SWIMv2 Project

Requirements analysis and specifications document

Nicolò Andronio

Andrea Brancaleoni

Table of Contents

Introduction

1.1. Project Scope

The SWIMv2's main target is to connect friends on a network, allowing them to offer and require help from other users. Each one can publish a set of skill to match their competence and aim. This application also implements a basic set of social functionalities: users can send and receive friendship requests or messages and can update their profile.

1.2. Target Users

There is no specific focus on users' types. Anyone can access the network.

1.3. Functionalities

The system will implement these functionalities:

- (Visitors) An user can register on the portal;
- (Visitors) Once registered, they can log into the site and access its features;
- Create and manage a profile;
- Add a set of skills to their profile;
- Send and accept friendship requests;
- Send and manage messages;
- Search users for help;
- Search friends for help;
- Provide feedbacks for other users;
- Propose new skill-sets;
- (Admin) Declare new skill-sets;
- (Admin) Validate skillset proposals.

1.4. Goals

Here follows what the project is aiming to accomplish:

- Interconnect users;
- Provide a network of professionals;
- Construct a simple & trusted matchmaking system.

1.5. Limitations

The software will suffer from the following limitations:

- Limited time for implementation;
- The system must be reliable;
- The system must handle concurrent situations;
- There is no particular security requirement: standard measures will be adopted;

Functionalities Details

2.1. Actors

There are four types of actors, plus a special one.

2.1.1. Unregistered users

They are visitors. They can obtain access to the website by logging in through the login panel or by registering to the website and undergoing a registration confirmation process. A registered user can be demoted to unregistered user if his/her cookies expire.

2.1.2. Registered users

They are users that correctly logged onto the project website. Logging out demotes a registered user to visitor.

2.1.3. Administrators

They are registered users with higher privileges. There is always at least one administrator user, which is created by the system developers. Any administrator can promote a registered user to administrator.

2.1.4. Super user

This user rank is not achievable by any means. It exists only for administration purposes. Super users are created and managed by developers. A super user is an administrator, with the additional powers to edit anything in the site. It can create, update, delete users and administrators at will.

2.2. Functionalities groups overview

In the following subsections, we will define different functionality domains, each of which contains a set of strictly related functionalities. For every group, we provide a general description, along with the interested actors (specified between parenthesis). The following main sections will explain the domain groups at a much higher level of detail.

2.2.1. Authentication *(Visitors, Registered users)*

Covers login, logout, registration, password retrieval and account retrieval.

2.2.2. Personal profile *(Registered users)*

Include creation, update and management of user profiles.

2.2.3. Skills management *(Registered users, Administrators)*

Contains functions related to the definition of a subset of skills grabbed from predefined skill sets. Also allows to create, review and validate skill sets proposal for addition.

2.2.4. Social *(Registered users, Administrators)*

Deals with friendship relations among users (sending / accepting requests and polling the database for friendship details). Also contains instant-messaging functionalities.

2.2.5. Help research *(Visitors, Registered users)*

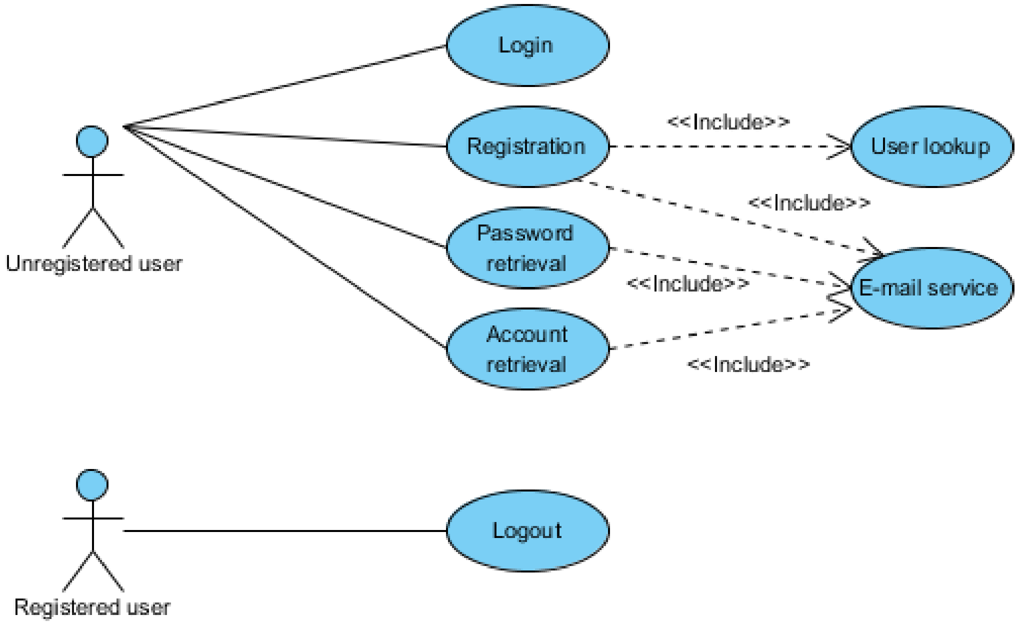
Covers matchmaking of help requests, friendship filters and feedbacking.

2.2.6. General administration *(Administrators, Super users)*

Allows privileges modification and deletion of existing accounts.

2.3. Functionalities groups description

2.3.1. Authentication



The authentication group provides functionalities for managing accounts. It supports creation and confirmation of new accounts, and provides access to existing ones.

2.3.1.1. Login

Allows visitors (unregistered users) to access the website through an username-password form. Logs login attempts, detects unauthorized accesses and protects against sql-injection/xss.

2.3.1.2. Registration

Asks for new user credentials and verifies that they are coherent by looking up other users accounts. Exploits the mailing service to send a confirmation e-mail. Handles confirmation links and account activations.

2.3.1.3. Password retrieval

Handles the case in which an user forgot his/her password. Access the user’s e-mail provided his/her username, resets password and send an e-mail report.

2.3.1.4. Account retrieval

Handles the case in which an user forgot his/her username. Provided an e-mail address and a correct answer to a security question, resets credentials and send an e-mail report.

2.3.1.5. Logout

Forces the session to end, deletes cookies and local informations and updates user’s status.

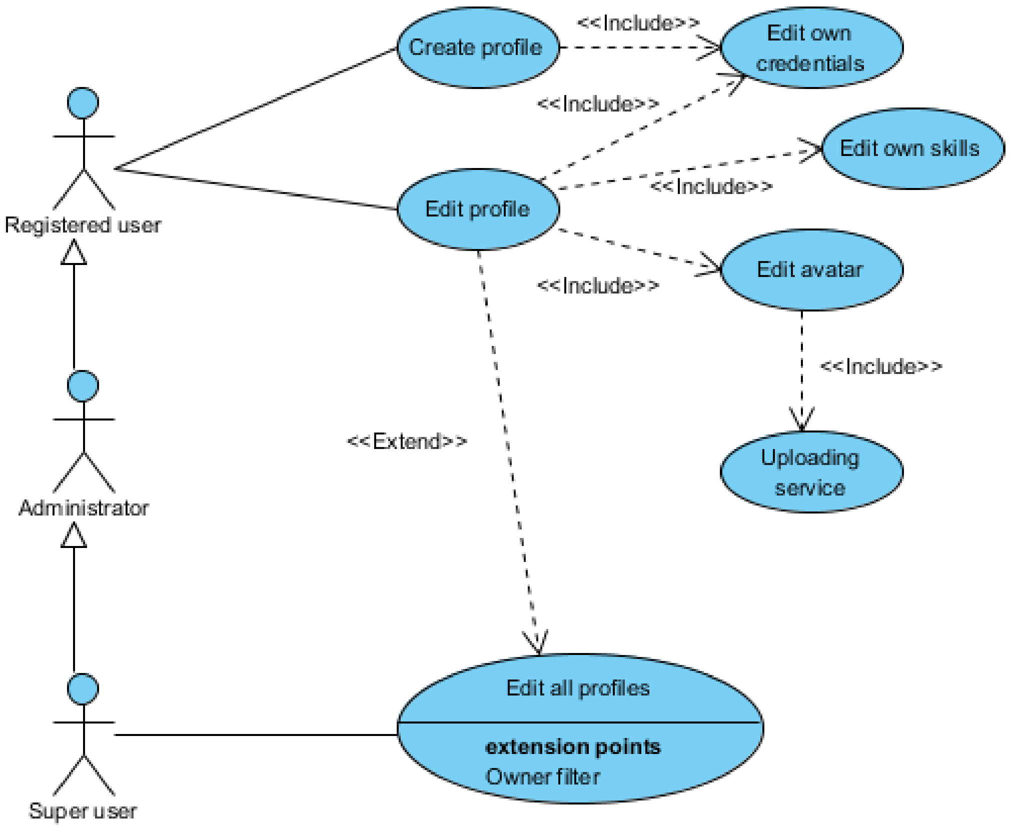
2.3.1.6. User lookup

Provides the capability of looking for other users' data.

2.3.1.7. E-mail service

Allows the system to send e-mails on the behalf of the website's administrators.

2.3.2. Personal profile



The personal profile function group contains function for creating and editing profiles, also including modification of credentials, skill sets and personal avatars.

2.3.2.1. Create profile

Create the user profile if he doesn't own one. Creation only requires the user to enter his/her basic credentials (username and password). Every other field is set to a default value. Also handles exceptions in the creation process.

2.3.2.2. Edit profile

Allows a registered user to edit informations that compose his/her profile. This includes personal description, birthday, gender, personal website and location, plus the special use cases listed in the following paragraph. All the informations that this functionality handles are totally optional.

2.3.2.3. Edit all profiles

A function reserved to super users, who have access to any profile. Every information can be modified, provided that it does not comply to the application's terms of usage. For this reason, any modification is registered in a shared log.

2.3.2.4. Edit own credentials

Each registered user can modify his/her access credentials (e-mail or password). This step requires him/her to know the account original password. Username cannot be changed.

2.3.2.5. Edit own skills

Allows the addition and deletion of skills, chosen from a predefined set.

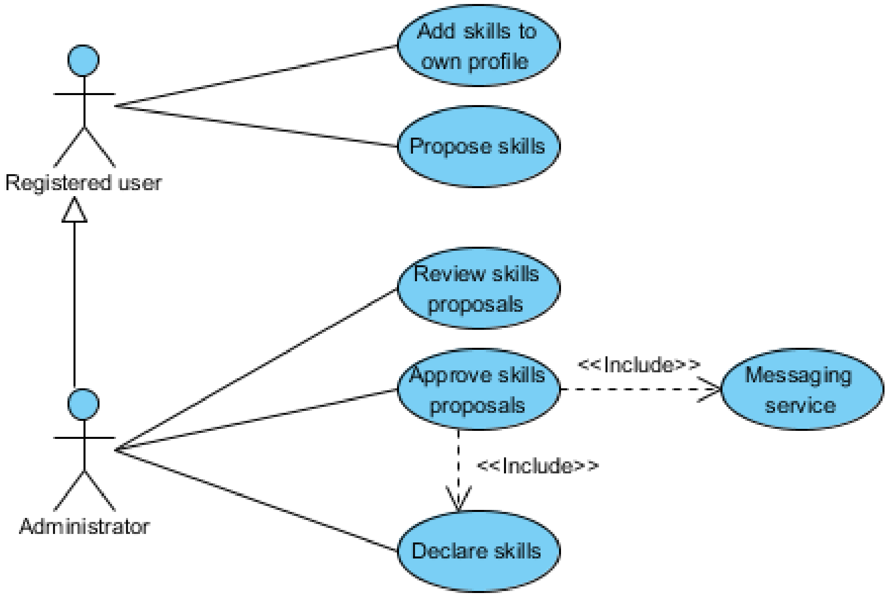
2.3.2.6. Edit avatar

A registered user can have only one avatar. It can be modified at will, with some dimension and size restrictions. It cannot contain more than G-rated images, otherwise it will be removed by super users. It can be chosen among a set of default avatars, fetched from an external source or uploaded.

