
User	G
Description	When a user is creating an account, he should have the possibility of entering following profile information: nickname, date of birth, profile picture, location, email address, interest (love, friendship, pen pal, activities), free description

Basic Search

ID	FR
User	G
Description	Unregistered users should be able to search for users based on age, range, location and gender. The result of this search will be a list of 10 users matching this criteria. Only nickname, age, location, picture and gender should be shown in the results.

Login Tracking

ID	FR3
User	G
Description	The nickname, age, location, picture and gender should be displayed of the 10 users last logged in into the application. This should be available to all non logged-in users.

Extended Profile

ID	FR4
User	R
Description	A registered user should get the option to extend his profile information with the following items: marital status, amount of children, eye color, hair style, clothing style, nationality, knowledge of language, occupation, degrees. Other 'random' fields should also be supported.

Random Favorites

ID	FR5
User	R
Description	A user should be able to add different 'favorites' belonging to self defined categories. For example: Movies : Lord of the Rings. Categories can be selected or created at will by the user.

Edit/Delete Profile information

ID	FR6
User	R
Description	Users should be able to edit/delete their profile information.

Search Users

ID	FR7
User	R
Description	Registered users should be able to find other users based on the profile information of those users in the application.

Viewing Profile

ID	FR8
User	R
Description	Any user should be able to see the profile of an other user. Everything but the email address should be shown.

User Wall

ID	FR
User	R
Description	Every user should have a wall where other users can post a public message on. If the target user blocked the posting user that last one won't be allowed to post his message to the wall.

Wall Reactions

ID	FR9
User	R
Description	Registered users should be able to react on wall postings by other users. Reactions should be grouped together in the interface.

Wall Management

ID	FR10
User	R
Description	Registered users should be able to delete all wall posts and all reactions to those post. This shouldn't be possible on other users walls, only the wall belonging to the user.

Messaging System

ID	FR11
User	R
Description	Users need the ability of using the internal messaging system. This system allows user to communicate in private between each other. They need the ability to read messages targeted to them, and have the ability to reply to these messages.

Likes List

ID	FR12
User	R
Description	Every user needs a liked list of all the liked user. The users needs to be able to add and remove users from this list.

Notifications from Liked List

ID	FR13
User	R
Description	When a users on once liked list has some profile update, the current user should be notified by this action. This allows them to follow the activity of other users.

Attentions

ID	FR14
User	R
Description	Attentions can be send between different users. Examples of these attentions could be Flowers, Hugs, Smileys... Users should have the ability to return the attention and be informed by attentions they receive.

Random Messages

ID	FR15
User	R
Description	A feature should exist where users can ask for a certain number of random selected users to chat with. They can limit the search based on gender and age.

Block User

ID	FR16
User	R
Description	One should be able to block other users if they want to. If a user is blocked he can't interact with the other user. Any messages/attentions/wall posts will be blocked from getting send to the user.

Reports Users

ID	FR17
User	R
Description	Anyone can report a user. Reporting a user will notify the administrators of the website about the incident. When reporting a user one should also get the ability to include a certain message which will inform the admin about the reason for the reporting.

Login / Logout

ID	FR18
User	R
Description	A user needs to able to login and logout in the application

Terminate Account

ID	FR19
User	R & A
Description	Every user can terminate his own account, disabling him to login in the future. All information regarding the user will be deleted from the server. An administrator can terminate the account of every user he wants, where a normal user can do this only for his own account

Profile Discovery

ID	FR20
User	A
Description	Administrators have the capability to view the profile of every user. All the user information is shown to the administrator.

Edit Profile

ID	FR21
User	A
Description	Administrators have the capability to edit the information of any user using the system.

Blocks and Reports

ID	FR22
User	A
Description	When a user gets blocked or reported, an administrator gets notified of this event. He then has the ability to act on this notification. The amount of reports and blocks of every user is also displayed to the admin.

Admin Messaging

ID	FR23
User	A
Description	The admin may message any user, ignoring any blocks or limits that are imposed for regular users.

User Access

ID	FR24
User	A
Description	Administrators have the option to disable user accounts for a certain period of time. This can be done for every user in the system. Disabled user won't be able to log-in into their account till the expiration date has passed.

3.3 Navigation and Usability Requirements

User experience is very important as this creates more confidence between the platform and its users. The navigational and usability requirements helps guarantee a minimal verifiable level of site design and usability.

Fast Account Creation

Description	A user should be able to create an account within a certain limited time-frame without the need of entering to much detailed information at the start of the registration.
User Class	Unregistered Users
Measuring Concept	Amount of seconds
Goal	sub 10 seconds

Menu Items

Description	There should be an easily understandable structure in the menu items. Possible confusion should be avoided for new users
User Class	All Users
Measuring Concept	Peer Review
Goal	/

Feedback

Description	When some user action has a different behavior compared to the default behavior expected, the user should receive some kind of feedback from the system. For example, when a user account is blocked by the admins, the user should be informed when his log-in fails
User Class	All Users
Measuring Concept	Verification
Goal	/

Style Guide

Description	To enable a homogeneous interface on every page for every user. The application should use a common style guide.
User Class	All Users
Measuring Concept	Verification

Flexibility

Description	The system should be flexible regarding user information. A user should not be obliged to fill in (certain) fields if he does not want to.
User Class	Registered Users
Measuring Concept	Verification

Updates

Description	Every user should be informed about actions on the platform concerning him. This to help users discover any activity that happened when they were not online.
User Class	All Users
Measuring Concept	Verification

4 Conceptual Design

4.1 Task Modeling

Following Concurrent Task Tree models will give the reader an overview on what activities a visitor to the website has to perform. As it would be too complex too treat the whole tree at once, it has been split up in multiple parts which will be treated separately. Most of the CTTs will be trivial to a reader having already experience with their notion. However, to eliminate any confusion, a small explanation is given for each of them.

4.1.1 Visit application

Figure 1 represents the root CTT of the application. When a user visits the website, he can choose to use the functionality available for any user or he can make an account/login to use the extended functionality for registered users.

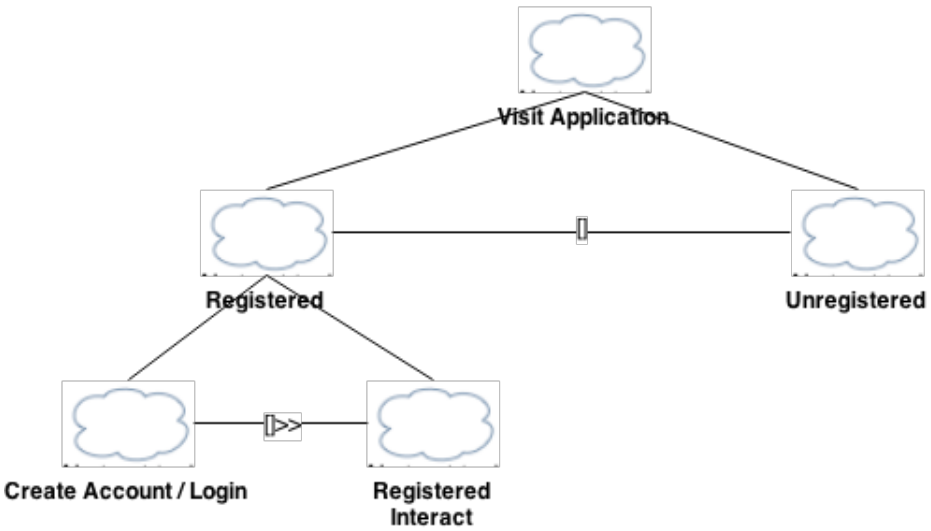


Figure 1: Root CTT: Visit Application

4.1.2 Unregistered

Figure 2 depicts the functionality for unregistered users. They can use the applications' search functionality with only a limited number of arguments and results.

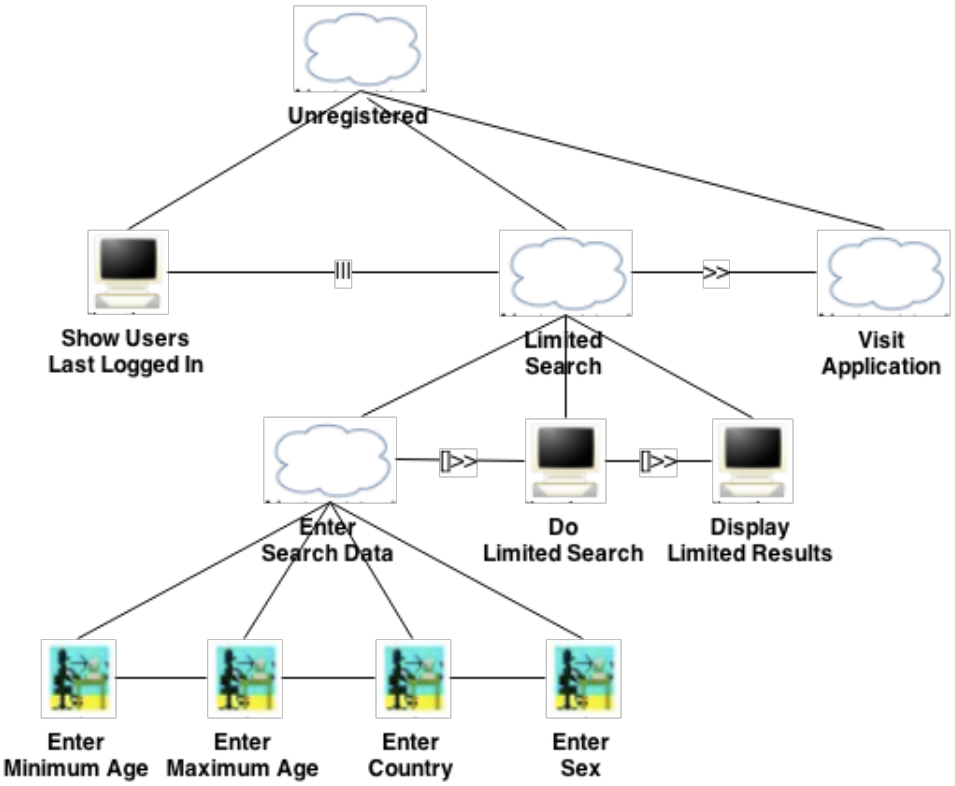


Figure 2: Unregistered User Activities

4.1.3 Create Account and Login

Figure 3 treats the activities needed to complete to create an account or login.
Concerning `Login`: the user needs to enter his/her username, password and choose if he/she wants his/her credentials to be remembered. Depending on the information entered, the system then logs the user in or shows 'unsuccessful feedback', allowing the user to try again.
Concerning `Create Account`: To register for a profile, a user needs to choose a username, password, enter a contact e-mailadress and his gender. If all required fields have been filled in, the system will create a profile for this user. Otherwise, non-filled or wrong-filled required fields will be highlighted, giving the user the chance to try again.

