Software Development Project
LSINF2255 / LINGI2255
2013-2014

Teacher : Pr. Kim Mens
kim.mens@uclouvain.be

14 mars 2014

This project consists of the development of a realistic application, representative of a typical industrial software system, under semi-professional working conditions. The topic of the application to be constructed, as well as the development methodology to be used, will be proposed by an industrial partner who participates in the organisation of this course. The project will be carried out by groups of 6 to 8 students.

Table des matières

1 Course Description **4**
1.1 Aims 4
1.2 Main themes 4
1.3 Course prerequisites 5
1.4 Evaluation methods 5
1.5 Teaching methods 5
1.6 Bibliography 5
1.7 Suggested Reading 5

2 Project Description **7**
2.1 Disclaimer 7
2.2 Description of the project as given by the client 7
2.2.1 Introduction 7
2.2.2 Context 8
2.2.3 Stakeholders 8
2.2.4 Success factors 8
2.2.5 Practically 9
2.2.6 Points of attention 11
2.3 Functional requirements 11
2.4 Non-functional requirements 12
2.5 Possible extensions 12
2.5.1 Security and user verification 13
2.5.2 Advanced search engine 13
2.5.3 Presentation of search results 13
2.5.4 Evaluation of exchanges 14
2.5.5 Internal mailbox system 14
2.5.6 Stimulating users 14
2.5.7 Newsletter 14
2.5.8 Interoperability 14
2.5.9 Content management 15
2.5.10 Supporting groups 15
2.5.11 Other 15
2.6 Role of the *Solidare-it !* stakeholders 15

3 Project Phases **17**

4 Deliverables **19**
4.1 Overall project flow 19
4.2 Reports 19
4.2.1 Requirements report 19
4.2.2 Architecture report 20
4.2.3 Design report 21
4.2.4 Final Deliverables 22
4.3 Project defense 22
4.4 Points of attention 22

5 Tools and models **24**
5.1 Models and notations 24
5.1.1 Activity diagrams 24
5.1.2 Use cases and user stories 24
5.1.3 Use case diagrams 25
5.1.4 Wireframes 25

5.1.5 Architectural views . . . 26
5.1.6 UML class diagram . . . 26
5.1.7 UML sequence diagrams . . . 26
5.1.8 State machine diagrams . . . 26
5.1.9 Data Modeling . . . 26
5.1.10 Patterns . . . 27
5.1.11 Development plan . . . 27
5.1.12 PERT . . . 27
5.1.13 Relational schema . . . 27
5.1.14 Other . . . 27
5.2 Tools . . . 27
5.2.1 Web application framework and programming language . . . 27
5.2.2 Wireframing tools . . . 28
5.2.3 Testing framework . . . 28
5.2.4 Code quality assurance tools . . . 28
5.2.5 Project management tools . . . 28
5.2.6 Version control systems . . . 28
5.2.7 DBMS . . . 29
5.2.8 Text editor . . . 29
5.2.9 Presentation tool . . . 29
5.2.10 Other . . . 29

6 Project Schedule 30

7 Appendix : Additional Details on the *Solidare-it !* System 31
7.1 *Solidare-it !* : A Realistic Solidarity Exchange Platform . . . 31
7.2 Wireframes . . . 31
7.2.1 Home page . . . 32
7.2.2 Registration page . . . 32
7.2.3 Individual Registration . . . 32
7.2.4 Organisation Registration . . . 32
7.2.5 Individual Welcome Page . . . 35
7.2.6 Organisation Welcome Page . . . 36
7.2.7 Registration of a User by an Organisation . . . 36
7.2.8 Adding Representatives of an Organisation . . . 36
7.2.9 Posting an Offer or a Demand . . . 38
7.2.10 Searching an Offer or a Demand . . . 38
7.2.11 Presenting the Search Results . . . 38
7.2.12 Profile description . . . 39
7.2.13 Group creation . . . 39
7.2.14 Evaluating an Exchange . . . 41
7.2.15 History . . . 42

1 Course Description

Credits	6 ECTS
Volume	15h + 45h
Semester	Q1
Language	English
Level	Master 1 (SINF21, INFO21)
Amount of students	~60

1.1 Aims

Students completing successfully this project will be able to work in a *team* to develop a large-scale, *industrial-like* software system of good *quality*, from the ground up, starting from its initial requirements to its final deployment, following a model-based development approach, in which *software models* play an important role to document the important design decisions and software artifacts throughout the development process.

More specifically, students having completed this course with success will be able to :

– Complete, in a rigorous and systematic way, the different software life cycle phases (requirements analysis, architectural analysis, design, implementation, validation, documentation and deployment) ;
– Apply a software development methodology currently practiced in industry ;
– Put in practice different methods and techniques to assure the quality of the produced software ;
– Estimate the time and resources needed to complete such a software development project, plan the tasks to be executed and the deliverables to be produced, and respect this planning ;
– Use a project management tool to assign and follow the planned software development tasks ;
– Understand the problems inherent to the development of large software systems having many different stakeholders and that consist of multiple components addressing different problems ;
– Work in team and manage the coordination and communication between the different team members ;
– Interact with a client to identify his requirements, to clarify imprecise specifications, and to take into account requested modifications throughout the development process ;
– Build upon existing code that may be poorly or badly documented.

Students will have developed skills and operational methodology. In particular, they have developed their ability to :

– Work in a team, by demonstrating through involvement in a team project the central elements of team building and team management ;
– Write reports showing clearly the progress of a project and justifying the choices made on the basis of rigorous arguments ;
– Follow a model-based development approach to document the important artefacts produced throughout the software development process.

1.2 Main themes

Depending on the topic of the project and the chosen software development methodology, which may vary from one year to another, the following themes may be addressed to some extent :

– The software lifecycle : products and processes ;
– Model-based development ;
– Requirements analysis : gathering and documenting software requirements ;
– Architectural analysis : software architectures, architectural styles and patterns ;
– Software design and documentation of important design decisions at each development step ;
– Programming techniques, software development environments, refactoring ;
– Software validation through unit tests, integration tests, functional and structural tests, and code reviews ;
– Project management, planning, resource estimation, reporting ;
– Version management by using a version management tool.

1.3 Course prerequisites

- Knowledge of object-oriented programming, algorithms and data structures (such as provided by the courses LSINF1101, LSINF1102, LFSAB1401, LFSAB1402 and LSINF1121) ;
- Having followed introductory courses on object-oriented design and on data management (such as the course LSINF1225) ;
- Having participated in the implementation of a small size software project (such as for example LFSAB1509).

1.4 Evaluation methods

The course evaluation will be based on :

- Individual participation to the weekly group meetings with the course assistants : 20% (see Section 4.1) ;
- Three intermediate reports : 40% (see Section 4.2) ;
- The final report, delivered system and documentation, presentation and demonstration of the final product : 40% (see Section 4.2.4).

Given that this course is based on the participation in a team project throughout the year, there will be *no* possibility to do a second session for this course.

1.5 Teaching methods

- Development (analysis, design, implementation, documentation, validation) of a large-scale application, a typical software product from the industry, in conditions of semi-professional work. (More details on this year's project can be found in Sections 2 and 7.)
- Teamwork of 6-8 developers (necessary to complete a big project), overseen by a project manager.
- Weekly meeting with the project leader (a teaching assistant or researcher in the institute) : presentation of the progress and difficulties, assessment of alternative options, proposed distribution of work within the team.

1.6 Bibliography

The current document is supposed to serve as the main reference document for this course, both for the students as well as for the teaching staff.

- A detailed description of the project, its phases, deliverables, and schedule are given in sections 2, 3, 4 and 6, respectively.
- Pointers to appropriate modelling notations and tools are given in Section 5.
- A set of tentative wireframes of the software system to be developed is presented in Section 7.
- Additional resources may be published on the iCampus course website throughout the year.

1.7 Suggested Reading

This section contains a non-exhaustive list of interesting books that would fit well on the bookshelf of any self-respecting software engineer or software developer. We list them here out of completeness and since you may want to consult them as complementary reading throughout this software development project or during the software engineering course.

Software Engineering Books

Software Engineering : Theory and Practice (Shari Lawrence Pfleeger & Joanne M. Atlee, 4th edition, Pearson Education, 2010)
This book gives a good introduction to the theory and practice of software engineering for an introductory course on software engineering and applies the concepts through two running examples throughout the book. The fourth edition of this book also covers modeling and agile methods.

Software Engineering (Ian Sommerville, 7th edition, Addison Wesley, 2004)
As an alternative to the book above (I wouldn't recommend you to read both since they cover more or less the same material) there is the seminal textbook on Software Engineering by Ian Sommerville. My preference goes a bit more to the former, however, but this can be a matter of personal taste.

Software Engineering : Principles and Practice (Hans van Vliet, 3rd edition, John Wiley & Sons, 2008)
As a third alternative to the two above there is also a good textbook on software engineering by Hans Van Vliet, but I must confess that I haven't read that one in detail yet, so I cannot tell you if I like it more or less than any of the two above. For a good introduction to software engineering, I think that any of these three could do.

Software Architecture Books

Documenting Software Architectures : Views and Beyond (Paul Clements, Felix Bachmann, Len Bass, David Garlan, et al., 2nd Edition, Pearson Education, 2011)
This book provides a complete and up to date explanation on how to capture software architecture in a commonly understandable form, it explains the notions of architectural views and styles, and much more.

Software Design Books

Design Patterns : Elements of Reusable Object-Oriented Software (The "Gang of Four" : Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Addison-Wesley, 1994)
This book describes recurring solutions to common problems in software design, presented in a structured format, called 'design patterns'. The book includes examples in C++ and Smalltalk. The book has been highly influential to the field of (object-oriented) software engineering and is regarded as an important source for object-oriented design.

Refactoring : Improving the Design of Existing Code (Martin Fowler, Addison Wesley, 1999)
Refactoring is about improving the design of existing code. It is the process of changing a software system in such a way that it does not alter the external behavior of the code, yet improves its internal structure. This book offers a thorough discussion of the principles of refactoring, including how to spot opportunities for refactoring (called 'bad smells'). The book contains a catalog of more than 40 proven refactorings with details as to when and why to use the refactoring, step by step instructions for implementing it, and an example illustrating how it works. Although the book uses Java as its principle programming language, the ideas are applicable to most object-oriented languages.

Software Testing Books

Test Driven Development : By Example (Kent Beck) In this book, the author teaches programmers, by example, how to adopt a test-driven development approach, where code is continually tested and refactored, as opposed to being tested very late in the software development process, when most harm is already done.

Project Management Books

The Mythical Man-Month : Essays on Software Engineering (Frederick P. Brooks Jr., 2nd edition, Addison-Wesley, 1995)
This is a book on software engineering and project management, whose central theme is that "adding manpower to a late software project makes it later". This idea has become known as Brooks' law and the book is now widely regarded as a classic on the human elements of software engineering.

2 Project Description

2.1 Disclaimer

This year's project is more than a mere case study. It is the actual description of a real initiative. In fact, the project description given in Section 2.2 below, was provided by the *Solidare-it!* consortium. Additional information can be found in Section 7, and throughout the project this customer can be contacted for further information and clarification questions (by sending an e-mail to the generic email address `solidare.it@gmail.com`, which will then be forwarded to the different team members of *Solidare-it!* ; if the question is relevant to other students, it may be posted to an iCampus forum afterwards).

To help them in developping their system, *Solidare-it!* is looking for engaged computer scientists and engineers (you !) that can help them in setting up and prototyping the desired internet platform. If the work your team will achieve throughout this project meets the expectations of the consortium, they may consider it as the basis of the platform that they will develop. For this reason, and since the proposed project is all about solidarity, each student will be requested to sign, at the start of the project, a convention where they confirm that they will make their final code available under an open source license.

2.2 Description of the project as given by the client

2.2.1 Introduction

"*Solidare-it!*, dare the real social network"

FIGURE 1 – The *Solidare-it!* logo.

Solidare-it! is an initiative that tries to enhance the solidarity and the exchange between citizens of all layers of society. From the name "Solidare-it !" we can distill the following things : "solidarity", "dare it !" and "IT". We try to stimulate people to dare becoming involved in real solidarity through information technology. The internet and the social networks which arise from it are used as tools to obtain this goal.

Solidare-it! is an internet platform that acts as a meeting place to facilitate the contact and exchange between its users and to connect people throughout all layers of society. The primary purpose of the platform is to create a bridge and an exchange between local civilians who could use some help, and those who are willing to offer their assistance. The group of people in need can be very diverse. It can be families with financial problems, refugees looking for a new home, an old lady needing some help with a move, etc. Supportive local civilians can offer a bed, a shower, a meal, administrative help, clothes, simply a conversation, or other things. The exchange offers the people in need in the first place a temporary solution for an urgent problem. It can however enhance long term benefits. If a connection between two people from different backgrounds is made, both will exchange culture, knowledge, stories and experiences. This may generate and enhance mutual understanding, empathy and involvement. People will inspire and motivate each other. And finally this can lead to the emergence of friendships and a reinforcement of the social fabric.

In addition to individual citizens, social organisations play an important role in this network too, since they are often confronted with people in need. By using the network, social organisations are able to facilitate their work by finding civilian support. Through the social organisations, people in need without internet access, mobile devices, or computer skills can also become part of the network. By using the network, social organisations get more visibility and recognition for their work.

The network is for everybody that instead of simply accepting things as they are, want to make a difference and believe that they have the responsibility and possibility of being part of the change towards a more inclusive world.

2.2.2 Context

Society is becoming ever more individualistic, especially in big cities. The gap between different social layers of the society is increasingly large. At the same time connectivity has never been bigger thanks to telecommunication technologies such as the internet, smartphones and social networks.

As a "homo empathicus", lots of people are sensitive to the problems and needs of other people. They often want to do more than a rather unengaged financial support once in a while. At the same time lots of individuals or social organizations could use a helping hand from time to time. The problem, however, is that at the moment there is no flexible and ad hoc way to match both needs. Need and help have difficulties to 'encounter' each other.

Therefore a group of volunteers started to develop the *Solidare-it!* project. *Solidare-it!* is the name of an internet platform that has to link both the needs with available help within society. The application has to be efficient, user friendly and simple.

Because a lot of people in need do not have access to the internet, social organizations can post a request and manage profiles for such people they are in contact with. This way, the work of social organizations can be alleviated.

The platform is thus an IT application that will generate a real exchange between people. This way, the platform will, apart from a solution to an urgent problem, also connect people from different social layers from society. This will create more understanding and empathy between different groups and will, hopefully, generate more public support for structural solutions.

The project is a grassroot initiative of engaged civilians. Therefore the team tries to develop the project in an inclusive way, based on *voluntary engagement*.

2.2.3 Stakeholders

Several stakeholders are involved in the project. On the one hand there are the initiators, a group of engaged civilians. But the most important stakeholders are of course the users. There are basically three types of users. The individuals in need, the people willing to help, and the social organizations working with people in need. In addition to that there are the web developers, building the platform, you !

In order to develop the project, three groups have been set up, the basic volunteers, organizing the project, a feedback group consisting of representatives of social organizations, and the IT group. The basic volunteers do the organizational and reflective work. The feedback group is used to check ideas and to obtain feedback from people in the field. The web developers will build the platform. Basically there will be the people setting up the basic system (you), and the computer scientists that will do the maintenance and further development later on.

2.2.4 Success factors

The first and logical requirement for a working platform is that the website and application have to be efficient. The search engine has to deliver in the best possible way, and with a reasonable response time the best match between an offer and a demand, or a search for interesting offers or demands. Results have to be presented in a clear, easy and intuitive way. Hereby the ranking of results plays a crucial role.

