

Course Information Management System

Group 2

David Chang
Linda Chowdhury
Oscar Fitinghoff
Patrik Parberg
Tomas Hansson

Table of Contents

1 Introduction	8
1.1 About the document	8
1.2 Glossary	8
1.3 Summary	9
2 System Overview	9
2.1 General Description.....	9
2.2 Overall Architecture Description	10
2.3 Detailed Architecture	12
Database System	12
Entity-relationship model	12
T-matrix.....	13
Database Structure.....	13
Block Diagrams	14
3 Design Considerations.....	23
3.1 Assumptions and Dependencies	23
3.2 General constraints	24
4 Graphical User Interface	24
4.1 Design of the System.....	24
4.2 Overview of the User Interface	26
4.3 Details of the Graphical User Interface.....	27
Menus	27
Start Page Menu	27
Personal Page – Logged Out	27
Personal Page – Logged In.....	28
Course Website – Logged Out	29
Course Website – Logged In.....	30
Course Administration	32
System Administration.....	34
Start Page.....	34
Personal Page	35
Front Page (News).....	35
Deadlines	36
Results	37
Schedule	37
Course Website	38
Front Page (News).....	38
Assignments	39
Apply for Course	39
Apply Confirmation	40
Deadlines	40
Course Description.....	41
Files	41
Information Pages	42
Results	42
Schedule	43
Course Administration Pages	44
Scheduled Activities.....	44
Add Activity Preview.....	45
Add Activity Confirmation	46

Edit Activity	46
Edit Activity Preview	47
Activity Edited Confirmation.....	47
Confirm Removal of Activity	48
Activity Removed Confirmation.....	48
Imported Schedule Confirmation	48
Confirm Remove Schedule	49
Remove Schedule Confirmation	49
Assignments	50
Add Assignment Preview.....	51
Add Assignment Confirmation	51
Edit Assignment	52
Edit Assignment Preview	53
Edit Assignment Confirmation.....	53
Confirm Remove Assignment.....	54
Remove Assignment Confirmation.....	54
Course Assistants	55
Confirm Add Course Assistant	55
Add Course Assistant Confirmation	56
Confirm Remove Course Assistant.....	56
Remove Course Assistant Confirmation.....	56
Edit Course Description	57
Edit Course Description Confirmation.....	58
Create Course Website – Create Description.....	59
Create Course Website – Import Schedule	60
Create Course Website – Import Schedule Confirmation.....	60
Create Course Website –Add Information Pages.....	61
Create Course Website – Add Information Pages Preview.....	62
Create Course Website – Add Information Pages Confirmation	62
Create Course Website – Add Deadlines	63
Create Course Website – Add Deadlines Preview	63
Create Course Website – Add Deadlines Confirmation.....	64
Create Course Website Summary	65
Create Course Website Confirmation	65
Deadlines	66
Add Deadline Preview	67
Add Deadline Confirmation	67
Edit Deadlines	68
Edit Deadline Preview.....	68
Edit Deadline Confirmation	69
Confirm Deadline Removal	69
Deadline Removal Confirmation	69
Information Pages	70
Add Information Page Preview	71
Add Information Page Confirmation.....	71
Edit Information Page	72
Edit Existing Information Page Preview	73
Information Page Edited Confirmation	73
Confirm Removal of Information Page.....	74
Information Page Removed Confirmation	74

News.....	75
Add News Preview.....	76
Add News Confirmation.....	76
Edit News.....	77
Edit Existing News Preview.....	77
News Edited Confirmation.....	78
Confirm Removal of News.....	78
News Removed Confirmation.....	78
Registrations.....	79
Verified Registrations Confirmation.....	79
Confirm Removal of Student Registration.....	80
Student Registration Removed Confirmation.....	80
Register Results.....	81
Results Registered Confirmation.....	81
Files.....	82
Add File Confirmation.....	82
Edit File.....	83
Edit File Confirmation.....	83
Confirm Remove File.....	84
Remove File Confirmation.....	84
System Administration Pages.....	85
Courses.....	85
Course Added Confirmation.....	85
Existing Courses List.....	86
Edit Existing Course.....	86
Course Edited Confirmation.....	86
Users.....	87
Add User Confirmation.....	87
Existing Users.....	88
Edit Password.....	88
Edit Password Confirmation.....	89
Edit User Privileges.....	89
Edit User Privileges – Course Privileges – Find Course.....	90
Edit User Privileges – Course Privileges – Select Course.....	90
Confirm Edit User Privileges.....	91
Edit User Privileges Confirmation.....	91
Confirm User Removal.....	91
User Removal Confirmation.....	92
5. Design Details.....	92
5.1 Class Responsibility Collaborator (CRC) Cards.....	92
Activity.....	92
ActivityController.....	93
Assignment.....	93
AssignmentController.....	93
BaseObject.....	93
BaseController.....	93
Cache.....	94
Course.....	94
CourseController.....	94
Deadline.....	94

DeadlineController	95
File.....	95
FileController	95
Information Page	95
InformationPageController.....	96
News.....	96
NewsController	96
Result.....	96
ResultController	96
Session.....	97
Schedule	97
ScheduleController.....	97
User	97
UserController.....	98
5.2 Class Diagram	98
5.3 State Charts	99
Log In	100
Add Post	100
Edit Post	100
Delete Post.....	101
Create Course Website Guide	101
5.4 Interaction Diagrams	102
Log In	102
Create Database Post.....	102
Edit Database Post.....	103
Delete Database Post.....	103
View Post from Cache.....	104
View Post from Database.....	104
Export Schedule into iCalendar Format	104
Upload File.....	105
Edit Uploaded File	105
Delete Uploaded File.....	106
Create Course Website	107
5.5 Detailed Design	108
Database	108
Detailed database table definitions.....	108
Classes	112
Class Activity	112
Class ActivityController.....	116
Class Assignment	117
Class AssignmentController.....	120
Class Cache	122
Class Course.....	123
Class CourseController	129
Class Deadline.....	131
Class DeadlineController	135
Class File.....	137
Class FileController.....	140
Class InformationPage	141
Class InformationPageController.....	144

Class News	145
Class NewsController.....	148
Class Result	149
Class ResultController.....	152
Class Session	153
Class Schedule.....	154
Class ScheduleController	155
Class User.....	157
Class UserController	162
Implementation Index of Requirements	166
5.6 Package diagram	168
6. Functional Test Cases.....	168
Test Case TC1: Authenticate to the System.....	168
Test Case TC2: View Personal Page.....	168
Test Case TC3: View Overview of Course News.....	169
Test Case TC4: Create Course Website	169
Test Case TC5: Edit Existing Course Description	170
Test Case TC6: View Course Description	170
Test Case TC7: Add Course News.....	170
Test Case TC8: Edit Existing Course News.....	171
Test Case TC9: Remove Existing Course News	171
Test Case TC10: View Course News	172
Test Case TC11: Add Information Page	172
Test Case TC12: Edit Existing Information Page	172
Test Case TC13: Remove Existing Information Page	173
Test Case TC14: View Information Page	173
Test Case TC15: Import Course Schedule	173
Test Case TC16: Remove Existing Course Schedule	174
Test Case TC17: Add Scheduled Activity	174
Test Case TC18: Edit Existing Scheduled Activity	175
Test Case TC19: Remove Existing Scheduled Activity	175
Test Case TC20: View Scheduled Activity from Course Schedule.....	175
Test Case TC21: View Scheduled Activity from Compiled Schedule	176
Test Case TC22: Export Schedule in iCalendar Format	176
Test Case TC23: Add Deadline.....	176
Test Case TC24: Edit Existing Deadline.....	177
Test Case TC25: Remove Existing Deadline	177
Test Case TC26: View Deadlines	178
Test Case TC27: View Overview of Deadlines	178
Test Case TC28: Upload File	178
Test Case TC29: Edit Existing File.....	179
Test Case TC30: Remove Existing File	179
Test Case TC31: Download File	179
Test Case TC32: Add Course Assignment.....	180
Test Case TC33: Edit Existing Course Assignment.....	180
Test Case TC34: Remove Existing Course Assignment.....	180
Test Case TC35: View Course Assignments	181
Test Case TC36: Register Results	181
Test Case TC37: View Results.....	181
Test Case TC38: View Registered Students.....	182

Test Case TC39: Confirm Application to Get Registered for Course.....	182
Test Case TC40: Apply for Course	182
Test Case TC41: Unregister Registered Student.....	183
Test Case TC42: Add User Account	183
Test Case TC44: Remove User Account.....	184
Test Case TC45a: Edit User Privileges	184
Test Case TC45b: Edit User Privileges.....	185
Test Case TC46: Add Course Assistant	185
Test Case TC47: Remove Course Assistant.....	185
Test Case TC48: Add Course.....	186
Test Case TC49: Edit Existing Course.....	186

1 Introduction

1.1 About the document

The purpose of this document is to describe the design of the system. While the requirements document focused on behaviour, the design document is covering the technical aspects, such as implementation. Mainly, the documentation in the document is written to make the implementation stages as clear and straightforward as possible by designing the system on paper before the actual coding.

Since the intended audience of this document is the developers who are going to implement the system, the scope includes a system overview (architecture), design considerations, description of the graphical user interface, design details and functional test cases.

The system will be named Course Information Management System (abbreviated CIMS).

To fully understand parts of this document, the reader should have taken part of the requirements document for this system.

1.2 Glossary

Apache Tomcat

A web container that provides an environment for Java code to run in cooperation with a web server.

Client-server architecture

An architectural model for distributed systems where the system functionality is offered as a set of services provided by a server. These are accessed by client computers that make use of the services.

Cookie

Information stored in the user's web browser by the server-side of the system, used to maintain certain information regarding that specific web browser.

Database

A computer database is a structured collection of 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.

Entity-relationship model (ER model)

An Entity-relationship model is a relational schema database modeling method used to model a system and its requirements, where an entity represents a discrete object and a relationship captures how two or more entities are related to one another.

iCalendar

A standard for group calendaring and scheduling, which enables calendaring data to be sent via e-mail or the Web that is automatically entered into the recipient's schedule.

JavaServer Pages (JSP)

Enables software developers to create dynamic web pages that run on the web server. The web pages can for example load and process data from a database server, as well as receive user input from end-users.

MySQL

A widely used open source relational database management system.

1.3 Summary

The system uses a three-tier client-server architecture consisting of the client (web browser handling the graphical user interface towards the user), the application server (containing the business logic) and the database server (storing data for the business objects).

The system will be implemented in Java ServerPages (JSP), run on a web server set up with Apache Tomcat software and the data within the system will be stored using MySQL database software.

The overall design of the system (web page structure) is divided into the four categories Course Administration, Personal Page, System Administration and Course Website. These categories have web pages representing different tasks within the scope of respective category.

The user interface is divided into three sections. The header section at the top of the web page will contain the name of the system. Below the header section to the left, the navigation section is located. The last section will be placed to the right of the navigation section, and will display the main content for that particular web page.

In a web page there will be different controls in the form of text fields, dropboxes, buttons and links which allows the user to insert data for sending, send data to the server for saving or redirect to another web page.

To manage the business objects and their data, there will be classes to represent the business objects themselves and controller classes to manage them (add, edit or delete). The system also has classes handling sessions and caching.

2 System Overview

2.1 General Description

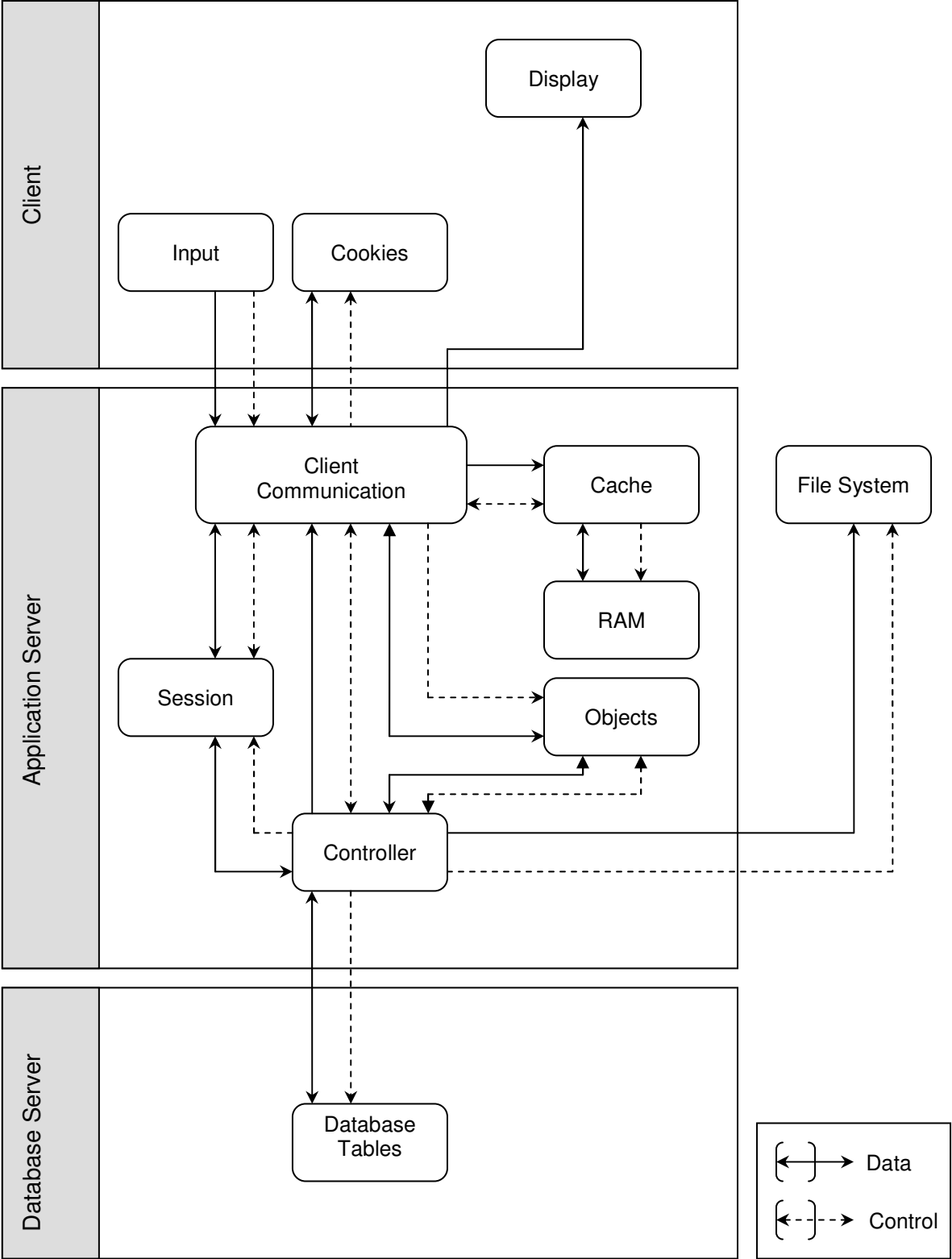
The system is meant to provide an easy way for lecturers without web design skills to create and update course websites, while at the same time providing a central information source for students who are enrolled in these courses.

Since we need to have the data stored on a central server we opted for a web based solution, allowing for users to access the system without the need to install a custom client application on the computer. This also allows the system to be accessed from public terminals, if the user isn't at home and doesn't possess a laptop of their own.

The design of the system was focused on making the user interface quick to use, requiring as few steps as possible to use what we expect will be the most commonly used functions, such as viewing schedules and news.

2.2 Overall Architecture Description

The system uses a three-tier client-server architecture, where the clients are web browsers and the servers consists of an application server and a database server.



The web browsers contain no business logic and manage the user interface. The web browser also stores cookies to enable the application server to identify the user. It also enables the user to send input to the application server.

The application server handles the incoming requests by retrieving and storing data from the database server. The application server uses caching to limit the number of database queries.

The application server consists of mainly two logical layers for processing requests. The first layer (client communication) interprets the user request and calls upon the appropriate controllers. There is also a cache layer used to lessen the load on the database server by storing frequently used information in memory.

Controllers are responsible for database communication and perform the creation/updating/removal of database records corresponding to actual business objects.

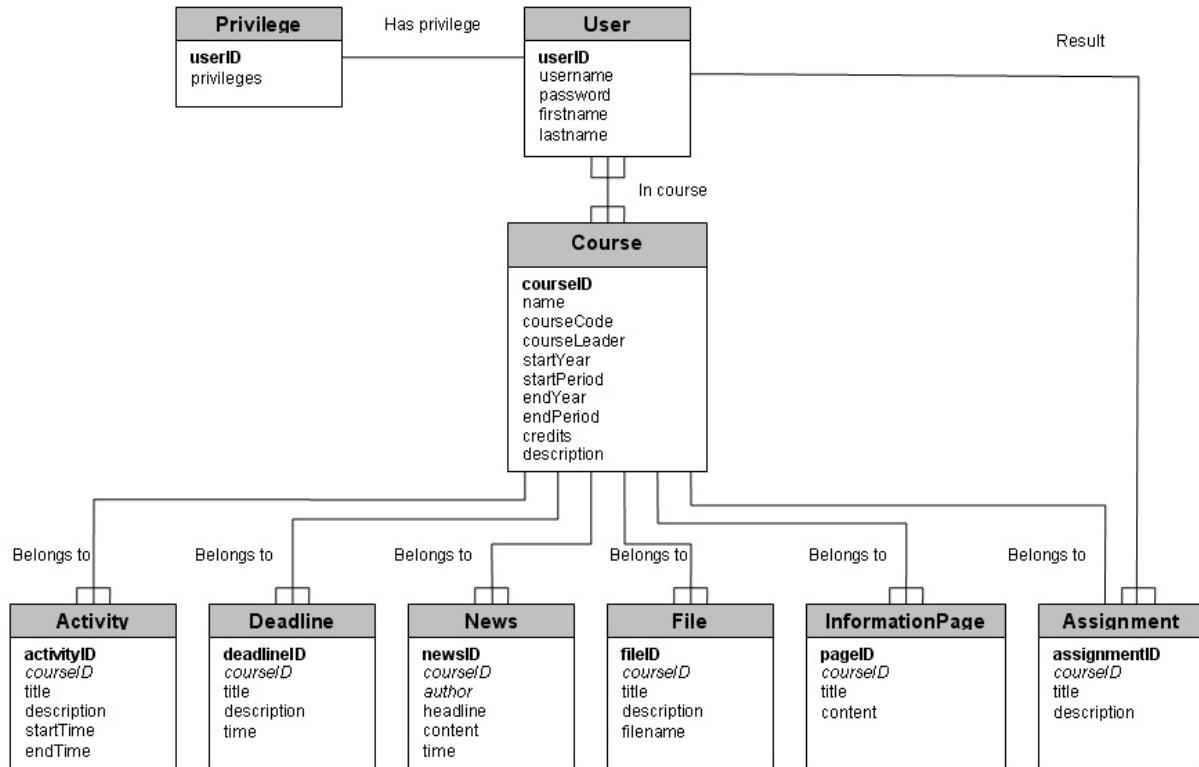
The session is a controller that is used to identify different users. It is used throughout the system and is therefore illustrated in the diagram.

Objects represent business objects and contain business logic and validation. These are mainly used to represent data in the database in a domain specific way, and do little more than allow for this data to be manipulated and making sure that the objects state obey the rules of the business object it represents.

2.3 Detailed Architecture

Database System

Entity-relationship model



The above is an entity-relationship model (ER model) of the database structure that will be used by the system. Primary keys are distinguished by bold font, and foreign keys are distinguished by italic font.

The central part of data in the system is the *Course* table where information about the courses is stored. Much of the rest of data in the system is related to the *Course* table. A course can for example have activities, deadlines, news, files, information pages and assignments. All these must be related to exactly one course.

Users can be in none, one or several courses, and a course can have none, one or several users in it. Users are in a course when they're applying to get registered for it, fully registered or are involved with teaching in the course. The user's status in a specific course is available in as a status field in the *InCourse* relation.

Privileges which are not course specific (system administrator privileges) are stored in the *Privilege* table.

Results are stored in the *Result* table and are related to one user and one assignment. Each assignment is in turn related to one course.

Files are stored in the file system, and only their metadata is stored in the database. No filename is stored in the database since files will be renamed so that their file-ID can be used to locate the file.

A schedule consists of several activities belonging to a course. A specific user's schedule can be found by finding the courses the student is in and then fetching the activities for that course.

T-matrix

Type	Name	I-Term(s)	E-Term(s)
Object	User	userID	username, password, firstname, lastname
Object	Course	courseID	name, courseCode, courseLeader, startYear, startPeriod, endYear, endPeriod, credits, description
Object	Activity	activityID	title, description, startTime, endTime
Object	Deadline	deadlineID	title, description, time
Object	News	newsID	author, headline, content, time
Object	File	fileID	title, description, filename
Object	InformationPage	pageID	title, content
Object	Assignment	assignmentID	title, description
Object	Privilege	userID	Privileges
1:N	ActivityBelongsTo	courseID, activityID	
1:N	DeadlineBelongsTo	courseID, deadlineID	
1:N	NewsBelongsTo	courseID, newsID	
1:N	FileBelongsTo	courseID, fileID	
1:N	nformationPageBelongsTo	courseID, pageID	
1:N	AssignmentBelongsTo	courseID, assignmentID	
N:N	InCourse	userID, courseID	Status
N:N	Result	userID, assignmentID	Grade

Above is the T-matrix for the database illustrated in the ER model.

Database Structure

Table	Attribute
-------	-----------

User	((userID), username, password, firstname, lastname)
Course	((courseID), name, courseCode, courseLeader, startYear, startPeriod, endYear, endPeriod, credits, description)
Activity	((activityID), courseID, title, description, startTime, endTime)
Deadline	((deadlineID), courseID, title, description, time)
News	((newsID), courseID, author, headline, content, time)
File	((fileID), courseID, title, description, filename)
InformationPage	((pageID), courseID, title, content)
Assignment	((assignmentID), courseID, title, description)
Privilege	((userID), privilege)
InCourse	((userID, courseID), status)
Result	((userID, assignmentID), grade)

This is the database structure after normalizing it from the T-matrix. Privileges which are not course specific are stored in a separate table to avoid wasting space since very few users (system administrators) will have any special privileges.

Block Diagrams

A block diagram is a type of flowchart, which quickly gives you an overview of the major process steps in the system. The processes to add course description, add news, add assignment etc. are very similar and therefore they are represented by the block diagram "Create Database Post". The exception is to upload a file because of the interaction with the file system. The same goes for "Edit Database Post" and "Delete Database Post".

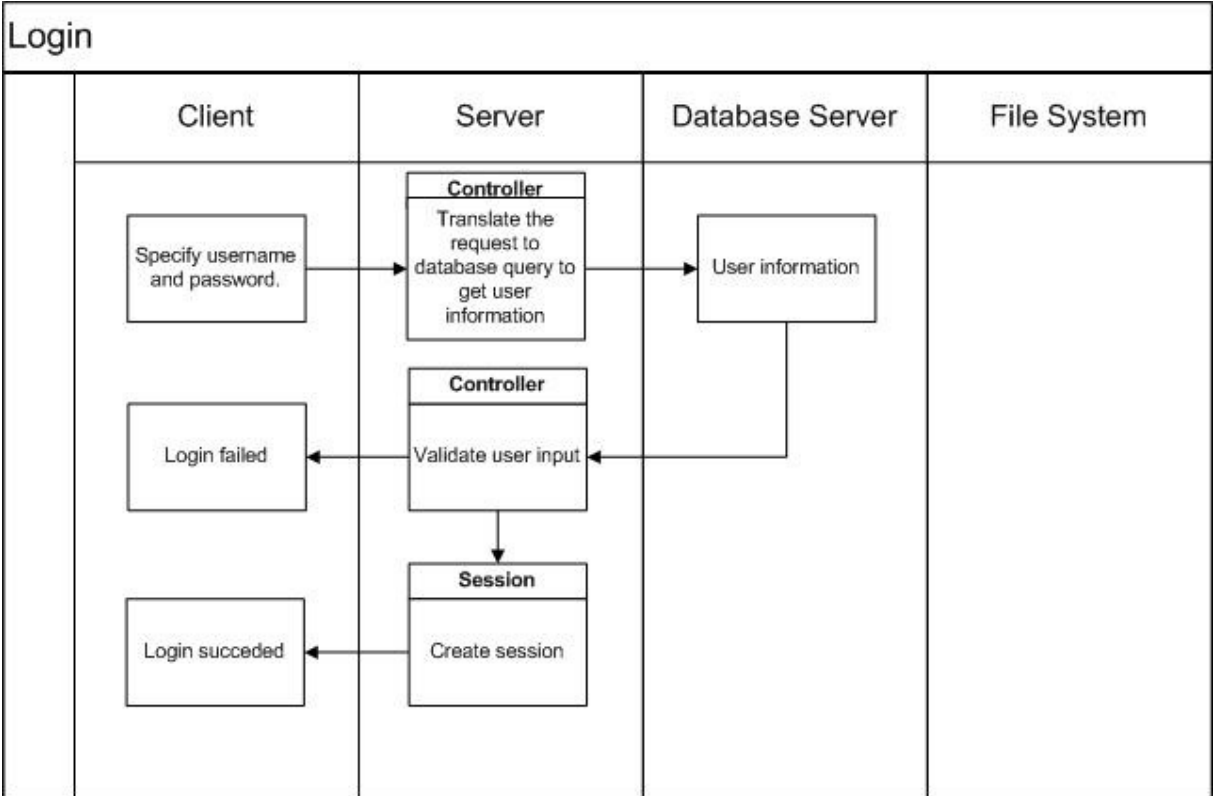


Figure 1 displays the major processes when a client requests to log in.

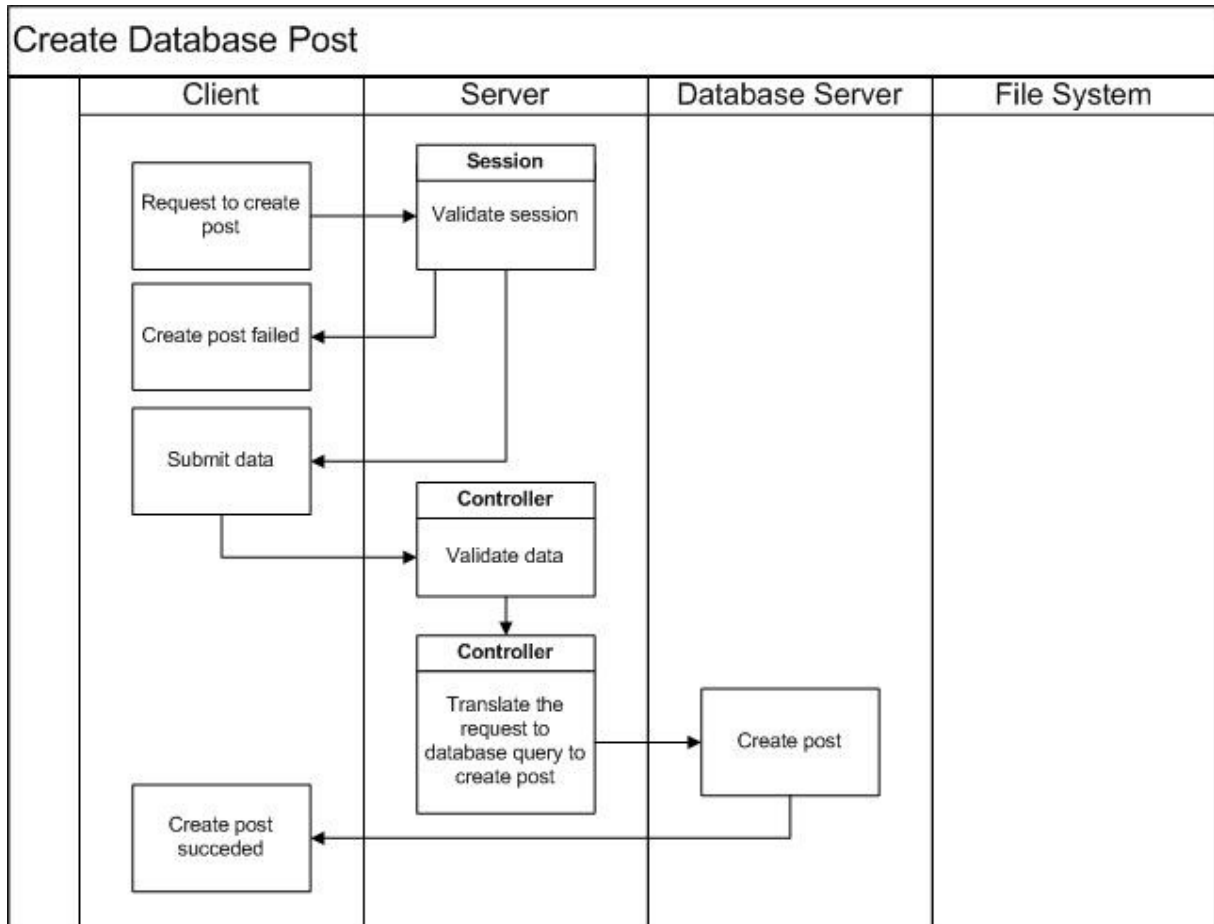


Figure 2 displays the processes to create a database post.