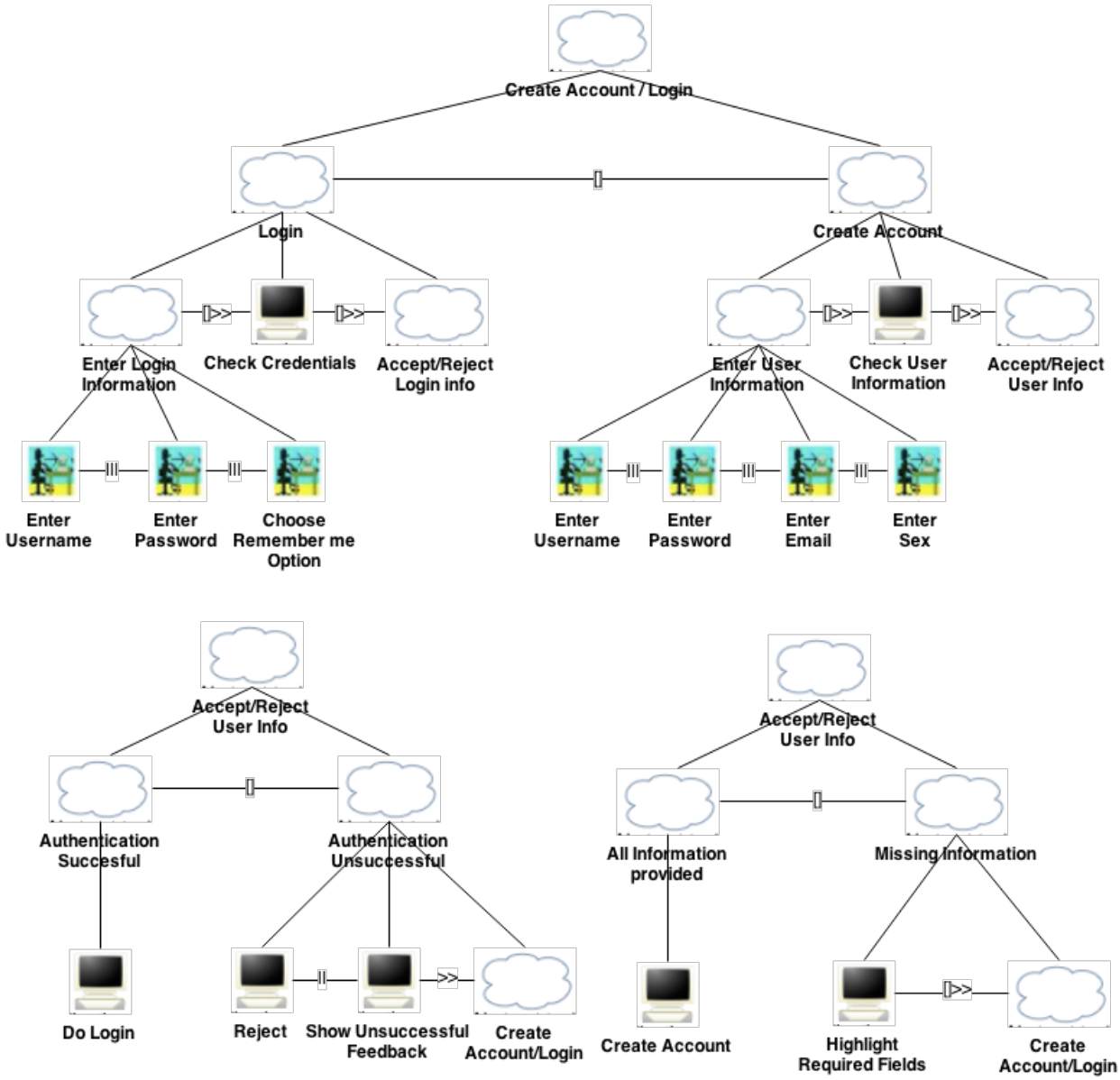


Figure 3: Creating an Account and Logging in

4.1.4 Registered Interact

Figure 4 shows the functionality to which a registered user has access: view his profile, view/search for a member, check his inbox and logout (bringing him/her back to the `login/create account` screen. Each of these functionalities will be treated in further detail in following sections.

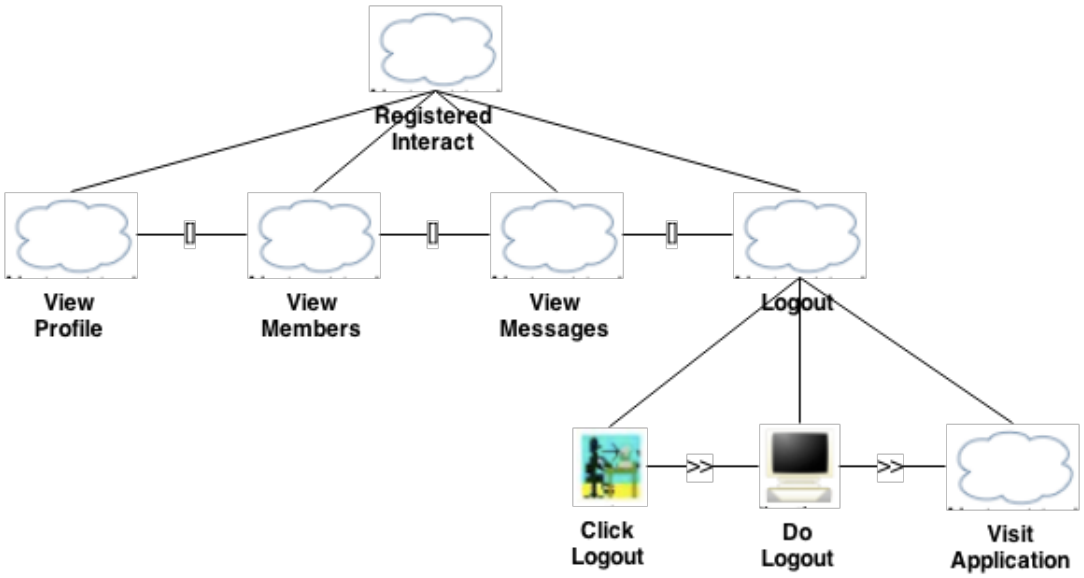


Figure 4: Root functionality for a registered user

4.1.5 View Profile

Figure 5 illustrates the activities the `View Profile` functionality offers to a registered user:

- `View Personal Profile`: Allows the user to check his own, personal, profile. This includes:
  - Checking and changing his personal profile data, such as name, password, e-mail, etc.
  - Add and remove likes.
  - Checking and manipulating his wall: posting, reacting and removing messages.
- `View Liked List`: send messages to, check user profiles of and dislike members liking the user.
- `View Attentions`: Checking and sending (figure 6) attentions.

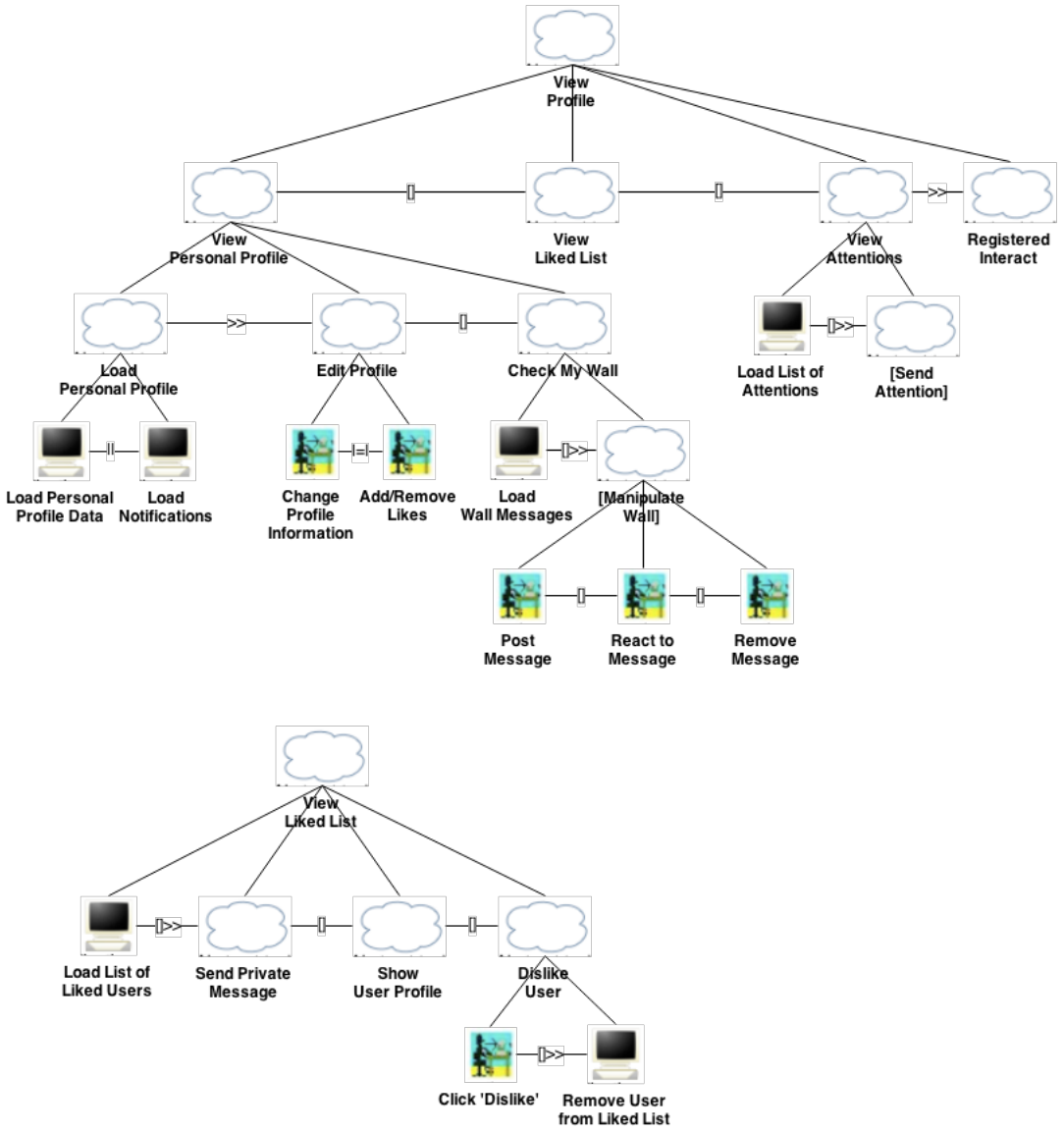


Figure 5: Functionality for a registered user to view his/her profile

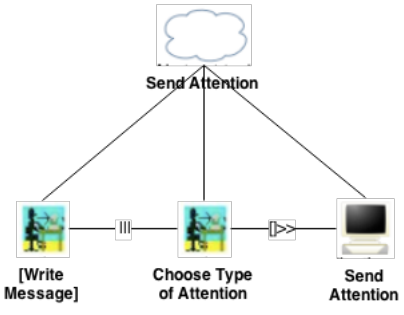


Figure 6: Sending an attention to a user

4.1.6 View Members

Figure 7 gives an overview of the functionality offered by the `Member` category of the system. It includes components to search for members (based on a wide variety of properties) and view their profiles (see section 4.1.7)

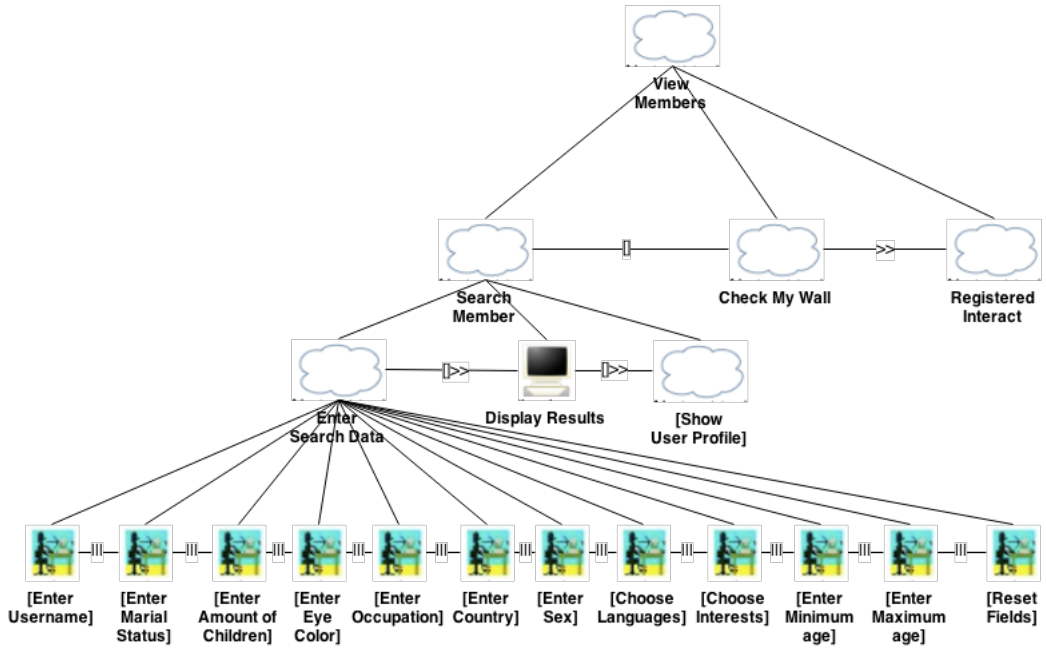


Figure 7: Functionality for a registered user to check and search for other members

4.1.7 Show User Profile

Figure 8 elaborates on the possibilities a user has to get in touch with another user: posting/reacting on a message on his wall, sending him a private message (see section 4.1.8), like/dislike him, sending him an attention and report/block him.

