Software Requirements Specifications
Version 1.0

Anders Deliens
Software Engineering 2013-2014 Group 1

November 14, 2013

1 Introduction

1.1 Purpose

The purpose of this SRS is to provide a detailed description of all the requirements for project Xiast. It will include a list of constraints, features and (user)interfaces.

This document is intended for the members of the development team behind Xiast as well as Professor Ragnhild Van Der Straeten and assistant Jens Nicolay.

Disclaimer: This document is a work in progress and is not yet complete. If you have any questions or suggestions, please let me know.

1.2 Scope

The application that will be discussed in this document is an online scheduling-tool for university classes, named Xiast (short for Xiast is a scheduling tool). It will compute the optimal distribution of classes, based on certain constraints, which are given by administrators and/or teachers. Students and teachers will be able to check their personal schedule online and via android smartphones. Teachers will also be able to add certain requirements to the classes and will be able to change some of these requirements or even cancel a class last-minute. The goal of this project is to make a scheduling-tool that is very personally modifiable and user-friendly.

1.3 Definitions, acronyms and abbreviations

- program-administrator = person who is able to configure every detail of the scheduling constraints: courses, classrooms, teacher, number of students, theory or practical classes...
- teacher = person who defines the details of all the classes assigned to him: assistants,required resources and facilities, assistants...
- student = someone who can register for programs and courses
- user = program administrator, teacher or student
- guest = someone who is not yet logged in and identified as user
- program = combination of multiple courses, as provided by the university
- WPO = lab sessions or exercise lessons
- HOC = theory lessons

1.4 References

IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, *IEEE Computer Society*, 1998

1.5 Overview

Section 2 will cover an overview of the general requirements. Using scenarios it will describe how the application should work. The functions that will be implemented and the user characteristics will be portrayed there, next to the constraints the application is bound by. The requirements will be further analysed and described in-depth in section 3.

This document largely follows the IEEE Std 830-1998.

2 Overall description

2.1 Product perspective

Xiast is an independent open-source product and will be free to use by anyone. Xiast aims to provide more personally modifiable rosters than other applications on the market, using a user-friendly interface. Alongside the web-application, there will be a mobile android application with some specific features and support for mobile users. Both the web and mobile application will use data stored on a database-server (Wilma).

2.2 Product functions

Xiast will accommodate three types of users: students, teachers and program-administrators. Each of these types has their own rights and own functions.

Every user will have acces to the home screen, which will contain information about Xiast, the team, the VUB... as well as an option menu where the user can choose his preferred language. There will be a button the user can click to go the log-in screen. Here the user will have to type in his username and password. Xiast will check the authenticity of the user and will determine wether the user is a student, teacher or program-administrator.

Students can register (and unregister) for a program or different courses. After a certain amount of time, no further registration will be allowed. The students will be able to check their personal schedule online and on their smartphone. They will be notified of any last-minute changes made to their schedule.

Teachers get certain courses assigned by program-administrators. They will be able to check their roster online and on their smartphone. They will be able to send their specifications regarding their courses (like maximum number of students, or the need of a overhead-projector, days they won't be able to teach...) to the program-administrator, who will try to implement these request in the scheduling, if possible. A teacher can do a scheduling for his own courses, and can mark this as a good roster for him, send it to a program-administrator, who will try to incorporate this roster in the final roster. If a teacher, for some reason, cannot make it to his scheduled class, he can notify the application, which will update the rosters for all students attending said class. Teachers can request certain changes to their personal roster, but only a program administrator is able to actually change the roster and the details of it.

Program administrators are in charge of a bundle of courses, which combined together form a standard program. They specify which courses are part of the program, how many courses their will be, which courses are obligatory and which aren't... They also assign a teacher to each course. They can make or delete courses and are able to modify every constraint regarding the scheduling.

