SubIt!

Group 1

Joel Westberg
Mikael Granholm
Simon Stenström
Sofie Björk
Henrik Eriksson Hegardt

Contents

1 Preface

1.1 Should You Read This Document?

SubIt! is a coursework submission system for students and teachers at a university. This document defines the requirements and functions of the software project. It will be most helpful to you if any of the following applies to you:

- you are involved in the development of SubIt!
- you are involved in the testing of SubIt!
- you are a system maintenance engineer at a university which uses SubIt!
- you are in the management at a university thinking about implementing SubIt! in your system environment.

2 Introduction

2.1 The need for the system

The system SubIt! will be primarily used by students and teachers at a university, though it could potentially also be used elsewhere. It will also be used by system administrators to a lesser extent, who will be able to maintain the data in the system. It will help students to submit their homework and lab assignments, and it will help the teachers grade those assignments. Students will no longer have to worry about where to send their homework, or if it is OK to hand it in by e-mail, and they no longer have to wonder what file format the teacher accepts, because the teacher will have to specify this.

The teacher will not have to think about where the students may have turned in their assignments or when. SubIt! will keep track of when assignments are turned in and who has turned their assignment in to late. It will list all the assignments and store them in one place so that the teacher can easily reach them and keep track of them. The system will make no distinction between *teachers* and *teacher assistants*.

2.2 SubIt!s functions

SubIt! is to be used as an organization tool for students and teachers. The system will, for a student, list his courses and let him know what assignments he is to turn in when, and in what file format. It will also tell him what the assignment is and of course allow the student to upload the assignment. The system is meant to be used in all courses at the university. A teacher using SubIt! will see the courses he/she teaches, and let him/her create new assignments using a standard template with fields such as Description, Deadlines and Accepted file formats. For each assignment created, the teacher will be able to list the reports handed in from students, see when the reports were turned in, and set a grade on each specific report.

2.2.1 Usage Narrative 1

Martin has just spent all day finishing his homework assignment in one of the courses he is reading, Numeriska Metoder. Exhausted, but pleased with what he has accomplished, all that is left is to hand it in. Gerd, his teacher, has for this course decided to use SubIt!, a system for handing in assignments over the internet. Martin opens up a web browser on his computer, and points it towards his university's SubIt! website. On the page that pops up, he is asked to identify himself, and he quickly supplies his username and password into the system.

Now logged in, SubIt! lists the courses in the system that Martin is currently a student in. He selects Numeriska Metoder from the list and is

presented with a list of the assignments in that course. He sees at a glance that the homework he submitted last week has been graded. Having been a bit worried about his reasoning on one of the questions, he is relieved to see that he Gerd has given him a passing grade.

The next assignment, he sees, is due in tomorrow. Clicking on the assignment in question, he is given the option to submit it. He is informed that Gerd has chosen to only accept pdf files for this assignment. Martin, having looked this up previously, has already saved his report in the pdf format. He submits the pdf file, and the submission is confirmed as handed in by the system. Satisfied, he logs out of the system and turns off his computer screen.

2.2.2 Usage Narrative 2

Gerd knows the last date for submission for an assignment in her course in Numeriska Metoder is tomorrow, but she is curious to see if anyone has submitted their report already. Hoping that she might be able to get a head start on the grading of this assignment, she logs into the SubIt! system. The system recognizes her as a teacher in a number of courses, including Numeriska Metoder, but also as a student in the course Italienska fortsättningskurs.

Gerd selects her Numeriska Metoder course, and sees a list of the assignments she has set up for this course. Should she wish to, she could also add extra assignments to the course. She selects the assignment due in tomorrow. The students who have already submitted their reports are listed together with a link to their submission. Gerd clicks on Martin's submission and the files Martin has submitted for that assignment are listed. She clicks the download button and receives Martin's PDF file. A while later, when Gerd has read Martin's report, she again logs in to the system and grades Martin's assignment.

2.3 How the System fits in with the Business or Strategic Objectives of the Commissioning Organization

Most universities receive money based on how many students they have. The management want their students to be happy with their university and they want new students to apply to their university. Things that make a University popular among students are of course good lecturers who know what they're talking about, a nice environment that the students want to spend time in, and interesting courses to study. These are the three attractive things to have, but to stand out from other places a university needs to do more than this. One thing they need, is to have a good administration. If things run smoothly, students and staff will not spend their time arguing about paperwork and registration. Instead the students can use the time to

study and actually learn something and the staff can go on to do their real work.

SubIt! solves one of the many problems a University can have with administration. Most universities today already have systems for grading and making course pages, and they probably do not want to upgrade those systems to do something it was never designed to do. For this reason, SubIt! is not integrated with other existing systems such as LADOK or First Class.

2.4 System Environment and Context

SubIt! will be used to facilitate grading and homework submission over the internet. Students and teachers alike should be able to access the application through a XHTML-compliant web browser from any operating system. The application runs on a PHP web server with an SQL database system for storing user information and homework submission details, while storing the actual files submitted on the server file system. The application and database should be able to handle several different file types which are specified by the teacher, depending on the assignment.

2.5 The Scope of the System

Topic	In	Out
Login system	X	
Encryption		X
Different user access levels	X	
Inter-user messaging system	X	
Complete course listing	X	
Ability for user to join any course	X	
Connection to other university systems		X
Connection to university login system		X
Ability to create/join project groups	X	
Multi-language support		X
Ability to see if a user is logged in		X
Discussion forum		X
Description page for a course	X	
File submission system	X	
Teacher ability to grade and comment a submission	X	
Teacher ability to see all submissions for any student	X	
Ability to export data from the system		X
Ability to delete documents		X
Ability to search among all users		X

2.6 Main factors in development of the system

In a system that allows the students of a university course to submit all their coursework through one easy-to-use submission system, there are a few factors that need to be taken into account when designing such a system.

- There must be a well designed interface that is easy to use and to understand, for teachers and students alike.
- A teacher should be able to easily see whether or not an assignment was handed in on time, or not. A student, when submitting such an assignment should also know, beyond a doubt, whether the system regards his submission as being on time or not, or else unnecessary conflict may arise.
- Reliability. The system should be reliable to use. No data loss should be allowed to happen.
- Security. The system should be secure from attacks. For instance, should the system not make it possible for someone to gain access to other user's accounts or submit files in their name.
- Deployability. The system should be fairly easy to deploy when completed so that it can be installed easily on many different universities.

2.7 Technologies and risks

Required Technology A server running the Apache Web server, with PHP5 and MySQL installed.

We have chosen to write this project in PHP. By using PHP and a simple browser-based interface we limit the project work load since we do not have to create both a server and a client interface. Furthermore our users won't have to download an application to be able to upload their assignments and will be able to do it from any computer with internet access. The MySQL database we will use to store all information about the courses and users is OpenSource and free. Yet, thanks to many commercial applications, it is well tested, supported and reliable.

During implementation there needs to be a central depository of information where all group members can upload their files. Likewise, a web server running PHP 5.2.5 where all group members have access is required during the implementation, preferably with some sort of Version Control system like SVN. Currently, we have no such server available. Besides all this we will need a model for how we should reliably store and handle all submissions on the server file system. The catalog structure will never be seen by any user, since the system itself will take care of all file handling and show the users a much more non-technical file view, but in the event

of a breakdown of the interface, or a server crash, the files should be easily recoverable.

