SubIt!

Group 1

Joel Westberg
Mikael Granholm
Simon Stenström
Sofie Björk
Henrik Eriksson Hegardt

Innehåll

1 Introduction

1.1 Purpose

The purpose of SubIt! Design Document is to describe the design and the architecture of SubIt!. The design is expressed in sufficient detail so as to enable developers to understand the underlying architecture of SubIt!.

1.2 Scope

This document is intended to act as a sufficient foundation for the implementation of SubIt!. The system is basically a webserver application with connections to a fileserver and a database. The setup of the system(s) handling the fileserver and the database is outside the scope of this document.

1.3 Intended Audience

This Design Document is intended to act as a technical reference tool for developers involved in the development of SubIt!.

1.4 Related Documents

Throughout this Design Document are references to the SubIt! Requirements Document. The reader are assumed to be familiar with this document or at least have access to it.

1.5 Glossary

1.5.1 Technical Terms

PHP
A computer programming language originally designed for producing dynamic web pages. [1]
Server
An application, or a device that performs services for connected clients as part of a client-server architecture. [2]
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. [3]

[1] http://en.wikipedia.org/wiki/PHP
[2] http://en.wikipedia.org/wiki/Server_(computing)
[3] http://en.wikipedia.org/wiki/Database

1.5.2 SubIt! Defined Words

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.
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.
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.
Teachers
Teachers teach something in a course. The can be assistants or co-lecturers or laboratory assistant.
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.
Assignment
The question/problem/subject that the students are to answer/solve/write about.
Submission
The file/files that a single student hands in.

1.6 Abstract

SubIt! is a system for handling student's submission to course specific assignments. This document defines what is to be implemented.

Chapter 2 gives an overview of the system in different detail levels, with focus on data and control flow between the system components.

Chapter 3 discusses assumptions on which design decisions are made.

Chapter 4 shows screenshots of the user interface functions and describes the general navigation within the system.

Chapter 5 describes the design in detail, including definitions of classes, functions and variables. Finally the database design is presented.

Chapter 6 provides a set of test cases for the final testing of the implemented system.

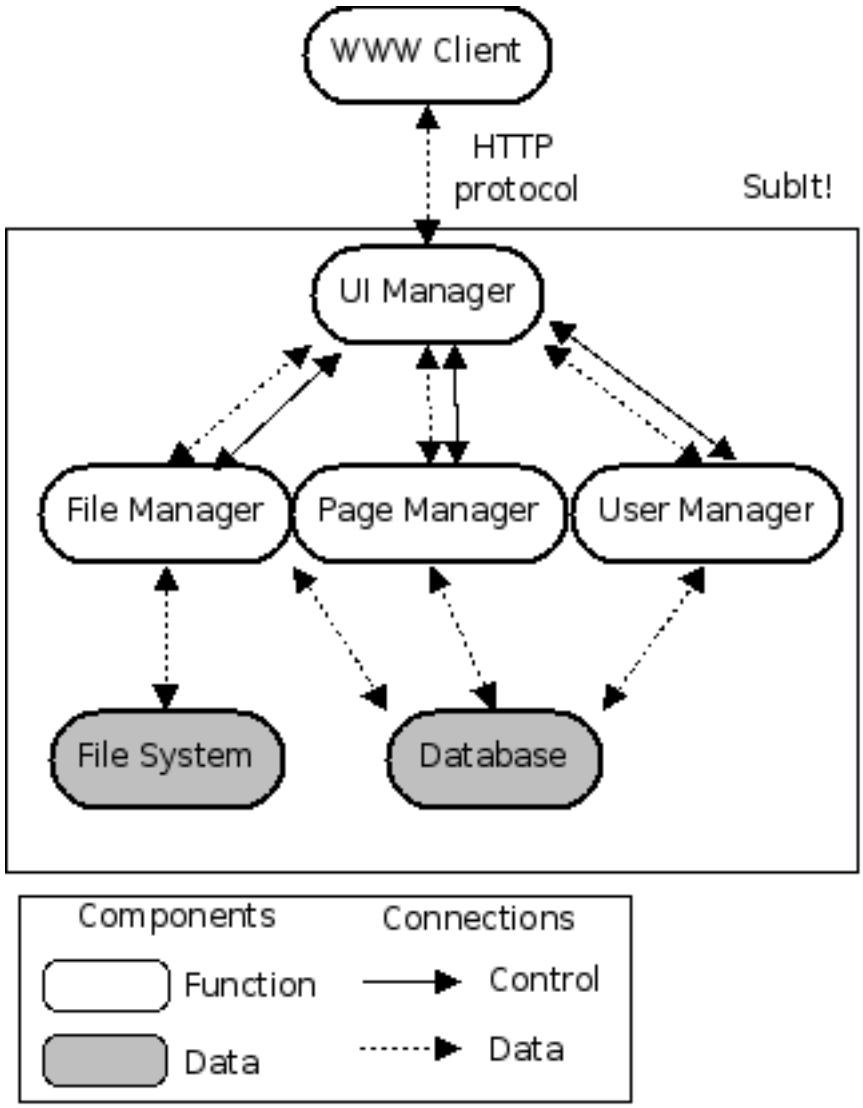
2 System Overview

2.1 General Description

The system will enable teachers and students to easily communicate, allowing creation of assignments in a course for a teacher, and allowing submissions to be made to said assignment for the student. This is the primary function of the system, but will also allow other things such as communcation through a messaging system, as well as grading of submissions.
The system is entirely web-based, and is designed to be viewed and fully functional in any modern web browser. The design is oriented toward usability, and navigating as well as using the system should not pose any difficulty to use for it's users.

2.2 Overall Architecture Description

The system is divided into four main parts that control the control and data flow as shown below. Since both files and other data (user privileges, page information, comments, etc.) are stored within the system, a file system and a database will be needed.



In scope

2.2.1 UI Manager

The UI manager is used to handle the interface and is the contact between the client and the logic of the system.

2.2.2 File Manager

The File Manager is used to access files and keeps track of file locations.

2.2.3 Page Manager

The Page Manager handles the information about the web pages. It responds to all incoming HTTP GET requests from clients.

2.2.4 User Manager

The User Manager keeps track of all the user data and their priveleges within the system.

Out of scope

2.2.5 Web Clients

The clients are using an internet browser to access the system. However, this layer is only used to display the information provided by the system, and is thus outside the scope of this document.

2.2.6 File System

The File System stores all the files. Since the File Manager keeps track of the file locations, any file system can be used, as long as it is functional.

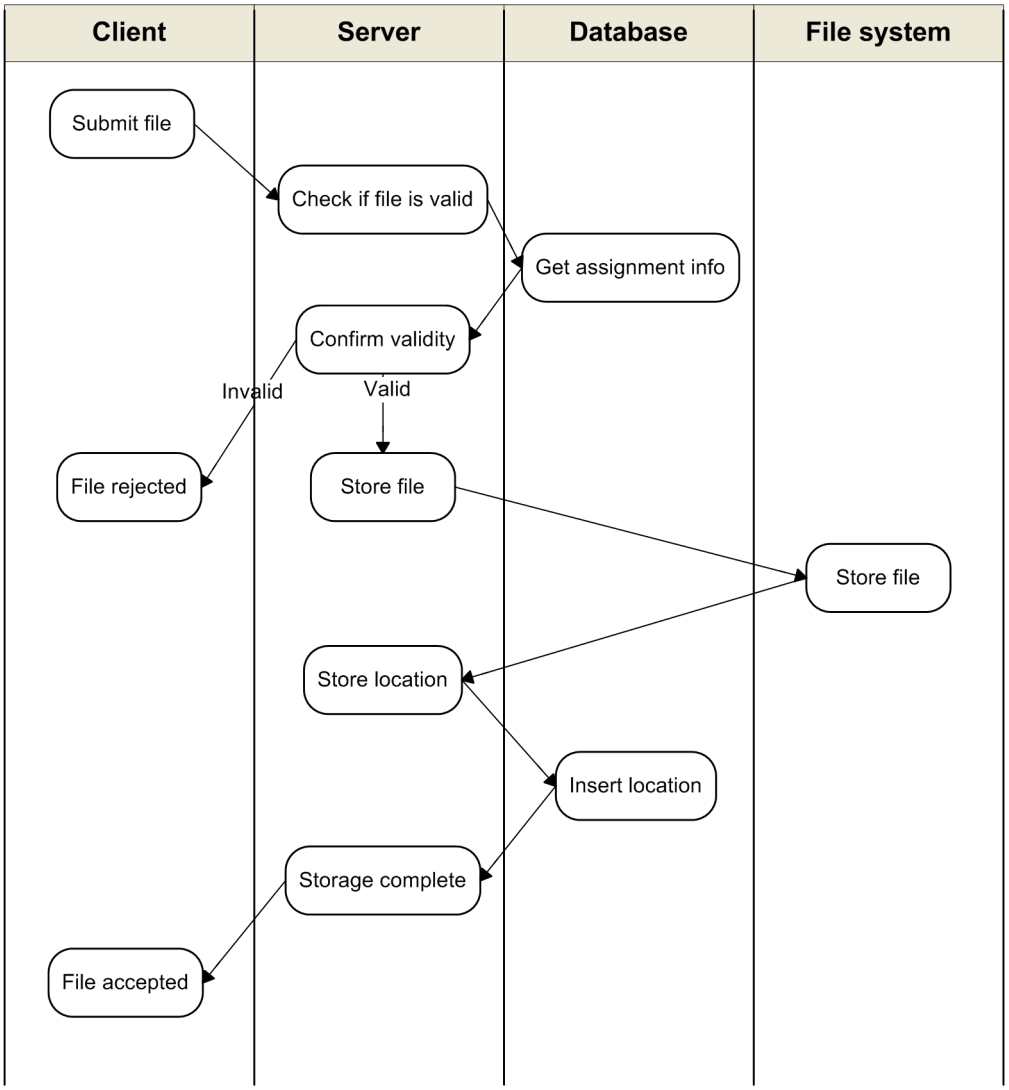
2.2.7 Database

The Database serves as the backbone of the system. It will store all information needed by any other part of the system. Any database engine can be used, the choice of database engine is thus out of the design scope.