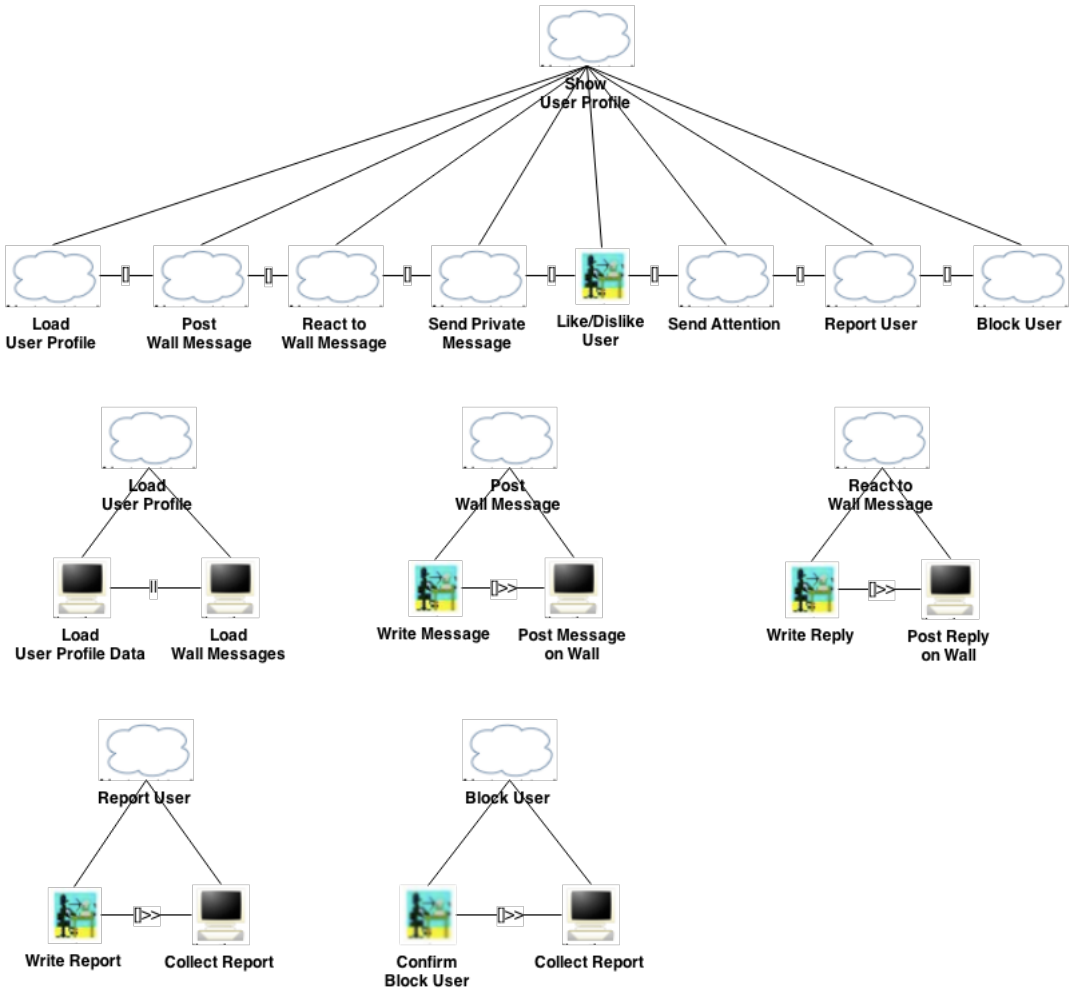


Figure 8: Functionality for a registered user to check the profile of another member

4.1.8 Send Private Message

Figure 9 depicts the activities a user needs to complete in order to send a private message to another user in the system. Chronologically, the user first has to choose a correspondent. The system will then automatically load the message thread (if exists) with that correspondent to give the user some context. Our user then only has to write and send his message to complete the `Send Private Message` activity.

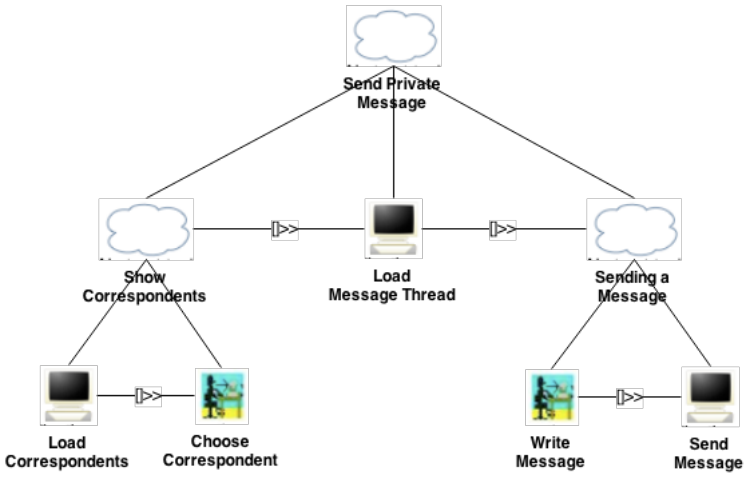


Figure 9: Functionality allowing a registered user to send a private message to another registered user

4.1.9 View Messages

Figure 10 shows the activities concerning the private messaging system of the web application. First of all the user has two options: or he sends a private message to someone he already has in mind (`Send Private Message`, see section 4.1.8); or he can send a message to a random user based on some criteria (e.g. gender and age).

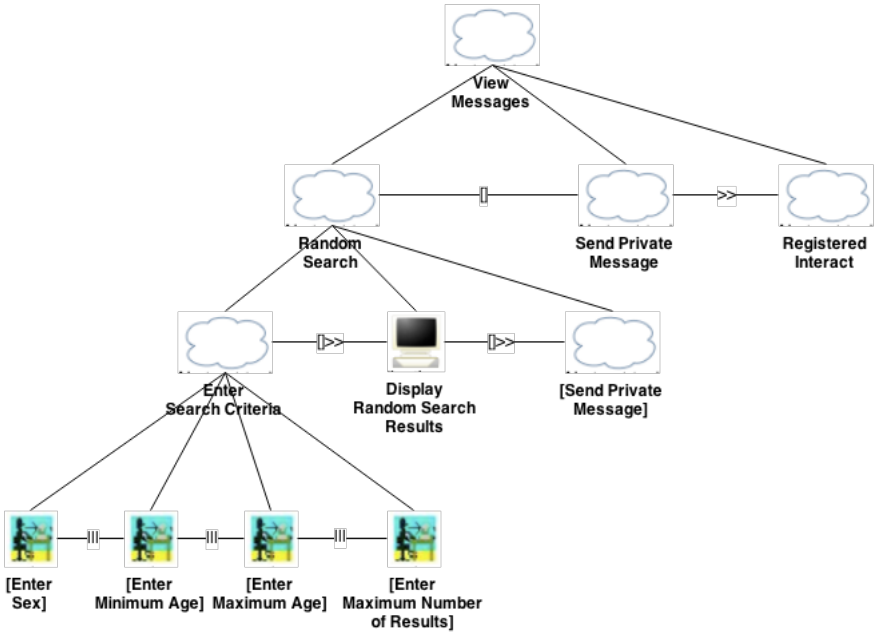


Figure 10: Functionality allowing a registered user to check his inbox for messages

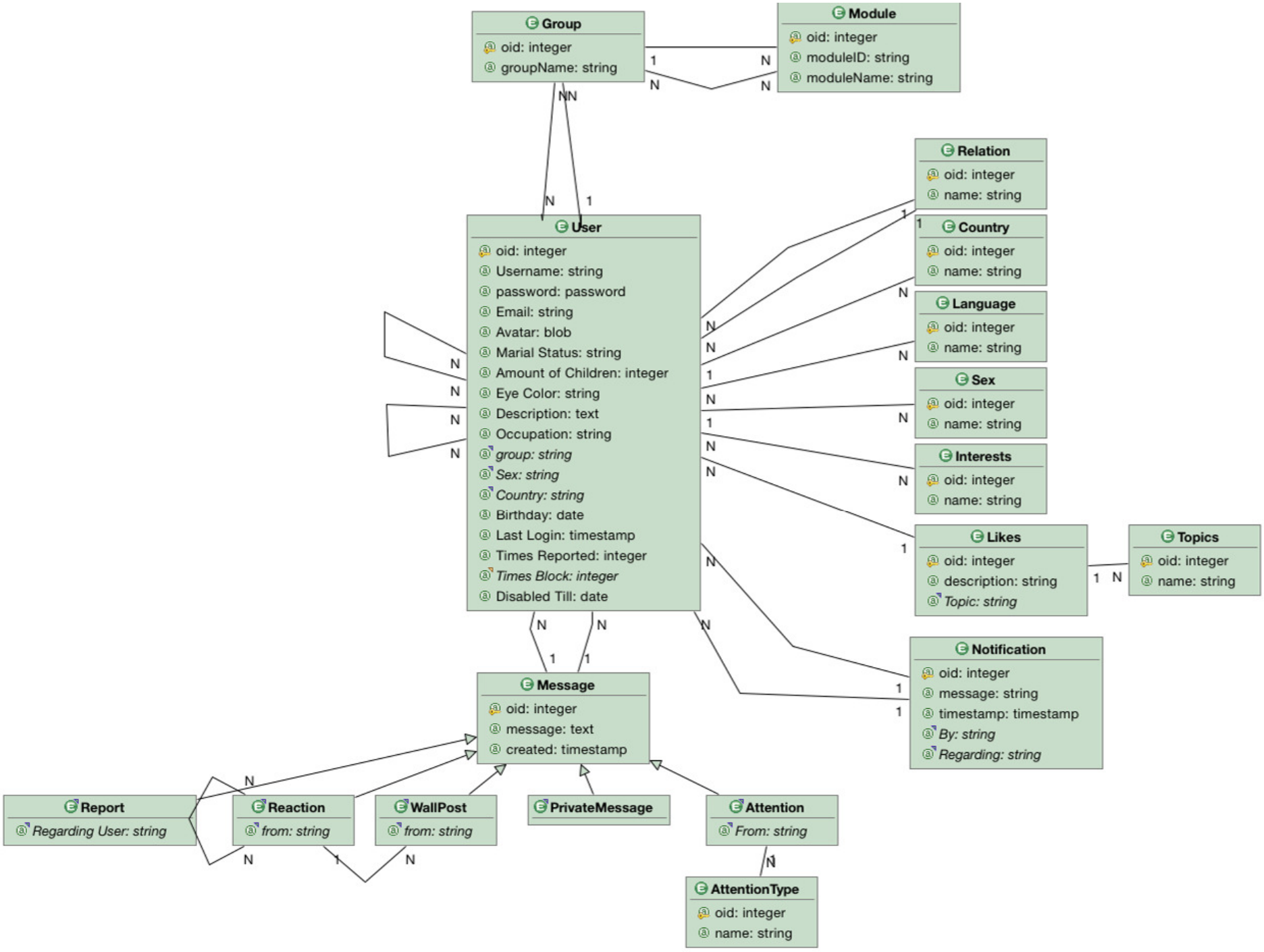
4.2 Information Modeling

In figure 4.2 we can see the domain model for our website. Most of the entities are self explanatory, but some will be treated further below.

- `Message` is a global entity used for all relationships between two users. We can see this by the two 1..N relationships with the User Entity. An ISA relationship is used to generate different types of messages our platform supports. For example: Private Messages, Attentions, Wall Posts, Reactions and Reports.

- The `Group and Module` Entities are generated by our used platform; Web Ratio[1]. These are used to identify the type of user and guide them to the proper site views.
- In section 3.2 we saw a user should be able to add random interests to his profile. Each such an interest falls under a certain topic, represented by the `Topic` Entity. This can be: Hobbies, Movies, Sports ... The specific link with the user is made by a `like`. For example the user likes a movie (Topic) with name *Lord of the Rings* (Like).
- All other profile information is linked to the `User` Entity directly. If some choice should be made between different options, this link with the user is made trough a relation with an other Entity. This is the case for Countries, Language, Sex and Interests.

[1] `http://www.webratio.com/`


Group
oid: integer
groupName: string
Module
oid: integer
moduleID: string
moduleName: string
User
oid: integer
Username: string
password: password
Email: string
Avatar: blob
Marial Status: string
Amount of Children: integer
Eye Color: string
Description: text
Occupation: string
group: string
Sex: string
Country: string
Birthday: date
Last Login: timestamp
Times Reported: integer
Times Block: integer
Disabled Till: date
Relation
oid: integer
name: string
Country
oid: integer
name: string
Language
oid: integer
name: string
Sex
oid: integer
name: string
Interests
oid: integer
name: string
Likes
oid: integer
description: string
Topic: string
Topics
oid: integer
name: string
Notification
oid: integer
message: string
timestamp: timestamp
By: string
Regarding: string
Message
oid: integer
message: text
created: timestamp
Report
Regarding User: string
Reaction
from: string
WallPost
from: string
PrivateMessage
Attention
From: string
AttentionType
oid: integer
name: string

4.3 Navigational Modeling and Site Structure Design

The main site structure is split into three main categories; Administrator, Registered and Public. These correspond to the three types of users we have on our platform. Each of these categories has an own Site View created in WebRatio. When applicable, parameter bindings are explained between the different pages of the website.