2.3.2.7. Uploading service

Used by the latter use case, it allows the upload of files up to a given size and limited to a small subset of mime types.

2.3.3. Skills management



This function group manage skills and skill sets. Controls the addition of skills to a registered user's profile and skill proposals. Allows administrators to handle this proposals and to declare new skills.

2.3.3.1. Add skills to own profile

Related to personal profile's "Edit own skills", this function checks whether the user can add a given skill to his/her profile and modifies the profile accordingly.

2.3.3.2. Propose skills

Handles skill proposals creation. A registered user can propose a skill if it doesn't exist yet in the database. Existence checking is performed case-insensitively, trimming each repeated space characters and obviously ignoring dangerous strings. Skills proposals are limited in number within a given time period (say, no more than X proposals per week). An administrator cannot make skills proposals.

2.3.3.3. Review skills proposals

Allows administrators to view new skills proposals. If an administrator logs in, he/she will be instantly notified of new waiting proposals.

2.3.3.4. Approve skills proposals

After an administrator has been notified of new proposal, he/she can decide whether to accept or decline them. If accepted, skills are officially "declared", i.e. added to the database. In addition, the user who made the request will be notified of the approval and his/her skill set will be updated with the new approved skills. If declined, the user will be notified anyway.

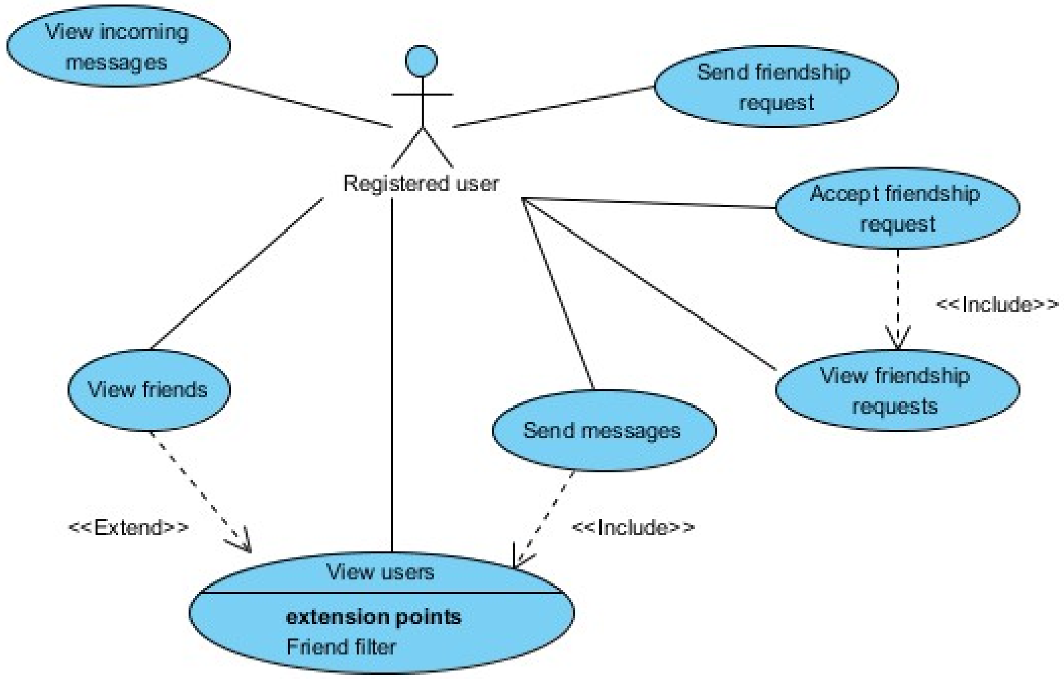
2.3.3.5. Declare skills

Allows administrators to declare new abilities. This function performs the usual security checks.

2.3.3.6. Messaging service

Borrowed from the Social function group, it contains those use cases which define the instant messaging service. Allows sending and receiving short messages on own profile.

2.3.4. Social



This function group provides social capabilities. Among these, it allows sending and receiving messages, consulting the userboard and friendboard and send or accept friendship requests.

2.3.4.1. View users
Lists all the system's users and allows searching by name.

2.3.4.2. View friends
Filters the users who are friends of the current registered user and displays only those.

2.3.4.3. Send friendship request
Sends a friendship request to an user. The targeted user must not be in the friend list of the caller. In addition, the system won't allow sending messages to users who already have a pending friendship request.

2.3.4.4. View friendship requests
Displays all the incoming friendship requests, along with time, hour and name of the requesting user. Unattended requests that are older than a given timeout will be deleted from the system.

2.3.4.5. Accept friendship requests
A request can be accepted or declined. In the case it is accepted, both the involved users will have their friend list updated and the request will be deleted from the pending list. A notification will be sent to the requesting user. Otherwise, it can be declined. The situation is analogous to the previous one, with the exception that those two users will not become friends.

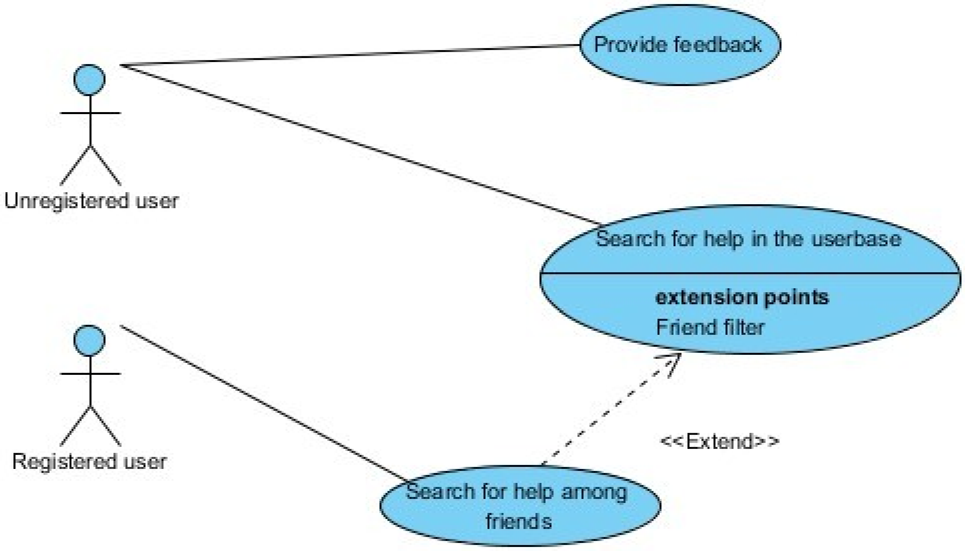
2.3.4.6. Send messages

A registered user can send private messages to any other registered user. Messages cannot be empty and will passed through a basic filter for preventing spam.

2.3.4.7. View incoming messages

Allows a registered user to view the list of incoming messages, along with sender name and hour of dispatch.

2.3.5. Help research



Controls the matchmaking system and provides means to give feedbacks.

2.3.5.1. Search for help in the userbase

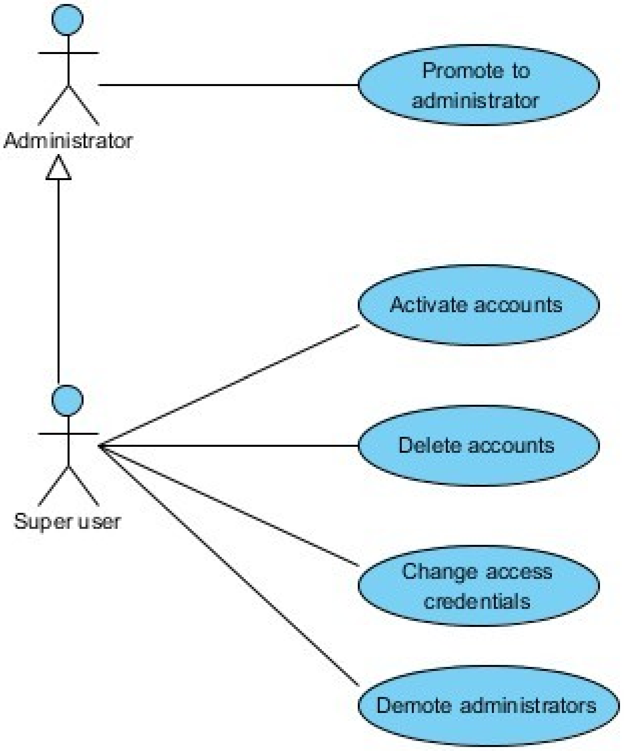
Any user can search for help. The seeker is required to specify at least a skill that the person sought must own in order to begin the research. The engine will provide a list of users who might be able to fill out the help request. In addition, results can be filtered by proficiency. If the user is not logged in or registered, he/she will be provided with the ability to send a message to the target and his/her response will be notified back via e-mail.

2.3.5.2. Search for help among friends

It is a restriction of the latter functionality. Results will also be filtered by membership to friend list.

2.3.5.3. Provide feedback
Unregistered users can provide feedback by filling out a form that requires e-mail and captcha confirmation. Registered user can give feedbacks with no restrictions, but any registered user can only provide one feedback per skill for each other user (for example: A says that B is very good at cooking, but average in repairing TVs).

2.3.6. General administration



Handles account and privileges administration.

2.3.6.1. Promote to administrator
Any administrator can promote registered user to administrators. An user cannot be promoted if he/she has been registered to the website for less than X weeks. In addition, an administrator cannot promote if he/she has been acting as administrator for less than X weeks. This security measures aim to prevent privileges abuse.

2.3.6.2. Demote administrators
Super users can demote administrators to normal registered users in the case they did not respect the code of conduct. A demoted administrator cannot be promoted anymore.

2.3.6.3. Activate accounts
Super users can activate accounts that are still in the confirmation process. This power users can avoid that users get stuck in the verification phase by broken links, e-mail filters and such.

2.3.6.4. Delete accounts

An account is yield for deletion is its owner breaks the terms of use of the service or behaves in a way that violates rules or netiquette. An account is not immediately deleted but there it is firstly deactivated for a week and then permanently deleted. In the deactivation state, its owner can log in but cannot send messages, search for help or send friendship requests. If the owner is an administrator, he/she cannot use his/her powers. A notification is sent via e-mail on the first day of deactivation and a day before deletion.

2.3.6.5. Change access credentials

Allows modification of passwords and/or e-mail for users that encounter problems in the account retrieval process.

Scenarios

Scenarios are written in the gherkin formal syntax.

3.1 Administrator Capabilities

Handles account and privileges administration.