2.3 Detailed Architecture

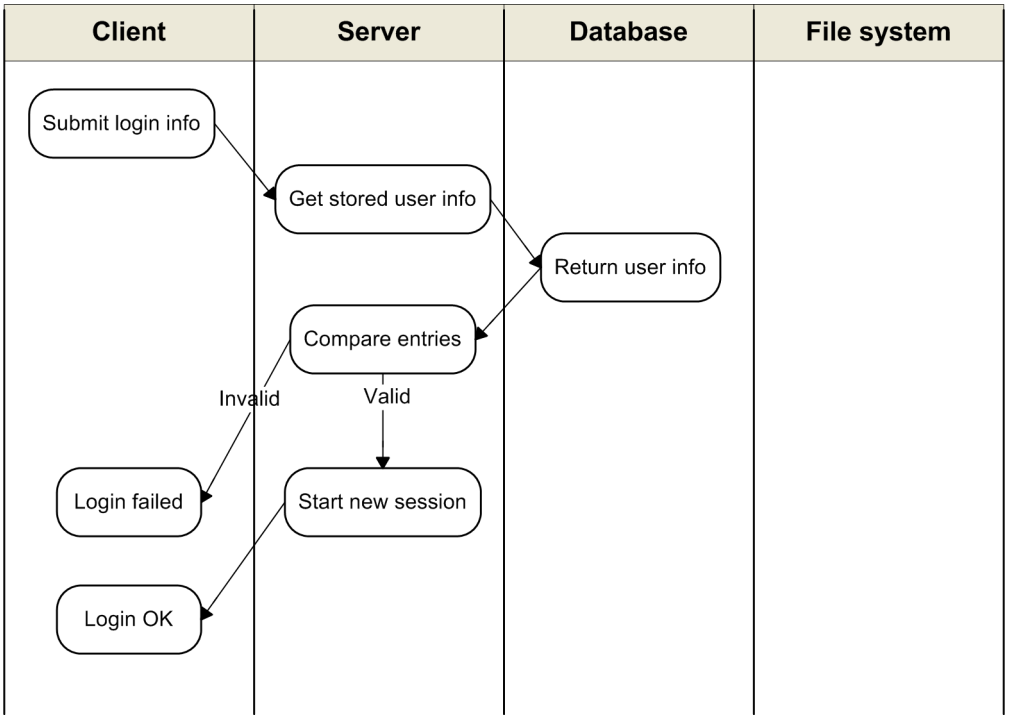
The diagrams below show the data flow between the different parts of the system.

2.3.1 File Submission



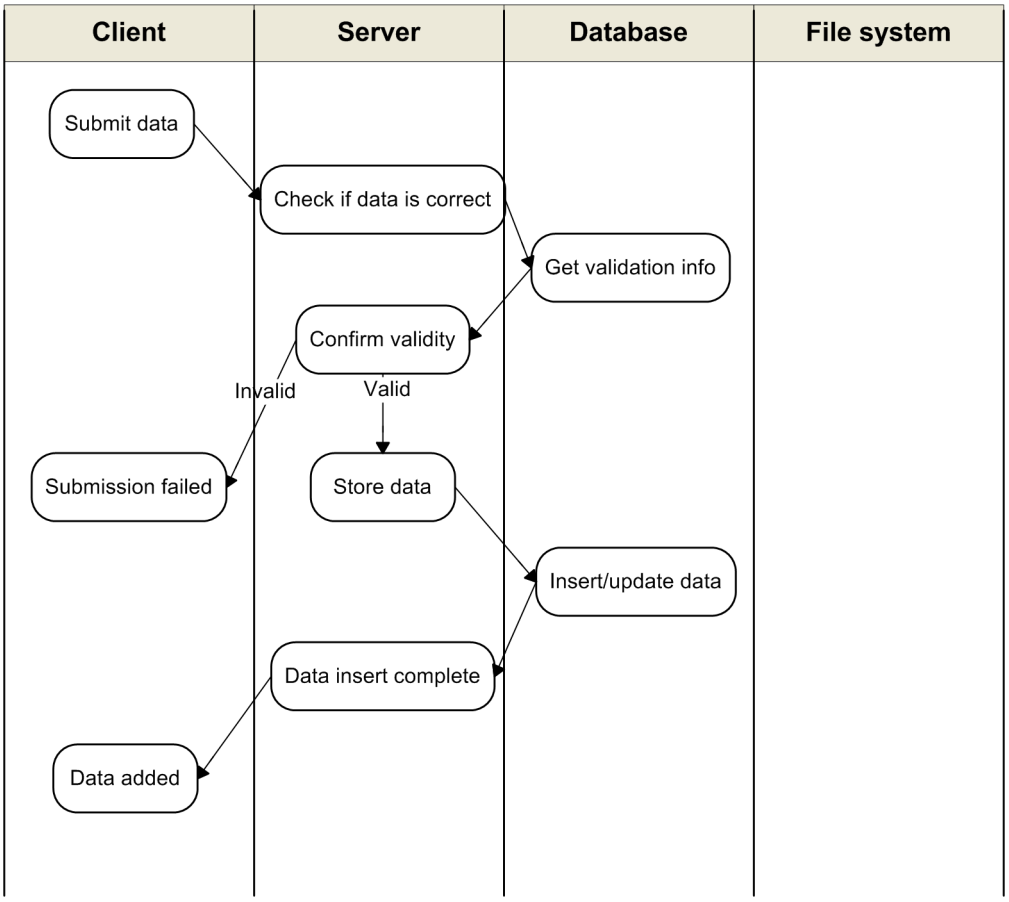
Upon submission, either an error message will be returned to the clientto the client if the file submitted was invalid, or an confirmation that the file was recieved. A correctly submitted file will be saved in the file system and it's location stored in the database.

2.3.2 Login Attempt



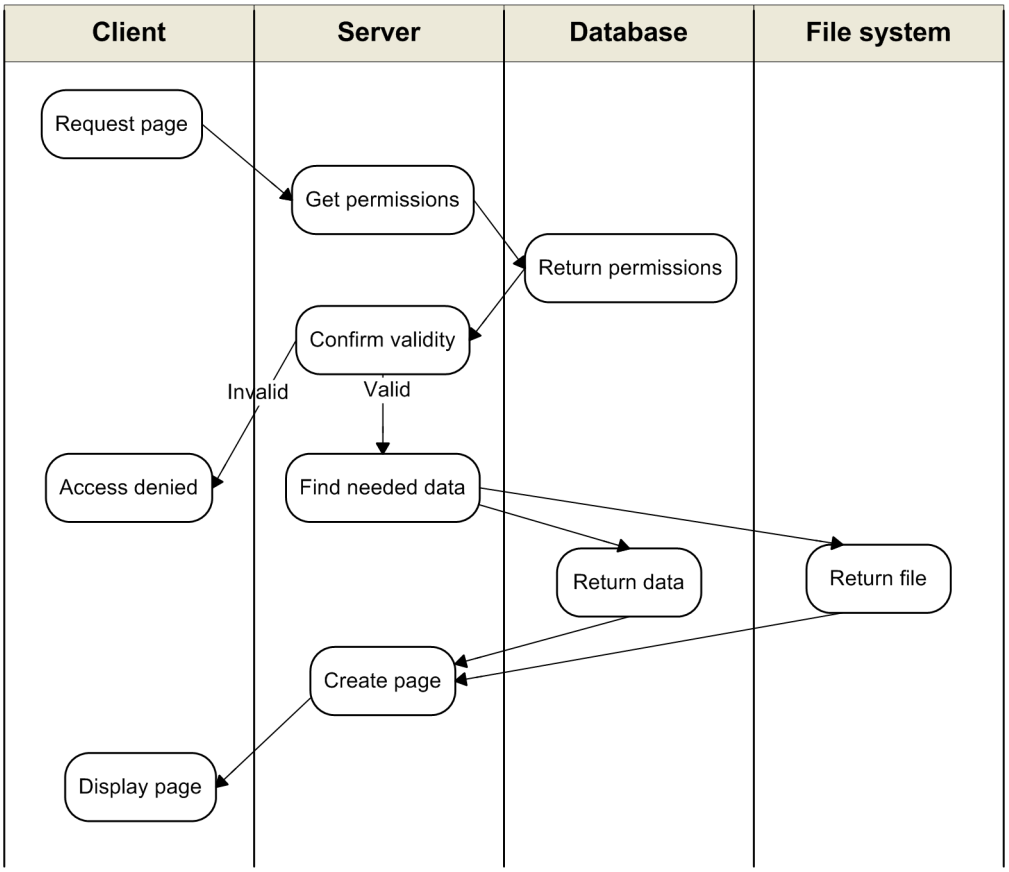
This diagram shows the flow of a regular login attempt through the system.

2.3.3 Insert data (Users, courses, assignments, etc



This diagram shows an attempt to insert data into the database such as adding, or changing an already existing user, course, assignment, etc.

2.3.4 Regular Usage



Shows the regular usage of the system. A page is requested by client, and a response is generated and sent to the client in return.

