

Design Document

Res version 2

Group 24

Erika Ekberg (eekeberg@kth.se)

Mattias Johansson (mattias4@kth.se)

Enault Adrien-Marie (enault@kth.se)

Table of Contents

1. Introduction.....	3
2. System Overview.....	4
2.1 General Description.....	4
2.2 Overall architecture description.....	4
2.3 Detailed architecture.....	5
3. Introduction.....	7
3.1 Assumptions and Dependencies.....	7
3.2 General constraints.....	7
4. Graphical User Interface.....	8
5. Design Details.....	19
5.1 Class Responsibility Collaborator (CRC) Cards.....	19
5.2 Class Diagram.....	22
5.3 State chart.....	23
5.4 Interaction Diagrams.....	24
5.5 Detailed Design	27
5.5.1 Database Design.....	27
5.5.2 Classes.....	29
5.5.3 References to RD.....	35
5.6 Package Diagram.....	36
6. Functional Test Cases.....	37

1. Introduction

In a school such as KTH, keeping track of the results of every student, each of whom follows its own and generally unique cursus can be really delicate without the proper tools, and all these results have to be centralized in one way or another, as it is not conceivable to have to gather them from several different places when needed. Such a tool already exists (the res system), but it is a not really user-friendly console-based system. Our new system offers a much more clear web-based interface, as well as new useful functionalities for students.

This Document describes in details the intended design of the system so the programmers will know what has to be done. The readership of this Document would typically be a programmer willing to join the project as coder or as project leader, but it could also be teams of programmer from a similar project interested in our design. So, someone with the required skill should, after having read this document, understand how the system works and start coding.

This Document deals with the overall and detailed architecture of the system, the graphical unit interface, details each class of the system and finally describes the functional test cases.

The system uses java Servlets to dynamically generates web pages, it won't be described in that document. For more information, please look at the following websites.

- official website: <http://java.sun.com/products/servlet/>
- wikipedia page: http://en.wikipedia.org/wiki/Java_Servlet

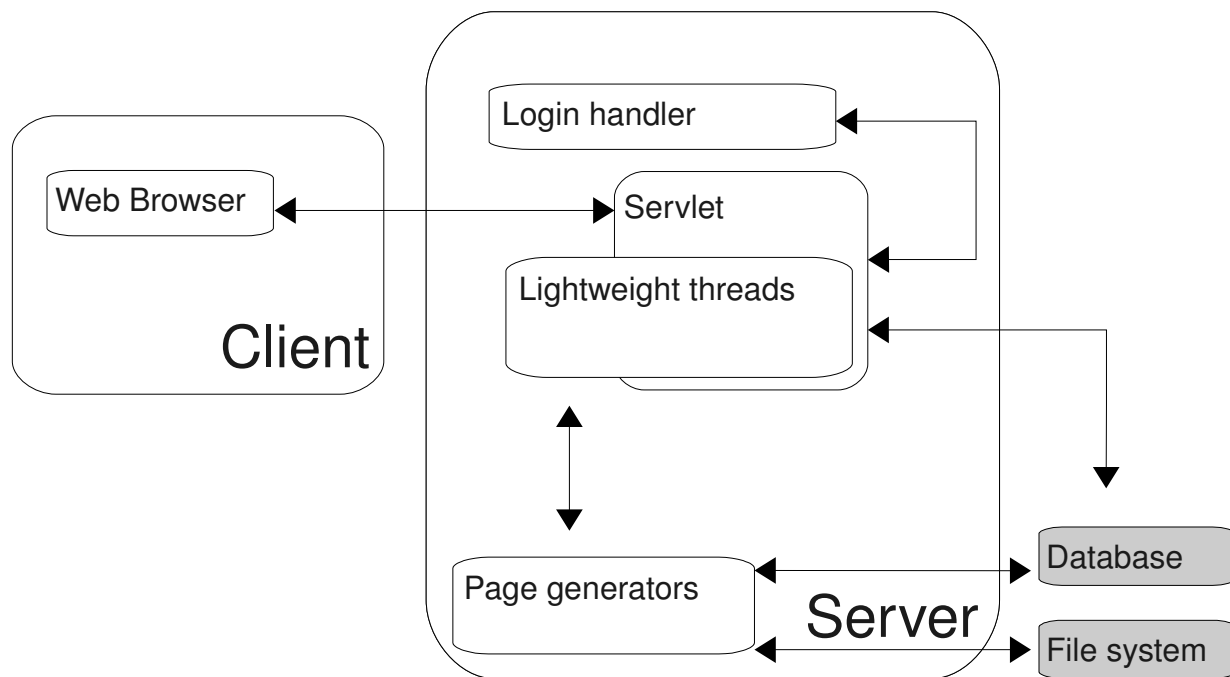
References to the Requirement Document of the same project will be made throughout this Document.

2. System Overview

2.1 General Description

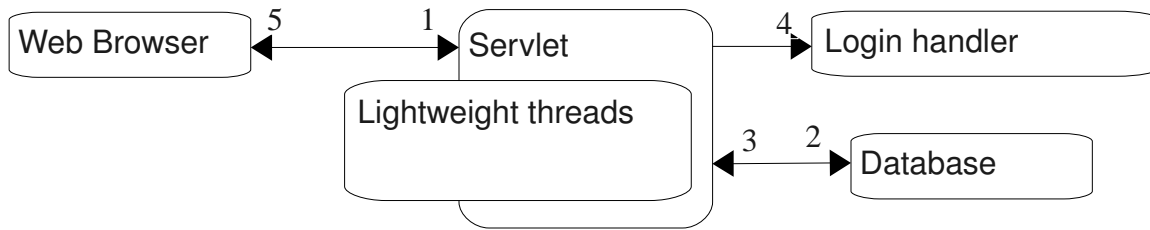
Res2 is a system that keeps track of the students' results on laboratory- / homework assignments on your university. The system exists of a server program that handles URL requests, so that a user can get access to the system from any Internet browser. We will use a SQL database to handle storage of data, so we won't have any problems with simultaneous users trying to access the data. To be able to access the system you will need a user account.

2.2 Overall architecture description



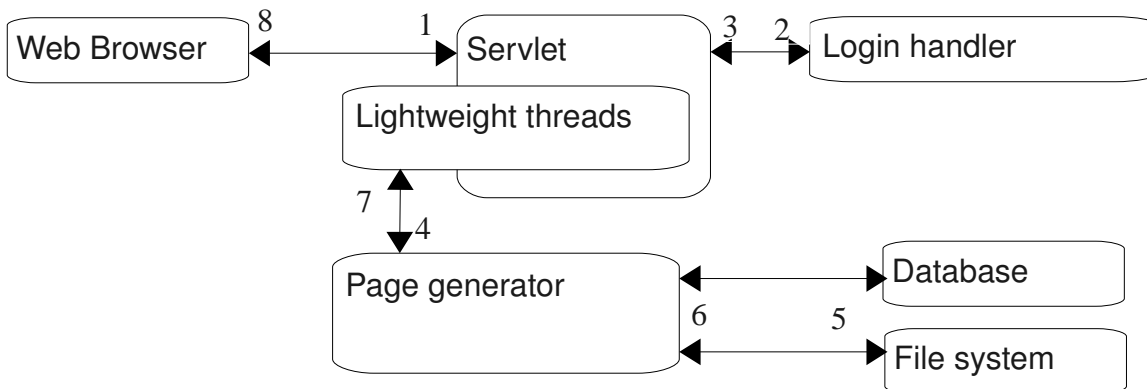
Our system consists of a Java servlet running on a server communicating with a database and a file system, to which you can connect with a web browser via a servlet. The server in itself consists of the servlet that handles the requests made by the client, create a lightweight process for each of them, and, through the different page generators, create a page corresponding to the request of the client and send it to him. It also check the identity of the user via the login handler. The user can use any standard web-browser to view the web pages.

2.3 Detailed architecture



Data flow, logging in

1. The Client sends its login information
2. The server queries the database to see if its password is valid
3. The database returns the information needed
4. The server check the information and stores and, if valid, stores the session in the login handler
5. The server notifies to the client that he now has been successfully identified
- 6.



Data flow, page browsing

1. The client sends a request through its web browser
2. The server interrogates the login handler about the identity of the client
3. The server checks if the client has the proper authorization
4. If yes, a thread is created, and through a page generator...
5. ... which queries the database and the file system...
6. ... which return the needed information to the page generator to create the page...
7. ... returns it to the servlet
8. The servlet then returns the page to the client

Control flow, client

The client can tell the servlet which page it wants it to generate

The client has control over its identity

The client has limited control over the database, depending on its access level

Control flow, server

The server has full control over the database, both reading and writing

The server has read authorization on the file system

3. Introduction

3.1 Assumptions and Dependencies

The SQL database used in this project will be postGre. The design makes it easy to change the database for another one (for example mySQL), but multi-support will be outside the scope of the project.