Secondly, the application has to be user friendly and easy to use. People in need might be prepared to spend a little bit more time looking for a solution, but people that are willing to offer help might quickly become demotivated by an inefficient or hard to use system. The website should also alleviate the work of the social organizations, not give them more work. Otherwise they will simply not use the application.

A good balance between offer and demand is crucial for the functioning of the platform. An important role here is reserved for the marketing and publicity team. But the art of motivating people also has to be built in into the functioning of the application (see further).

The website has to generate positive experiences, in order for people to keep on using it. Therefore the security of the website is of utmost importance too. We'll have to build in several security measures to limit to a maximum abuse or negative experiences.

Finally, realistic expectations are crucial for people to use and to keep on using the website. People in need have to be very well aware that a solution cannot always be found. Nevertheless, we have to design the application in such a way that the chance to find a solution is maximized. All users have to be aware of the risks and the limitations of the platform. The application will have to manage the expectations of all users.

2.2.5 Practically

The first step of the website is the making of a user profile. This step is important for several reasons. A distinction will be made between individual users and social organizations. Several criteria of the profile will also be used in the search algorithm, such as the sex and the address for example. A second important parameter of the profile will be the "verification". In order to guarantee more security for users, people can either connect or upload their ID card data. If a copy of the ID card is uploaded, the readability and correctness of the data has to be checked by a volunteer. People that have uploaded their ID-card data will get a "verified" flag. If a social organization is making a profile, the existence of this organization also has to be checked by a volunteer or an automated mechanism. A profile from a social organization will get the flag "Social Organisation". The responsible person of the social organization that made the profile of the social organization will become the administrator for the whole social organization. He will be able to invite colleagues by entering their names and email addresses, or by uploading a spreadsheet containing the same information. The colleagues will have to click on a link in an email message they get to become automatically part of the social organization. They only have to enter a few more pieces of data such as their telephone number. A member of a social organization can now make a profile for a person in need. This can be done by entering the necessary data (see corresponding wireframes in Section 7), or by uploading a spreadsheet containing the same information.

Once a profile is made, a person can either create an offer or a demand, of execute directly a search. If a person clearly knows which type of exchange he/she wants to do, he/she creates an offer or a demand. In order to create an offer or a demand, the person will have to fill out two main frames. One frame is the general frame, the second frame is the specific frame. In the general frame, the person answers the basic questions like "who ?", "where ?" and "when ?". In the second part, the person gives a description of the "what ?" question. There will be a predefined taxonomy of possible things to be exchanged. This taxonomy will be updated and enlarged in the future. The taxonomy has the form of a tree structure, as illustrated by Figure 2.

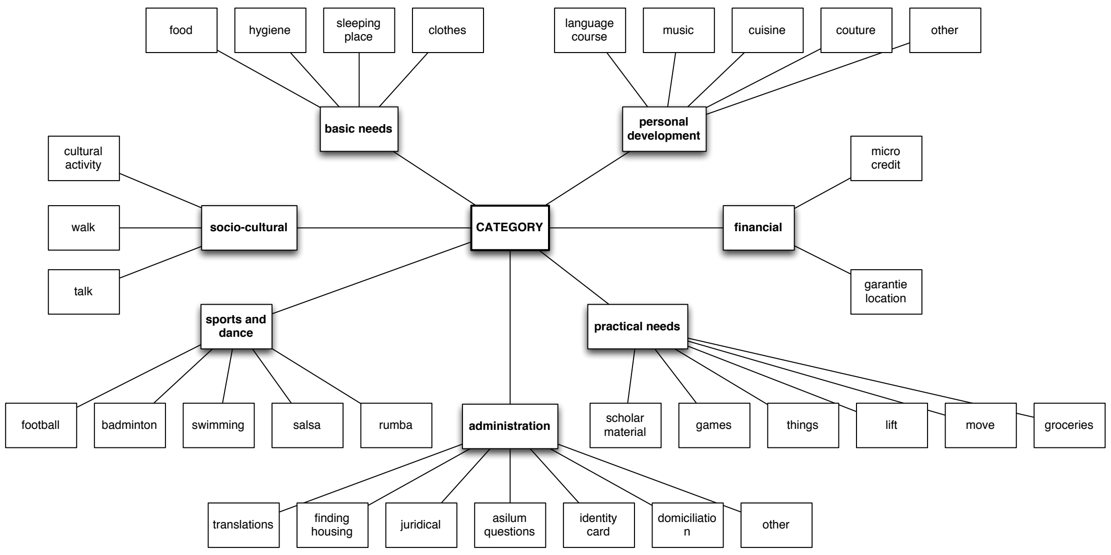


FIGURE 2 – Taxonomy of possible things to be exchanged.

E.g. people can chose between basic needs, socio-cultural, administration, practical needs, personal development, etc. Within the "basic needs" category, people can choose between food, hygiene, sleeping place, or clothes. To make the subject of search even more detailed, they can than add a free description. Once all mandatory fields are filled in, the person can execute a search.

A second possibility is that a person does not really have a clear view of what he is looking for. In this case he will just execute a general search, without really making a "demand or an offer". A person can for example only search for a person with whom he/she already did an exchange. Basically, both searches are almost the same. The big difference is that an offer or a demand will be published, and stays valid for a certain amount of time, while a simple search cannot be found by other people. It must however be possible for a user to save a search so that he can perform it again later, although it is better to post a "demand or offer" if a search has to be executed often.

Once a demand or offer is posted, or if a search is executed, the search results will be shown. It is still possible to filter and organize the search results in such a way that suits the user. The result page is built up in such a way that an overview is given on top of the page, and that detailed information is available on click at the bottom of the page.

Once a "match" is found, the person will probably first do some research about the other person of the "match". In order to do so, this person can have a look at the profile description of the other person. This profile description will contain the current offers and demands the person has actually posted, the history of exchanges done and the evaluations he has got and given (see later).

If the person who found the match feels secure about the other person of the match, he/she can send an internal mail to his/her match. The other person will receive a mail in his personal mailbox with a short message. The conversation is built around the match that has been found. This can be either an offer-demand match, of just an offer or a demand, if the person just executed a search. The mailing system is internal, but a person can of course choose to receive a notification in his personal mailbox. At the bottom of the internal email, a person can click the accept/decline/maybe buttons to answer the mail (there is no common reply button). The mailing system is internal for security reasons, and to keep track of the amount of exchanges done. People can now arrange the rest of their exchange. Of course, people can also give their personal telephone numbers or email addresses in order to arrange for the exchange. In this case we just want to remind them with a popup to be very careful when doing this. It can be interesting to ask people what date and time the exchange will happen. This will then be shown in the calendar of their home page, and they will be notified on time by email (or in a later version perhaps even SMS).

If people have accepted an exchange, they will be asked to fill out an evaluation form the next time they arrive on the website. A pop-up will ask them after they have logged in again if the exchange has already been done. If yes, they will be guided directly to the evaluation form. The evaluation is rather simple. People can choose between positive, neutral or negative. In the latter case, they have to give a comment before they can continue. The mandatory evaluation is needed to guarantee a degree of social control on the website. This is important for security reasons, and can also be used to rank the search results (see later). But the evaluation system can also be used as a way of stimulating people (see later).

After the evaluation people see their personal welcome page. On this page they can find all relevant information about their activity on the platform. They can have a look at the saved searches, the saved matches they have found, but not yet accepted, they can launch another search, change their profile data, etc. If the person logged in is a person of a social organization, he will have a welcome page containing a summary, and the names of the people he is managing. If he clicks on a person, he'll see the page of that person almost as if he was the person himself.

We would also like to have a few special features integrated in the website. There should be a possibility to follow a person. In this case you can request to receive a notification if the person in question posts a demand or an offer. There should also be the possibility to create groups. There are two different types of groups. One type is an ordinary profile created by a social organization, but instead of an individual, a family, it is a group of persons e.g. a group of asylum seekers having a hunger strike in a church. A second type of group is created to keep a lot of people informed about the same offers and demands. In this way the football team of Brussels can participate in exchanges for example. This last type of groups is not so important, but we want to integrate it because we think people are sometimes more motivated to get involved if they are in group.

2.2.6 Points of attention

One of the most important factors of success of the application will undoubtedly be the *search engine* that matches offers and demands, or that performs searches. The web developers will have to think about the best way to program the search engine. Some parts of the profile of a person can be used when showing the search results, such as for example the address of a person. The history of a person can also be used to find appropriate results. The evaluations could also be used to rank the search results. It could, for example, be better to put matches with people having already a lot of good evaluations on top of the ranking. How many search results will be loaded at once ? How will search results be refreshed, etc. ? These are all very interesting questions for the developers. Think carefully about which of these features will or be part of the initial core functionality and which should be considered as future extensions.

Another very interesting aspect of the application will be the motivation of the users. How do we keep people motivated to participate in the exchange of solidarity without anything in return ? A lot of the success of the website will depend on this. We propose to build in some *soft incentives* to motivate people. At the home page of the website there will be a challenge proposed by a famous person if X exchanges of a certain kind will be done. There may also be a few small movies of famous persons who stimulate people to "dare the solidarity". But we are thinking about other incentives as well. One idea is, instead of having an evaluation with three possible choices (negative, positive or neutral), to allow people to give a score from 1 to 5. We are however not convinced that the target audience will be sensitive to this type of competition. It will be good however to give people who do a lot of positive exchanges some type of recognition. Every week (or month) we will select somebody who has done a lot of good exchanges. This person will be an example to stimulate other people (ranked 1,2,3 to have 3 testimonials on the home page). The exchanges of the week will be integrated in the weekly newsletter. We will also integrate a big "Dare-it !" button on the home page, to stimulate people to click on it easily. And of course, these days you cannot have an attractive website without at least a small animation that explains the concept.

Another important aspect of motivation is teasing persons to participate, by sending them examples and suggestions for interesting activities, but also by helping them optimize their search. If they did not get any answer or match after a while, we can suggest them to make the search a little wider ; maybe they will then find something else that interests them as well.

Another interesting issue is how to maximize the automation of several aspects of the website. At this moment, copies of ID cards still have to be verified by a volunteer of the organization. The same holds for social organizations. How can this work be automated ?

Regarding the mailbox system, we think it is best to organize conversations around subjects (offers or demands or a match between both). But it will also be possible to send a direct mail to a person of course. The reason why we prefer the first solution, however, is that in that case we can keep track of the exchanges that are done, and we can demand for an evaluation at certain moments in time. But when is the subject of such a conversation closed ? When a person has accepted it ? No, because somebody can have a same exchange several times with different persons. We therefore need a mechanism to close subjects. If the indicated time has elapsed, we should ask if either the exchange has occurred, if the exchange is not needed anymore, or to make a new and maybe a little bit wider search. If not, we can simple ask the person to repost the demand or offer with a new timeframe (all other data should be copied automatically). The designers of the applications should think about these possibilities carefully.

Finally, these days is is very important too, for a system to have an easy connection with social media like FB, Twitter and the like. People should easily be able to post demands on their FB account, or send it to the email address of a friend. Friends can also be invited to join the framework, etc.

2.3 Functional requirements

Your first task will be to distill from the description above and from informal communication with the customer, a list of minimal functional requirements that should be implemented in order for the delivered system to be accepted by the customer. In addition to the description provided in Section 2.2, Section 7 provides additional details on the *Solidare-it!* web application. More specifically, it presents a set of wireframes, proposed by the customer, of what the web application would look like to an end user. We strongly advise you to read that section now.

2.4 Non-functional requirements

In addition to the functional requirements, the system should also satisfy several non-functional requirements. First of all, a social platform such as the one proposed may become effective only when a significant amount of users posting offers and demands is reached. But in order to deal with many users, the system should be sufficiently *scalable* and *responsive* :

Scalability Since the application to be developed is a web application, the web server will need to be able to handle requests from many users, maybe even simultaneously. How will the performance of your application evolve as the number of simultaneous users increases. Is there a critical point ? How will you test this ? Can your prevent intentional overloading (denial of service attacks) ? Could your application be deployed (or adapted to work) on a server farm with load-balancing if the extra server power would be needed ? Could the database become a bottle neck ? Would caching of data be an appropriate remedy ?

Responsiveness How responsive is the application to typical requests made by its users ? In modern web applications, there is more to client-side performance than the time it takes to fully load a page. More important is how quick or responsive the site 'feels' to a user. Are there any pages in your web application where it would make sense to load parts in background while the user can already start interacting with the rest of the page ? And how would you implement such asynchronous loading ? Might the map on the home page be a candidate for this sort of asynchronous loading ? What about search results ? Does it make sense to already start showing the most relevant results while the rest are still loading ? Or even to use server-side paging and let the user navigate through the result pages ?

You should think carefully about how to address these non-functional requirements, and how to verify that they are satisfied. Since deploying the system in a real setting with a significant amount of users will be hard or impossible to achieve during the development of this project, your team should build a simulator to at least be able to try out and test the system under realistic conditions.

The safety and security aspects of the system shouldn't be underestimated either :

Security The basic functionality required by the system regarding user verification is that they upload or send by mail or e-mail a scan or copy of their ID card, which is then verified manually by a volunteer of the *Solidare-it!* organization (see Section 2.2.5). A similar process exists for verifying the identity of a social organization. Only after such verification has happened these users will get a 'verified' flag which will give them access to certain functionalities to which normal visitors of the site don't have access (like posting offers or demands or accepting exchanges). Unauthorized and unverified users should not have access to these functionalities, neither should they be able to trick the system into getting such access.

Finally, but not less importantly, there are of course the non-functional requirements that have to do with the internal quality of the system to be developed, such as how reusable and maintainable it is.

Reusability How reusable is your code ? Is your business logic isolated so that new rules can be easily implemented ? Can the views/interfaces be easily swapped when e.g. we want to provide the same functionality as a mobile application ?

Maintainability Is the developed system maintainable ? Is it modular and well designed ? Is the code readable, well-structured and well-documented ?

2.5 Possible extensions

In this section we highlight several extensions that could be added on top of the core behaviour of the system to be developed. Each group should develop at least one of these extensions (to be selected early on as part of the requirements report to be submitted ; see Section 4.2.1). Each extension addresses some more complex functionality that goes beyond the basic functionality of the web application. The extensions are chosen to provide a challenge, allowing your group to differentiate itself (and the system developed), from the other groups. Although the customer will remain available for guiding you with these extensions, there can be bit more liberty in developing these extensions, and your development of these extensions can actually provide useful ideas and feedback to the customer about the feasibility, implications, complications and implementation of these extensions.

2.5.1 Security and user verification

An obvious extension that is of crucial importance for this particular type of application is the *security* aspect. The basic solution of having users upload or send their ID card information, and having it then checked manually by a volunteer, is clearly a non-ideal solution which could and should be optimized.

Things that should be explored in this extension are the automatic verification of ID cards (for example with an ID card reader), or using SMS or credit card verification as an additional security to verify that a user really is who he pretends to be. Feel free to contact the *Solidare-it!* team for more ideas on how to best address this user verification issue.

Another issue related to user verification is to avoid the accidental abuse or deliberate theft of user identity. On the one hand, after a user has logged in, we don't want them to have to retype their user name and password for each action they perform on the site. So probably we need to have some sort of session ID. But how will this be stored ? Will you let sessions expire ? Can you prevent a malicious user from intercepting the session ID and using it to get unlawful access ? Are there any user actions for which it would be safer to demand a new login ?

All these questions and related ones should be explored in this extension which focuses on security, in particular related to user identity.

2.5.2 Advanced search engine

As stated in Section 2.2, the core behaviour of the system should at least include some basic search functionality to allow users to search for interesting offers or demands in the database.

But this basic search functionality could still be extended in various ways, for example, by taking into account localization information (enabling search by proximity), user preferences (based on user profile and usage history) or rankings.