3 Design Considerations

3.1 Assumptions and Dependencies

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

- are people more than 15 years old.
- understand the English language.
- know how to use a web browser and have some experience with using computers.
- have access to the Internet.
- use an updated version of a standard web browser (such as Mozilla Firefox or Internet Explorer).

We assume that the web server

- has an updated version of PHP.
- has an updated version of PostgreSQL.

3.2 General Constraints

The system will not work if

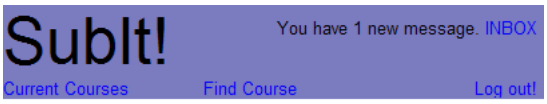
- the server is not running.
- the network is down.
- the server hard drive is full.

4 Graphical User Interface

4.1 Overview

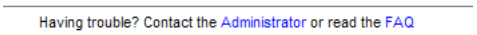
The graphical user interface is presented to the user in a web browser. It is a simple interface, with a header and footer that allows the user to navigate to the most often used pages. When the user navigates him/herself somewhere, there is a path of links underneath the header that shows the user links to other pages higher up in the hierarchy.

4.2 Header



The header is displayed in all views of the system except for the log in page. Bellow the header is the navigation path, which consists of links to pages higher up in the hierarchy.
In the header the "Current Courses" link leads to a display of the current courses. "Find courses" leads to the Find Course view. "Log out" logs the user out of the system and "Inbox" leads to the private message inbox.

4.3 Footer



The footer is displayed in all views of the system.

4.4 Login screen

References to Requirement Document:
User Functional Requirement 1.8.
When logging in the current courses view is displayed.

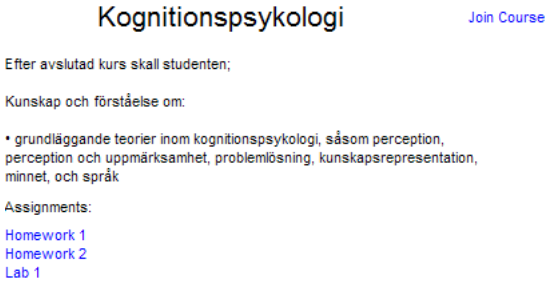
4.5 List current courses

References to Requirement Document:
User Functional Requirement 1.9.
The names of the courses are links to the course descriptions of those courses.

4.6 Find course

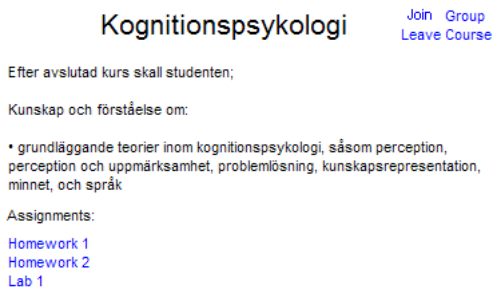
References to Requirement Document:
User Functional Requirement 1.1.
When the search button is pressed, a list of courses matching the search is displayed.
The names of the courses are links and lead to the course description.

4.7 Student view of course (not registered to course)



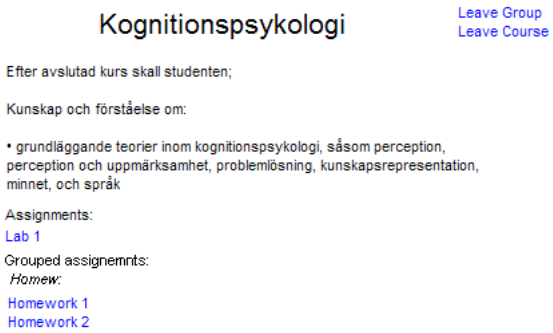
References to Requirement Document:
User Functional Requirement 1.2.
"Join Course" allows a student to join a course.
The names of the assignments are links that lead to the assignment description of that particular assignment.

4.8 Student view of course (not in Project Group)



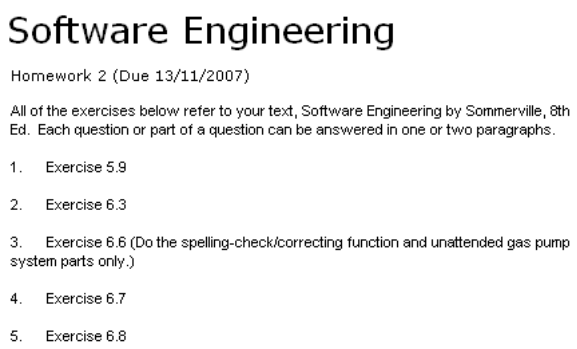
References to Requirement Document:
User Functional Requirement 2.9.
"Join Group" allows a student to join a project group.
"Leave Course" sets a student as inactive in the course.
The names of the assignments are links that lead to the assignment description of that particular assignment.

4.9 Student view of course (in project group)



References to Requirement Document:
User Functional Requirements 1.3, 2.1 and 2.10.
"Leave Group" allows a student to leave his/her project group.
"Leave Course" sets a student as inactive in the course.
The names of the assignments are links that lead to the assignment description of that particular assignment.

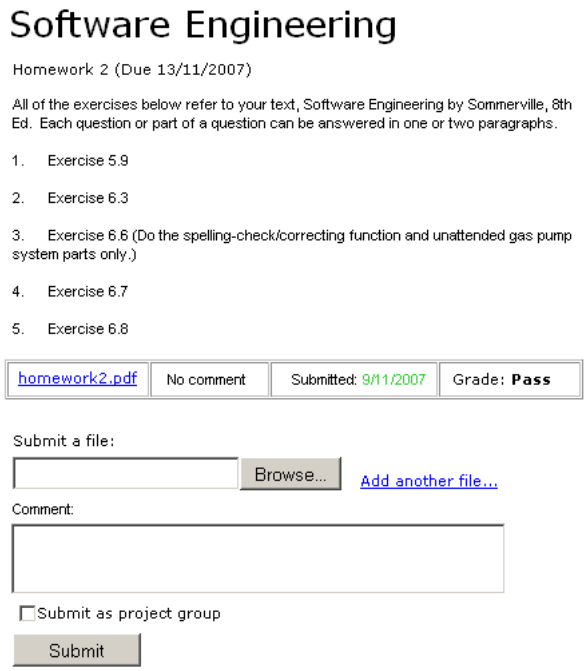
4.10 Student view of assignemnt (not registered to course)



4.11 Join project group

References to Requirement Document:
User Functional Requirement 2.9
The button adds the user to the project group specified in the text field from the system.

4.12 Student submit files



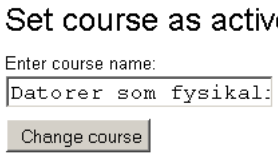
References to Requirement Document:
User Functional Requirement 2.3, 2.4, 2.5, 2.8 and 2.11.
The filename is a link to a previously handed in file. Other handed in files are also listed here.
The date color indicates if the assignment was handed in on time.
"Browse" allows the user to browse the hardrive to find a file to upload.
"Add another file" makes another file field pop up above the comment field.
The "Submit as project group" checkbox should be checked if the user wants to hand in the submission as a project group.
"Submit" uploads the files and saves the comment.

4.13 Student view Project Group

References to Requirement Document:
User Functional Requirement 1.7.
"Leave group" removes the user from this group.

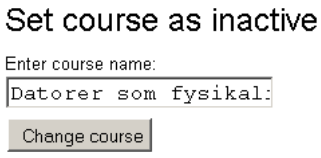
The assignment name is a link to the assignment page.
The button removes the user specified in the text field from the system.

4.14 Set course as active



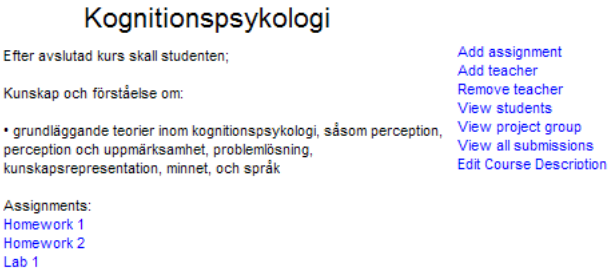
References to Requirement Document:
User Functional Requirement 5.3
The button sets the course specified in the text field as active for the user.

4.15 Set course as inactive



References to Requirement Document:
User Functional Requirement 5.3
The button sets the course specified in the text field as inactive for the user.

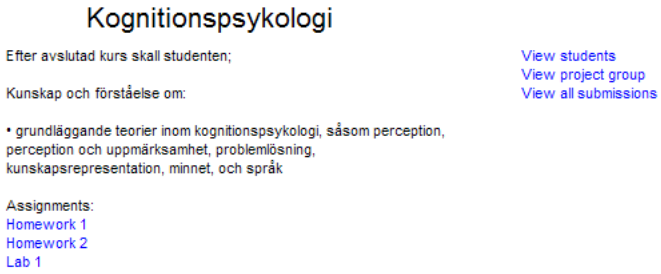
4.16 Course Leader view of course



"Add assignment" sends the course leader to the add assignment view.
"Add teacher" sends the course leader to the add teacher view.
"Remove teacher" sends the course leader to the remove teacher view.
"View students" shows the course leader a list of all students in the course.
"View project groups" shows the project groups in the course.
"View all submissions" shows the submissions in the course.
"Edit course description" sends the course leader to the add course decription view.