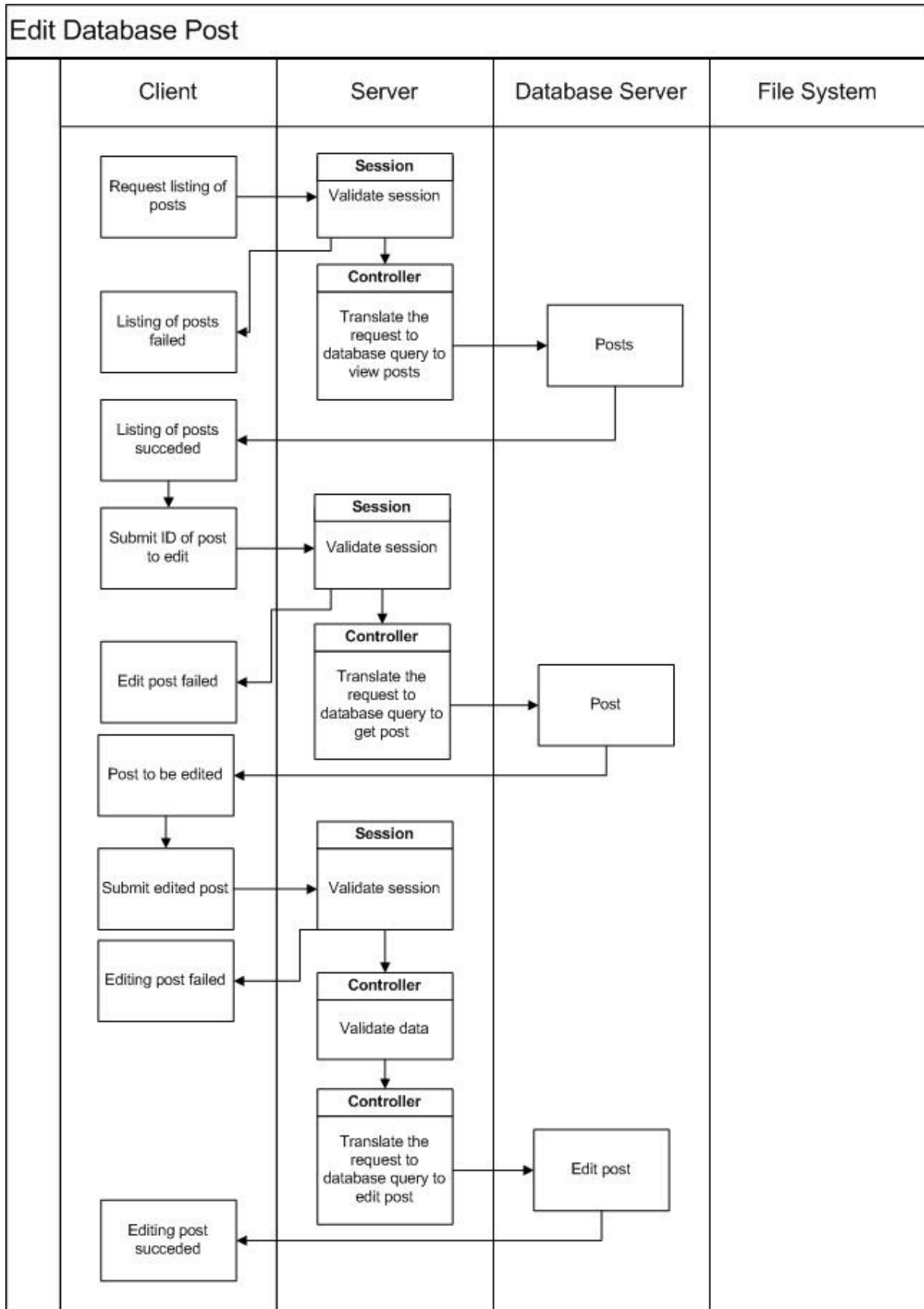


Figure 3 displays the processes to edit a database post.

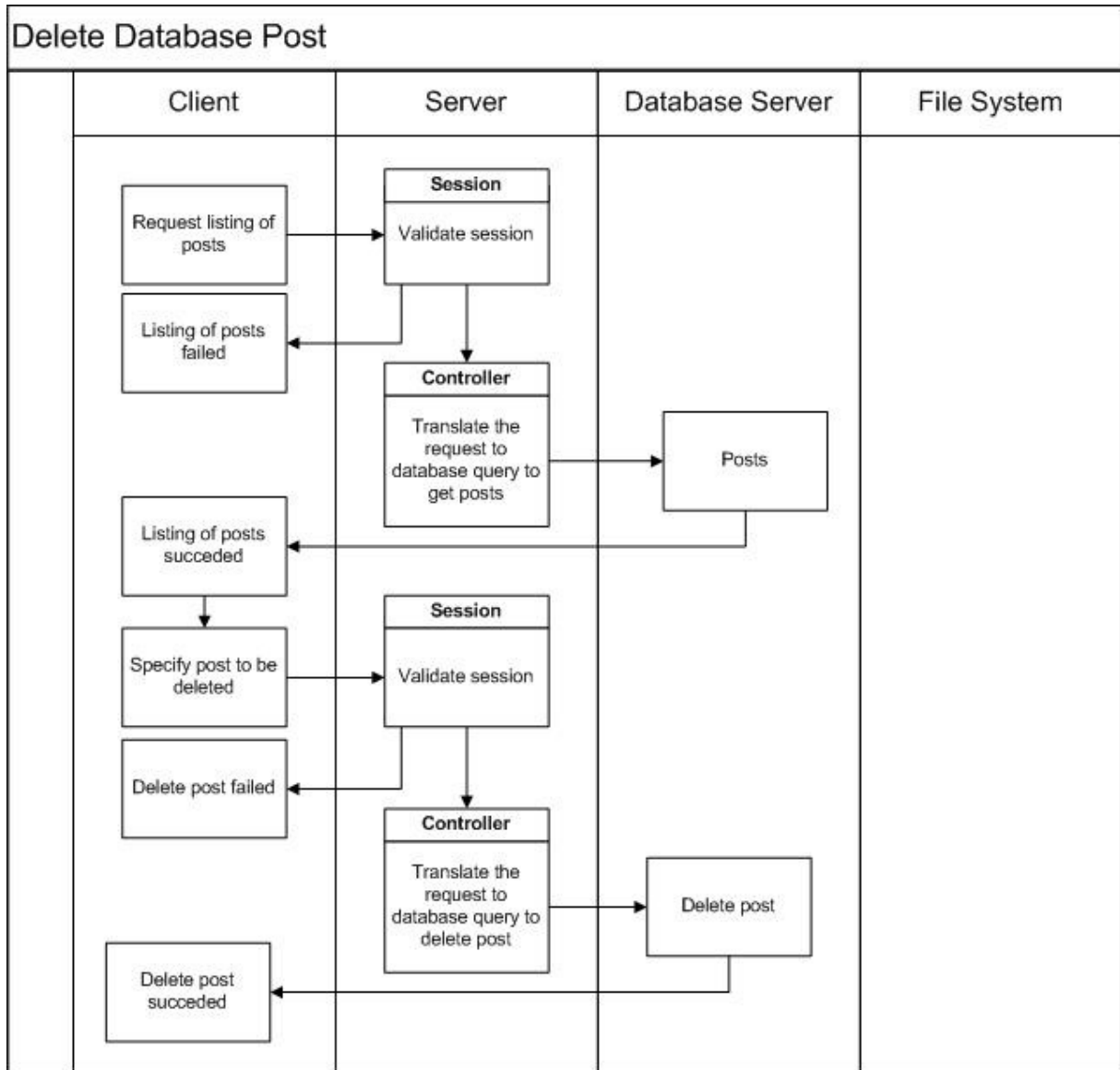


Figure 4 displays the processes to create a database post.

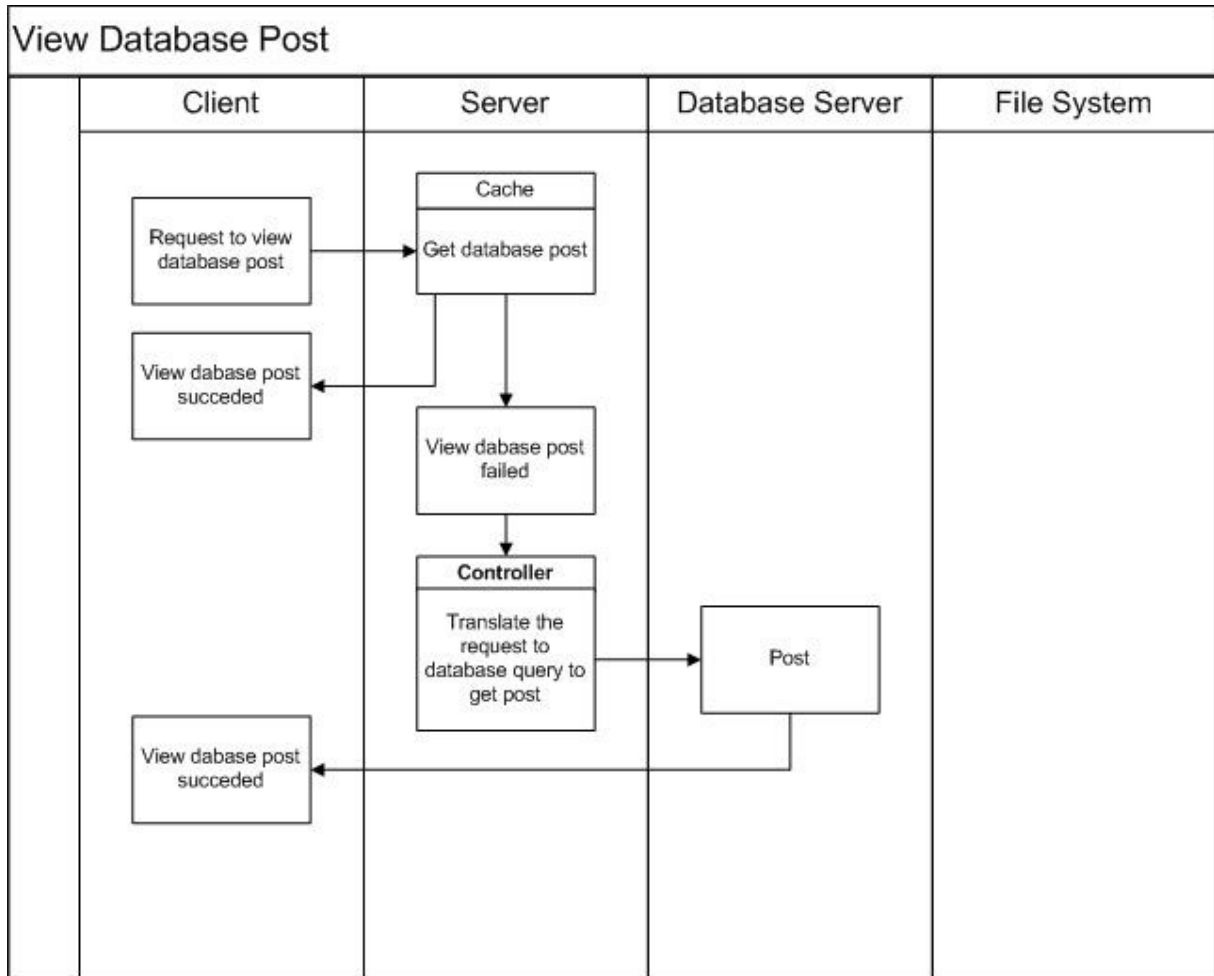


Figure 5 displays the processes when a client requests to view a database post.

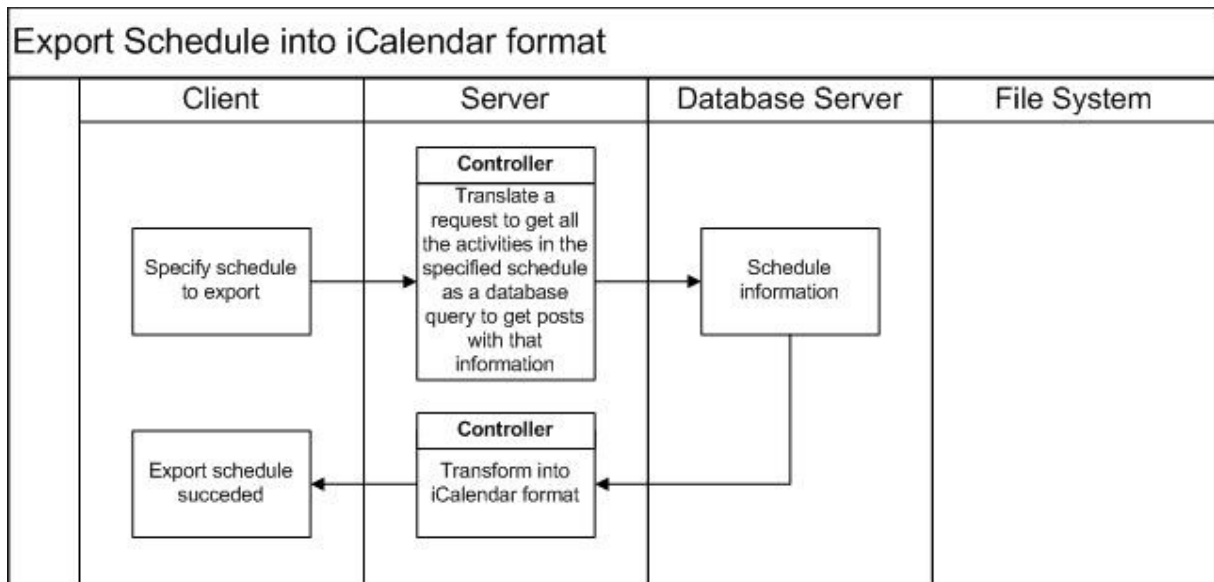


Figure 6 displays the processes to export a schedule into iCalendar format.

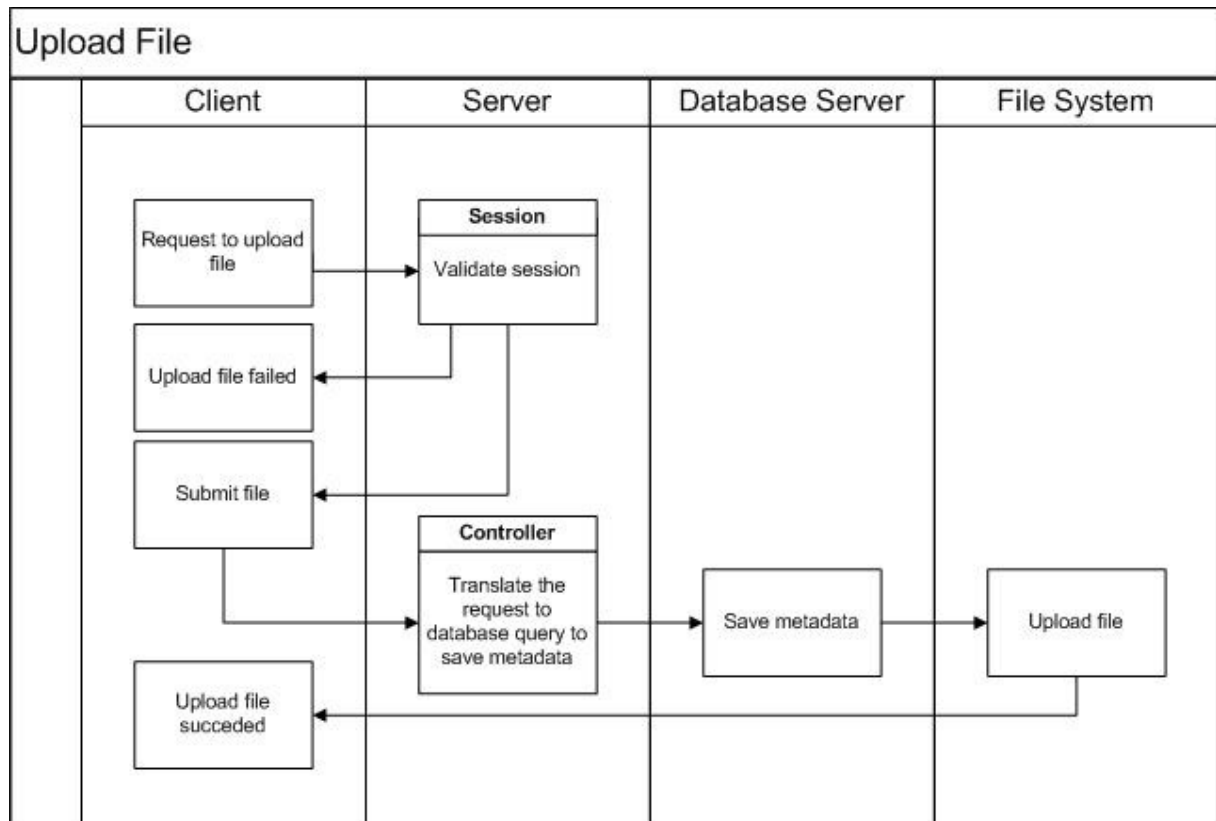


Figure 7 displays the processes when a client requests to upload a file.

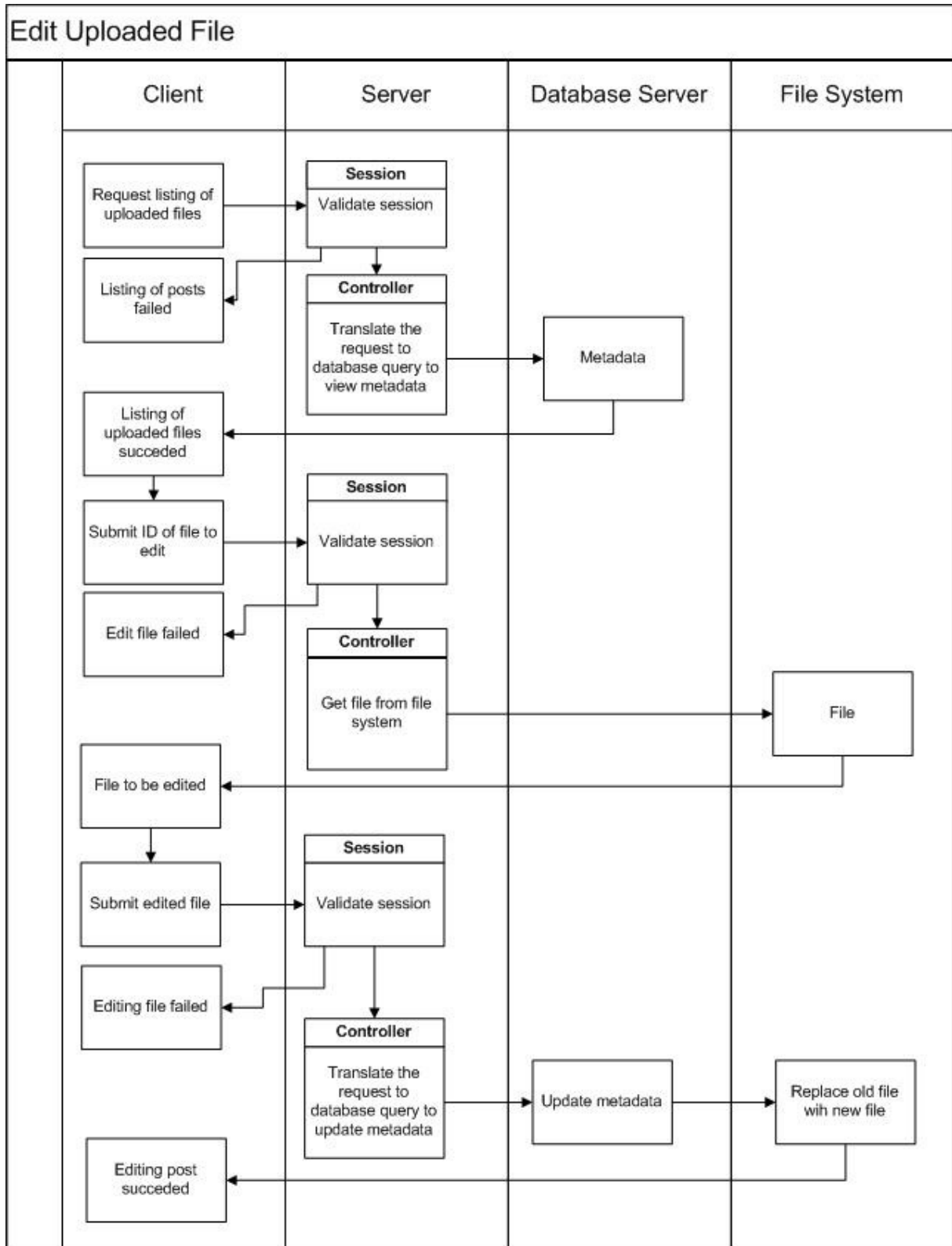


Figure 8 displays the processes when a client requests to edit an uploaded file.

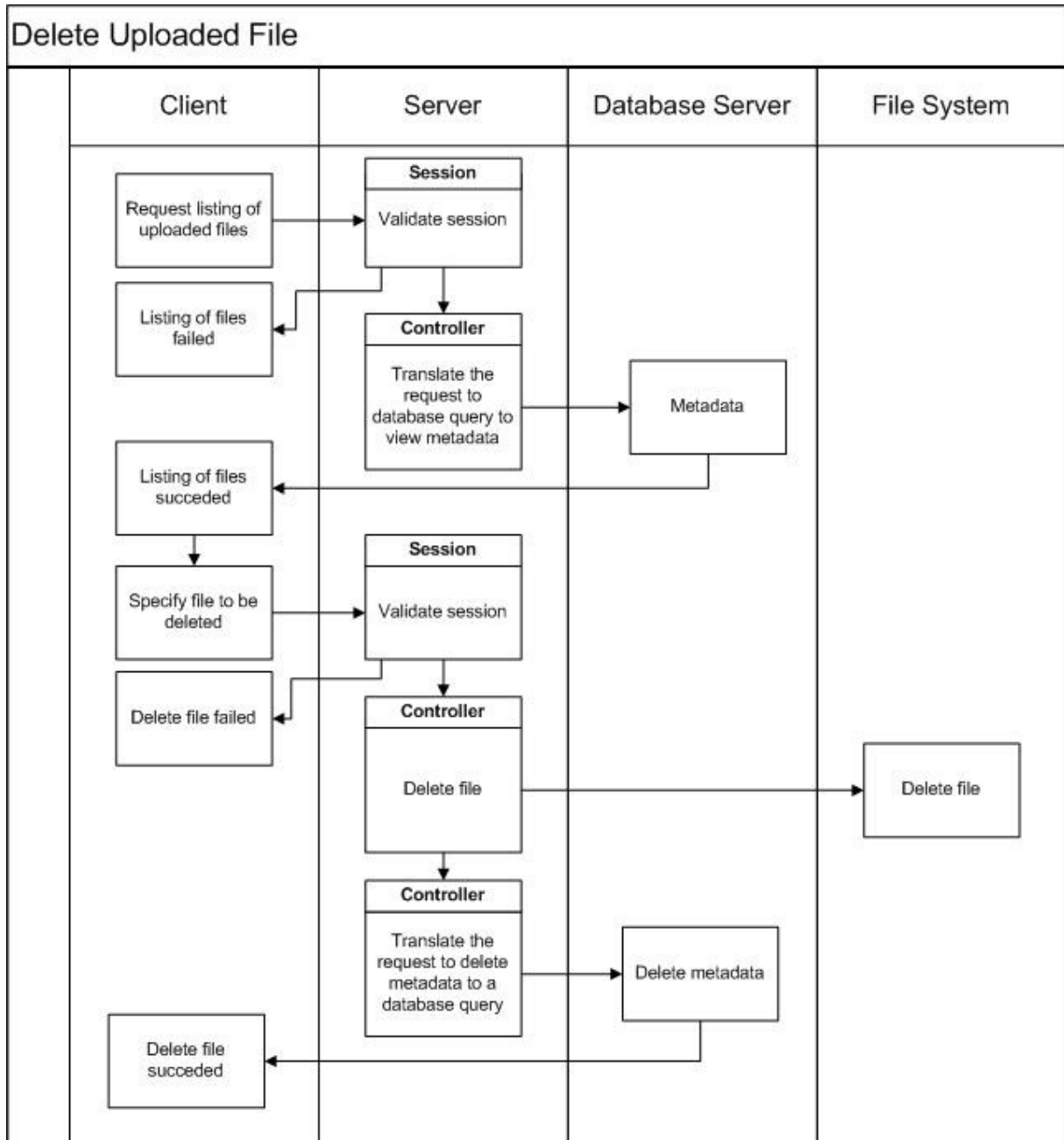


Figure 9 displays the processes when a client requests to delete an uploaded file

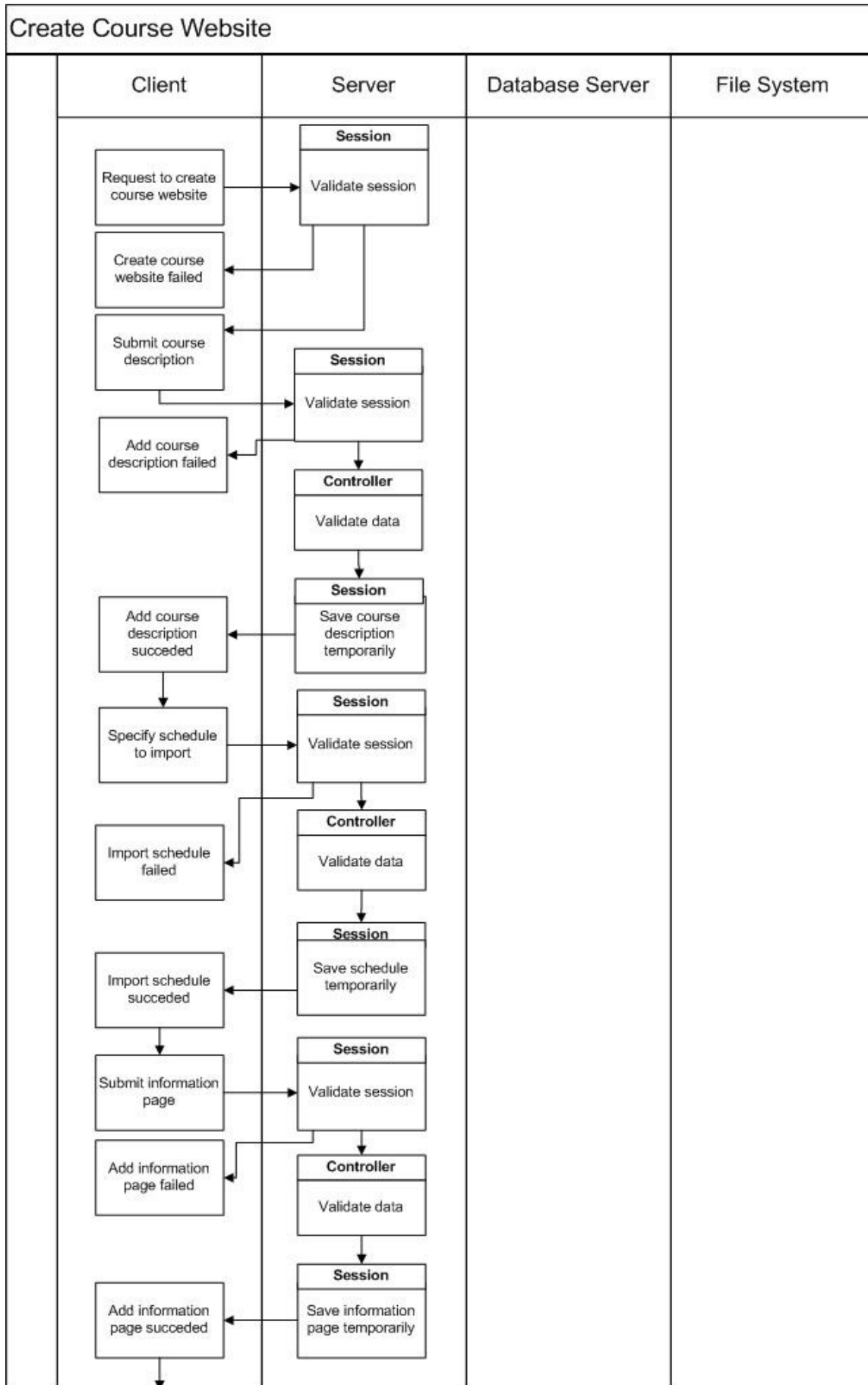


Figure 10 displays the processes when a client requests to create website (part 1).