The program administrator will input certain courses and constrains and Xiast will compute and display the best roster possible given said constraints. If there are any overlaps, Xiast will highlight them and the program administrator has the choice to either make a change manually or let Xiast try come up with alternative solutions. After any (manual) changes are made, the administrator has to give his fiat before the roster gets picked and made visible for all other users. When a new schedule is being made and an overlap between courses occurs, Xiast will try to adjust the scheduling of certain courses by ignoring their lowest priority constraints. By ignoring these low priority constraints, a new roster without overlaps may emerge. The administrator of the course which has had a change in constraints or which needs to be rescheduled will receive a notification. A program administrator may choose to change the scheduling or constraint of the courses manually or even keep the overlap.

2.3 User Characteristics

Users don't need any particular experience or expertise to use this application, just some basic knowledge of how to work with a computer and internet (on smartphones).

2.4 Constraints

Since the application must be able to run on Android devices, there is a limitation to memory. Xiast must be able to store its data in an efficient way, so that it runs smoothly, even on devices with little internal memory and low processor speeds. The schedules should still be nicely displayed on smaller smartphone screens. If the mobile network connection is unstable, the application should still be able to display the schedule, without having to reload everything.

The application has to have a Wilma back-end and a browser (computer or android smartphone) front-end

Only Java, JavaScript, HTML, CSS, SQL and associated open-source frameworks and libraries may be used as programming language. Only free and open-source software may be used for this product.

The application has to be easy to install and the user interface must be appealing and simple.

2.5 Assumptions and dependencies

To use this application, an internet connection is required. This connection is necessary for the application to fetch data from the server. If a user doesn't have access to internet, he will still be able to check previously saved schedules.

3 Specific Requirements

This section will be updated in the future with flow-charts, graphics and mock-ups to clarify certain requirements, as well as improve the readability.

3.1 External interfaces

The only link with an external system is the one to the Wilma back-end server, which contains all the scheduling data.

3.2 Functional Requirements

This next section will contain a detailed list of all functional requirements that need to be implemented, divided into sections for every user class. Every requirement will have a unique ID, which makes it easier to make references. The ID consist of FR (=Functional Requirement) followed by the first letter of the user class (G=Guest;U=User;S=Student;T=Teacher and P=Program-administrator) and then a number.

User Class 1: Guest

Functional Requirement 1.1 **ID**: FRG1
TITLE: Log in
DESCRIPTION: When a guest correctly enters the username and password of an existing account, he will be logged in, giving him the appropriate rights.
PRECONDITION: "Home" screen is showing.
SCENARIO: A guest clicks on the log in tab. The log in screen is showing. The guest enters his username and password and clicks the "log in" button.
The system logs the guest in as user and gives him/her the appropriate rights and shows the appropriate "Account" screen.
EXCEPTIONS: At any point in this scenario, the guest can click on any of the other tabs on the home screen. In this case, the guest will be redirected to the appropriate screen. If a guest enters

the wrong password or username, the system will display an error message stating that either the password was incorrect or the username doesn't exist. The guest will be than be redirected to the log in screen, where he can try to log in again.
POSTCONDITION: The guest is now logged in as a user and the system is showing his/her "Account" screen.

Functional Requirement 1.2 **ID**: FRG2 (see also FRU2)
TITLE: Cycling tabs
DESCRIPTION: Guests can cycle through the tabs on the home screen.
PRECONDITION: "Home" screen is showing
SCENARIO: Guests can click on any of the tabs on top of the home screen and the system will direct them to the correct page.
EXCEPTIONS: None
POSTCONDITION: The appropriate screen is showing.

Functional Requirement 1.3 **ID**: FRG3
TITLE: Choosing language
DESCRIPTION: Guests can choose their language by clicking on the language tab.
PRECONDITION: Home screen is showing
SCENARIO: A guest clicks on the language tab. A list of all available language will be shown to the guest and he will be able to choose his preferred language.
EXCEPTIONS: If the guest clicks on another tab before choosing the preferred language, he will be redirected to the appropriate page, without any changes to the language.
POSTCONDITION: The guest is returned to the screen he was using before clicking on the language tab, and this screen is now translated to the preferred language. All the other pages of the application will also be translated to the preferred language.