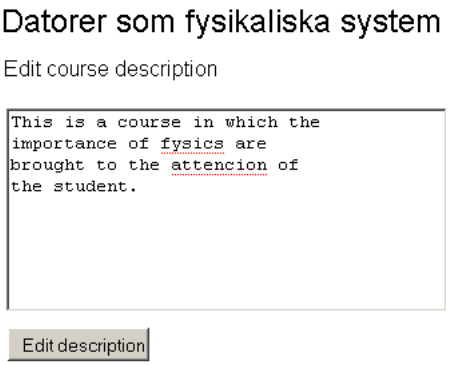
The names of the assignments are links that lead to the assignment description of that particular assignment.

4.17 Teacher view of a course



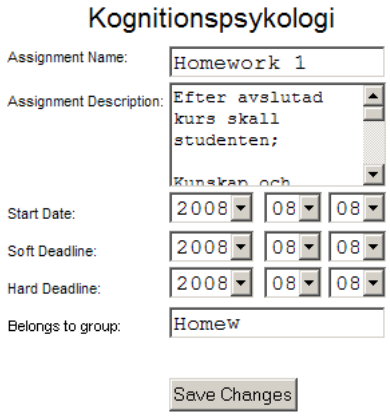
"View students" shows the course leader a list of all students in the course.
"View project groups" shows the project groups in the course.
"View all submissions" shows the submissions in the course.
The names of the assignments are links that lead to the assignment description of that particular assignment.

4.18 Course Leader edit course description



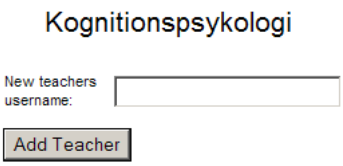
References to Requirement Document:
System Functional Requirement 4.2.
"Description" field shows and allows editing of current course description.
The "Edit description"-button saves the description to the database.

4.19 Course leader add/edit an assignment



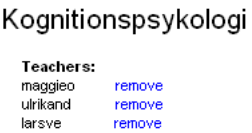
References to Requirement Document:
User Functional Requirement 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9.
The fields are prefilled if any information was previously entered.
"Save Changes" saves the changes.

4.20 Course leader add teacher to a course



References to Requirement Document:
User Functional Requirement 4.1.
"Add teacher" adds the user with the provided username as teacher to the course.

4.21 Course leader list teacher from a course



References to Requirement Document:
User Functional Requirement 4.1.
The "remove" link removes the teacher on that row from being teacher in the course.

4.22 Teacher/Course leader list students of a course

References to Requirement Document:
User Functional Requirement 3.1.
The names of the students are links to the view "submissions of a student" page.

4.23 Teacher/Course leader list project groups

References to Requirement Document:
User Functional Requirement 1.7.
The names of the groups are links to the view that project group page.

4.24 Teacher/Course leader list all submissions of a course

References to Requirement Document:
User Functional Requirement 3.6.
The assignment names are links to the "grade submission" page.
The writer names are links to the "List all submissions of a (student/project group)" page.
If the submission is handed in as a project group, there is a cross in the "As PG" column.
If the submission is already graded, there is a grade in the "grade" column.

4.25 Teacher/Course leader list submissions of a student

Italienska fortsättningskurs

Submissions of Eva Tall:

Submission	As PG	Grade
Hemtenta1		C
Labb1	X	B
Labb2		F
Labb3		E

References to Requirement Document:
User Functional Requirement 3.3.
The assignment names are links to the "grade submission" page.
If the submission is handed in as a project group, there is a cross in the "As PG" column.
If the submission is already graded, there is a grade in the "grade" column.

4.26 Teacher/Course leader list submissions of a project group

Italienska fortsättningskurs

Members of BosseMVKninjas:

Bo Ek

Cecilia Björk

Eva Bok

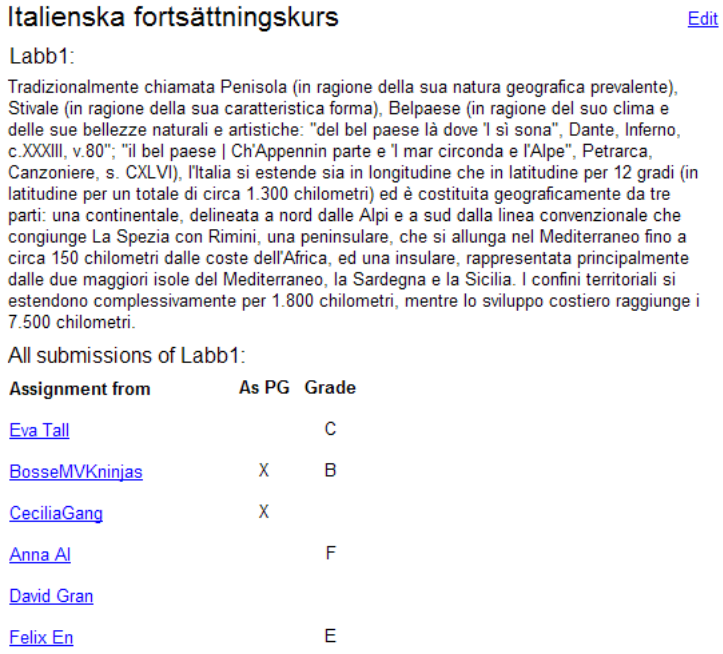
Felix Al

Submissions of BosseMVKninjas:

Submission	Grade
Labb1	
Labb2	F
Labb3	E

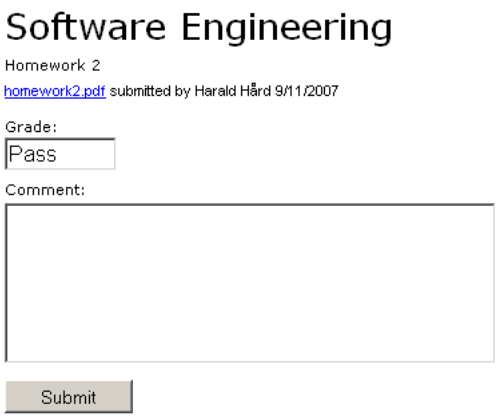
References to Requirement Document:
User Functional Requirement 1.7 and 3.3.
The students names are links to that student's submissions page.
The assignment names are links to the "grade assignment" page.
If the submission is already graded, there is a grade in the "grade" column.

4.27 Course leader view of assignment



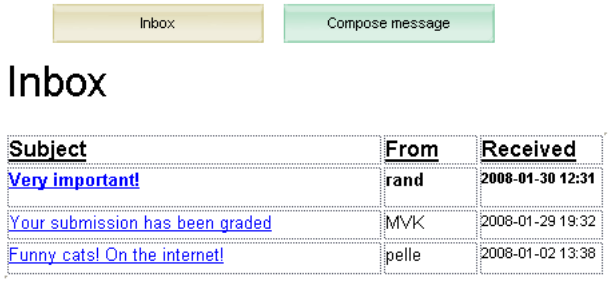
References to Requirement Document:
User Functional Requirement 1.3 and 3.6.
"Edit" is a link to the "Edit course description" page.
The writer names are links to the "List all submissions of a (student/project group)" page.
If the submission is handed in as a project group, there is a cross in the "As PG" column.
If the submission is already graded, there is a grade in the "grade" column.

4.28 Teacher/Course leader grade a submission



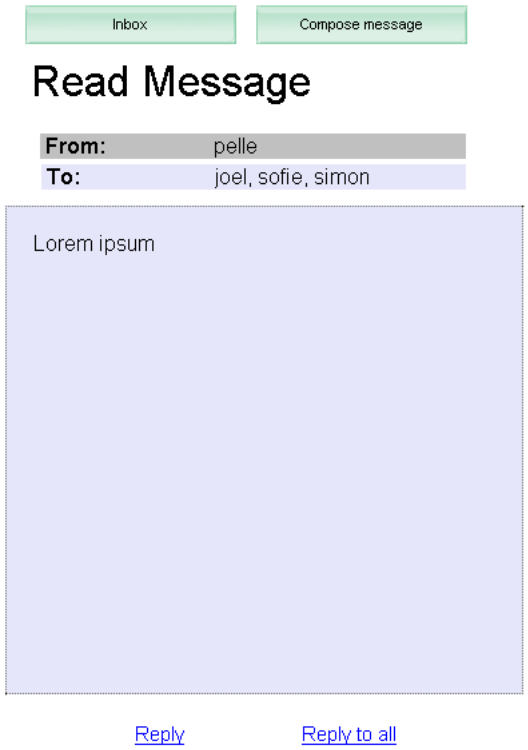
References to Requirement Document:
User Functional Requirement 3.2, 3.4 and 3.5.
The filename is a link to the file.
The Grade text field is unchecked and any sign can be inserted.
The "Submit" button saves the grade and comment.

4.29 View inbox



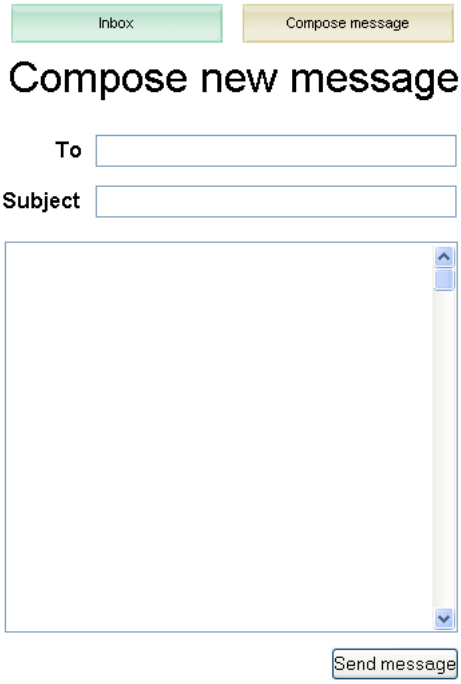
References to Requirement Document:
User Functional Requirement 2.6 and 4.1.
"Inbox" updates the inbox view.
"Compose message" shows the compose message view.
The subjects of the messages are links to display that message in the read message view.
The "from" column indicates from what user name the message was sent.
The "recieved" column indicates when the message was sent.
Bold text indicates that the message is unread.