In addition to returning only exact search results, a more advanced search engine could also return "close matches". If a person would not find an exact result, the system could propose some "close matches" in order to motivate the user to perform an exchange anyway. For example, when looking for German language courses, if none are found, the system could perhaps offer the opportunity to launch a search on, or display results related to, other languages.

Related to the close matches, the system could also send some suggestions to persons that were not active on the website for a while. The suggestions would be found in a similar way as the "close matches". After X weeks of inactivity, these suggestions could be sent automatically, to incite the users to revisit the site.

For all these extensions you will need to think carefully how these advanced searches, close matches and suggestions will be organized and implemented.

2.5.3 Presentation of search results

Related, but complementary, to the previous extension is how the search results should be displayed to the end user. The basic functionality of the system would be to just list all results in some fixed predefined order (for example, by date), together with some pagination management when all results wouldn't fit on a single page.

A more advanced presentation and representation could use a more appropriate data structure allowing to sort the results according to one or more parameters : by proximity, by date, by kind, by rating, etc. ; and this in increasing or decreasing order. Changing the filtering criteria should have an immediately visible effect on the results being presented.

Since it is likely that a user may want to reuse the same criteria when presenting the results of future searches, it should be possible to save the preferred ordering for a given user.

Rather than presenting the results as a simple sorted list, alternative presentations can be thought of depending on how the data is sorted. For example on a map for presenting results based on proximity, on a calendar when ordered by date, etc.

2.5.4 Evaluation of exchanges

The basic functionality of the system includes the ability of evaluating a completed exchange with a positive, neutral or negative rating. In fact this evaluation is even imposed to the user, and in case of a negative rating the user is obliged to fill in an additional comment.

However, more interesting evaluation systems could be thought of, without leading to a competition aspect between users, and that would stimulate the users more to give a rating after each exchange.

This extension is deliberately left quite open-ended but could be discussed in more detail with the customer. Make sure to already have some concrete ideas of more advanced notation systems if you would select this extension.

2.5.5 Internal mailbox system

The core behaviour of the system should include a kind of internal mailbox system, allowing two users interested in performing an exchange, to have a conversation to set up an exchange. This conversation will be linked to the specific offer or demand that triggered the conversation.

This extension would explore interesting variations of the mailbox system, for example to allow a non-profit organization to broadcast some important information to several people (for example, to inform all people living in a certain city about an important social event), to allow groups of people interested in sharing an exchange to communicate together, or to provide appropriate tools to the administrators to analyze the internal communications to discover fraudulent or undesired behaviour.

Again this is a quite open extension that could be discussed in more detail with the customer.

2.5.6 Stimulating users

To stimulate visitors to use the system, the basic version of the system should display, on the home page (see Section 7.2.1), a selection of interesting offers and demands that have been posted. To select these offers or demands, the system could select for example those made by users that received mostly positive ratings for their recent exchanges (as these users are likely to propose interesting offers or demands worth mentioning).

In addition to this basic functionality, Section 2.2.6 mentioned some additional ‘soft incentives’ to motivate the users. For example, there could be a challenge proposed by a famous person, with a progress bar that shows to what extent the challenge has already been achieved. There could be a few small videos made by famous people on the first page. And there could be some way of giving people with a lot of positive exchanges (based on the evaluation system) some kind of recognition, by showing part of their profile (unless they explicitly desire not to) on the home page.

Many other incentives such as the above could be thought of. The customer most likely has some additional ideas related to this.

2.5.7 Newsletter

Another idea mentioned in Section 2.2.6 to motivate users is to send them a weekly or monthly newsletter. The most basic version of such a newsletter would be a single newsletter that could be generated by the system administrator, or even produced and sent automatically by the system itself.

A more advanced variant would be to send each individual user a personalised newsletter, which could be generated automatically by the system in batch, based on the user’s preferences and interests and based on his saved searches or last searches. The newsletter could be sent both to the user’s internal and private mailbox. The frequency of receiving such a newsletter and where he wants to receive it can be set by the user in the preferences of his account.

2.5.8 Interoperability

Another interesting extension to the system is to foresee the possibility of the *Solidare-it!* platform to collaborate or interoperate with, for example, other existing social platforms or *local exchange trading sys-*

tems (`http://en.wikipedia.org/wiki/Local_exchange_trading_system`) that offer a similar kind of solidarity exchange as *Solidare-it!*.

In this extension you will need to explore how a gate can be opened to similar existing social exchange systems ? At the very minimum support should be offered for sharing exchanges between such systems, i.e. importing and exporting between such systems should be supported, but more advanced collaborations could be envisioned as well.

2.5.9 Content management

Yet another extension would be to investigate in more detail how non-IT experts could easily manage the content of the system (for example, to add or change texts on the website) without having to become an IT expert. This extension would involve finding out what kind of typical changes the customer would like to support and then to find out what would be the simplest way for a non-IT manager of the system to make such changes. (For example, changing a simple text or configuration file as opposed to changing source code and having to recompile the system. Or having a dedicated easy to use admin interface for the system that can be used by non experts too.)

2.5.10 Supporting groups

As stated in Section 7.2.13, some functionality could be added to the application to allow several individuals to act together as a group when proposing a certain offer or demand or when accepting a certain exchange. How to support the creation of groups and how the notion of groups interacts with the already existing functionality should be studied carefully. Should a group be considered as a special kind of user just like an organisation ? What is the difference between a group, a single user and an organisation ? Does a group have a main responsible person, or are decisions taken by common agreement ? Should each individual belonging to a group individually accept the offer or demand when the group accepts it ? When a user belongs to a group, should the group exchanges he participates in become part of his individual history as well ?

2.5.11 Other

Obviously, the above list of possible extensions indicated by the customer is not exhaustive. If you would come up with some other possible extensions (e.g., ‘gamification’) that you think could be relevant to the project, and that you would like to implement, feel free to propose them to the customer with a motivation of why you think it would be a relevant extension.

2.6 Role of the *Solidare-it!* stakeholders

After all this you may wonder what the role of the *Solidare-it!* stakeholders is in all this.

For the development of the *Solidare-it!* platform, the *Solidare-it!* initiative holders intend to rely solely on . . . solidarity ! This means that they *only* work with volunteers. That is the reason why they contacted several universities for potential help with the building of their product, for example by computer science and engineering students.

From the systems produced by the students, there can be several outcomes. If one of the developed systems is of sufficient quality and satisfies all requirements, the *Solidare-it!* initiative holders may consider to take your system as the basis of the platform that they want to make publicly available. In fact, the *Solidare-it!* team (which are all volunteers that have other jobs as well) put their development on hold, to avoid double work and with the hope of being able to reuse one of your projects. This means a certain risk to them as it may delay their project by 6 months if none if your projects is of sufficient quality. We are more than confident, however, that with your expertise and enthusiasm, this will not be the case and that they will actually have a luxury problem of not knowing what project to choose as their basis, because they are *all* of very good quality.

Alternatively, they may decide to reuse only part of your project, for example only part of the source code, or the user interface, or perhaps the database, and build a new project from a mixture of different projects.

In addition to those very direct results, another interesting outcome would be to take inspiration from the eventual running systems and extensions that you will all develop. Some of you may come up with interesting or alternative ways to interact with the end user, others may come up with some interesting extensions that were not originally planned for. Yet all of these may be useful to get insights in what would work best in the final product that will actually be made publicly available. Finally, it may also give them an idea of what technology is most appropriate to implement their platform, since you will all be given some liberty in choosing the most appropriate language and/or web application framework.

As stated before, during your project, the *Solidare-it!* initiative holders will remain available for remote guidance. More specifically, they can be contacted for clarification or other questions throughout your development and will try to answer your questions in a reasonable time (at most a few days). You can contact the *Solidare-it!* team by sending an e-mail to the generic email address `solidare.it@gmail.com`, which will automatically be forwarded to the different team members of *Solidare-it!*. If your question is relevant to other students, it may be posted to an iCampus forum afterwards. They *Solidare-it!* team will also intervene at particular moments during this project, for example at the start of the project to explain the scope and objectives of the project, after the requirements phase to help in evaluating and providing feedback on your requirements reports, and at the end they will participate as customer to the project defenses to see your final product at work and to evaluate to what extent it matches their initial requirements.

3 Project Phases

Your software development project will consist of several phases. In this chapter, we describe the main objective of each of those phases. [1] A more detailed overview of the overall flow of the project and of the expected deliverables for each phase can be found in Chapter 4, whereas Chapter 6 presents a detailed week by week schedule of the imposed deadlines.

Requirements analysis

The main goal of the requirements analysis phase is to capture the functional and non-functional requirements of the system to be developed, based upon the provided documents and additional information obtained from the customer. The documented requirements will then be checked and validated by the customer to ensure that they accurately reflect his expectations.

A *requirements specification* of a software system is a complete description of the behavior of the system to be developed and includes a set of use cases or usage scenarios that describe the interactions the users will have with the software. In addition to that, it describes the non-functional requirements. Non-functional requirements impose additional constraints on the design or implementation, such as performance requirements, quality standards or security constraints. The software requirements specification document enlists all necessary requirements that are required for the project development. To derive the requirements you need to have clear and thorough understanding of the product to be developed. This may require a detailed communication between the project team and the customer.

Note that it is essential for the requirements specification document to describe the problem to be solved, but that choosing *how* the problem will be solved should be left to the next phases.

Architectural design

When building a larger software system, before diving into its detailed design, you may want to decompose the system first into units of manageable size, such as subsystems or modules, with clearly defined interrelationships, allowing those subsystems to be designed and developed independently afterwards. This is the purpose of the architectural design phase, where you will establish the overall architecture of your software system. Intuitively, a software architecture describes at a high level the decomposition of the system in different components, their relations, and the properties of both components and relations.

When designing the high-level architecture of your system, your architectural decisions should be based not only on the expected system functionality, but also on its non-functional requirements (performance, security, appropriate user interfaces) and the long-term intended use of the system (reuse, likely modifications in the future). The goal is not necessarily to design the ideal software architecture for the system, because such an architecture might not even exist. Rather the goal is to design an architecture that meets all the customer's requirements, while staying within cost and schedule constraints.

Detailed design

After having conceived a high-level architectural design of your software system in the previous phase, the goal of this phase is to add more detail, deciding how each of the individual modules will be designed at a modular level, so that you can easily write the code and implement the design in the next phase. The design process describes the system components using a common language with which to communicate ; object orientation is a particularly appealing basis to do so.

Since low-level design may require a bit more improvisation and creativity than architectural design, it is important to carefully evaluate your design decisions and to pay careful attention to well-known design principles, heuristics and conventions. Your experience with low-level design from your previous programming courses might come in handy.

1. Fragments of this description are borrowed from Pfleeger & Atlee's book on Software Engineering [2].

Implementation and testing

With a modular system design, your development team can work in parallel on the design, implementation, and testing of separate software units. Although much of coding is an individual endeavor, all of coding must be done with your team in mind. Following the principle of information hiding, each separate unit should be described in terms of an interface that exposes what other programmers need to know to use the unit, while hiding design decisions that might change in the future.

Clearly, there are many ways to implement a design, and many languages and tools are available to help you. Apart from making sure that the system satisfies its functional requirements, you need to make sure that its non-functional requirements are met as well. Furthermore, care should be taken to write the code in a way that is understandable not only to yourself and your team, for example when you need to revisit your code for testing, but also to others as the system evolves over time. Attention should be payed to the fact that the code is readable, concise, well-structured, modular, reusable, robust, well-documented, understandable, efficient and correct.

Once you have coded your program components, it is time to test them. There are many techniques that you can use to test your code components individually and as they are integrated with those of your colleagues. Unit testing is the development activity that tests each individual component separately. Integration testing puts components together to help you identify problems as the combined components are tested together.

The objective of unit and integration testing is to ensure that the code implements the design properly ; that is, that the programmers wrote code to do what the designers intended. But you should also test the system itself, to ensure that the system does what the customer wants it to do.

4 Deliverables

Whereas the previous chapter 3 introduced the main phases of your software development project, in this chapter we detail the different deliverables expected for each phase, and describe the overall flow of your project. More details on the actual schedule can be found in chapter 6.

4.1 Overall project flow

This software development project will be carried out in groups of 6 to 8 students. During the first kick-off week, the groups will be created and one of the course assistants will be assigned to your group as overall project leader. You will also need to assign one of your team members as internal project manager. He or she will act as spokesperson and main responsible for the team.

As soon as your group has been created, it is the group's responsibility to contact its project leader as soon as possible to fix a date for the weekly follow-up meetings. Remember that individual presence of each team member at these weekly meetings is required and that active participation to these meetings will be taken into account as part of the course evaluation.

During the weekly meetings with its project leader, your team will present the progress and difficulties, assessment of alternative options, proposed distribution of work within the team.

The development of your project will be split in 4 main phases, with an important delivery after each phase and a final presentation and demonstration of the developed system at the end.

During the architectural design phase, you should give careful thought on how to split the development in subtasks and subteams. Whereas the entire team would have a shared responsibility in completing the kernel of the system to implement, subteams should be created, one in charge of developping a chosen extension in more detail, and another implementing a simulator by which the project can be tested under real-like conditions (given that in the timeframe of the project you won't have the time to have it used or tested by a significant amount of real users). Figure 3 illustrates this separation of the development work in three complementary parts.

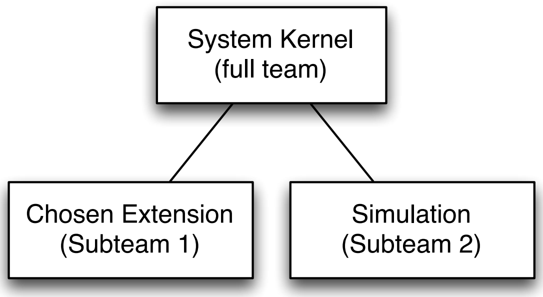


FIGURE 3 – Separation of development in three parts.

4.2 Reports

For this project, each group is required to write three intermediate reports and one final report at the end of the project. Each report corresponds to a specific project phase. The expected content of each of these reports is detailed hereafter.

4.2.1 Requirements report

At the end of the requirements phase (for which we count 2 weeks, not including the first kick-off week), your team is required to submit a requirements report that shall include at least the following items. This report will be submitted to the customer for verification. It cannot exceed 15 pages.

- An *activity diagram* (cf. 5.1.1) to provide a high-level view of the behavior and flow of the software system to be developed.
- Either a set of *use cases* (cf. 5.1.2) or *user stories* (cf. Section 5.1.2) to capture the different usage scenarios that the system should be able to handle, optionally complemented with a *use case diagram* (see 5.1.3) that summarizes the different use cases.
- A list of *non-functional requirements*, categorized per kind, specified under the form of *user stories* or some other means, that the system should respect.

The requirements report will form a kind of contract with the customer, of what the customer may expect from the system to be delivered. Be aware that you may need to account later on for requirements not achieved. If some requirements are optional and will only be implemented if sufficient time remains, please indicate this carefully. Ideally, during the testing phase, all these requirements should be thoroughly verified and tested, so that you can provide evidence to the customer that your system meets the objectives set out at the start.

In addition to the above, although this is normally not part of the requirements, given that the customer already provided rather detailed *wireframes* in the project description (see chapter 7), your report should also include :

- Potential clarification questions that your team may still have regarding the *wireframes* (cf. 5.1.4) or project description provided by the customer. [2]
- Suggestions for *adaptations* of the proposed wireframes, or alternative ways of interacting with the user, to be approved by the customer.