3.1.1 Promote To Administrator Successfully

```
Given i am an administrator
    And foo is an user
    When i click on the "Promote foo To Administrator" button
Then foo should be an administrator
    And the system should send foo a notification alert "You have been
    promoted  to administrator"
```

3.1.2 Cannot Promote To Administrator Too Young Users

```
Given i am an administrator
    And foo is an user
    When i click on the "Promote foo To Administer" button
    But foo is too young
Then the system should send me a notification alert "foo is too young; you
    can't promote them to administrator yet"
```

3.1.3 Cannot Promote To Administrator Demoted Users

```
Given i am an administrator
    And foo is an user
    When i click on the "Promote foo To Administer" button
    But foo has been demoted
Then the system should send me an error alert "foo has been demoted; nobody
    can promote him"
```

3.1.4 Demote Administrator

```
Given i am a super user
    And foo is an administrator
    When i click on the "Demote foo" button
    And i confirm on the messagebox
Then foo should not be an administrator anymore
    And foo should be demoted
    And the system should send me a notification alert "foo has been demoted"
    And the system should send foo a notification alert "You have been
    demoted"
```

3.1.5 Force Account Activation

```
Given i am a super user
    And foo has registered an account
    But foo's account has not been activated
    When i click on the "Activate foo's account" button
    And i confirm on the messagebox
Then foo's account should be activated
    And the system should send me a notification alert "foo's account has
    been activated"
    And the system should send foo a welcome e-mail
```

3.1.6 Super User Deletes Account

```
Given i am a super user
    And foo is not a super user
    When i click on the "Delete foo's account" button
    And i confirm on the messagebox
Then foo's account should not exist
    And the system should send an "Account Deleted" e-mail to foo
```

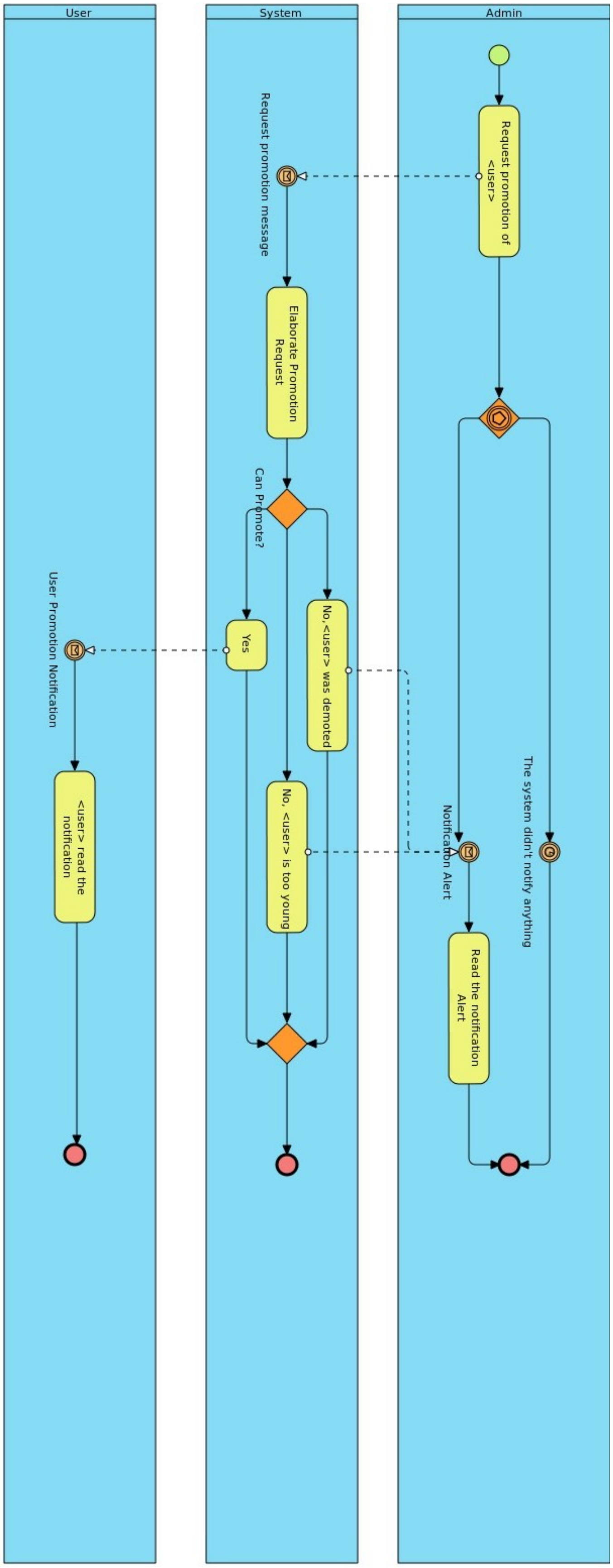
3.1.7 Force User Password Change

```
Given i am a super user
    And foo is an user
    And i want to force foo's password change
    When i fill out the new password form with dummy password
    And i click on the "Force password change" button
Then foo's passoword should be dummy
    And the system should send me a notification alert "foo's password
    changed"
```

3.1.8 Force Username Change

```
Given i am a super user
    And foo is an user
    And bar is foo's username
    And dummy username is not taken
    When i want to change foo's username
    And i fill out the new username form with dummy username
    And i click on the "Force username update" button
Then foo's username should be dummy
    And the system should send me a notification alert "foo's name changed to
    dummy"
```

* In the following image: Promotion of an user


User
System
Admin
Request promotion of <user>
Request promotion message
Elaborate Promotion Request
Can Promote?
Yes
No,<user> was demoted
No, <user> is too young
User Promotion Notification
<user> read the notification
Notification Alert
The system didn't notify anything
Read the notification Alert

3.2 Authentication

The authentication group provides functionalities for managing accounts. It supports creation and confirmation of new accounts, and provides access to existing ones.

3.2.1 Successful Login

```
Given i am not logged in
    When i insert my e-mail and my password in the login form
    And i press the login button
Then i should be logged in
```

3.2.2 Unsuccessful Login With Wrong Password

```
Given i am not logged in
    When i insert my e-mail and my password in the login form
    But my password was wrong
    And i press the login button
Then i should be not logged in
    And the system should send an error alert "Wrong Password Entered"
```

3.2.3 Unsuccessful Login With Wrong User

```
Given i am not logged in
    When i insert my e-mail and my password login form
    But my e-mail was wrong
    And i press the login button
Then i should not be logged in
    And the system should send an error alert "Wrong e-mail Entered"
```

3.2.4: Registration

```
Given i am not logged in
    When i insert my e-mail and my password in the registration form
    And i press the register button
Then i should receive a confirmation email
    And the system should ask me to check my inbox
```

3.2.5 Registration Finalization

```
Given i received a confirmation email
    When i press the finalization link
Then my account should be created
    And i should be logged in
```

3.2.6 Registration But User Alredy Exists

```
Given i am not logged in
    When i insert my e-mail and my password in the registration form
    And i press the register button
    But my e-mail is already taken
Then the system should send an error alert "e-mail already taken"
```

3.2.7 Retrieve Password

```
Given i am not logged in
    When i insert my e-mail in the password retrieval form
    And i press the password retrieval button
Then i should receive a password retrieval e-mail
    And the system should ask me to check my inbox
```

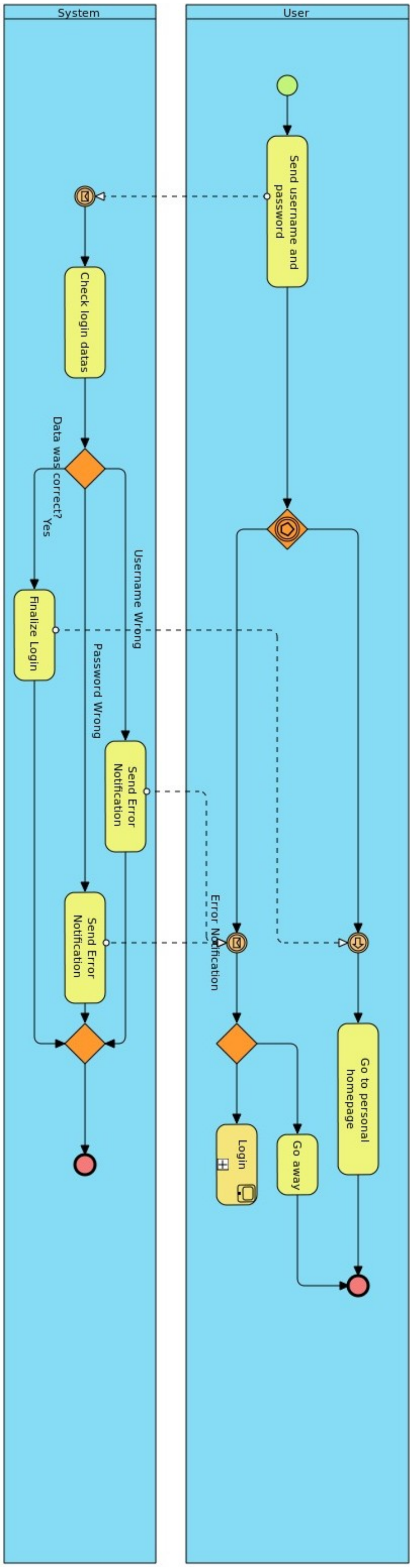
3.2.8 Retrieve Password Finalization

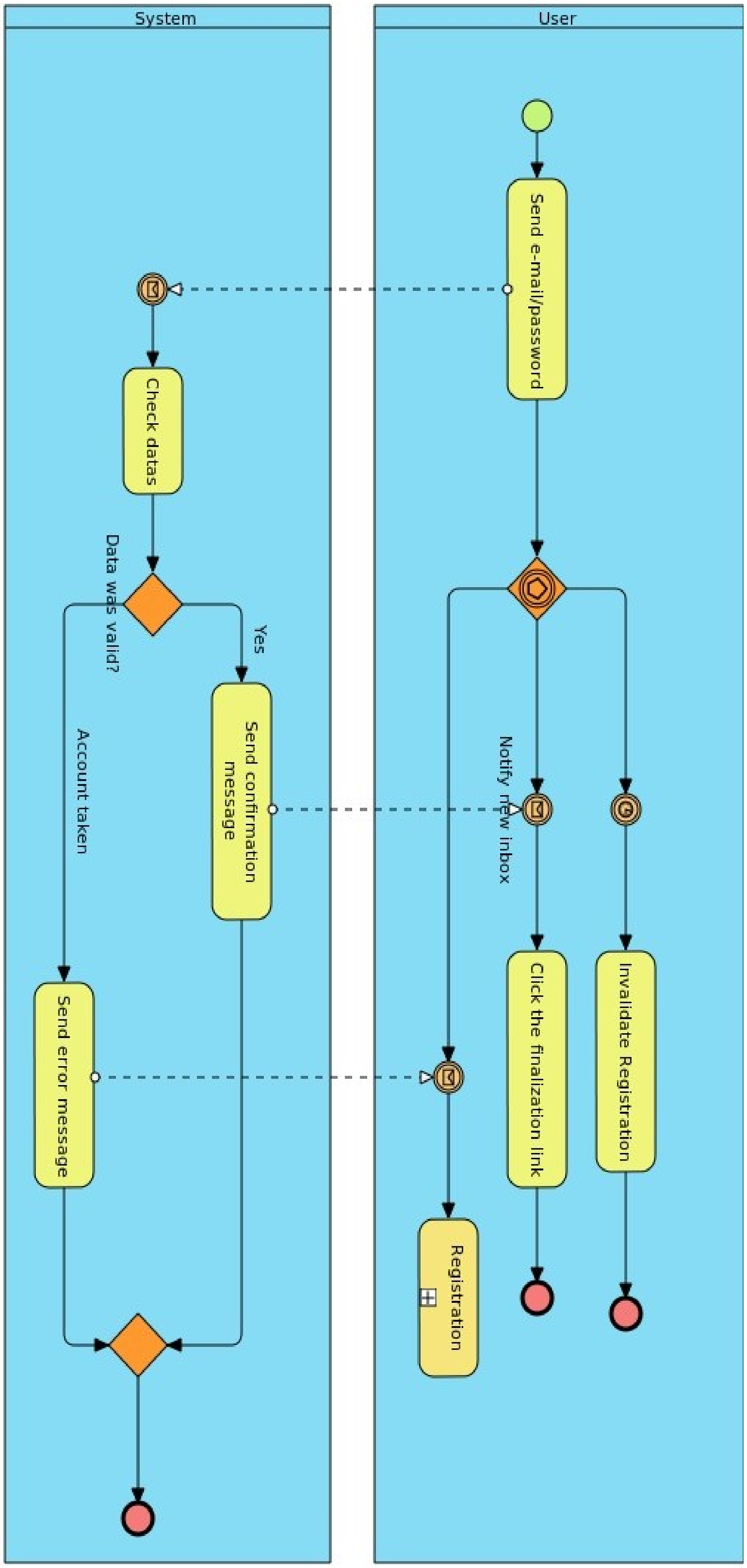
```
Given i receive a password retrieval email
    And i press the password retrieval link
    When i insert my password in the new password form
    And i press the new password button
Then my password should be updated
    And i should be logged in
```