3 Glossary

3.1 Technical Terms

3.1.1 FirstClass

A client/server groupware, email, online conferencing, voice/fax services, and bulletin-board system. It is used in both education and business. [1]

3.1.2 LADOK

A Swedish system for documenting the grades and results of university students. [2]

3.1.3 PHP

A computer programming language originally designed for producing dynamic web pages. [3]

3.1.4 Server

An application, or a device that performs services for connected clients as part of a client-server architecture. [4]

3.1.5 MySQL

A multithreaded, multi-user SQL database management system which has more than 10 million installations. The basic program runs as a server providing multi-user access to a number of databases. [5]

3.1.6 Apache Webserver

A web server notable for playing a key role in the initial growth of the World Wide Web. [6]

3.1.7 XHTML

The Extensible HyperText Markup Language, or XHTML, is a markup language that has the same depth of expression as HTML, but also conforms to XML syntax. [7]

[1] http://en.wikipedia.org/wiki/FirstClass
[2] http://sv.wikipedia.org/wiki/LADOK
[3] http://en.wikipedia.org/wiki/PHP
[4] http://en.wikipedia.org/wiki/Server_(computing)
[5] http://en.wikipedia.org/wiki/MySQL
[6] http://en.wikipedia.org/wiki/Apache_Webserver
[7] http://en.wikipedia.org/wiki/XHTML

3.1.8 XML

Is classified as an extensible language because it allows its users to define their own elements. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet. [8]

3.1.9 CSS

In web development, Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in a markup language. [9]

3.1.10 JavaScript

A scripting language most often used for client-side web development. It is a dynamic, weakly typed, prototype-based language. [10]

3.1.11 Java Applet

A Java applet is an applet delivered in the form of Java bytecode. Java applets can run in a Web browser using a Java Virtual Machine (JVM), or in Sun's AppletViewer, a stand-alone tool for testing applets. [11]

3.1.12 SVN

Subversion (SVN) is a version control system initiated in 2000 by CollabNet Inc. It allows users to keep track of changes made to any type of electronic data, typically source code, web pages or design documents. [12]

3.1.13 Database

A computer database is a structured collection of records or data that is stored in a computer system so that a computer program or person using a query language can consult it to answer queries. [13]

[8] http://en.wikipedia.org/wiki/XML
[9] http://en.wikipedia.org/wiki/CSS
[10] http://en.wikipedia.org/wiki/JavaScript
[11] http://en.wikipedia.org/wiki/Java_Applet
[12] http://en.wikipedia.org/wiki/Subversion_(software)
[13] http://en.wikipedia.org/wiki/Database

3.2 SubIt! Defined Words

3.2.1 Start Date

Is assigned to an assignment. This is the first date that the assignment is displayed to the students participating in the course, and the first date that student can submit submissions.

3.2.2 Soft Deadline

Is assigned to an assignment. This is the date that the teachers of a course want the submissions to be made. If a submission is made later than this date, it will be viewed as a late assignment.

3.2.3 Hard Deadline

Is assigned to an assignment. This is the last date to hand in a submission. After this date, it is no longer possible to submit the assignment.

3.2.4 Teachers

Teachers teach something in a course. The can be assistants or co-lecturers or laboratory assistant.

3.2.5 Course Leader

There is only one course leader for each course. This is the one teacher who decides who the other teachers are and who decides about assignments.

3.2.6 Assignment

The question/problem/subject that the students are to answer/solve/write about.

3.2.7 Submission

The file/files that a single student hands in.