Finally, we also ask you to indicate already in the requirements report :

- The names, coordinates and affiliations of all group members.
- Who will play the role of project manager and spokesperson for your team.
- What the *extension* is that your group will develop. You can either pick one of the extensions listed in Section 2.5 or suggest one of your own. In fact, since your selection will need to be confirmed by the teaching staff (to avoid having too many groups choosing the same extension, and to check the appropriateness of the self-proposed extensions) we ask you to suggest at least three different extensions, in order of preference. The teaching staff will then let you know which extension will be assigned to your group. You don't need to analyse the requirements for these extensions in detail yet. If you provide us with a single paragraph explaining the extension, this is already sufficient. You can develop the details of the extension that was assigned to you later, together with the next phase for example.

4.2.2 Architecture report

As for the requirements phase, you will have 2 weeks for the architecture phase, after which your team will submit an architecture report, of at most 15 pages, that shall include the following items :

1. Choice of the *web application framework* and *programming language* 5.2.1 that will be chosen for your implementation, together with a rigorous argumentation of your choice.
2. An *architectural model* (cf. 5.1.5) that describes the high-level structure of your web application, including :
   - A *logical architectural view* of your software which provides a detailed modularisation of the system into a set of key concepts, taken mostly from the problem domain, in the form of objects or classes, and exploiting the principles of abstraction, encapsulation and inheritance. A conceptual *UML class diagram* (cf. 5.1.6) is an appropriate notation in which to express this logical view, optionally complemented with one or more conceptual *UML sequence diagrams* (cf. 5.1.7). Optionally, if the system calls for it, you could also give a description of the (discrete) behaviour of part of the system as a *UML state machine diagram* (cf. 5.1.8).
   - A *physical architectural view* of your software, which describes how the logical architectural view is mapped to the physical components and connectors on your implementation platform (for example, different processors and other devices and how they communicate). In other words,

2. Note that, since the customer can be contacted by email throughout your project development, we hope that most of these questions would already have been answered before submitting this report.

the physical view describes how the software maps onto the hardware and reflects its distributed aspects.
For example, does the software run on a single computer, or is it partly running on a server, or different services on different servers, and partly on the user's machine ? Is there a native client application on the user's machine or just a web application that runs remotely on the server ? Is all data stored on the server ? On a different server perhaps ? Or is the data, or part of it, stored locally ? And how does this data get synchronized ?
Having such a physical architectural view can be useful to think about several non-functional requirements of the system such as availability, reliability (fault-tolerance), performance (throughput), and scalability.
  – If applicable, a particular architectural style or pattern that is used. For example, it may be the case that the chosen web application framework imposes the use of a certain architectural pattern or architectural style (such as the model-view-controller architecture or MVC).
  – A clear justification of your architectural choices with respect to the alternatives you considered, based on relevant development goals.

3. A detailed *development plan* (cf. 5.1.11) consisting of :
   – The identification of the kernel subsystem that you will implement first in your prototype and a decomposition of this system in several modules that can be developed independently ;
   – The identification of the chosen extension that will be implemented after completion of the kernel subsystem, and a decomposition of this extension and the simulator that will be built to test the kernel system and its extension, in several modules that can be developed independently ;
   – An estimation of the time and resources needed to complete these different modules ;
   – A distribution of subsystems, modules and responsibilities among the members of your team ;
   – The sequence of subsequent increments, up to the full system, to be implemented if time permits (plus an indication of which modules could be dropped in case of lack of time) ;
   – A plan and flow of the tasks to be executed and deliverables to be produced.
4. Either an *ORM diagram* (Object Role Modeling, cf. 5.1.9), or alternatively an *ER diagram* (Entity Relationship, cf. 5.1.9), to describe the data that the system will need to handle.

4.2.3 Design report

For the design phase you will get at most three weeks. Since you will need to design not only the kernel system, but also the chosen extension and the simulator to test the kernel and its extension, you will need to subdivide the work in subgroups responsible for each of these parts. At the end of the design phase, we expect another report of no more than 15 pages that includes :

1. One or more *detailed, implementation-oriented, UML class diagrams* (cf. 5.1.6) describing the static structure of the system in terms of classes and operations.
2. *UML sequence diagrams* (cf. 5.1.7) describing the dynamic behavior of the system.
3. A *relational schema* (cf. 5.1.13), corresponding to the higher-level ORM or ER schema of the previous phase, of the database that the system will be using.
4. A structured and detailed discussion of how the *simulator* and *extension* will be conceived, and how they will interact with the rest of the system.
5. A *revised* version of your *development plan* based on the more detailed design decisions. What has been revised (tasks, modules, assignments of responsibilities) with respect to the earlier plan, and why ? What has already been achieved ? At the end you will need to report on how well you managed to follow up on this revised plan and where you went over schedule.
6. Optionally, an indication of the design patterns, coding conventions, programming heuristics and other means that will be used to improve the overall quality of the final system (cf. 5.1.10).

4.2.4 Final Deliverables

Finally, at the end of the implementation and testing phase, for which you get four weeks (since this is where the real coding will be done), you need to submit a final report, together with the source code and documentation of the developed system as well as the test suites used to test the system :

Final Report Your final report, which will count at most 20 pages, will collect and consolidate the three previous reports, possibly revised after feedback from your project manager on your previous reports, plus the documentation of your prototype. This report will also include a brief description of the test procedures applied on the prototype, notably, the unit tests, integration tests and system tests. The report will end with a conclusion : strong points and limitations of your project ; what did you learn ; what would you improve if you would have had more time ? The final report should be self-contained and targeted to the customer, who may take over your product after delivery to develop it further or to deploy in a real-life setting.

Source code The source code, duly documented, must be made available electronically, together with the installation procedure and a short user manual. This includes the source code of the kernel system, the database used, the extension implemented and the simulator that was created to test the web system and its extension in a realistic setting and to fill the database with realistic data.

Tests Together with the source code of your implemented system, you also need submit the tests you implemented, together with a short description of how these tests can be run. In addition to your unit and integration tests, make sure to include test cases covering the typical usage scenarios of the system, test cases that correspond to exceptional and abnormal situations, and test cases covering the non-functional requirements.

4.3 Project defense

The final project defense will be held in presence of the customer (*Solidare-it!*)[3], your project leader and the course professor, in the last week before the Christmas holidays. The defense will consists of 2 parts :

– a presentation of maximum 25 minutes including questions ;
– a demonstration of the implemented system (maximum 20 minutes). including questions ;

4.4 Points of attention

To conclude this section, we provide some final remarks to take into account when producing the various reports for this course :

– Given the significant number of items to deliver for each report, and the limited time available for each phase of the project, make sure to start early and to organize and distribute the work well among group members. Assign responsibilities and respect your timing.
– When writing your reports, take into account that they are not just reports based on which you will be evaluated for this course, but that they should contain relevant documentation for the customer, who may take over your project after delivery. Also respect the imposed size of the reports, by focussing on the essence.
– Since this course is given in English, given that some of your team members may be international exchange students who are not French speaking, and given that the customer may also have an international team, all documents (reports and presentations) need to be made in English. You can help each other out to correct and increase the quality of your produced documents, but you won't be penalized for the quality of your English.
– Make sure that your reports always take into account the previous remarks made by either the customer or the teaching staff on your earlier reports.
– Ensure traceability between your reports. Each report should rely on the preceding one(s), with explicit and appropriate connections.

3. Since the presence of the customer is required, it is likely that these defenses will be held in the evening, after normal working hours

- In each report, justify and discuss important architectural, design or implementation decisions, as well as possible alternatives that were rejected and why.
- Conclude each report with a brief critical evaluation of your models, their main limitations, aspects not covered by your models, as well as aspects of your system that can be improved.
- Finally, when submitting your reports, respect the deadlines. Late submissions will receive a penalty.

5 Tools and models

This chapter provides a list of suggested tools and models to use throughout your project. However, as this chapter is still under construction, other or better references and definitions than the ones presented here may be suggested throughout the year. Nevertheless, the material presented here already provides a good starting point to get your project going.

5.1 Models and notations

5.1.1 Activity diagrams

During the requirements analysis phase, to provide a high-level view of the system behavior, you are asked to use an *activity diagram*. The main purpose of an activity diagram is to provide a high level view of the system's functionalities, that can be understood by users or customers without a technical background. It does not focus on implementation details, but rather on high-level activities to be performed by the system and its users, and how the control flows dynamically from one activity to another. In that sense, an activity can be seen essentially as a kind of flow chart for modeling the activity flow of the system.

If you want, in addition to that, you could also use an *activity diagram* to describe the organisational process of how you will work as a team, and how the control flows between the different activities to be carried out by the team members. Be aware, however, that these are two completely different uses of the same notation.

For more information on activity diagrams, you can consult the following links[4] :

- `http://en.wikipedia.org/wiki/Activity_diagram`
- `http://www.tutorialspoint.com/uml/uml_activity_diagram.htm`
- `http://www.agilemodeling.com/artifacts/activityDiagram.htm`
- `http://sourcemaking.com/uml/modeling-business-systems/external-view/activity-diagrams`
- `http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_activitydiagram.html`

5.1.2 Use cases and user stories

Use cases The description of 'use case' given in this paragraph is taken from `http://www.agilemodeling.com/artifacts/systemUseCase.htm`. Some concrete examples of both informal and more formal use cases can also be found there. Another interesting resource that describes some of the issues with writing uses cases is `http://alistair.cockburn.us/Structuring+use+cases+with+goals`.

A use case describes a way in which a real-world actor interacts with the system. Use cases can be written in both an informal manner and a formal manner. During your initial requirements modeling efforts you will essentially be using informal use cases written in a very brief, bulleted list style. They should contain just enough information to get the idea across and no more. Formal uses cases just describe much more detail than the corresponding informal use case, but are often a bit of overkill. Often it is better to keep your models as simple as possible and only document them more thoroughly if it adds actual value.

User stories As an alternative to use cases, user stories are often used with agile software development methodologies as the basis for defining the functions a software system must provide, and to facilitate requirements management. A user story consists of one or more sentences in the everyday language of the end user or user of a system that captures what a user does or needs to do as part of his or her job function. It captures the 'who', 'what' and 'why' of a requirement in a simple, concise way, limited in detail by what can be hand-written on a small paper notecard.

User stories are written by or for the user and are that user's primary way to influence the functionality of the system being developed. User stories may also be written by developers to express non-functional requirements (security, performance, quality, etc.).

4. These links were last verified on 15.09.2013.

User stories are a quick way of handling customer requirements without having to create formalized requirement documents and without performing administrative tasks related to maintaining them. The intention of the user story is to be able to respond faster and with less overhead to rapidly changing real-world requirements.

Use cases versus user stories While both user stories and use cases serve the purpose to capture specific user requirements in terms of interactions between the user and the system, there are major differences between them.

User stories :

- Provide a small-scale and easy-to-use presentation of information. They are generally formulated in the everyday language of the user and contain little detail, thus remaining open to interpretation. They should help the reader understand what the software should accomplish.
- Should be accompanied by acceptance testing procedures for clarification of behavior where stories appear ambiguous.
- Are typically used with agile software development methodologies.

Use cases :

- Describe a process and its steps in detail, and may be worded in terms of a formal model. A use case is intended to provide sufficient detail for it to be understood on its own. A use case has been described as 'a generalized description of a set of interactions between the system and one or more actors, where an actor is either a user or another system'.
- May be delivered in a stand-alone document.

In practice, we'll leave the choice up to you whether to use use cases or user stories, depending on your personal preferences or previous experience.

5.1.3 Use case diagrams

Use case diagrams are a simple representation of how and which users interact with a software system. This type of diagram is typically used in conjunction with the more textual descriptions of the individual use cases. While a particular use case itself might drill into a lot of detail about every possibility, a use-case diagram helps to provide a higher-level view of the system as a whole. They provide a simplified graphical representation of what the system is supposed to do.

For more information on use case diagrams, you can consult the following links[5] :

- `http://en.wikipedia.org/wiki/Use_Case_Diagram`
- `http://www.tutorialspoint.com/uml/uml_use_case_diagram.htm`
- `http://www.agilemodeling.com/artifacts/useCaseDiagram.htm`
- `http://sourcemaking.com/uml/modeling-business-systems/external-view/use-case-diagrams`
- `http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_usecasediagram.html`

5.1.4 Wireframes

Wireframes are commonly used to prototype the content, lay out and functionality of the pages in a web application. A *wireframe* is a visual illustration of a web page. It is meant to show all of the items that are included on a particular page, without defining the look and feel or graphic design yet. It's simply meant to illustrate the features, content and links that need to appear on a page so that your design team can then mock up a visual interface and your programmers can understand the page features and how they are supposed to work.

Wireframing is typically done early in the project, after the requirements have been elicited, so that the functionality is known, but early enough so that feedback from the customer or end-user can still be gained before the actual design and implementation is started.

`http://en.wikipedia.org/wiki/Website_wireframe`

5. These links were last verified on 15.09.2013.

5.1.5 Architectural views

Since many different notations could be used to describe the overall high-level architecture of the software system to be developed, we will leave the choice of the particular notation to the discretion of the students. We do emphasize, however, especially if a more ad-hoc notation would be used, the importance of clearly stating the meaning of the different elements in the notation that will be used. For example, if you make a diagram that contains different kinds of boxes and arrows, make sure to explain what each type of box and arrow signifies, and be consistent in your notation. (For example, avoid having two arrows that look exactly the same but that have an entirely different semantic meaning.)

Regarding what to model, it can be useful to document the overall structure of, and interaction between, the different components and packages of which your system is composed. This is sometimes referred to as the "development view" of the software architecture because it illustrates the system from a developer's perspective. A UML package diagram is sometimes used to represent this view.

An overview of the physical architecture of your system could be useful to show the topology of the various physical software components involved, as well as the physical connections between these components. This is sometimes referred to as the "physical view" of the software architecture.

Other architectural views include the "logical view" which is concerned with the functionality that the system provides to its end-users and which are often represented using UML class and sequence diagrams.

A fourth interesting architectural view is the process view which deals with the dynamic aspects of the software system, explains the system processes and how they communicate, and focuses on the runtime behavior of the system. A UML activity diagram is often used to represent this view.

These four views of the architecture (i.e., the logical, development, process and physical view, together with the selected use cases and scenarios, constitute Philippe Kruchten's '4+1' architectural view model [1]. to describe the architecture of a software system. The views are used to describe the system from the viewpoint of different stakeholders, such as end-users, developers and project managers.

Note that the chosen web application framework may also impose a certain architectural style (e.g. MVC) in which case this should also be clearly explained here.

5.1.6 UML class diagram

...section remains to be completed...

5.1.7 UML sequence diagrams

...section remains to be completed...

5.1.8 State machine diagrams

Optionally, during the analysis or design phase, if the system calls for it, the description of the system behavior or part of it could be described as a state machine which specifies the discrete behavior of (part of) the designed system through finite state transitions. A typical example where such a diagram could be useful in a matchmaking process like the one described in the project description would be to describe the different possible states of a possible match. From the moment where an offer or demand is made, to when a possible match is found and the different parties involved get informed of a potential match, to the acceptance (or not) of the match by those parties, and then to the actual execution and final evaluation of the executed match by each of the two parties. Other examples may exist.

A short introduction to what state machine diagrams are, what purpose they serve, and what the notation looks like can be found here : `http://www.uml-diagrams.org/state-machine-diagrams.html`.

5.1.9 Data Modeling

...section remains to be completed...

Object Role Modeling *...section remains to be completed...*

Entity Relationship *...section remains to be completed...*

5.1.10 Patterns

Design patterns, architectural patterns, coding conventions and idioms can all be very useful means of improving the overall quality of the system, if chosen with care. Therefore, if they are used in the project it would be good to document and motivate this explicitly since often they tell a lot about the structure of the system.

5.1.11 Development plan

...section remains to be completed...

5.1.12 PERT

...section remains to be completed...

5.1.13 Relational schema

...section remains to be completed...