4.3.1 Public Site View

The main layout for the public available view for everyone is fairly simple. We have two different pages. `Create/Login` for the creation and login of new or existing users. And the homepage where some of the basic functionality is available for everyone to try out.

Figure 11: Public Site View

4.3.2 Administrator Site View

The view in figure 12 is displayed for users belonging to the admin class of the system. They get two main areas (displayed in blue). A logout area and the main dashboard. The home page of the dashboard is `Find Users`. Here the admin can search for a user and take actions. For each user he has the option to go to the edit page for this user, this is done by passing the user id of the selected user to the `Edit User` page. The same is possible for the `Report Summary` page of the users.

The `update` page contains all the notifications geared towards the administrators. This makes it possible for them to act on certain problems on the site about which they get notified.

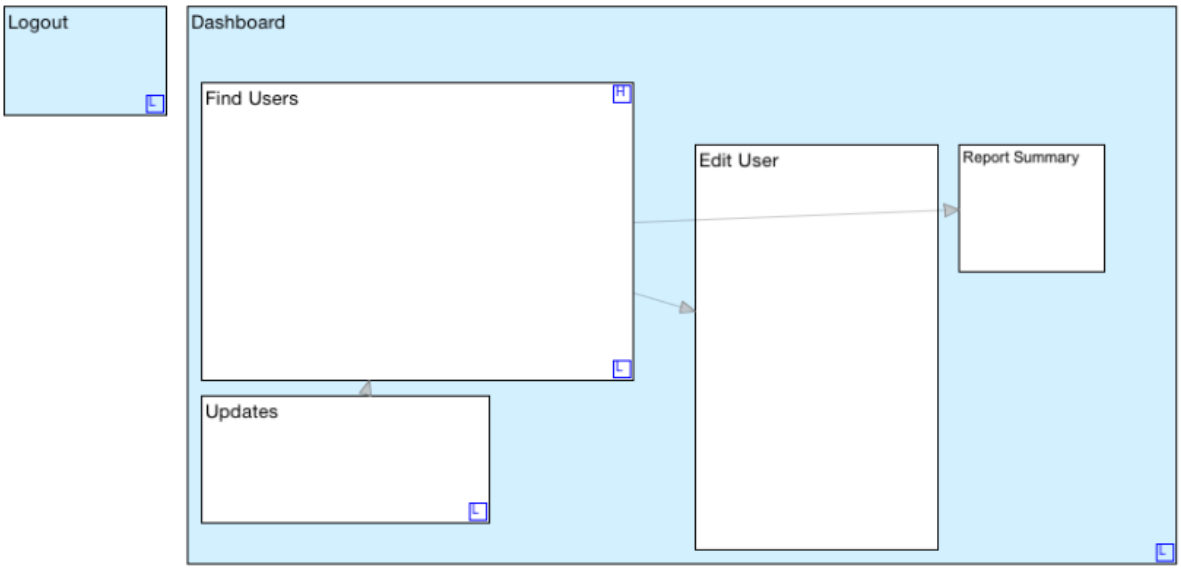


Figure 12: Administrator Site View

4.3.3 Registered Site View

The most complex view we have is for the registered users of our application (section 3.1.2). Here we have four main areas: Logout, Profile, Members and Messaging. Profile includes everything regarding the users own profile, Members regarding other users of the system and Messaging the messaging system allowing communication between the users.

All flows in this diagram have only one parameter bound, the user id of a user which is selected. For example, the incoming flow in the Messaging section contains the user id of the target user. This way the page knows which user to send the personal message to. A detailed parameter binding can be found in a later section.

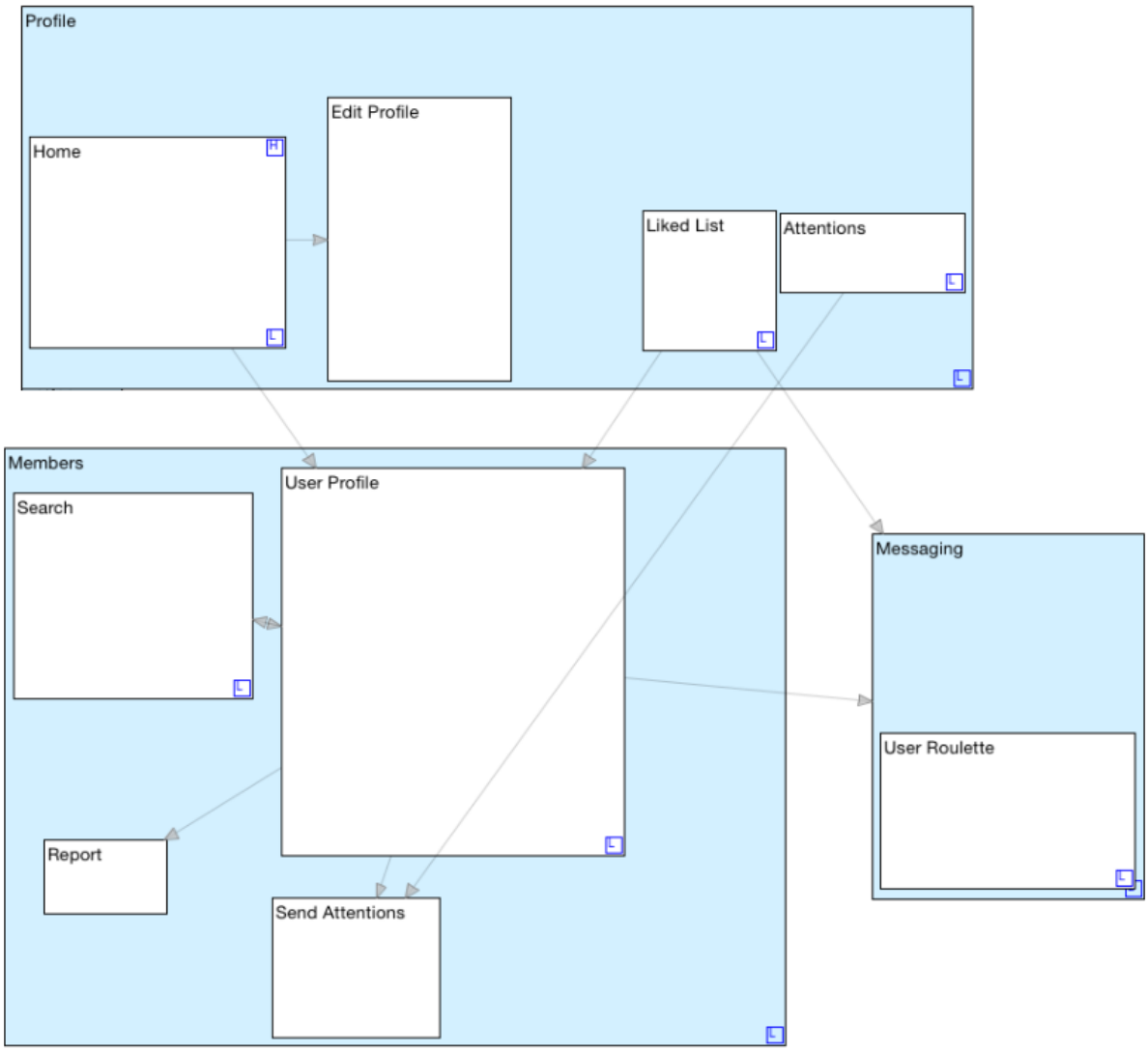


Figure 13: Registered Site View

5 Implementation Design

5.1 Introduction

In this section we will elaborate on all the necessary IFML models, generated in WebRatio[2], to implement the project. First the different modules, containing the application logic, we used are discussed.

Next we take a look at all the different pages we have in the application and how they interact with these modules.

5.2 IFML: Modules

5.2.1 Overview

An overview of all the different models in the application.

[2] http://www.webratio.com/

Figure 14: Modules Overview

5.2.2 Login

Figure 15 describing the Login Module.

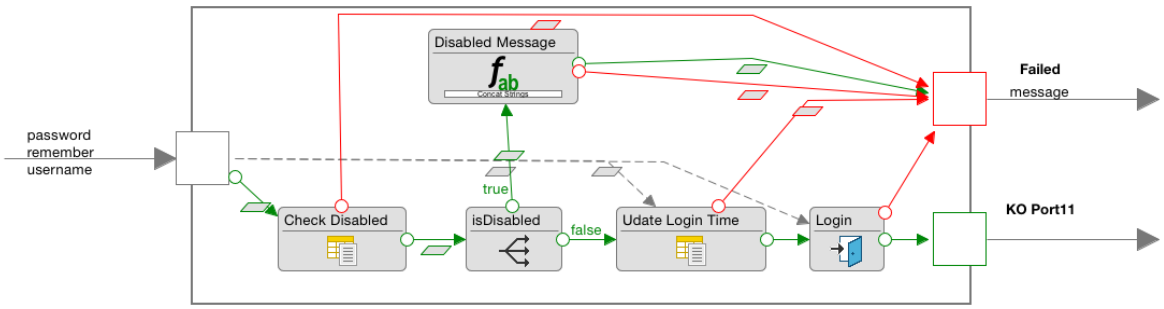


Figure 15: IFML of Login Module

When a user logs in, the login information is forwarded to this module. `Check Disabled` will check if the particular user account is disabled or not. If this returns true the Exit flow will be picked.

Otherwise we log the login time of the users and pass the information to the default WebRatio login module. This module will forward the user to the correct site view.

5.2.3 Logout

Figure 16 describing the Logout Module.

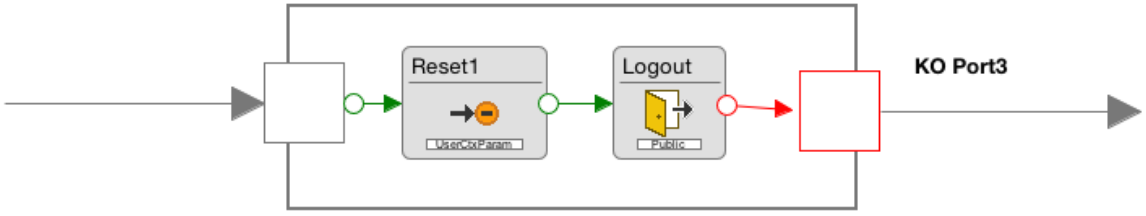


Figure 16: IFML of Logout Module

This module will remove the user id parameter from the context and invoke the logout module of WebRatio.

5.2.4 Profile Selection

Figure 17 describing the Profile Selection Module.

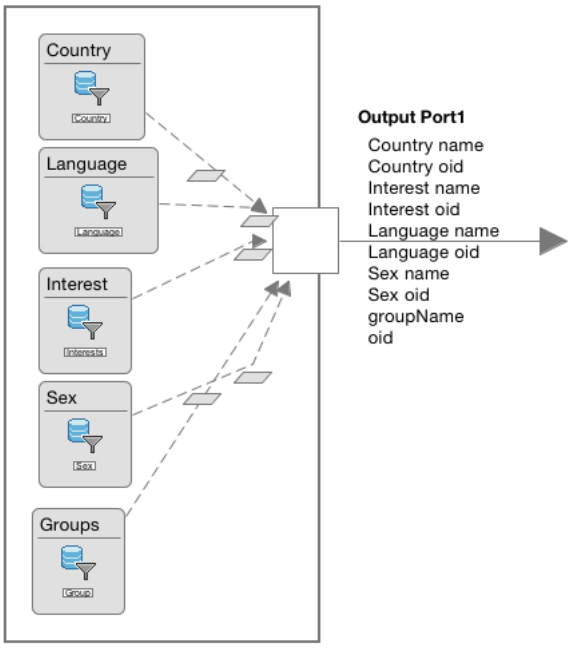


Figure 17: IFML of Profile Selection Module

Returns all the necessary information for selection field of a user input form. This was created to avoid duplication in the gathering of this information.

5.2.5 Update User

Figure 18 describing the Update User Module.