In addition, the server must have a Java Virtual Machine enabling the use of the Java Servlets. As the Java Servlets are usually shipped with the standard package, any recent distribution should fit.

The Server part will be tested on a computer running under Solaris X, even though it should be able to run under a lot more of OS, and it is more than likely that the development will be done on Windows-based or Linux-based platforms.

As for the client, nothing is required but a computer with a XHTML-compliant browser, such as late versions of Firefox, Internet Explorer, Safari or Opera.

3.2 General constraints

Our primary goal will be the reliability. The data at hand is quite sensitive, and loss is not acceptable. Though the safety of the data will primarily rely on the database itself (postGre SQL in our case) which should meet our expectations in terms of reliability, it is our responsibility to ensure that our functions are secure. By that, we mean that our function should not harm the data, functions which write in the database should be written with extreme precaution, and data should not be updated unless really needed. This requires that our system must not do unwanted changes "behind the back of a user". Also, our interface will have to be clear enough not to confuse our user.

This said, we will also have to keep in mind performances, as an unavailable system is a useless system.

4. Graphical User Interface

The graphical user interface of the system is a web-page. You are required to log in to access the pages. There is a menu where you can choose what action to perform, and every option in that menu will take you to a sub page with a form to fill in, or will display the information you need.

See the screenshots below for further details.

NB: This provides only the different fields and choices offered to the user, the final presentation will be different. The functions mentioned here are the functions that our page generators will call if this specific button is pushed. Actually, all the pages are (but the login page) displayed by the call of the function `display()`.

Grant and withdraw teacher access to a user

Whenever the user tries to access any page when he is not logged in, this form will be displayed (by the functions `<< accepted() >>` or `<< rejected() >>` if the login has failed at least once).



The screenshot shows a Windows Internet Explorer browser window titled "Log in - Windows Internet Explorer". The address bar displays "D:\Documents\MVK\GUIlogin.html". The browser's address bar contains the text "Log in" and a search button. Below the browser window, a login form is displayed. The form consists of three rows: the first row has a label "Username" and an empty text input field; the second row has a label "Password" and an empty text input field; the third row contains a "login" button.

The fields are named `<< username >>` and `<< password >>`.

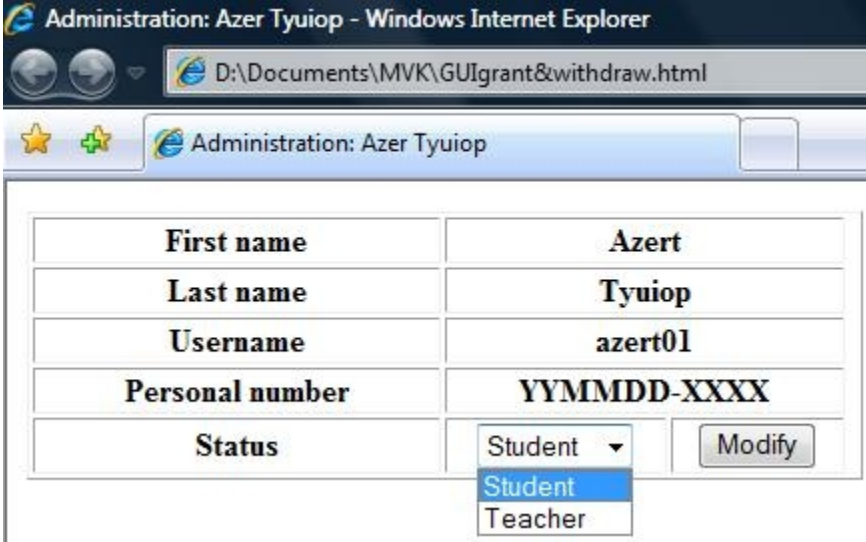
The button calls the function `<< check() >>`.

Grant and withdraw teacher access to a user

Requirements:

- " Grant access
- " Withdraw access

An administrator will be able, through the menu, to access this page, after having filled in a form asking for the user name that has to be administrated (not shown here, as not really relevant).



The screenshot shows a Windows Internet Explorer browser window titled "Administration: Azer Tyuiop". The address bar displays "D:\Documents\MVK\GUIgrant&withdraw.html". The browser's address bar shows "Administration: Azer Tyuiop". The main content area contains a form with the following fields:

First name	Azert
Last name	Tyuiop
Username	azert01
Personal number	YYMMDD-XXXX
Status	<input type="text" value="Student"/> <input type="button" value="Modify"/>

The "Status" field has a dropdown menu open, showing "Student" (highlighted) and "Teacher".

The drop menu is named « status ».

The "Modify" button calls updateAccess().

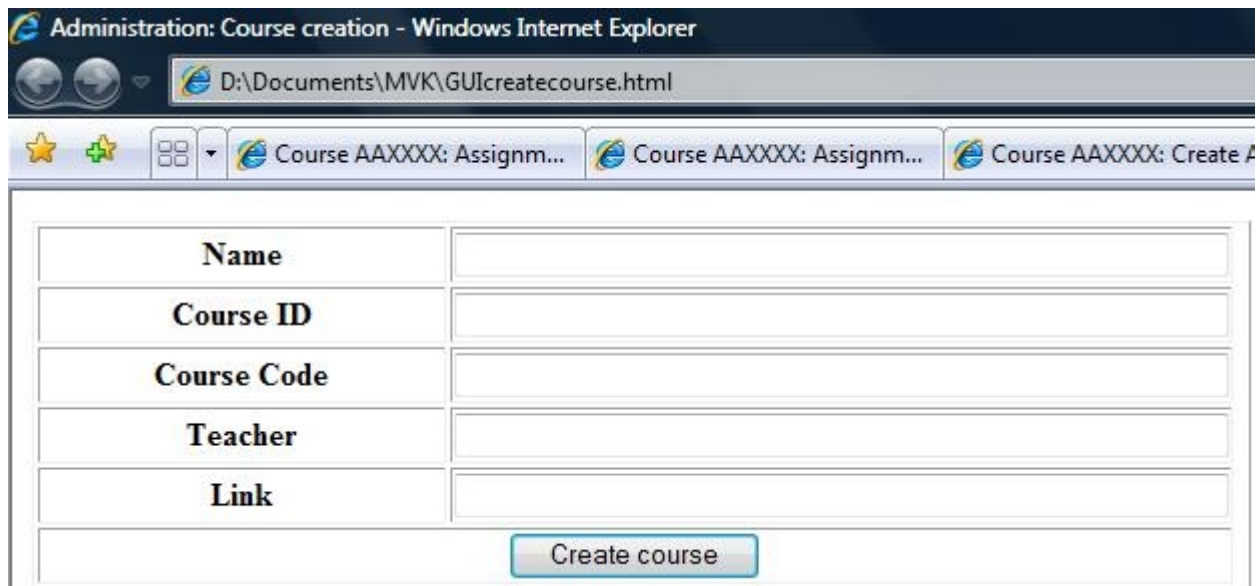
Course Creation

Requirements:

" Create course/add a course

" Request course creation

A teacher or an administrator can create a course with a slight difference. A course created by a teacher (a request) must be validated by an administrator before anyone can sign up, or any assignment added.



The screenshot shows a Windows Internet Explorer browser window titled "Administration: Course creation - Windows Internet Explorer". The address bar displays "D:\Documents\MVK\GUI\createcourse.html". The browser has three tabs open, with the active tab being "Course AAXXXX: Create A". The main content area contains a form with five input fields and a "Create course" button.

Name	<input type="text"/>
Course ID	<input type="text"/>
Course Code	<input type="text"/>
Teacher	<input type="text"/>
Link	<input type="text"/>

The fields are "name", "courseid", "coursecode", "teacher" (for a teacher, that field will automatically be filled with its name and greyed out), "link".

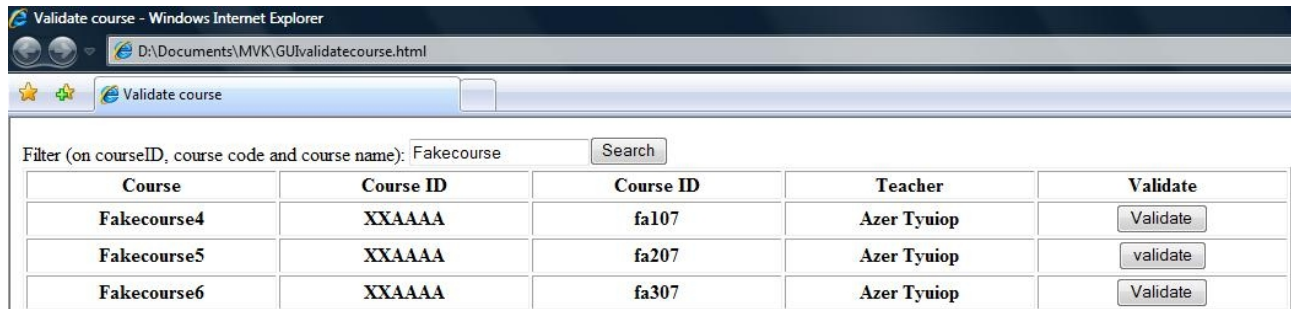
"Create course" calls createCourse() for an administrator and requestCourse() for a teacher.