5.1.14 Other

Of course apart apart from the above any other kinds of diagrams, models or notations that are deemed useful by the students can be used to specify and document their system such as : goal diagrams, feature diagrams, agent models, state machine diagrams, formal specification schemas, object diagrams, deployment diagrams, package diagrams, component diagrams, interaction diagrams, message sequence charts, and many more. If you decide to use such another notation, please indicate clearly why it was chosen, what purpose it serves in your analysis and why you preferred it over one of the other suggested notations.

5.2 Tools

5.2.1 Web application framework and programming language

We will leave the choice of the most appropriate web application framework and underlying programming language to the discretion of the students. Although some examples will be mentioned below, be warned that this list is all but exhaustive. Not only do there exist many alternatives for the frameworks mentioned below, there are also many frameworks that exist for programming languages other than those mentioned below. Be careful to choose your framework and supporting language well, and make sure to be able to justify that choice with objective arguments. To help you in your search we point out that several web-pages exist that list or compare existing web application frameworks (for example, `http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks`).

Django is "a high-level web framework for the **Python** programming language that encourages rapid development and clean, pragmatic design." More information on Django can be found here : `https://www.djangoproject.com/`

Ruby on Rails is "an open-source web framework for the **Ruby** programming language that lets your write beautiful code by favoring convention over configuration." More information on Ruby on Rails can be found here : `http://rubyonrails.org/`.

PHP offers many different web application frameworks such as Symfony2 (`http://symfony.com`), Zend (`http://www.zend.com`), Drupal (`https://drupal.org/`), Silex (`http://silex.sensiolabs.org/`) or CakePHP (`http://cakephp.org/`).

Backbone.js is "a JavaScript library, based on the modelâ€Şviewâ€Şpresenter design pattern, mainly designed for developing single-page web applications." More information on Backbone.js can be found here : `http://backbonejs.org/`.

Seaside is a "framework for developing sophisticated web applications in the Smalltalk programming language." More information on Seaside can be found here : `http://www.seaside.st/`.

Java also offers many different web application frameworks such as Struts2 (`http://struts.apache.org`) and Spring (`http://java-source.net/open-source/web-frameworks/spring`).

5.2.2 Wireframing tools

Several tools exist to create such wireframes, such as the *pencil project* (`http://pencil.evolus.vn/`), an open-source GUI prototyping tool that is available for all platforms. But even without such a tool, any drawing application or even an application like Excel may do.

...section remains to be completed ...

5.2.3 Testing framework

...section remains to be completed ...

5.2.4 Code quality assurance tools

5.2.5 Project management tools

Throughout this project, your team should use a project management or collaboration tool, to assign and follow up on the planned software development tasks. Such a tool will help your team to organize its work, and allow the team leader to know what still needs to be worked on and who's working on what.

For example, in the course LFSAB1509 which many of you followed, the Trello[6] collaboration tool was used. If that tool worked fine for you then, and if your current team agrees, you can of course use it for this project as well. But you should know that there exists a huge variety of other project management tools out there as well.

Many are general purpose applications that are not directly aimed at software development (neither is Trello), which doesn't make them less useful. Some of these project management tools have built-in code repositories and subversion browsers (or are built around them). A few have built-in bug and issue tracking. All of them can help you keep track of activities and team members. [7]

In the end, given the large variety of such tools that exist we will leave the final choice of what tool to choose to you and your team : choose the one that works best for you. We do advise you to make this choice from the start, so that you and your team members can install the tool and start using it right away, from the very start of the project.

5.2.6 Version control systems

Just like it is essential to use a project management tool, using a source code version control system is essential in any software development project. Again, there are many possible choices to make and we will leave the final choice up to you and your team. Source code version control systems can range from low-level subversion tools like CVS or Subversion (SVN), optionally with a higher-level client to make it easier for you to interact with it, to the newer breed of distributed version control systems, like Git, that have recently gained quite some momentum. An important difference between Subversion and Git is that the latter is decentralized, which means that you can do practically anything offline, because everybody has their own repository. Making branches and merging between branches afterwards is really easy. On top of the Git version control system, you can use GitHub (`https://github.com/`), which is a kind of web-based hosting service for software development projects that brings the power of the web to Git. Alternatives to GitHub exist, for example Bitbucket (`https://bitbucket.org`), but Github seems to be more popular.

---

6. Cf. `http://trello.com/`

7. The following page, for example, discusses 15 such tools, followed by a discussion which suggest even more tools : `http://www.smashingmagazine.com/2008/11/13/15-useful-project-management-tools/`.

5.2.7 DBMS

. . .section remains to be completed . . .

5.2.8 Text editor

No particular constraints will be imposed on what text editor you will use to write your reports, the choice will be left to you, even though we do encourage the use of Latex. [8]
. . .section remains to be completed . . .

5.2.9 Presentation tool

. . .section remains to be completed . . .

5.2.10 Other

Of course apart from the above any other kind of tool deemed useful by the students can be used to help them develop their system.

8. The use of Latex allows you to write high quality reports, would be a good exercise for your thesis, and is easier to collaborate when using a versioning tool.

6 Project Schedule

This section provides a tentative [9] schedule for this software development project.

The weeks in the table correspond to academic weeks. Week 1 is the first week of the academic year, that is, the week of Monday 16 September, and so forth. Week 14 is the last week of the first semester, that is the week of Monday 16 December (i.e., the last week before the Christmas).

Although a weekly theory slot is foreseen every Tuesday from 10 :45 until 12 :45 in the SCES03 auditorium, this slot will be used only occasionally, in which case you will be informed about it beforehand. The very first course will be on Tuesday 17 September. Your presence is strongly advised. The professor will give a general introduction to the course, its objectives, practical aspects, evaluation criteria, etc. The customer will also be present to introduce the system to be developed.

There is no reserved weekly slot for your meetings with the assistant who will play the role of project leader. Once your group has been created, it is your responsibility to contact your project leader to fix a date for your weekly meeting with him.

TABLE 1 – default

Week	Activity	Deliverable	deadline
1	kick-off week	Project introduction (2h) Groups created	Tue 17.09.2013, 10h45, SCES03 **Fri 20.09.2013**
2	**requirements**	Introduction to the requirements phase meeting with project leader	Tue 24.09.2013, 10h45, SCES03 by appointment
3		meeting with project leader Requirements report	by appointment **Fri 04.10.2013** midnight
4	**architecture**	Introduction to the architecture phase meeting with project leader feedback on requirements	Tue 08.10.2013, 10h45, SCES03 by appointment Fri 11.10.2013
5		meeting with project leader Architecture report	by appointment **Fri 18.10.2013** midnight
6	**design**	Introduction to the design phase meeting with project leader	Tue 22.10.2013, 10h45, SCES03 by appointment
7		meeting with project leader	by appointment
8		meeting with project leader Design report	by appointment **Fri 08.11.2013** midnight
9	**implementation**	Introduction to the implementation phase meeting with project leader	Tue 12.11.2013, 10h45, SCES03 by appointment
10	**and testing**	meeting with project leader	by appointment
11		meeting with project leader	by appointment
12		Code + tests	**Fri 06.12.2013** midnight
13	deployment	Running system deployed on web Final report	Fri 13.12.2013 **Fri 13.12.2013** midnight
14	project defence	Defence and demonstration with presence of customer	**Thu 19.12.2013**, 18 :00-21 :30 **Fri 20.12.2013**, 18 :00-20 :30

9. Although the schedule is supposed to be more or less complete, some last minute adaptations may be made to this schedule throughout the year, given that this is the first year we organize this course under its current form.

7 Appendix : Additional Details on the *Solidare-it!* System

This section provides additional details on the *Solidare-it!* web application to be developed, on top of the more general description which was already given in Section 2.2. A significant part of the text in this subsection has been borrowed from and was inspired upon documents created by, and discussions held with, members of the *Solidare-it!* consortium. In particular, it lists a bunch of wireframes which the customer already developed to guide their own internal discussions on what functionality the system should foresee. We emphasize that these wireframes are here for illustrative purpose only, and that you are allowed to propose alternative ways of presenting the information and interacting with the user of the application, as long as the customer agrees.

7.1 *Solidare-it!* : A Realistic Solidarity Exchange Platform

Although people have never been more connected to each other than ever before, through mobile phones, social networks, the internet and other means of communication, we observe that our society is ever more impregnated by individualism, competitiveness and indifference. In response to these observations, *Solidare-it!* is an initiative, launched by a group of committed citizens, who believe that while awaiting structural political solutions to these social issues, civilians can and should help each other in solving urgent social problems. More specifically, the idea is to use the power of current-day communication technologies, to build a real social network that truly connects people and enhances the *exchange of solidarity*. Solidarity involves no obligation and no demand for return in any sense. Solidarity is based on altruism and the only expected return is the satisfaction of being able to help each other. We hope, and believe, that many civilians are willing to offer a more personal solidarity, besides a rather unengaged financial contribution to a social organisation. To give them this opportunity, the goal of this project is to develop a supporting tool in the form of an internet platform.

Solidare-it! is an internet platform that links people with a specific need with people who have the solution. The platform is basically a forum where offer and demand related to solidarity is matched in the best possible manner. Social organizations have a special place within the platform since they can launch a demand for solidarity within a large group of citizens. The main goal of this project is to develop a *solidarity exchange platform* of the kind described in Section 2.2. It should be sufficiently *realistic* to be able to serve as the basis for the further development of the actual platform.

The main purpose of the system, which carries the same name as the *Solidare-it!* initiative, is to create a "real" social network, that enhances the contact and exchange between citizens of all layers of society and that promotes mutual aid and solidarity. More specifically, it should act as a kind of *marketplace* for citizens or social organisations to offer assistance to individual or groups of people in need. To ease its adoption, the system should be structured as a web-application where people with a need can be put into contact with people that can offer the needed assistance. The system thus constitutes a platform where offers and demands regarding solidarity can be posted and matched in the most accurate way possible.

7.2 Wireframes

To facilitate your development task, the customer, *Solidare-it!*, has already brainstormed a lot about what the platform could [10] look like. The results of these brainstorms lead to a set of wireframes that, even though not final yet, give a very good idea of the expected functionality and layout of the final application the customer expects to be delivered. Nevertheless, you should not feel too restricted by these wireframes. If you think that a certain functionality is missing or should appear differently or in a different place, feel free to propose such changes to the customer.

In what follows we briefly describe each of these wireframes one by one. Since these descriptions are the result of informal discussions between the teaching staff and the customer, there may still be some loose ends. Some open questions may remain that will need to be cleared out with the customer. We suggest that you attentively write down any possible questions or ambiguities to be discussed further with the customer. [11]

10. We say 'could', not 'should', since changes to the layout and maybe even to the functionality are still expected.
11. For clarification or other questions throughout the development, students can contact the *Solidare-it!* team by sending an e-

7.2.1 Home page

The home page of the application, i.e. the first page a visitor of the *Solidare-it!* platform will be confronted with, contains essentially some general information and latest news regarding the *Solidare-it!* concept. Figure 4 shows a wireframe [12] of *Solidare-it!*'s home page.

As illustrated by that figure, the page contains on the left a Facebook 'like' button, a frame describing the concept of the platform, with underneath a short film explaining the concept, and a calendar of relevant events organised by, for example, the social organizations that are registered on the platform.

In the middle, there is a list of testimonies of satisfied users of the platform. However, after recent discussions with the customer, it is decided to replace this frame by a frame showing the three "top exchanges of the week or month", as a way to give recognition to people who have done beautiful exchanges, and thus stimulate others to do the same. Basically, it is the same as the testimonies, but gives more recognition to certain users.

Underneath that frame, there is a frame with a selection of typical or recent offers and demands that have been posted. This is also a way of teasing visitors of the site to participate. They can click on these offers or demands to read some basic information on these offers or demands, but if they want more information or want to respond to these offers or demands, they'll have to login or register first.

Underneath that frame there is yet another frame showing a map of all/recent exchanges and where they have happened in Belgium.

On the right there is a progress bar for a social challenge posted by some famous person (for example, help 100 young unemployed persons find a job), again in order to motivate people to do more exchanges. Right underneath there is a big 'Dare-it !' button to get registered as a user to the platform, and underneath a few movies made by famous people to motivate new people to join the platform. The other functionalities available on this page speak for themselves.

7.2.2 Registration page

After having clicked on the big 'Dare-it !' button on the home page (Figure 4), the registration page appears (Figure 5), enabling a user to participate either as an individual user to the platform, i.e., as a person who wants to offer or is in need of help [13] (which brings you to the 'individual registration' page, cf. Figure 6), or as an organization, i.e., a social organization that can serve as a proxy for certain persons in need that don't have access to the platform (which brings you to the 'organisation registration' page, cf. Figure 7). Organizations can later add such persons in need to the platform, and manage their profile.

Note that this registration page appears only when you have not logged in yet. If you have already created a login before, it suffices to click the 'login' button on the top right of the home page (or any other public screen) to login to the system.

7.2.3 Individual Registration

This page (Figure 6) allows an individual to register. To avoid abuse and for reasons of security, for registering, an individual is strongly advised to upload a scan of his ID card, which will be checked by a volunteer of *Solidare-it!*. However, people are not *required* to upload their ID card, because there may be people without an ID card and that do not have access to a local organisation that is willing to post their demand, so they have to be able to post something by themselves. Uploading an ID card will give a user a special 'verified' flag however, which gives that user access to more functionality than other users. One can also only search on people that have this flag.

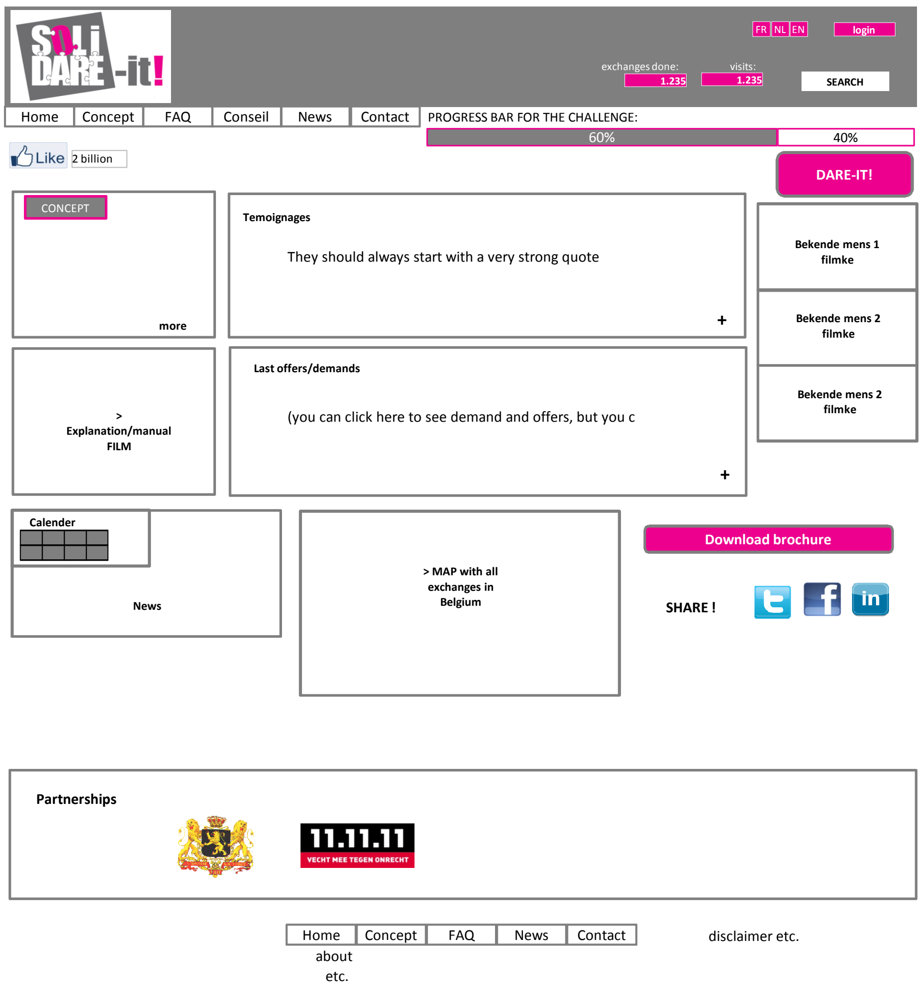
7.2.4 Organisation Registration

Registring an organisation is very similar to registring an individual user, as illustrated by Figure 7, except that here no ID card information but legal information on the organisation needs to be provided

mail to the generic email address of *Solidare-it!* : `solidare.it@gmail.com`, which will then be forwarded to the different team members of *Solidare-it!*. If the question is relevant to other students, it may be posted to an iCampus forum afterwards.