User Class 2: User All the functional requirements that follow, will apply to all three types of users: students, teachers and program-administrators.

Functional Requirement 2.1 **ID**: FRU1
TITLE: Log out
DESCRIPTION: Users can log out, becoming a guest.
PRECONDITION: User is logged in.
SCENARIO: Whenever a user clicks the log out button, he will be asked to confirm this decision and after confirmation he will be logged out. He will be returned to the home screen as a guest.
EXCEPTIONS: If the guest clicks cancel after clicking the log out button, he stays logged in, remaining on the same page he was on before clicking the log out button. If the user clicks another tab after clicking the log out button, the system will interpret this as a cancel log out command.
POSTCONDITION: The user is logged out and is now a guest.

Functional Requirement 2.2 **ID**: FRU2 (see also FRG2)
TITLE: Cycling tabs
DESCRIPTION: Users can cycle through the tabs on the home screen.
PRECONDITION: "Home" screen is showing
SCENARIO: Users can click on any of the tabs on top of the home screen and the system will direct them to the correct page.
EXCEPTIONS: None
POSTCONDITION: The appropriate screen is showing.

Functional Requirement 2.3 **ID**: FRU3
TITLE: Viewing schedule.
DESCRIPTION: Users can view their personal schedule and filter the display of the schedule with different modifiers.
PRECONDITION: None

SCENARIO: Users click on the view button and will be redirected to the view page. Here they will be able to see their schedule. They can denote the month, week or day they want to see as well as which program or course they want to view. They can also choose to see the schedule of one course or of one classroom.
EXCEPTIONS: When there is no schedule to be shown, a message will inform the users. When no specifications are made for the time to be displayed, the current week will be shown.
POSTCONDITION: The correct personal schedule is being displayed.

User Class 3: Student

Functional Requirement 3.1 **ID**: FRS1
TITLE: View courses
DESCRIPTION: Students can take a look at all the courses they follow.
PRECONDITION: None
SCENARIO: When a student clicks on the "Courses" button, he will be redirected to a page where a list of all his registered courses will be displayed.
EXCEPTIONS: If the student should not have any registered courses, a message will notify the student.
POSTCONDITION: The correct list of registered courses of the student is displayed.

Functional Requirement 3.2 **ID**: FRS2
TITLE: Search for course or program
DESCRIPTION: Students can search a course or program
PRECONDITION: Students must be on the "Courses" page in order to perform a search.
SCENARIO: The student can type in a query inside the search box on the "courses" screen and a list of all courses/programs matching the query will be displayed. There will be an option for the student to limit his search to courses from certain programs, or years only.
EXCEPTIONS: When there are no courses/programs that match the search criteria, a notification will alert the student.
POSTCONDITION: A list matching the search criteria will be displayed.

Functional Requirement 3.3 **ID**: FRS3
TITLE: Registering for a course or program
DESCRIPTION: Students can register for courses and programs
PRECONDITION: Students must be on the "Courses" page in order to register for courses or programs.
SCENARIO: If a student wants to register for a course, he searches this course using the search box. When the course is found, the student can click a "plus"-symbol next to the course. A pop-up will ask for confirmation and after the confirmation the student will be registered.
EXCEPTIONS: When there are no courses that match the search criteria, a notification will alert the student. If a student presses cancel he will not be registered for the course.
POSTCONDITION: The student is now registered for the course/program.

Functional Requirement 3.4 **ID**: FRS4
TITLE: Unregister for a course or program
DESCRIPTION: Students can unregister for courses and programs.
PRECONDITION: Students must be on the "Courses" page in order to register for courses or programs and must be registered to at least one course.
SCENARIO: If a student wants to unregister for a course, he searches this course in the list of his registered courses on his "courses" screen. When the course is found, the student can click a "minus"-symbol next to the course. A pop-up will ask for confirmation and after the confirmation the student will be unregistered.
EXCEPTIONS: When a student presses cancel he will not be unregistered.
POSTCONDITION: The student is no longer registered for the course/program.