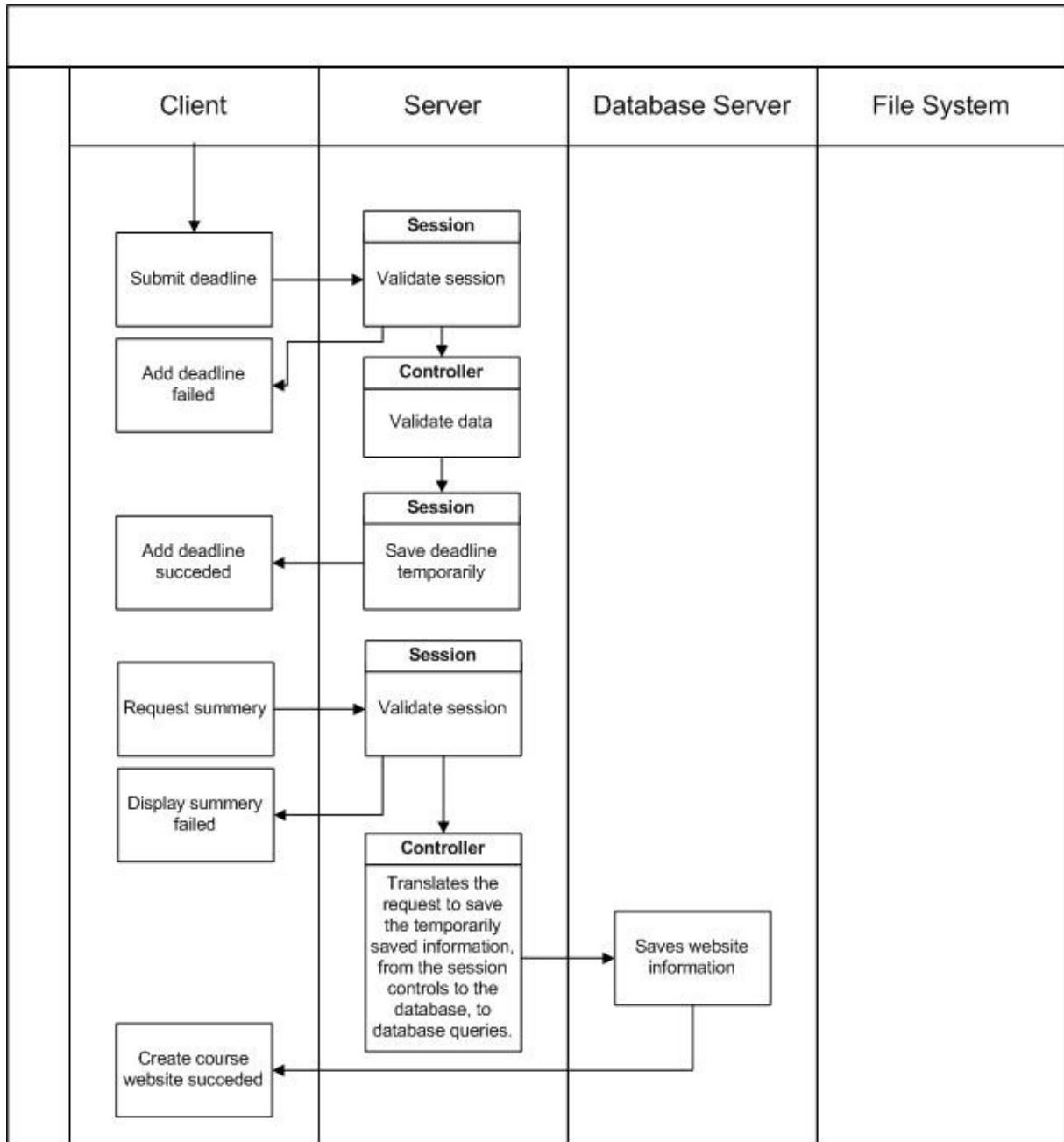


Figure 11 displays the processes when a client requests to create website (part 2).

3 Design Considerations

3.1 Assumptions and Dependencies

The software will be implemented using Java ServerPages (JSP) without the use of Enterprise Java Beans (EJB). In theory it should be possible to run on any operating system that supports the Apache Tomcat webserver, however it will only be tested to work with Apache Tomcat running on Linux. The version of Apache Tomcat that will be supported is version 6.0.

The system is designed to be used with the MySQL Relational Database Management System (RDBMS) version 5.0

Users of the system will be running the Mozilla Firefox web browser version 2.0 when using the system. They will not need any specific training in order to use the system.

3.2 General constraints

One of the more likely bottlenecks of the system will be the database server, since it's used to store most of the information in the system it will be queried frequently, which could cause a high load on the server. To mitigate this the system will cache information in memory and reduce the amount of queries needed to the database server

4 Graphical User Interface

4.1 Design of the System

The overall design of the system is described by the figure below, where there are four major categories; Course Administration, Personal Page, System Administration and Course Website, which are represented by the rounded shadowed rectangles. The webpages in each category is represented by rounded rectangles and an element of a webpage is represented by a rectangle, e.g. "Export into iCalendar format" is an element of the Schedule webpage.

When accessing the Personal Page and the Course Website the default page is the News webpage. The default page when accessing the Registration webpage, in the category Course Administration, is View Registered Students page.

The design of the system has been centered on the so called Personal Page. The personal page will provide logged in users with access to the features their user privileges allow, such as viewing ones results if they are a student, or administrating of course websites if they are a course leader.

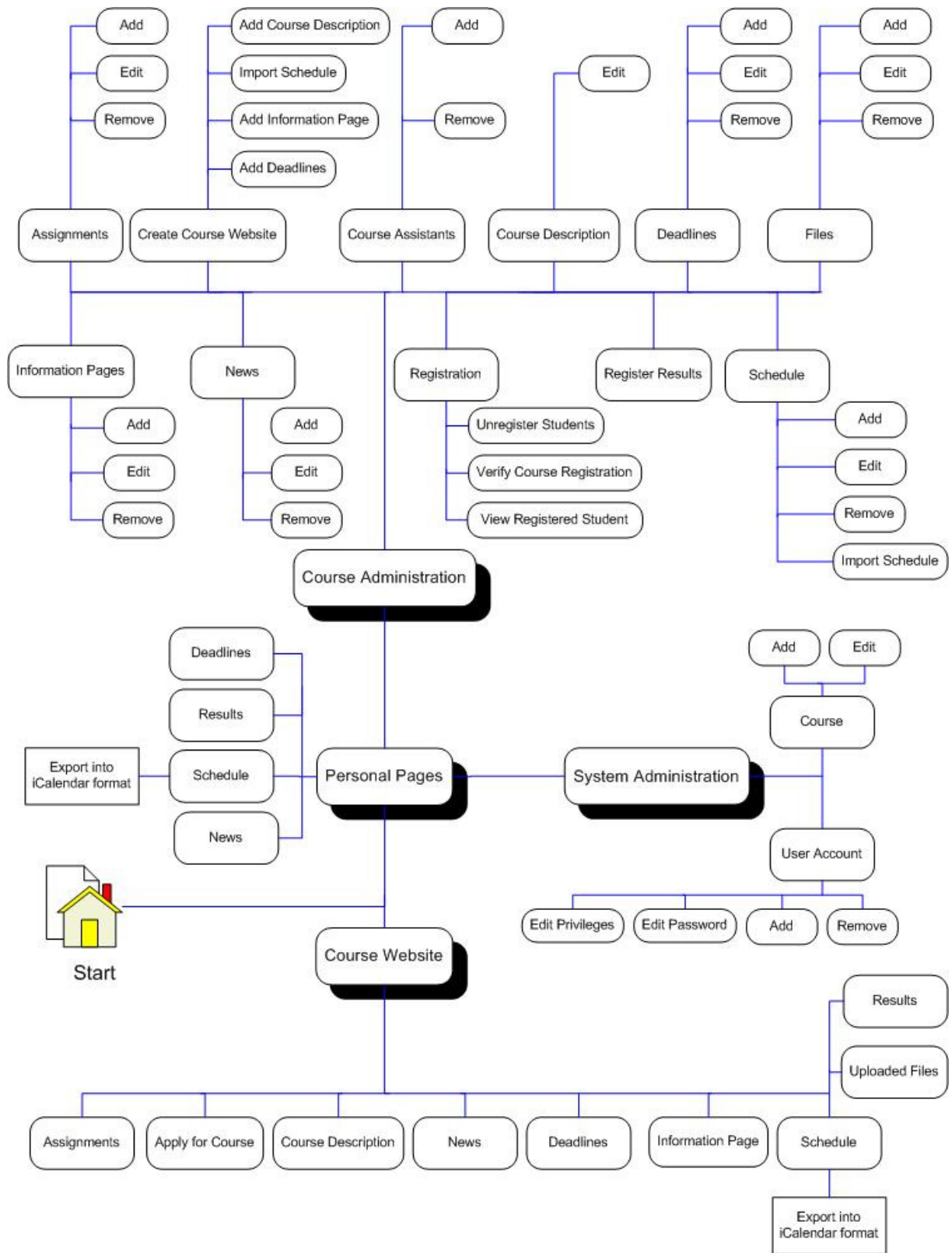
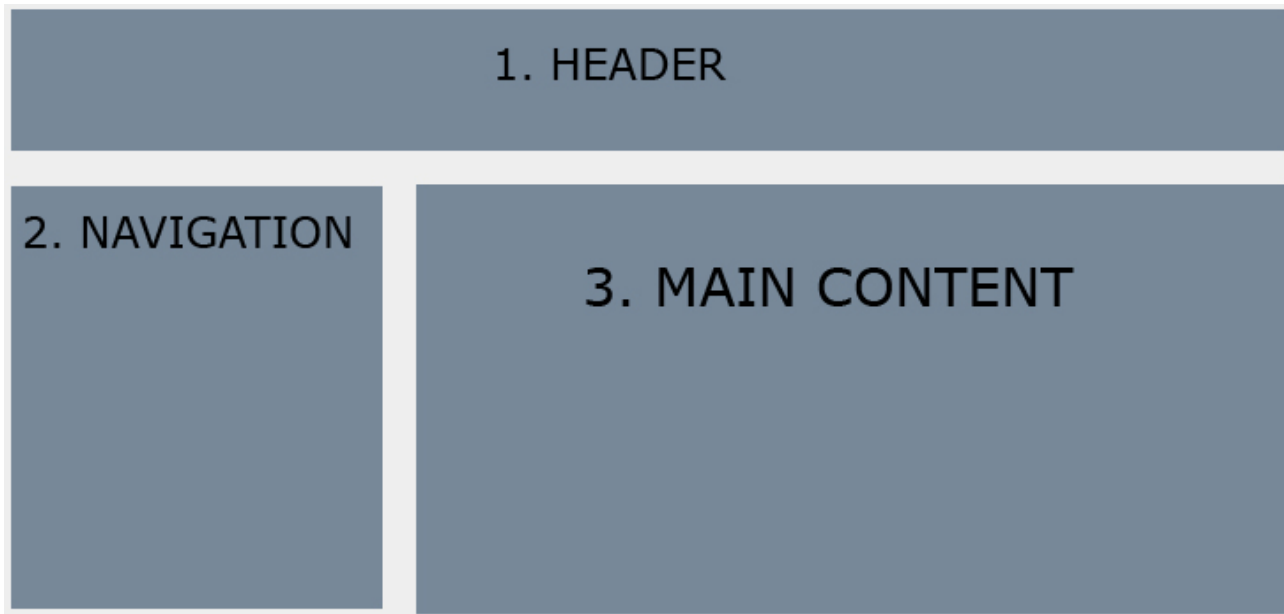


Figure 5 describes the overall design of the system.

4.2 Overview of the User Interface



The graphical user interface (GUI) is divided into three sections:

1. The header (at the top), with the name of the system.

2. To the left, the navigation menu (listing of links). The content of the navigation menu will depend on the status of the user (whether the user is logged in or not, and what privileges the user is assigned). The menu available to users when logged in will remain mostly the same as they navigate to different parts of the system, and they'll have quick access to both functions related to the current page they're at, and at the same time be able to navigate to other parts of the system. Also, login/logout-related objects will be placed at the top of the navigation menu.

3. To the right, the main content is displayed. The page displayed in this section will correspond to the part of the system being used by the user.

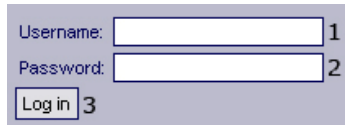
General notes:

Users who are not logged in will still be able to access their, and others, personal page. When not logged in the menu items available are more limited, such as that one can't view results unless logged in, and they will not "follow" the user as they navigate to different parts of the system.

4.3 Details of the Graphical User Interface

Menues

Start Page Menu



Username: 1
Password: 2
 3

Functional Requirements:

1.1

Controls:

1. txtUsername – textfield to get username from user.
2. txtPassword – textfield to get password from user.
3. btnLogin – Button to start the login procedure by invoking the authenticate method.

Methods:

authenticate – Validates the provided username/password combination against what is stored in the database.

Personal Page – Logged Out



Username:
Password:

Personal Links

- Course News 1
- Schedule 2
- Deadlines 3

Courses

- Software Engineering 4

Functional Requirements:

1.1
2.2
2.3
7.4
8.3

Controls:

1. InkCourseNews – Redirects the user to the news overview page.
2. InkSchedule – Redirects the user to the compiled schedule page.
3. InkDeadlines – Redirects the user to the deadlines overview page.
4. InkCourse1 – Redirects the user to the specific course website.

Methods:

None

Personal Page – Logged In

The screenshot shows a vertical list of buttons on a light purple background. The buttons are arranged in two sections. The first section is titled 'Personal Links' and contains five buttons: 'Log out' (1), 'Course News' (2), 'Schedule' (3), 'Deadlines' (4), and 'Results' (5). The second section is titled 'Courses' and contains four buttons: 'Software Engineering' (6), 'Digital Design' (7), 'Logic' (8), and 'Linear Algebra' (9). Each button is a dark purple rectangle with white text and a small white number in the bottom right corner.

Log out	1
Personal Links	
Course News	2
Schedule	3
Deadlines	4
Results	5
Courses	
Software Engineering	6
Digital Design	7
Logic	8
Linear Algebra	9

Functional Requirements:

2.3
7.4
8.3
11.2

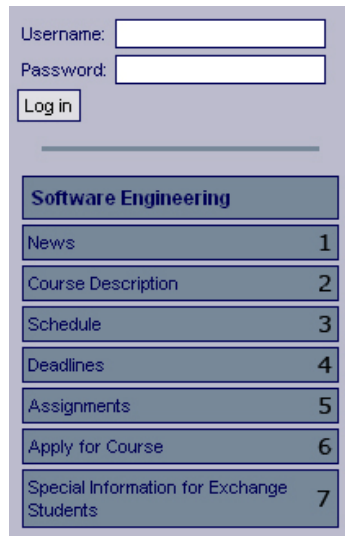
Controls:

1. InkLogOut – Invokes the logout method.
2. InkCourseNews – Redirects the user to the news overview page.
3. InkSchedule – Redirects the user to the compiled schedule page.
4. InkDeadlines – Redirects the user to the deadlines overview page.
5. InkResults – Redirects the user to the results page.
6. InkCourse1 – Redirects the user to the specific course website.
7. InkCourse2 – Redirects the user to the specific course website.
8. InkCourse3 – Redirects the user to the specific course website.
9. InkCourse4 – Redirects the user to the specific course website.

Methods:

Logout – Logs out the user.

Course Website – Logged Out



The screenshot shows a login form with two input fields: 'Username:' and 'Password:'. Below these is a 'Log in' button. A horizontal line separates the login section from a navigation menu. The menu is titled 'Software Engineering' and contains seven items, each with a number in a small box on the right:

- News 1
- Course Description 2
- Schedule 3
- Deadlines 4
- Assignments 5
- Apply for Course 6
- Special Information for Exchange Students 7

Functional Requirements:

- 1.1
- 4.2
- 5.2
- 6.2
- 7.3
- 8.2
- 10.2
- 12.3

Controls:

1. InkNews – Redirects the user to the news page for the specific course.
2. InkDescription – Redirects the user to the description page for the specific course.
3. InkSchedule – Redirects the user to the schedule page for the specific course.
4. InkDeadlines – Redirects the user to the deadline page for the specific course.
5. InkAssignments – Redirects the user to the assignment page for the specific course.
6. InkApply – Redirects the user to the apply page for the specific course.
7. InkInformationPage – Redirects the user to a information page for the specific course.

Methods:

None

Course Website – Logged In

Log out	1
<hr/>	
Software Engineering	
News	2
Course Description	3
Schedule	4
Deadlines	5
Assignments	6
Uploaded Files	7
Results	8
Special Information for Exchange Students	9
<hr/>	
Personal Links	
Course News	10
Schedule	11
Deadlines	12
Results	13
<hr/>	
Courses	
Software Engineering	14
Digital Design	15
Logic	16
Linear Algebra	17

Functional Requirements:

2.3
4.2
5.2
6.2
7.3
7.4
8.2
9.2
10.2
11.2
12.3

Controls:

1. InkLogOut – Invokes the logout method.
2. InkNews – Redirects the user to the news page for the specific course.
3. InkDescription – Redirects the user to the description page for the specific course.
4. InkSchedule – Redirects the user to the schedule page for the specific course.
5. InkDeadlines – Redirects the user to the deadline page for the specific course.
6. InkAssignments – Redirects the user to the assignment page for the specific course.

7. InkUploadedFiles – Redirects the user to the upload files page for the course.
8. InkApply – Redirects the user to the apply page for the specific course.
9. InkInformationPage – Redirects the user to a information page for the specific course.
10. InkAllNews – Redirects the user to the news overview page.
11. InkAllSchedule – Redirects the user to the compiled schedule page.
12. InkAllDeadlines – Redirects the user to the deadlines overview page.
13. InkAllResults – Redirects the user to the results page.
14. InkCourse1 – Redirects the user to the specific course website.
15. InkCourse2 – Redirects the user to the specific course website.
16. InkCourse3 – Redirects the user to the specific course website.
17. InkCourse4 – Redirects the user to the specific course website.

Methods:

logout – Logs out the user.

Course Administration

Log out	1
<hr/>	
Software Engineering	
News	2
Course Description	3
Schedule	4
Deadlines	5
Assignments	6
Uploaded Files	7
Apply for Course	8
Special Information for Exchange Students	9
Course Leader Links	
Assignments	10
Course Assistants	11
Course Description	12
Deadlines	13
Information Pages	14
Files	15
News	16
Registration	17
Results	18
Schedule	19
<hr/>	
Personal Links	
Course News	20
Schedule	21
Deadlines	22
<hr/>	
Courses	
Software Engineering	23

Functional Requirements:

- 2.3
- 4.1
- 4.2
- 5.1
- 5.2
- 6.1
- 6.2
- 7.2
- 7.3
- 7.4
- 8.1

8.2
8.3
9.1
9.2
10.1
10.2
11.1
12.1
12.2
12.3
12.4
13.2

Controls:

1. InkLogOut – Invokes the logout method.
2. InkNews – Redirects the user to the news page for the specific course.
3. InkDescription – Redirects the user to the description page for the specific course.
4. InkSchedule – Redirects the user to the schedule page for the specific course.
5. InkDeadlines – Redirects the user to the deadline page for the specific course.
6. InkAssignments – Redirects the user to the assignment page for the specific course.
7. InkUploadedFiles – Redirects the user to the upload files page for the course.
8. InkApply – Redirects the user to the apply page for the specific course.
9. InkInformationPage – Redirects the user to a information page for the specific course.
10. InkManageAssignment – Redirects the user to the manage assignment page for the specific course.
11. InkManageCourseAssistant – Redirects the user to the manage course assistant page for the specific course.
12. InkManageDescription – Redirects the user to the manage description page for the specific course.
13. InkManageDeadline – Redirects the user to the manage deadlines page for the specific course.
14. InkManageInformationPage – Redirects the user to the manage information page for the specific course.
15. InkManageFile – Redirects the user to the manage file page for the specific course.
16. InkManageNews – Redirects the user to the manage news page for the specific course.
17. InkManageRegistration – Redirects the user to the manage registration page for the specific course.
18. InkManageResults – Redirects the user to the manage results page for the specific course.
19. InkManageSchedule – Redirects the user to the manage schedule page for the specific course.
20. InkAllNews – Redirects the user to the news overview page.
21. InkAllSchedule – Redirects the user to the compiled schedule page.
22. InkAllDeadlines – Redirects the user to the deadlines overview page.
23. InkCourse1 – Redirects the user to the specific course website.

Methods:

logout – Logs out the user.

System Administration



Functional Requirements:

13.1
13.3
13.4

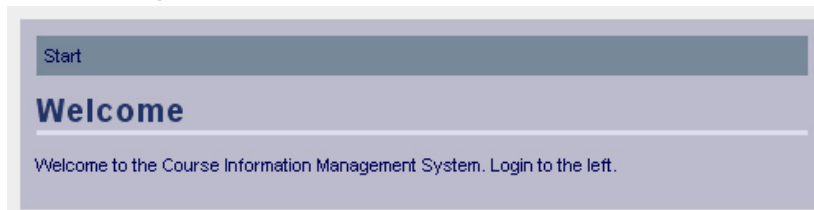
Controls:

1. InkLogout – Invokes the logout method.
2. InkAddCourse – Redirects the user to the add course page.
3. InkAddUser – Redirects the user to the add user page.

Methods:

logout – Logs out the user.

Start Page



Functional requirements:

1.1

Controls:

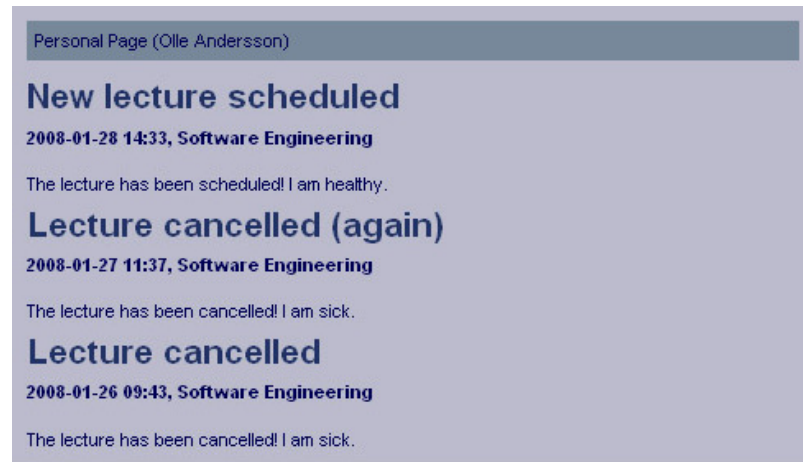
None

Methods:

None

Personal Page

Front Page (News)



The screenshot shows a personal page for 'Ole Andersson'. It contains three news items, each with a title, a timestamp, a course name, and a short message.

Personal Page (Ole Andersson)

New lecture scheduled
2008-01-28 14:33, Software Engineering
The lecture has been scheduled! I am healthy.

Lecture cancelled (again)
2008-01-27 11:37, Software Engineering
The lecture has been cancelled! I am sick.

Lecture cancelled
2008-01-26 09:43, Software Engineering
The lecture has been cancelled! I am sick.

Functional Requirements

- 2.1
- 2.2
- 2.3

Controls

None

Methods

printCompiledNews – Prints compiled news from all courses the student is enrolled in.

Deadlines

Personal Page (Olle Andersson) / Deadlines

POD Reviews Software Engineering	20th November, 2007
You will review the Project Overview Documents for 3 other groups. These reviews will not be anonymous – you must stand for what you say. You will be provided with a review form to be filled out for each POD that you review. You will then send a copy of each review to the respective project coordinators of the PODs you review and one copy of all three reviews to me. You will find details here.	
Homework 4 Logic	22nd November, 2007
Exercises in natural deduction (3.18, 3.21 and 3.23) shall be handed in.	
Use Cases Software Engineering	27th November, 2007
You will review the requirements and use cases for 2 other groups. As in the POD reviews, these reviews will not be anonymous. You will be provided with a review form to be filled out for each set of requirements and use cases that you review. You will then send a copy of each review to the respective project coordinators and one copy of all three reviews to me. You will find details here.	
Essay 4 Human Computer Interaction	1st December, 2007
Deadline for the essay on the topic of heuristical reviews.	
Use Case Reviews Software Engineering	4th December, 2007
You will turn in a complete description of the functional and non-functional requirements for your project as well as a complete set of use cases.	

Functional Requirements

2.2
8.3

Controls

None

Methods

printOverviewDeadlines – Prints all relevant deadlines from all the courses the student is enrolled in.