Validate Course

Requirements:

" Validate course

When a teacher has requested a course creation, as long as it has not been validated by an administrator, nobody can join it and no assignment can be created for it.



Validate course - Windows Internet Explorer
D:\Documents\MVK\GUI\validatecourse.html
Validate course

Filter (on courseID, course code and course name): Fakecourse

Course	Course ID	Course ID	Teacher	Validate
Fakecourse4	XXAAAA	fa107	Azer Tyuiop	<input type="button" value="Validate"/>
Fakecourse5	XXAAAA	fa207	Azer Tyuiop	<input type="button" value="validate"/>
Fakecourse6	XXAAAA	fa307	Azer Tyuiop	<input type="button" value="Validate"/>

The field name is called "filter".

"Update" calls filter(). "Validate" calls validateCourse().

Update course

Requirements:

" Assign assistant

" Remove assistant

When a teacher lists the courses, for the courses where he is the teacher, he has the option `Update` instead of `Join`. (See below for the list course/join course). For an assistant, all the options will be grayed out but the `Update` button of the Assignments row.

Name	name	<input type="text"/>	<input type="button" value="Update"/>
Course ID	courseid	<input type="text"/>	<input type="button" value="Update"/>
Course Code	coursecode	<input type="text"/>	<input type="button" value="Update"/>
Teacher	teacher	<input type="text"/>	<input type="button" value="Update"/>
Link	link	<input type="text"/>	<input type="button" value="Update"/>
Add assistant	<input type="text"/>		<input type="button" value="Add"/>
Remove assistant	Azert Tyuiop (azert04) ▾		<input type="button" value="Remove"/>
Assignments	LAB1 ▾		<input type="button" value="Add"/> <input type="button" value="Update"/>
<input type="button" value="List students"/>			

The fields are named "name", "courseid", "coursecode", "teacher" (grayed out for a teacher), "link", "assistant". The drop menu of the Remove assistant row is named "assistantList" and contains the names and usernames of the assistants for the course. The drop menu of the Assignments row is named "assignments" and contains the assignments of the course. In the "assistant" field, the user is supposed to give the unique "username" of the assistant he wants to add (such as "azert04").

The "Update" buttons will call the methods `update*()`, where * is the name of the corresponding field. On the Assistant rows, "Add" will call `addAssistant()`. "Remove" will call `removeAssistant()`. On the last row (Assignments) "Add" will lead to the `Create assignment` page and "Update" will lead to the `Update Assignment` page. The "List students" button will lead you to the `List students` page.

Join course / list course

Requirements:

"Join course

You access this page by clicking the `List course` item on the menu.



The field name is called "filter".

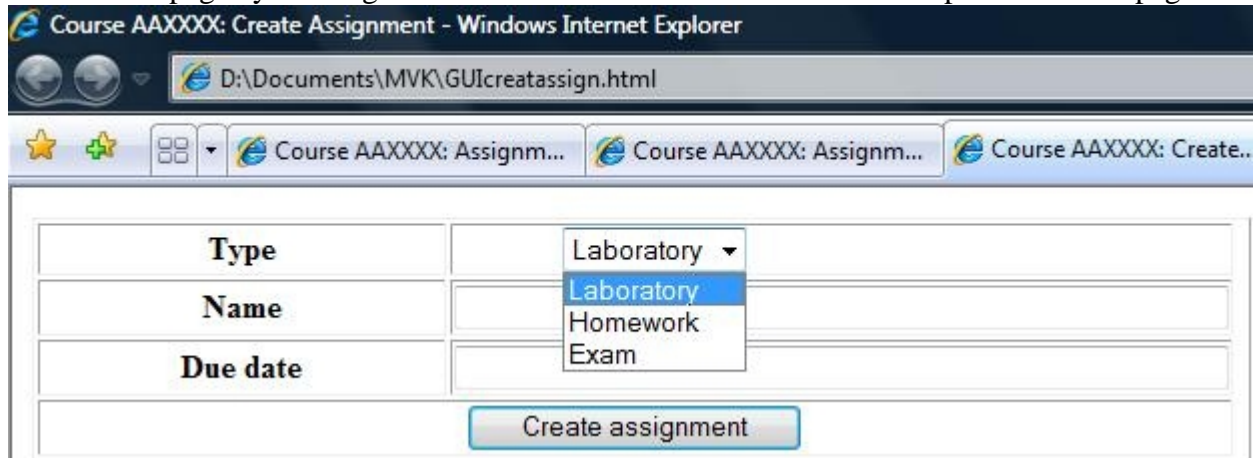
"Search" calls `filter()`. "Join" calls `joinCourse()`. If the student has already joined the course, he can see Fakecourse1 via the `See` button, which will lead him to the page `See assignments` (see below). Here, our user, Azer Tyu is offered to update his course Fakecourse3, which will lead him to the `Update course` page, because he is either the teacher or an assistant of this course.

Create assignment

Requirements:

" Create assignment

You access this page by clicking the button `Add` of the last row of the `Update course` page.



The screenshot shows a web browser window titled "Course AAXXXX: Create Assignment - Windows Internet Explorer". The address bar shows the URL "D:\Documents\MVK\GUIcreatassign.html". The browser has three tabs open, all related to "Course AAXXXX". The main content area displays a form with three input fields: "Type", "Name", and "Due date". The "Type" field has a dropdown menu open, showing three options: "Laboratory", "Homework", and "Exam". Below the form is a button labeled "Create assignment".

The drop menu is called "type". The fields are named "name", "date".

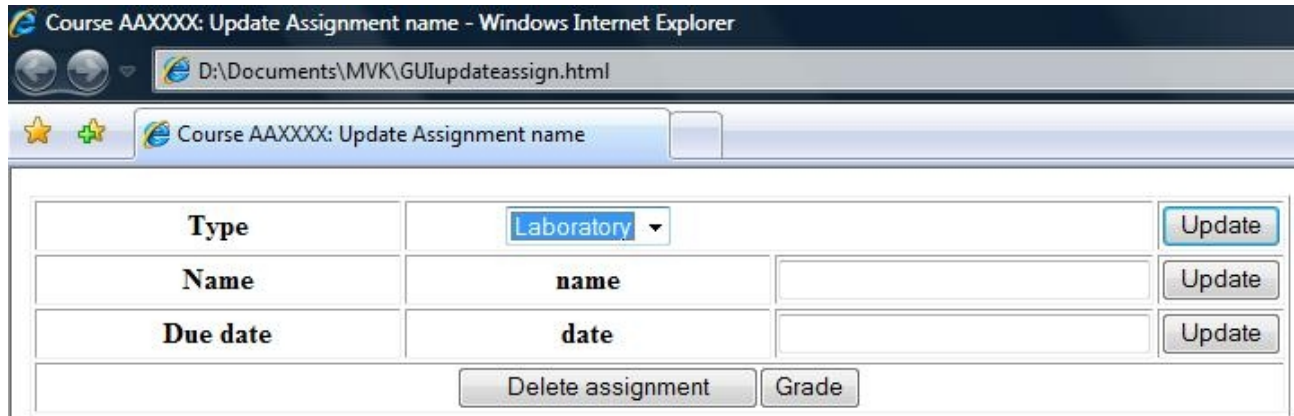
"Create assignment" calls the function `createAssignment()`.

Update assignment

Requirements:

" Update assignment

You access this page by clicking the button `Update` of the last row of the `Update course` page.



The screenshot shows a web browser window titled "Course AAXXX: Update Assignment name - Windows Internet Explorer". The address bar shows the URL "D:\Documents\MVK\GUIupdateassign.html". The browser window contains a form with the following fields and buttons:

Type	Laboratory	Update
Name	name	Update
Due date	date	Update
Delete assignment		Grade

The drop menu is called "type". The fields are named "name", "date".