4.30 Read message



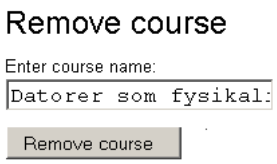
References to Requirement Document:
User Functional Requirement 1.5.
"Inbox" shows the inbox view.
"Compose message" shows the compose message view.
The Reply and Reply all buttons displays the Compose Message view, with to and subject fields already filled.

4.31 Compose new message



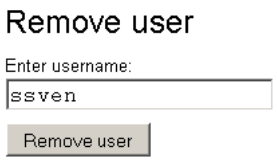
References to Requirement Document:
User Functional Requirement 1.4.
"Inbox" shows the inbox view.
"Compose message" shows the compose message view.
The Send message-button sends the message to the user name(s) specified in the "To:" text field.

4.32 Administrator view of Remove course



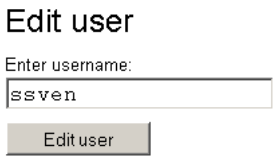
References to Requirement Document:
User Functional Requirement 5.2
The button removes the course specified in the text field.

4.33 Administrator view of Remove user



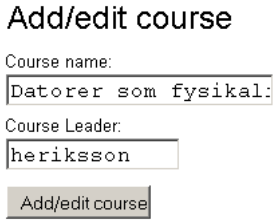
References to Requirement Document:
User Functional Requirement 5.1
The button removes the user specified in the text field from the system.

4.34 Edit user



References to Requirement Document:
User Functional Requirement 5.1
The button sends the user to the Add/Edit user view.

4.35 System Administrator add/edit course



References to Requirement Document:
System Functional Requirement 5.2.
The "Course name:" field contains the name of the course.
The "Course Leader" field contains the username of the course leader of the course.
The "Add/edit course" button will save the data to the database.

4.36 System Administrator add/edit user

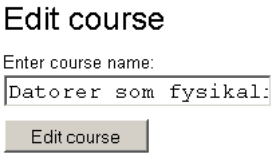
References to Requirement Document:
System Functional Requirement 5.1.
The "Name:" field contains the name of the user.
The "Personal identification number:" field contains the pin of the user.
The "E-mail:" field contains the email of the user.
The "Username" field contains the username of the user.
The "Password:" field contains the password of the user.
The "Add/edit course" button will save the data to the database.

4.37 System Administrator view of administer system

References to Requirement Document:
"Add user" sends the system administrator to the add user view.
"Remove user" sends the system administrator to the remove user view.
"Edit user" sends the system administrator to the edit user view.
"Add course" sends the system administrator to the add course view.
"Remove course" sends the system administrator to the remove course view.
"Edit course" sends the system administrator to the edit course view.
"Set course as active" sends the system administrator to the set course as active view.

“Set course as inactive” sends the system administrator to the set course as inactive view.

4.38 System administrator edit course



References to Requirement Document:
System Functional Requirement 5.2.
“Enter course name:” field contains the course name to be edited.
The “Edit course”-button sends the System Administrator to the edit course view for the course specified.

5 Design Details

5.1 Class Responsibility Collaborator Cards

Page	
Generates and outputs website Coordinates classes Handles all user input (logins, submissions, etc)	User Course ProjectGroup Assignment Submission Message

User	
Knows real name Knows user name Knows personal identification number Allows change of all above data Can confirm username/password combination Can retrieve active courses Can retrieve project group(s) participating in	Course ProjectGroup

ProjectGroup	
Knows course Knows name Knows course Alows change of above data Can retreive list of users Can add user to group Can remove user from group	None

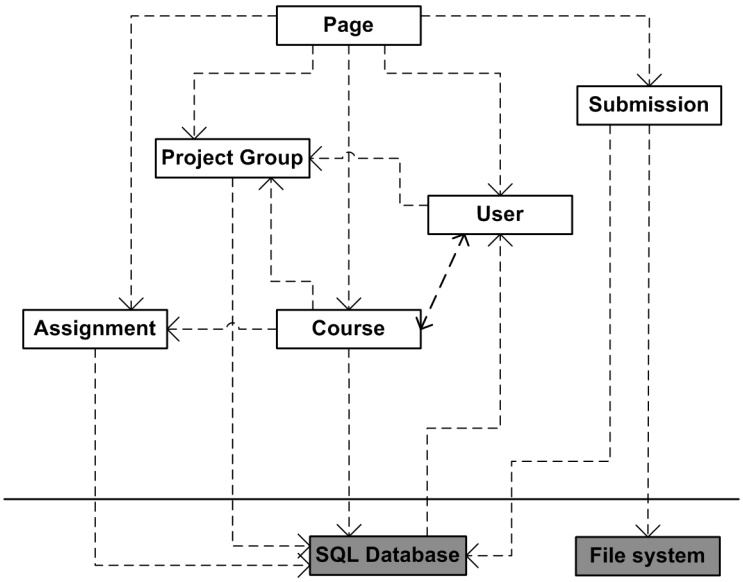
Course	
Knows description Knows name Knows course leader Allows change of above data Can retrieve project groups Can retrieve teachers Can retrieve participating users Allows addition of new teacher Allows removal of teacher Allows addition of a new participant Allows removal of a participant	User ProjectGroup Assignment

Assignment	
Knows course Knows soft deadline Knows hard deadline Knows start date Knows description Allows change of all above data Can retrieve submissions for assignment	None

Submission	
Knows assignment Knows creator Knows project group Knows location of files Knows student comment Knows grade Knows teacher comment Allows change of above data Allows addition of files Can retrieve all files for submission Allows setting and retreival of submission timestamps	None

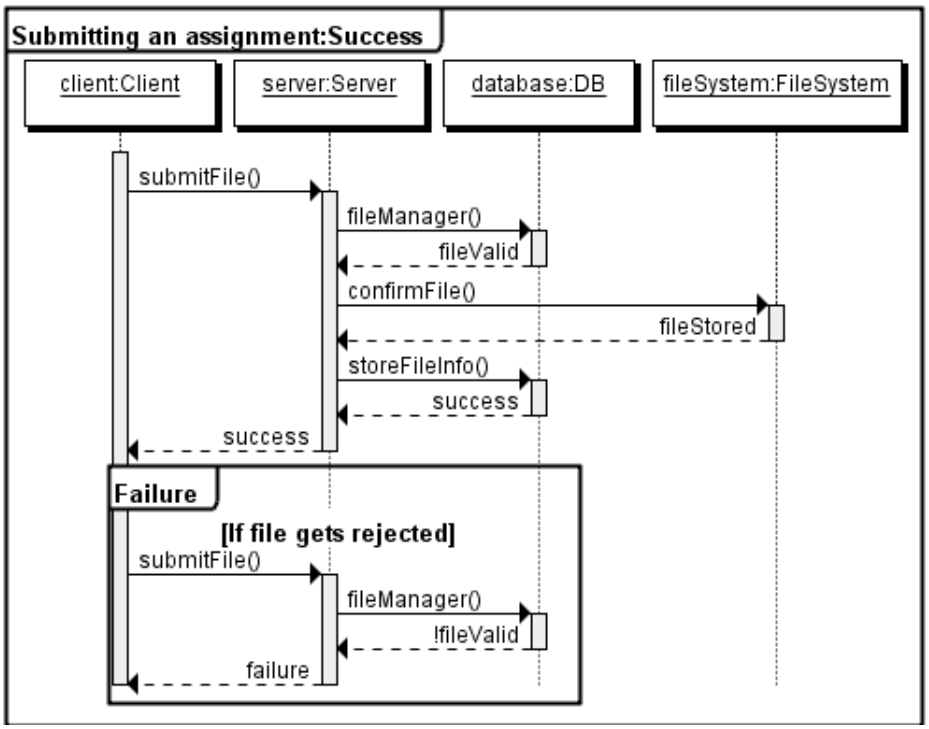
Message	
Knows sender Knows recipients Knows title Knows message body Knows read Allows change of all above data	None