3.2.9 Logout

```
Given i am logged in
    When i press on the logout button
Then i should not be logged in
    And the system should send me a notification alert "Successfully logged
    out"
```

* In the following images: Registration and Login


System
User
Send username and password
Check login datas
Data was correct?
Yes
Username Wrong
Password Wrong
Finalize Login
Send Error Notification
Send Error Notification
Error Notification
Login
Go away
Go to personal homepage


System
User
Send e-mail/password
Check datas
Data was valid?
Yes
Account taken
Send confirmation message
Send error message
Notify new inbox
Click the finalization link
Invalidate Registration
Registration

3.3 Lookup

Provides the capability of looking for other users' data. (PRE: There is a set of users in the database with some skills)

3.3.1 Search Other Users

```
When i fill out the search bar form with "foo"
    And i press the search button
Then i should be view search results
    And the results should contain users compatible with "foo"
```

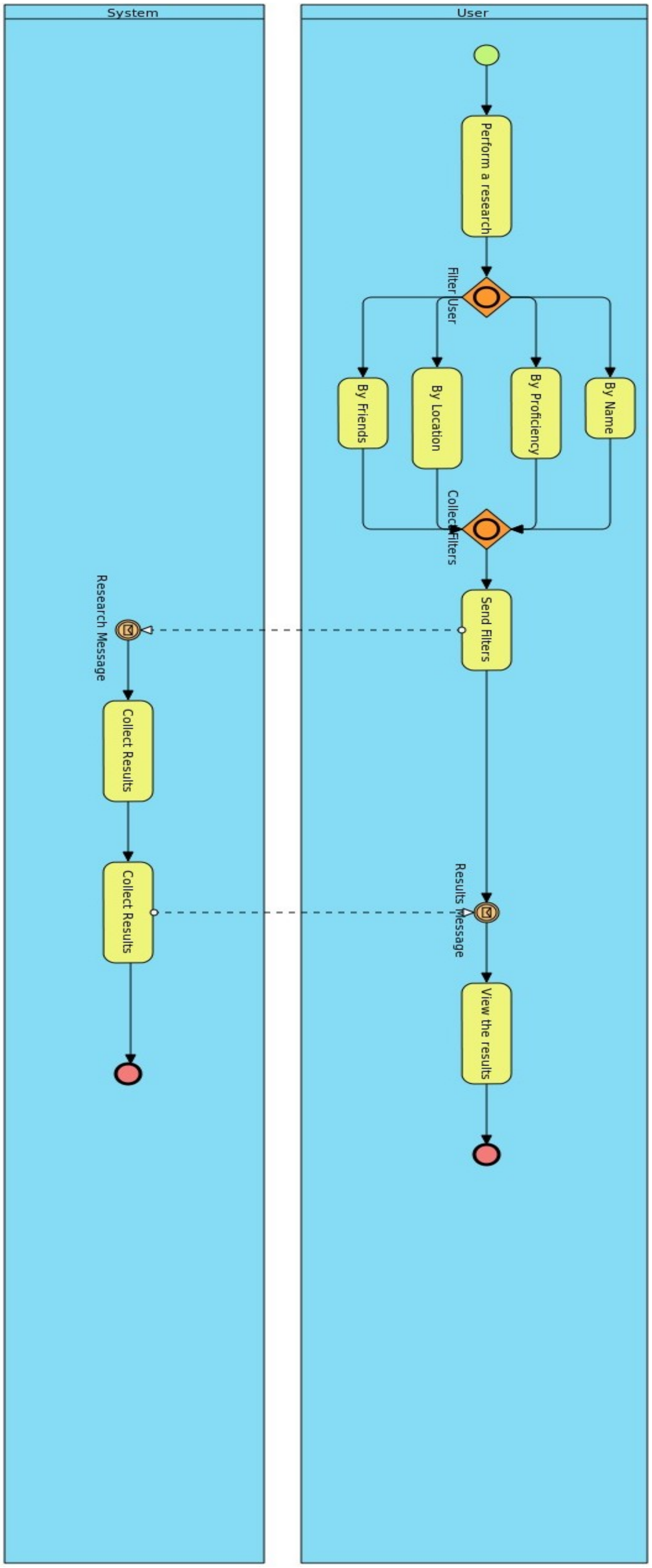
3.3.3 Filter Users By Proficiency

```
Given i have performed a research
    When i fill the proficiency filter form with k stars in skill foo
    And i press the search button
Then i should see only result users with at least k stars in skill foo
```

3.3.4 Filter Users By Friends

```
Given i have performed a research
    When i check the "only friends" checkbox
    And i press the search button
Then i should see only my friends in the results
```

* In the following image: help research


System
User
Perform a research
Filter User
By Friends
By Location
By Proficiency
By Name
Collect Filters
Send Filters
Research Message
Collect Results
Collect Results
Results Message
View the results

3.4 Mail Service

Allows the system to send e-mails on the behalf of the website’s administrators.

3.4.1 Mail Doesn't Exists

```
Given the <email> account doesn't exist
    When i send a message to <email>
    And the system receive a <email> not found
Then the <email> should be blacklisted
```

3.4.2 Registration Timeout Triggered

```
Given i send a registration email to <email>
    When the timeout is triggered
Then the <email> is removed from the reserved email list
```

3.5 Message Service

Provides a causal messaging service environment (PRE: Given i am logged in)

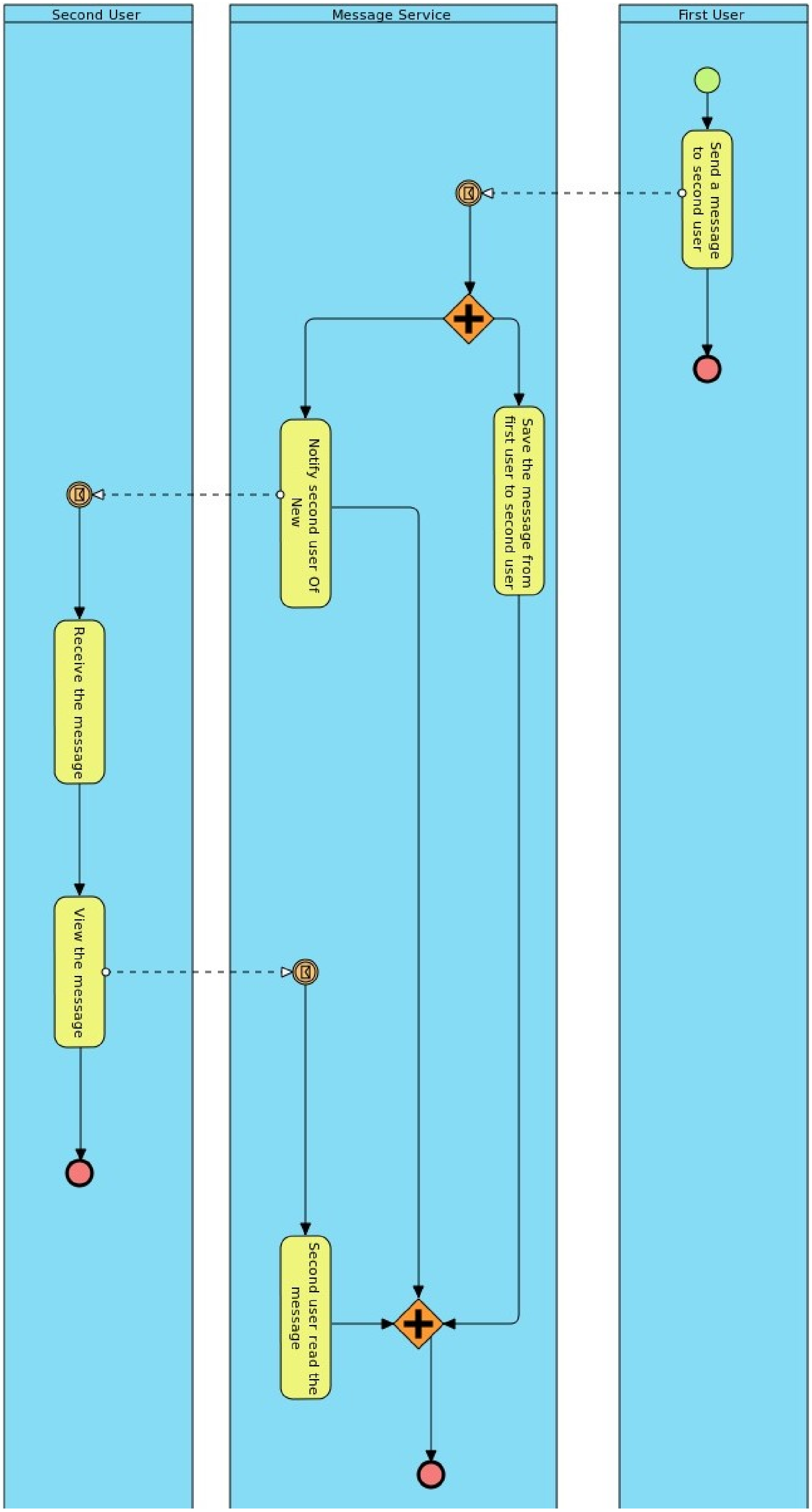
3.5.1 Send Message

```
Given i send a message "foo" to user bar
Then bar should have an inbox message "foo" from me
    And the system should notify bar of new inbox messages
```

3.5.2 View Messages

```
Given bar sends a message "foo" to me
    And the system notify me of new inbox messages
When i press on the inbox link
    Then i should view message "foo" from bar
```

* In the follwing image: Send a Message


Second User
Message Service
First User
Send a message to second user
Notify second user Of New
Save the message from first user to second user
Receive the message
View the message
Second user read the message

3.6 Profile Management

The personal profile function group contains functions for creating and editing profiles, also including modification of credentials, skill sets and personal avatars. (PRE: Given i am logged in)

3.6.1 Redirection To Profile Editing

```
    Given i logged in for the first time
    Then i should be redirected to the profile editing page
```

3.6.2 Edit Profile

```
Given i choose to edit my profile
    When i update my profile data
Then users should view my new profile data
      And the system should send me a notification alert "Profile Edited
Successfully"
```

3.6.3 Edit Profile, With Wrong Data

```
Given i choose to edit my profile
    When i update my profile data
    But my profile <type> was not valid
Then users should not view my new profile data
    And the system should send me an error notification "<error>"
```

3.6.4 Edit Other User Profile

```
Given I am a super user
    And i choose to update foo's profile
    When i update foo's profile data
Then users should view foo's new profile data
    And the system should send me a notification alert "Foo's Profile Edited
    Successful"
```

3.6.5 Edit Other's Profile, With Wrong Data

```
Given i am a super user
    And i choose to update foo's profile
    When i update foo's profile datas
    But foo's profile <type> was not valid
Then users should not view foo's new profile data
    And the system should send me an error notification "<error>"
```