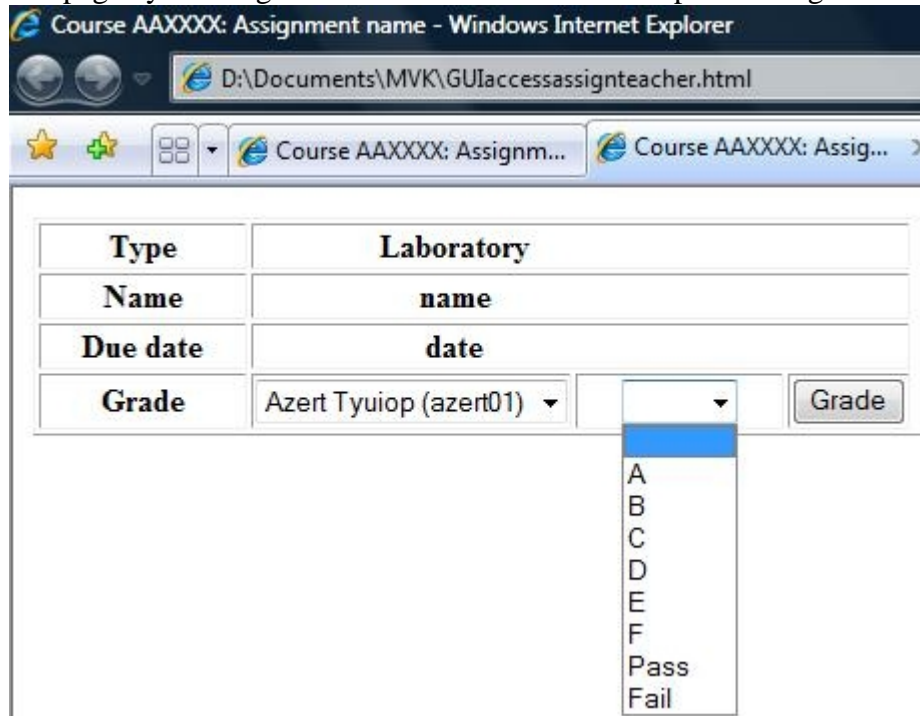
The "Update" buttons will call the methods `update*()`, where `*` is the name of the corresponding field. "Delete assignment" calls the function `deleteAssignment()`. `Grade` leads you to the `Grade assignment` page (see below). For an assistant, all the options will be grayed out but the `Grade` button.

Grade assignment

Requirements:

- " Access assignment
- " Update grades

You can access this page by clicking the `Grade` button of the `Update Assignment` page.



The drop menu containing the names of the students who have signed up is called "students". The other one containing the grade is called "grade".

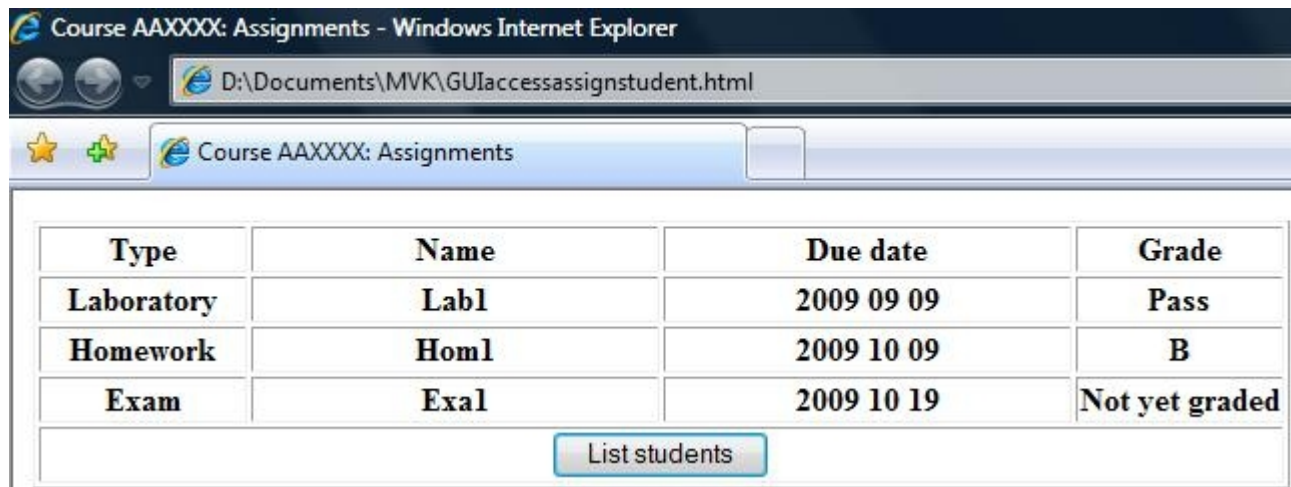
"Grade" will call `grade()`.

See assignments

Requirements:

" Show grades

You can access this page by clicking the button `See` of the `List courses` page.



The screenshot shows a Windows Internet Explorer browser window titled "Course AAXXXX: Assignments". The address bar displays the URL "D:\Documents\MVK\GUIaccessassignstudent.html". The browser's address bar shows "Course AAXXXX: Assignments" with a search box to its right. Below the browser window is a table with four columns: "Type", "Name", "Due date", and "Grade". The table contains three rows of data: "Laboratory" with name "Lab1" and due date "2009 09 09" (Grade "Pass"), "Homework" with name "Hom1" and due date "2009 10 09" (Grade "B"), and "Exam" with name "Exal" and due date "2009 10 19" (Grade "Not yet graded"). Below the table is a button labeled "List students".

Type	Name	Due date	Grade
Laboratory	Lab1	2009 09 09	Pass
Homework	Hom1	2009 10 09	B
Exam	Exal	2009 10 19	Not yet graded

[List students](#)

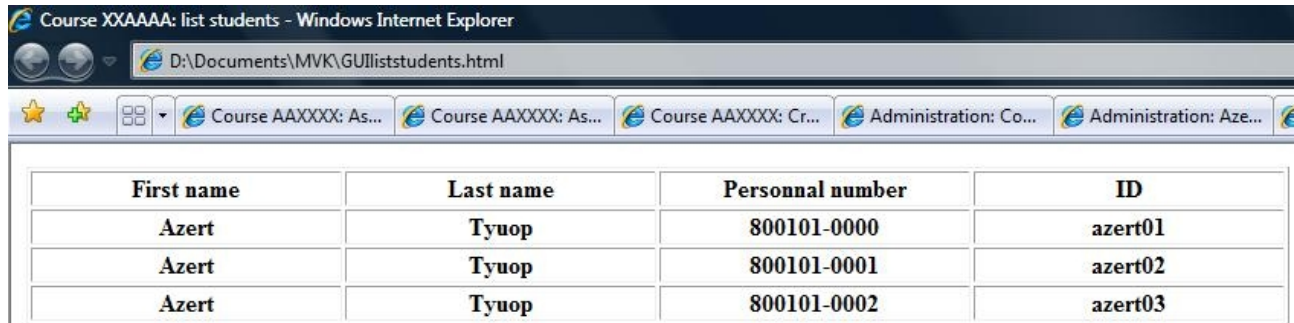
The `List students` button leads you to the `List student` page.

List students

Requirements:

" List students

You can access this page by clicking the [List students](#) button on the [See assignments](#) page or the [Update course](#) page. This lists the students of course from where you come.



The screenshot shows a Windows Internet Explorer browser window with the title "Course XXAAAA: list students". The address bar displays "D:\Documents\MVK\GUIliststudents.html". The browser has several tabs open, including "Course AAXXXX: As...", "Course AAXXXX: Cr...", and "Administration: Aze...". The main content area displays a table with the following data:

First name	Last name	Personnal number	ID
Azert	Tyuop	800101-0000	azert01
Azert	Tyuop	800101-0001	azert02
Azert	Tyuop	800101-0002	azert03

5. Design Details

5.1 Class Responsibility Collaborator (CRC) Cards

User	
Keep track of the user logged in for identification purpose: <ul style="list-style-type: none">● Name● KTH ID● Personal security number● Mail address● Status (like D-04 etc.)● User access level● List of courses Retrieve this info from database	Database

Database	
Acts as an interface between objects and the SQL Database. Keep track of database address.	User

Assignment	
Keep track of: <ul style="list-style-type: none">● Type● Name● Due date● URL Update grades Display grades of users	Database User

Login	
User authentication Create a user (session) object Display the main page when logged in	User MainPage

Course	
Keep track of: <ul style="list-style-type: none"> ● Teacher ● Course name ● Course ID/code ● List of students ● List of assignments ● List of assistants ● URL ● If valid or not 	Database User Assignment

Page generation

HTMLPage	
Providing framework page	Database User

AdministrateUserPage	
Display a page with a form that handles user administration	Database User

CourseCreationPage	
Display a page with a form that handles course creation	Database User

ValidateCoursePage	
Display a page with a form that handles course validation	Database User

UpdateCoursePage	
Display a page with a form that handles course	Databases

updates	User
---------	------

JoinCoursePage	
Display a page with a form that handles course joining	Database User

CreateAssignmentPage	
Display a page with a form that handles assignment creation	Database User

UpdateAssignmentPage	
Display a page with a form that handles assignment updates	Database User

GradeAssignmentPage	
Display a page with a form that handles assignment grading	Database User