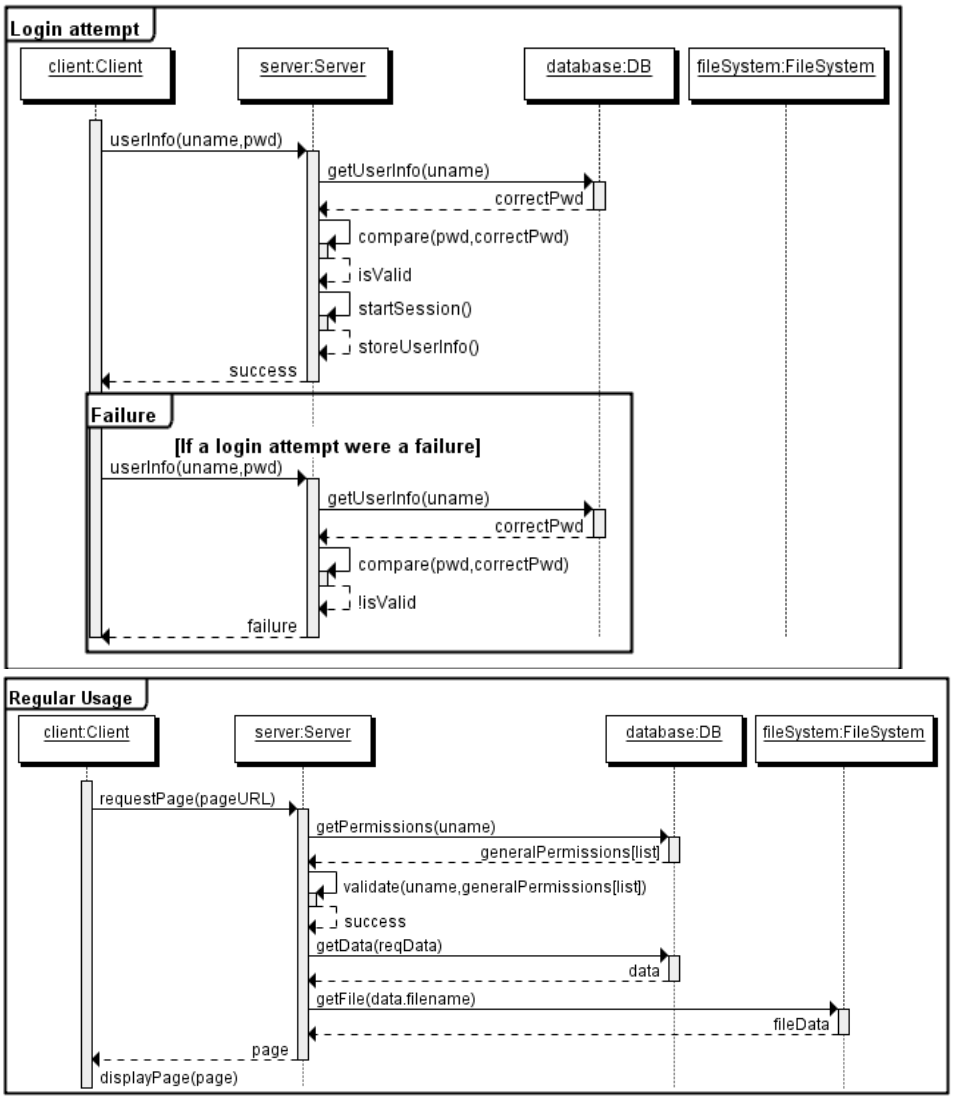
5.2 Class Diagram

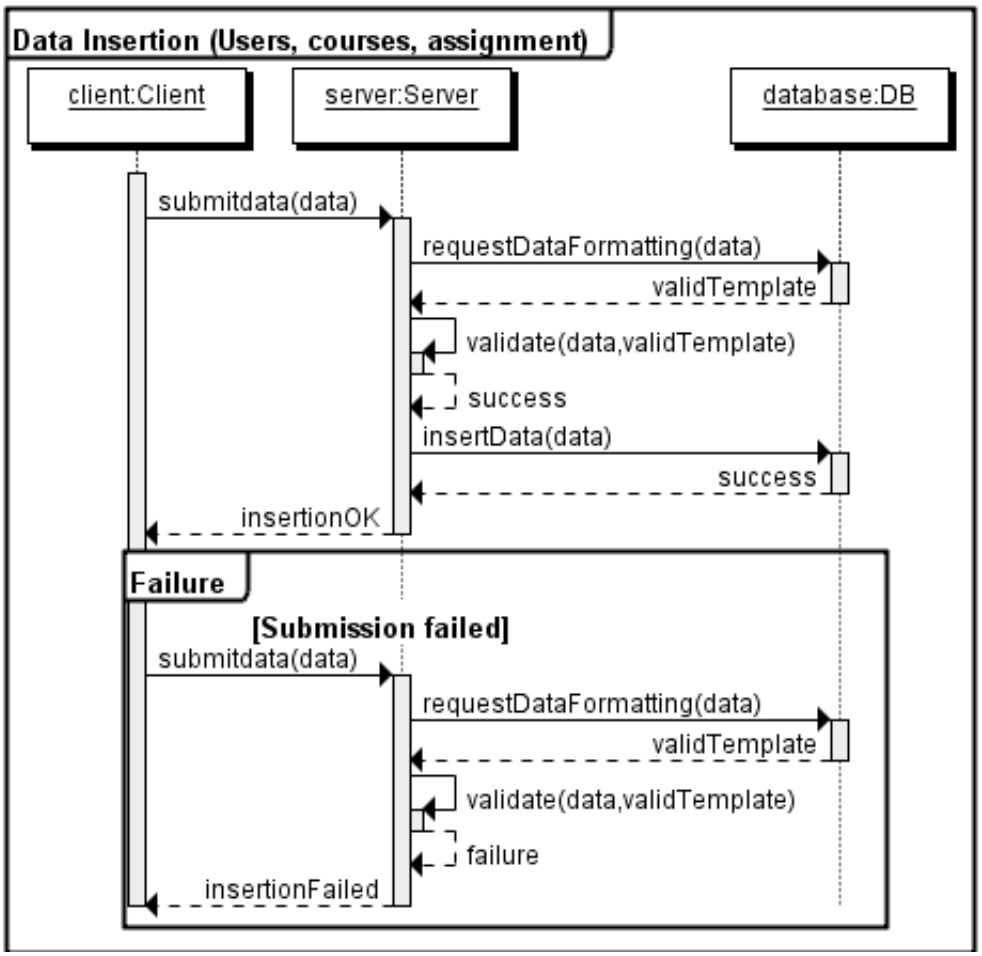


5.3 State Charts

5.4 Interaction Diagrams




Login attempt
client:Client
server:Server
database:DB
fileSystem:FileSystem
userInfo(uname,pwd)
getUserInfo(uname)
correctPwd
compare(pwd,correctPwd)
isValid
startSession()
storeUserInfo()
success
Failure
[If a login attempt were a failure]
userInfo(uname,pwd)
getUserInfo(uname)
correctPwd
compare(pwd,correctPwd)
!isValid
failure
Regular Usage
client:Client
server:Server
database:DB
fileSystem:FileSystem
requestPage(pageURL)
getPermissions(uname)
generalPermissions[list]
validate(uname,generalPermissions[list])
success
getData(reqData)
data
getFile(data.filename)
fileData
page
displayPage(page)



5.5 Detailed Design

5.5.1 class Page

Functions
General Note:
All functions below take into account the priveleges of the logged in user.

- *printAssignmentPage(int id)*
  Generates and shows the assignment page for the specified assignment. If no id is given, the page for creating a new assignment is shown.

- *printSubmissionPage(int id)*
  Generates and shows the submission page for the specified submission. If no id is given, the page for making a new submission is shown.

- *printCoursePage(int id)*
  Generates and shows the course page for the specified course. If no id

is given, the page for creating a new course is shown.

- *printPGPage(int id)*
  Generates and shows the project group page for the specified project group.
- *printStudentList(int course_id)* Generates and shows a list of all students in the specified course.
- *printAssignmentPage(int id)*
  Generates and shows the assignment page for the specified assignment.
- *printInbox()*
  Generates and shows the inbox.
- *printMessage(int id)*
  Generates and shows the page for the specified private message.
- *printCompose(int[] to)*
  Generates and shows the page for composing a new private message. Any element in the argument array will be already be filled in as receivers of the message.
- *printLoginPage()*
  Generates and shows the page with the login form.
- *printPGList(int course_id)*
  Generates and shows a list of all Project Groups for the specified course.
- *printSubmissionList(int type, int id)*
  Generates and shows a list of all submissions. The argument type specifies what type of list to generate, be it for a student, or for an entire course. The id argument allows specification of student or course.
- *printCreateUserPage()*
  Generates and shows the page for adding a user to the system.

Variables
This class has no class variables.

5.5.2 class Assignment

Functions

- *getId()*
  Returns id.

- *getCourse()*
  Returns course.
- *getSoftDeadline()*
  Returns softDeadline.
- *getHardDeadline()* Returns hardDeadline.
- *getStartDate()*
  Returns startDate.
- *getDescription()*
  Returns description.
- *getFiletypes()*
  Returns filetypes.
- *setCourse(int id)*
  Sets course.
- *setSoftDeadline(long deadline)*
  Sets softDeadline.
- *setHardDeadline(long deadline)*
  Sets hardDeadline.
- *setStartDate(long deadline)*
  Sets startDate.
- *setDescription(string desc)*
  Sets description.
- *setFiletypes(string[] types)*
  Sets filetypes.
- *getSubmissions()*
  Returns an array of id's for all submissions for this assignment.
- *save()*
  Saves all data to the database.

Variables

- int id
- int course
- long softDeadline

- long hardDeadline
- long startDate
- string description
- string[] filetypes

5.5.3 class User

Functions

- *getRealName()*
  Returns the real name of a user as a String.
- *setRealName(String realName)*
  Sets the real name of a user.
- *getUserName()*
  Returns the user name of a user as a String.
- *setUserName(String userName)*
  Sets the user name of a user.
- *getPIDN()*
  Returns the personal identification number of a user as a String.
- *setPIDN(String pIDN)*
  Sets the personal identification number of a user.
- *isPassword(String pw)*
  Generates and shows the assignment page for the specified assignment.
- *setPassword(String pw)*
  Sets the password of a user.
- *getCourses()*
  Returns an array with the CourseIDs of the courses the user is participating in.
- *setProjectGroup()*
  Returns the ProjectGroupID of the project group a user has joined.
- *save()*
  Saves the changes to the database.