3.6.6 Upload A File (Avatar) Successfully

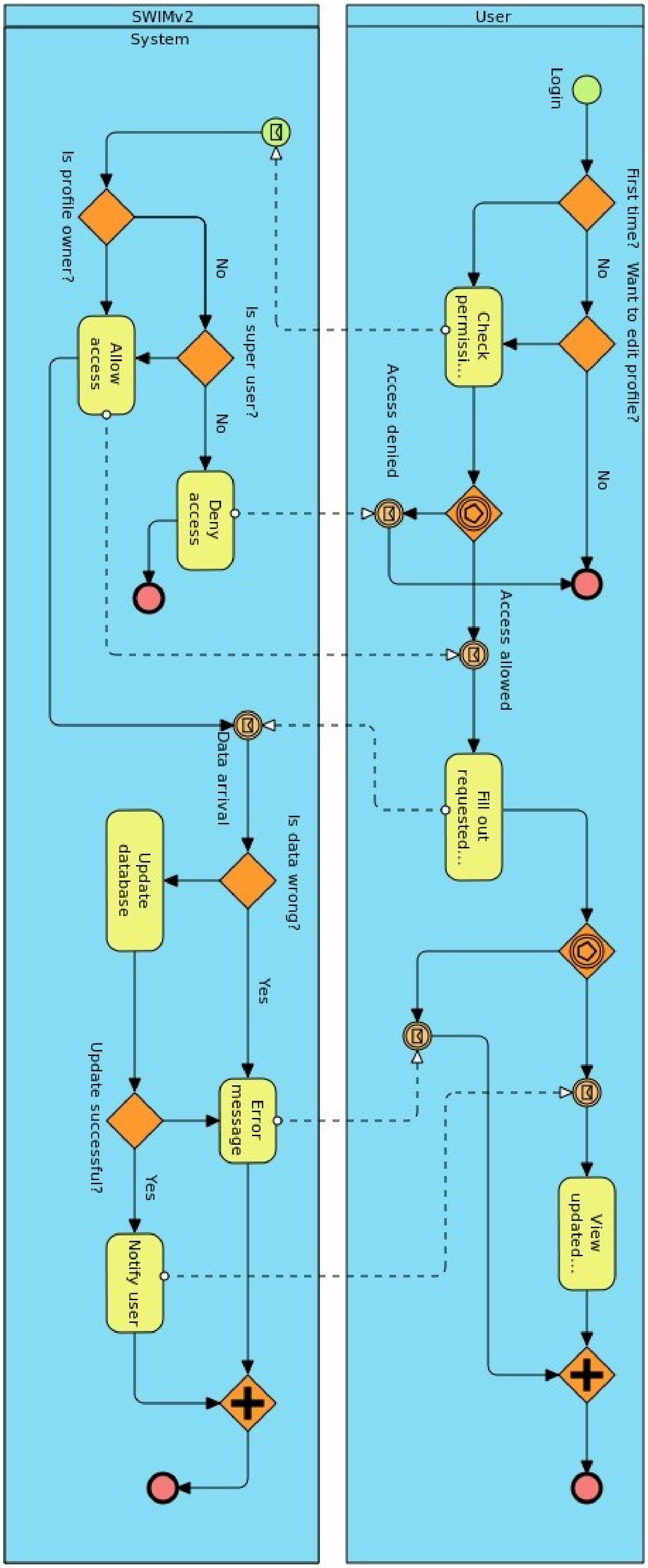
```
Given i fill the upload form with <filename> with myme <mymetype> and
    dimension <dimension>
    When i click on the upload button
```

```
Then the system should send me a notification alert "File <filename>
    successfully uploaded"
    And i should receive a link to the uploaded file
```

3.6.7 Upload A File Unsuccessfully

```
Given i fill the upload form with <filename> with myme <mymetype> and
    dimension <dimension>
    When i click on the upload button
Then the system should send me an error notification "File <filename> not
    uploaded. <error>"
```

* In the following image: Edit profile


SWIMv2
System
User
Login
First time?
No
Want to edit profile?
Check permissi...
No
Access denied
Access allowed
Fill out requested...
View updated...
Is profile owner?
No
Is super user?
Allow access
No
Deny access
Data arrival
Is data wrong?
Update database
Yes
Error message
Update successful?
Yes
Notify user

3.7 Skill Manager

This function group manages skills and skill sets. Controls the addition of skills to a registered user's profile and skill proposals. Allows administrators to handle these proposals and to declare new skills.

3.7.1 Propose A New Skill

```
Given i am an user
    And i am not an administrator
    When i fill out the new skill's form with dummy
    And i press on the submit button
Then dummy skill should be added to the skills proposals queue
    And the system should send me a notification alert "Skill proposal to be
    moderated"
```

3.7.2 Propose A New Skill When You Exceed Your Daily Quota

```
Given i am a user
    And i am not an administrator
    When i fill out the new skill's form with dummy
    And i press on the submit button
    But i exceeded my Daily Skills Quota
Then the system should send me a warning alert "You exceeds your daily skills
    quota. Propose others tomorrow."
```

3.7.3 Approve Skills

```
Given i am an administrator
    And foo proposed dummy skill
    When i press on dummy skill "Approve" button
Then dummy skill should be inserted in the skill list
    And dummy skill should be removed from the approval queue
    And the system should send me a notification alert "Dummy skill approved"
    And the system should send foo an email with "Dummy skill approved"
    And foo should have a dummy skill
```

3.7.4 Remove Proposed Skills From Approval Queue

```
Given i am an administrator
    And foo proposed dummy skill
    When i press on dummy skill "Remove" button
Then dummy skill should be removed from the approval queue
    And the system should send me a notification alert "Dummy skill removed
    from approval queue"
    And the system should send foo an email with "Dummy skill not approved"
```

3.7.5 Declare New Skills

```
Given i am an administrator
    When i fill out the "Add new skill" form with dummy skill
    And i press on the submit button
Then dummy skill should be inserted in the skills list
    And the system should send me a notification alert "Dummy skill approved"
```

3.7.6 Provide Feedback On A User Skill (Registered User)

```
Given i am an user
    When i fill out the "Provide a feedback" form with k points for dummy
    skill for user foo
    And k is greater then 0
    And k is less then or equal to 5
    And i press the "Send feedback" button
Then foo should have a new feedback regarding their dummy skill with k points
    And the system should notify foo of new feedbacks
```

3.7.7 Provide Feedback On A User Skill (Unregistered User)

```
Given i am not logged in
    When i fill out the "Provide a feedback" form with k points for dummy
    skill for user foo
    And i press the "Send feedback" button
Then the system should send me a notififcation alert "Login to review this
    skill <LOGIN LINK>"
```

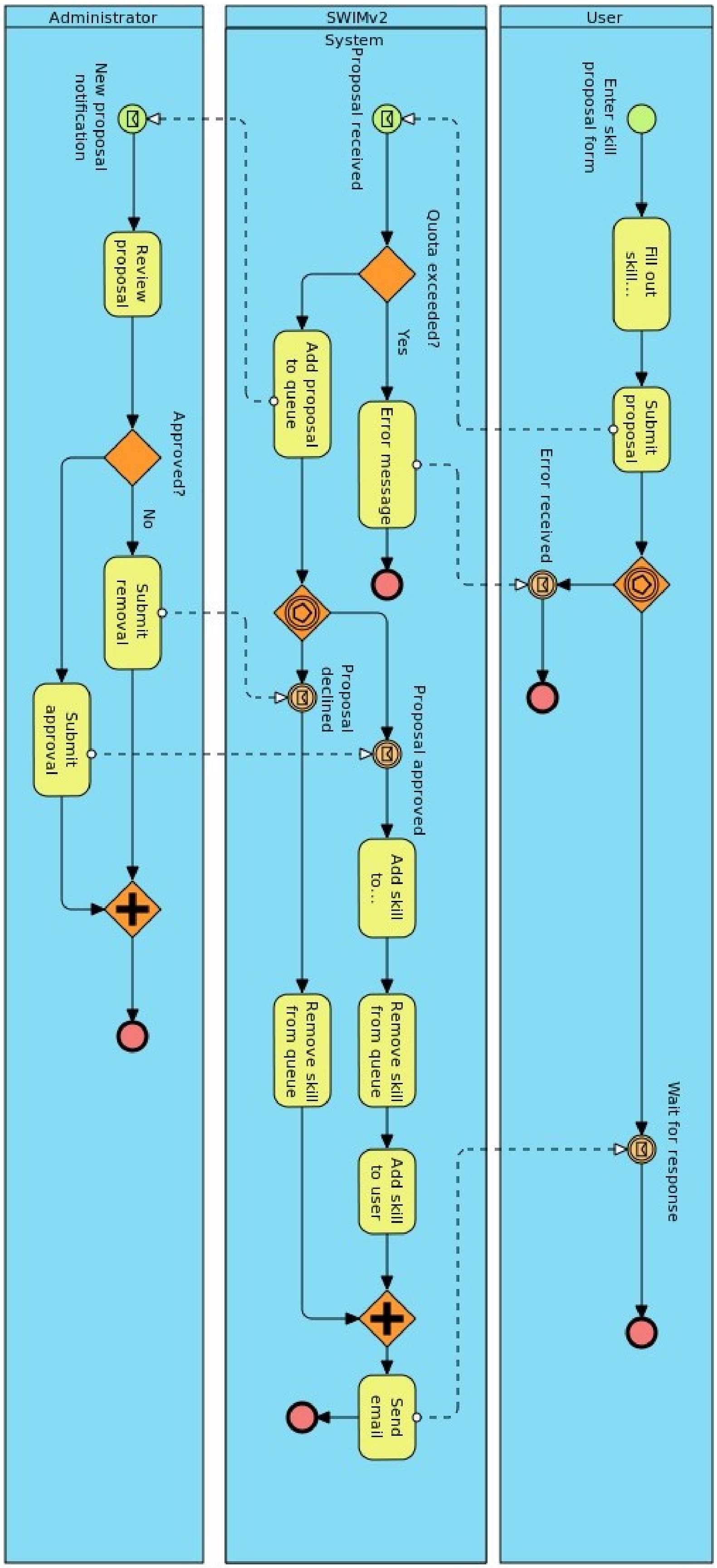
3.7.8 Add A Skill

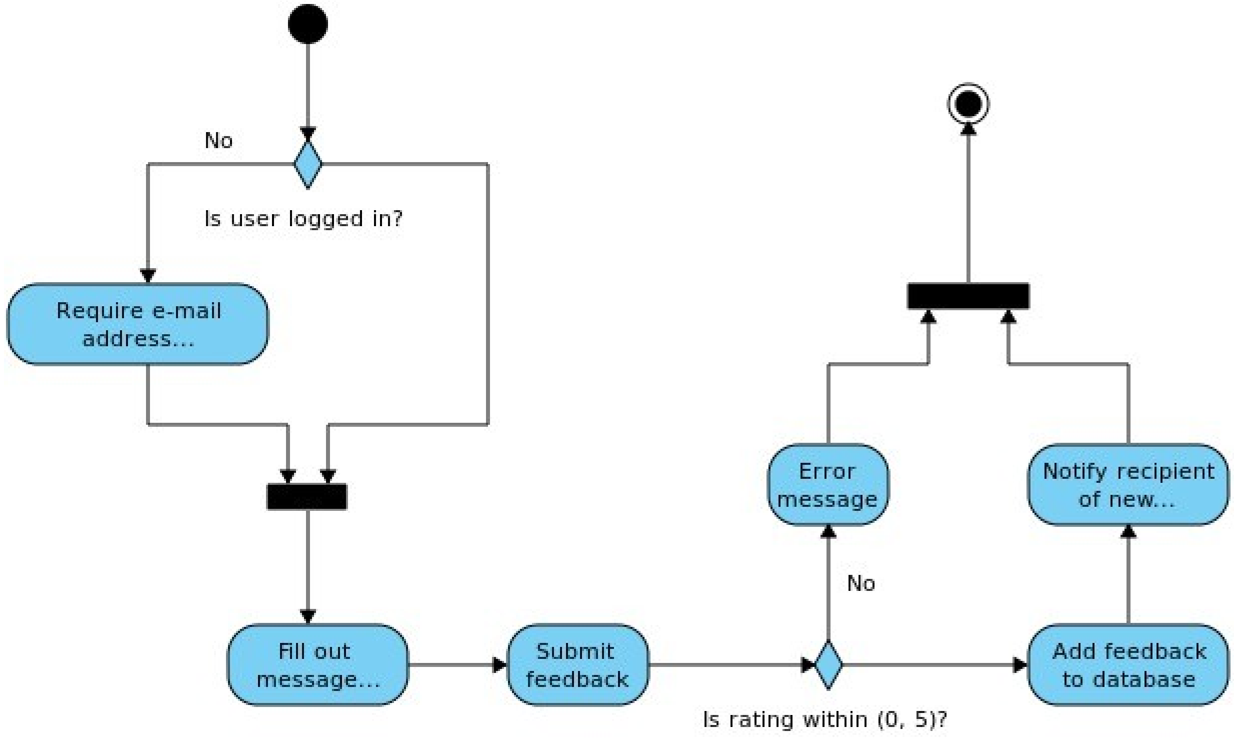
```
Given i am a user
    When i select foo skill from the ones available
    And i click on the add button
Then i should have the foo skill
    And the system should send me a notification alert "Skill foo added"
```

3.7.9 Remove A Skill

```
Given i am a user
    When i click on dummy skills remove button
Then i should no longer have the dummy skill
    And the system should send me a notification alert "Skill foo removed"
```

* In the following image: Skill Proposal and Approval


Administrator
SWIMv2
System
User
Enter skill proposal form
Fill out skill...
Submit proposal
Error received
Wait for response
Proposal received
Quota exceeded?
Yes
Add proposal to queue
Error message
Proposal declined
Proposal approved
Add skill to...
Remove skill from queue
Remove skill from queue
Add skill to user
Send email
New proposal notification
Review proposal
Approved?
No
Submit removal
Submit approval



3.8 Social Network

This function group provides social capabilities. Among these, it allows sending and receiving messages, consulting the user board and friend board and send or accept friendship requests. (PRE: Given i am logged in).