4 User requirements

4.1 Functional Requirements

4.1.1 Any user

1.1 Shall be able to list and search amongst the different courses available.
Behavior described in UC7.

1.2 Shall be able to join a course as a student.
Behavior described in UC7.

1.3 Shall be able to view the course description for any course.
Behavior described in UC7.

1.4 Shall be able to send messages to users and project groups.
Behavior described in UC3.

1.5 Shall be able to read any received messages.
Behavior described in UC4.

1.6 Shall be able to read details regarding the assignments in a course.
Behavior described in UC9.

1.7 Shall be able to list the members of a project group.
Behavior described in UC5.

1.8 Shall be able to identify and authenticate themselves to the system.
Behavior described in UC1, UC2, UC6.

1.9 Shall be able to list the courses in which they are active as a student, teacher or course leader. *UC1*

4.1.2 Students

2.1 Shall be able to announce an end to his/her participation in a course.
Behavior described in UC8.

2.2 Shall be able to announce an end to his/her inactivity in a course.
Behavior described in UC7.

2.3 Shall be able to make submissions in courses he/she is enrolled in.
Behavior described in UC9.

2.4 Shall be able to submit several different files to the same assignment.
Behavior described in UC9.

2.5 Shall be able to write a comment on a submission.
Behavior described in UC9.

2.6 Shall be made aware when his/her submission has been graded.
Behavior described in UC14.

2.7 Shall be able to read teacher's comments on his/her submissions.
Behavior described in UC10.

2.8 Shall be able to view his/her own submissions.
Behavior described in UC10.

2.9 Shall be able to create or join a project group in a course.
Behavior described in UC11.

2.10 Shall be able to leave a project group.
Behavior described in UC12.

2.11 Shall be able to make a submission on behalf of a project group.
Behavior described in UC9.

4.1.3 Teachers

3.1 Shall be able to list all students in a course.
Behavior described in UC13.

3.2 Shall be able to view a submission.
Behavior described in UC13.

3.3 Shall be able to list the submissions of a student.
Behavior described in UC13.

3.4 Shall be able to grade a submission.
Behavior described in UC14.

3.5 Shall be able to comment on a submission.
Behavior described in UC14.

3.6 Shall be able to get an overview of submissions and associated grades.
Behavior described in UC15.

4.1.4 Course leaders

4.1 Shall be able to add and remove teachers from the course.
Behavior described in UC16 and UC17.

4.2 Shall be able to change course description.
Behavior described in UC21.

4.3 Shall be able to create an assignment with a specified name.
Behavior described in UC18.

4.4 Shall be able to set a description for an assignment.
Behavior described in UC18 and UC19.

4.5 Shall be able to set a specified file type for an assignment.
Behavior described in UC18 and UC19.

4.6 Shall be able to set a start date for an assignment.
Behavior described in UC18 and UC19.

4.7 Shall be able to set a soft deadline for an assignment.
Behavior described in UC18 and UC19.

4.8 Shall be able to set a hard deadline for an assignment.
Behavior described in UC18 and UC19.

4.9 Shall be able to change an assignment at any time.
Behavior described in UC19.

4.10 Shall be able to group assignments.
Behavior described in UC20.

4.11 Shall be able to send messages to the whole course.
Behavior described in UC22.

4.1.5 System administrator

5.1 Shall be able to create, remove and edit users.
Behavior described in UC23, UC24 and UC25.

5.2 Shall be able to create and remove courses.
Behavior described in UC26 and UC27.

5.3 Shall be able to set a course as active or inactive.
Behavior described in UC26.

5.4 Shall be able to set a course leader in a course.
Behavior described in UC28.

4.2 Non-functional requirements

1 The system should be entirely in comprehensible English.

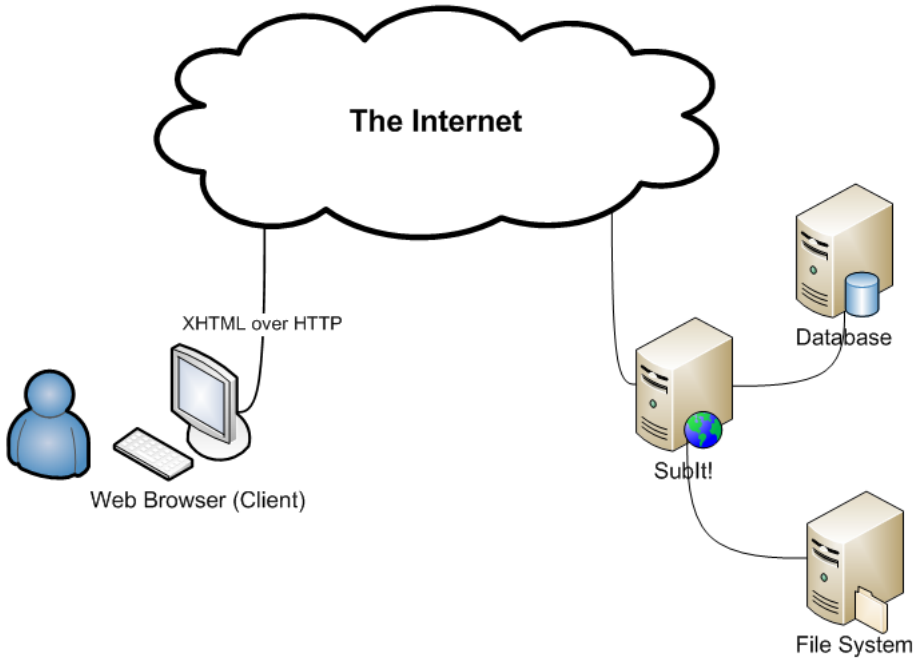
2 Course leaders shall have the same privileges as teachers.

3 Any user can be student, teacher or course leader in any other course.

4 The system requires an administrator.

5 The system shall be able to safely identify the user.

5 System architecture



The image above describes the relations between the different parts of the system. The client connects to the SubIt! system through the Internet. The system is divided into a database part and a file system part.

The database part handles all information that is stored in the system. It contains the assignment descriptions and all the user information.

The file system contains all uploaded submissions. The system it self keeps track of the file structure but in case of a system crash, an administrator can also recover the files from here.

6 System requirements

6.1 User types

There should be 2 types of users, regular users and system administrators.

> **Rationale**: The system administrator need to have the right to perform options such as adding new users to the system or creating a new course. This should be well outside the scope of the majority of the users of the system.

Priority: 4 *Critical* *(Requirement #1)*

Relates to *Non-Functional User Requirement 3.*

Relates to *Non-Functional User Requirement 4.*

6.1.1 Types of regular users

Any user of the system who is not a system administrator will have certain rights depending on what they are in relation to a course; Student, Teacher or Course Leader.

> **Rationale**: In order to divide the user group further and make sure that a student can't see another students submission, while still giving teachers the right to view and grade those same submissions. A teacher is not always a course leader, and sometimes he or she might even be a student in a different course. For that reason, these attributes have to be according to course, and not a global user level.

Priority: 4 *Critical* *(Requirement #2)*

Relates to *Non-Functional User Requirement 3.*

6.1.2 The System Administrator

The system administrator is a regular user, but with extra privileges that allow him to administer the system.

> **Rationale**: There is no reason not to allow a system administrator to also act as a regular user. A system administrator might very well also be a teacher.

Priority: 3 *High* *(Requirement #3)*

Relates to *Non-Functional User Requirement 4.*

6.1.3 The Course Leader

In any course, the course leader will always be able to do the same things a teacher of the course can do.

> **Rationale**: A course leader is, normally, a teacher with the privileges to make changes to the course. Thus, he should always have all abilities of a teacher.

Priority: 4 *Critical* *(Requirement #4)*

Relates to *Functional User Requirement 2.*

6.2 Log in system

All users should be able to safely log in to and be authenticated by the system.

> **Rationale**: For the system to be safe, and for a user to be able to rely on that no one else can upload files or read submissions as that user, the system needs some way to authenticate a user.

Priority: 3 *High* *(Requirement #5)*

Relates to *Functional User Requirement 1.8.*

6.3 List active courses

All users should be able to list all currently active courses, and search within that list.

> **Rationale**: In order to find courses the user wants to join, or otherwise find information regarding a course, it is necessary that a searchable list of all currently active courses in the system can be presented.

Priority: 2 *Medium* *(Requirement #6)*

Relates to *Functional User Requirement 5.3.*

6.4 Join a course

Any user should be able to, at any time, join any active course in the system as a student, if they are not already active in the course.

> **Rationale**: No courses can have limited registration. It would be up to the teacher of a course to impose such restrictions should he feel it is necessary, and if so that would be outside the scope of the system.

Priority: 4 *Critical* *(Requirement #7)*

Relates to *Functional User Requirement 1.2*.

6.5 Read course description

Any user should be able to read the description for any course.

> **Rationale**: If nobody can read the description, there's no point in having them. Also it will help students join the right course if they can read the description.

Priority: 3 *High* *(Requirement #8)*

Relates to *Functional User Requirement 1.3*.

6.6 List courses user is active in

Any user should be able to view a list of courses he/she is active in as a student, teacher or course leader.

> **Rationale**: To make the system easy to use, a list should be available for the users to see which courses they are currently participating in.

Priority: 3 *High* *(Requirement #9)*

Relates to *Functional User Requirement 1.9*.

6.7 Announce inactivity

Any user may at any time declare themselves as inactive in any course. This will not remove the user from the course roster, but the user will no longer receive information from the course, or have it listed among the courses he/she is active in.

> **Rationale**: If a student who has participated in a course no longer wishes to continue his/her participation, he/she should be able to become inactive in the course, and thus no longer receive news or such information about the course. Since the student is not removed from the roster, no submissions will be lost.

Priority: 1 *Low* *(Requirement #10)*

Relates to *Functional User Requirement 2.1*.

6.8 Denounce inactivity

Any user who is inactive in a course should be able to revert back to active status.

Rationale: If you've become inactive by accident, or simply change your mind afterwards, it seems appropriate to be able to reverse it.

Priority: 1 *Low* *(Requirement #11)*

Relates to *Functional User Requirement 2.2.*

6.9 View details about an assignment

Any user should be able to view the details of any assignment available in any course.

Rationale: There should be no limitations to who gets to see what in regard to information regarding assignments in any course.

Priority: 3 *High* *(Requirement #12)*

Relates to *Functional User Requirement 1.6.*

6.10 Submissions

6.10.1 Submit an assignment

Any student of a course should be able to submit files to any assignment, as long as submissions are accepted and the submission meets any other requirements specified for that assignment.

Rationale: While submission is allowed, it is critical that submissions can be made by any registered students of that course. No restrictions on how many files a student can upload will be imposed, as a teacher can sometimes ask a student to clarify something in his/her submission before receiving the grade. In some cases, multiple files might be required for other reasons as well, as a course leader might require both a report and source code for a certain assignment.