Variables

- int id
- String userName, realName, pIDN

5.5.4 class Course

Functions

- *getCourseId()*
  Returns the course id.
- *getCourseName()*
  Returns the course name.
- *setCourseName(string s)*
  Set the course name to *s*.
- *getDescription()*
  Returns the course description.
- *setDescription(string s)*
  Set the course description to *s*.
- *getCourseLeader()*
  Returns the user id of the course leader.
- *setCourseLeader(int i)*
  Set the course leader to user with id *i*.
- *getTeachers()*
  Returns an array with user ids of the teachers assigned to the course.
- *addTeacher(int i)*
  Add user with id *i* to the list of teachers in the course.
- *removeTeacher(int i)*
  Remove the user with id *i* from the list of teachers in the course.
- *getAssignments()*
  Returns an array with the assignment ids of the course assignments.
- *addParticipant(int i)*
  Add user with user id *i* to the list of participants in the course.
- *removeParticipant(int i)*
  Remove user with user id *i* from the list of participants in the course.
- *save()*
  Save to database.

Variables

- *int id*
  Course id.
- *int array teachers*
  Array with user id of teachers assigned to course.
- *string description*
  Course description.
- *string name*
  The name of the course.
- *int courseLeader*
  The user id the course leader.
- *int array assignments*
  Array with assignment id of the course assignments.

5.5.5 class ProjectGroup

- *getId()*
  Returns the id as an int.
- *getCourse()*
  Returns the CourseID of as an int.
- *setCourse(int CourseID)*
  Sets the course of a the project group.
- *getName()*
  Returns the name of the project group as a String.
- *setName(String name)*
  Sets the name of the project group.
- *getUsers()*
  Returns the userIDs of the users in the project group.
- *addUser(int userID)*
  Adds a user to the project group.
- *removeUser(int userID)*
  Removes a user from the project group.

Variables

- int id, course
- String name

5.5.6 class Message

Functions

- *getId()*
  Returns the id.
- *getFrom()*
  Returns the senders id.
- *setFrom(int id)*
  Sets sender id.
- *getTo()*
  Returns the recipients.
- *addTo(int id)*
  Adds id to list of recipients.
- *setTo(mixed ids)*
  Sets id's in array to be the recipients. Accepts arrays of id's or comma formated string as argument.
- *getTitle()*
  Returns the title of the message.
- *setTitle(string name)*
  Sets the title.
- *getHideTo*
  Returns hideTo.
- *setHideTo(boolean b)* Sets hideTo.
- *getBody()*
  Returns the body.
- *setBody(string message)*
  Sets message body.
- *save()*
  Saves the message. Equivalent to sending it.
- *getRead()*
  Return read status.
- *setRead(bool read)*
  Sets mesage read status.

Variables

- int[] to
- int from, id
- string body, title
- bool read, hideTo

5.5.7 class Submission

Functions

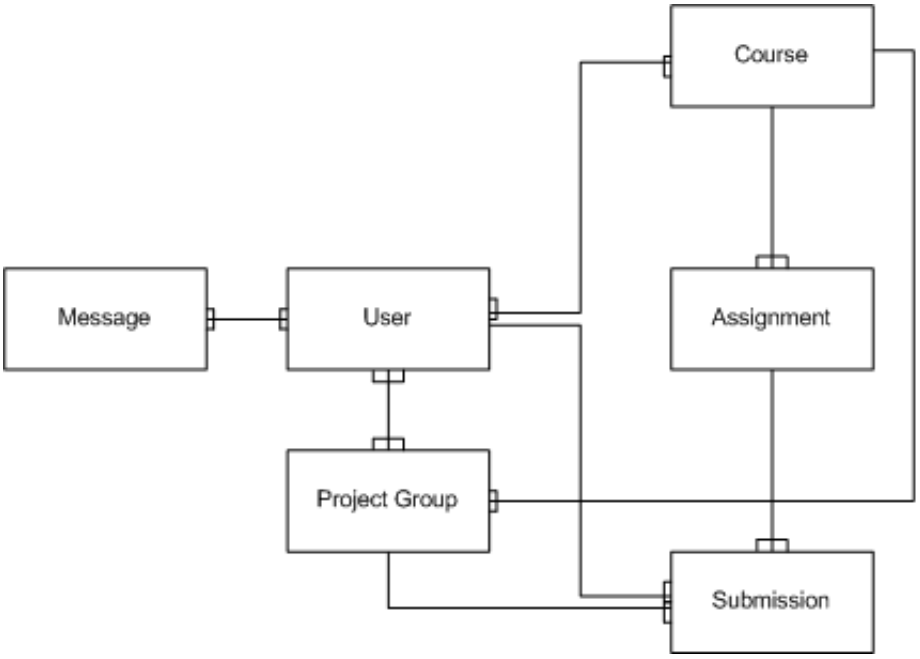
- *getId()*
  Returns the id as an int.
- *getAssignment()*
  Returns the assignment of as an int.
- *setAssignment(int assignmentID)*
  Sets the assignment.
- *getCreator()*
  Returns the creator of the submission as an int.
- *setCreator(int userID)*
  Sets the creator of the submission.
- *getProjectGroup()*
  Returns the project group as an int.
- *setProjectGroup(int projectGroup)*
  Sets the project group.
- *getFiles()*
  Returns the files of the submission as a String[].
- *addFile(String file)*
  Adds a file to the files of the submission.
- *getSComment()*
  Returns the sComment as a String.
- *setSComment(String sComment)*
  Sets the sComment.
- *getTComment()*
  Returns the tComment as a String.

- *setTComment(String tComment)*
  Sets the tComment.
- *getGrade()*
  Returns the grade as a String.
- *setGrade(String grade)*
  Sets the grade.
- *getTimestamps()*
  Returns the timestamps as a long[].
- *setTimestamps(long[] timestamps)*
  Sets the timestamps.
- *save()*
  Saves the data to the database.

Variables

- int id, assignment, creator, projectgroup
- String sComment, tComment, grade
- String[] files
- long[] timestamps

5.5.8 Database Design



user

column name	type	references
id	int	
username	varchar(255)	
password	varchar(40)	
pin	varchar(20)	
email	varchar(255)	
name	varchar(255)	
admin	int	

course

column name	type	references
id	int	
name	varchar(255)	
desc	text	
course_leader	int	user.id

course_participants

column name	type	references
user	int	user.id
course	int	course.id
teacher	boolean	

assignment

column name	type	references
id	int	
course	int	course.id
description	text	
hard_deadline	timestamp	
soft_deadline	timestamp	
starttime	timstamp	

assignment_filetypes

column name	type	references
assignment	int	assignment.id
filetype	varchar(10)	

submission

column name	type	references
id	int	
assignment	int	assignment.id
user	int	user.id
project_group	int	project_group.id
grade	varchar(10)	
s_comment	text	
t_comment	text	

file

column name	type	references
id	int	
submission	int	submission.id
filename	varchar(255)	

project_group

column name	type	references
id	int	
name	varchar(255)	
course	int	course.id

project_group_members

column name	type	references
project_group	int	project_group.id
user	int	user.id

message

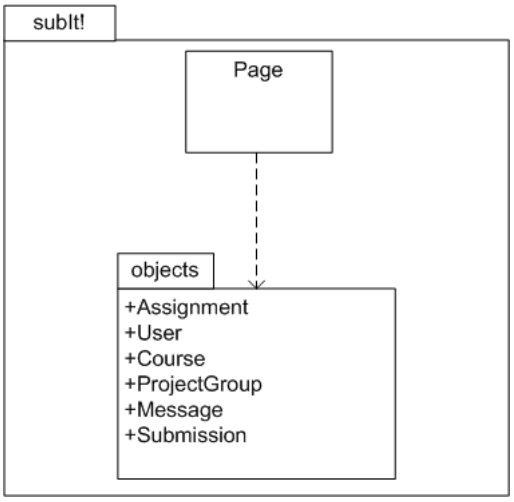
column name	type	references
id	int	
sender	int	user.id
body	text	
title	text	
hideTo	boolean	

message_to

column name	type	references
message	int	message.id
to	int	user.id
read	boolean	

5.6 Package Diagram

SubIt! consists of two packages. The package *subIt!* represents the system as a whole. In the package *subIt!* are the class Page which generates the pages that is sent to the client webbrowser and the subpackage *objects* on which the class Page depends. The subpackage *objects* contains classes that represents the different functions of the SubIt! system.



6 Functional Test Cases

In all test cases except the Log in to the System, it is assumed that the user has been validated to the system as the sort of user that can execute the test.

6.1 Log in to the system

Functionality

Existing user identifies and authenticates himself/herself to the system.

Requirement

This test corresponds to Requirement #5 from the Requirements Document.

Inputs

Valid username and password.

Output and Observable Effects

User specific active courses page is displayed.

Procedure

1. Navigate to the SubIt! log in page.
2. Enter a valid username and password.
3. Press the "Log in" button.
4. Verify that the user specific active courses page is displayed.

6.2 List active courses

Functionality

Display a list of courses available.

Requirement

This test corresponds to Requirement #6 from the Requirements Document.

Inputs

None.

Output and Observable Effects

List of available courses is displayed.

Procedure

1. Select "Find course" in the system header.
2. Make sure the search textfield is empty.
3. Press the "Search" button.
4. Verify that a list of available courses is displayed.

6.3 Join a course

Functionality

Mark a valid user as active in an existing course.

Requirement

This test corresponds to Requirement #7 from the Requirements Document.

Inputs

The course name.

Output and Observable Effects

The course to be joined is shown in the list of active courses.

Procedure

1. Select "Find course" in the system header.
2. Enter the name of an existing course.
3. Press the "Search" button.
4. Select the course in the list of search results.
5. Select "Join course" in the course page.
6. Select "Current courses" in the system header.
7. Verify that the course just joined is listed as an active course.

6.4 List courses user is active in

Functionality

List courses that the user is active in.

Requirement

This test corresponds to Requirement #9 from the Requirements Document.

Inputs

None.