3.8.1 View User's Friends

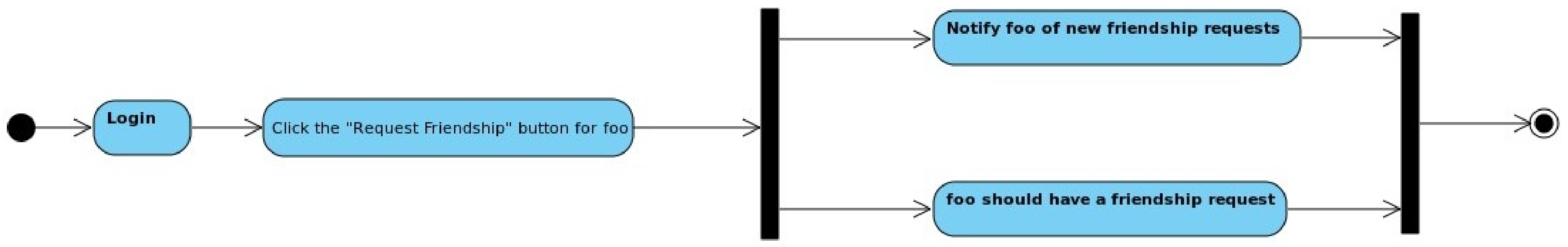
```
When i click the "View Friends" link
Then i should view my friends list
```

3.8.2 Send Friendship Request To A User

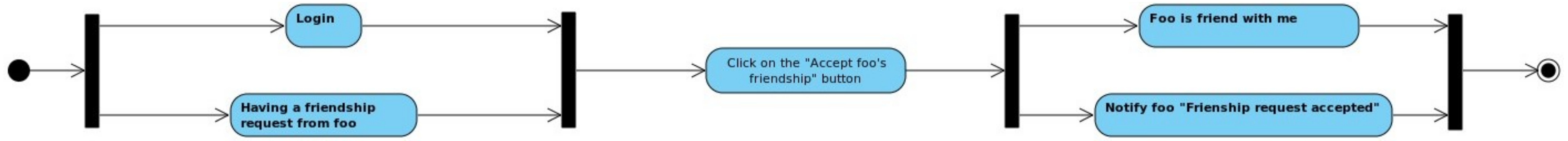
```
Given i am not friend with foo
    When i click the "Request Friendship" button for foo
Then foo should have a friendship request from me
    And the system should notify foo of new friendship requests
```

3.8.3 Accept Friendship Request

```
Given i have a friendship request from foo
    When i click on the "Accept foo's friendship" button
Then i should be friend with foo
    And foo should be friend with me
    And the system should send foo a notification alert "Friendship request
    accepted"
```



Login
Click the "Request Friendship" button for foo
Notify foo of new friendship requests
foo should have a friendship request



Login
Having a friendship request from foo
Click on the "Accept foo's friendship" button
Foo is friend with me
Notify foo "Frienship request accepted"

Alloy models

4.1. Static model

Here follows the static model of the project, written in Alloy formal syntax:

```
module SWIMv2

// Boolean data type
abstract sig Bool {}
one sig True extends Bool { }
one sig False extends Bool { }

// Concepts being used as signature properties
sig Email {}
sig Password {}
sig Name {}
sig Image {}
sig Text {}

// A skill only has a name, but it is not a string by itself
sig Skill {
    name: one Name,
    description: one Text,
    proposer: one User,
    approver: lone Administrator
}

one sig SkillDatabase {
    skills : set Skill
}

fact {
    // No two skills can have the same name
    no skill1, skill2 : Skill | skill1 != skill2 and skill1.name =
skill2.name

    // If two skill have different name then they need to have different
description
    all skill1, skill2: Skill | skill1.name != skill2.name implies
skill1.description != skill2.description
}
```

```
// An external visitor. We know no information about him
sig Visitor {}

// Registered user
sig User {
    // Profile
    email: one Email,
    username: one Name,
    password: one Password,
    avatar: lone Image,
    info: lone String,
    // Skills
    skills: set Skill,
    // Social
    friends: set User,
    // Permissions
    demoted: one Bool
}

// Administrator
sig Administrator extends User {}

// Super user
sig SuperUser extends Administrator { }

fact {
    // The same email cannot be used by more than 1 user
    no user1, user2 : User | user1 != user2 and user1.email = user2.email

    // An user cannot be friend with himself
    all user : User | !(user in user.friends)

    // The friendship relation is symmetric
    all user1, user2 : User | (user1 in user2.friends) => (user2 in 
user1.friends)

    // There not exist demoted administrators or super users
    no admin : Administrator | admin.demoted in True

    // Users' skills must be within the database
    all user : User | one db : SkillDatabase | all skill : Skill |
        skill in user.skills => skill in db.skills
}
```

```
// Messages. For simplicity, we assume a message is sent only to one user
sig Message {
    sender: one User,
    recipient: one User,
    content: one Text
}

fact {
    // Users cannot send messages to themselves
    no message : Message | message.sender = message.recipient
}

abstract sig Rating {}
one sig OneStar, TwoStars, ThreeStars, FourStars, FiveStars extends Rating {}

// Feedback records
sig Feedback {
    author: one User,
    target: one User,
    rating: one Rating,
    skill: one Skill,
    comment: one Text
}

fact {
    // Users cannot provide feedback for themselves
    all feedback : Feedback | feedback.author != feedback.target

    // A feedback should be about a skill that the target owns
    all feedback : Feedback | feedback.skill in feedback.target.skills

    // There cannot exist more than 1 feedback on the same user about the
same skill from the same author
    no feed1, feed2 : Feedback |
        feed1 != feed2 and
        feed1.author = feed2.author and
        feed1.target = feed2.target and
        feed1.skill = feed2.skill

    // All skills in feedback must be approved
    all feedback: Feedback | #(feedback.skill.approver) = 1
}
```

4.2. Dynamic model

Alloy isn't thought to represent dynamic conditions and operations. However, it is possible to model also those situations. We have modeled two possible actions as a sample of Alloy proficiency: addition and remotion of skills and approval/proposal of new skills.

```
// Skills edit

pred addSkill[user, result : User, skill : Skill] {
    // Preconditions
    skill !in user.skills and #(skill.approver) = 1
    // Postconditions
    result.email = user.email
    result.username = user.username
    result.password = user.password
    result.avatar = user.avatar
    result.info = user.info
    result.friends = user.friends
    result.demoted = user.demoted

    result.skills = user.skills + skill
}

pred delSkill[user, result : User, skill : Skill] {
    skill in user.skills and #(skill.approver) = 1

    result.email = user.email
    result.username = user.username
    result.password = user.password
    result.avatar = user.avatar
    result.info = user.info
    result.friends = user.friends
    result.demoted = user.demoted

    result.skills = user.skills - skill
}

assert skillSetAdditionIntegrity {
    all user, temp : User, skill : Skill |
        addSkill[user, temp, skill] =>
        #(temp.skills) > #(user.skills) and skill in temp.skills
}
```

```
assert skillSetDeletionIntegrity {
    all user, temp : User, skill : Skill |
        delSkill[user, temp, skill] =>
        #(temp.skills) < #(user.skills) and !(skill in temp.skills)
}

assert skillSetIdempotency {
    all user, temp1, temp2 : User, skill : Skill |
        addSkill[user, temp1, skill] and delSkill[temp1, temp2, skill] =>
        user.skills = temp2.skills
}

check skillSetAdditionIntegrity for 3
check skillSetDeletionIntegrity for 3
check skillSetIdempotency for 3

// Skills proposals

pred proposeSkill[db, db' : SkillDatabase, user : User, skill : Skill] {
    // Preconditions
    skill.proposer = user and (skill.approver & User) = none and skill !in
db.skills
    // Postconditions
    db'.skills = db.skills + skill
}

pred approveSkill[db, db' : SkillDatabase, admin : Administrator, skill,
skill' : Skill] {
    skill in db.skills and #(skill.approver) = 0

    skill'.name = skill.name
    skill'.description = skill.description
    skill'.proposer = skill.proposer

    skill'.approver = admin
    db'.skills = (db.skills - skill) + skill'
}

assert skillProposal {
    all db, db' : SkillDatabase, user : User, skill : Skill |
        proposeSkill[db, db', user, skill] =>
        #(db'.skills) > #(db.skills) and skill in db.skills
}
```

```
assert skillApproval {
    all db, db' : SkillDatabase, admin : Administrator, skill, skill' : Skill
|
        approveSkill[db, db', admin, skill, skill'] =>
        skill' in db.skills and skill'.approver = admin
}

check skillProposal for 3
check skillApproval for 3
```

4.3. Graphic representation of static model

Running the predicate showSeriousSystem, we obtained the following model: see the xml instance file in the zip archive.

System details

5.1. Constraints

The system must be implemented obeying to specific constraints, which have been specified by the client in the original specification document.

5.1.1. Design constraints

The system must be implemented in Java using the J2EE platform and EJB technology.

5.1.2. Performance constraints

The system response time must be acceptable. Each page should load within a relatively small timeout. In addition, it must be able to handle many concurrent requests: the order of magnitude of the userbase is unknown; however, considering the type of product being developed and the internet average user's habits, we can estimate an initial community of hundreds of user. So, concurrency manager should handle easily at least ten concurrent requests.

5.1.3. Database constraints

The database must be relational. Transactions must support ACID properties, because every bit of data is important and should be accessible instantly after its modification (commit). We cannot use BASE databases.

5.1.4. Hardware constraints

Not specified.

5.1.5. Software constraints

Not specified.

5.1.6. Compliance to standards

Whenever possible, the code and organization of the system should obey to existing standards. This allows for an easier maintainability and quicker accessibility for new programmers.

5.2. Software properties

The produced software must offer a standard set of properties. Here follows a description of how this system complies to those properties.

5.2.1. Reliability

The system must assure that every data entered by users is persisted until the user decides to unsubscribe from the service. Integrity and durability are granted by the underlying DBMS.

5.2.2. Availability

The system should be available for as much time as possible, ideally 24/7 for every day of the year. However, our code has little control on this property. We can debug and test our web application to be reasonably sure that software faults are handled, addressed and corrected without compromising the stability of the other components, but we have to trust the underlying server technology too.

5.2.3. Security

User credentials are the only sensitive data in this domain: every other information provided by the user can be disclosed to the public in the range of our system. We can achieve security by using a security layer over the communication protocol and storing such data by means of hashing. In addition, we will provide a Java service that checks and sanitize user input, to prevent sql injection and xss attacks.