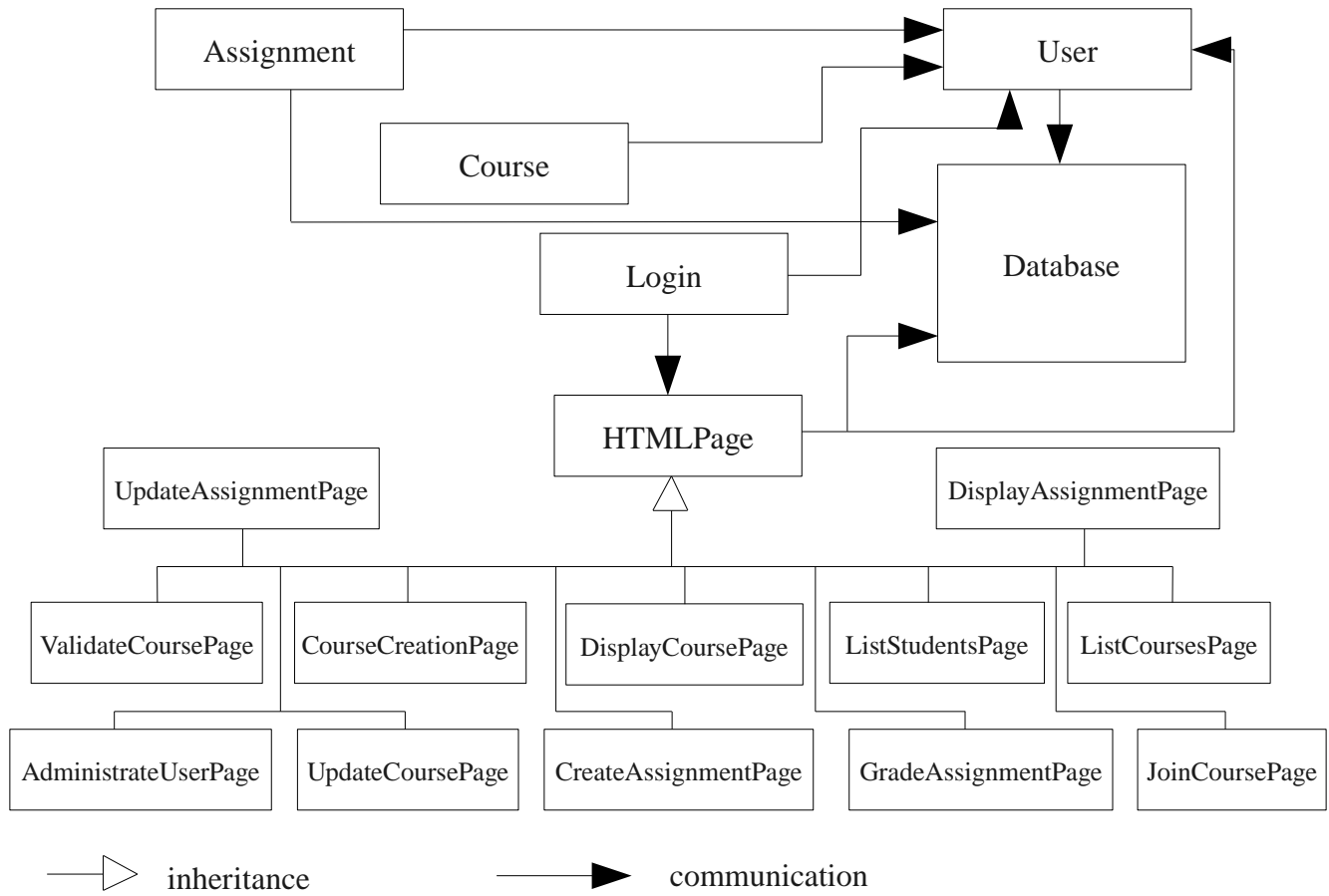
DisplayAssignmentPage	
Display information about an assignment	Database User

ListStudentsPage	
Display a page with a form that handles listing of students	Database User

ListCoursesPage	
Display a page with a form that handles course listing	Database User

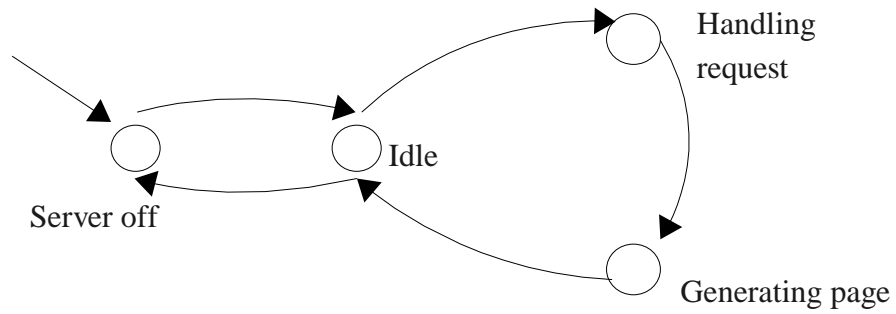
DisplayCoursePage	
Display information about the course	Database User

5.2 Class Diagram

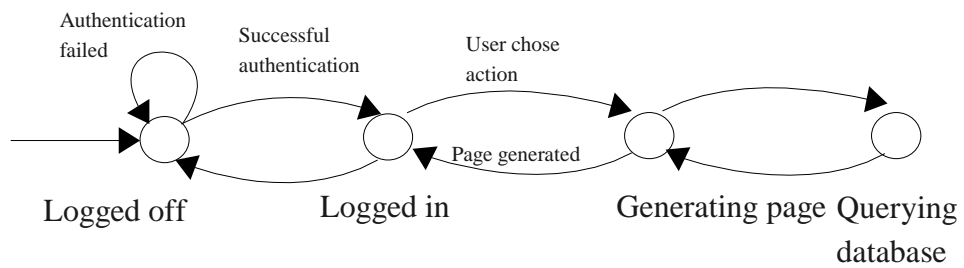


5.3 State chart

Server



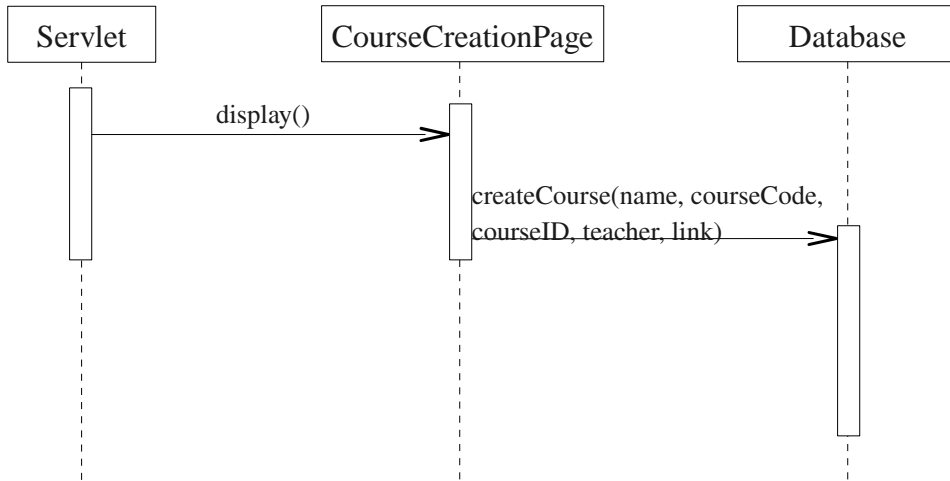
One session state chart



5.4 Interaction Diagrams

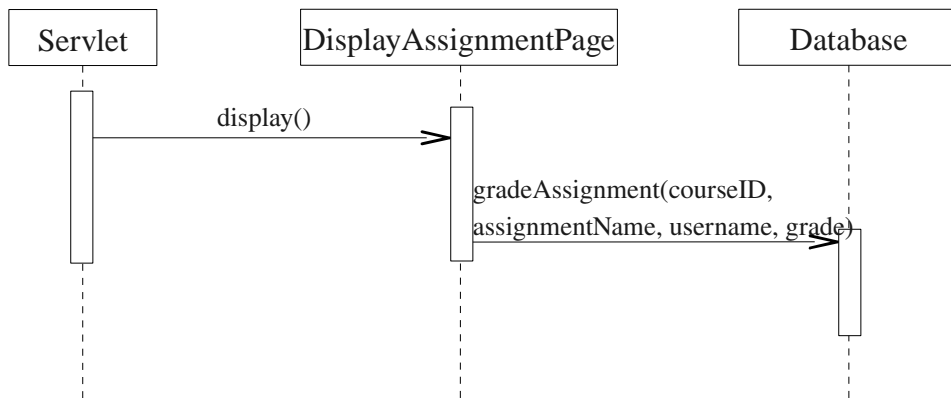
Add a course

References use case on page 20, Requirements Document.