Priority: 4 *Critical* *(Requirement #13)*

Relates to *Functional User Requirement 2.3.*

Relates to *Functional User Requirement 2.4.*

6.10.2 Date a submission

When a file is submitted, the exact time and date of submission should be saved.

Rationale: The teacher will in some cases want to know if the submission was handed in on time before a soft deadline or not. This will make sure that this information can be retrieved by the teacher.

Priority: 3 *High* *(Requirement #14)*

Relates to *Functional User Requirement 3.2.*

6.10.3 Student comment on a submission

A student should, when he is handing in his submission, be able to leave a comment for the teacher who will view the submission.

Rationale: Sometimes a student might want to leave some comment for the teacher who will grade his submission, and so that option should be provided. It is entirely up to the teacher whether to read it or not.

Priority: 2 *Medium* *(Requirement #15)*

Relates to *Functional User Requirement 2.5.*

6.10.4 Student view of his/her submissions

A student should always be able to and view the files he/she has submitted in the system and all details about the submission.

Rationale: In order for the student to be able to ensure for himself that his submission came through correctly and without any corruption, he should be able to view the files. It is also important for the student to be able to see if a submission has been graded and what grade he/she got on this submission.

Priority: 3 *High* *(Requirement #16)*

Relates to *Functional User Requirement 2.7.*

Relates to *Functional User Requirement 2.8.*

6.11 Project Groups

6.11.1 Creating/joining a project group

Any user should, within any course, be able to join a project group. Should the project group the user wishes to join not exist, it will be created with the user as it's sole member. Project groups will be bound to the course in which they are created, and not usable in any other course.

Rationale: A project group may be used in some courses, where lab work or other assignments might be done in pairs or groups. A created project group will allow users of that group to act in a course as representatives of the group. If someone would want to use the same project group in multiple courses, they can easily create them there as well.

Priority: 2 *Medium* *(Requirement #17)*

Relates to *Functional User Requirement 2.9.*

6.11.2 Submit assignments as a group

Any student who is a member of a project group should be able to hand in assignments in the name of the group.

Rationale: In order to make project groups worth anything at all, it's a good thing to be able to submit files in the name of the group.

Priority: 2 *Medium* *(Requirement #18)*

Relates to *Functional User Requirement 2.11.*

6.11.3 List members of a project group

It should be possible to list all members of a project group.

Rationale: Any user should be able to see the members of project groups to see which students are member of the group.

Priority: 1 *Low* *(Requirement #19)*

Relates to *Functional User Requirement 2.12.*

6.11.4 Leave a project group

It should be possible for any user at any time to leave a project group.

Rationale: Any student should have the freedom of leaving a project group whenever he or she wishes to. There is no reason for why the system should disallow such behavior.

Priority: 1 *Low* *(Requirement #20)*

Relates to *Functional User Requirement 2.10.*

6.11.5 Delete a project group

Deletion of a project group will occur if no assignments in the groups name have been handed in, and there are no members of the group.

> **Rationale**: If a project group becomes completely empty of any members or submissions, there is no reason to store data about the group any longer.

Priority: 1 *Low* *(Requirement #21)*

6.12 View all students in a course

A teacher should have the ability to see all students who are enrolled in his course.

> **Rationale**: A teacher should be able to get a list of all students in his course, in order to make grading of a student easier.

Priority: 2 *Medium* *(Requirement #22)*

Relates to *Functional User Requirement 3.1.*

6.13 View submissions in a course

A teacher of a course should have the ability to view all details about the submissions of all students in the course he is teaching.

> **Rationale**: A teacher must be able to view submissions for the course he is teaching, or else the system is quite useless.

Priority: 4 *Critical* *(Requirement #23)*

Relates to *Functional User Requirement 3.2.*

6.14 View a student's submissions

A teacher should be able to view all submissions a specific student has made in the course he is teaching.

> **Rationale**: Being able to list the submissions for one specific student will allow the teacher to easily see if a student has completed all required coursework.

Priority: 2 *Medium* *(Requirement #24)*

Relates to *Functional User Requirement 3.3.*

6.15 View all submissions in a course

A teacher should be able to view a list of all submissions made in the course he is teaching, as well as any grades set.

> **Rationale**: It will help the teacher to grade submissions to get an overview over all assignments where he can see which of them have not yet been graded.

Priority: 2 *Medium* *(Requirement #25)*

Relates to *Functional User Requirement 3.6.*

6.16 Set a grade

A teacher should be able to grade the submission of a student.

> **Rationale**: At a university, grading of an assignment is central. Grading within the system is thus a necessity.

Priority: 3 *High* *(Requirement #26)*

Relates to *Functional User Requirement 3.4.*

6.17 Comment on a submission

A teacher should be able to comment a submission.

> **Rationale**: Sometimes a teacher might want to comment on a submission he has just graded. The system should allow this.

Priority: 1 *Low* *(Requirement #27)*

Relates to *Functional User Requirement 3.5.*

6.18 Add or remove teachers

A course leader should have the ability to add or remove any user in the system as a teacher in the course he is teaching.

> **Rationale**: A course leader is usually able to decide freely who his teachers/assistants are, and should be able to do so within the system as well.

Priority: 2 *Medium* *(Requirement #28)*

Relates to *Functional User Requirement 4.1.*

6.19 Change course description

A course leader should be able to write a course description which can be read by any user of the system.

> **Rationale**: A course description, which can contain things such as course news, should be able to be updated as often as the course leader wants.

Priority: 2 *Medium* *(Requirement #29)*

Relates to *Functional User Requirement 4.2.*

6.20 Create and edit assignments

A course leader should have complete control over all assignments in his course, as well as the ability to add new ones.

> **Rationale**: The course leader needs to be able to specify assignments as he pleases.

Priority: 4 *Critical* *(Requirement #30)*

Relates to *Functional User Requirement 4.3.*

Relates to *Functional User Requirement 4.9.*

6.20.1 Set name of an assignment

A course leader should be able to give any assignment a name.

> **Rationale**: To make it easier to identify the assignments in a course, they should be named. The course leader is best fit to decide this name.

Priority: 3 *High* *(Requirement #31)*

Relates to *Functional User Requirement 4.3.*

6.20.2 Set description for an assignment

A course leader should be able to write a description for an assignment, which can contain information such as instructions and guidelines.

> **Rationale**: Students need to know what the point of the assignment, so they know what to do and hand in.

Priority: 3 *High* *(Requirement #32)*

Relates to *Functional User Requirement 4.4.*

6.20.3 Set accepted file types

A course leader should be able to specify which file types are allowed to be submitted for an assignment.

> **Rationale**: In order to ensure the teacher receives a type of file he can read, it is helpful to be able to specify allowed file formats for assignments, and thus limiting students to those formats specified.

Priority: 2 *Medium* *(Requirement #33)*

Relates to *Functional User Requirement 4.5.*

6.20.4 Set a start date

A course leader should be able to specify a start date for an assignment, when the instructions of the assignment can first be read and submissions become possible.

> **Rationale**: Some teachers prefer to make assignments available maybe a week or so before deadline. This should be possible to do within the system as well.

Priority: 1 *Low* *(Requirement #34)*

Relates to *Functional User Requirement 4.6.*

6.20.5 Set a hard deadline

A course leader should be able to set a deadline for when submissions will no longer be accepted.