Functional Requirement 3.5 **ID**: FRS5
TITLE: Viewing course details
DESCRIPTION: Students can view the details of courses.
PRECONDITION: Students must be on the "Courses" page in order to view the details of a course.
SCENARIO: If a student wants to view the details of a course, he searches the course in the list of his registered courses on his "courses" screen or by using the search box. When the course is found, the student can click on the course and the description of the course will be displayed. By clicking the close button, the student can return to the "courses" screen.
EXCEPTIONS: When the description of the course is empty, a message will notify the student.
POSTCONDITION: The student is viewing the course details.

User Class 4: Teacher

Functional Requirement 4.1 **ID**: FRT1
TITLE: View courses
DESCRIPTION: Teachers can take a look at all the courses that are assigned to them.
PRECONDITION: None
SCENARIO: When a teacher clicks on the "Courses" button, he will be redirected to a page where a list of the courses that are assigned to him, will be displayed.
EXCEPTIONS: If the are no courses assigned to the teacher, a message will notify him of that.
POSTCONDITION: The correct list of all assigned courses will be displayed.

Functional Requirement 4.2 **ID**: FRT2
TITLE: Last-minute cancelling
DESCRIPTION: Teachers can cancel a scheduled course last-minute.
PRECONDITION: A course must be scheduled and the teacher must go to his "view" screen.
SCENARIO: When a teacher can't make it to a certain class due to unexpected circumstances, he can cancel that course by going to the "view" screen. He searches for the course that needs to be cancelled in the schedule and clicks it. A pop-up will appear asking the teacher to confirm cancellation or to go back without cancelling. This cancellation is immediately visible for all the students attending this course.
EXCEPTIONS: When the teacher aborts the cancellation, nothing happens and he is returned to the "view" screen.
POSTCONDITION: The course is no longer scheduled and this change is visible in all schedules containing this course.

Functional Requirement 4.3 **ID**: FRT3
TITLE: Edit details of course.
DESCRIPTION: Teachers can change the details of a course.
PRECONDITION: A course must be assigned to the teacher.
SCENARIO: The teacher goes to his "courses" screen and can click on any of the courses assigned to him. This will bring forth another screen with different sections. One of these sections will allow the teacher to adjust the description of the course (which can be read by the students, see FRS5). Another section will contain a list of possible facilities needed for the course. The teacher selects whether he wants to adjust the facilities of the WPO or the HOC (since these can differ quite a lot). Another section will contain his preferences for how this course should be scheduled: which days, classrooms.. he prefers. After making changes to either the description, the facilities or his preferences, the teacher has to hit a save button.
EXCEPTIONS: The teacher can hit the cancel adjustments button, so no changes will occur.
POSTCONDITION: The course description, facilities and preferences are changed and updated correctly.

Functional Requirement 4.4 **ID**: FRT4
TITLE: Teacher auto-scheduling

DESCRIPTION: Teachers can order a scheduling of their own courses and mark one as a "proposal".
PRECONDITION: Courses must be assigned to the teacher.
SCENARIO: The teacher can go to his "view" screen, where his current schedule will be shown. On top of the screen there will be a "schedule" button. When the teacher presses this button, Xiast will perform a scheduling of all the courses assigned to the teacher, taking the preferences and facilities of each course into account. The new roster will be displayed and the teacher can cycle between his current schedule and the new schedule by clicking on either the "current" button or the "schedule" button. If the scheduling doesn't contain any conflicts or overlaps, the teacher can mark this schedule as a "non-conflicting proposal".
EXCEPTIONS: If the newly scheduled roster contains overlaps, the teacher will not be able to mark it as a "non-conflicting proposal". Instead, it will be marked with "conflicting proposal" so that it remains clearly visible that there are still conflicts in this schedule. The conflicts will be highlighted in order to make them more visible.
POSTCONDITION: A newly scheduled roster is marked as "(non-)conflicting proposal" and this proposal is visible for the program-administrator.