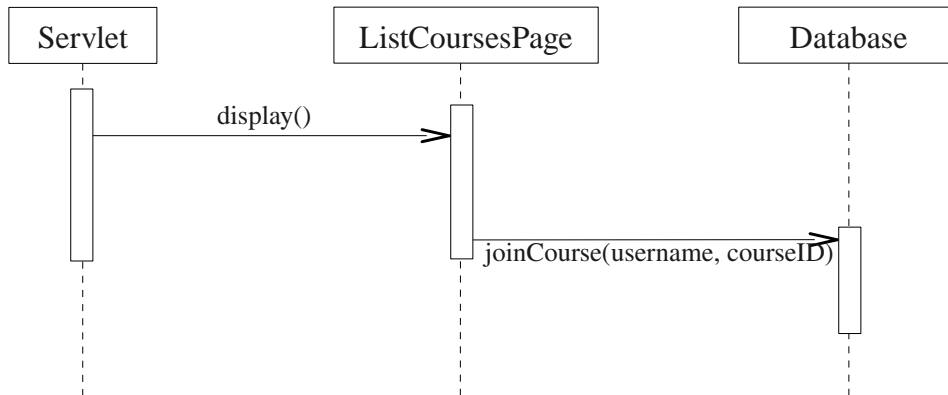
Report a grade

References use case on page 21, Requirements Document

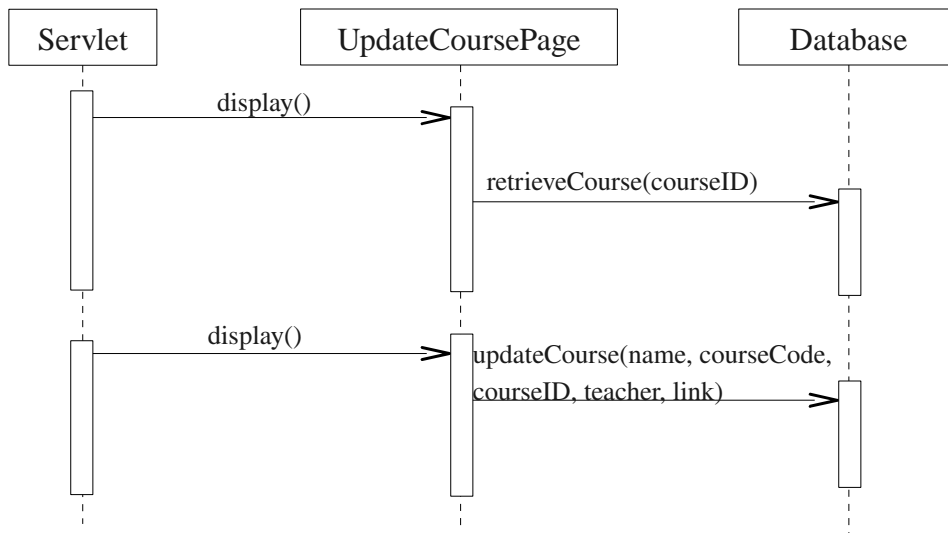


Register in a course

References use case on page 23, Requirements Document.

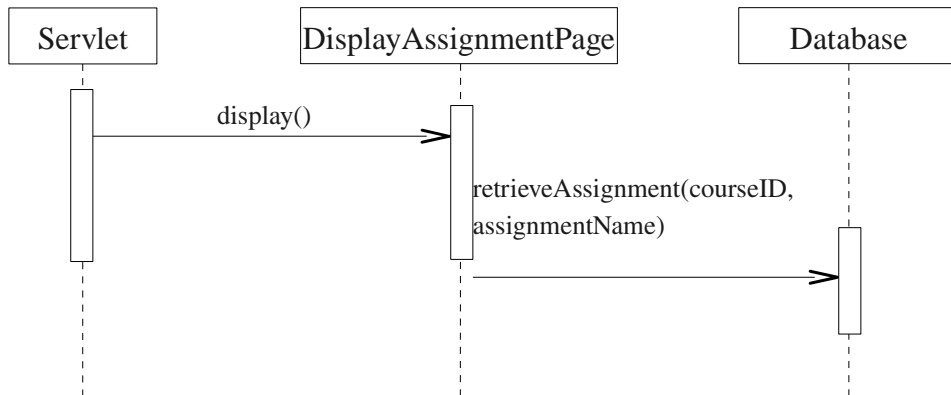


Update a course

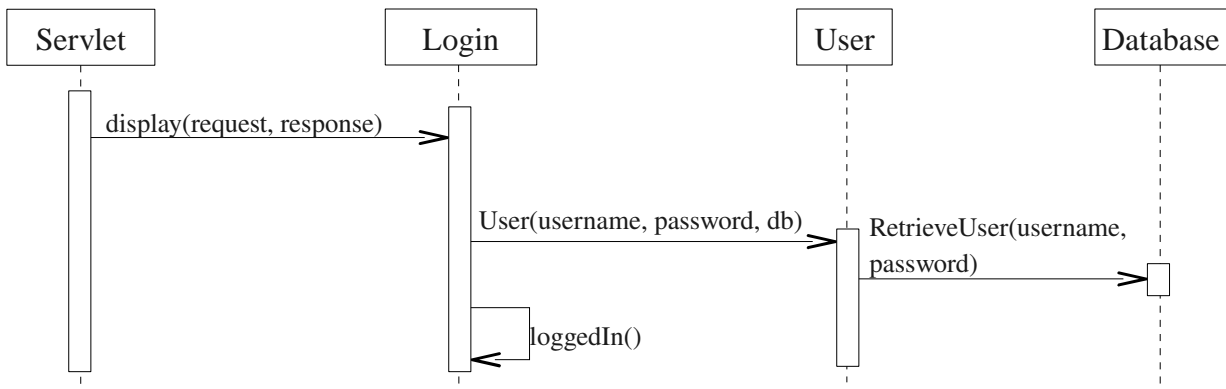


View course results

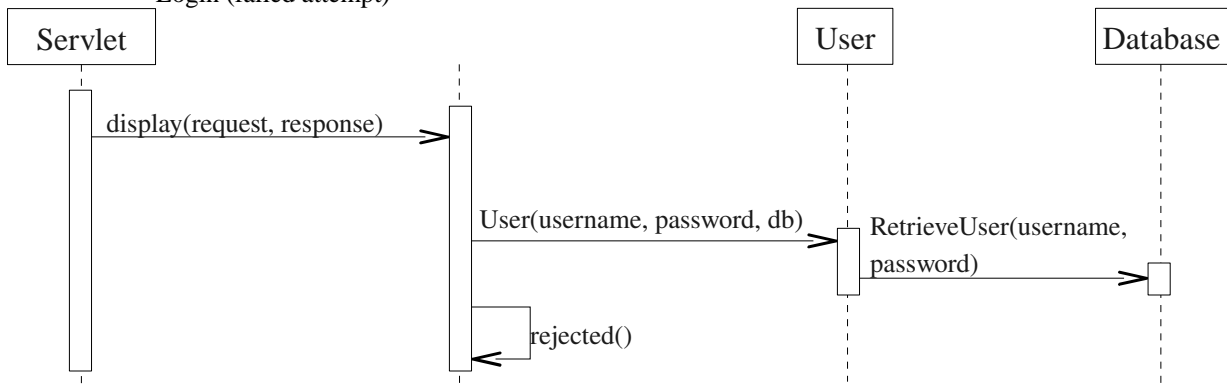
References use case on page 26, Requirements Document.



Login (successful attempt)

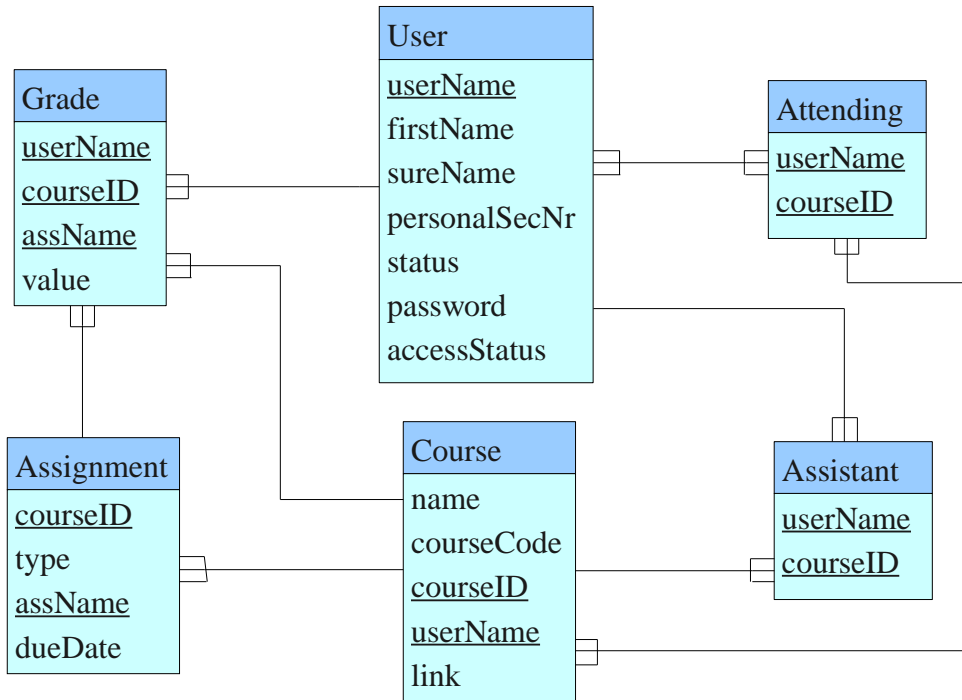


Login (failed attempt)



