SubIt!

Group 1

Joel Westberg
Mikael Granholm
Simon Stenström
Sofie Björk
Henrik Eriksson Hegardt

Functional User Requirements

Students

- Shall be able to submit assignments to a specific course for the appropriate assignment.
- Shall be able to submit several different files to the same assignment.
- Shall be able to attach a non-file message to the assignment.
- Shall be able to list and search amongst the different courses available.
- Shall be able to sign up for a course and change it to inactive if the student will not participate in that specific course.
- Shall be able to create a project group for a specific task. If all member of a project group leaves the group and there are no assignments submited, the group will be deleted.
- Shall be able to join or leave a project group.
- Shall be able to list the members of a created project.
- Shall be able to read instructions for a specific assignment submitted by the teacher or course leader.
- Shall be able to send messages to course leaders, teachers, project group and other students.
- Shall be able to read comments on specific assignments.

Teachers

- Shall be able to read an assignment submitted by a student. The teacher should also be able to add comments to the assignment which are shared with the student.
- Shall be able to grade a submitted assignment.
- Shall be able to list the students and get an overview of turned in assignments and associated grades.

Course leaders

- Shall have the same privileges as teachers.
- Shall be able to add and remove teachers from the course.
- Shall be able to add and change course description.

- Shall be able to create and organize assignments with a specified file type, start date, soft deadline and hard deadline.
- Shall be able to send messages to a whole course.

System administrator

- Shall be able to create, remove and change users.
- Shall be able to create, remove and change course.
- Shall be able to add and remove course leader from a course.

Functional System Requirements

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)*

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 usergroup further and make sure that a student cant 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)*

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)*

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)*

Joining 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 #5)*

Active course listing

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)*

Project Groups

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 labwork 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 #7)*

Leaving a project group

It should be possible for any user to at any time 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 #8)*

Deleting 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 hand-ins, there is no reason to store data about the group any longer.

Priority: 1 *Low* *(Requirement #9)*

Announcing 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 recieve information from the course, or have it listed among the courses he is active in.

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

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

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 #11)*

Submitting assignments

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 handin 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 #12)*

Student notes on hand-in

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

Rationale: Some times 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 #13)*

Submission dates

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 information can be easily found by the teacher.

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

Messaging system

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 #15)*

Send message to an entire course

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

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

Priority: 1 *Low* *(Requirement #16)*

Automatic private messages

When a grade is made on an assignment, 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 assignment has been graded.

Priority: 1 *Low* *(Requirement #17)*

Add, change or remove user

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

> **Rationale**: A system administrator is the only one with priviliges 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 (*personnummer*).

Priority: 3 *High* *(Requirement #18)*

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 #19)*

Adding, changing or removing 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 #20)*

Adding teachers to a course

The leader of a course should have the ability to add any user in the system as a teacher in that course.

> **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 #21)*

Change course description

A course leader should be able to write a course description which will be displayed to any who views that course page.

> **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, and it should be displayed whenever someone goes to that course page.

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

Creating, changing and removing assignments

A course leader should have complete controll 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 #23)*

Accepted filetypes

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 an assignments, and thus limiting students to those formats specified.

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

Setting 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 #25)*

Setting 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 #26)*

Setting 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 handed in in order to receive bonus points for the exam. Such a deadline should also be visible to the student.

Priority: 2 *Medium* *(Requirement #27)*

Ability to group assignments

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

> **Rationale**: Grouping together all the lab 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 #28)*

Viewing submitted assignments in a course

A teacher of a course should have the ability to view all details about the submissions of all students for assignments in that course.

> **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 #29)*

Student view of his/her submissions

A student should always be able to retrieve and view the files he has submitted in the system.

> **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.

Priority: 3 *High* *(Requirement #30)*

Setting a grade

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

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

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

Commenting a hand-in

A teacher should be able to comment a hand-in.

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

Priority: 1 *Low* *(Requirement #32)*

Teacher view of all hand-ins in the course

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 submission for one specific student will allow the teacher to easily see if a student has completed all required coursework.

Priority: 1 *Low* *(Requirement #33)*

Non-functional User Requirements

- A user should not be able to read or tamper with a private message sent to someone other than this user.
- A student should not be able to read or tamper with an assignment uploaded by someone else.
- A teacher should not be able to read or tamper with an assignment uploaded in a course that that this user does not teach.
- The system should be entirely in comprehensible English.
- The user interface should be simple, easy to use and estetically appealing.
- The system should be quick. It should not take a lot of time to upload or download a file, and it should not take a lot of time to navigate through the system.

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 allow people to overwrite other people's files.
- The system shall not in any case reveal sensitive information to users that are not course leaders, system administrators or teachers.
- The system shall make sure files uploaded in a specific course are available to the teachers of that course.
- The system shall be fast and responsive.

- The system shall be able to adapt to changed harddrives or expanded harddisk space.
- The requirements of the system should be low and not require state-of-the-art"hardware.
- The system shall make it easy for users to access their submitted work.