> **Rationale**: At some point a student usually can't expect to have his teacher grade his work if he hands it in too late. Within the system it should beyond at that point no longer be possible to submit anything.

Priority: 2 *Medium* *(Requirement #35)*

Relates to *Functional User Requirement 4.8.*

6.20.6 Set a soft deadline

A course leader should be able to set a soft deadline for a submission.

> **Rationale**: A soft deadline are used in some courses to indicate a time by which an assignment must be submitted in order to receive bonus points for the exam. Such a deadline should also be visible to the student.

Priority: 2 *Medium* *(Requirement #36)*

Relates to *Functional User Requirement 4.7.*

6.20.7 Group assignments

A course leader should have the ability of grouping assignments together, should he wish to.

> **Rationale**: Grouping together all the assignments in a course, and all project assignments in a course, will allow students to easier see if they have completed all the lab exercises. It will make for an easier overview of how much work is yet to be done in a course, and what type of work it is.

Priority: 1 *Low* *(Requirement #37)*

Relates to *Functional User Requirement 4.10.*

6.21 Add, edit or remove user

A system administrator should have the ability to add and remove users from the system, as well as change information about the user.

> **Rationale**: A system administrator is the only one with privileges high enough to add or remove a user from the system. He is also the only user capable of changing information about a user, such as name or personal identification number.

Priority: 3 *High* *(Requirement #38)*

Relates to *Functional User Requirement 5.1.*

6.22 Add or remove course

System administrators shall be able to add new courses to the system as well as remove them from the system. On removal, data regarding the course will be lost.

> **Rationale**: Only system administrators have the ability to add a new course into the system, and can, if necessary, delete a course from the system as well. Because of the loss of data that will occur should a course be deleted, it is essential that they alone can perform this action.

Priority: 3 *High* *(Requirement #39)*

Relates to *Functional User Requirement 5.2.*

6.23 Set a course as active or inactive

System administrators shall be able to set a course as active or inactive.

> **Rationale**: As a way to keep tabs on which courses are currently active, and for students to easily find which course they are members of, a course should either be active or inactive. The system administrator must be able to specify this.

Priority: 2 *Medium* *(Requirement #40)*

Relates to *Functional User Requirement 5.3.*

6.24 Add or remove course leader

A system administrator should be able to set any user in the system as course leader of any course, as well as remove the course leader from any course.

> **Rationale**: Someone must be able to do this to maintain the system, and the system administrators are the most suitable.

Priority: 3 *High* *(Requirement #41)*

Relates to *Functional User Requirement 5.4.*

6.25 Messaging system

6.25.1 Send a message

Any user should be able to send a message to any other user or project group.

> **Rationale**: A student might need to communicate with his teacher, or with the members of his project group. Therefore a messaging system is useful for the user.

Priority: 1 *Low* *(Requirement #42)*

Relates to *Functional User Requirement 1.4.*

6.25.2 Read a message

Any user should be able to read any messages they have received.

> **Rationale**: In order to make a messaging system useful, people need to be able to read their messages.

Priority: 1 *Low* *(Requirement #43)*

Relates to *Functional User Requirement 1.5.*

6.25.3 Send message to an entire course

A course leader should be able to send a message to all students in the course he is teaching.

> **Rationale**: A course leader might want to send messages to an entire course. This is not an ability the average student or teacher should have and is thus reserved for course leaders.

Priority: 1 *Low* *(Requirement #44)*

Relates to *Functional User Requirement 4.11.*

6.25.4 Automatic private messages

When a grade is set on an submission, an automatic message should be sent to the student who receives the grade.

> **Rationale**: In order to make it easier on the student, a message should automatically be sent to the student when his/her submission has been graded.

Priority: 1 *Low* *(Requirement #45)*

Relates to *Functional User Requirement 2.6.*

6.26 Non-functional System Requirements

- The user interface for SubIt! shall be implemented as XHTML and CSS without frames or java applets.
- The system shall not reveal any information about the system users other than name and username to system users that are not course leaders, system administrators or teachers.
- The system shall be able to adapt to changed hard drives or expanded hard disk space.

7 System evolution

7.1 Fundamental Assumptions

When designing this system, some things are taken for granted. We assume that

- Users are people more than 15 years old.
- Users understand the English language.
- Users know how to use a web browser and have some experience with using computers.
- Users have access to the Internet.
- Some courses at universities require students to hand in homework assignments.

7.2 Anticipated Changes

7.2.1 Changes due to Hardware Evolution

Since the system requires no specific hardware, so when there is change in hardware, it will not affect the system. Hence the system will probably not need any changes due to hardware evolution.

7.2.2 Changes due to Software Evolution

With time, new software is developed and new standards are set. This might lead to SubIt! not working properly. In that case SubIt! would need to be updated to meet the new standards.

7.2.3 Changes due to Changing User Needs

If some time in the future, universities will no longer require students to hand in assignments, SubIt! will be worthless. No changes could be made to change the system to no longer have the purpose it is supposed to have.

8 Appendices

8.1 Usecases

Global extensions

These extensions are generally applicable to any and all use cases.

*a. At any time system fails:

1. System user contacts System Administrator about system restart.
2. System user logs in to system and continues with previously recorded data.

*b. At any time System user's computer fails:

1. System user restarts computer.
2. System user logs in to system and continues with previously recorded data.

*c. At any time connection to system is lost:

1. System user solves connection problem.
2. System user continues without data loss.

8.1.1 UC1 - Log in to the system

Primary Actor: System user

Stakeholders and interests:

- System user: Wants the system to recognize his/her privileges.

Preconditions:

- User knows his/her user name and password.

Success Guarantee: User is successfully logged in with correct privileges.

Minimal Guarantee: None.

Trigger: User navigates to the SubIt! log in page.

Main Success:

1. System requires user to provide user name and password.
2. User enters user name and password.
3. System validates User.
4. System sets user privileges.
5. System displays courses user participates in.

Extensions:

3a. User name does not exist.

1. System signals user name does not exist error.
   1a. User enters another user name.
   1b. User contacts System administrator about adding a user.

3b. The password is incorrect.

1. System signals password incorrect error.
2. User enters another password.

Frequency of Occurrence: Could be nearly continuous.

8.1.2 UC2 - Log out off the system

Primary Actor: System user

Stakeholders and interests:

- System user: Wants to make sure no other person uses his/her account.

Preconditions:

- User is logged in to the system.

Success Guarantee: User is successfully logged out.

Minimal Guarantee: None.

Trigger: User chooses to log out.

Main Success:

1. System clears current session information and unsaved data is lost.
2. System displays SubIt! log in page.

Frequency of Occurrence: Could be nearly continuous.

8.1.3 UC3 - Send a private message

Primary Actor: System user

Stakeholders and interests:

- System user: Wants to send and receive private messages to and from other users.

Preconditions:

- User is logged in to the system.
- User knows the user name or project group name of the intended recipient.

Success Guarantee: User's private message is sent to the intended recipient.

Minimal Guarantee: None.

Trigger: User chooses to send private message.

Main Success:

1. System requests recipient, subject and message.
2. User inputs the user name or project group name of the intended recipient.
3. User inputs the subject of the message.
4. User writes the message.
5. User chooses to send the message.
6. System sends the message.

Extensions:

6a. The user name or project group name entered does not exist.