5.2.4. Maintainability

Standard compliance will aim future developers during software maintenance.

5.2.5. Portability

By using Java technologies, our product can be installed virtually anywhere, provided that the target operating system has a JVM (version >= 1.6) installed.

5.3. Technical requirements

5.3.1. Hardware

The computer which the system will be installed on needs to have the following requirements.

5.3.1.1. Memory

At least 2GB of working memory (RAM). In this early stage of development, we still don't know how much our bundled application will weight; however, based on previous estimations, we suggest to have at least 20 to 40 MB of free disk space. In addition, the database will need a lot of space to store application's data. Therefore, we suggest to reserve at least 10GB of free space on a persistent medium, preferably separated from the installation module.
Fast memory is highly recommended: prefer solid state disk to hard disk, for they have low latency and faster access times.

5.3.1.2. Network interface

The application needs to instantiate a web server, so a network interface is required.

5.3.1.3. Processor

EJB technology uses a virtual single thread and is not optimized for parallel computation, therefore the benefit of a multi-core processor will impact less on the overall performance. However, given the number of layers of abstraction, the faster, the better.

5.3.2. Software

The system needs several software modules in order to work properly.

5.3.2.1. Database management system

The latest version of MySql (community edition) will suffice to cover every aspect we need.

5.3.2.2. Network protocols

The application need to work on several network protocols, including:

- HTTP, on TCP port 80
- HTTPS, on TCP port 443
- FTP, on TCP port 21 and 22
- FTPS, on TCP port 23
- MySql protocol, on TCP and UDP port 3306

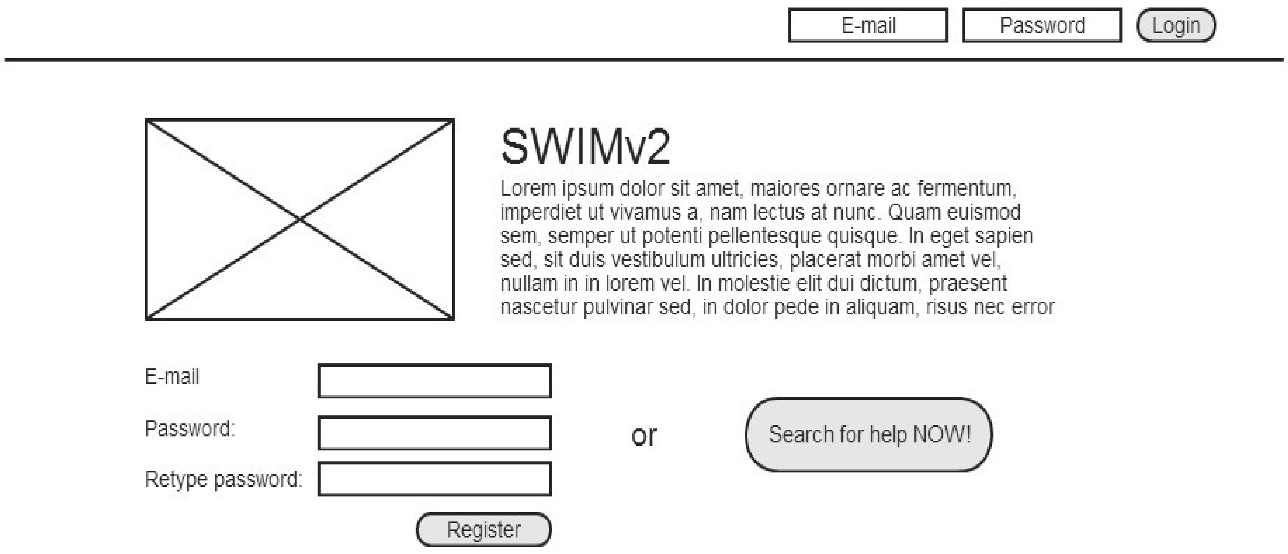
5.3.2.3. Java virtual machine

Any instance of JVM, from version 1.6 onward.

User interface

In this paragraph, we present some mockups for the user interface of the system. Details should not be considered literally, because there is ample space for changing. Some elements may be added in future revisions, while some may be removed as well. This section represents only a vague vision of the wanted look&feel.

6.1. Home page

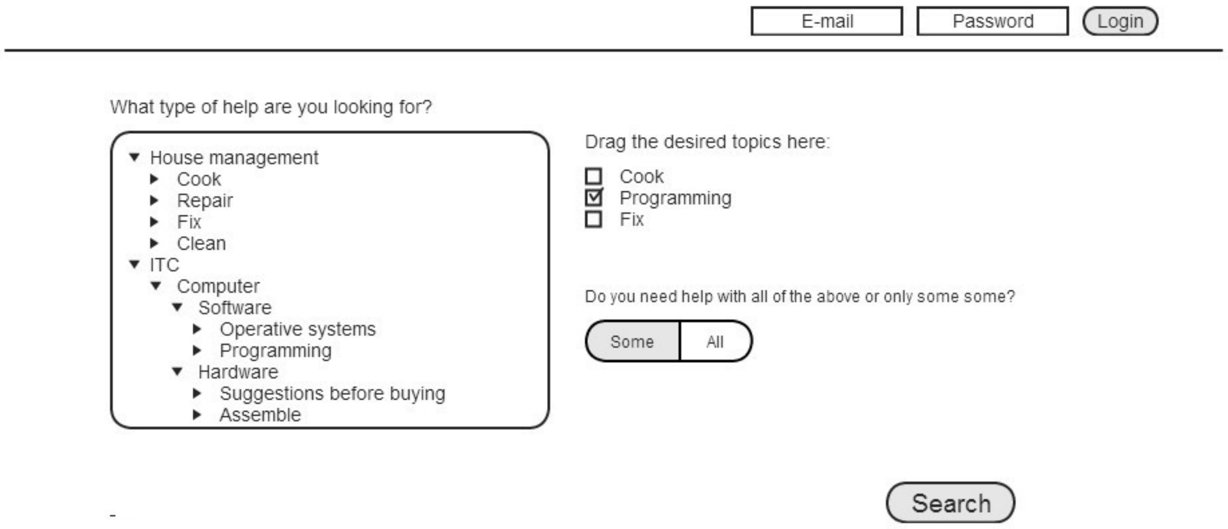


Lorem ipsum dolor sit amet, maiores ornare ac fermentum, imperdiet ut vivamus a, nam lectus at nunc. Quam euismod sem, semper ut potenti pellentesque quisque. In eget sapien sed, sit duis vestibulum ultricies, placerat morbi amet vel, nullam in in lorem vel. In molestie elit dui dictum, praesent nascetur pulvinar sed, in dolor pede in aliquam, risus nec error

The white rectangle on the left will host a large picture that depicts the site's functionality, while the paragraph on the right will contain a brief description of offered services. An external user can either login, register or go further and search for help as an anonymous user. The login panel is fixed on the upper right corner of the main control bar, which will fill almost entirely the upper space of any page. The registration panel is positioned under the main picture and is flanked by the bigger search button. We want visitors to know that they can exploit our services for free without necessary registration.

Although not represented in this mockup, there could be help links on the bottom of the page.

6.2 Research panel



What type of help are you looking for?

- House management
  - Cook
  - Repair
  - Fix
  - Clean
- ITC
  - Computer
    - Software
      - Operative systems
      - Programming
    - Hardware
      - Suggestions before buying
      - Assemble

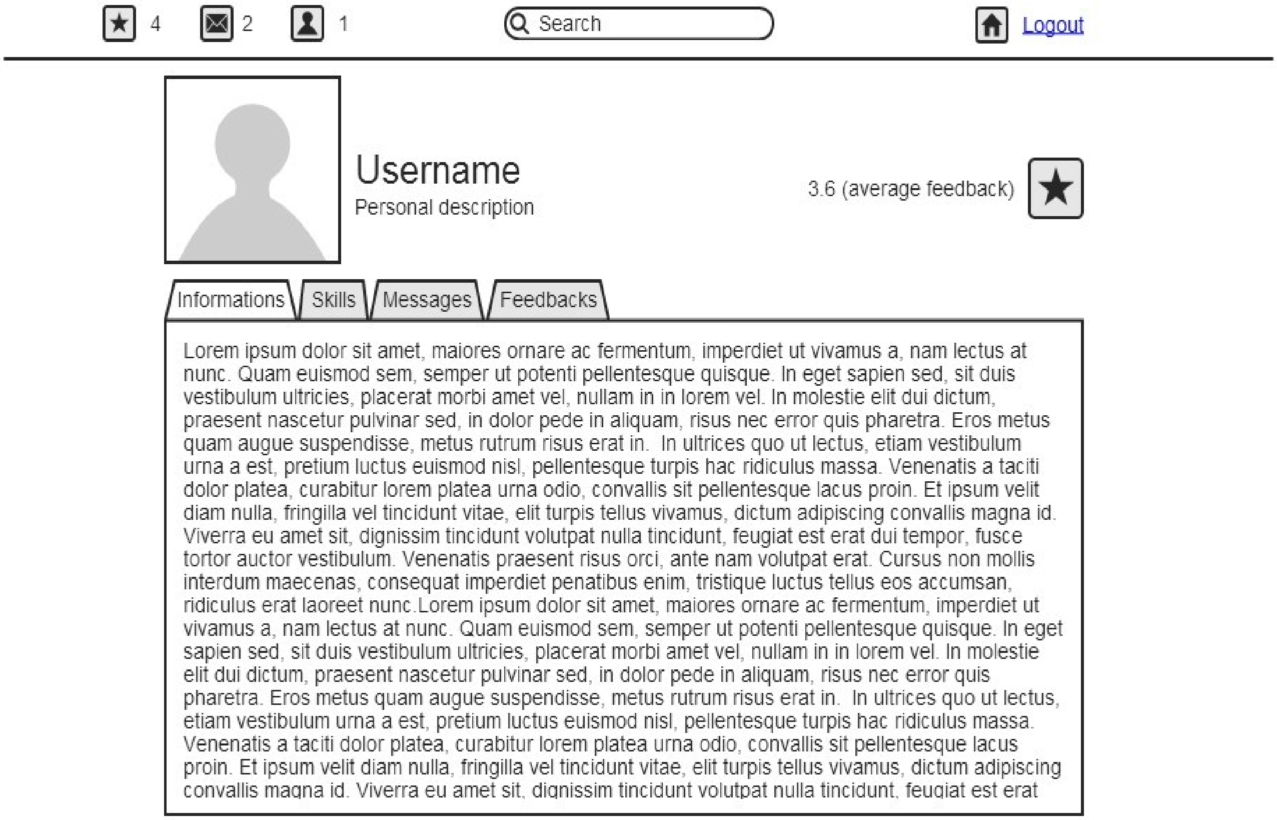
Drag the desired topics here:

□ Cook
☑ Programming
□ Fix

Do you need help with all of the above or only some some?

This screen shows how an external visitor should see the research page. The box that contains skills will not feature a tree view, because it is not required by the specifications and is rather hard to implement correctly. However, it will be possible to filter skills in a different way. Skills from this box can be dragged onto the right empty space and they will appear as checkboxes. The user can check or uncheck them (in the case he/she dragged out a wrong skill) and can decide whether to look for someone that is capable in every selected skill or only in some one.
Logged users will also be asked if they want to search through all the userbase or only among friends.