Use Cases

UC - Create a project group

Primary Actor: Student

Stakeholders and interests:

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

Preconditions:

- Student is logged in as student in the system.
- Student has chosen a course to create a project group in.

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

Main Success:

1. Student chooses to create/join project group.
2. Student enters a project group name.
3. System creates the new project group.

Extensions:

*a. At any time System fails:
   1. Student contacts System Administrator about System restart.
   2. Student logs in to system and continues with previously recorded data.

*b. At any time Student's computer fails:
   1. Student restarts computer.
   2. Student logs in to system and continues with previously recorded data.

*c. At any time connection to System is lost:
   1. Student solves connection problem.
   2. Student continues without data loss.

2a. Project group already exists.
   1. Student joins the group with the given name and a list with group members are shown.
   2. Student parts the group and enters another project group name.

2b. Project group name is mistyped.

1. System shows a list of results.

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 turned in assignments).

Frequency of Occurrence: Could be nearly continous.

UC - Find a course and register as course taker

Primary Actor: Student

Stakeholders and interests:

- Students: Want to find courses and register themselfs as course takers.
- Course leader: Wants to be able to see all students taking their coure.

Preconditions:

- Student is logged in as student in the system.

Success Guarantee: Student finds course and System registers student as course taker.

Main Success:

1. Student enters search criteria for course and the system shows results matching search criteria.
2. Student chooses to register as course taker and the system saves the information.

Extensions:

*a. At any time System fails:

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

*b. At any time Student's computer fails:

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

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

1. Student solves connection problem.
2. Student continues without data loss.

1a. No course matching search criteria found.

1. System signals no match error.
   1a. Student changes search criteria.
   1b. Student waits till System administrator added the course.

1b. Several courses matches search criteria.

1. System shows a list of results.
   1a. Student chooses one of the courses from the list.
   1b. Student starts a new search.

2a. Student already registered to course as student/teacher/course leader.

1. System signals student already registered error.

Frequency of Occurrence: Could be nearly continous.

UC - Grade a submitted assignment

Primary Actor: Teacher

Stakeholders and interests:

- Teacher - Want to view and grade submitted assignment.
- Course Leader - Want to view and grade submitted assignment.
- Student - Want submitted assignment to be viewed and graded.

Preconditions:

- Teacher is logged in.
- Teacher is authenticated as teacher in the system for specified course.
- Teacher has found the course in the system.
- There are submitted assignments not graded.

Success Guarantee: Teacher is able to view and grade submitted assignment.

Main Success:

1. System presents a list of all submitted assignments in the course.
2. Teacher selects one of the not graded submitted assignment.
3. Teacher downloads submitted document and reads it.

4. Teacher marks grade in the system, writes a comment and submits the information to the system.
5. Teacher repeats step 2-4 until done.

Extensions:

*a. At any time System fails:
   1. Teacher contacts System Administrator about System restart.
   2. Teacher logs in to system and continues with previously recorded data.

*b. At any time Teacher's computer fails:
   1. Teacher restarts computer.
   2. Teacher logs in to system and continues with previously recorded data.

*c. At any time connection to System is lost:
   1. Teacher solves connection problem.
   2. Teacher continues without data loss.

3a. Submitted document is not valid.
   1. Teacher sends message to student concerned.
   2. Teacher writes a comment to submitted assignemnt.
   3. Teacher waits for student to resubmit assignment.

Frequency of Occurrence: Could be nearly continous.

UC - View corrected assignments

Primary Actor: Teacher

Stakeholders and interests:

- Teachers: Want to be able to see a list of corrected homework assignments.
- Course leader: Wants fast and easy grading without misstakes due to lost papers.
- Students: Want fast grading without misstakes due to lost papers.

Preconditions:

- Teacher is logged in.
- Teacher if authenticated as teacher in the system for a specified course.
- Teacher has found the course in the system.

- Teacher has earlier graded students of the course.

Success Guarantee: Teacher has been presented a list of students of the course period and a list of handed in assignments for a student.

Main Success:

1. Teacher sees a list of all students.
2. Teacher selects a student and sees a list of all handed in assignments.
3. Teacher repeats step 1-2 until done.

Extensions:

*a. At any time System fails:
   1. Teacher contacts System Administrator about System restart.
   2. Teacher logs in to system and continues with previously recorded data.

*b. At any time Teacher's computer fails:
   1. Teacher restarts computer.
   2. Teacher logs in to system and continues with previously recorded data.

*c. At any time connection to System is lost:
   1. Teacher solves connection problem.
   2. Teacher continues without data loss.

2a. No assignments are handed in.
   1. List of assignments is empty.

Frequency of Occurrence: Could be nearly continous.

UC - See if a submitted assignment has been graded

Primary Actor: Student

Stakeholders and interests:

- Students: Wants to see if some assignments already handed in has been graded yet.

Preconditions:

- Student is logged in to the system.
- Student has navigated himself to the homework assigments of the course.