Functional Requirement 4.5 **ID**: FRT5
TITLE: Teacher manual adjustments
DESCRIPTION: Teachers can manually adjust their schedule.
PRECONDITION: Courses must be assigned to the teacher. The teacher has done a scheduling.
SCENARIO: The teacher can change the times when the courses are being scheduled in order to remove certain existing overlaps. If conflicts are resolved, the courses lose their highlighted state. The teacher can mark the schedule as "non-conflicting proposal".
EXCEPTIONS: Conflicts occur, the teacher cannot mark the schedule as "non-conflicting proposal". Instead, it will be marked with "conflicting proposal".
POSTCONDITION: Manual changes have been made and the new roster is marked as a "(non-)conflicting proposal".

User Class 5: Program-administrator

Functional Requirement 5.1 **ID**: FRP1
TITLE: View programs and courses
DESCRIPTION: Program-administrators can look at all the programs that they have to control.
PRECONDITION: None
SCENARIO: When a program-administrator clicks on the "Programs" button, he will be redirected to a page where a list of the programs that he is in control of, will be displayed. If he presses on one of the programs, a list containing all the courses of that program will be displayed.
EXCEPTIONS: If the are no programs or if there are no courses in a program, a message will notify the program-administrator.
POSTCONDITION: The correct list of all the programs under control of the program-administrator will be displayed.

Functional Requirement 5.2 **ID**: FRP2
TITLE: Add Program
DESCRIPTION: Program-administrators can add a program to their repertoire.
PRECONDITION: None
SCENARIO: When a program-administrator wants to add a program, he goes to his "Programs" view and presses on the "plus"-symbol. A pop-up window shows up, where he has to fill in a name for the program. After hitting the save button, the program is created.
EXCEPTIONS: When the program-administrator presses cancel, no program is created and he is redirected to the "Programs" view.
POSTCONDITION: A new program is created.

Functional Requirement 5.3 **ID**: FRP3
TITLE: Add Course
DESCRIPTION: Program-administrators can add a course to one of their programs.
PRECONDITION: The program-administrator should have at least one program under his control.
SCENARIO: When a program-administrator wants to add a course, he goes to his "Programs" view and presses on the program where he wants to add the course. Then he presses on the "plus"-symbol. A form appears on the screen, where he can fill in the name of the course, assign a teacher to it, state how many hours the course has, note if its an obligatory course or an optional one, define how many hours are WPO and how many are HOC. After hitting the save button, the course is added to the program.
EXCEPTIONS: When the program-administrator presses cancel, the course isn't added and he returns to the "program" screen.
POSTCONDITION: The new course is added to a program.

Functional Requirement 5.4 **ID**: FRP4
TITLE: Edit program/course
DESCRIPTION: Program-administrators can edit the courses inside their programs.
PRECONDITION: The program-administrator should have at least one program under his control, containing a minimum of one course.
SCENARIO: The program-administrator goes to his "Programs" screen and clicks on the program he wants to edit. The list of courses of this program is shown. The program-administrator can click on the "minus"-symbol to remove the course from the program, or he can click on the course. He then has the option to click on "edit course" to change every detail of the course. After hitting save, the form is updated.
EXCEPTIONS: When the program-administrator presses cancel, the course isn't changed and he returns to the "program" screen. **POSTCONDITION**: Course details are changed and/or courses are removed from a program.