Results

Personal Page (Olle Andersson) / Results			
Date	Course	Assignment	Grade
1st October, 2007	Human Computer Interaction	Essay 1	A
10th October, 2007	Human Computer Interaction	Essay 2	B
21st October, 2007	Digital Design	Laboration 2	P
28th October, 2007	Human Computer Interaction	Essay 3	C
1st November, 2007	Software Engineering	POD	A
20th November, 2007	Logic	Homework 1	P
3rd December, 2007	Software Engineering	Use Cases	A

Functional Requirements

2.2
11.2

Controls

None

Methods

printResults – Prints all results assigned to the course assignments from all the courses the student is enrolled in.

Schedule

Personal Page (Olle Andersson) / Schedule				
Monday (1 Dec)	Tuesday (2 Dec)	Wednesday (3 Dec)	Thursday (4 Dec)	Friday (5 Dec)
08.00-10.00 Digital Design Recitation, Kista			08.00-10.00 Software Engineering Lecture, E1	8.00-10.00 Digital Design Lecture, M1
	10.00-12.00 Software Engineering Lecture, D1	10.00-12.00 Software Engineering Lecture, D1		10.00-12.00 Logic Recitation, D33
13.00-15.00 Logic Lecture, E1	13.00-15.00 Logic Recitation, E1		13.00-15.00 Logic Lecture, Q1	13.00-15.00 Software Engineering Lecture, D1
15.00-17.00 Software Engineering Lecture, E1			15.00-17.00 Digital Design Recitation, Kista	

1 Export this schedule to iCalendar format

Functional Requirements

2.2
7.4

7.5

Controls

1. InkExportSchedule – Invokes the method exportCompiledSchedule.

Methods

printCompiledSchedule – Prints the compiled schedule where activities, from all courses the student is enrolled in, are included.

exportCompiledSchedule – Exports the compiled schedule into iCalendar format.

Course Website

Front Page (News)



Functional Requirements

5.2

Controls

None

Methods

printNews – Prints all news for the course.

Assignments

Software Engineering / Assignments

The following are the graded assignments that you need to complete to finish the course.

- POD
- Use Cases

POD

The purpose of the POD is to give an introductory overview of the project. After reading this fairly brief document, the reader should have an idea of: 1. Who are the users and what problem does the system solve for them? 2. The main uses of the system. 3. The context/environment in which the system is to be used. 4. The scope of the system. 5. The main factors that need to be taken in to account when designing and building the system. 6. Technologies and Risks It should be between 3 and 5 pages in length and contain 6 sections corresponding to the 6 points listed above. In Section 2, you must include at least 2 usage narratives in addition to a general description of the main uses of the system. A usage narrative is a situated example of the use of the system. It is a brief paragraph in which you invent a fictional but specific actor and briefly capture the mental state of that person – why he wants what he wants or what conditions drive him to act as he does. Capture how the world works in that particular case, from the start of the situation to the end. These narratives should give the reader a good general idea of what the system is about. Here is an example of a usage narrative for the famous Automated Teller Machine (Bankomat) – ATM – example:

Use Cases

You will turn in a complete description of the functional and non-functional requirements for your project as well as a complete set of use cases.

Functional Requirements

10.2

Controls

None

Methods

printCourseAssignments – Prints all assignments from the course in question.

Apply for Course

Software Engineering / Apply

Apply for Course

Are you sure you would like to apply for this course?

After you have been applied for the course the course will show up on your page.

Apply 1

Functional Requirements

12.3

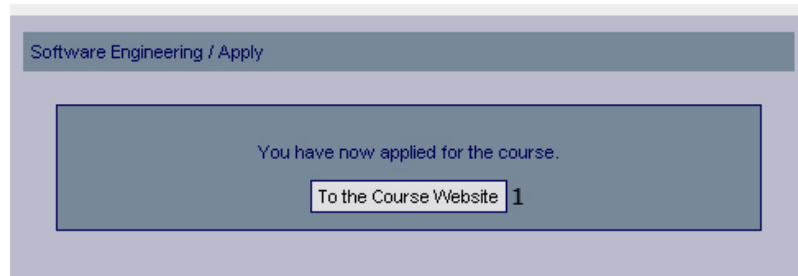
Controls

1. Apply – Button to confirm applying to a course.

Methods

applyForCourse – Updates the student's status for the course in question to reflect the student applying for the course.

Apply Confirmation



Functional Requirements

12.3

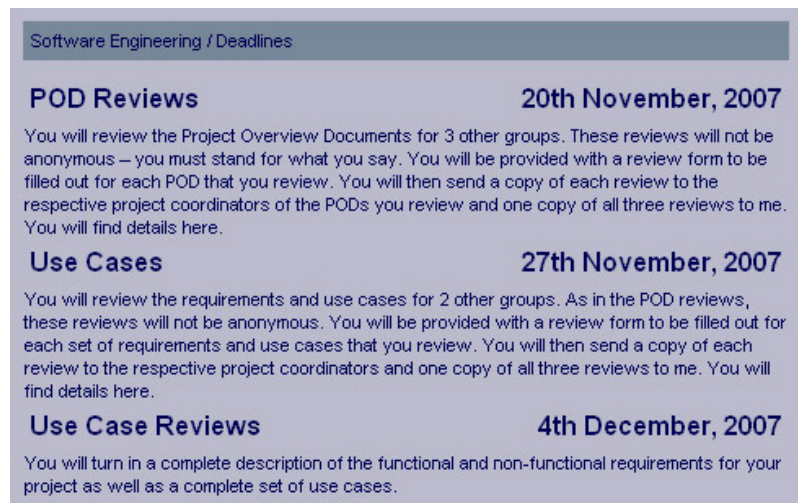
Controls

1. btnCourseWebsite – Redirects the user to the course website.

Methods

None

Deadlines



Functional Requirements

8.2

Controls

None

Methods

printDeadlines – Prints all the current deadlines for the course in question.

Course Description

Software Engineering / Course Description

Course Code: DD1363

Credits: 7,5

Begins in period: 2007, 1

Ends in period: 2008, 4

Second course in computer science giving theoretical knowledge and practical experience of working in a program development project.

Goals

After attending this course, the student is expected to be able to:

- describe a broad range of software engineering techniques, processes and methodologies that have been developed over the past 30 years,
- perform requirements analysis and formulation, system architecture and design, system implementation, and system testing,
- evaluate the applicability of a particular software engineering technique, process or methodology to a given project from both a technical and financial perspective,
- use a variety of tools (both commercial and academic) that can be used to design and implement software systems,
- evaluate whether a specific software engineering tool is technically and economically viable for a given project,
- find information in the main sources of information regarding software engineering technology,
- be effective in both oral and written technical communication,

Functional Requirements

4.2

Controls

None

Methods

printCourseDescription – Prints the description of the course

Files

Software Engineering / Uploaded Files

File	Date	Size
PowerPoint from Lecture 1	1 3rd November, 2007	501 kb
PowerPoint from Lecture 2	2 10th November, 2007	437 kb
PowerPoint from Lecture 3	3 17th November, 2007	358 kb
Extra Material on Use Cases	4 24th November, 2007	603 kb
PowerPoint from Lecture 4	5 1st December, 2007	920 kb
PowerPoint from Lecture 5	6 10th December, 2007	899 kb

Functional Requirements

9.2

Controls

1-6. InkFile1-InkFile6 – Invokes the viewFile method.

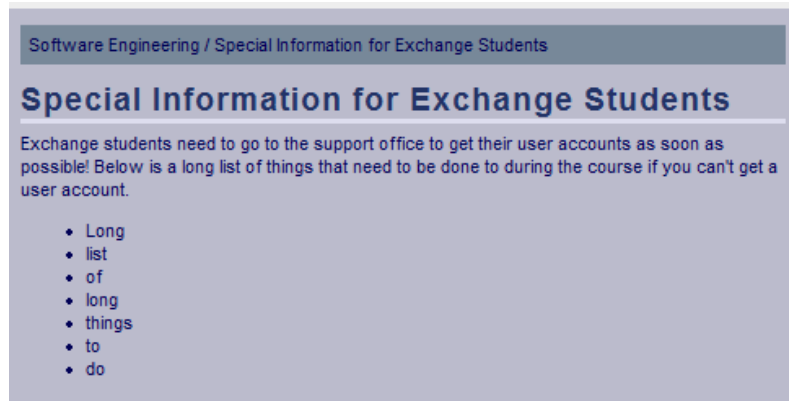
Methods

viewFile – Gives a view of the selected file.

printFile – Prints the selected file.

saveFile – Saves the selected file to a specified directory.

Information Pages



Software Engineering / Special Information for Exchange Students

Special Information for Exchange Students

Exchange students need to go to the support office to get their user accounts as soon as possible! Below is a long list of things that need to be done to during the course if you can't get a user account.

- Long
- list
- of
- long
- things
- to
- do

Functional Requirements

6.2

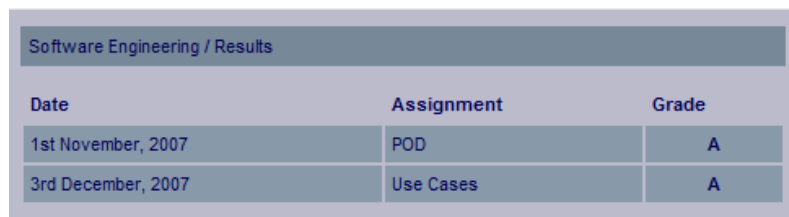
Controls

None

Methods

printInformationPage – prints the information for the course.

Results



Software Engineering / Results

Date	Assignment	Grade
1st November, 2007	POD	A
3rd December, 2007	Use Cases	A

Functional Requirements

11.2

Controls

None

Methods

printCourseResults – Prints all results assigned to the course assignments from the course in question.

Schedule

Software Engineering / Schedule				
Monday (1 Dec)	Tuesday (2 Dec)	Wednesday (3 Dec)	Thursday (4 Dec)	Friday (5 Dec)
			08.00-10.00 Lecture, E1	
	10.00-12.00 Lecture, D1	10.00-12.00 Lecture, D1		
				13.00-15.00 Lecture, D1
15.00-17.00 Lecture, E1				

1 Export this schedule to iCalendar format

Functional Requirements

7.3

7.5

Controls

1. InkExportSchedule- Invokes the printCourseSchedule method.

Methods

printCourseSchedule – Prints the schedule where activities, from the specific course the student selected.

printSchedule – Prints the schedule for the course.

Course Administration Pages

Scheduled Activities

Course Administrator / Schedule

Add Activity

Title:

Begin:

End:

Description:

13

Import Schedule

iCalendar file: 15

16

Edit Existing Activity

Title	Date	Begin	End	Edit	Remove
Lecture, E1	1st December, 2007	15:00	17:00	Edit17	Remove 22
Lecture, D1	2nd December, 2007	10:00	12:00	Edit18	Remove 23
Lecture, D1	3rd December, 2007	10:00	12:00	Edit19	Remove 24
Lecture, E1	4th December, 2007	08:00	10:00	Edit20	Remove 25
Lecture, D1	5th December, 2007	13:00	15:00	Edit21	Remove 26

Remove Schedule

Removing the schedule removes all the activities associated with this course.

27

Functional Requirements

- 7.1
- 7.2

Controls

1. txtActivityTitle – The title of the activity.
2. drpBeginDay – The day the activity starts.
3. drpBeginMonth – The month the activity starts.
4. drpBeginYear – The year the activity starts.
5. drpBeginHour – The hour the activity starts.
6. drpBeginMinute – The minute the activity starts.
7. drpEndDay – The day the activity ends.
8. drpEndMonth – The month the activity ends.

9. drpEndYear – The year the activity ends.
10. drpEndHour – The hour the activity ends.
11. drpEndMinute – The minute the activity ends.
12. txtActivityDescription – The description of the activity.
13. btnPreviewActivity – Shows a preview of the activity with provided information.
14. txtiCalFileLoc – The location of the iCal file.
15. btnBrowseFiles – A graphic alternative to control 14 which lets the user navigate through local files.
16. btnImportSchedule – Invokes the method importSchedule.
- 17-21. lnkEditActivity1 - lnkEditActivity5 – Invokes the editActivity method.
- 22-26. lnkRemoveActivity1 - lnkRemoveActivity5 – Invokes the removeActivity method.

Methods

importSchedule – Uploads the provided iCal schedule to the file system.

editActivity – Retrieves the data for the specified activity and allows the user to edit that data.

removeActivity – Removes the specified activity.

Add Activity Preview

Course Administrator / Schedule / Add Activity

Title:	Lecture, D1
Begin:	15.00, 1st December, 2007
End:	17.00, 1st December, 2007
Description:	

Back Save

1 2

Functional Requirements

7.2

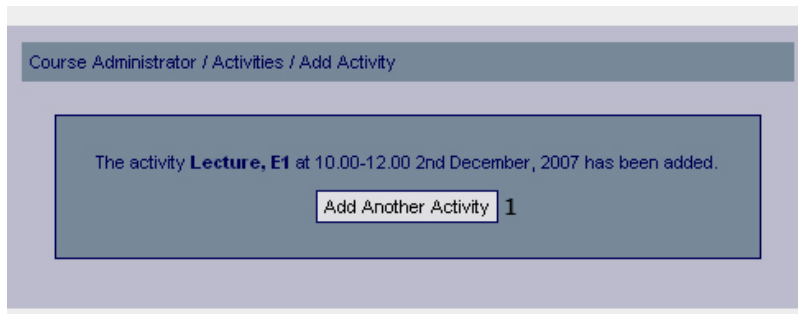
Controls

1. btnBack – Returns the user to the previous page.
2. btnAddActivity - Invokes the method addActivity.

Methods

addActivity – Adds the activity with the provided information to the database.

Add Activity Confirmation



Functional Requirements

7.2

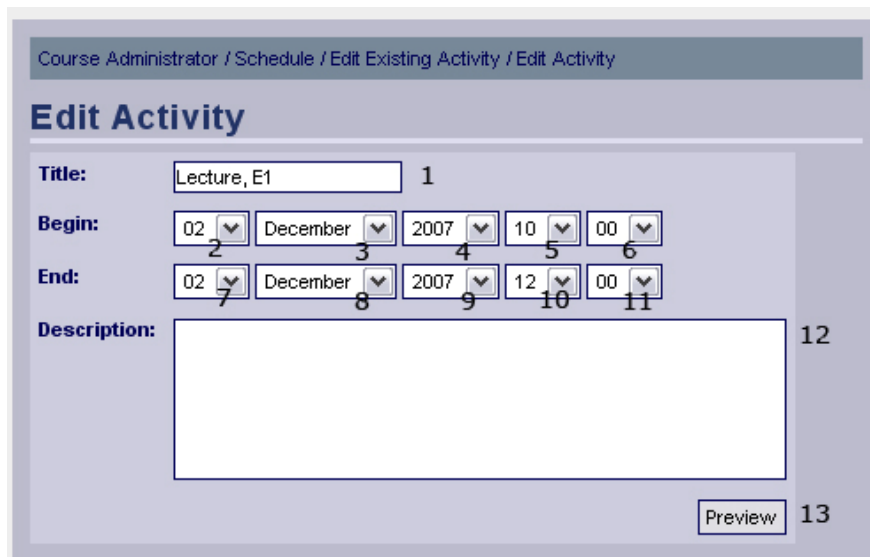
Controls

1. btnAddActivity – Redirects the user to the add activity page.

Methods

None

Edit Activity



Functional Requirements

7.2

Controls

1. txtActivityTitle – The title of the activity.
2. drpBeginDay – The day the activity starts.
3. drpBeginMonth – The month the activity starts.
4. drpBeginYear – The year the activity starts.
5. drpBeginHour – The hour the activity starts.
6. drpBeginMinute – The minute the activity starts.
7. drpEndDay – The day the activity ends.
8. drpEndMonth – The month the activity ends.
9. drpEndYear – The year the activity ends.

10. drpEndHour – The hour the activity ends.
11. drpEndMinute – The minute the activity ends.
12. txtActivityDescription – The description of the activity.
13. btnPreviewEditedActivity – Shows a preview of the edited activity.

Methods

None

Edit Activity Preview

Course Administrator / Schedule / Edit Existing Activity

Title:	Lecture, D1
Begin:	15.00, 1st December, 2007
End:	17.00, 1st December, 2007
Description:	

Back Save

1 2

Functional Requirements

7.2

Controls

1. btnBack – Returns the user to the previous page
2. btnEditActivity – Invokes the method editActivity.

Methods

editActivity – Updates the activity in the database, with the information provided.

Activity Edited Confirmation

Course Administrator / Schedule / Edit Existing Activity / Edit Activity

The activity **Lecture, E1** at 10.00-12.00 2nd December, 2007 has been changed.

Edit Another Activity 1

Functional Requirements

7.2

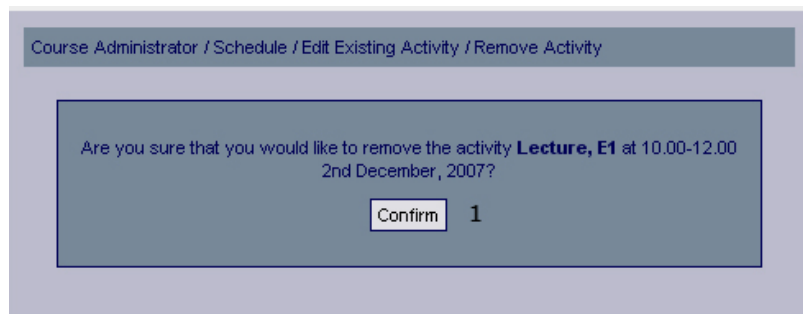
Controls

1. btnEditAnotherActivity – Redirects the user to the Scheduled activities page.

Methods

None

Confirm Removal of Activity



Functional Requirements

7.2

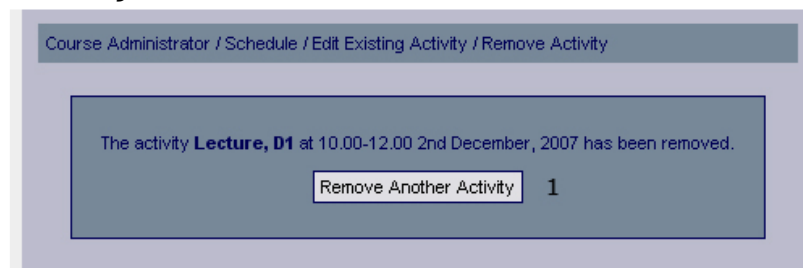
Controls

1. btnConfirmActivityRemoval - Invokes removeActivity.

Methods

removeActivity – Removes the scheduled activity from the database.

Activity Removed Confirmation



Functional Requirements

7.2

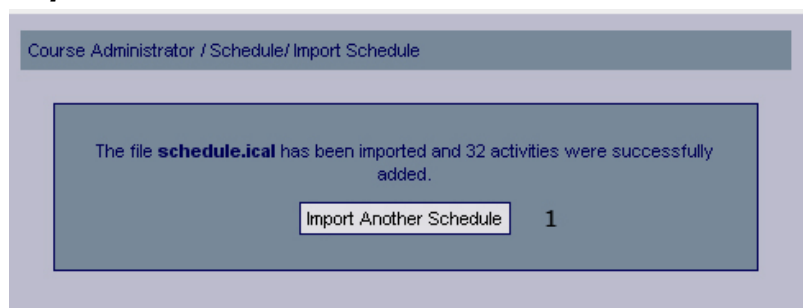
Controls

btnRemoveAnotherActivity – Redirects the user to the Scheduled activities page.

Methods

None

Imported Schedule Confirmation



Functional Requirements

7.1

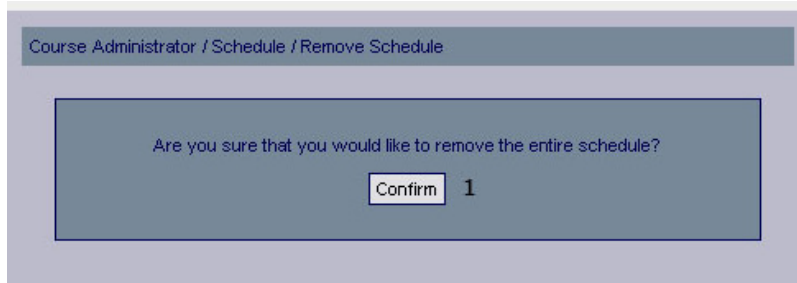
Controls

1. btnImportAnotherSchedule – Redirects the user to the Scheduled activities page.

Methods

None

Confirm Remove Schedule



Functional Requirements

7.1

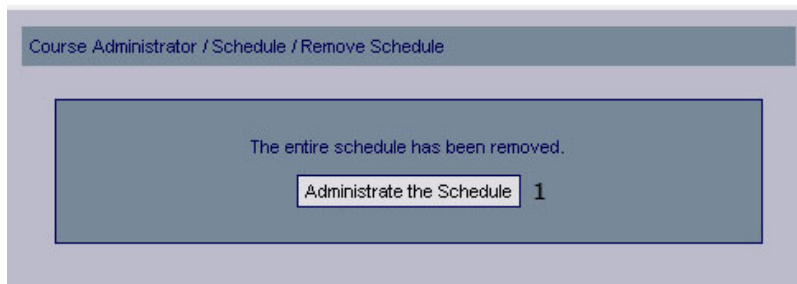
Controls

1. btnConfirmRemoveSchedule – Confirms that the schedule should be removed.

Methods

None

Remove Schedule Confirmation



Functional Requirements

7.1

Controls

1. btnAdministrateSchedule – Redirects the user to the schedule page.

Methods

None

Assignments

Course Administrator / Assignments

Add Assignment

Title: 1

Description: 2

3

Edit Existing Assignment

Title				
POD	Edit	4	Remove	6
Use Cases	Edit	5	Remove	7

Functional Requirements

10.1

Controls

1. txtAddAssignmentTitle – The title of the new assignment.
2. txtAddAssignmentDescription – A text that describing the new assignment.
3. btnPreviewAddAssignment – Invokes the displayAddAssignment method.
- 4-5. lnkEditAssignment1 - lnkEditAssignment2 – Invokes the editAssignment method.
- 6-7. lnkRemoveAssignment1 - lnkRemoveAssignment2 – Invokes the removeAssignment method.

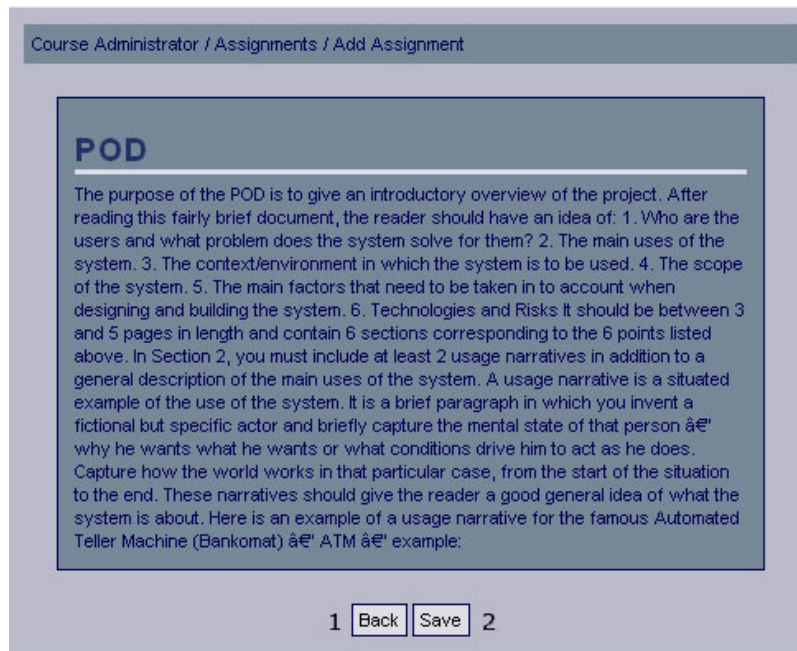
Methods

displayAddAssignment - Validates the input and displays a preview of the new assignment.

editAssignment – Retrieves the data for the specified assignment and allows the user to edit the data.

removeAssignment – Deletes the specified assignment from the database.

Add Assignment Preview



Course Administrator / Assignments / Add Assignment

POD

The purpose of the POD is to give an introductory overview of the project. After reading this fairly brief document, the reader should have an idea of: 1. Who are the users and what problem does the system solve for them? 2. The main uses of the system. 3. The context/environment in which the system is to be used. 4. The scope of the system. 5. The main factors that need to be taken in to account when designing and building the system. 6. Technologies and Risks It should be between 3 and 5 pages in length and contain 6 sections corresponding to the 6 points listed above. In Section 2, you must include at least 2 usage narratives in addition to a general description of the main uses of the system. A usage narrative is a situated example of the use of the system. It is a brief paragraph in which you invent a fictional but specific actor and briefly capture the mental state of that person "why he wants what he wants or what conditions drive him to act as he does. Capture how the world works in that particular case, from the start of the situation to the end. These narratives should give the reader a good general idea of what the system is about. Here is an example of a usage narrative for the famous Automated Teller Machine (Bankomat) "ATM" example:

1 2

Functional Requirements

10.1

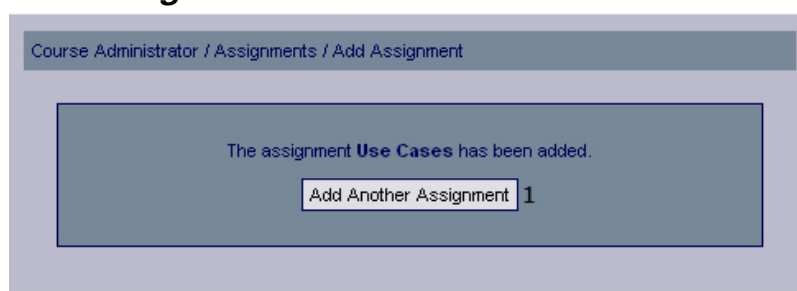
Controls

1. btnBackAddAssignment – Redirect the user to the previous page (Add Assignment) and allows the user to edit the input.
2. btnSaveAddAssignment – Invokes the method addAssignment.

Methods

addAssignment – Saves the new assignment.

Add Assignment Confirmation



Course Administrator / Assignments / Add Assignment

The assignment **Use Cases** has been added.

1

Functional Requirements

10.1

Controls

1. btnAddAssignment – Redirects the user to the add assignment page.

Methods

None

Edit Assignment

Course Administrator / Assignments / Edit Existing Assignment / Edit Assignment

Edit Assignment

Title: 1

Description: 2

The purpose of the POD is to give an introductory overview of the project. After reading this fairly brief document, the reader should have an idea of:

1. Who are the users and what problem does the system solve for them?
2. The main uses of the system.
3. The context/environment in which the system is to be used.
4. The scope of the system.
5. The main factors that need to be taken in to account when designing and building the system.
6. Technologies and Risks

It should be between 3 and 5 pages in length and contain 6 sections corresponding to the 6 points listed above.

In Section 2, you must include at least 2 usage narratives in addition to a general description of the main uses of the system. A usage narrative is a situated example of the use of the system. It is a brief paragraph in which you invent a fictional but specific actor and briefly capture the mental state of that person ' why he wants what he wants or what conditions drive him to act as he does. Capture how the world works in that particular case, from the start of the situation to the end. These narratives should give the reader a good general idea of what the system is about. Here is an example of a usage narrative for the famous Automated Teller

3

Functional Requirements

10.1

Controls

1. txtEditAssignmentTitle – The title of the new assignment.
2. txtEditAssignmentDescription – A text that describing the new assignment.
3. btnPreviewEditAssignment – Invokes the displayEditAssignment method.

Methods

displayEditAssignment - Validates the input and displays a preview of the edited assignment.