Success Guarantee: Students assignments shown with grades or without.

Main Success:

1. Student chooses homework assignment.
2. Student sees the assignment and the grade.
3. Student repeats step 1-2 until satisfied.

Extensions:

*a. At any time System fails:
    1. Student contacts System Administrator about System restart.
    2. Student logs in to system and continues with previously recorded data.

*b. At any time Student's computer fails:
    1. Student restarts computer.
    2. Student logs in to system and continues with previously recorded data.

*c. At any time connection to System is lost:
    1. Student solves connection problem.
    2. Student continues without data loss.

2a. No assignments are handed in.
    1. List of assignments is empty.

Frequency of Occurrence: Could be nearly continous.

UC - Private messaging

Primary Actor: Student

Stakeholders and interests:

- Student: Wants to send a Private Message (PM) to Course Leader.
- Course Leader: Wants to receive the message.

Preconditions:

- Student is logged in to the system.
- Student knows the username of the Course Leader.

Success Guarantee: Students PM is recieved by the Course Leader.

Main Success:

1. Student chooses to send Private Message.
2. Student writes the adress of the Course Leader in the adress field.
3. Student writes the subject of the message in the message field.
4. Student writes the message.
5. Student sends the message.

Extensions:

*a. At any time System fails:
   1. Student contacts System Administrator about System restart.
   2. Student logs in to system and continues with previously recorded data.

*b. At any time Student's computer fails:
   1. Student restarts computer.
   2. Student logs in to system and continues with previously recorded data.

*c. At any time connection to System is lost:
   1. Student solves connection problem.
   2. Student continues without data loss.

5a. The user of the adress defined does not exist.
   1. System signals incorrect adress error.
   2. Student writes a new adress.

Frequency of Occurrence: Could be nearly continous.

UC - Administrating

Primary Actor: System Administrator

Stakeholders and interests:

- System Administator: Wants to create a course and set a Course Leader to that course.
- Course Leader: wants to have an account and a course to teach.

Preconditions:

- System Administator is logged in to the system.
- System Administator knows the contact information of the Course Leader to be.
- System Administator knows the name of the new course.

Success Guarantee: A new course is created and the new teacher has an account and is the Course Leader for that course.

Main Success:

1. System Administator chooses to create new user account.
2. System Administator creates the new user.
3. System Administator creates a new course.
4. System Administator sets the newly created user as Course Leader in the newly created course.

Extensions:

*a. At any time System fails:

1. System Administrator restarts System.
2. System Administator logs in and continues with previously recorded data.

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

1. System Administator restarts computer.
2. System Administrator logs in and continues with previously recorded data.

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

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

2a. The user already exists.

1. The system signals user exists error.
2. System Administator no longer have to create the user.

3a. The course already exists.

1. The system signals course exists error.

1a. System Administator creates a new course with an other course code.

1b. System Administator no longer have to create a new user.

4a. The course already has a Course Leader.

1. The system signals course leader exists error.
2. System Administator no longer have to set a Course Leader.

Frequency of Occurrence: Could be nearly continous.

UC - Create a project group

Primary Actor: Student

Stakeholders and interests:

- Students: Leave course so they will not get the information feed released from that particular course. Student shall also be able to drop out of an assigned project group.
- Course leader: Wants to know what students are active in the current course.

Preconditions:

- Student is registered to the specific class.
- Student has joined a project group the correspondent class.
- Student has successfully logged onto the system.
- Student has been able to select the course of interest.

Success Guarantee: Student has marked himself/herslef as inactive in the affected course and left the corresponding project group.

Main Success:

1. Student selects to leave group and submits information to the System.
2. System stores submitted information.
3. Student marks himself/herself as inactive in course and submits information to the System.
4. System stores submitted information.

Extensions:

*a. At any time System fails:

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

*b. At any time Student's computer fails:

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

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

1. Student solves connection problem.
2. Student continues without data loss.

1a. Group is empty and has no submitted assignments.
    1. System deletes group concerned.

3a. Student is allready inactive in course.
    1. System signals allready inactive error.

Frequency of Occurrence: Could be nearly continous.

UC - Assignment information retrieving and submission

Primary Actor: Student

Stakeholders and interests:

- Student wants to read assignments given by the course leader in a specific course. After the student is done with his assignment he shall be able to submit it to the corresponding assignment.

Preconditions:

- Student has successfully logged onto the system.
- Successfully selected the course of interest.

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

Main Success:

1. The student selects the assignment of interest and reads the instructions attached to it.
2. The student uploads the file containing the work he has done.
3. The student attaches a message to the file.
4. The student submits the assignment.

Extensions:

*a. At any time System fails:
    1. Student contacts System Administrator about System restart.
    2. Student logs in to system and continues with previously recorded data.

*b. At any time Student's computer fails:
    1. Student restarts computer.
    2. Student logs in to system and continues with previously recorded data.

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

1. Student solves connection problem.
2. Student continues without data loss.

2a. 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 continous.