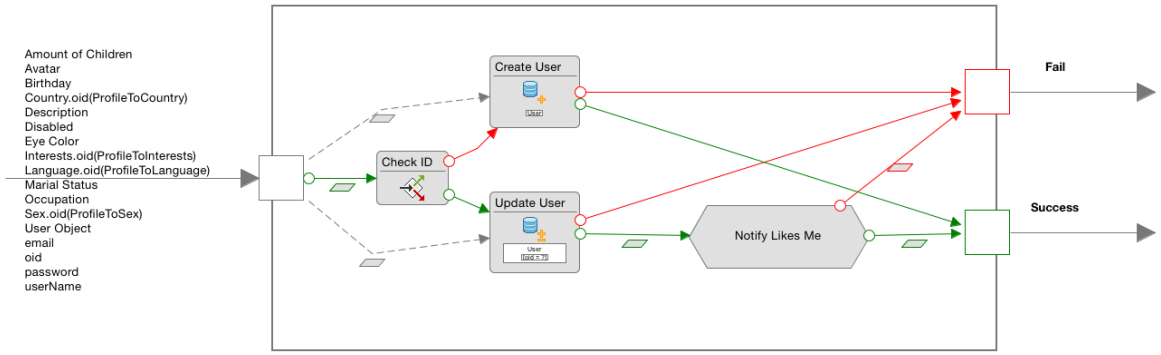


Figure 18: IFML of Update User Module

This module takes as input all the possible user profile settings and updates the user with it. If no oid key is given for a particular user, an new user entry will be created in the database. When a user profile is updated, the `Notify Likes Me` module is invoked to notify users of an update who like this particular user.

5.2.6 New Like

Figure 19 describing the New Like Module.

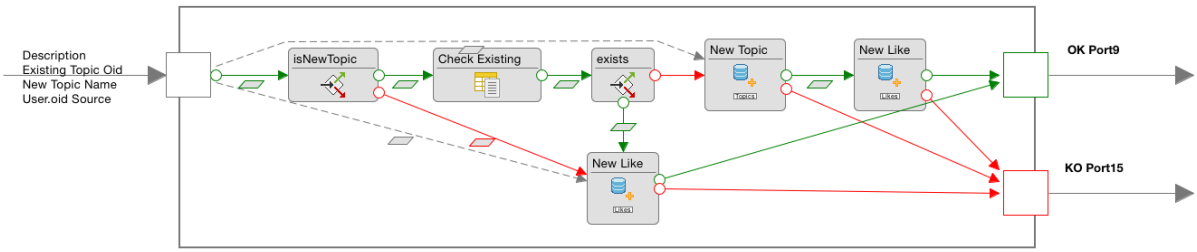


Figure 19: IFML of New Like Module

When a user adds a new like to his profile, this module will be invoked. It will try to match the topic specified by the user to others already in the database like; movies, tv, sport ... If a match is found, only the like will be added. If not, we create a new topic and use the topic id to create the new like.

5.2.7 Like User

Figure 20 describing the Like User Module.

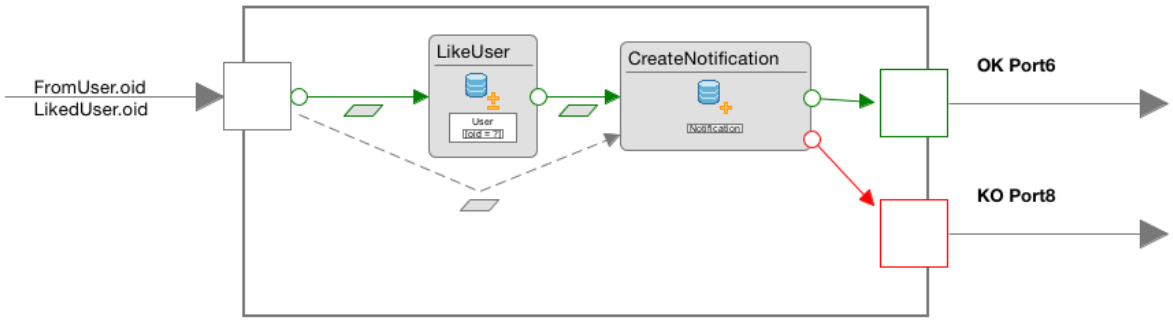


Figure 20: IFML of Like User Module

When a users likes an other users, this module will be used. This is done by adding a new relation in the database between the users. This action will also create a notification for the liked user, which he will see in his profile.

5.2.8 Dislike User

Figure 21 describing the Dislike User Module.

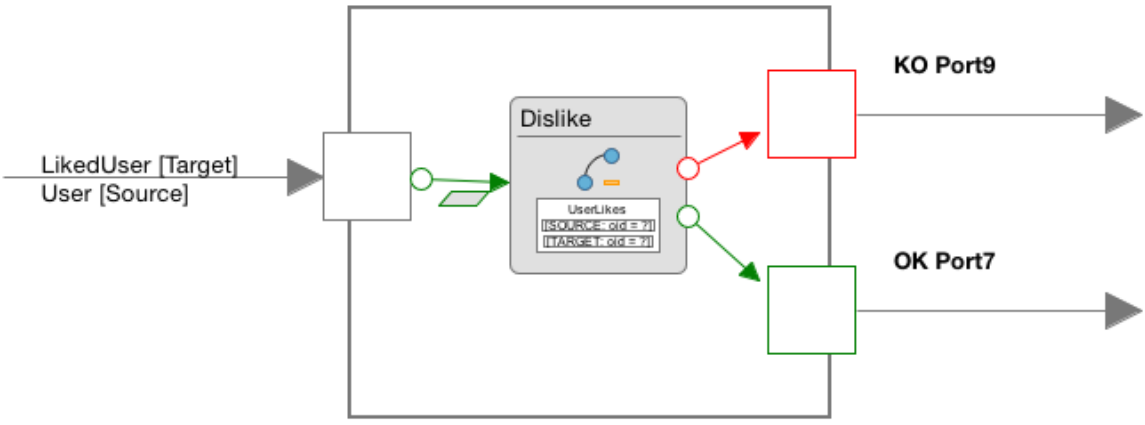


Figure 21: IFML of Dislike Module

Removes the 'like' relationship between the liked user and the source user.

5.2.9 Post Wall

Figure 22 describing the Post Wall Module.

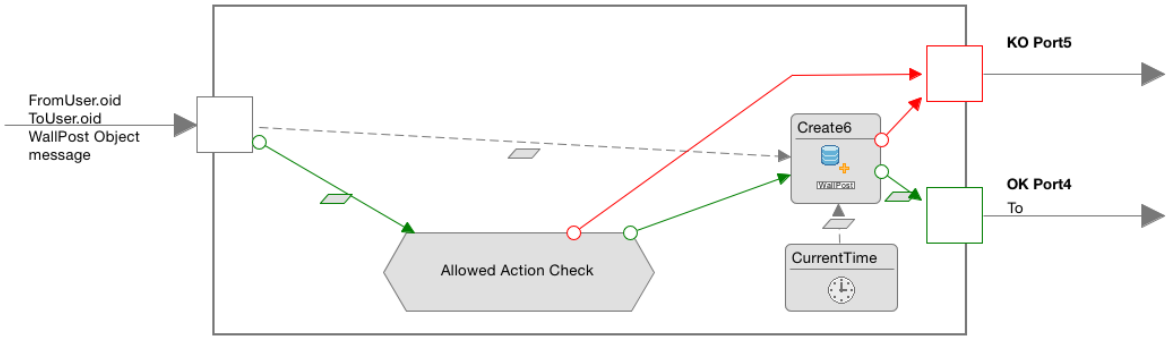


Figure 22: IFML of Post Wall Module

Module for posting a reaction on someone else's wall. When this action is performed we will first check if the user sending the message is allowed to do this. If the receiving user has blocked the sending user, the message wont be posted. This allows the user to protect his profile from unwanted users.

5.2.10 Post Reaction

Figure 23 describing the Post Reaction Module.

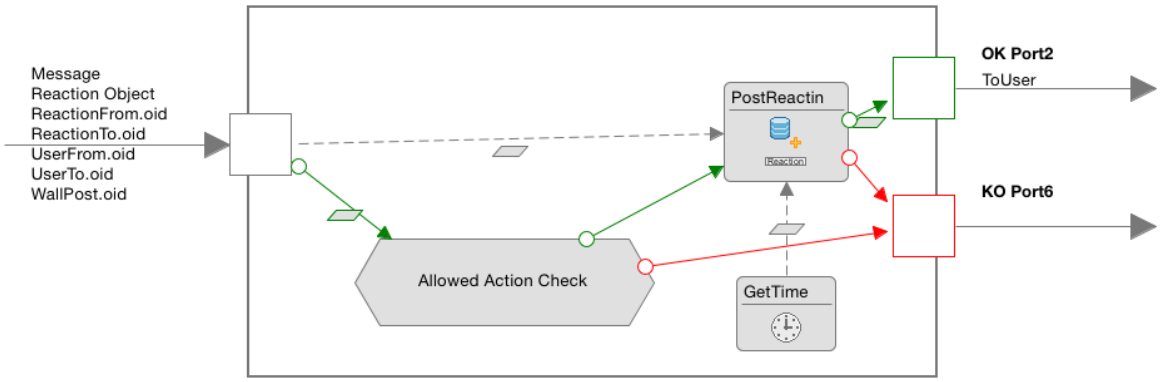


Figure 23: IFML of Post Wall Module

Posts a reaction on another message. For each post, it it checked if the user didn't block the sending user. If he did, the action will fail.

5.2.11 Send Attentions

Figure 24 describing the Send Attention Module.

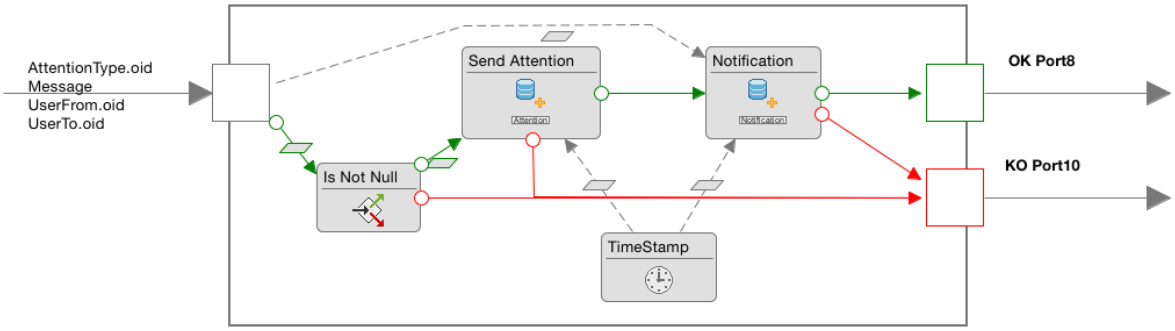


Figure 24: IFML of Send Attention Module

This module sends an attention to another user. The receiving user also gets an notification, stating that he received an attention from an other user. We also check the mandatory usertype.oid is included.

5.2.12 Personal Message

Figure 25 describing the Personal Message Module.

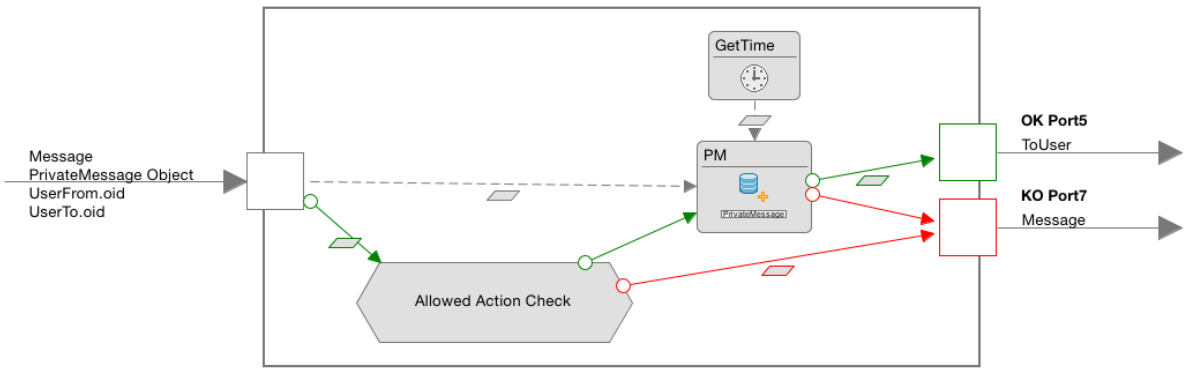


Figure 25: IFML of Personal Message Module

Module allowing message sending between users. It makes use of the `Allowed Action Check` module to handle blocked users.

5.2.13 Allowed Action Send

Figure 26 describing the Allowed Action Send Module.