1. System signals recipient does not exist error.
2. User writes a new user name or project group name.

Frequency of Occurrence: Could be nearly continuous.

8.1.4 UC4 - Read private message

Primary Actor: System user

Stakeholders and interests:

- System user: Wants to read their private messages.

Preconditions:

- System user is logged in to the system.

Success Guarantee: System user reads his/her private message.

Minimal Guarantee: None.

Trigger: System user chooses to read his/her private message.

Main Success:

1. System shows private message in the message inbox of the user.
2. System user selects a private message to read.
3. System shows the message.

Extensions:

1a. No messages in the inbox.

    1. A message indicating the inbox is empty is shown.

Frequency of Occurrence: Could be nearly continuous.

8.1.5 UC5 - List project group members

Primary Actor: System user

Stakeholders and interests:

- System user: Wants to know who are in any project group.

Preconditions:

- User is logged in to the system.
- User has selected the course of interest.

Success Guarantee: The system shows a list of group members of the intended group.

Minimal Guarantee: None.

Trigger: User chooses to list project groups.

Main Success:

1. System displays a list of project groups.
2. User chooses a project group.
3. System displays a list of the members of the selected group.

Extensions:

1a. No project groups exists in the course.
    1. An empty list is shown.

3a. The project group has no members.
    1. An empty list is shown.

Frequency of Occurrence: Could be nearly continuous.

8.1.6 UC6 - Change password

Primary Actor: System user

Stakeholders and interests:

- System user: Wants to make sure no one else can use their account.

Preconditions:

- User is logged in to the system.

Success Guarantee: The user's password is changed.

Minimal Guarantee: None.

Trigger: User chooses to change the password.

Main Success:

1. System requires the user to provide old password and the new password twice.
2. User enters the passwords.
3. System saves the changes.

Extensions:

3a. Old password is wrong.

1. System signals wrong password error.

3b. The two entries of the new password do not match.

1. System signals passwords do not match error.

Frequency of Occurrence: Could be nearly continuous.

8.1.7 UC7 - Register for a course

Primary Actor: System user

Stakeholders and interests:

- Student: Wants to find courses and register themselves for it.
- Course leader: Wants any user to be able to register for their course.

Preconditions:

- User is logged in to the system.

Success Guarantee: User finds course and system registers student as course taker.

Minimal Guarantee: None.

Trigger: User chooses to search for a course.

Main Success:

1. System requests search criteria.
2. User enters search criteria for course.
3. System shows results matching search criteria.
4. User selects a course to join.
5. System shows course description.
6. User chooses to register as course taker.
7. System saves the information.

Extensions:

3a. No course matching search criteria found.
    1. System signals no match error.
        1a. User changes search criteria.

5a. User already registered to course.
    1a. User is not marked as inactive in course.
        1. Steps 6 and 7 are unavailable.

7a. User already registered to course.
    1a. User is marked as inactive in course.
        1. System changes user's status to active.

Frequency of Occurrence: Could be nearly continuous.

8.1.8 UC8 - Declare inactivity in a course

Primary Actor: Student

Stakeholders and interests:

- Student: Wants to mark himself/herself inactive in a course so that he/she will not get the information from that particular course.
- Course leader: Wants to know what students are active in the current course.

Preconditions:

- Student is registered to the specified course.
- Student is logged in to the system.
- Student has selected the course of interest.

Success Guarantee: Student has marked himself/herself as inactive in the specified course.

Minimal Guarantee: None.

Trigger: Student chooses to become inactive in a course.

Main Success:

1. System asks for confirmation on marking student as inactive.
2. Student confirms.
3. System saves submitted information.

Extensions:

3a. Student is already inactive in course.

1. System signals already inactive error.

Frequency of Occurrence: Could be nearly continuous.

8.1.9 UC9 - Make a submission

Primary Actor: Student

Stakeholders and interests:

- Student: Wants to read assignments given by the course leader in a specific course. After that the student shall be able to submit his/her solution to the corresponding assignment.

Preconditions:

- Student is logged in to the system.
- Successfully selected the course of interest.

Success Guarantee: Student is able to read an assignment information and when completed the submission, be able to submit the work to the specified assignment in the system with an attached message.

Minimal Guarantee: None.

Trigger: Student chooses to view assignments.

Main Success:

1. The system presents a list of available assignments.
2. The student selects the assignment of interest.
3. The system presents the assignment description to the user.
4. The system ask if the user wants to submit as a group or as an individual, and what files to upload.
5. The student makes a choice and uploads the files containing the work he has done.
5. The student writes a comment on the submission.
6. The system uploads the files, saves the written comment and time stamps the submission.

Extensions:

6a. The uploaded file is of the wrong type.

1. System signals file type error.
2. Student removes the file and uploads another file.

Frequency of Occurrence: Could be nearly continuous.

8.1.10 UC10 - View status of submissions

Primary Actor: Student

Stakeholders and interests:

- Student: Want to see if some submission has been graded yet.

Preconditions:

- Student is logged in to the system.
- Student has selected the course in which he/she wants to make a submission.

Success Guarantee: Student's submission shown with grades or without.

Minimal Guarantee: None.

Trigger: Student chooses to view submissions.

Main Success:

1. The system presents a list of available submissions.
2. The student selects a submission.
3. The system presents the submission and, if set, a grade and comment from the teacher.

Extensions:

1a. There are no submissions.

1. List of submissions is empty.

Frequency of Occurrence: Could be nearly continuous.

8.1.11 UC11 - Create a project group

Primary Actor: Student

Stakeholders and interests:

- Students: Want to be able to turn in submissions as a project group.
- Teachers: Want to be able to see submissions turned in from project groups.

Preconditions:

- Student is logged in to the system.
- Student has selected the course where the project group is to be created.

Success Guarantee: The student creates a new project group joins it and the system saves the information.

Minimal Guarantee: None.

Trigger: Student chooses to create/join project group.

Main Success:

1. System requests project group name.
2. Student enters a project group name.
3. System creates the new project group.

Extensions:

3a. Project group name already exists.

1. Student joins the group with the given name and a list with group members are shown.

3b. Project group name is mistyped.

1. System creates the project group with the mistyped name.
   1a. Student don't mind and keeps the mistyped name.
   1b. Student parts the group and the project group disappears (since it has no members nor submissions).

Frequency of Occurrence: Could be nearly continuous.

8.1.12 UC12 - Leave a project group

Primary Actor: Student

Stakeholders and interests:

- Student: Want to mark his/her intention of becoming inactive in the project group.

Preconditions:

- Student has successfully logged in to the system.
- Student is registered to the course in which the project group exists.
- Student has selected the course in which the project group exists.
- Student is a member of the project group he/she wants to leave.

Success Guarantee: Student has left the intended project group.

Minimal Guarantee: None.

Trigger: Student wants to leave a project group.

Main Success:

1. Student selects the project group he/she want to leave.
2. Student chooses to leave the project group.
3. System saves submitted information.

Extensions:

3a. Student is not a member of the selected project group.
    1. System signals not a member error.

3b. Project group is empty and has no submissions.
    1. System deletes project group concerned.

Frequency of Occurrence: Could be nearly continuous.

8.1.13 UC13 - View a student submission

Primary Actor: Teacher

Stakeholders and interests:

- Teachers: Want to be able to see a student submissions.
- Course leader: Wants fast and easy grading without mistakes due to lost papers.
- Students: Want fast grading without mistakes due to lost papers.

