INGI2255 - Software Engineering Project - Requirements, methodology and planning

Aurian De Potter Eddy Ndizera Ivan Ahad
Anthony Dechamps Ludovic Frastré Arnaud Dethise
Jonathan Legat

25 septembre 2015

1 Introduction

This report contains all the information about how we are going to develop the software. First, we will talk about the method and all the tools that we chose to use throughout the development. Secondly, there is the planning of all four phases of the project. Then comes our analysis of all the requirements that we were asked to put in the software. And finally, we present three use cases.

2 Software development method

We are briefly going to explain the way the group will develop the project. After being introduced to it by our customers, we had to choose between a few development methods to use throughout the making of the project. The group then picked the **Agile** method.

The first step of this method is writing down user stories. These are used to present all the functionalities of the software which is the website. Afterwards, the group has to discuss about the importance and the difficulty of each user story, and then decide which of them have the top priority.

When this is done, we will use a kanban to organize the process of the project. The kanban contains all the user stories, and is divided into the following categories : **Backlog** (it contains every user story at the beginning), **Doing this iteration**, **Done this iteration**, and **Done**". Every member of the group picks the user stories he thinks he is able to implement, so it allows us to know which member has done each task. For the kanban, we will use the Trello Website.

The programming language used will be Python, with the **Django** framework. We will use the third version of Python, and the 1.8 version of Django. Lastly, we will use the **Bootstrap** framework for the design of the website (the css part essentially).

To easily share the project's code, we will use a Control Version System, which is in this case **Mercurial**. Mercurial, and its underlying service called BitBucket, is preferred

since it allows us to make private repositories.

We also decided to divide the project into 4 iterations, corresponding to the 4 phases presented to us. With the Agile method, there will be a working deliverable for each of these phases. Through each iteration, we will have to add up functionalities to the working project, until each of them is implemented.

There are a few reasons why we picked this way of working. First of all, this method was genuinely our first choice since it is the one we are the most comfortable with, as we have been using such a method for a few projects now. Moreover, we liked its versatility since every member of the group has to be involved in every aspect of the project. Furthermore, after each iteration, we will have a meeting with our Assistant. This meeting will allow us to get a feedback of each deliverable. The feedback received will then help us to know if changes need to be done, if there are unnecessary things, or if our project is moving towards the right direction. Finally, with the Agile method, we will first implement all the "must have" functionalities, which allow us to focus on the core of the software, and then add up the functionalities with less priority.

3 Planning

The following planning gives a rough idea about when we will implement the different functionalities (found in the requirement analysis below). It is organized so that the most important functionalities are done on the first and second phases. Depending on how well we progress in the development, this planning could change. In fact, some functionalities could be postponed or implemented earlier.

Phase 1	— The user can register as a pair to a tournament by filling in the form. — The user can register for activities during a tournament (barbecue, etc) and choose preferences such as taking responsibilities and payment method — Common page for staff members — The system can create pools and match ups — After a pool has been generated, a staff member can manually reorganize it — An admin can close the registration for a tournament (and trigger the creation of pools)
Phase 2	— The user can register on a tournament alone and be matched with another player — An admin can start or close a tournament — The user can leave a comment when registering — A staff member can see and edit all the user information — The user can register a personal court as being available for the tournament — A staff member can access the courts list — Users who registered their own court can see information about players who will play on those. — A staff member can manually assign a court for each match, or modify it in case of raining — A staff member can enter or edit match results data

Phase 3	— A staff member can visualize and print the evolution of the tournament in a fashionable and displayable form (both graphic and text), step by step, using the knockoff, and print the pool matchup sheet — An admin can create an admin account — An admin can modify his credentials — An admin can create a staff account — An admin can delete an account — The user can re-use old data linked to his email address to automatically fill a registration form — The system must confirm via email addresses — A staff member can use a mail list / newsletter functionality
Phase 4	— Files sharing between staff members — Communication channel between staff members — The user can use a payment method among several options — The system must query AFT rankings — The system can create smaller pools when it rains

4 Requirement analysis

Below are the different requirements asked by the clients. We sort them depending on if it's functionnal or non-functional. Also, we added some requirements not explicitly said by the clients but that we still consider to be relevant for the project (such as the privacy stuff). **The requirements in italic are optional**.