12. Section 5.1.4 briefly explains the notion of 'wireframe'.

13. The platform makes no distinction between the two types of persons ; every participant can make either offers or demands.


FR NL EN login

exchanges done: 1.235 visits: 1.235 SEARCH

Home Concept FAQ Conseil News Contact

PROGRESS BAR FOR THE CHALLENGE:

60% 40%

Like 2 billion

DARE-IT!

CONCEPT

more

Temoignages

They should always start with a very strong quote

+

Bekende mens 1 filmke

Bekende mens 2 filmke

Bekende mens 2 filmke

> Explanation/manual FILM

Last offers/demands

(you can click here to see demand and offers, but you c

+

Calender

News

> MAP with all exchanges in Belgium

Download brochure

SHARE !

Partnerships

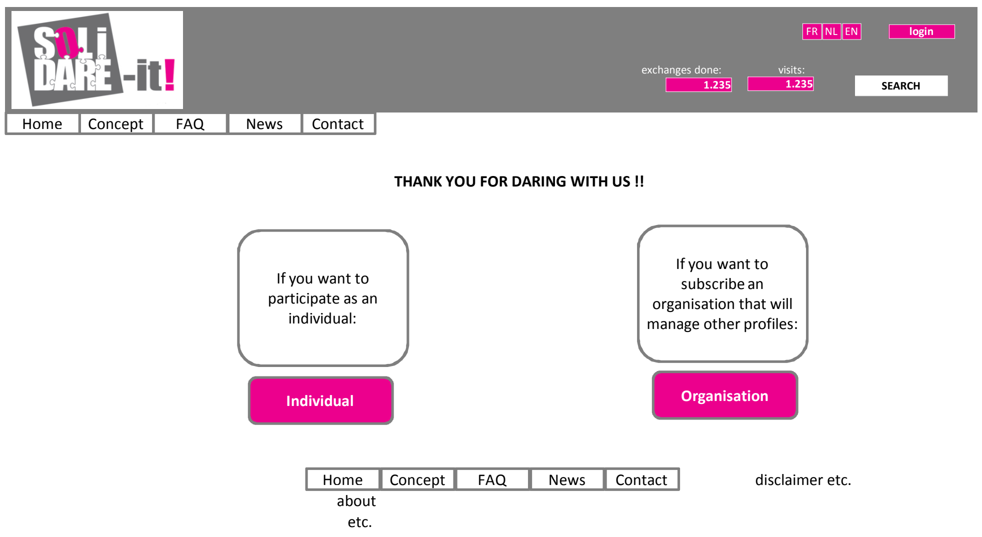
11.11.11 VECHT MEE TEGEN ONRECHT

Home Concept FAQ News Contact

about etc.

disclaimer etc.

FIGURE 4 – *Solidare-it!*'s home page.


FR NL EN login

exchanges done: 1.235 visits: 1.235 SEARCH

Home Concept FAQ News Contact

THANK YOU FOR DARING WITH US !!

If you want to participate as an individual:

Individual

If you want to subscribe an organisation that will manage other profiles:

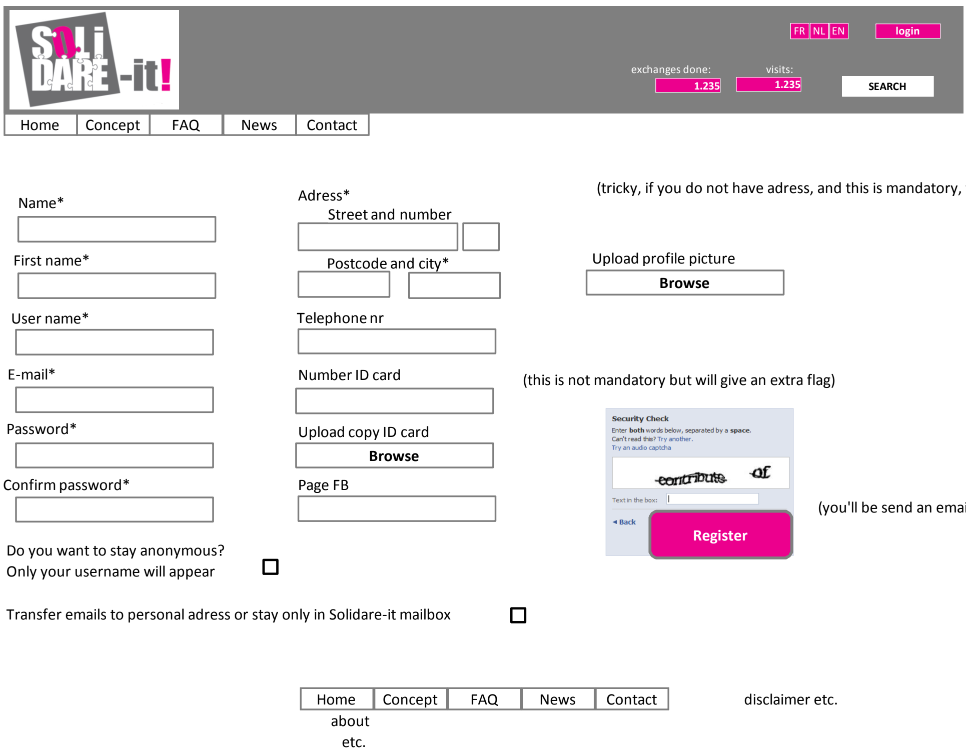
Organisation

Home Concept FAQ News Contact

about etc.

disclaimer etc.

FIGURE 5 – *Solidare-it!*'s registration page.



FR NL EN login

exchanges done: 1.235 visits: 1.235 SEARCH

Home Concept FAQ News Contact

Name*

First name*

User name*

E-mail*

Password*

Confirm password*

Adress*
Street and number

Postcode and city*

Telephone nr

Number ID card

Upload copy ID card
Browse

Page FB

(tricky, if you do not have adress, and this is mandatory,

Upload profile picture
Browse

(this is not mandatory but will give an extra flag)

Security Check
Enter both words below, separated by a space.
Can't read this? Try another.
Try an audio captcha
Text in the box:
◂ Back
Register

(you'll be send an emai

Do you want to stay anonymous?
Only your username will appear ☐

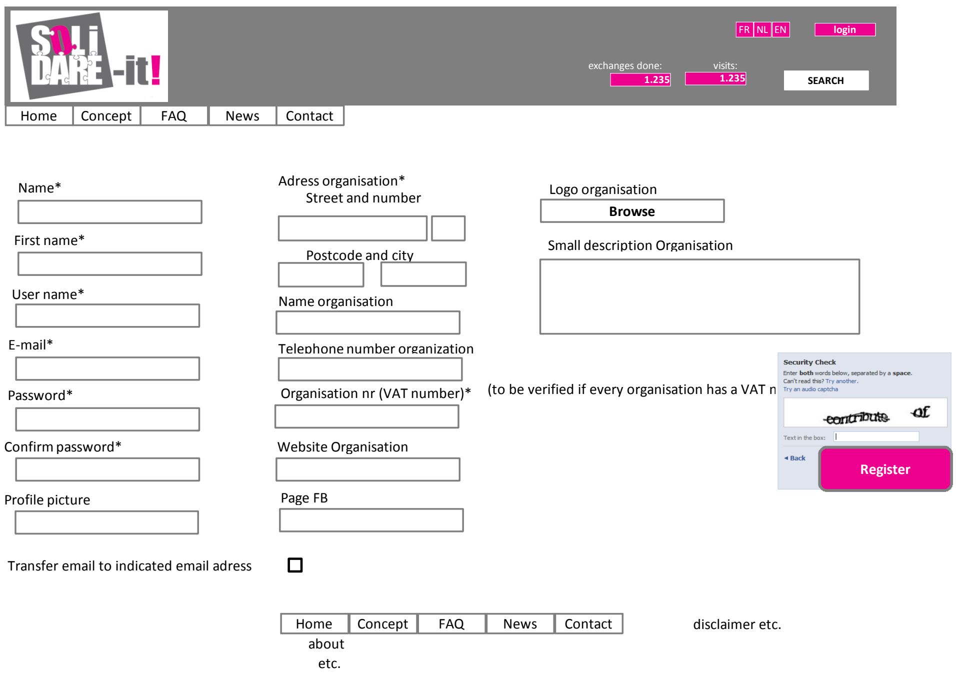
Transfer emails to personal adress or stay only in Solidare-it mailbox ☐

Home Concept FAQ News Contact
about
etc.

disclaimer etc.

FIGURE 6 – Registering to *Solidare-it!* as an individual.

instead (which will again be verified manually by a volunteer who works for *Solidare-it!*).



FR NL EN login

exchanges done: 1.235 visits: 1.235 SEARCH

Home Concept FAQ News Contact

Name*

First name*

User name*

E-mail*

Password*

Confirm password*

Profile picture

Adress organisation*
Street and number

Postcode and city

Name organisation

Telephone number organization

Organisation nr (VAT number)*

Website Organisation

Page FB

Logo organisation
Browse

Small description Organisation

(to be verified if every organisation has a VAT n

Security Check
Enter both words below, separated by a space.
Can't read this? Try another.
Try an audio captcha
Text in the box:
◂ Back
Register

Transfer email to indicated email adress ☐

Home Concept FAQ News Contact
about
etc.

disclaimer etc.

FIGURE 7 – Registering to *Solidare-it!* as an organisation.

7.2.5 Individual Welcome Page

After logging in as an individual, you will see the welcome page of Figure 8. The welcome page contains a summary of the most important activity of a user. At the left corner there is a summary of some basic information, including a small calendar indicating when exchanges should happen. At the right corner you can find a profile picture, as well as the possibility to access the personal information of the user such as his profile or history. On the left column of the page the user can execute a Search (see Section 7.2.10) or Create a demand or offer (see Section 7.2.9) by pressing the corresponding button. Below he finds a summary of his saved searches, his saved matches, and possibly some similar matches. It would be good to immediately update the info of those things as soon as the person visits his welcome page (for example when a request has been deleted or responded to). In the right column a user finds a summary of other useful information of the recent activity such as to which category it belongs and which people the user is following. At the bottom, the user has the possibility to create a group (which allows him to integrate other people in his activity, so that results are sent to the entire group), or to simply invite a friend to join the network as well.

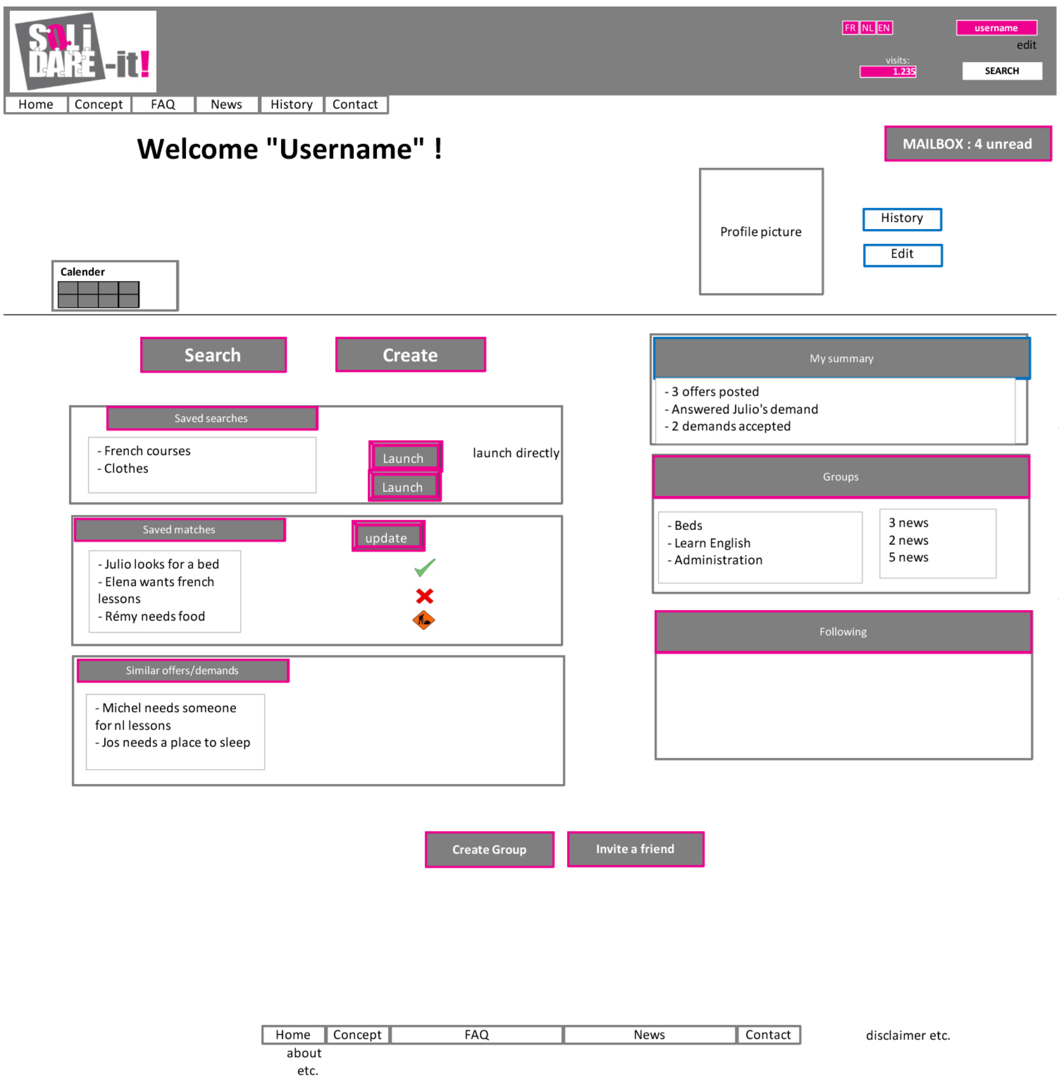
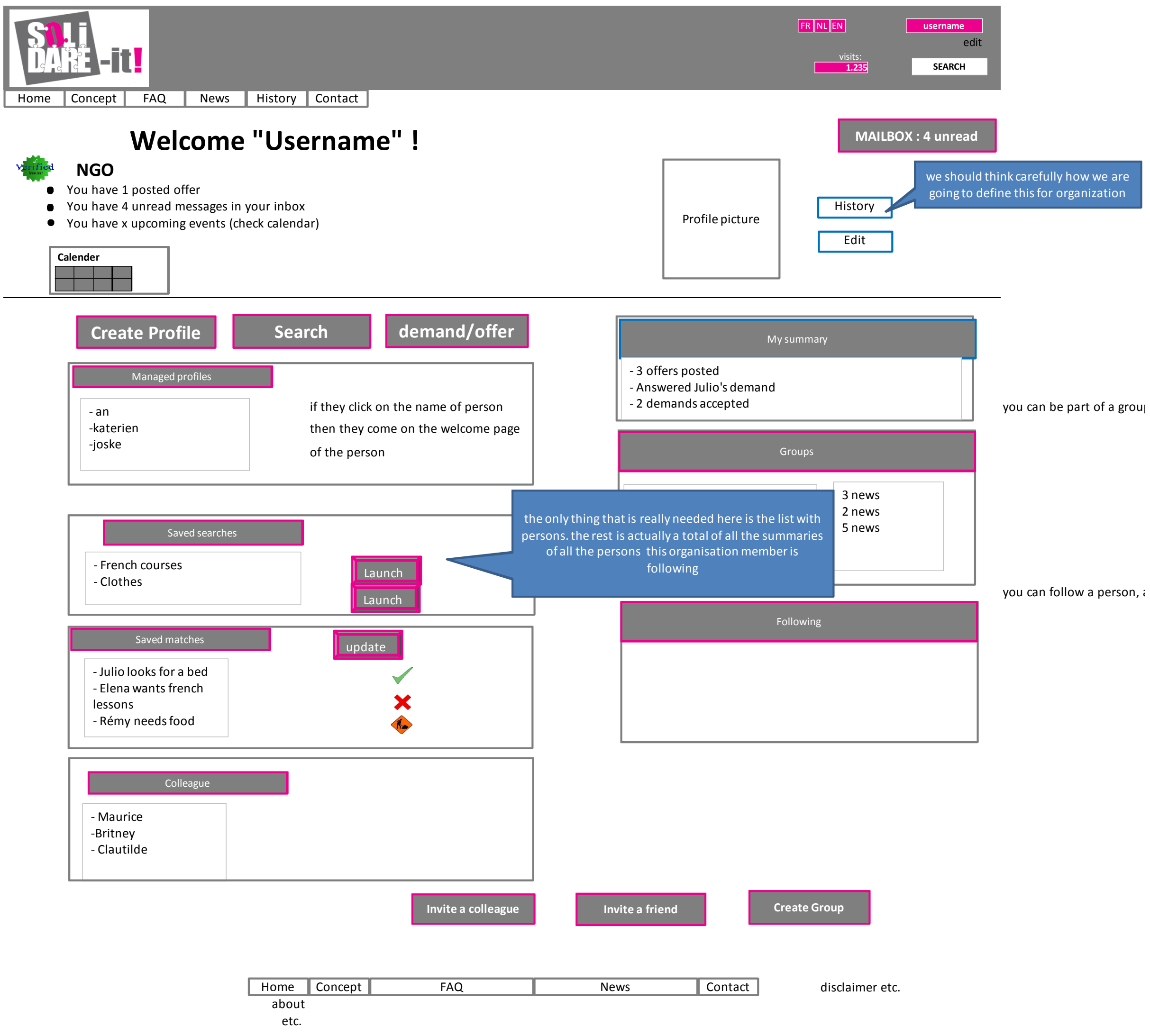