Preconditions:

- Teacher is logged in to the system.
- Teacher is authenticated as teacher in the system for the specified course.
- Teacher has selected the course of interest.

Success Guarantee: Teacher successfully downloads the submitted file.

Minimal Guarantee: None.

Trigger: Teacher chooses to view students.

Main Success:

1. System presents a list of all students.
2. Teacher selects a student.
3. System presents a list of the selected student's submissions.
4. Teacher selects a file to download.

Extensions:

1a. There are no students registered to the course.
    1. List of students is empty.

3a. There are no submissions.
    1. List of submissions is empty.

Frequency of Occurrence: Could be nearly continuous.

8.1.14 UC14 - Grade a submission

Primary Actor: Teacher

Stakeholders and interests:

- Teacher: Want to view and grade submission.
- Course Leader: Want to view and grade submission.
- Student: Want submission to be viewed and graded.

Preconditions:

- Teacher is logged in to the system.
- Teacher is authenticated as teacher in the system for the specified course.
- Teacher has selected the course of interest.
- The submission is not graded.
- Teacher has read the submission.

Success Guarantee: Teacher is able grade and comment the submission.

Minimal Guarantee: None.

Trigger: Teacher selects one of the not graded submissions.

Main Success:

1. System requests a grade and a comment.
2. Teacher sets a grade, writes a comment and submits the information to the system.
3. The system saves the information.
4. The system sends a private message to the student, saying that the submission has been graded.

Extensions:

3a. No comment is written.

1. The system saves the information without the comment.

3b. No grade is set.

1. The system signals no grade error.

Frequency of Occurrence: Could be nearly continuous.

8.1.15 UC15 - List all submissions that are not graded

Primary Actor: Teacher

Stakeholders and interests:

- Teacher: Wants to list all submissions that are not yet graded.
- Course Leader: Wants to list all submissions that are not yet graded.

Preconditions:

- Teacher is logged in to the system.
- Teacher has selected the course of interest.

Success Guarantee: The submissions are listed.

Minimal Guarantee: None.

Trigger: Teacher chooses to list submissions.

Main Success:

1. System shows a list of all handed in submissions.
2. Teacher chooses to view only the submissions that are not graded.
3. System shows a list of all ungraded submissions.

Extensions:

1a. No submissions have been made.

    1. System shows an empty list.

3a. All submissions are graded.

    1. System shows an empty list.

Frequency of Occurrence: Could be nearly continuous.

8.1.16 UC16 - Add a teacher to a course

Primary Actor: Course leader

Stakeholders and interests:

- Student: Wants to know who to contact if there are any problems.
- Teacher: Wants access to the teacher privileges.
- Course leader: Wants the teachers to make his/her job easier.

Preconditions:

- Course leader is logged in to the system.
- Course leader has selected the course where he/she wants to add a teacher.
- Course leader knows the user name of the teacher.

Success Guarantee: A new teacher is added to the course.

Minimal Guarantee: None.

Trigger: Course leader chooses to add a teacher.

Main Success:

1. System asks for the teacher's user name.
2. Course leader provides the user name.
3. System saves the information.

Extensions:

3a. The specified user does not exist.

1. System signals user name does not exist error.
2. Course leader corrects the user name.

3b. The specified user name is already a teacher of the course.

1. System signals already a teacher error.

Frequency of Occurrence: Could be nearly continuous.

8.1.17 UC17 - Remove a teacher

Primary Actor: Course leader

Stakeholders and interests:

- Student: Wants to know who to contact if there are any problems.
- Course leader: Wants the teachers to make his/her job easier.

Preconditions:

- Course leader is logged in to the system.
- Course leader has selected the course from which he/she wants to remove a teacher.
- Course leader knows the user name of the teacher.

Success Guarantee: The teacher is no longer a teacher in the course.

Minimal Guarantee: None.

Trigger: Course leader chooses to remove a teacher

Main Success:

1. System asks for the teacher's user name.
2. Course leader provides the user name.
3. System saves the information.

Extensions:

3a. The specified user does not exist.

1. System signals user name does not exist error.
2. Course leader corrects the user name.

3b. The specified user name is not a teacher of the course.

1. System signals not teacher error.

Frequency of Occurrence: Could be nearly continuous.

8.1.18 UC18 - Create an assignment

Primary Actor: Course leader

Stakeholders and interests:

- Student: Wants to know what assignments to hand in and read descriptions of them.
- Teacher: Wants to find the assignments at one place to be able to read descriptions of them.
- Course leader: Wants to specify file types etc so that reading the submissions gets easier.

Preconditions:

- Course leader is logged in to the system.
- Course leader has selected the course where he/she wants to add an assignment.

Success Guarantee: The assignment specification is created.

Minimal Guarantee: None.

Trigger: Course leader chooses to add an assignment

Main Success:

1. System asks for assignment name, description, file types accepted, start date, soft deadline and hard deadline.
2. Course leader provides the information he/she thinks is necessary.
3. System adds the assignment with provided information.

Extensions:

3a. No name is specified.

1. System signals no assignment name error.
2. Course leader specifies an assignment name.

3b. No start date is specified.

1. System automatically set the start date to today

3b. Invalid date is specified.

1. System signals invalid date error.
2. Course leader corrects the date.

Frequency of Occurrence: Could be nearly continuous.

8.1.19 UC19 - Change an assignment

Primary Actor: Course leader

Stakeholders and interests:

- Student: Wants to know what assignments to hand in and read descriptions of them.
- Teacher: Wants to know what assignments to grade and read descriptions of them.
- Course leader: Wants to change the information about the assignments and the demands so that reading the submissions gets easier.

Preconditions:

- Course leader is logged in to the system.
- Course leader has selected the assignment he/she wants to change.

Success Guarantee: The assignment is changed.

Minimal Guarantee: None.

Trigger: Course leader chooses to change the assignment.

Main Success:

1. System shows the current information.
2. Course leader changes the information he/she feels like.
3. System saves the changes.

Extensions:

3a. Course leader leaves assignment name blank.
   1. System signals no assignment name error.
   2. Course leader specifies an assignment name.

3b. Course leader leaves start date blank.
   1. System automatically set the start date to today.

3c. Invalid date is specified.
   1. System signals invalid date error.
   2. Course leader corrects the date.

Frequency of Occurrence: Could be nearly continuous.

8.1.20 UC20 - Group assignments

Primary Actor: Course leader

Stakeholders and interests:

- Student: Wants to know what assignments that are grouped together.
- Teacher: Wants to know what assignments that are grouped together.
- Course leader: Wants to group assignments that belong to the same part of the course.

Preconditions:

- Course leader is logged in to the system.
- Course leader has created an assignment.
- Course leader has selected the course where he/she wants to group assignments.

Success Guarantee: The assignments are grouped.

Minimal Guarantee: None.

Trigger: Course leader chooses to group assignments.

Main Success:

1. System asks what assignments are to be grouped.
2. Course leader selects assignments.
3. System saves the changes.

Extensions:

3a. Course leader selects no assignment.

   1. System signals no assignment name error.
   2. Course leader specifies at least one assignment to add to the group.

Frequency of Occurrence: Could be nearly continuous.