5.5 Detailed Design

5.5.1 Database Design



User:

Column:	Type:	Constraints
<u>userName</u>	varchar(20)	primary key
surName	varchar(20)	not null
firstName	varchar(20)	not null
personalSecNr	varchar(10)	not null
status	varchar(4)	
password	varchar(40)	not null
accessStatus	varchar(10)	not null

Course:

Column:	Type:	Constrains
<u>courseID</u>	varchar(10)	primary key
name	varchar(20)	not null
courseCode	varchar(6)	not null
userName	varchar(20)	not null
link	varchar(100)	

Assistant:

Column:	Type:	Constrains
<u>userName</u>	varchar(20)	foreign key
<u>courseID</u>	varchar(10)	foreign key

Attending:

Column:	Type:	Constrains
<u>userName</u>	varchar(20)	foreign key
<u>courseID</u>	varchar(10)	foreign key

Assignment:

Column:	Type:	Constrains
<u>courseID</u>	varchar(10)	foreign key
<u>assName</u>	varchar(20)	primary key
type	varchar(20)	not null
dueDate	varchar(10)	not null

Grade:

Column:	Type:	Constrains
<u>courseID</u>	varchar(10)	foreign key
<u>userName</u>	varchar(20)	foreign key
<u>assName</u>	varchar(20)	foreign key
value	varchar(10)	not null

5.5.2 Classes

Class: User

- Boolean loggedIn;
- String username;
- String firstName;
- String surName;
- String personalNumber;
- String status;
- String accessStatus;
- Database db;

+ User(String username, String password, Database db);
//Attempts to login the user and retrieve the user information from the database

Class: Login

-

+ boolean display(ServletResponse response);
//Displays the login screen
+ boolean rejected(ServletResponse response);
//Tells the user that the login failed
+ boolean loggedIn(ServletResponse response);
//Displays the main page (And main menu) after logging in

Class: HTMLPage

- User user;
- Database db;

+ HTMLPage(ServletRequest request, ServletResponse response, Database db, User user);
+ boolean display();
//Displays the header of a subpage, and calls the overloaded body() .
+ abstract boolean body();

Class: UpdateAssignmentPage

-

+ boolean body();
 //Displays the body of this sub page

Class: DisplayAssignmentPage

-

+ boolean body();
 //Displays the body of this sub page

Class: ValidateCoursePage

-

+ boolean body();
 //Displays the body of this sub page

Class: CourseCreationPage

-

+ boolean body();
 //Displays the body of this sub page

Class: DisplayCoursePage

-

+ boolean body();
 //Displays the body of this sub page

Class: ListStudentsPage

-

+ boolean body();
 //Displays the body of this sub page

Class: ListCoursePage

-

+ boolean body();
//Displays the body of this sub page

Class: AdministrateUserPage

-

+ boolean body();
//Displays the body of this sub page

Class: UpdateCoursePage

-

+ boolean body();
//Displays the body of this sub page

Class: CreateAssignmentPage

-

+ boolean body();
//Displays the body of this sub page

Class: GradeAssignmentPage

-

+ boolean body();
//Displays the body of this sub page

Class: JoinCoursePage

-

+ boolean body();
//Displays the body of this sub page

Class: Database

```
- String address;  
    //The address to the database  
- String username;  
    //The username to the database  
- String password;  
    //The password for the database  
- Database db;  
    // A handle to the databaseconnection  
  
+ boolean connect();  
    //Attempts to connect to the database, return true if successful, false otherwise void disconnect();  
    //Disconnects from the database server  
+ boolean updateUser(String username, String firstName, String surname, String personalNumber,  
String status, String password, String accessStatus);  
    //Attempts to update the user information, returns true if successful (Administrator), false  
    //otherwise.  
+ boolean createUser(String username, String firstName, String surname, String personalNumber,  
String status, String password, String accessStatus);  
    //Creates a new user  
+ boolean CreateCourse(String name, String courseCode, String courseID, String teacher, String link);  
    //This attempts to create a course, true if successful, false otherwise  
+ boolean validateCourse(String courseID);  
    // Validates the course requested by a teacher  
+ boolean requestCourse(String name, String courseCode, String courseID, String teacher, String  
link);  
    //Sends a request to create a course, must be validated by an administrator  
+ Course retrieveCourse(courseID);  
    //Returns an object to describe this course  
+ boolean createAssignment(String courseID, String type, String name, String dueDate);  
    //Attempts to create a new assignment attached to the specified course. True if successful, false if  
    //failed (Due to you not being the teacher of the course or for whatever reason)  
+ boolean updateAssignment(String courseID, String type, String name, String dueDate);  
    //NULL fields are preserved in the database, the other fields are updated. Returns true if successful  
    //and false if it failed  
+ boolean addAssistant(String username, String courseID);  
    //Tags the user as assistant in the course specified  
+ boolean removeAssistant(String username, String courseID);  
    // Withdraw the assistant permissions from this user in the course specified  
+ boolean isAssistant(String username, String courseID);
```



```

    //Returns true if the specified user is an assistant in the given course
+ vector<String[]> listStudents(String courseID);
    //Returns a list of students attending to the given course
+ boolean gradeAssignment(String courseID, String assignmentName, String username, String grade);
    // Updates the grade of a specific assignment handed in by the given user
+ Assignment retrieveAssignment(String courseID, String assignmentName);
    // This call will return an object describing the assignment and a list of students attending to the
    //relevant course
+ Course[] listCourses();
    // Returns a list of all courses
+ boolean joinCourse(String username, String courseID);
    // The user joins the course, true if successful
+ Database(String address, String username, String password);
    //Constructs the database object, storing the login information

```

Class: Assignment

```

- String courseID;
- String type;
- String name;
- String dueDate;
- String username;
- String grade;

```

```

+ String getCourseID();
    // Returns what course this assignment belongs to
+ String getType();
    //Returns what type of assignment it is
+ String getName();
    //Returns the name of the assignment
+ String getDueDate();
    //Returns the due date of the assignment
+ String getUsername();
    //Returns the username of the person handed in this assignment
+ String getGrade();
    // Returns the grade of this assignment

```

Class: Course

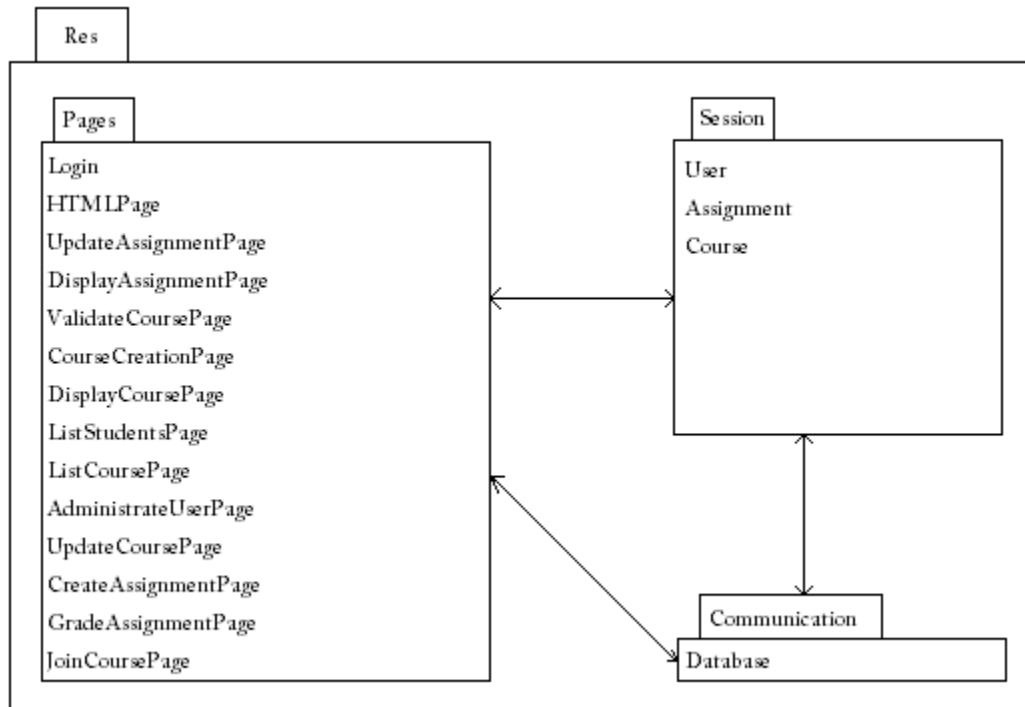
- String name;
- String courseCode;
- String courseID;
- String teacher;
- String link;