Edit Assignment Preview

Course Administrator / Assignments / Edit Existing Assignment

POD

The purpose of the POD is to give an introductory overview of the project. After reading this fairly brief document, the reader should have an idea of: 1. Who are the users and what problem does the system solve for them? 2. The main uses of the system. 3. The context/environment in which the system is to be used. 4. The scope of the system. 5. The main factors that need to be taken in to account when designing and building the system. 6. Technologies and Risks It should be between 3 and 5 pages in length and contain 6 sections corresponding to the 6 points listed above. In Section 2, you must include at least 2 usage narratives in addition to a general description of the main uses of the system. A usage narrative is a situated example of the use of the system. It is a brief paragraph in which you invent a fictional but specific actor and briefly capture the mental state of that person 'why he wants what he wants or what conditions drive him to act as he does. Capture how the world works in that particular case, from the start of the situation to the end. These narratives should give the reader a good general idea of what the system is about. Here is an example of a usage narrative for the famous Automated Teller Machine (Bankomat) 'ATM' example:

1 2

Functional Requirements

10.1

Controls

1. btnBackEditAssignment – Redirect the user to the previous page (Edit Assignment) and allows the user to edit the input.
2. btnSaveEditAssignment – Invokes the method saveEditAssignment.

Methods

saveEditAssignment – Saves the edited assignment.

Edit Assignment Confirmation

Course Administrator / Assignments / Edit Existing Assignment / Edit Assignment

The assignment **Use Cases** has been changed.

1

Functional Requirements

10.1

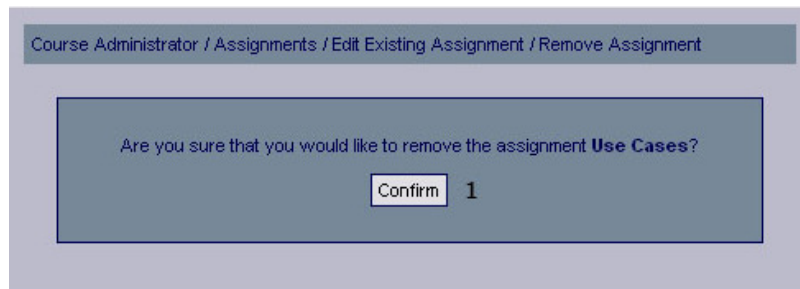
Controls

1. btnEditAssignment – Redirects the user to the assignment page.

Methods

None

Confirm Remove Assignment



Functional Requirements

10.1

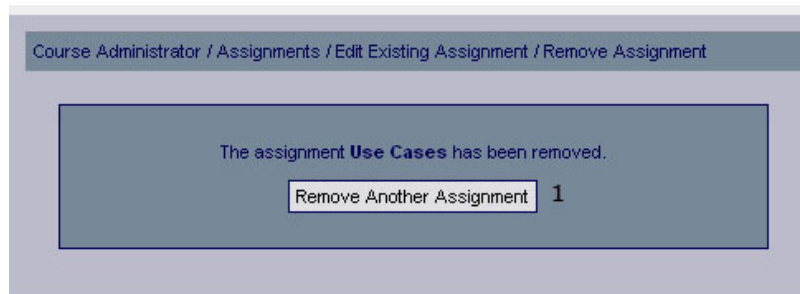
Controls

1. btnConfirmRemoveAssignment - Confirms that the selected assignment should be removed.

Methods

None

Remove Assignment Confirmation



Functional Requirements

10.1

Controls

1. btnRemoveAssignment – Redirects the user to the assignment page.

Methods

None

Course Assistants

Course Administrator / Course Assistants

Add Course Assistant

Username: 1

2

Edit Existing Course Assistant

Username	First name	Last name	Remove
user1	Olle	Johansson	Remove 3
user2	Kalle	Andersson	Remove 4

Functional Requirements

13.2

Controls

1. txtUsername – Username of the user to add.
2. btnAddCourseAssistant – Invokes the checkUser method.
3. InkRemoveCourseAssistant1 - InkRemoveCourseAssistant2 – Invokes the removeCourseAssistant method.

Methods

checkUser – Determines whether the user account exists or not, if the user account exist it redirects the user to the confirmation page.

removeCourseAssistant – Removes the specified course assistant.

Confirm Add Course Assistant

Course Administrator / Course Assistants / Add Assistant

Are you sure that you would like to add **user1** as a course assistant in the course?

1

Functional Requirements

13.2

Controls

1. btnConfirmCourseAssistant – Invokes the addCourseAssistant method.

Methods

addCourseAssistant – Adds a new course assistant for the course.

Add Course Assistant Confirmation



Functional Requirements

13.2

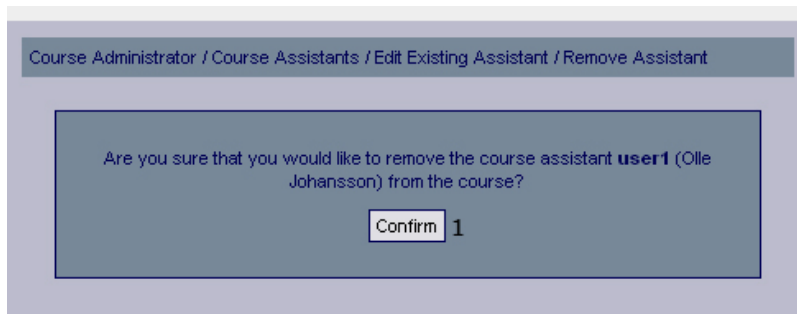
Controls

1. btnAddAnotherCourseAssistant – Redirects the user to the add course assistant page.

Methods

None

Confirm Remove Course Assistant



Functional Requirements

13.2

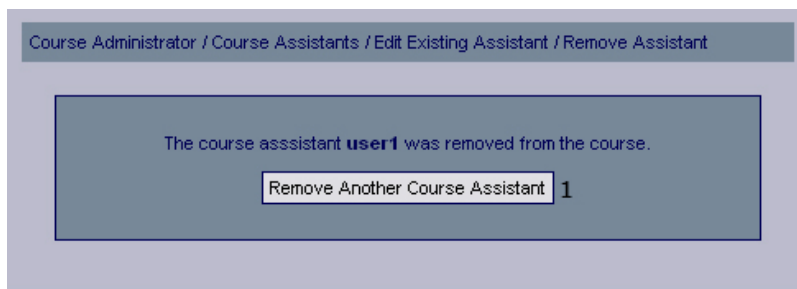
Controls

1. btnConfirmRemoveCourseAssistant – Invokes the removeCourseAssistant method.

Methods

removeCourseAssistant – Removes a course assistant from the course.

Remove Course Assistant Confirmation



Functional Requirements

13.2

Controls

1. btnRemoveAnotherCourseAssistant – Redirects the user to the remove course assistant page.

Methods

None

Edit Course Description

Course Administrator / Course Description / Edit Description

Edit Course Description

Course Name: 1

Credits: 2

Begins in Period: 3 4

Ends in Period: 5 6

Description:

Save 8

Functional Requirements

4.1

Controls

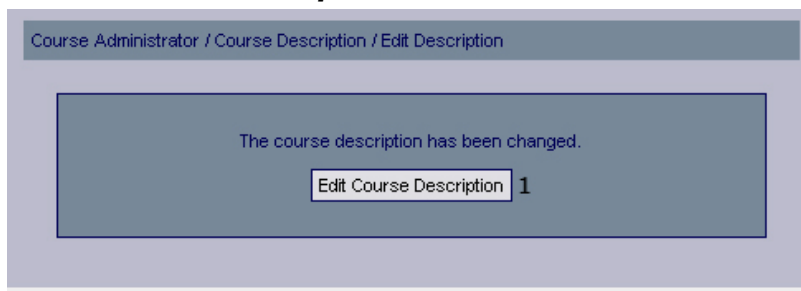
1. txtCourseName – The name of the course.
2. txtCourseCredit – The credit of the course.
3. drpCourseBeginYear – The year that the course starts.
4. drpCourseBeginPeriod – The period that the course starts..

5. drpCourseEndYear – The year that the course ends.
6. drpCourseEndPeriod – The period that the course ends.
7. txtCourseDescription – The descriptions of the course.
8. btnSaveCourseDescription – Invokes the saveCourseDescription method.

Methods

saveCourseDescription – Saves the changes to the course description.

Edit Course Description Confirmation



Functional Requirements

4.1

Controls

1. btnEditCourseDescription – Redirects the user to the edit course description page.

Methods

None

Create Course Website – Create Description

Course Administrator / Create Course Website / Description

Course Description

Course Name: 1

Credits: 2

Begins in Period: 3 4

Ends in Period: 5 6

Description: 7

8

Functional Requirements

- 3.1
- 4.1

Controls

1. txtCourseName – The name of the course.
2. txtCourseCredit – The credit of the course.
3. drpCourseBeginYear – The year that the course starts.
4. drpCourseBeginPeriod – The period that the course starts..
5. drpCourseEndYear – The year that the course ends.
6. drpCourseEndPeriod – The period that the course ends.
7. txtCourseDescription – The descriptions of the course.
8. btnSaveCourseDescription – Invokes the saveCourseDescription method.

Methods

btnContinueToSchedule – Saves the course description and continues to the next step in the creation guide (Import Schedule).

Create Course Website – Import Schedule



The screenshot shows a web interface for importing a schedule. At the top, a breadcrumb trail reads 'Course Administrator / Create Course Website / Schedule'. Below this, the title 'Import Schedule' is displayed. The main form area contains an 'iCalendar file:' label followed by a text input field (labeled '1') and a 'Browse...' button (labeled '2'). Below the input field are two buttons: 'Skip' (labeled '3') and 'Import' (labeled '4').

Functional Requirements

3.1
7.1

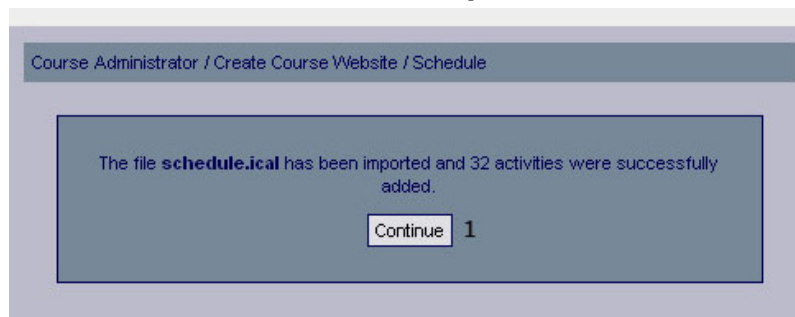
Controls

1. txtScheduleDirectory – The directory of the schedule in iCalendar format.
2. btnBrowseSchedule – Selects the schedule the user selects to import.
3. btnSkipSchedule – Redirects the user to the next step in the creation guide (Add information Page).
4. btnImportSchedule – Invokes the importSchedule method.

Methods

importSchedule – Validates the input, generates a schedule that can be viewed through the system and saves it.

Create Course Website – Import Schedule Confirmation



The screenshot shows a confirmation message box. At the top, a breadcrumb trail reads 'Course Administrator / Create Course Website / Schedule'. The message box contains the text: 'The file **schedule.ical** has been imported and 32 activities were successfully added.' Below the message is a 'Continue' button (labeled '1').

Functional Requirements

3.1
7.1

Controls

1. btnContinueToAddInformationPage – Redirects the user to the add information page.

Methods

None

Create Course Website –Add Information Pages

Course Administrator / Create Course Website / Information Pages

Add Information Page

Title: 1

Content: 2

3 4

Functional Requirements

3.1

6.1

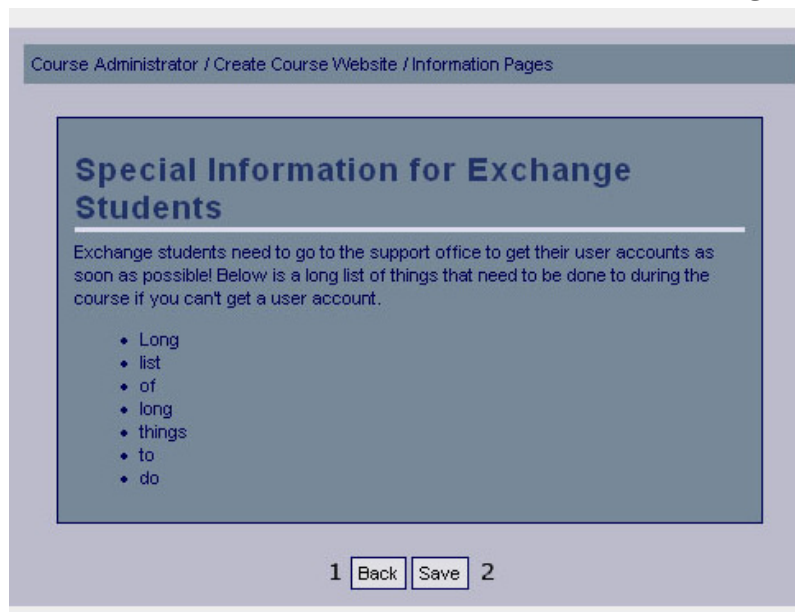
Controls

1. txtInformationPageTitle – The title of the information page.
2. txtInformationPageContent – The content of the information page.
3. btnSkipInformationPage – Redirects the user to the next step (Add Deadlines) in the creation guide.
4. btnPreviewInformationPage – Invokes the displayAddInformationPage method.

Methods

displayAddInformationPage - Validates the input and displays a preview of the new information page.

Create Course Website – Add Information Pages Preview



Functional Requirements

- 3.1
- 6.1

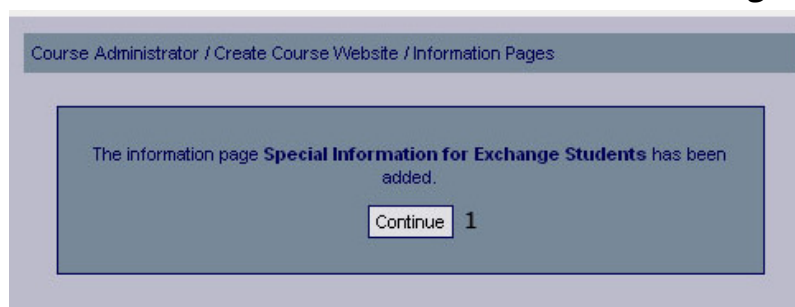
Controls

1. btnBackAddInformationPage - Redirect the user to the previous page (Add Information Page) and allows the user to edit the input.
2. btnSaveAddInformationPage – Invokes the method addInformationPage.

Methods

addInformationPage – Saves the information page.

Create Course Website – Add Information Pages Confirmation



Functional Requirements

- 3.1
- 6.1

Controls

1. btnContinueToAddDeadlines – Redirects the user to the next step (Add Deadlines) in the creation guide.

Methods

None

Create Course Website – Add Deadlines

The screenshot shows a web form titled "Add Deadline" within a "Course Administrator / Create Course Website / Deadlines" context. The form includes a "Title:" text input field (labeled 1), a "Time:" section with five dropdown menus for day (01, labeled 2), month (January, labeled 3), year (2008, labeled 4), minutes (00, labeled 5), and hours (00, labeled 6), and a "Description:" text area (labeled 7). At the bottom right, there are "Skip" and "Preview" buttons (labeled 8 and 9 respectively).

Functional Requirements

3.1

8.1

Controls

1. txtTitleAddDeadline – The title of the new deadline.
2. drpDateDay – The day part of the date the deadline is for.
3. drpDateMonth – The month part of the date the deadline is for.
4. drpDateYear – The year part of the date the deadline is for.
5. drpTimeMinutes – The minutes part of the time the deadline is for.
6. drpTimeHour – The hour part of the time the deadline is for.
7. txtDescriptionAddDeadline – A text describing the new deadline.
8. btnPreviewDeadline – Invokes the displayAddDeadline method.

Methods

displayAddDeadline– Validates input and displays a preview of the new deadline.

Create Course Website – Add Deadlines Preview

The screenshot shows a preview of a use case. The title is "Use Cases" and the date is "27th November, 2007". The text of the use case reads: "You will review the requirements and use cases for 2 other groups. As in the POD reviews, these reviews will not be anonymous. You will be provided with a review form to be filled out for each set of requirements and use cases that you review. You will then send a copy of each review to the respective project coordinators and one copy of all three reviews to me. You will find details here." At the bottom, there are "Back" and "Save" buttons (labeled 1 and 2 respectively).

Functional Requirements

3.1

8.1

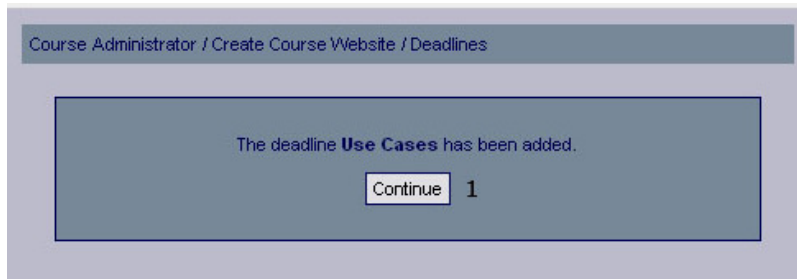
Controls

1. btnBackAddDeadline - Redirect the user to the previous page (Add Deadline) and allows the user to edit the input.
2. btnSaveAddDeadline - Invokes the method addDeadline.

Methods

addDeadline - Saves the deadline.

Create Course Website – Add Deadlines Confirmation



Functional Requirements

- 3.1
- 8.1

Controls

1. btnContinueToCreationGuidePreview - Invokes displayCreationGuide

Methods

displayCreationGuide - Displays a summary of the objects created by the user throughout the creation guide.

Create Course Website Summary

Course Administrator / Create Course Website

Course Description

Course Code: DD1363
Credits: 7,5
Begins in period: 2007, 1
Ends in period: 2008, 4

Schedule

- Lecture, E1 (15.00-17.00, 1st December, 2007)
- Lecture, D1 (10.00-12.00, 2nd December, 2007)
- Lecture, D1 (10.00-12.00, 3rd December, 2007)
- Lecture, E1 (08.00-10.00, 4th December, 2007)
- Lecture, D1 (13.00-15.00, 5th December, 2007)

Information Pages

- Special Information for Exchange Students

Deadlines

- POD
- Use Cases
- Use Case Reviews

1 2

Functional Requirements

3.1
8.1

Controls

1. btnBackCreationGuide - Redirect the user to the add deadline page and allows the user to edit the input.
2. btnCreateCourseWebsite – Invokes the createCourseWebsite method.

Methods

createCourseWebsite – Creates the course website.

Create Course Website Confirmation

Course Administrator / Create Course Website

The course website for course **DD1363** has been created.

1

Functional Requirements

3.1

Controls

1. btnCourseWebsite – Redirects the user to the new course website.

Methods

None

Deadlines

The screenshot shows a web interface for managing deadlines. At the top, there is a breadcrumb 'Course Administrator / Deadlines'. Below it is a section titled 'Add Deadline' with a form containing: a 'Title' text box (labeled 1), a 'Time' section with five dropdown menus for day (01, labeled 2), month (January, labeled 3), year (2008, labeled 4), minutes (00, labeled 5), and hours (00, labeled 6), a 'Description' text area (labeled 7), and a 'Preview' button (labeled 8). Below this is a section titled 'Edit Existing Deadline' containing a table of existing deadlines.

Title	Time	Edit	Remove
POD	2007-11-01 18:00	9	12
Use Cases	2007-11-30 18:00	10	13
Use Case Reviews	2007-12-07 18:00	11	14

Functional Requirements

8.1

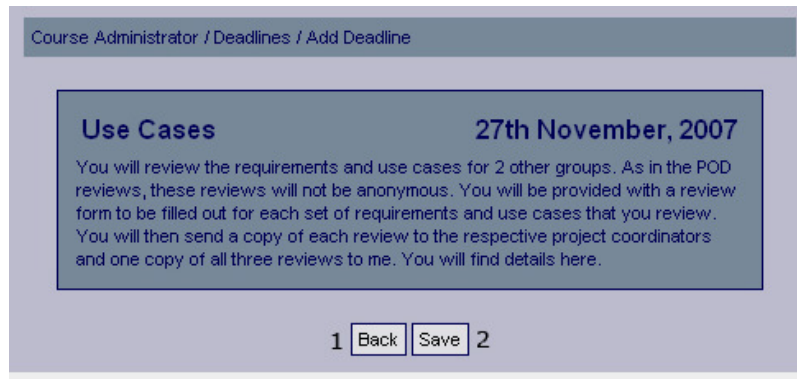
Controls

1. txtTitle – The title of the new deadline.
2. drpDateDay – The day part of the date the deadline is for.
3. drpDateMonth – The month part of the date the deadline is for.
4. drpDateYear – The year part of the date the deadline is for.
5. drpTimeMinutes – The minutes part of the time the deadline is for.
6. drpTimeHour – The hour part of the time the deadline is for.
7. txtDescription – A text describing the new deadline.
8. btnPreviewDeadline – Shows a preview of the deadline with the provided content..
- 9-11 InkEditDeadline1 - InkEditDeadline3 – Invokes the editDeadline method.
- 12-14. InkRemoveDeadline1 - InkRemoveDeadline3 – Invokes the removeDeadline method.

Methods

- editDeadline – Retrieves the data for the specified deadline and allows the user to edit that data.
- removeDeadline – Removes the specified deadline.

Add Deadline Preview



Functional Requirements

8.1

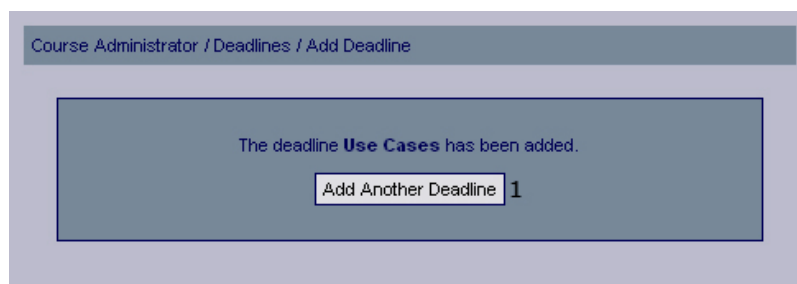
Controls

1. btnBackAddDeadline – Redirect the user to the previous page (Add Deadline) and allows the user to edit the input.
2. btnSaveAddDeadline – Invokes the method addDeadline.

Methods

addDeadline – Saves the new deadline.

Add Deadline Confirmation



Functional Requirements

8.1

Controls

1. btnAddDeadline – Redirects the user to the add deadline page.

Methods

None

Edit Deadlines

Course Administrator / Deadlines / Edit Existing Deadline / Edit Deadline

Edit Deadline

Title: 1

Time: 2 3 4 5 6

Description: 7

8

Functional Requirements

8.1

Controls

1. txtTitle – The title of the new deadline.
2. drpDateDay – The day part of the date the deadline is for.
3. drpDateMonth – The month part of the date the deadline is for.
4. drpDateYear – The year part of the date the deadline is for.
5. drpTimeMinutes – The minutes part of the time the deadline is for.
6. drpTimeHour – The hour part of the time the deadline is for.
7. txtDescription – A text describing the new deadline.
8. btnPreviewDeadline – Shows a preview of the deadline with the provided content.

Methods

None

Edit Deadline Preview

Course Administrator / Deadlines / Edit Existing Deadline

Use Cases **27th November, 2007**

You will review the requirements and use cases for 2 other groups. As in the POD reviews, these reviews will not be anonymous. You will be provided with a review form to be filled out for each set of requirements and use cases that you review. You will then send a copy of each review to the respective project coordinators and one copy of all three reviews to me. You will find details here.

1 2

Functional Requirements

8.1

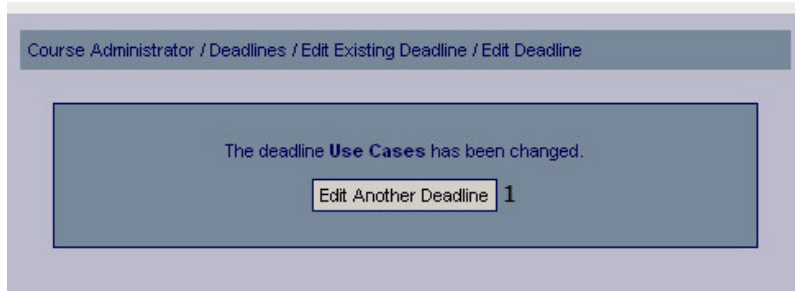
Controls

1. btnBackEditDeadline – Redirect the user to the previous page (Edit Deadline) and allows the user to edit the input.
2. btnSaveEditAssignment – Invokes the method saveEditDeadline.

Methods

saveEditDeadline – Saves the edited deadline.

Edit Deadline Confirmation



Functional Requirements

8.1

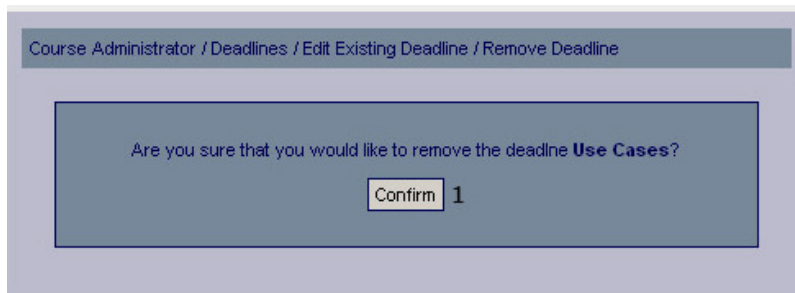
Controls

1. btnEditAnotherDeadline – Redirects the user to the list of existing deadlines.

Methods

None

Confirm Deadline Removal



Functional Requirements

8.1

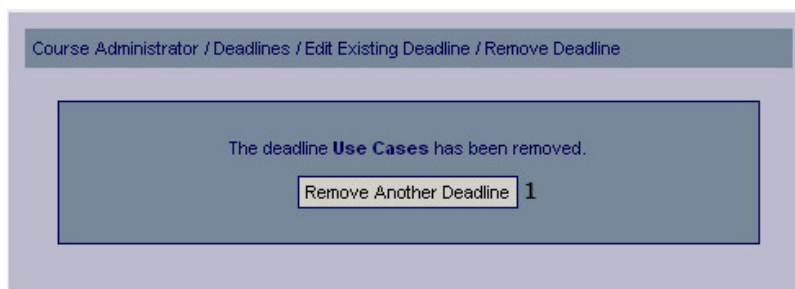
Controls

1. btnConfirm – Confirms that the selected deadline should be removed.

Methods

None

Deadline Removal Confirmation



Functional Requirements

8.1

Controls

1. btnDeleteAnotherDeadline – Redirects the user to the list of existing deadlines.

Methods

None

Information Pages

Course Administrator / Information Pages

Add Information Page

Title: 1

Content: 2

3

Edit Existing Information Page

Title	
Special Information for Exchange Students	<input type="button" value="Edit"/> 4 <input type="button" value="Remove"/> 5

Functional Requirements

6.1

Controls

1. txtInfoPageTitle – The title of the information page.

2. txtInfoPageContent – The content of the information page.

3. btnPreviewInfoPage – Shows a preview of the information page with the provided content.