8.1.21 UC21 - Change course information

Primary Actor: Course leader

Stakeholders and interests:

- Student: Wants updated course information, to know what he/she is supposed to do.
- Teacher: Wants updated course information, to know what the students are supposed to do.
- Course Leader: Wants to have all important information stored at the same place.

Preconditions:

- Course leader is logged in to the system.
- Course leader has selected the course of interest.

Success Guarantee: The course information is updated and saved.

Minimal Guarantee: None.

Trigger: Course leader chooses to change course information.

Main Success:

1. System asks for course description.
2. Course leader writes the text.
3. Course leader chooses to save the new course description.
4. System saves the changes.

Frequency of Occurrence: Could be nearly continuous.

8.1.22 UC22 - Send message to all course members

Primary Actor: Course leader

Stakeholders and interests:

- Student: Wants up to date information.
- Teacher: Wants the students to know of any changes.
- Course leader: Wants to get important information out in an easy way.

Preconditions:

- Course leader is logged in to the system.
- Course leader has selected the course to which he/she wants to send a message.

Success Guarantee: The message is sent to the course takers.

Minimal Guarantee: None.

Trigger: Course leader chooses to message all students of the course.

Main Success:

1. System requests subject and message.
2. Course leader enters subject and message.
3. System saves sends the message to all students of the course, that are not marked as inactive.

Extensions:

3a. Course leader leaves subject or message body blank.

1. Message is sent without that information.

Frequency of Occurrence: Could be nearly continuous.

8.1.23 UC23 - Create a new user

Primary Actor: System Administrator

Stakeholders and interests:

- System Administrator: Wants the system to work.
- Teacher: Wants to be able to use the system.
- Student: Wants to be able to use the system.

Preconditions:

- System Administrator is logged in to the system.
- System Administrator knows the user information of the new user.

Success Guarantee: A new user is created with the specified user information.

Minimal Guarantee: None.

Trigger: System Administrator chooses to create a new user.

Main Success:

1. System requests user name, real name, personal identification number and password.
2. System Administrator enters a user name.
3. System Administrator enters the user's real name.
4. System Administrator enters the user's personal identification number.
5. System Administrator selects a user password.
6. System saves the information.

Extensions:

6a. The user name already exists.

1. The system signals user name exists error.
2. System Administrator chooses another user name.

6b. No user name is specified.

1. The system signals user name missing error.
2. The System Administrator specifies a user name.

6c. No real name is specified.

1. The system signals real name missing error.
2. The System Administrator specifies a real name.

6d. The user's personal identification number already exists.

1. The system signals user exists error.
2. The user is already created and another account should not be created.

Frequency of Occurrence: Could be nearly continuous.

8.1.24 UC24 - Edit a system user

Primary Actor: System Administrator

Stakeholders and interests:

- System Administrator: Wants the system to work.
- Teacher: Wants to be able to use the system.
- Student: Wants to be able to use the system.

Preconditions:

- System Administrator is logged in to the system.
- System Administrator knows the user name of the user to be edited.

Success Guarantee: The user information is updated.

Minimal Guarantee: None.

Trigger: System Administrator chooses to edit a user.

Main Success:

1. System requests user name.
2. System Administrator enters a user name.
3. System Administrator chooses to edit the user information.
4. System presents the existing user information.
5. System Administrator edits the user information.
6. System saves the information.

Extensions:

4a. No user name is specified.

1. The system signals user name missing error.
2. The System Administrator specifies a user name.

4b. User name does not exist.

1. The system signals no such user error.
2. The System Administrator specifies another user name.

6a. No user name is specified.

1. The system signals user name missing error.
2. The System Administrator specifies a user name.

6b. No real name is specified.

1. The system signals real name missing error.
2. The System Administrator specifies a real name.

6c. The user's personal identification number is missing.

1. The system signals personal identification missing error.
2. The System Administrator enters a personal identification number.

6d. The user's personal identification number already exists.

1. The system signals user exists error.
2. The System Administrator enters another personal identification number.

Frequency of Occurrence: Could be nearly continuous.

8.1.25 UC25 - Remove a system user

Primary Actor: System Administrator.

Stakeholders and interests:

- System Administrator: Wants the system to work.
- Teacher: Wants to be able to use the system.
- Student: Wants to be able to use the system.

Preconditions:

- System Administrator is logged in to the system.
- System Administrator knows the user name of the user to be removed.

Success Guarantee: The user is removed.

Minimal Guarantee: None.

Trigger: System Administrator chooses to remove a user.

Main Success:

1. System requests user name.
2. System Administrator enters a user name.
3. System removes the user.

Extensions:

3a. The user name does not exist.

1. The system signals user name does not exist error.
2. System Administrator enters another user name.

3b. No user name is specified.

1. The system signals user name missing error.
2. The System Administrator specifies a user name.

Frequency of Occurrence: Could be nearly continuous.

8.1.26 UC26 - Create a course

Primary Actor: System Administrator

Stakeholders and interests:

- System Administrator: Wants the system to work.
- Teacher: Wants to be able to use the system.
- Student: Wants to be able to use the system.

Preconditions:

- System Administrator is logged in to the system.
- System Administrator knows the name of the new course.

Success Guarantee: A new course is created.

Minimal Guarantee: None.

Trigger: System Administrator chooses to create a new course.

Main Success:

1. System asks for course name and end date.
2. System Administrator inputs the course name and an end date.
3. System saves the information.

Extensions:

3a. The course already exists.

1. The system signals course exists error.
2. The System Administrator chooses another course name.

3b. The date is invalid.

1. The system signals invalid date error.
2. The System Administrator corrects the date.

3c. No date is specified.

1. The course is added without a end date.

Frequency of Occurrence: Could be nearly continuous.

8.1.27 UC27 - Remove a course

Primary Actor: System Administrator

Stakeholders and interests:

- System Administrator: Wants the system to work.
- Teacher: Wants to be able to use the system.
- Student: Wants to be able to use the system.

Preconditions:

- System Administrator is logged in to the system.
- System Administrator knows the name of the course.

Success Guarantee: The course no longer exists.

Minimal Guarantee: None.

Trigger: System Administrator chooses to remove a course.

Main Success:

1. System requests course name.
2. System Administrator inputs the course name.
3. System saves the information.

Extensions:

3a. The course does not exist.

1. The system signals course does not exist error.
2. The System Administrator writes another course name.

Frequency of Occurrence: Could be nearly continuous.

8.1.28 UC28 - Assign a course leader to a course

Primary Actor: System Administrator

Stakeholders and interests:

- System Administrator: Wants the system to work.
- Teacher: Wants to be able to use the system.
- Student: Wants to be able to use the system.

Preconditions:

- System Administrator is logged in to the system.
- System Administrator knows the user name of the Course Leader to be added.
- System Administrator knows the name of the course.

Success Guarantee: The course leader is given Course Leader privileges for the course.

Minimal Guarantee: None.

Trigger: System Administrator chooses to set course leader.

Main Success:

1. System presents current course leader.
2. System Administrator inserts user name for the new course leader.
3. System saves the information.

Extensions:

1a. The course does not have a course leader.

1. A message indicating that there is no course leader specified is shown.

3a. The user name specified does not exist.

1. System signals no such user error.
2. System administrator enters another user name.

Frequency of Occurrence: Could be nearly continuous.

Index