Functional :

- User :
  - The user can register as a pair to a tournament by filling in the form,
  - The user can register to a tournament alone and be matched with another player,
  - *The user can reuse old data linked to his email address to automatically fill a registration form*,
  - The user can register for activities during a tournament (barbecue, etc) and choose preferences such as taking responsibilities and payment method,
  - The user can register a personal court as being available for the tournament,
  - *Users who registered their own court can see information about players who will play on those*,
  - *The user can use a payment method among several options*,
  - The user can leave a comment when registering

  .
- Admin :
  - An admin can start or close a tournament,
  - An admin can close the registration for a tournament (and trigger the creation of pools),
  - An admin can create an admin account,
  - An admin can modify his credentials,
  - An admin can create a staff account,
  - An admin can delete an account ,
- Staff :
  - After a pool has been generated, a staff member can manually reorganize it,
  - A staff member can manually assign a court for each match, or modify it in case of raining,
  - A staff member can use a mail list / newsletter functionality,
  - A staff member can enter or edit match result data,
  - A staff member can see and edit all the user informations,
  - A staff member can visualize and print the evolution of the tournament in a fashionable and displayable form (both graphic and text), step by step, using the knockoff, and print the pool match-up sheet,
  - A staff member can access the courts list,
  - *A staff member can manually match two users as a pair for a tournament*,
  - Common page for staff members,
  - *Files sharing between staff members*,
  - *Communication channel between staff members* ,
- System :
  - *The system can create pools and match ups*,
  - *The system must confirm via email addresses*,

    — *The system must query AFT rankings*,
    — *The system can create smaller pools when it rains*,

Nonfunctional :

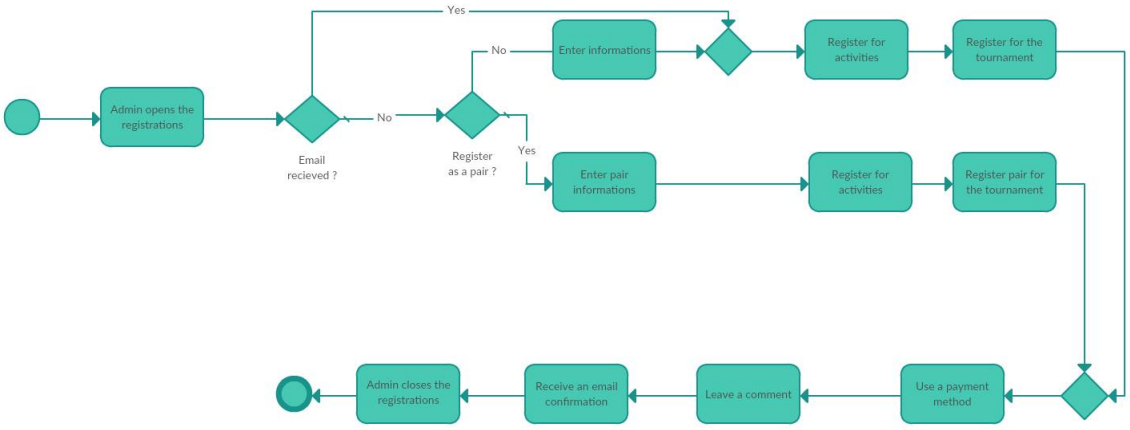
— Interface :
    — The interface must be user friendly,
    — *The interface must be mobile friendly*,
— Documentation :
    — The software must be well documented.
— Privacy :
    — Turn off indexation of private data.
— Reasons to use a new software :
    — Interface is easier to use,
    — Many tasks are automated,
    — Code base is simpler and more modular if future modifications are wanted,
    — More flexibility for users such as solo enrollment,

5 Use cases

Here are a few use cases showing how our website will react upon receiving an action (registration of a player, registration of the owner of a court and creation of a group). Note that those use cases are subject to change when developping the project but it gives a good idea about the general process.

- **Registration of a player (See Figure 1)**
  - The admin enables the registration for a tournament.
  - The user must access the registration page.
  - The user fill his personal information on the page. (Or load from the previously used email address.)
  - The user must enter the personal information of his tennis partner if he plays in duo.
  - The user gets a confirmation email. (Past this point he must contact the staff/admin to modify his information.)
  - The admin closes the registration just before the tournament.

FIGURE 1 – Diagram for registration of player



- **Registration of an owner of a court**
  - The owner must access the registration page.
  - The own fills his court information on the page.
  - The owner receives an email confirmation.
  - When the administrator opens a new tournament, he will be notified of all the previously used courts and he will ask the owners about their availability.
  - The staff will check the court and mark it as suitable.
  - When a court is assigned to a tournament, the owner is notified by an email.
- **Creation of a group**
  - Once the registrations are closed, the system will generate the groups for each category.
  - The staff will review the generated list. If there are wishes to fulfill, the list can be manually reorganized. At this point, the staff can also contact the users with payment issues.
  - Once the groups are checked, the staff confirms their composition.
  - The system sends an email to the users to inform them of the location and time of their matches and their payment status. The leader is also informed of what

is needed from him.

6 Conclusion

As discussed in the report, we presented all the elements that are required for the development of the software, and multiple information about this process, such as the method and tools chosen and the planning of the tasks to do for each intermediate deadlines.