4. InkEditInfoPage – Invokes the editInfoPage method.

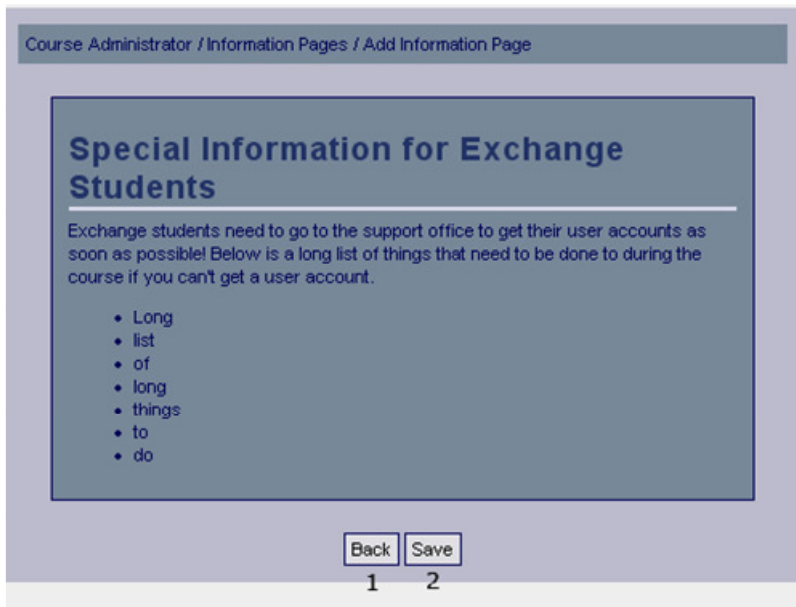
5. InkRemoveInfoPage – Invokes the removeInfoPage method.

Methods

editInfoPage – Retrieves the data for the specified information page and allows the user to edit the data.

removeInfoPage – Removes the specified information page.

Add Information Page Preview



Functional Requirements

6.1

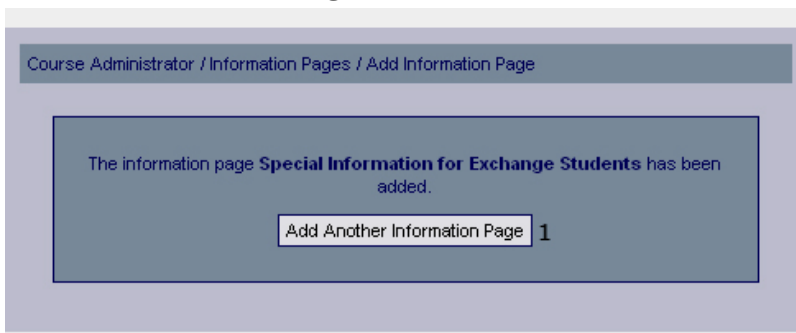
Controls

1. btnBack – Returns the user to the previous page.
2. btnAddInfoPage – Invokes the method addInfoPage.

Methods

addInfoPage – Adds the information page to the database.

Add Information Page Confirmation



Functional Requirements

6.1

Controls

1. btnAddInformationPage – Redirects the user to the add information page.

Methods

None

Edit Information Page

Course Administrator / Information Pages / Edit Existing Information Page / Edit Information Page

Edit Information Page

Title: Special Information for Exchange Students 1

Content: 2

```
<div id="breadcrumbs"><a href="#">Software Engineering</a> /  
Special Information for Exchange Students</div>  
  
<h1>Special Information for Exchange Students</h1>  
  
Exchange students need to go to the support office to get their user  
accounts as soon as possible! Below is a long list of things that need  
to be done to during the course if you can't get a user account.  
  
<ul>  
  <li>Long</li>  
</ul>
```

Preview 3

Functional Requirements

6.1

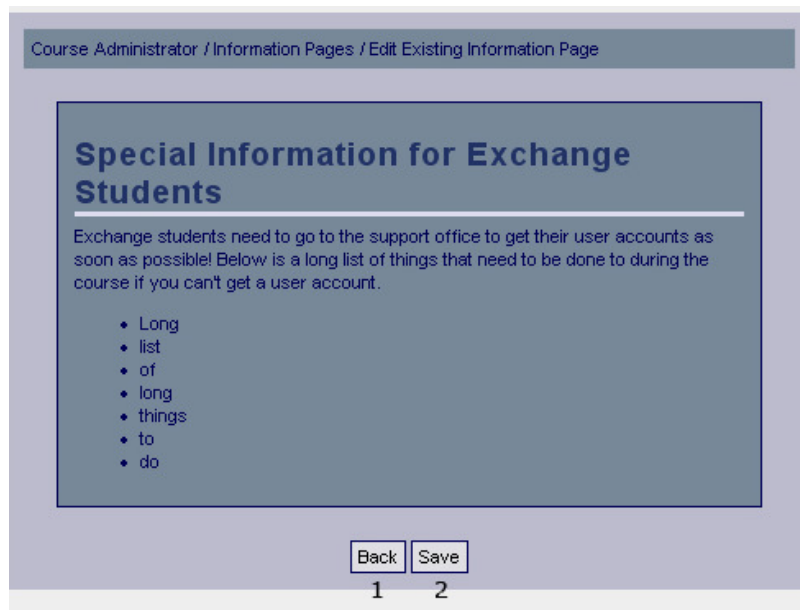
Controls

1. txtInfoPageTitle – The title of the information page.
2. txtInfoPageContent – The content of the information page.
3. btnPreviewInfoPage – Shows a preview of the information page with the provided content.

Methods

None

Edit Existing Information Page Preview



Functional Requirements

6.1

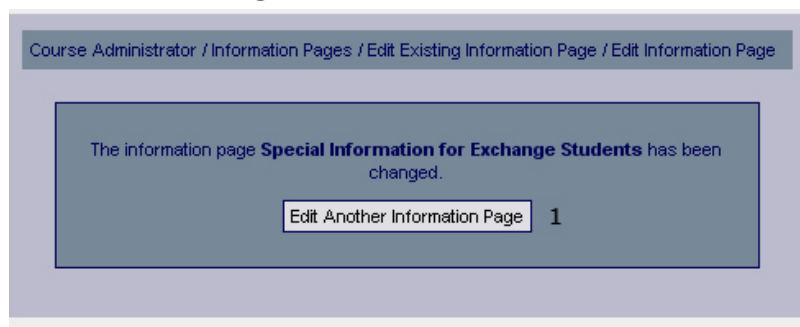
Controls

1. btnBack – Returns the user to the previous page.
2. btnEditInfoPage – Invokes the method editInfoPage.

Methods

editInfoPage – Updates the information page in the database, with the provided information.

Information Page Edited Confirmation



Functional Requirements

6.1

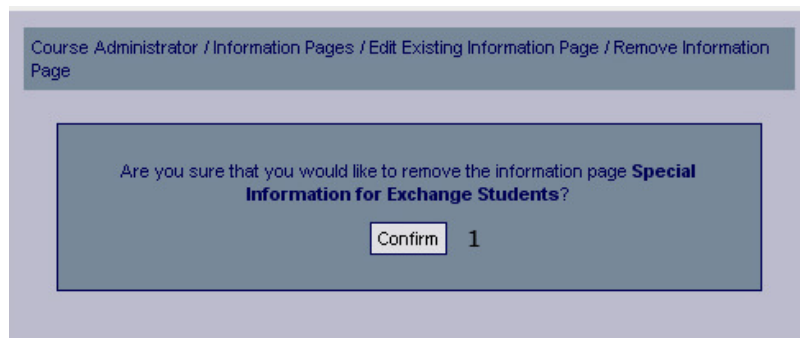
Controls

1. btnEditAnotherInfoPage – Redirects the user to the “Information pages” page.

Methods

None

Confirm Removal of Information Page



Functional Requirements

6.1

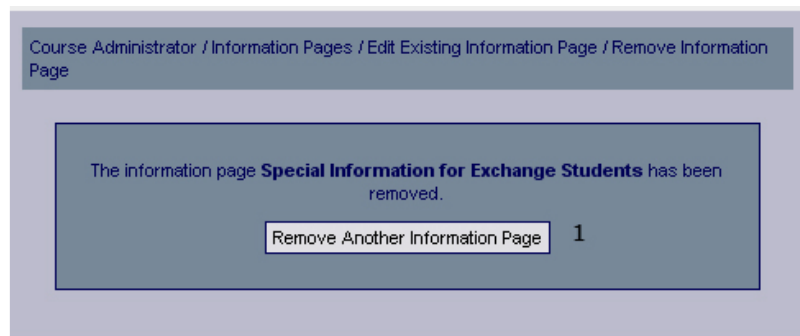
Controls

1. btnConfirmInfoPageRemoval – Invokes the method removeInfoPage.

Methods

removeInfoPage – Removes the information page in question from the database.

Information Page Removed Confirmation



Functional Requirements

6.1

Controls

1. btnRemoveAnotherInfoPage – Redirects the user to the "Information pages" page.

Methods

None

News

Course Administrator / News

Add News

Headline: 1

Content: 2

3

Edit Existing News

Title			
New lecture scheduled	Edit	4	Remove 7
Lecture cancelled (again)	Edit	5	Remove 8
Lecture cancelled	Edit	6	Remove 9

Functional Requirements

5.1

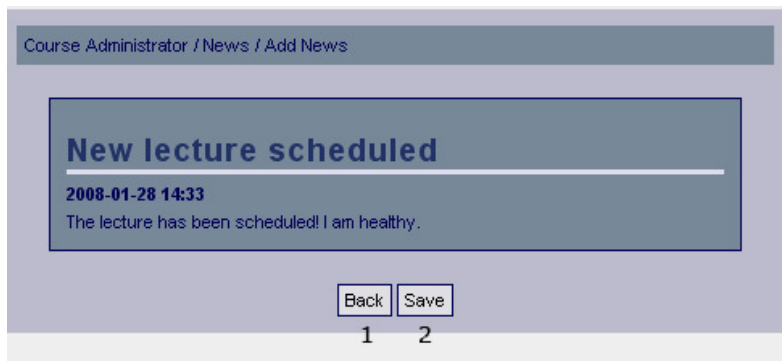
Controls

1. txtNewsHeadline – The headline of the news post.
2. txtNewsContent – The content of the news post.
3. btnPreviewAddNews – Shows a preview of the activity, with the provided information.
- 4-6. InkEditNews1 - InkEditNews3 – Invokes the editNews method.
- 7-8. InkRemoveNews1 - InkRemoveNews3 – Invokes the removeNews method.

Methods

- editNews – Retrieves the data for the specified news and allows the user to edit the data.
- removeNews – Removes the specified news.

Add News Preview



Functional Requirements

5.1

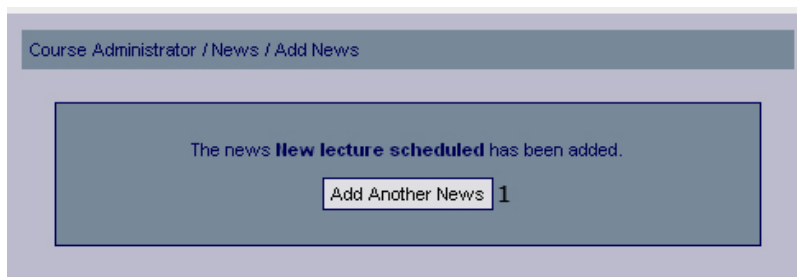
Controls

1. btnBack – Returns the user to the previous page.
2. btnAddNews – Invokes the method addNews.

Methods

addNews – Adds the news post to the database.

Add News Confirmation



Functional Requirements

5.1

Controls

1. btnAddNews – Redirects the user to the add news page.

Methods

None

Edit News

Course Administrator / News / Edit Existing News / Edit News

Edit News

Headline: 1

Content: 2

3

Functional Requirements

5.1

Controls

1. txtNewsHeadline – The headline of the news post.
2. txtNewsContent – The content of the news post.
3. btnPreviewEditNews – Shows a preview of the updated news post, with the provided information.

Methods

None

Edit Existing News Preview

Course Administrator / News / Edit Existing News

New lecture scheduled

2008-01-28 14:33

The lecture has been scheduled! I am healthy.

1 2

Functional Requirements

5.1

Controls

1. btnBack – Returns the user to the previous page.
2. btnEditNews – Invokes the method editNews.

Methods

editNews – Updates the news post in the database, with the information provided.

News Edited Confirmation



Functional Requirements

5.1

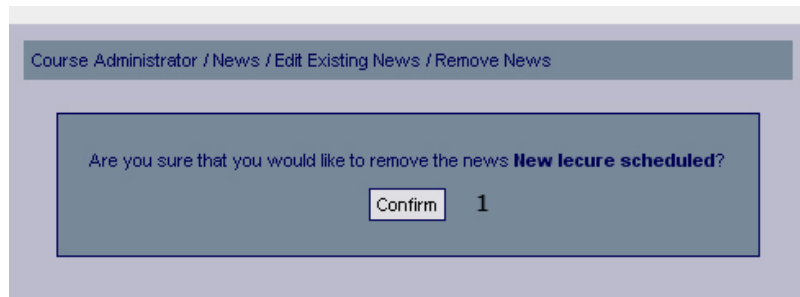
Controls

1. btnEditAnotherNews – Redirects the user to the news management page for system administrators.

Methods

None

Confirm Removal of News



Functional Requirements

5.1

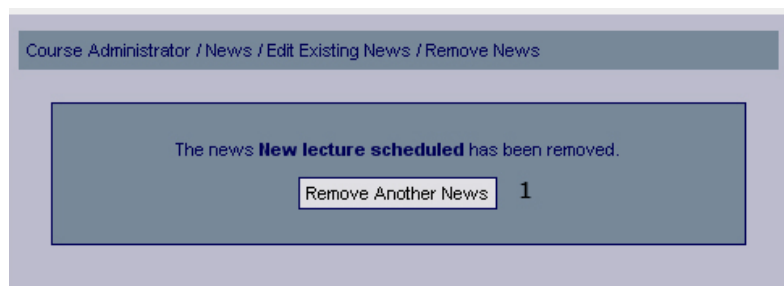
Controls

1. btnConfirmNewsRemoval – Invokes the method removeNews.

Methods

removeNews – Removes the news post in question from the database.

News Removed Confirmation



Functional Requirements

5.1

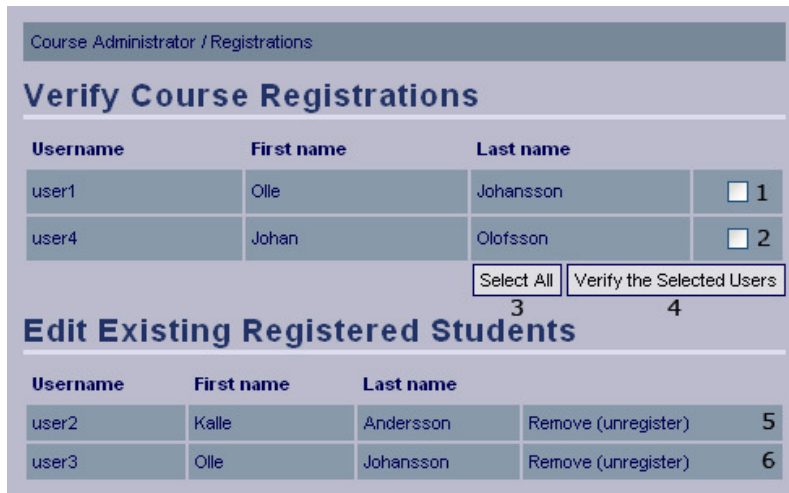
Controls

1. btnRemoveAnotherNews – Redirects the user to the news management page for system administrators.

Methods

None

Registrations



Course Administrator / Registrations

Verify Course Registrations

Username	First name	Last name	
user1	Olle	Johansson	<input type="checkbox"/> 1
user4	Johan	Olofsson	<input type="checkbox"/> 2

Select All 3 Verify the Selected Users 4

Edit Existing Registered Students

Username	First name	Last name		
user2	Kalle	Andersson	Remove (unregister)	5
user3	Olle	Johansson	Remove (unregister)	6

Functional requirements

12.1

12.2

12.4

Controls

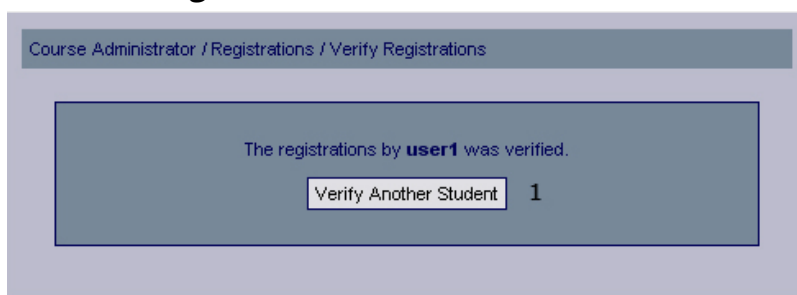
1. chkX_1 – Checkbox to select user1 (Olle Johansson) for verification.
2. chkX_2 – Checkbox to select user4 (Johan Olofsson) for verification.
3. btnSelectAll – Checks all the checkboxes.
4. btnVerifySelected – Invokes the method verifyRegistrations.
- 5-6. InkRemoveStudent1 - InkRemoveStudent2 – Invokes the removeStudent method.

Methods

verifyRegistrations – Updates the status for the selected students.

removeStudent – Removes the student from the specified course.

Verified Registrations Confirmation



Course Administrator / Registrations / Verify Registrations

The registrations by **user1** was verified.

Verify Another Student 1

Functional requirements

12.2

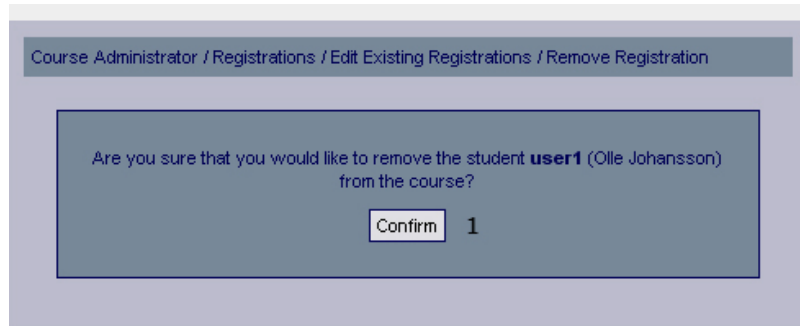
Controls

1. btnVerifyAnotherStudent – Redirects the user to the Registrations page.

Methods

None

Confirm Removal of Student Registration



Functional requirements

12.4

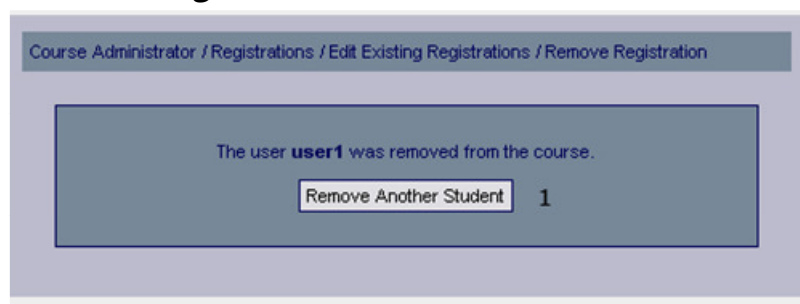
Controls

1. btnConfirmRegistrationRemoval – Invokes the method removeStudentReg.

Methods

removeStudentReg – Removes the student registration in question from the database.

Student Registration Removed Confirmation



Functional requirements

12.4

Controls

1. btnRemoveAnotherStudent – Redirects the user to the Registrations page.

Methods

None

Register Results

Username	First name	Last name	POD	Use Cases
user2	Kalle	Andersson	B 1	2
user3	Olle	Johansson	C 3	4

Save the results

5

Functional requirements

11.1

Controls

1. txtGrade_UserAssign_1_1 – The grade of the first assignment (POD) for the first student.
2. txtGrade_UserAssign_1_2 – The grade of the second assignment (Use Cases) for the first student.
3. txtGrade_UserAssign_2_1 – The grade of the first assignment (POD) for the second student.
4. txtGrade_UserAssign_2_2 – The grade of the second assignment (Use Cases) for the second student.
5. btnSaveRes – Invokes the method saveResults.

Methods

saveResults – Updates the results in the database.

Results Registered Confirmation

Course Administrator / Results

2 new results were saved.

Register More Results

1

Functional requirements

11.1

Controls

1. btnRegisterMoreResults – Redirects the user to the Register results page.

Methods

None

Files

Course Administrator / File

Add File

Title: 1

Description: 2

File: 3 4

5

Edit Existing News

File	Date	Size	Edit	Remove
PowerPoint from Lecture 1	3rd November, 2007	501 kb	Edit 6	Remove 12
PowerPoint from Lecture 2	10th November, 2007	437 kb	Edit 7	Remove 13
PowerPoint from Lecture 3	17th November, 2007	358 kb	Edit 8	Remove 14
Extra Material on Use Cases	24th November, 2007	603 kb	Edit 9	Remove 15
PowerPoint from Lecture 4	1st December, 2007	920 kb	Edit 10	Remove 16
PowerPoint from Lecture 5	10th December, 2007	899 kb	Edit 11	Remove 17

Functional Requirements

9.1

Controls

1. txtAddFileTitle – The title of the file.
2. txtAddFileDescription – The description of the file.
3. txtAddFileSource – The directory of the file.
4. btnSelectAddFile – Selects the file the user specified for upload.
5. btnSaveAddFile – Invokes the uploadAddFile method.
- 6-11. InkEditFile1 - InkEditFile6 – Invokes the editFile method.
- 12-17. InkRemoveFile1 - InkRemoveFile6 – Invokes the removeFile method.

Methods

- uploadAddFile – Uploads the file and saves the title and description.
editFile – Retrieves the data for the specified file and allows the user to edit the data.
removeFile – Removes the file.

Add File Confirmation

Course Administrator / Files / Add File

The file **PowerPoint from Lecture 2** has been added.

1

Functional Requirements

9.1

Controls

1. btnAddFile – Redirects the user to the add files page.

Methods

None

Edit File

Course Administrator / Files / Edit Existing File / Edit File

Edit File

Title: PowerPoint from Lecture : 1 1

Description: The PowerPoint slides from the second lecture. 2

File: 3 Browse... 4

Save 5

Functional Requirements

9.1

Controls

1. txtEditFileTitle – The new title of the file.
2. txtEditFileDescription – The new description of the file.
3. txtEditFileSource – The directory of the new file, if the user wants a new file.
4. btnSelectEditFile – Selects the new file the user specified for upload.
5. btnSaveEditFile – Invokes the uploadAddFile method.

Methods

uploadEditFile – Uploads the new file, if specified, and saves the changes.

Edit File Confirmation

Course Administrator / Files / Edit Existing File / Edit File

The file **PowerPoint from Lecture 2** has been changed.

Edit Another File 1

Functional Requirements

9.1

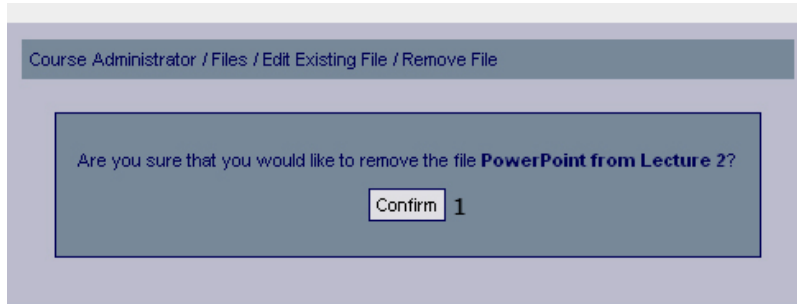
Controls

1. btnEditFile – Redirects the user to the add files page.

Methods

None

Confirm Remove File



Functional Requirements

9.1

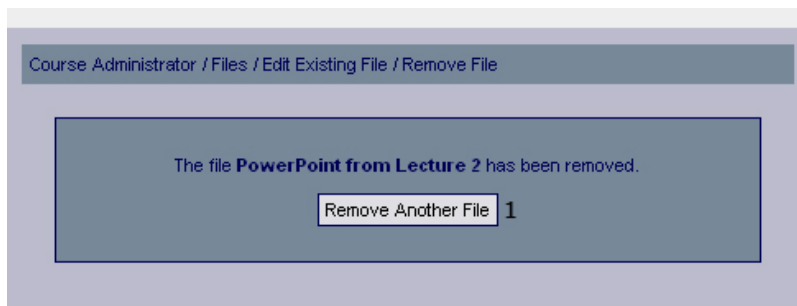
Controls

1. btnConfirmRemoveFile – Confirm that the selected file should be removed.

Methods

None

Remove File Confirmation



Functional Requirements

9.1

Controls

1. btnRemoveFile - Redirects the user to the add files page.

Methods

None

System Administration Pages

Courses

The screenshot shows two forms in a web application. The top form is titled 'Add Course' and has a breadcrumb 'System Administrator / Courses'. It contains a text input field labeled 'Course Code:' with a '1' next to it, and a button labeled 'Add' with a '2' next to it. The bottom form is titled 'Edit Existing Course' and also has a breadcrumb 'System Administrator / Courses'. It contains a text input field labeled 'Course Code:' with a '3' next to it, and a button labeled 'Search' with a '4' next to it.

Functional Requirements

13.3

Controls

1. txtCourseCode – Course code for the course to be added.
2. btnAddCourse – Invokes the addCourse method.
3. txtCourseCode – Course code for the course to edit.
4. btnSearchCourse – Invokes the searchCourse method.

Methods

addCourse - Adds the course with the given course code to the system.
searchCourse – Searches for the course with the given course code.

Course Added Confirmation

The screenshot shows a confirmation page with a breadcrumb 'System Administrator / Courses / Add'. The main content area has a message: 'A new course with course code **DD1363** has been added.' Below the message are two buttons: 'Assign Course Leader' with a '1' next to it, and 'Add Another Course' with a '2' next to it.

Functional Requirements

13.3

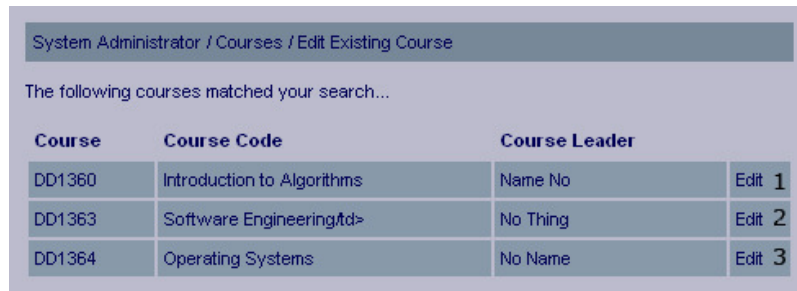
Controls

1. btnAssignCourseLeader – Redirects the user to the user privileges page for courses.
2. btnAddAnotherCourse – Redirects the user to the course management page for system administrators.

Methods

None

Existing Courses List



Course	Course Code	Course Leader	
DD1360	Introduction to Algorithms	Name No	Edit 1
DD1363	Software Engineering/td>	No Thing	Edit 2
DD1364	Operating Systems	No Name	Edit 3

Functional Requirements

13.3

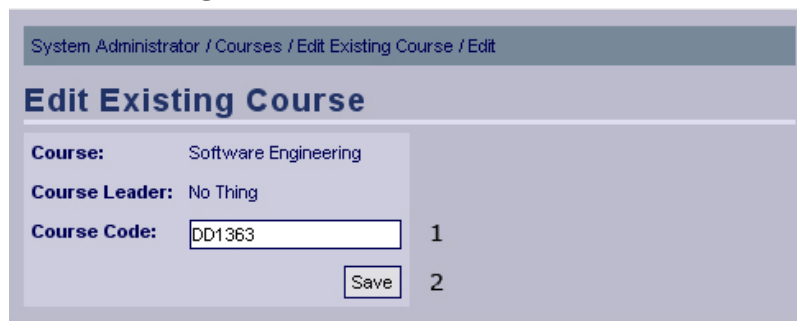
Controls

1-3. InkEditCourse1 - InkEditCourse3 – Invokes the editCourse method.