FIGURE 8 – *Solidare-it!*'s welcome page for individuals.

7.2.6 Organisation Welcome Page

After logging in as an organisation, you will see the welcome page of Figure 9. One important functionality that is available here that does not exist for individuals is the ability to create a new profile for one of the persons for which the organisation is serving as proxy to seek help (for example, a refugee finding a place to stay). How to do this is further detailed in Figure 10.



Welcome "Username" !

NGO

- You have 1 posted offer
- You have 4 unread messages in your inbox
- You have x upcoming events (check calendar)

we should think carefully how we are going to define this for organization

- an
-katerien
-joske

if they click on the name of person then they come on the welcome page of the person

- 3 offers posted
- Answered Julio's demand
- 2 demands accepted

you can be part of a grou

3 news
2 news
5 news

the only thing that is really needed here is the list with persons. the rest is actually a total of all the summaries of all the persons this organisation member is following

- French courses
- Clothes

you can follow a person, a

- Julio looks for a bed
- Elena wants french lessons
- Rémy needs food

- Maurice
-Britney
- Clautilde

about
etc.

FIGURE 9 – *Solidare-it!*'s welcome page for organisations.

7.2.7 Registration of a User by an Organisation

This page (Figure 10) shows how an organisation can register a user who cannot create a registration himself (for example, because he doesn't have an ID card or because he doesn't have a computer or access to the internet).

7.2.8 Adding Representatives of an Organisation

This page (Figure 11) shows how an organisation can add additional collaborators to manage the organisation's profile and members. Note that they will only receive an email, after which they still need to

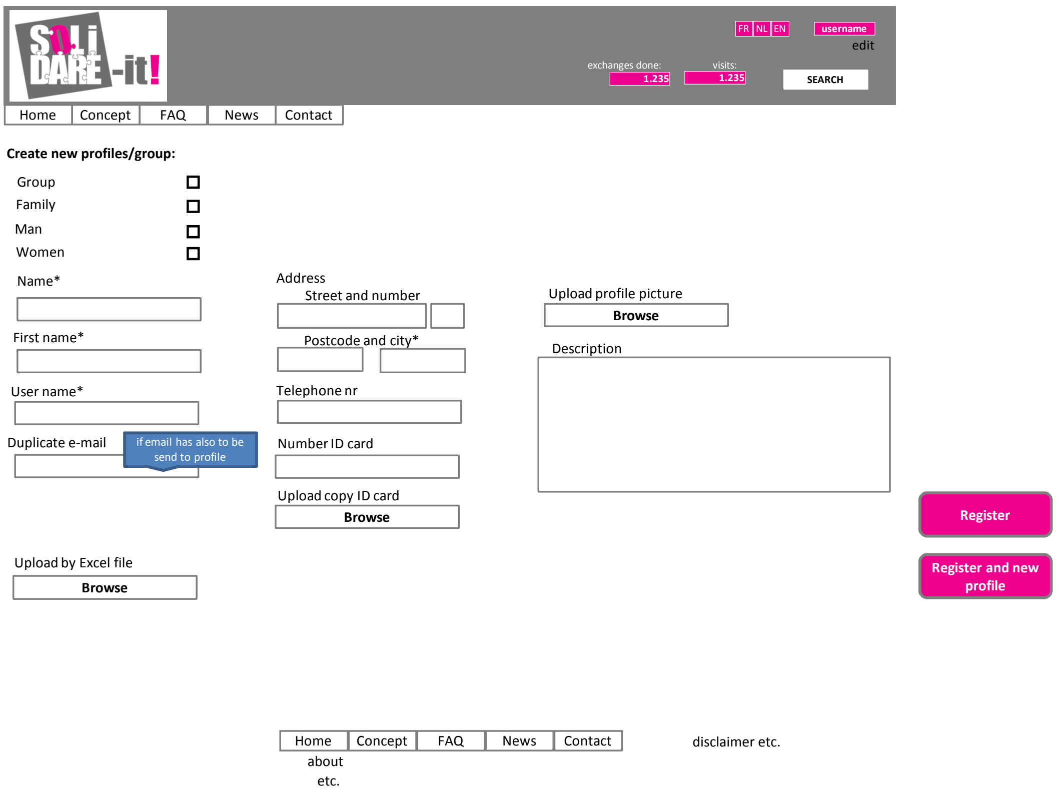


FIGURE 10 – Registration of a user by an organisation.

register themselves to the system first (again for obvious reasons of safety), via a pre-filled form of the type shown in Figure 7.

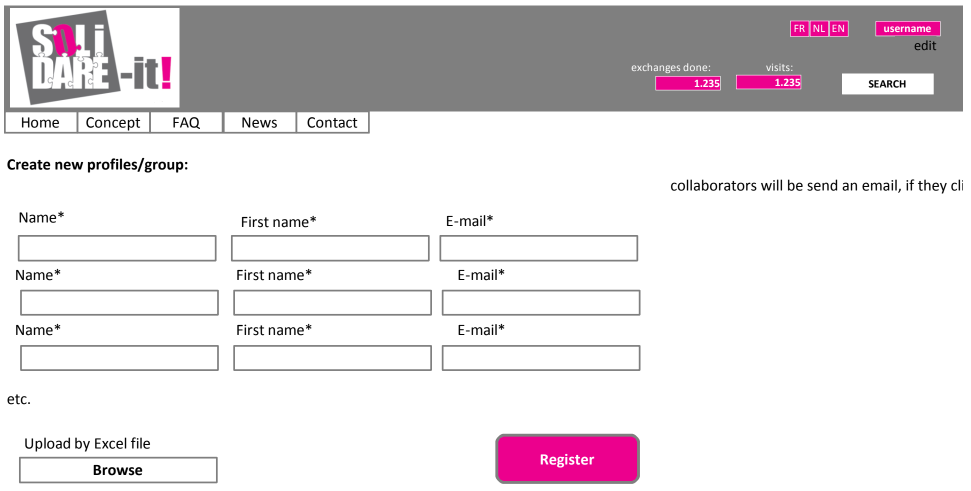


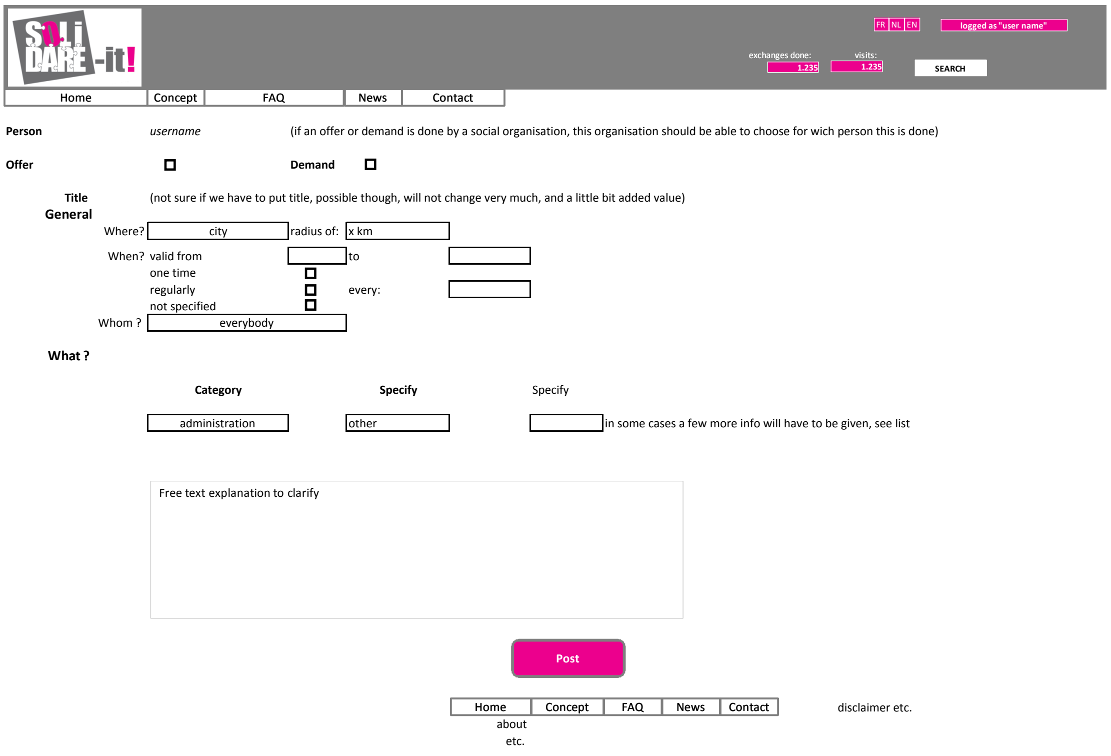
FIGURE 11 – Adding representatives of an organisation.

7.2.9 Posting an Offer or a Demand

Figure 12 shows what the page to let an organisation create an offer or a demand for help could look like. This page can be reached via the Create button on the welcome page of the organisation (Figure 9). Information that needs to be provided to create an offer or a demand :

- Who is the person for which the organisation is acting as proxy for this offer or demand ?
- Where will the offer or demand take place ?
- When will it take place ?
- For whom is the help being requested or offered ? (E.g., for everyone, for a particular individual, for a family, a man, a woman, . . .)
- What type of help is requested or being offered ?

Once all this information has been provided, it suffices to click on the Post button at the bottom of the page to create the actual offer of demand.



Person *username* (if an offer or demand is done by a social organisation, this organisation should be able to choose for wich person this is done)

Offer ☐ **Demand** ☐

Title (not sure if we have to put title, possible though, will not change very much, and a little bit added value)

General

Where? city radius of: x km

When? valid from ____ to ____

one time ☐

regularly ☐ every: ____

not specified ☐

Whom ? everybody

What ?

Category	**Specify**	Specify
administration	other	in some cases a few more info will have to be given, see list

Free text explanation to clarify

FIGURE 12 – Creation of an offer or a demand by an organisation.

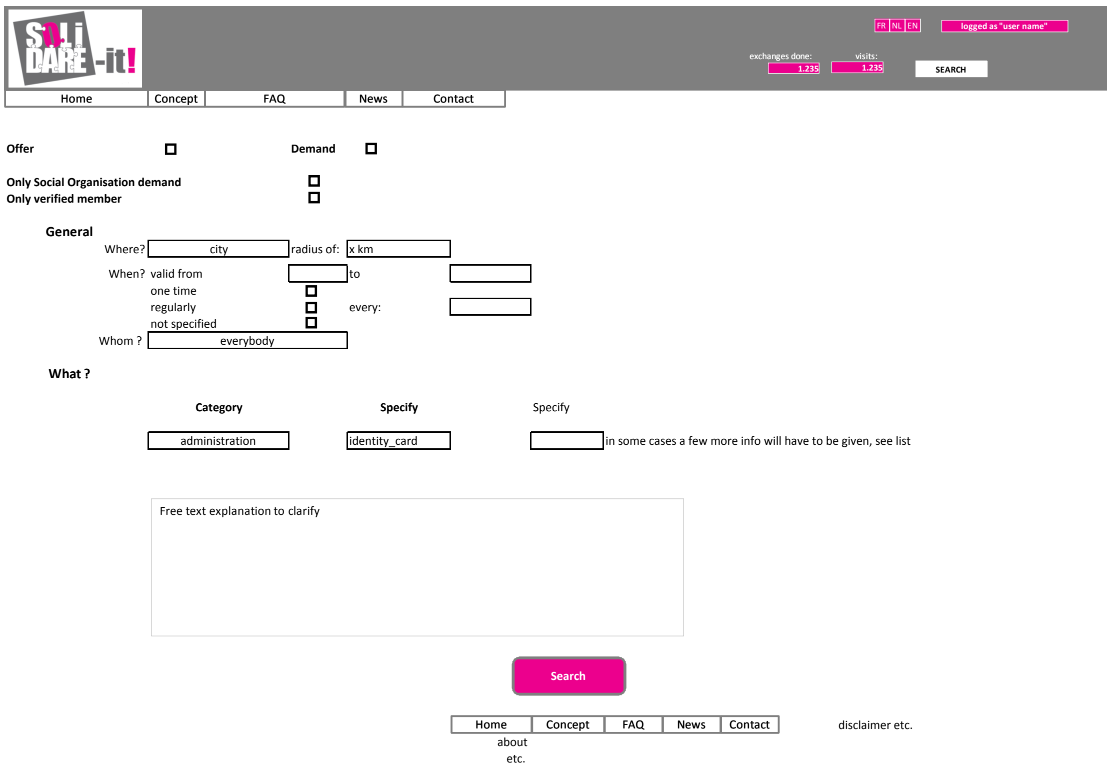
The creation of an offer or demand by an individual is essentially the same as for an organisation (Figure 12) except that for an individual you don't need to say for what person you are making the query : you always act as yourself. The only difference therefore would be that the first field 'Person' would not be there.

7.2.10 Searching an Offer or a Demand

The page for searching an offer or a demand (Figure 13) which can be reached via the Search button on the welcome pages (Figures 8 or 9) is very similar to the page(s) for posting an offer or a demand (Figure 12).