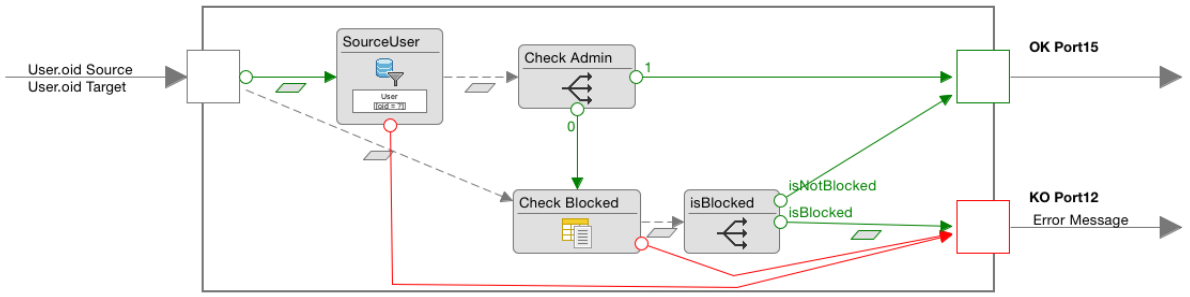


Figure 26: IFML of Allowed Action Send Module

This module checks if the Target user didn't block the source user. It is used whenever two users interact with each other and helps users manage their profile. If you block a user he can't send you messages, attentions ...

5.2.14 Report User

Figure 27 describing the Allowed Action Send Module.

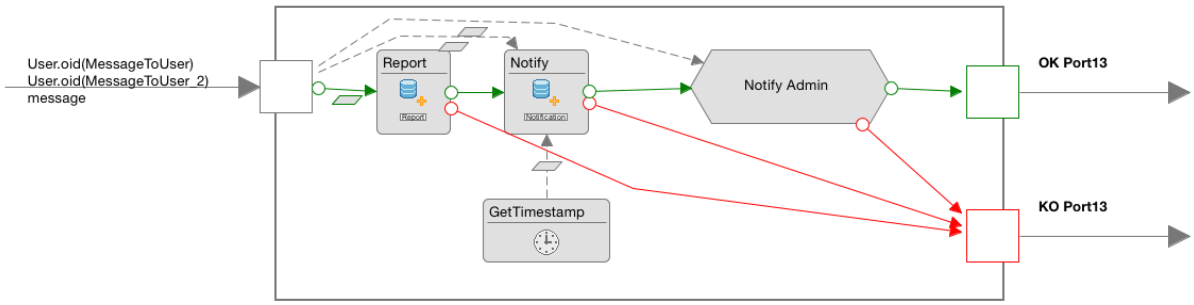


Figure 27: IFML of Report User Module

Module used to report a user to the administrators. This will create a report and the reported users gets notified he has been reported for some conduct. The `Notify Admin` module is also engaged to report the user to the administrator. He may use this as a basis to review a user.

5.2.15 Block User

Figure 28 describing the Block User Module.

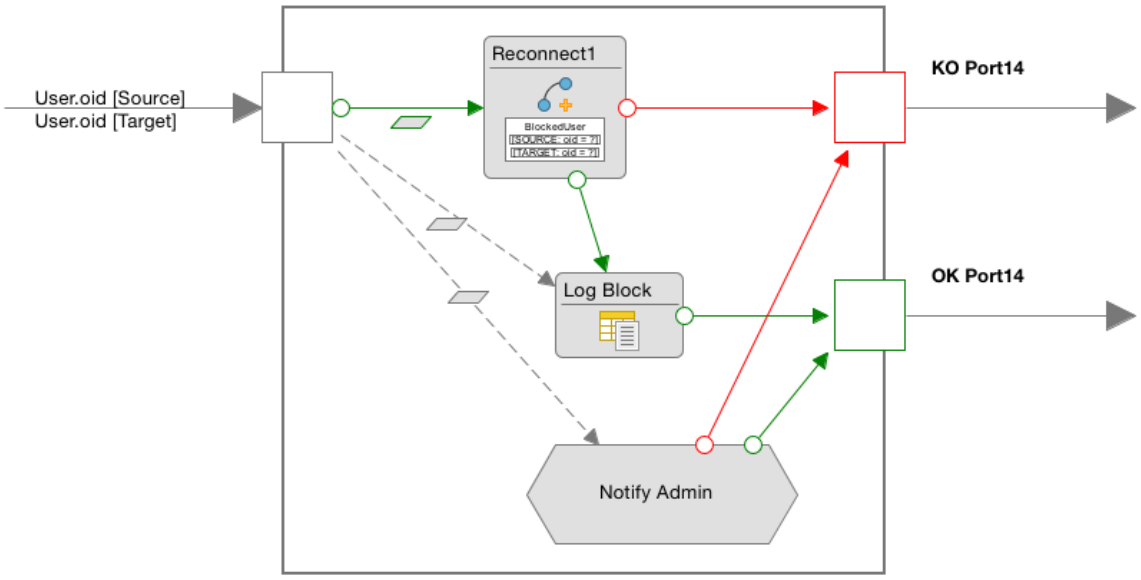


Figure 28: IFML of Block User Module

Module used to block a certain user from the ability to send messages or use other forms of communication. When a user gets blocked, a notification is send to the administrators so they can take further action if necessary. Every block is also logged in the database.

5.2.16 Notify Admin

Figure 25 describing the Notify Admin Module.

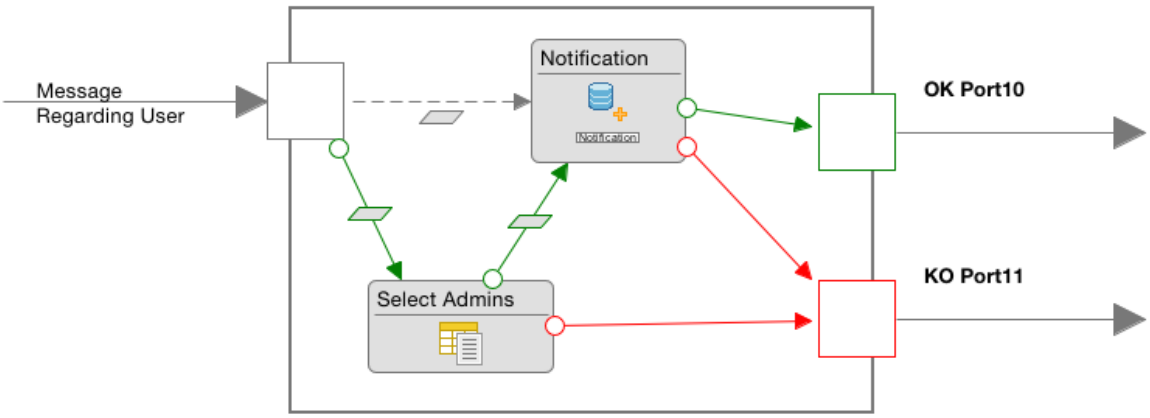


Figure 29: IFML of Notify Admin Module

Here we search for all administrators we have in the system and send them a notification containing the input message. May be used when a user is reported or when we want to send a notification to all administrators.

5.2.17 Delete Message

Figure 30 describing the Delete Message Module.

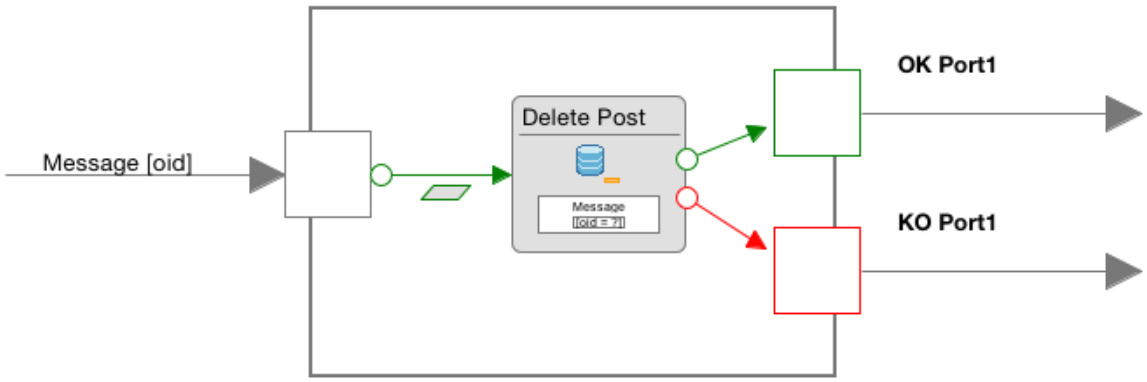


Figure 30: IFML of Delete Message Module

Delete a certain message based on the Message oid.

5.2.18 Messaging Page

Figure 31 describing the Messaging Page Module.

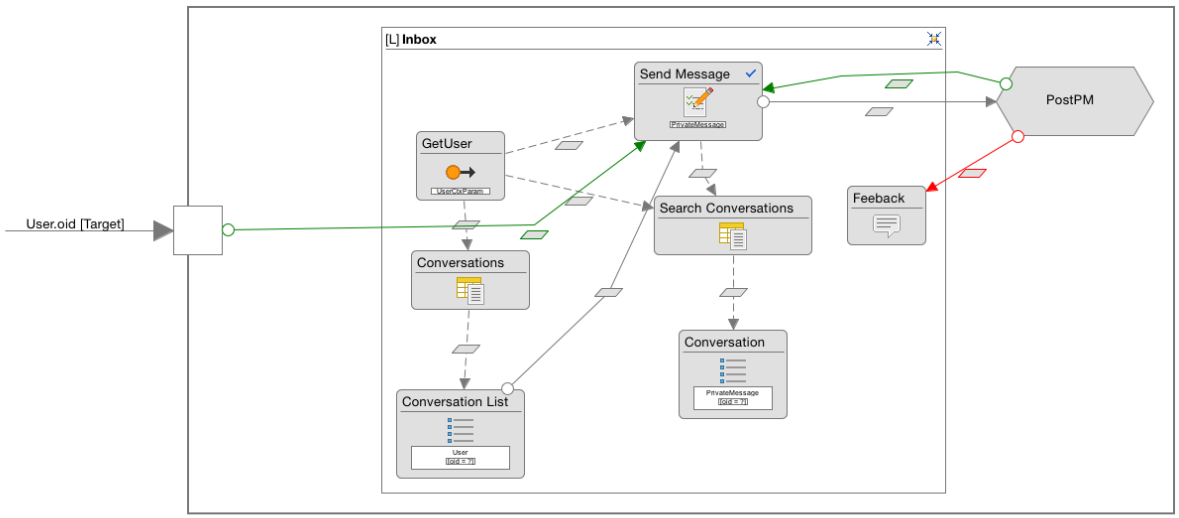


Figure 31: IFML of Messaging Page Module

This module differs from all others because it also includes the views to be displayed to the user. This was done because the module will be reused by the Administrator site view and Registered site view. A non mandatory input for this field is the `Target User oid`. This makes it possible to preselect a conversation with another user when displaying the messaging system.

Based on the current logged in user, `Conversations` will look for all users currently in a conversation with this user. They are then displayed in an interactive list where one can click to view the ongoing conversation. The `Send Message` form makes it possible to type in a message. If a value is set for the receiving user, `Search Conversations` will look for the matching conversation and display this in the list of messages.

Finally the `PostPm` Module (25) is used to post a new personal message trough the use of the form.

5.3 IFML: Views

In section 4.3 we already talked about the global structure of all the pages on the site. In this section we will dive deeper in the IFML models for each particular pages.

5.3.1 Home (Public)

Figure 32 describing the Home View.

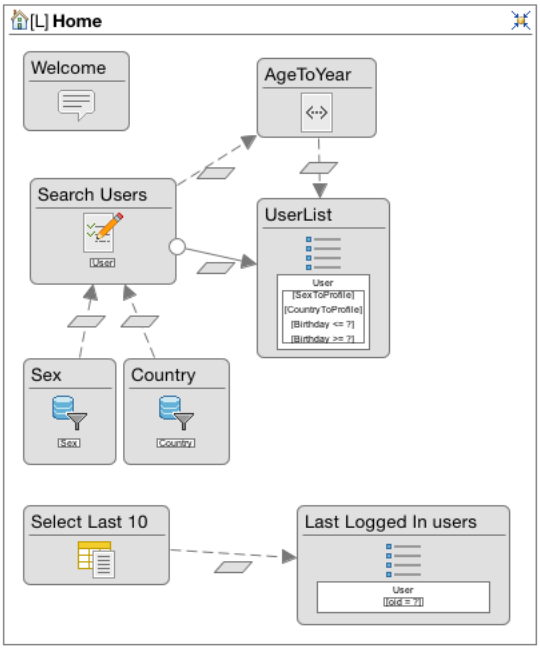


Figure 32: IFML of Home View

The home page provides two main functions. First it allows a visitor to use a search form to search for other users, the results are displayed in a list of maximum 10 users. `AgeToYear` converts the minimum age and maximum age the user can put in the form, to dates so it can be matched to a birthday between two dates. For this functionality a custom script needs to be used to transform the ages to a date range, where the birthday of the user needs to fall in between. The script is displayed in listing 1.

```groovy
#input int minAge, int maxAge
#output Date afterDate, Date beforeDate

import java.util.Calendar

Calendar cal = Calendar.getInstance();
Calendar cal2 = cal.clone();

if(maxAge == null) {maxAge = 100};
if(minAge == null) {minAge = 0};
afterYear = cal.get(Calendar.YEAR) - maxAge;  //1950
beforeYear = cal.get(Calendar.YEAR) - minAge; //1970

cal.set(Calendar.YEAR, afterYear);
afterDate = new Date(cal.getTimeInMillis());

cal2.set(Calendar.YEAR, beforeYear);
beforeDate = new Date(cal2.getTimeInMillis());

return ["afterDate": afterDate, "beforeDate": beforeDate]
```

Listing 1: AgeToYear Groovy Script

Secondly `Select Last 10` selects the users id's of the last ten user who logged in. Their information is then displayed in a list. Doing this with basic WebRatio elements provided difficult, therefore we chose to create a custom database query displayed in listing 2.

```sql
SELECT oid
FROM user
ORDER BY login DESC
LIMIT 10
```

Listing 2: SQL Query Select Last 10

5.3.2 Create/Login (Public)

Figure 33 describing the Home View.