Methods

editCourse – Retrieves the data for the specified course and allows the user to edit the data.

Edit Existing Course



Functional Requirements

13.3

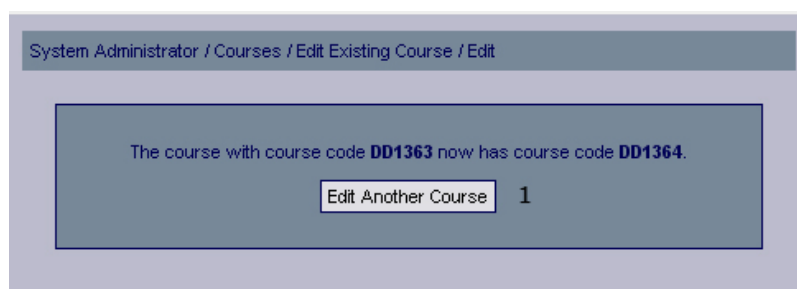
Controls

1. txtCourseCode – The new desired course code for the course.
2. btnSaveCourseCode – Invokes the method setCourseCode.

Methods

setCourseCode – Saves the given course code to the database.

Course Edited Confirmation



Functional Requirements

13.3

Controls

1. btnEditAnotherCourse – Redirects to the course management page for system administrators.

Methods

None

Users

The screenshot shows a web interface for user management. At the top, there is a breadcrumb trail: "System Administrator / Users". Below this, there are two main sections. The first section is titled "Add User" and contains three controls: a text input field for "Username:" labeled "1", a text input field for "Password:" labeled "2", and a button labeled "Add" labeled "3". The second section is titled "Edit Existing User" and contains two controls: a text input field for "Username, first name or last name:" labeled "4", and a button labeled "Search" labeled "5".

Functional Requirements

13.4

Controls

1. txtUsername – Username of the user to add.
2. txtPassword – Password of the user to add.
3. btnAdd – Invokes the addUser method.
4. txtSearchString – Search string used for finding a user.
5. btnSearch – Invokes the findUser method.

Methods

addUser – Adds a new user using the provided username and password

findUser – Attempts to find a user using the given search string.

Add User Confirmation

The screenshot shows a confirmation message in a web interface. At the top, there is a breadcrumb trail: "System Administrator / Users / Add". Below this, there is a message box with a blue background and white text that reads: "A new user with username nice_user has been added." Below the message, there are two buttons: "Assign Privileges to User" labeled "1" and "Add Another User" labeled "2".

Functional Requirements

13.4

Controls

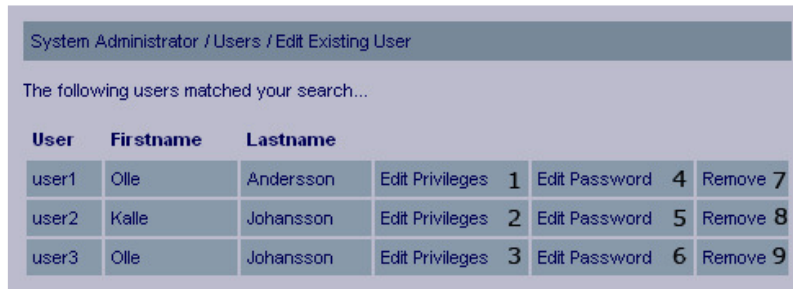
btnAssignUserPrivileges – Redirects the user to the edit user privileges page.

btnAddAnotherUser – Redirects the user to the add user page.

Methods

None

Existing Users



System Administrator / Users / Edit Existing User

The following users matched your search...

User	Firstname	Lastname	Edit Privileges	1	Edit Password	4	Remove	7
user1	Olle	Andersson	Edit Privileges	2	Edit Password	5	Remove	8
user2	Kalle	Johansson	Edit Privileges	3	Edit Password	6	Remove	9

Functional Requirements

13.1

13.4

Controls

1-3. InkEditPrivileges1 - InkEditPrivileges3 – Invokes the editPrivileges method.

4-6. InkEditPassword1 - InkEditPassword3 – Invokes the editPassword method.

7-9. InkRemoveUser1 - InkRemoveUser3 – Invokes the removeUser method.

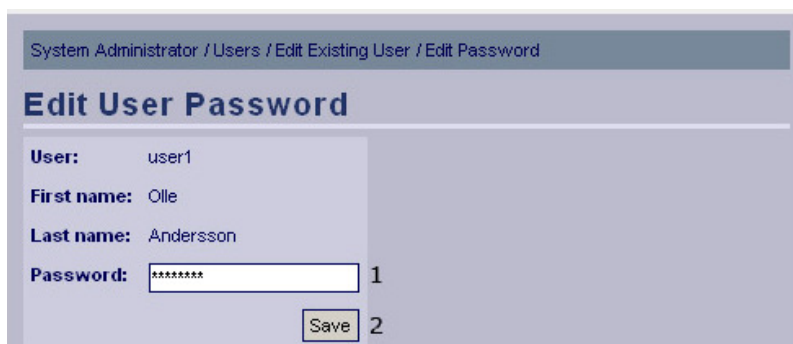
Methods

editPrivileges – Let the user edit the specified users privileges.

editPassword – Let the user edit the specified users password.

removeUser – Let the user remove the specified user.

Edit Password



System Administrator / Users / Edit Existing User / Edit Password

Edit User Password

User: user1

First name: Olle

Last name: Andersson

Password: 1

2

Functional Requirements

13.4

Controls

1. txtPassword – The new password for the user.

2. btnSavePassword – Invokes the savePassword method.

Methods

savePassword – Saves the new password for the user in the database.

Edit Password Confirmation



Functional Requirements

13.4

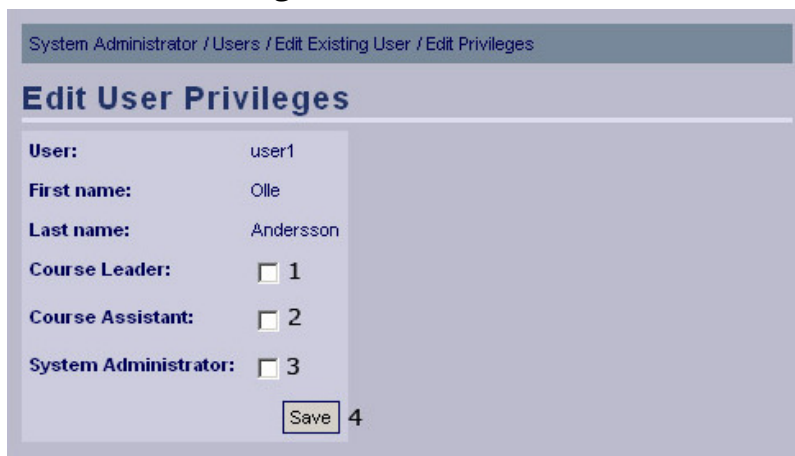
Controls

1. btnEditAnotherUser – Redirects the user to the existing users page.

Methods

None

Edit User Privileges



Functional Requirements

13.1

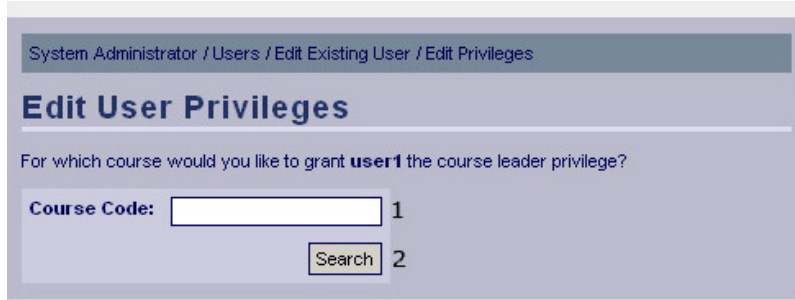
Controls

1. chkCourseLeader – Indicates whether the user should have course leader privileges.
2. chkCourseAssistant – Indicates whether the user should have course assistant privileges.
3. chkSystemAdministrator – Indicates whether the user should have system administrator privileges.
4. btnSavePrivileges – Invokes the checkPrivileges method.

Methods

checkPrivileges – Determines whether more input is required to assign the selected privileges (happens if course leader or assistant has been selected), if not sends the user to the ask for confirmation screen.

Edit User Privileges – Course Privileges – Find Course



System Administrator / Users / Edit Existing User / Edit Privileges

Edit User Privileges

For which course would you like to grant **user1** the course leader privilege?

Course Code: 1

2

Functional Requirements

13.1


Controls

1. txtCourseCode – Course code of the course for which to give the user privileges.
2. btnSearch – invokes the findCourse method.

Methods

findCourse – Attempts to find courses matching the given course code.

Edit User Privileges – Course Privileges – Select Course



System Administrator / Users / Edit Existing User / Edit Privileges

The following courses matched your search...

Course Code	Course	Select	
DD1360	Introduction to Algorithms	Select	1
DD1363	Software Engineering	Select	2
DD1364	Operating Systems	Select	3

Functional Requirements

13.1

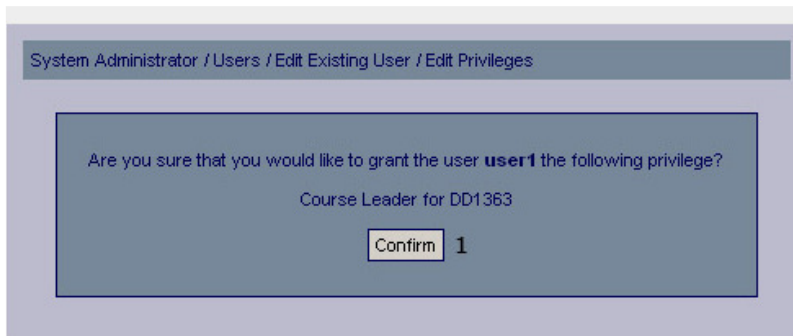
Controls

- 1-3. InkSelectCourse1 - InkSelectCourse3 – Invokes the editPrivileges method.

Methods

editPrivileges – Adds the specified course to the specified users privileges.

Confirm Edit User Privileges



Functional Requirements

13.1

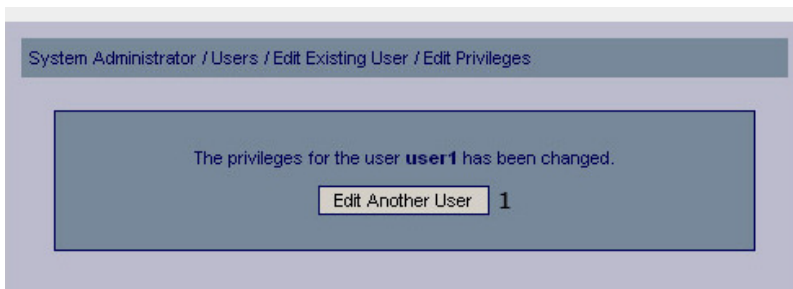
Controls

1. btnConfirmPrivileges – Invokes the savePrivileges method.

Methods

savePrivileges – Saves the new privileges in the database.

Edit User Privileges Confirmation



Functional Requirements

13.1

Controls

1. btnEditAnotherUser – Redirects the user to the existing users page.

Methods

None

Confirm User Removal



Functional Requirements

13.4

Controls

1. btnConfirm – Invokes the removeUser method.

Methods

removeUser – Removes the user and sends the user a confirmation.

User Removal Confirmation



Functional Requirements

13.4

Controls

1. btnRemoveAnotherUser – Redirects the user to the existing users page.

Methods

None

5. Design Details

5.1 Class Responsibility Collaborator (CRC) Cards

Activity	
Responsibilities	Collaborators
Knows course it belongs to Knows title Knows description Knows start day Knows start month Knows start year Knows start hour Knows start minute Knows end day Knows end month Knows end year Knows end hour Knows end minute	ActivityController

Validate data	
---------------	--

ActivityController	
Responsibilities	Collaborators
Add activity Remove activity Update activity Fetching an activity from the database	Activity

Assignment	
Responsibilities	Collaborators
Knows course it belongs to Knows title Knows description Knows deadline Validate data	AssignmentController

AssignmentController	
Responsibilities	Collaborators
Add assignment Remove assignment Update assignment Fetching an assignment from the database	Assignment

BaseObject	
Responsibilities	Collaborators
Defines common methods for all business objects	

BaseController	
Responsibilities	Collaborators
Defines common methods for all controllers	

Cache	
Responsibilities	Collaborators
Cache object (save to primary memory) Retrieve cached object Check if valid cache exists Manage cache item timeout	

Course	
Responsibilities	Collaborators
Knows course name Knows course code Knows news Knows deadlines Knows results Knows assignment Knows information pages Knows users Knows files Validate data	CourseController

CourseController	
Responsibilities	Collaborators
Add course Update course description Register a user for a course Unregister a user from a course Apply a user for a course Fetching a course from the database	Course User

Deadline	
Responsibilities	Collaborators
Knows course it belongs to Knows title Knows description Knows day Knows month Knows year Knows hour Knows minute Validate data	DeadlineController

DeadlineController	
Responsibilities	Collaborators
Add deadline Remove deadline Update deadline Generate a list of all deadlines for a course Generate a list of all deadlines for a user Fetching a deadline from the database	Course Deadline User

File	
Responsibilities	Collaborators
Knows course it belongs to Knows title Knows description Knows filename Validate data	FileController

FileController	
Responsibilities	Collaborators
Add file metadata to database Remove file metadata from database Update file metadata in database Save file to file system Remove file from file system Update file in file system Fetching a file from the database	File

Information Page	
Responsibilities	Collaborators
Knows course it belongs to Knows title Knows content Validate data	InformationPageController

InformationPageController	
Responsibilities	Collaborators
Add information page Remove information page Update information page Fetching an information page from the database	InformationPage

News	
Responsibilities	Collaborators
Knows course it belongs to Knows headline Knows content Validate data	NewsController

NewsController	
Responsibilities	Collaborators
Add news Remove news Update news Generate a list of all news for a course Generate a list of all news for a user Fetching a news post from the database	Course News User

Result	
Responsibilities	Collaborators
Knows the result associated to a assignment and user Validate data Fetching a result from the database	ResultController

ResultController	
Responsibilities	Collaborators
Add result Remove result Update result	Assignment Result User

Session	
Responsibilities	Collaborators
Knows logged-in status Creates user-session Ends user-session	User Privilege

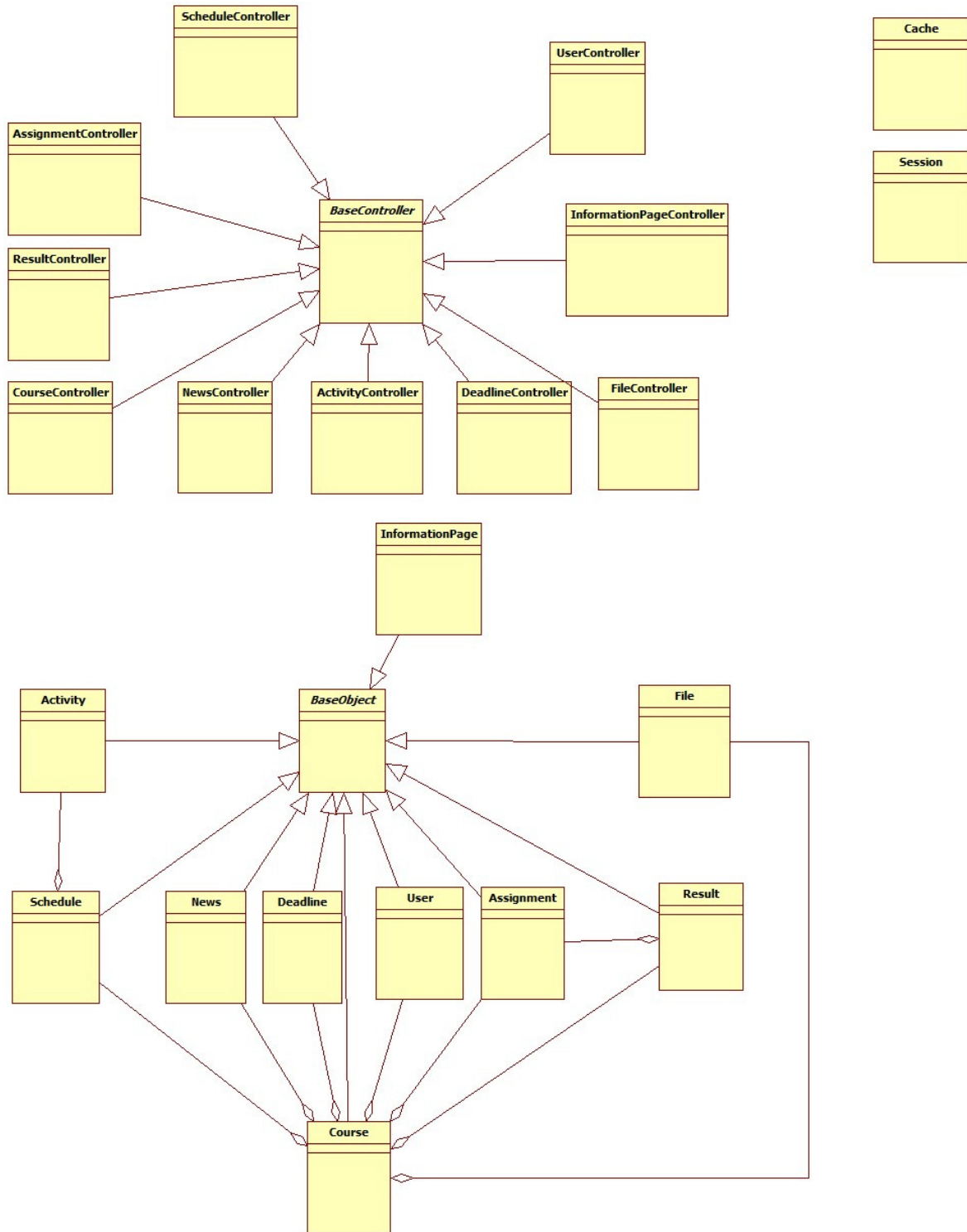
Schedule	
Responsibilities	Collaborators
Knows all activities related to a course	ScheduleController

ScheduleController	
Responsibilities	Collaborators
Generate a list of all activities for a course Generate a list of all activities for a user Export a list of activities to iCalendar format Import activities from the iCalendar format Remove all activities for a course	Activity Course Schedule User

User	
Responsibilities	Collaborators
Knows username Knows password Knows firstname Knows lastname Knows privileges Validate data	Course UserController

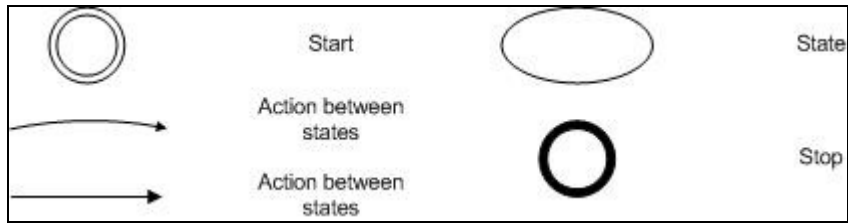
UserController	
Responsibilities	Collaborators
Add user Update user password Remove user Fetching a user from the database Add system administrator privilege for a user Remove system administrator privilege for a user Add course leader privilege for a user Remove course leader privilege for a user Add course assistant privilege for a user Remove course assistant privilege for a user	Course User

5.2 Class Diagram

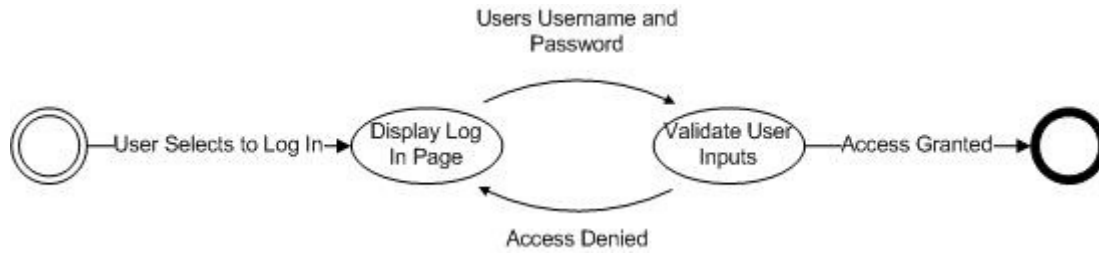


5.3 State Charts

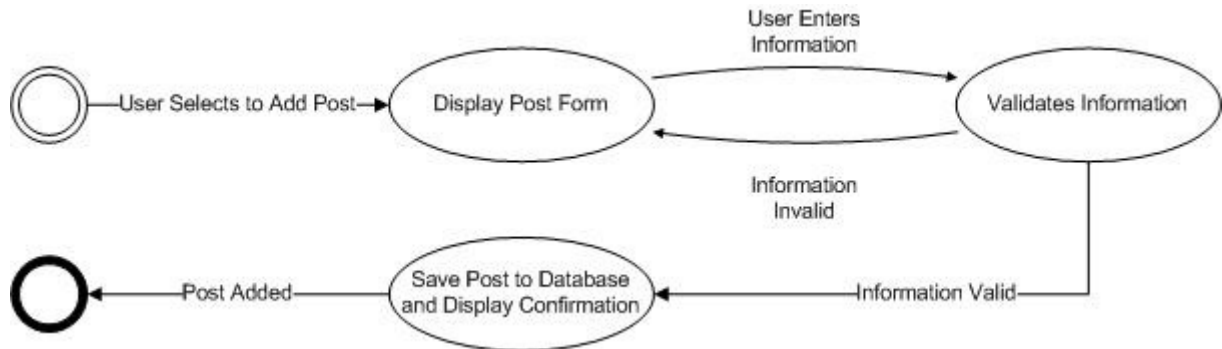
Description of the different components in the State Charts are described in this box.



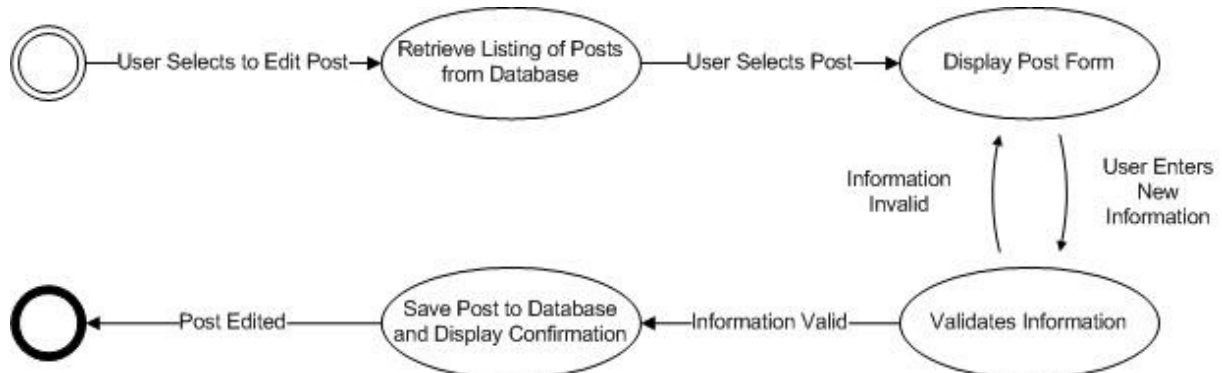
Log In



Add Post

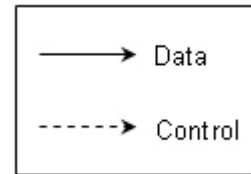


Edit Post



5.4 Interaction Diagrams

The sequence diagram models the flow of logic within the system where a horizontal arrow represents the interaction between two objects. The dotted vertical lines represents the time, where the time flows from top to bottom. The solid lines represent data flow in the system, and the dashed lines represent a transfer of control.



We decided to create one sequence diagram, "Create Database Post", for the use cases add course description, add deadline, add course leader, add privileges, etc. because they have similar sequence of actions. The same goes for the sequence diagrams "Edit Database Post", "Delete Database Post" and the view sequence diagrams. There are two view sequence diagrams, one that describes the flow if a post can be retrieved from the cache and the other if the post can't be retrieved from the cache and has to get it from the database server.

Log In

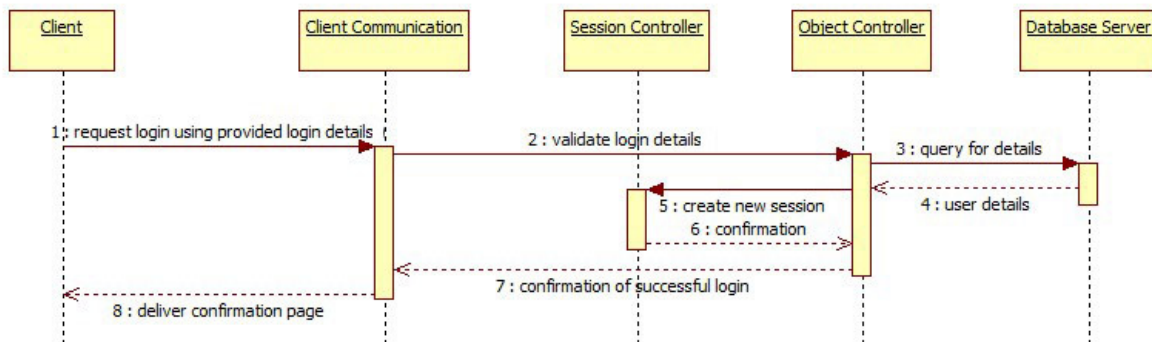


Figure 6 displays the sequence of action when a client requests to log in.

Create Database Post

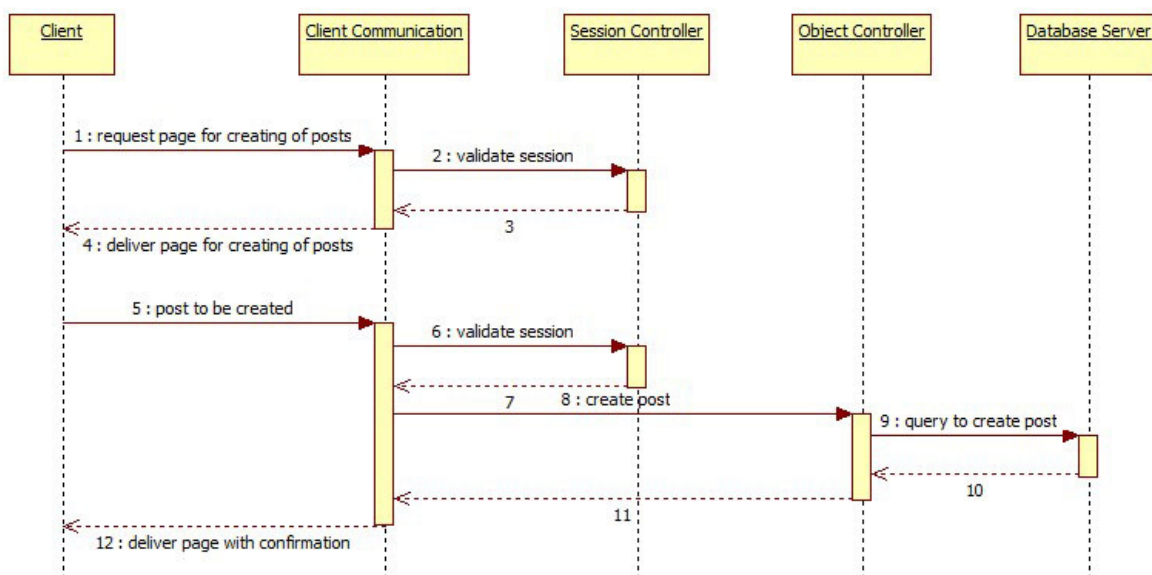


Figure 7 displays the sequence of actions to create a database post.