Functional Requirement 5.5 **ID**: FRP5
TITLE: View preferences/facilities courses
DESCRIPTION: Program-administrators can see the preferences and facilities required for each course in their program.
PRECONDITION: The program-administrator should have at least one program under his control, containing a minimum of one course.
SCENARIO: The program-administrator goes to the "Programs" screen and clicks on the program which contains the course of which he wants to see the details of. He then presses on the course and then hits the option "view preference/facilities". A pop-up window shows the preferences and facilities put forward by the teacher of the course. By hitting back, the program-administrator can return to his "Programs" view.
EXCEPTIONS: When the teacher of the course hasn't filled in the preferences and facilities of the course, the program-administrator will be alerted with a notification.
POSTCONDITION: Course preferences and facilities are shown on the screen.

Functional Requirement 5.6 **ID**: FRP6
TITLE: Auto-scheduling
DESCRIPTION: Program-administrators can command Xiast to perform a scheduling on the courses of their programs and mark it as "final".
PRECONDITION: The program-administrator should have at least one program under his control, containing a minimum of one course.
SCENARIO: The program-administrator goes to his "Programs" screen and clicks on the program he wants to schedule. He clicks on "Schedule" and he is redirected to the "Schedule" view. Xiast shows the best schedule it could compute. If the program-administrator approves of this schedule he can mark this schedule as "final" (or as "non-conflicting proposal"). Now this schedule will become visible for every user.
EXCEPTIONS: If the newly scheduled roster contains overlaps, the schedule cannot be marked

as a “non-conflicting proposal”. Instead, it will be marked with “conflicting proposal” so that it remains clearly visible that there are still conflicts in this schedule. The conflicts will be highlighted in order to make them more visible.
POSTCONDITION: The new “final” schedule is available for everyone or the newly scheduled roster is marked as “non-conflicting proposal”.

Functional Requirement 5.7 **ID**: FRP7
TITLE: Manual scheduling
DESCRIPTION: Program-administrators can manually perform changes to a schedule.
PRECONDITION: The program-administrator should have at least one program under his control, containing a minimum of one course.
SCENARIO: The program-administrator can change the times when the courses are being scheduled in order to remove certain existing overlaps. If conflicts are resolved, the courses lose their highlighted state. He can mark the schedule as “non-conflicting proposal” or as “final”.
EXCEPTIONS: Conflicts occur, the schedule cannot be marked as “non-conflicting proposal”. Instead, it will be marked as “conflicting proposal”.
POSTCONDITION: Manual changes have been made and the new roster is marked as a “(non-)conflicting proposal”.The program-administrator goes to his “Programs” screen and clicks on the program he wants to schedule. He clicks on “Schedule” and he is redirected to the “Schedule” view. Xiast shows the best schedule it could compute. If the program-administrator approves of this schedule he can mark this schedule as “final”. Now this schedule will become visible for every user.
POSTCONDITION: The new “final” schedule is available for everyone or the newly scheduled roster is marked as “non-conflicting proposal”.

3.3 Performance requirements

No particular performance related objectives were put forward for this project, however the aim of this project is to make Xiast as performant as possible. The number of students should not be of any concern, since students don’t communicate with each other. The number of teachers or program-administrator also doesn’t influence the performance of the application, however time to negotiate between different teachers and program-administrators to come up with a solution for overlaps will increase, since they have to reach a consensus about the subject.

3.4 Design constraints

There are no design constraints except that the application must be able to run on android devices and only free, open-source software/programming languages may be used.

3.5 Software system attributes

Reliability The application should work the way it is intended to work at all time. We aim not to have more than 50 errors the first year after launch and the number of errors should decrease over the years.

Availability The application is available for the user as long as it is installed on their smartphone or as long as they have access to a web-browser. Since this projects works with an external server (Wilma), availability of the database cannot be regulated by any of the team-members. However, the goal is to have as less down-time as possible. Should there be a server failure, the team will contact the person responsible for the server and try to help get it back up as soon as possible.

Security The database will be protected from SQL-injections.

Portability The application will be available for android devices and any web-browser.

3.6 Other requirements

The application should at least be available in Dutch and English.