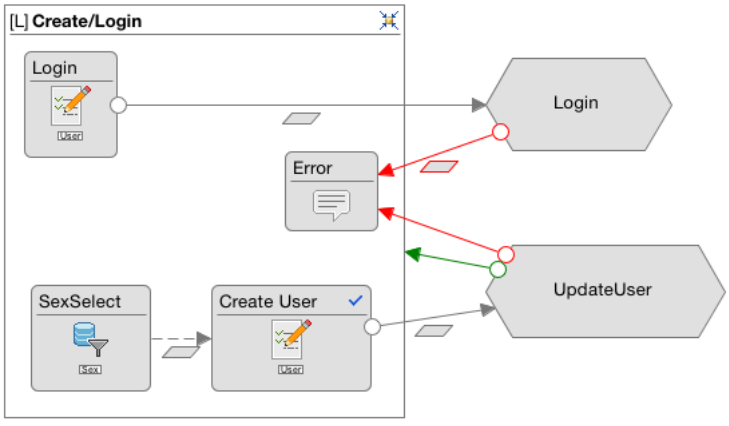


Figure 33: IFML of Create/Login View

The Create/Login page is very simple, it consists of two forms where the user needs to provide the required information. The `Create User` form is validated for the necessary field. The username field is also checked to be unique with other users in the database. Any feedback from the `Login Module` (15) or `UpdateUser` (18) is also displayed to the user.

5.3.3 Home (Registered)

Figure 34 describing the Home View of the Registered site view.

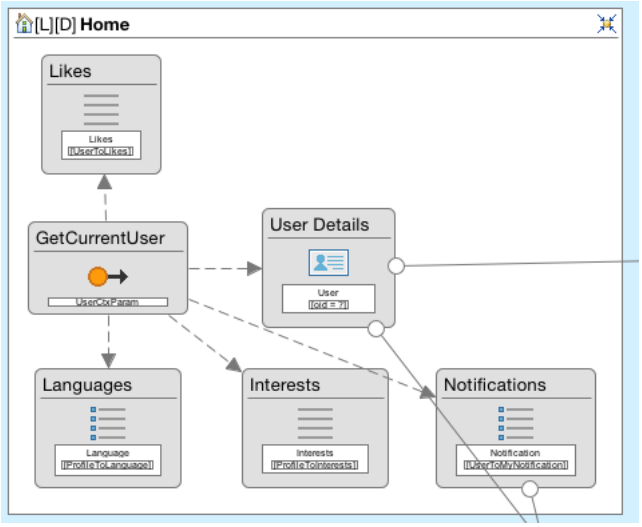


Figure 34: IFML of Home View

Based on the currently logged in user, all his personal information and notifications are displayed. 3 different outgoing flows are provided.

- **Edit Profile:** From User Details we allow the user to progress to the page to edit his profile.
- **My Wall:** A user can go to his personal public profile which includes his personal wall.
- **Follow Notification:** When a notification on the users profile, is the result of an action of an other user, like sending him an attention, the user gets the option to go directly to this users profile.

5.3.4 Edit Profile

Figure 35 describing the Edit Profile View.

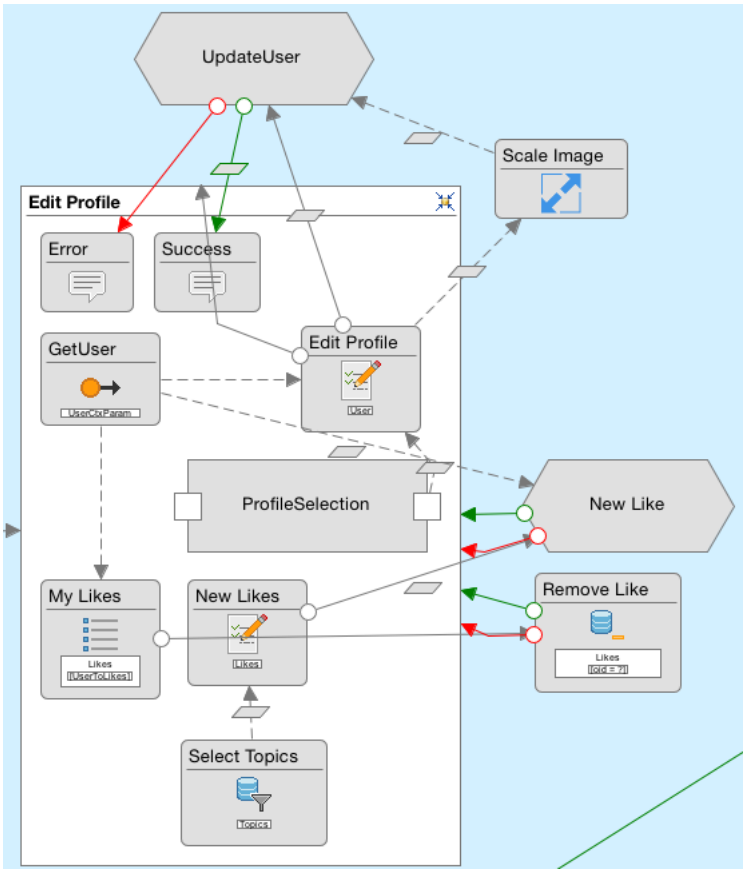


Figure 35: IFML of Edit Profile View

The user has the option to edit most of his profile information on this page. All information is therefor send to the `Update User` module(18). The user also has the option to add or remove different likes from his profile. For this the `New Like` module (19) is used.

5.3.5 Liked List

Figure 36 describing the Liked List View.

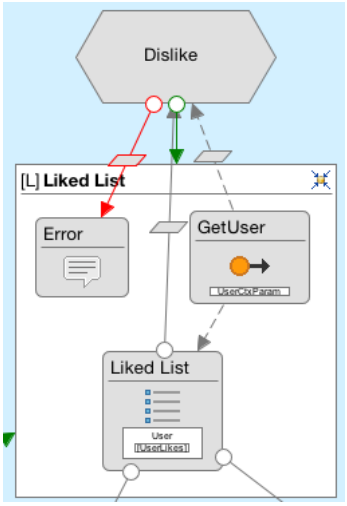


Figure 36: IFML of Liked List View

View shows a list of all the users liked by the current user. From this list tree flows are supported.

- **Dislike:** Remove the selected user from the list.
- **Profile:** Go to the selected users profile page.
- **Send Message:** Go to the messaging view of the site and preselect the selected user to view the conversation and have the option to send a message.

5.3.6 Attentions

Figure 37 describing the Attentions View.

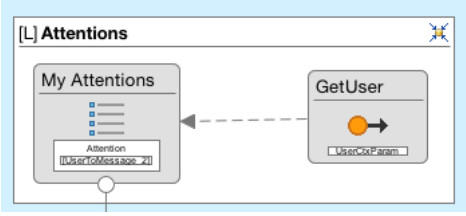


Figure 37: IFML of Attentions View

View showing an overview of all received attentions to the user. The outgoing flow enables the sending of of the same attention back to the sending user.

5.3.7 Search

Figure 38 describing the Search View.

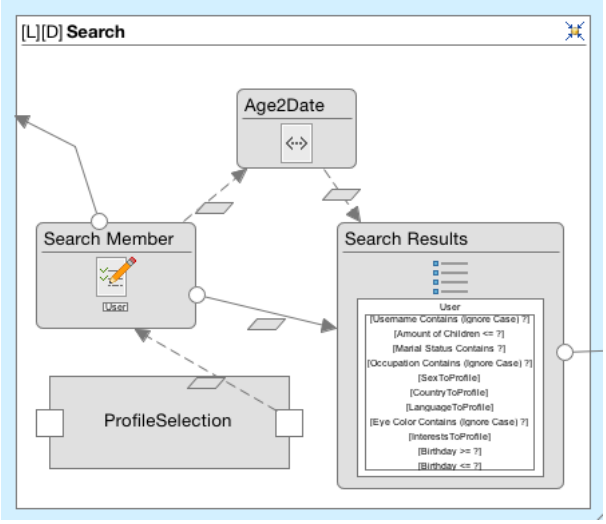


Figure 38: IFML of Search View

This view enables the user to selectively add search parameters to filter the search result. All the users matching the search terms will be displayed in the list with their public information. The user has an option to filter on an maximum and minimum age. The same script as in listing 1 on page 32, is used.

5.3.8 Report

Figure 39 describing the Report View.

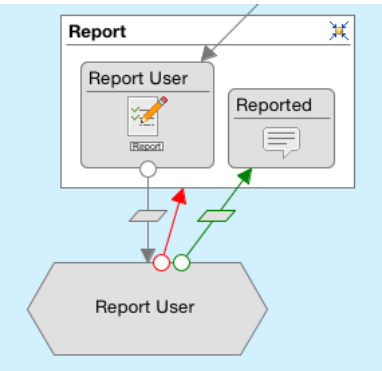


Figure 39: IFML of Report View

Page for reporting other user to the administrators of the site. The user has the option to attach a personal message, which will be included an shows to the admin as extra information in the report.

5.3.9 User Profile

Figure 40 describing the User Profile View. This view is the most complex view in the application. To give an overview of all the options on this page, we split them up below with a short explanation.

Profile Information The `User Details` component shows all the information of the selected user profile. Other information is also shown in the corresponding lists.

Wall Posts All the wall post to the user are also displayed. Their is an form to make a new pubic wall post to the users profile using the `PostWall` Module (5.2.9).

Reactions Reacting on every wall post is also possible. This is done by clicking on a certain wall post and filling in the reaction form. Subsequently the `PostReaction` Module (5.2.10) is used to post and notify the users.

Action Bar This components offers different flows to take actions towards the selected users.

- **(Dis)like:** Depending on the current like status of this user, a dislike/like flow button is show to the user. A successful action will lead to the overview of all the liked users. A successful dislike will show a positive feedback to the user.
- **Block User:** Using the `Block User` Module (5.2.15), blocks the selected user removing the ability for him to communicate with the current logged in user.
- **Report User:** Sends the user to the Report User page(5.3.8).
- **Send Message:** Forward user to the messaging view of the application (5.2.16) and preselect the selected user.
- **Send Attention:** Send the user to the `Send Attention` View targeting the selected user (5.3.6).

5.3.10 Send Attentions

Figure 41 describing the Send Attentions View.