7.2.11 Presenting the Search Results

Figure 14 shows how the search results would be presented to the user having performed a search :



Offer ☐ Demand ☐

Only Social Organisation demand ☐
Only verified member ☐

General
Where? city radius of: x km
When? valid from ___ to ___
one time ☐
regularly ☐ every: ___
not specified ☐
Whom ? everybody

What ?

Category	Specify	Specify
administration	identity_card	___ in some cases a few more info will have to be given, see list

Free text explanation to clarify

FIGURE 13 – Searching for an offer or a demand.

- some fields to filter the results ;
- exact matches, followed by a list of results (1 per line) ;
- close matches [14], followed by a list of results (1 per line) ;
- action : accept, maybe, decline, save, contact (the person), follow (the person), follow (the organisation), report (suspicious things), share (on FB or Twitter) ;
- when clicking on a certain user you get a short description of that user in the bottom text field.

7.2.12 Profile description

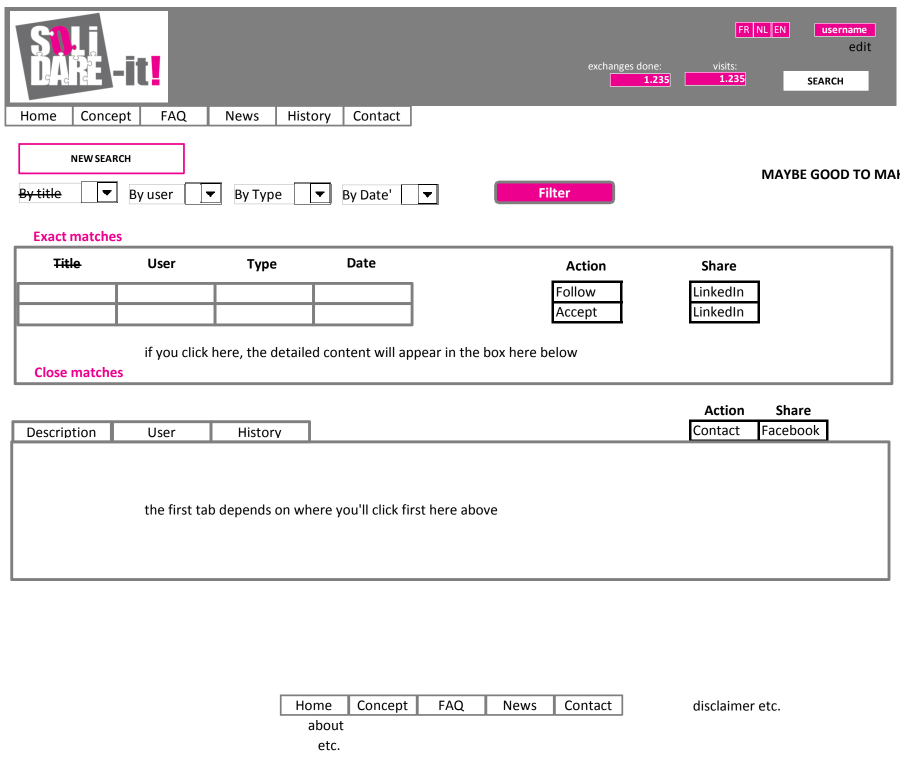
Figure 15 shows the profile of a given person, generated by the system based on the user's history.

Note that there is no such thing as a profile for an organisation yet, because they are just proxies for persons. For now, the *Solidare-it!* team believes that this information may not be too interesting to people using the site, and may make the website more complicated than needed without adding a lot of extra value.

7.2.13 Group creation

The idea of this page (Figure 16) is to allow for several individuals together to act as a group when offering a certain exchange. Probably this functionality (and its interaction with the already existing functionality explained above) needs to be given more detailed thought, but the idea behind is that when acting as a group of friends, the stage fever of proposing help may be less then when acting as an individual. Also, certain offers may logically require a group (e.g., proposing to play a football match). For certain demands too (looking for a place to stay) sometimes a set of individuals (e.g., a family) may need to be considered as a group rather than as a single individual.

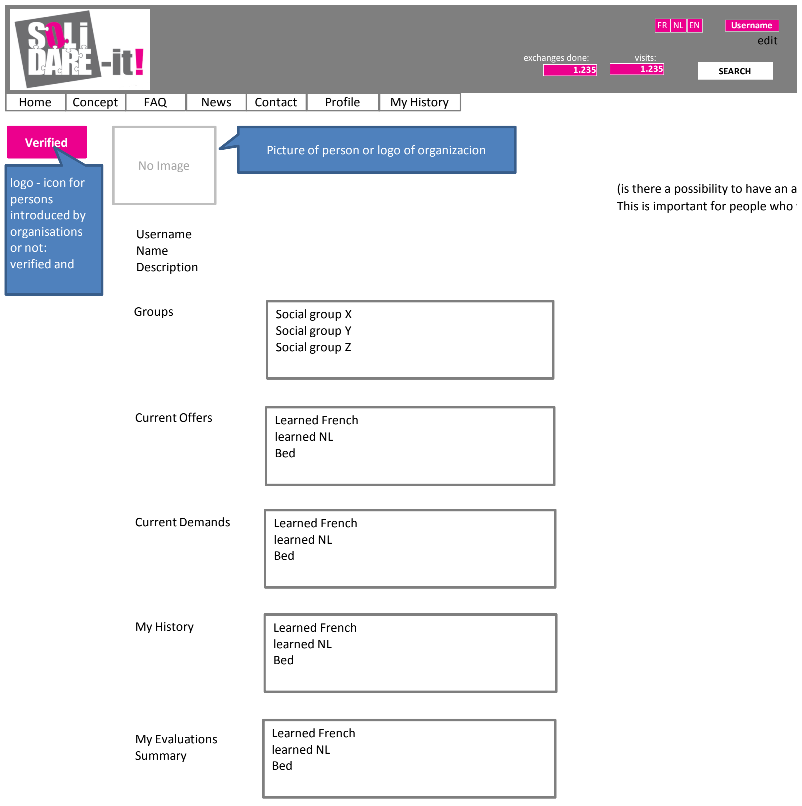
14. At this point it is not decided yet how the close matches will be calculated.



if you click here, the detailed content will appear in the box here below

the first tab depends on where you'll click first here above

FIGURE 14 – Presenting the results of searching for an offer or a demand.



logo - icon for persons introduced by organisations or not: verified and

(is there a possibility to have an a
This is important for people who

FIGURE 15 – Presenting the profile of a given user.

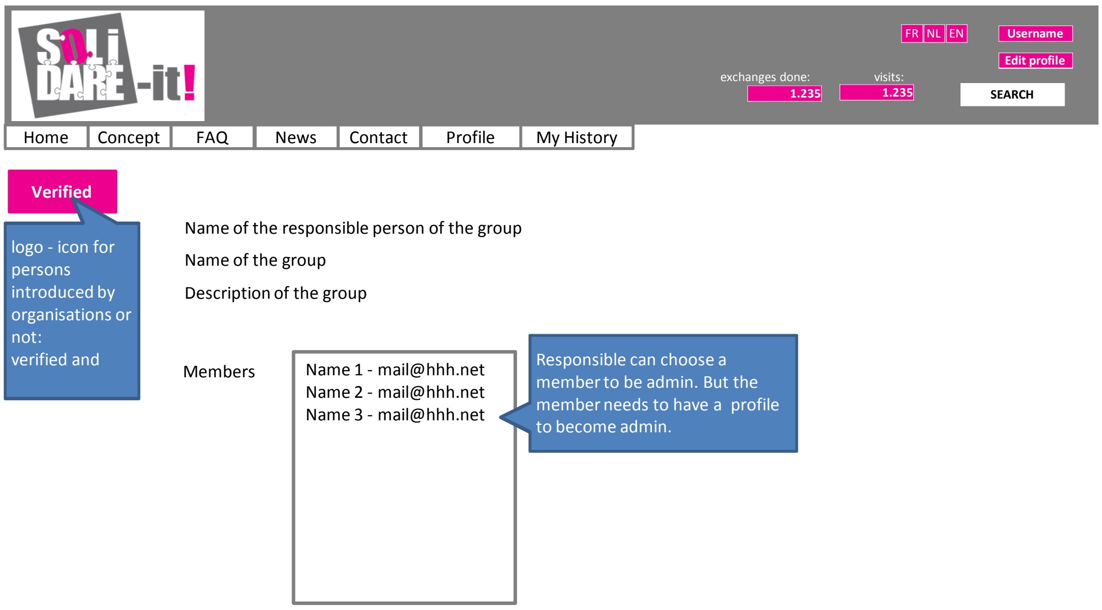

FR NL EN
Username
Edit profile
exchanges done: 1.235
visits: 1.235
SEARCH
Home Concept FAQ News Contact Profile My History
Verified
logo - icon for persons introduced by organisations or not: verified and
Name of the responsible person of the group
Name of the group
Description of the group
Members
Name 1 - mail@hhh.net
Name 2 - mail@hhh.net
Name 3 - mail@hhh.net
Responsible can choose a member to be admin. But the member needs to have a profile to become admin.

FIGURE 16 – Creation of a group of individuals jointly participating in an offer or a demand.

7.2.14 Evaluating an Exchange

Figure 17 shows the evaluation information that needs to be given obligatory for each accepted exchange (even if it would not be executed, in which case the user will need to explain why it was not executed in the end).

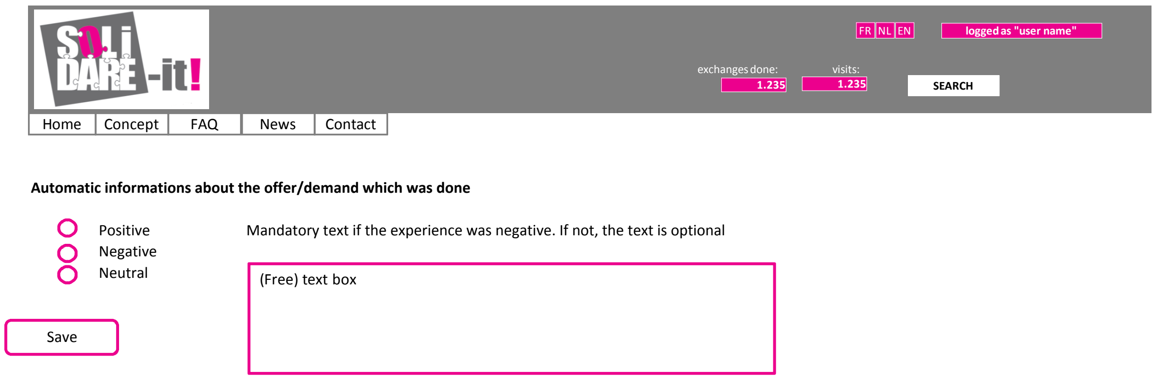
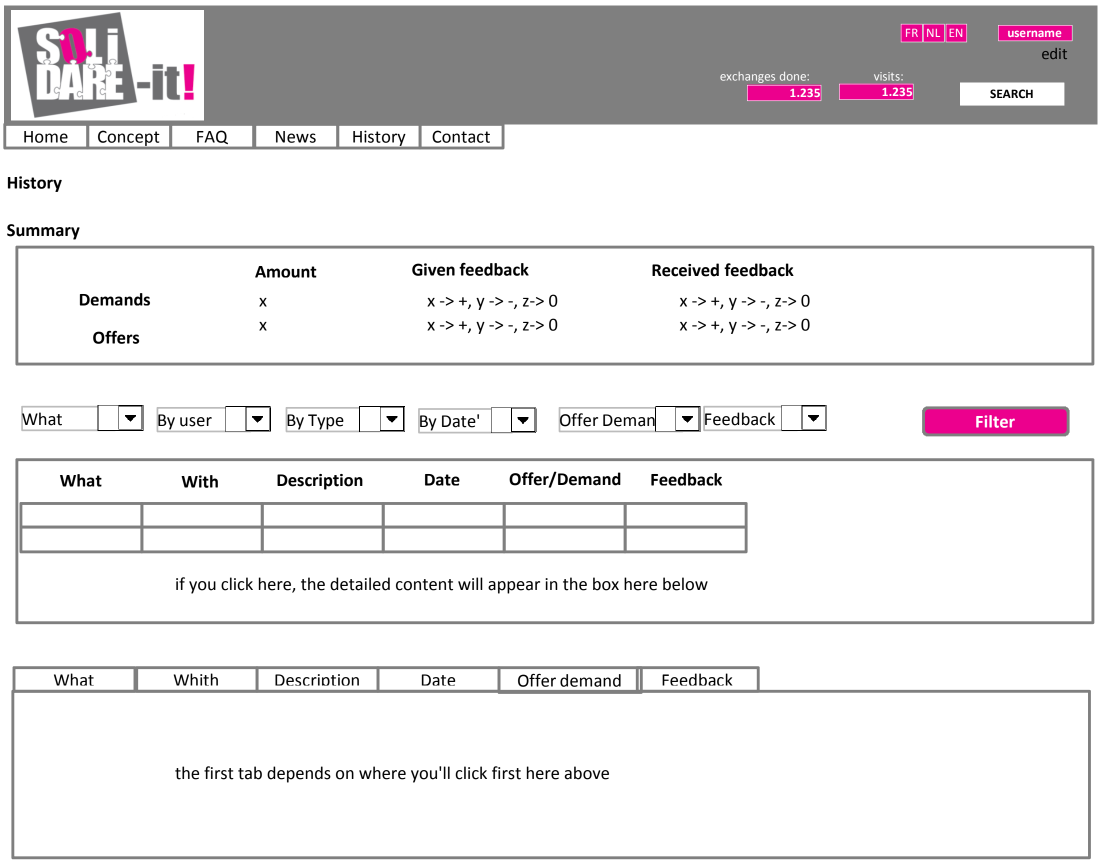

FR NL EN
logged as "user name"
exchanges done: 1.235
visits: 1.235
SEARCH
Home Concept FAQ News Contact
Automatic informations about the offer/demand which was done
Positive
Negative
Neutral
Mandatory text if the experience was negative. If not, the text is optional
(Free) text box
Save

FIGURE 17 – Evaluating a performed exchange.

The idea is to track as good as possible if an exchange is really done or not. If people have accepted an exchange, and they revisit the website later, they will be asked via a popup if they have actually done the exchange. If yes, they will be asked to evaluate it. The evaluation is mandatory but is supposed to take only a very small moment. Even if a social organisation serves as a proxy for a user, the evaluation remains obligatory. The social organisation will have to discuss the experience with the person in question anyway, and will then have to fill in the evaluation.

7.2.15 History

Finally, Figure 18 is a wireframe that describes a page giving some information on the history of an individual. There are also a few selection fields here to filter the historical information, for example, on certain types of exchanges.



History

Summary

	Amount	Given feedback	Received feedback
Demands	x	x -> +, y -> -, z-> 0	x -> +, y -> -, z-> 0
Offers	x	x -> +, y -> -, z-> 0	x -> +, y -> -, z-> 0

What	With	Description	Date	Offer/Demand	Feedback

if you click here, the detailed content will appear in the box here below

What	Whith	Description	Date	Offer demand	Feedback

the first tab depends on where you'll click first here above

FIGURE 18 – History of an individual.

The choice of having a separate history page, as opposed to merging it with a user’s profile page, was a deliberate one. The profile page will just contain a small summary of the history, but the history page may contain too much information to integrate it completely with the profile page.

Références

[1] Philippe Kruchten. The 4+1 view model of architecture. *IEEE Software*, 12(6) :42–50, November 1995.

[2] Shari Lawrence Pfleeger and Joanne M. Atlee. *Software Engineering*. Pearson Education, fourth edition, 2010.