+ String getName();
 // Returns the name of the course
+ String getCourseCode();
 //Returns the coursecode
+ String getCourseID();
 // Returns the course ID
+ String getTeacher();
 // Returns the username of the teacher of this course
+ String getLink();
 // Returns the URL to the course page

5.5.3 References to RD

Requirement from the RD (page 9)	Implementation in the DD
Grant access	Class: AdministrateUserPage
Withdraw access	Class: AdministrateUserPage
Create assignments	Class: CreateAssignmentPage
Update assignments	Class: UpdateAssignmentPage
List students	Class: ListStudentsPage
Add course	Class: CourseCreationPage
Assign assistants	Class: UpdateCoursePage
Remove assistants	Class: UpdateCoursePage
Update grades	Class: GradeAssignmentPage
Join a course	Class: JoinCoursePage
List Courses	Class: ListCoursePage
Show grades	Class: DisplayCoursePage
Create course	Class: CourseCreationPage
Validate course	Class: ValidateCoursePage
Course creation	Class: CourseCreationPage
Access assignments	Class: DisplayAssignmentPage

5.6 Package Diagram



6. Functional Test Cases

Grant access

Description of the functionality being tested:

The system administrator should be able to grant a user teachers-access.

Reference to RD:

This functionality was first described on page 9 of the Requirements Document.

Inputs and preconditions:

There should exist atleast one user in the system without teacher's-access. The tester need to be logged in as administrator.

Expected output:

After the test the user who was granted teacher-access should be able to request a new course to be created (only teachers can do this).

Procedure:

1. Select user-administration from the main menu.
2. Pick the user you want to grant access.
3. Select Teacher from the drop list next to Status .
4. Press Modify

Withdraw access

Description of the functionality being tested:

The system administrator should be able to withdraw a user teachers-access.

Reference to RD:

This functionality was first described on page 9 of the Requirments Document.

Inputs and preconditions:

There should exist atleast one user in the system with teacher's-access. The tester need to be logged in as administrator.

Expected output:

After the test the user who was withdrawn teacher-access should not be able to request a new course to be created (only teachers can do this).

Procedure:

1. Select user-administration from the main menu.
2. Pick the user you want to withdraw the access from.
3. Select student from the drop list next to Status .
4. Press Modify

Create assignment

Description of the functionality being tested:

Teachers should be able to create assignment for their courses.

Reference to RD:

This functionality was first described on page 9 of the Requirments Document.

Inputs and preconditions:

The tester need to be logged in as a teacher with at least one course that he is leading.

Expected output:

After the test, the tester should be able to see the new assignment on the courses page.

Procedure:

1. Select `My Courses` from the main menu.
2. Pick the course you want to add the assignment too, and click it.
3. Click `Add new assignment` on the course information page.
4. Fill in `Type` , `Name` and `Due Date` ,press `Create assignment`

Update assignment

Description of the functionality being tested:

Teachers should be able to update assignments for their courses.

Reference to RD:

This functionality was first described on page 9 of the Requirments Document.

Inputs and preconditions:

The tester need to be logged in as a teacher with at least one course that he is leading. The course need to have a assignment.

Expected output:

After the test, the tester should be able to see the new assignment info on the courses page.

Procedure:

1. Select `My Courses` from the main menu.
2. Pick the course you want to update the assignment for.
3. Click `Update assignment` next to the assignment you want to update.
4. Do the changes you want and klick `Update`

List Students

Description of the functionality being tested:

Teacher should be able to list students in a course

Reference to RD:

This functionality was first described on page 9 of the Requirments Document.

Inputs and preconditions:

The tester need to be logged in as a teacher who is leading atleast one course. That course need to have some students taking it.

Expected output:

The teacher should get a list of students taking the course he selected.

Procedure:

1. Select My Courses from the main menu.
2. Pick the course you want to list the students for.
3. Click the course.
4. Click List Students .

Assign assistants

Description of the functionality being tested:

Teachers should be able to assign assistants to a course.

Reference to RD:

This functionality was first described on page 9 of the Requirements Document.

Inputs and preconditions:

The tester need to be logged in as a teacher who is leading at least one course. There need to be atleast one assistant user in the system.

Expected output:

After the test, when you go to the course page you should see the assistant listed on the course you added him/her too.

Procedure:

1. Select `My Courses` from the main menu.
2. Pick the course you want to add the assistant too.
3. Click the `Update Course` -button.
4. Write the name of the assistant you want to add in the `Add Assistant` field.
5. Click the `Add` -button.

Remove assistants

Description of the functionality being tested:

Teachers should be able to remove an assistant from a course

Reference to RD:

This functionality was first described on page 9 of the Requirements Document.

Inputs and preconditions:

The tester need to be logged in as a teacher with at least one course that he is leading. That course need to have at least one assistant assigned to it.

Expected output:

After the test, the tester should not see the removed assistant listed on the course info page.

Procedure:

1. Select `My Courses` from the main menu.
2. Pick the course you want to remove the assistant from.
3. Click the `Update Course` -button.
4. Select the assistant you want to remove from the drop down list.
5. Press the `Remove` -button.

Update grades

Description of the functionality being tested

A teacher or an assistant should be able to update a student's grade on a specific assignment

Reference to the RD

This functionality was first described on page 9 of the Requirements Document.

Inputs and preconditions

The inputs to this functionality is an assignment that can be graded, and a student that handed in this assignment.

The preconditions are that you should be logged in to the system as a teacher, or assistant, in the relevant course.

Expected outputs

When you are done with this task, the student should have the new updated grade visible to him in the system.

Procedure

1. Click the main menu, choose 'Update grades'
2. Click the course to which the assignment belong
3. Choose the new grade in the given assignment
4. Click 'Update'

Join a course

Description of the functionality being tested

A student should be able to join a course.

Reference to the RD

This functionality was first described on page 9 of the Requirements Document.

Inputs and preconditions

You should be logged in to the system, and there should be a course available to join.

Expected outputs

When you are done with this task, the student should be listed as attending to the course he just joined.

Procedure

1. Click the main menu, choose 'Join a course'
2. Click the course you want to join

List courses

Description of the functionality being tested

A user of the system should be able to list the courses that is given at the university.

Reference to the RD

This functionality was first described on page 9 of the Requirements Document.

Inputs and preconditions

There should be atleast one course to be listed, and you should be logged in to the system.

Expected outputs

When you successfully issued this command, you should be given a list of all the courses at the university, and a textarea that acts like a filter to let you find the course you specified.

Procedure

1. Click the main menu, choose 'List courses'

Show grades

Description of the functionality being tested

A student should be able to show his grades in a course.

Reference to the RD

This functionality was first described on page 9 of the Requirements Document.

Inputs and preconditions

You should be logged in as a student attending to a course that has got assignments.

Expected outputs

The output should be a list of your assignments in the given course, and the grades you've got.

Procedure

1. Click the main menu, choose 'Show grades'
2. Click the course that you are interested in

Leave course

Description of the functionality being tested

A student should be able to leave a course that he is currently attending to.

Reference to the RD

This functionality was first described on page 9 of the Requirements Document.

Inputs and preconditions

You have to be logging in as the given student, and attending to atleast one course that you want to leave.

Expected outputs

When you choose to leave a course, you stop beeing listed as attending to this course. However, the assignments that you have handed in for this course should be left in the database for future reference.

Procedure

1. Click the main menu, choose 'Leave course'
2. Click the course that you want to leave
3. Confirm that you want to leave this course

Create course

Description of the functionality being tested

The system administrator should be able to create a new course.

Reference to the RD

This functionality was first described on page 9 of the Requirements Document.

Inputs and preconditions

You should be logged in as system administrator.

Expected outputs

When you are done with this task, there should be a new course listed in the system.

Procedure

1. Click the main menu, choose 'Create new course'
2. Fill the form with all data required
3. Click 'Create'

Request course

Description of the functionality being tested

A teacher should be able to request a new course to be created.

Reference to the RD

This functionality was first described on page 9 of the Requirements Document.

Inputs and preconditions

You are logged in as teacher in the system.

Expected outputs

When you are done with this task, a new course will be listed for the system administration to be verified.

Procedure

1. Click the main menu, choose 'Request course creation'
2. Fill the form with all data required
3. Click 'Request creation'

Validate course

Description of the functionality being tested

A student should be able to leave a course that he is currently attending to.

Reference to the RD

This functionality was first described on page 9 of the Requirements Document.

Inputs and preconditions

You have to be logging in as system administrator, and there should be atleast one course creation request.

Expected outputs

When you validate the course, it should become visible in the system.

Procedure

1. Click the main menu, choose 'Validate course'
2. Click the course that you want to validate
3. Click 'Validate course'