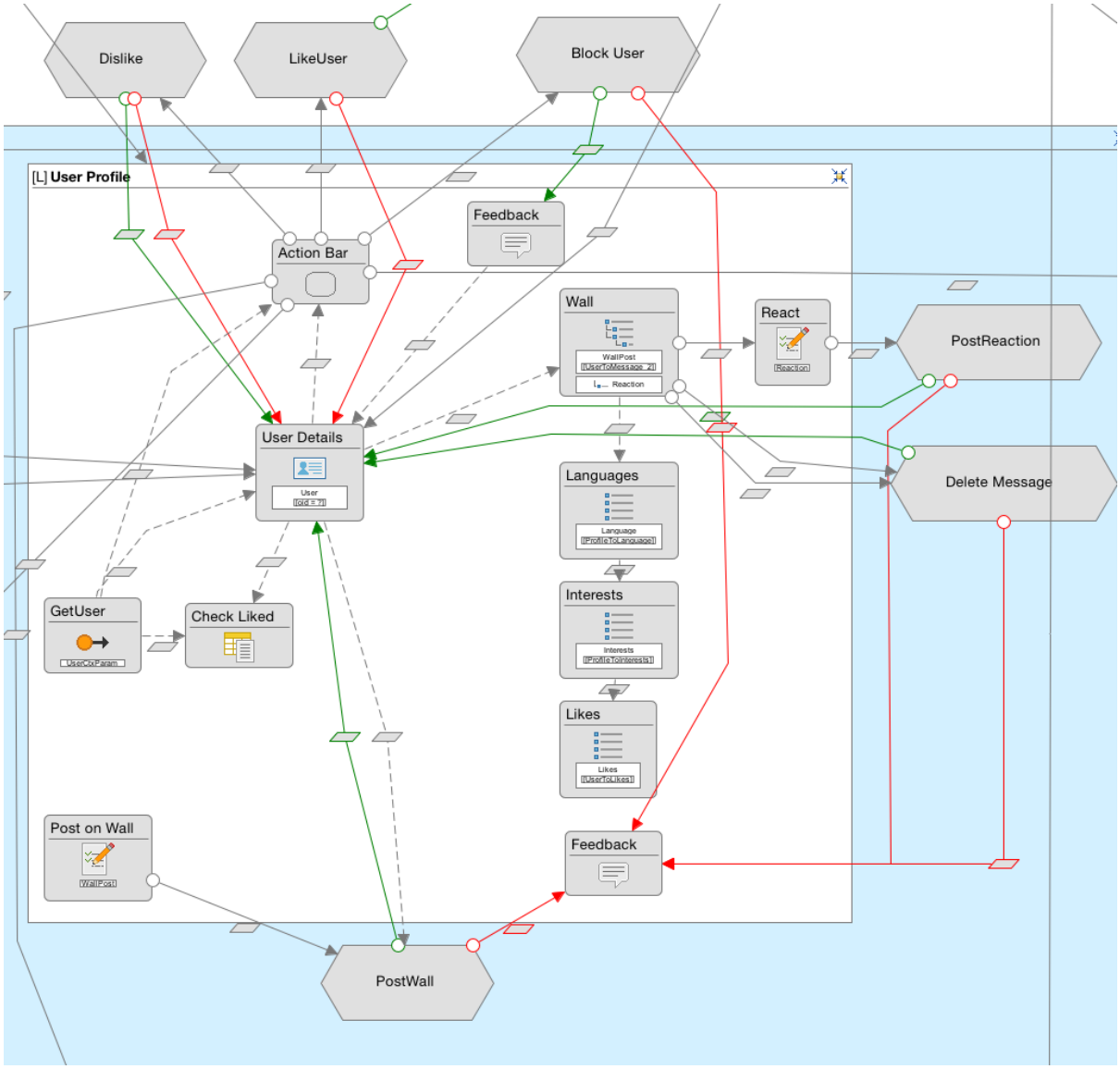


Figure 40: IFML of User Profile View

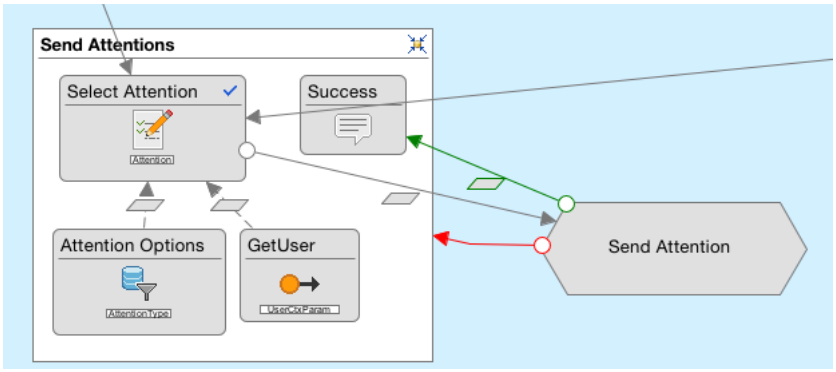


Figure 41: IFML of Send Attentions View

User gets the option to pick from a list from attentions and attach a message. Then the `Send`

`Attention` module (5.2.11) is used to send the message and notify the user he received a new attention.

5.3.11 User Roulette

Figure 42 describing the User Roulette View.

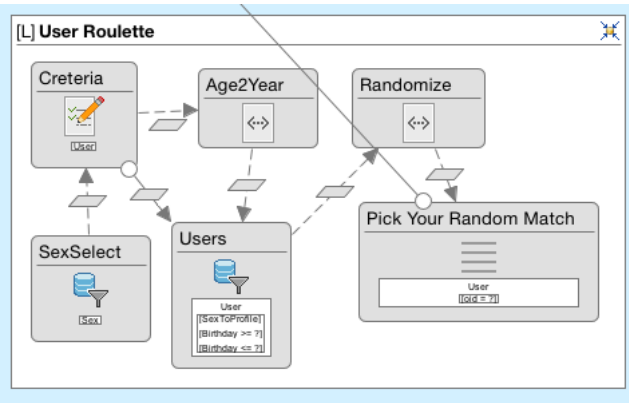


Figure 42: IFML of User Roulette View

Based on the by the user defined criteria, a list of users is selected. To randomize the users we shuffle the resulting array list of all the user.oid in from the selection. This is done using a goovy script which can be found below.

```
#input Integer maxResults, Integer[] list
#output Integer[] mixedList

import java.util.*;
import java.util.Collections;
import java.util.Arrays;

log.error("====== STARTING RANDOM PERSON SCRYPT =======")
java.util.Collections.shuffle(Arrays.asList(list));
def result = Arrays.copyOfRange(list, 0, maxResults)
return ["mixedList": result]
```

Listing 3: Randomize Groovy Script

5.3.12 Report Summary (Admin)

Figure 43 describing the Report Summary (Admin) View.

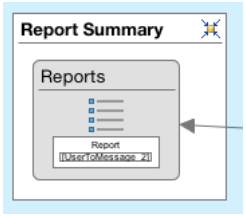


Figure 43: IFML of Report Summary (Admin) View

A view created for the administrators, it displays all the reports from the incoming user oid. This should help the admins to identify possible troublemakers and take further action if required.

5.3.13 Find Users (Admin)

Figure 44 describing the Find User (Admin) View.

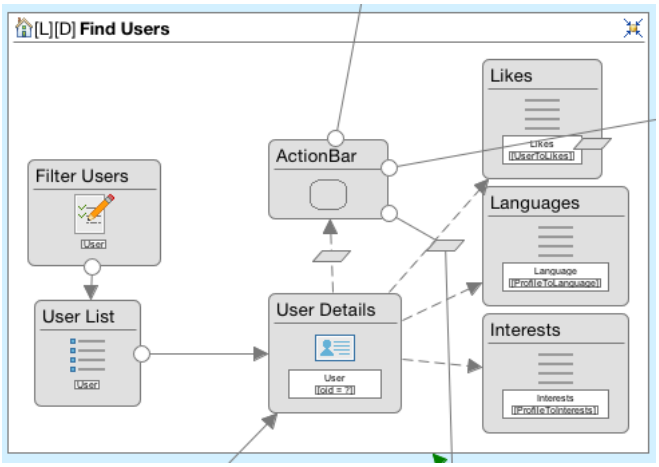


Figure 44: IFML of Find User (Admin) View

Like a regular user, admins have the options to search for a user based on different criteria. When a user is selected from the `User List`, his user details are loaded. It is then possible to pick one of the following flows from the `ActionBar`.

- **Edit Profile:** Send the user to the `Edit Users` view (5.3.14).
- **Reports:** Forward to the `Report Summary` view (5.3.13) to display all the reports concerting the selected user.
- **Send Message:** Link to the messaging system where a conversation will be opened based on the selected user.

5.3.14 Edit User (Admin)

Figure 45 describing the Edit User (Admin) View.

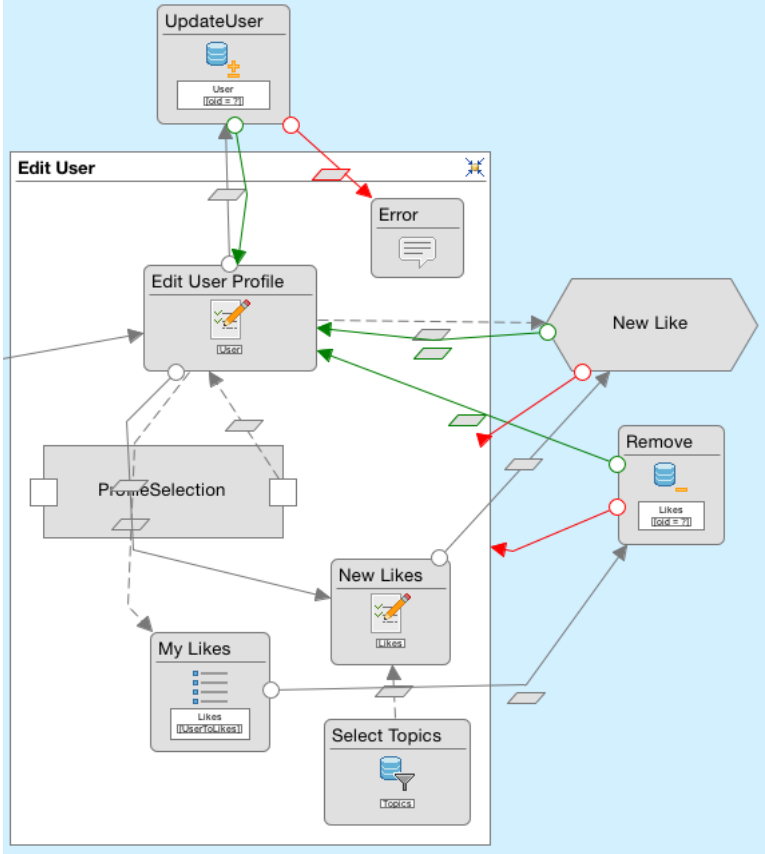


Figure 45: IFML of Edit User (Admin) View

Administrators have the capability to edit all the information of a particular user. Based on the incoming flow, the user information is displayed in the form. This view used the database modules directly because no special validation / notifying needs to be done when the admin performs his actions.

5.3.15 Updates (Admin)

Figure 46 describing the Updates (Admin) View.

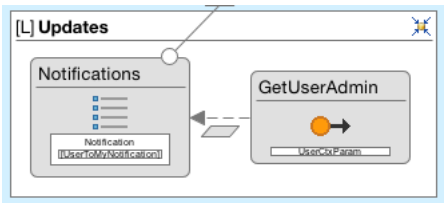


Figure 46: IFML of Updates (Admin) View

A view where an administrator can view all his notifications regarding things like reports and users being blocked by other users. This helps identifying certain users who are misbehaving on the platform.

5.4 Style and Template Design

The basic layout is provided by WebRatio and his Layout Editor. This gives us the capabilities to edit the global layout for every view based on a grid system. We also opted to install an other template to

be used by WebRatio. This template is based on the Pure CSS framework [3]. It provides a clean and minimalistic CSS framework based on a Grid system and some default coloring options, table styles ...

The combination of both these technologies offers us a fast way to design different web pages with a professional and clean looking user interface.

6 Reflection and Problems

Reflecting back on all the work we did for this project, most requirements were fairly easy to implement. We did encounter some problems when requiring some non-standard functionality of the WebRatio Platform. We chose to do our implementation in WebRatio, because we feel it is part of the experience of making IFML models and using them directly to implement a working website.

This leads us to one of the hardest challenges of the platform. When we needed some non standard functionality of the platform we needed to write our own query's and groovy scripts. Mainly for the script the documentation was limited, leading to some serious overhead to create only some simple scripts. In other traditional implementation frameworks for creating an online application, these problems seem to vanish, mainly due to the programming experience that is built up by using them.

The same can be said about the layout and template capabilities. WebRatio provides great tools to design and use simple elements and build a site. This works great for most boilerplate designs, but when some real customization needs to be done for a page, the learning curve for this seems fairly steep. Therefore we couldn't implement some of the user interface designs we wanted to improve the user experience. A process which should be fairly easy when we used an other framework.

7 Conclusion

Concluding our work on Date4Live, we are happy with the overall result. We started by creating all the functional requirements and audience modeling. Followed by building the CTT's and the overall design structure of the site. Next we started work on the IFML modeling in WebRatio for each page, creating separate modules when seen fit. In the end, we succeeded in fulfilling all the functional requirements and provided a working online application designed using IFML modeling in the WebRatio Framework. This all keeping a clean set of IFML Models and View and while using most of the given functionality from our platform like user groups and separated site views.
Once again, the online version of the site can be found at:

`http://date4life.timwitters.me/Date4Life/`

[3] Pure CSS Website: `http://purecss.io/`