Output and Observable Effects

A list of user specific active courses is displayed.

Procedure

1. Select "Current courses" in the system header.
2. Verify that a list of user specific active courses is displayed.

6.5 Submit an assignment

Functionality

Submit a file to an open assignment (an assignment where the hard deadline is not passed).

Requirement

This test corresponds to Requirement #13 from the Requirements Document.

Inputs

Local path to the file to be submitted.

Output and Observable Effects

File is displayed in the list of submitted files.

Procedure

1. Select "Current courses" in the system header.
2. Select the course associated with the open assignment.
3. Select the open assignment.
4. Enter the local path to the file to be submitted in the text field.
5. Press the "Submit" button.
6. Verify that the file is listed at the assignment page.

6.6 Create a project group

Functionality

Create a project group

Requirement

This test corresponds to Requirement #17 from the Requirements Document.

Inputs

The student types the name of the group and will join that group.

Output and Observable Effects

The group that you have joined and the group members of that particular group will be displayed.

Procedure

1. Select "Current courses" in the system header.
2. Select the course of interest.
3. Student enters the name of the group in the groupname textfield.
4. Press the "Submit" button.
5. Verify that project group exists.

6.7 Submit assignment as project group

Functionality

Lets a group to submit an assignment.

Requirement

This test corresponds to Requirement #18 from the Requirements Document.

Inputs

One of more files containing the group assignments.

Output and Observable Effects

Verify that the correct files got submitted.

Procedure

1. Select "Current courses" in the system header.
2. Select the course of interest.
3. Press the "Browse" button.
4. To add more files to the project press "Add another file...".
5. Check the "Submit as project group" checkbox.
6. Press "Submit" button.
7. Verify that the correct files got submitted.

6.8 Leave a project group

Functionality

Let students leave groups.

Requirement

This test corresponds to Requirement #20 from the Requirements Document.

Inputs

None.

Output and Observable Effects

If the student is not in a group they will see a "Join group" label in the far right upper corner.

Procedure

1. Select "Current courses" in the system header.
2. Select the course of interest.
3. If the student is in a group in the current course the student will be able to press "Leave Project Group".
4. Verify that the student is not in a project group.

6.9 View all students in a course

Functionality

Enables teachers and course leaders to view what students that are participating in the course.

Requirement

This test corresponds to Requirement #22 from the Requirements Document.

Inputs

None.

Output and Observable Effects

The student gets a list over the current students that are active in the correspoding course.

Procedure

1. Select "Current courses" in the system header.
2. Select the course of interest.
3. Press the "View students" link.
4. Verify that a list of current students is displayed.

6.10 View a submission in a course

Functionality

Enables the teachers and course leaders to view a specific submission in the corresponding course.

Requirement

This test corresponds to Requirement #23 from the Requirements Document.

Inputs

None.

Output and Observable Effects

The program that is associated to the assignment's file extension is started.

Procedure

1. Select "Current courses" in the system header.
2. Select the course of interest.
3. Press the "View students" link.
4. Select the student of interest.
5. Select the assignment of intereset.
6. Verify that the file is loaded and the corresponding program is loaded.

6.11 View a student's submissions

Functionality

A teacher or course leader can view a student's submission.

Requirement

This test corresponds to Requirement #24 from the Requirements Document.

Inputs

None.

Output and Observable Effects

A list over the student's submissions.

Procedure

1. Select "Current courses" in the system header.
2. Select the course of interest.
3. Press the "View students" link.
4. Select the student of interest.
5. Verify that submissions made by the student are displayed.

6.12 List all submissions in a course

Functionality

Enables teachers an course leaders to view all of the submitted submissions.

Requirement

This test corresponds to Requirement #25 from the Requirements Document.

Inputs

None.

Output and Observable Effects

A list over the submissions is shown.

Procedure

1. Select "Current courses" in the system header.
2. Select the course of interest.
3. Press the "View all submissions" link.
4. Verify that a list of all submissions made in the course is displayed.

6.13 Set a grade

Functionality

Allows the teachers and course leaders to set a grade for a submission or course for a specific student.

Requirement

This test corresponds to Requirement #26 from the Requirements Document.

Inputs

The grade that the student has earned and optional comments on the work.

Output and Observable Effects

The corresponding assignment has a grade after submitted grade.

Procedure

1. Select "Current courses" in the system header.
2. Select the course of interest.
3. Press the "View students" link.
4. Select the student of interest.
5. Select the assignment of interest.
6. Fill in the "Grade:" textbox the corresponding grade the student have earned.
7. Optional: The teacher could fill in a comment about the work.
8. Press the "Submit" button.
9. Verify that the grade is set.

6.14 Add teacher

Functionality

Lets the course leader add a teacher to the course that the course leader administrates.

Requirement

This test corresponds to Requirement #28 from the Requirements Document.

Inputs

The username of the teacher.

Output and Observable Effects

A new teacher is added to the teachers squad.

Procedure

1. Select "Current courses" in the system header.
2. Select the course of interest.
3. Select "Add teacher" link.
4. Enter the name of the teacher to be added in the "New teachers username:" box
5. Press "Add teacher" button.
6. Verify that the added user is teacher in course.

6.15 Remove teacher

Functionality

Lets the course leader remove an existing teacher from the course that the course leader administrates.

Requirement

This test corresponds to Requirement #28 from the Requirements Document.

Inputs

None.

Output and Observable Effects

A teacher is removed from the teachers squad.

Procedure

1. Select "Current courses" in the system header.
2. Select the course of interest.
3. Select "Remove teacher" link.
4. Select "remove" on the corresponding teacher to remove the teacher.
5. Verify that the user is no longer teacher in course.

6.16 Edit course description

Functionality

Edit the description of the course.

Requirement

This test corresponds to Requirement #29 from the Requirements Document.

Inputs

New course description.

Output and Observable Effects

The course description is changed.

Procedure

1. Select "Current courses" in the system header.
2. Select the course from the list of active courses.
3. Select "Edit course description" from the course page.
4. Enter the new course description.
5. Press the "Save changes" button.
6. Verify on the course page that the description is changed.

6.17 Create assignment

Functionality

Add an assignment to a course.

Requirement

This test corresponds to Requirement #30 from the Requirements Document.

Inputs

Assignment name, description, start date, soft deadline, hard deadline

Output and Observable Effects

Assignment is added to the assignment list of the course.

Procedure

1. Select "Current courses" in the system header.
2. Select the course from the list of active courses.
3. Select "Add assignment" from the course page.
4. Enter Assignment name and description and select a start date, soft deadline and hard deadline
5. Press the "Save changes" button.
6. Verify on the course page that the assignment is added.

6.18 Add user

Functionality

Adds a new user to the system.

Requirement

This test corresponds to Requirement #38 from the Requirements Document.

Inputs

Username, password, personal identification number

Output and Observable Effects

The user is added to the system.

Procedure

1. Select "Add user" from the start page.
2. Enter Username, password, personal identification number
3. Press the "Save changes" button.
4. Log out from the system.
5. Verify that the user exist by logging in as the new user.

6.19 Remove user

Functionality

Removes a user from the system.

Requirement

This test corresponds to Requirement #38 from the Requirements Document.

Inputs

Username.

Output and Observable Effects

The user is removed from the sytem.

Procedure

1. Select ”Remove user” from the start page.
2. Enter the username of the user to be removed.
3. Press the ”Remove user” button.
4. Log out from the system.
5. Verify that the user is removed by trying to log in as the user.

6.20 Add course

Functionality

Adds a new course to the sytem.

Requirement

This test corresponds to Requirement #39 from the Requirements Document.

Inputs

Course name, course description.

Output and Observable Effects

Course is added to the course list.

Procedure

1. Select "Add course" from the start page.
2. Enter course name and course description.
3. Press the "Create course" button.
4. Select "Find course" from the system header.
5. Enter the course name in the search field.
6. Verify that the course is listed.

6.21 Remove course

Functionality

Removes a course from the system.

Requirement

This test corresponds to Requirement #39 from the Requirements Document.

Inputs

Course name.

Output and Observable Effects

The course is removed from the system.

Procedure

1. Select "Remove course" from the start page.
2. Enter course name.
3. Press the "Remove course" button.
4. Select "Find course" from the system header.
5. Enter the course name in the search field.
6. Verify that the course is not listed.

6.22 Set course as inactive

Functionality

The course is marked as inactive.

Requirement

This test corresponds to Requirement #40 from the Requirements Document.

Inputs

Course name.

Output and Observable Effects

Removes the course from the active courses list.

Procedure

1. Select "Set course as inactive" from the start page.
2. Enter course name.
3. Press the "Set course as inactive" button.
4. Select "Find course" from the system header.
5. Enter the course name in the search field.
6. Verify that the course is not listed.

6.23 Set course as active

Functionality

The course is marked as active.

Requirement

This test corresponds to Requirement #40 from the Requirements Document.

Inputs

Course name.

Output and Observable Effects

Adds the course to the active courses list.

Procedure

1. Select "Set course as active" from the start page.
2. Enter course name.
3. Press the "Set course as active" button.
4. Select "Find course" from the system header.
5. Enter the course name in the search field.
6. Verify that the course is listed.

6.24 Send message

Functionality

User sends a message to other user(s) of the system.

Requirement

This test corresponds to Requirement #42 from the Requirements Document.

Inputs

Recipient username(s), message subject and message

Output and Observable Effects

The message is sent to Recipients.

Procedure

1. Select "Inbox" in the system header.
2. Select "Compose message" in the message view.
3. Enter recipent(s) username(s), message subject, and message.
4. Press the "Send message" button.
5. Verify that the confirmation message is shown.