6.3. User profile



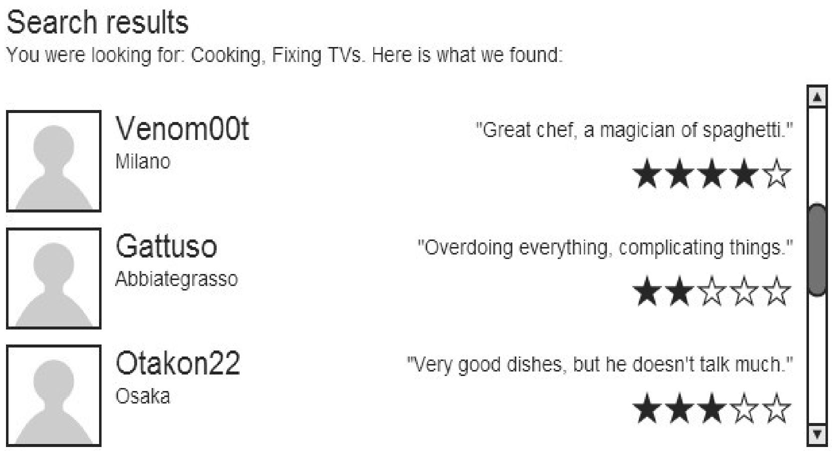
Lorem ipsum dolor sit amet, maiores ornare ac fermentum, imperdiet ut vivamus a, nam lectus at nunc. Quam euismod sem, semper ut potenti pellentesque quisque. In eget sapien sed, sit duis vestibulum ultricies, placerat morbi amet vel, nullam in in lorem vel. In molestie elit dui dictum, praesent nascetur pulvinar sed, in dolor pede in aliquam, risus nec error quis pharetra. Eros metus quam augue suspendisse, metus rutrum risus erat in. In ultrices quo ut lectus, etiam vestibulum urna a est, pretium luctus euismod nisl, pellentesque turpis hac ridiculus massa. Venenatis a taciti dolor platea, curabitur lorem platea urna odio, convallis sit pellentesque lacus proin. Et ipsum velit diam nulla, fringilla vel tincidunt vitae, elit turpis tellus vivamus, dictum adipiscing convallis magna id. Viverra eu amet sit, dignissim tincidunt volutpat nulla tincidunt, feugiat est erat dui tempor, fusce tortor auctor vestibulum. Venenatis praesent risus orci, ante nam volutpat erat. Cursus non mollis interdum maecenas, consequat imperdiet penatibus enim, tristique luctus tellus eos accumsan, ridiculus erat laoreet nunc.Lorem ipsum dolor sit amet, maiores ornare ac fermentum, imperdiet ut vivamus a, nam lectus at nunc. Quam euismod sem, semper ut potenti pellentesque quisque. In eget sapien sed, sit duis vestibulum ultricies, placerat morbi amet vel, nullam in in lorem vel. In molestie elit dui dictum, praesent nascetur pulvinar sed, in dolor pede in aliquam, risus nec error quis pharetra. Eros metus quam augue suspendisse, metus rutrum risus erat in. In ultrices quo ut lectus, etiam vestibulum urna a est, pretium luctus euismod nisl, pellentesque turpis hac ridiculus massa. Venenatis a taciti dolor platea, curabitur lorem platea urna odio, convallis sit pellentesque lacus proin. Et ipsum velit diam nulla, fringilla vel tincidunt vitae, elit turpis tellus vivamus, dictum adipiscing convallis magna id. Viverra eu amet sit, dignissim tincidunt volutpat nulla tincidunt, feugiat est erat

Depending on one’s security preferences, logged users and/or visitors will be able to see their profile. The profile contains a picture of the user, an username (i.e. a display name chosen by the user) and a tabbed control with various sections. Informations tab contains general information written by the user himself/herself. Skills hosts a list of all skills for this user, along with their proficiency measured in stars. Messages tab allows the owner of the profile to view incoming messages and every other user to the the owner private messages. Finally, in the feedback pane, users will be able to see the latest feedbacks for the profile owner.

In the upper control bar, there are several icons. From left to right, we have:

- New feedbacks: counts the number of new feedbacks received and allows to quickly see own feedback tab;
- Incoming messages: counts the number of new messages and allows to quickly see own message tab;
- Friendship requests: counts the number of incoming friendship requests and allows to approve / decline them.
- Search bar: type in the name of some skills to quickly jump to the search page;
- Home: shows a list of options for the account, among which: credentials change, profile edit and preferences.
- Logout: button to log out.

6.4. Search results



Search results

You were looking for: Cooking, Fixing TVs. Here is what we found:

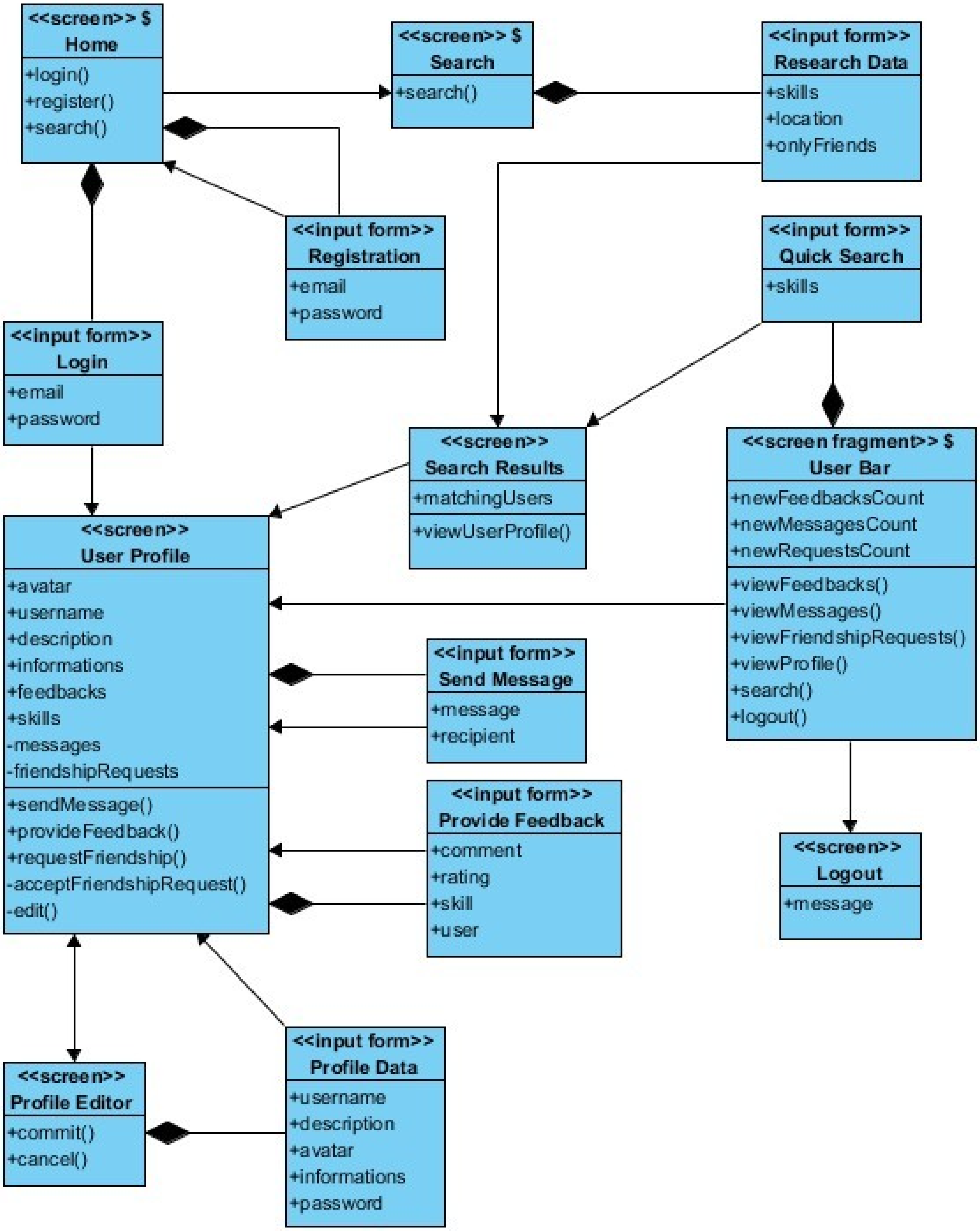
Venom00t
Milano
"Great chef, a magician of spaghetti."
★★★★☆

Gattuso
Abbiategrasso
"Overdoing everything, complicating things."
★★☆☆☆

Otakon22
Osaka
"Very good dishes, but he doesn't talk much."
★★★☆☆

In this panel, we overlooked the main control bar, since it has been already explained. The search results lists user results according to the required filters. It shows user names (which are links to their profiles), location, average feedback for target skills and a feedback chosen by some policy.

6.5. Interaction diagram



This diagram above borrows some symbols from UX and Class diagrams to build an intuitive interaction map of the system. Here is the description of some used formalisms:

6.5.1. <<screen>>

Represents a web page. Attributes of this stereotype convey the informations about the entities they refer to: such information can be atomic or composed (in case of lists or sets). Instead, operations (methods) visualize the user possible actions: there is a method for each outgoing link (ex.: “search()”) and one for each submit action of related forms (ex.: “login()”). In addition, methods can perform different tasks while remaining on the same page (ex.: “requestFriendship()”). Public members are visible by everyone, while private ones are accessible only by the profile owner.

6.5.2. <<input form>>

Represents a single web form. Attributes of this stereotypes indicate the fields needed to fill out the form. If no operation is specified, then the default submit() operation is assumed.

6.5.3. <<screen fragment>>

Represents a fragment of a page that is written separately and then included or imported by means of scripts. Elements of this stereotype are the same as in <<screen>>.

6.5.4. <<screen>> $

They are static screens, i.e. there is at least a link to them in every other page of the site.

6.5.5. <<screen fragment>> $

They are static fragments, i.e. included by all pages. In our case, the only instance of this stereotype is the userbar. Its functions, however, are accessible only by logged users. This doesn’t mean that the fragment is not included: it’s always there, but its appearance changes depending on the user.

6.5.6. Composition

These arcs link a screen with the input forms it contains.

6.5.7. Arrows

They link screens between which it is possible to navigate directly. There are at most as many outgoing links as the number of operations in a screen.

6.6. Interaction scenarios

This section depicts particular scenarios used to describe the interaction between the users and the system through the user interface. Each scenario is an “history about use” and explains how a certain goal can be achieved by pointing out what actions to perform.

6.6.1. Search for help as a visitor

Actor: a student living abroad that has problems with his TV
Description: Since the actor knows a few people in the new city, he decides to look for help for his broken TV. Told by a friend of the site, he types the url in his browser address bar and stumbles upon the *home page*. Interested in a quick but reliable work, he skips the registration process and *clicks “Search for help NOW!”*. He is presented the search page: he fills out the required fields, *dragging “Fixing TVs” onto the skill field*. When the result page is loaded, there appear some results, displaying avatar, rating and recent feedbacks from the users. The actor decides to contact the user with the highest rating and therefore *sorts the list by rating* and *clicks on the first listed user*’s avatar. Once on the user profile, he reads his contact informations and, since he is not a registered user, decides to contact him by e-mail.

6.6.2. Create a new profile to advertise yourself

Actor: a freelancer computer technician that wants to broaden his clientbase
Description: The actor has just opened his new lab and wants to increase his clientbase. In order to achieve this, he thinks to advertise himself as a technician and consultant through social networks and SWIMv2 among them. From the *home page*, he *fills out the e-mail and password textboxes* and *clicks the “Register” button*. The system shows him a notification stating that a new message has been sent to the provided e-mail. The actor then logs into his mail account, opens up the system mail and then clicks the activation link written within. A *new page opens* and the system pops up a welcome message for the newly activated account and *automatically logs him in*. Since it is the first time that the user logs in, the system suggests him to complete his profile. He *clicks on the home button* to open a menu, from which he *chooses “Edit profile”* and proceeds to *change username, description, informations and avatar*. He *clicks the update button* and his profile is now visible on the site.

Changelog

1. Updates during DD creation

- Removed “filter by location” from use cases descriptions
- Removed “filter by location” from scenarios
- Removed “filter by location” from interfaces mockups
- Removed “filter by location” from interaction scenario
- Filtering by location is no longer an available feature.