Edit Database Post

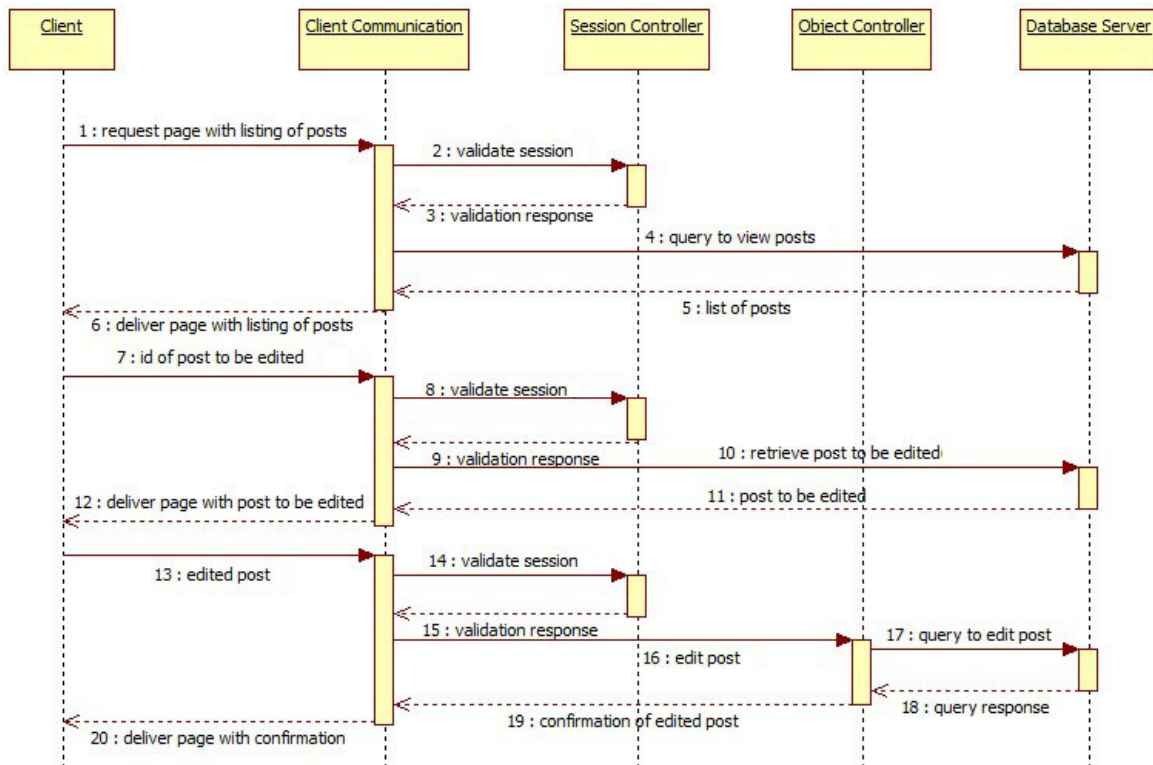


Figure 8 displays the sequence of actions to edit a database post.

Delete Database Post

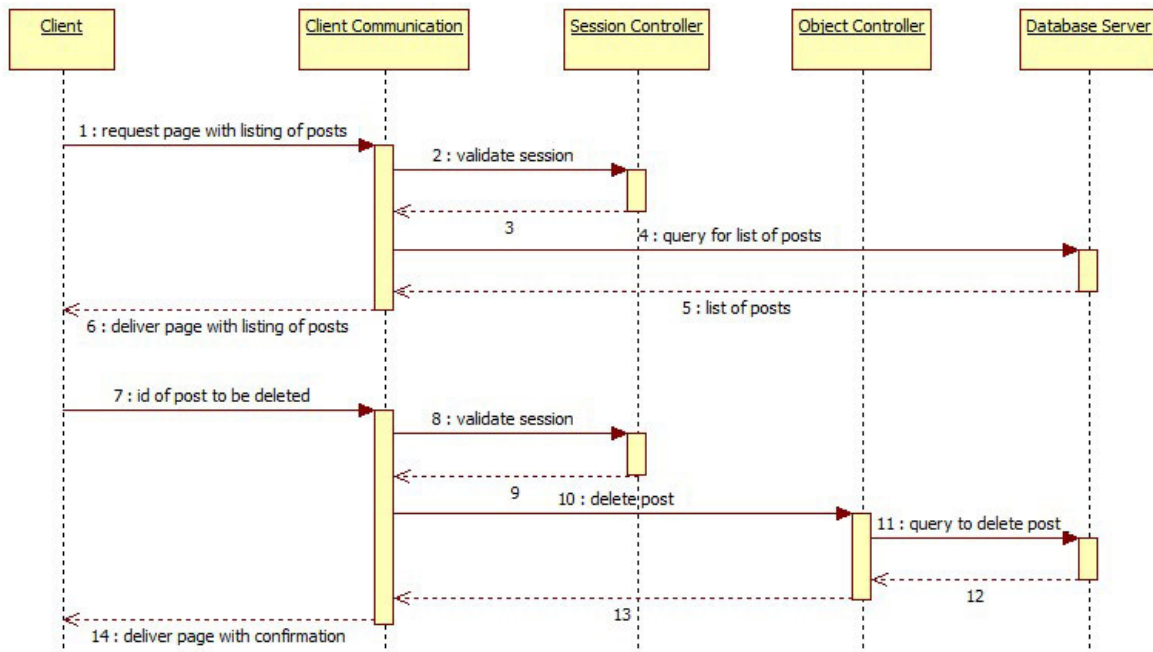


Figure 9 displays the sequence of actions to create a database post.

View Post from Cache

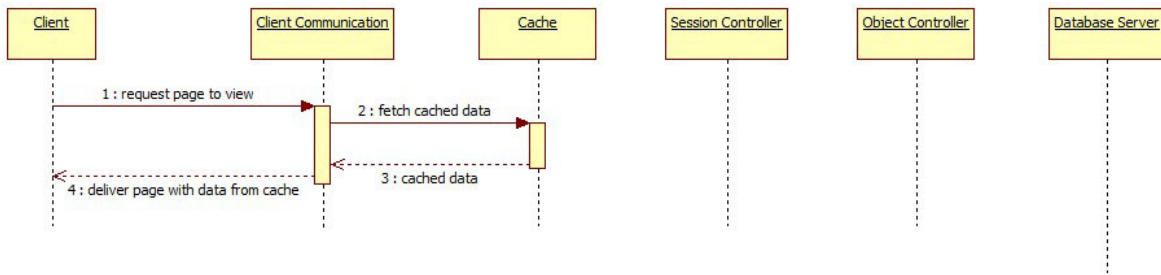


Figure 10 displays the sequence of actions to view a database post from cache.

View Post from Database

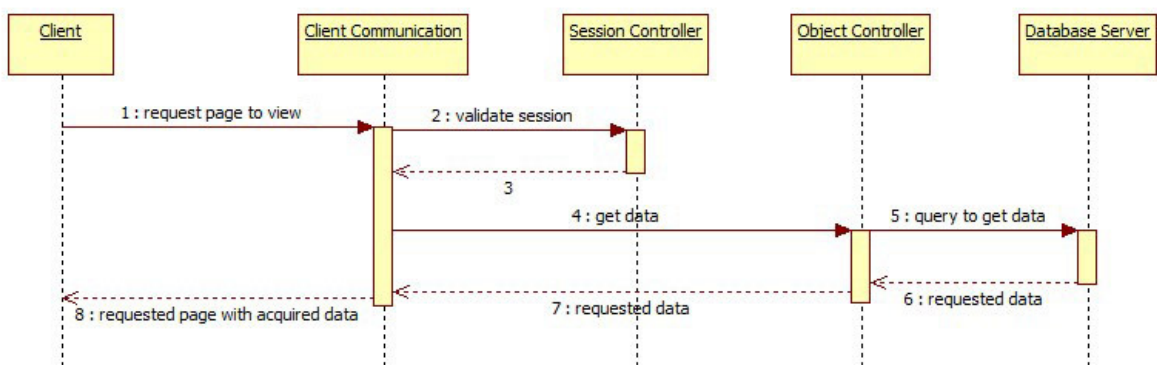


Figure 11 displays the sequence of actions to view a post from the database server when the post isn't available in the cache.

Export Schedule into iCalendar Format

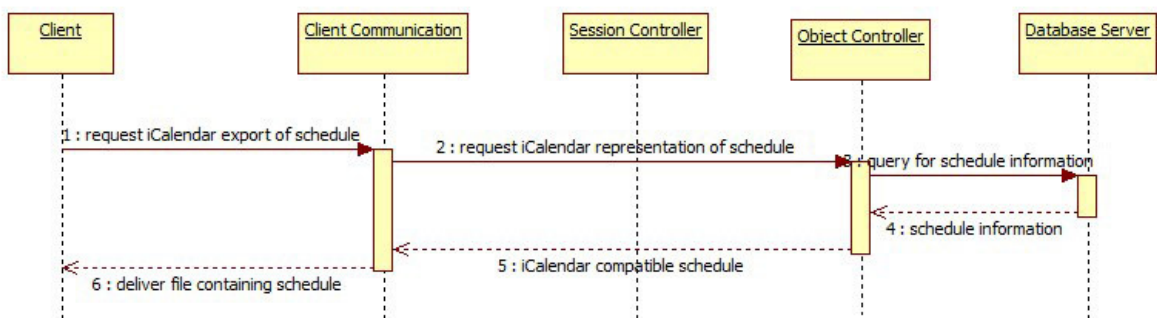


Figure 12 displays the sequence of actions to export a schedule into iCalendar format.

Upload File

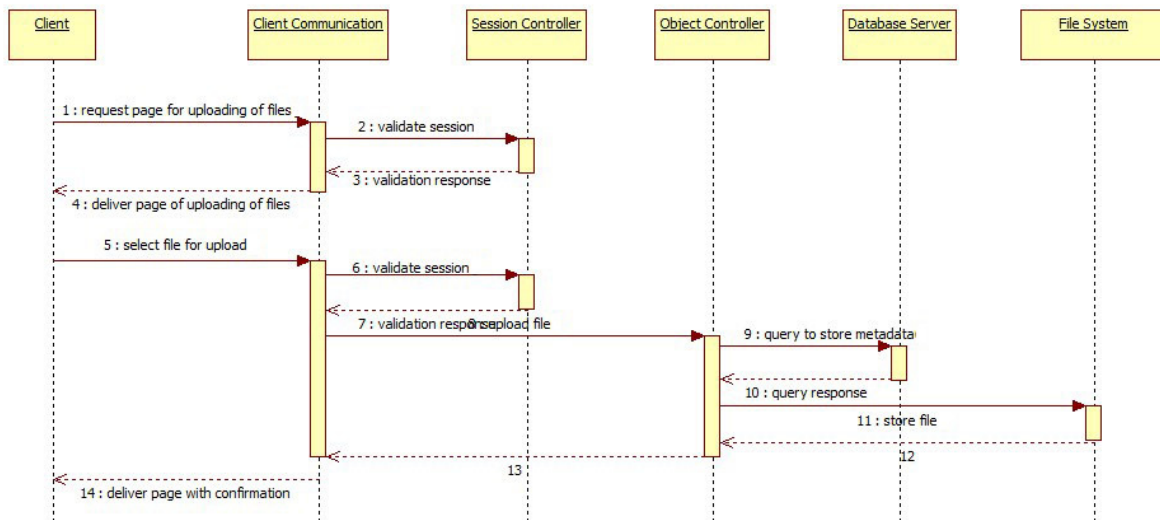


Figure 13 displays the sequence of actions when a client requests to upload a file.

Edit Uploaded File

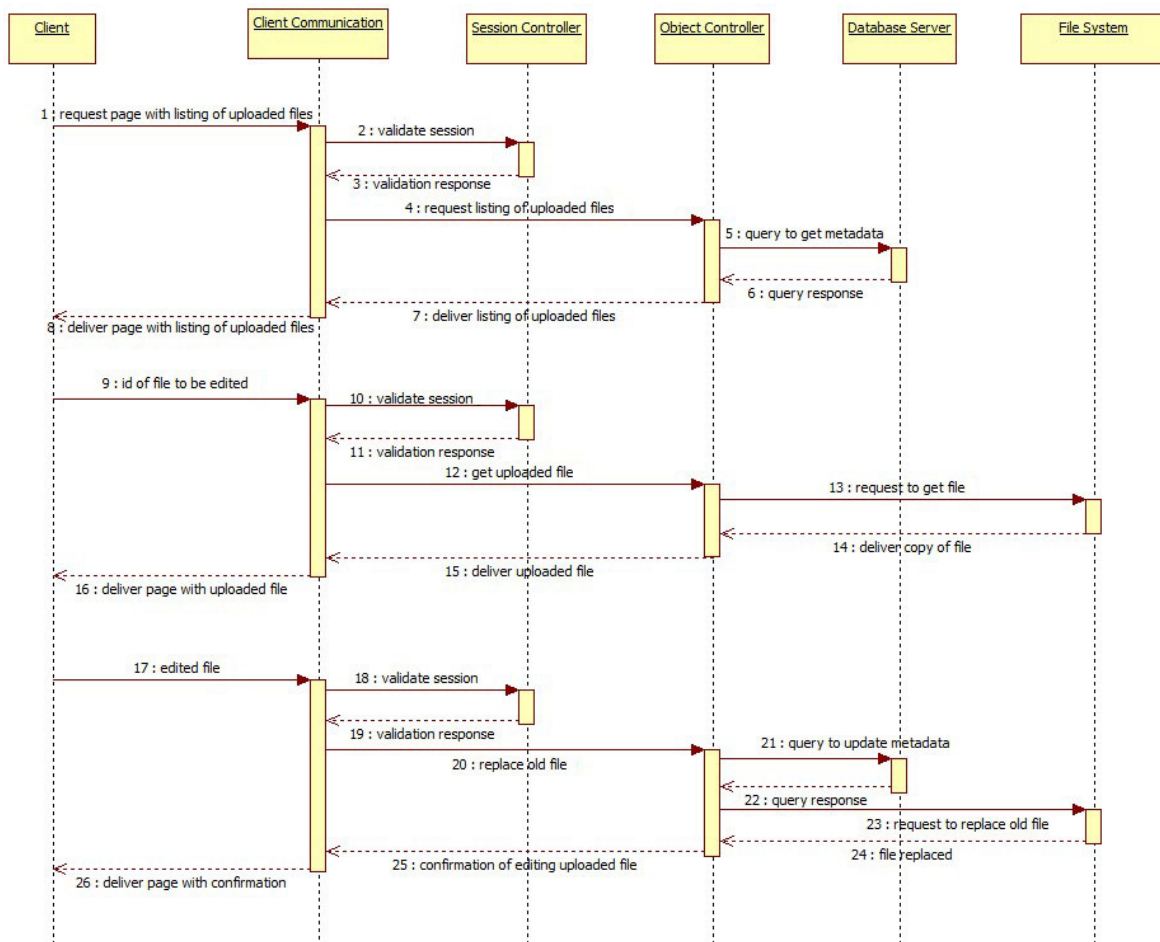


Figure 14 displays the sequence of actions when a client requests to edit an uploaded file.

Delete Uploaded File

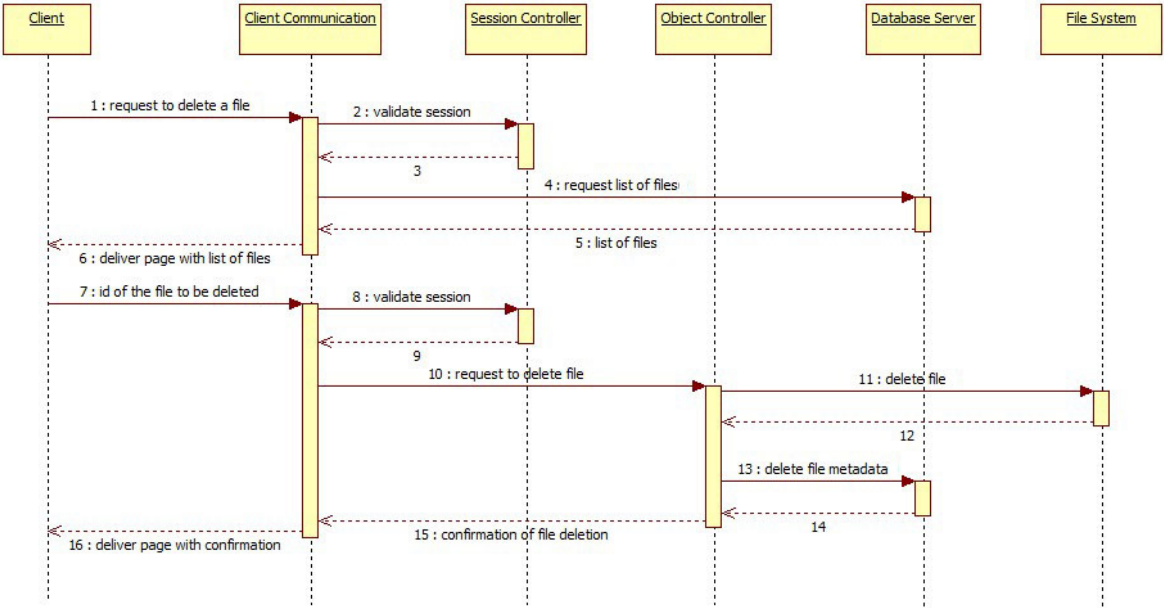
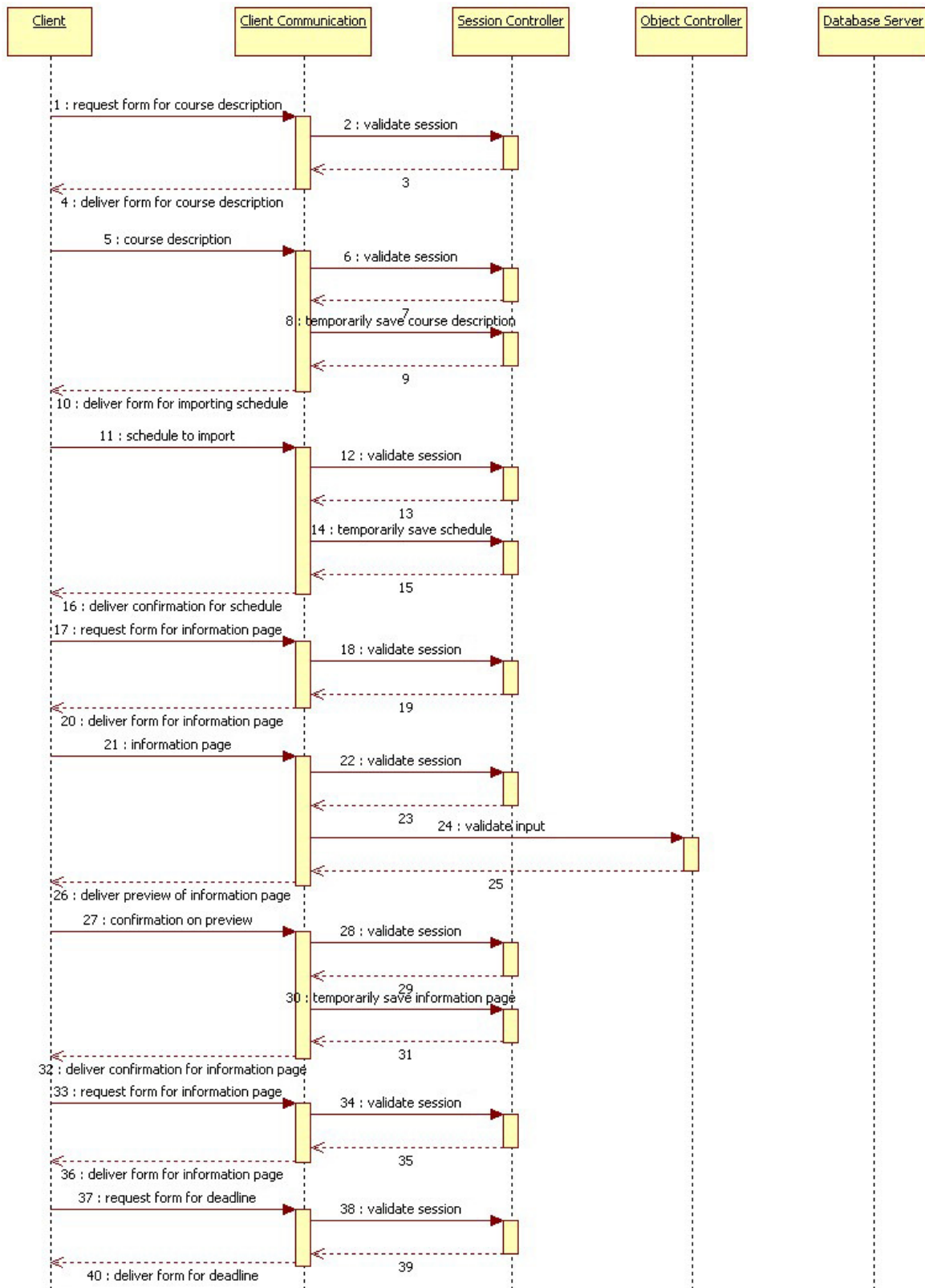


Figure 15 displays the sequence of actions when a client requests to delete an uploaded file.

Create Course Website



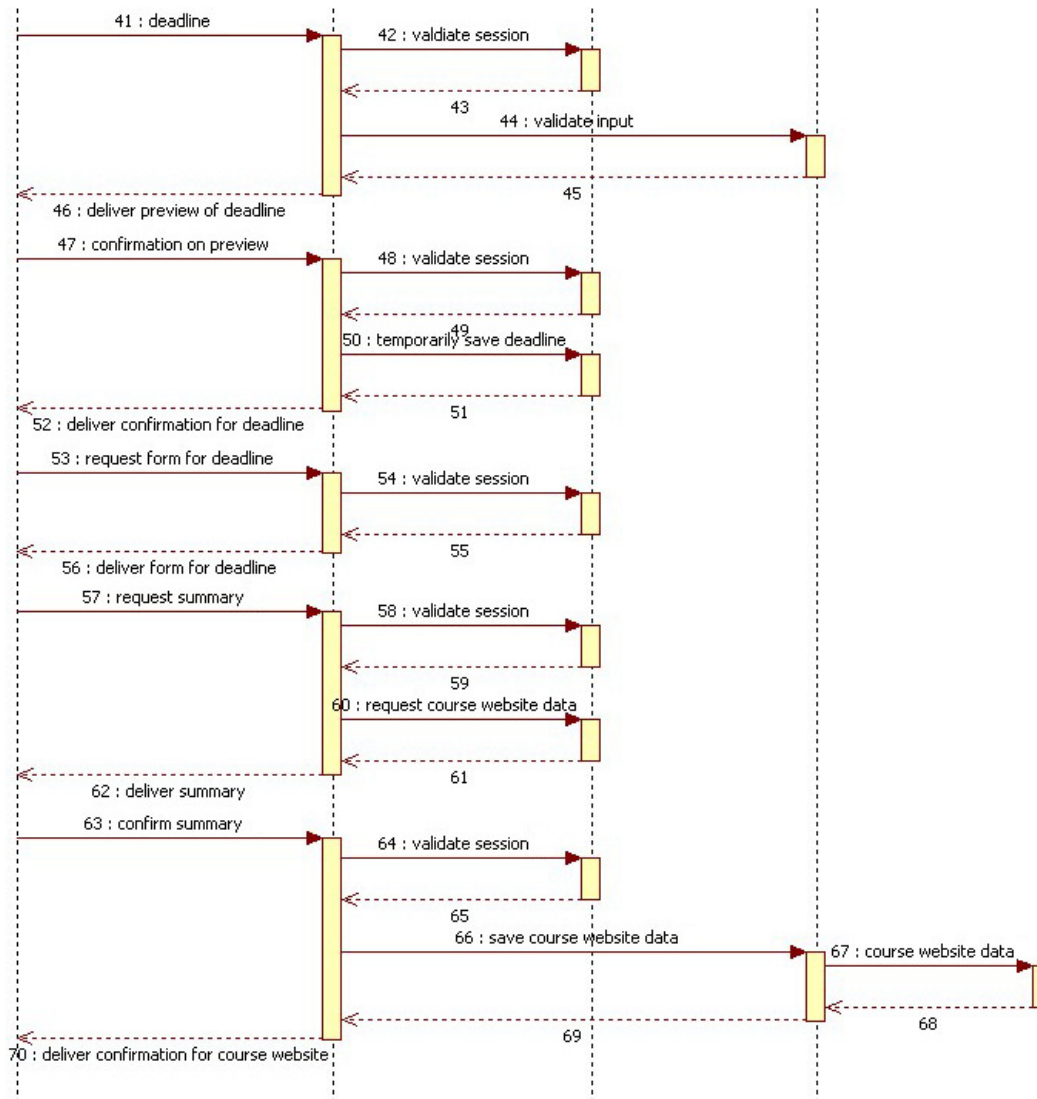


Figure 16 displays the sequence of actions when a client uses the guide for creating a course website.

5.5 Detailed Design

Database

Detailed database table definitions

Table: User

Columns:						
Key	Column name	Data type	Not null	Unique	Index	Extra
Primary	userID	integer	X	X	X	auto_increment
	username	varchar(50)	X	X		
	password	varchar(64)	X			
	firstname	varchar(50)	X			
	lastname	varchar(50)	X			

Table: Course**Columns:**

Key	Column name	Data type	Not null	Unique	Index	Extra
Primary	courseID	integer	X	X	X	auto_increment
	courseCode	varchar(20)	X	X		
Foreign	courseLeader	integer				
	courseName	varchar(50)	X			
	startYear	integer				
	startPeriod	smallint				
	endYear	integer				
	endPeriod	smallint				
	credits	float				
	description	text	X			

Table: Activity**Columns:**

Key	Column name	Data type	Not null	Unique	Index	Extra
Primary	activityID	integer	X	X	X	auto_increment
Foreign	courseID	integer	X	X	X	
	title	varchar(100)	X			
	description	text	X			
	startTime	DateTime	X			
	endTime	DateTime	X			

Table: Deadline**Columns:**

Key	Column name	Data type	Not null	Unique	Index	Extra
Primary	deadlineID	integer	X	X	X	auto_increment
Foreign	courseID	integer	X	X	X	
	headline	varchar(100)	X			
	content	text	X			
	time	DateTime	X			

Table: News**Columns:**

Key	Column name	Data type	Not null	Unique	Index	Extra
Primary	newsID	integer	X	X	X	auto_increment
Foreign	courseID	integer	X	X	X	
Foreign	author	integer	X			
	headline	varchar(100)	X			
	content	text	X			
	time	DateTime	X			

Table: File**Columns:**

Key	Column name	Data type	Not null	Unique	Index	Extra
Primary	fileID	integer	X	X	X	auto_increment
Foreign	courseID	integer	X	X	X	
	title	varchar(100)	X			
	description	text	X			
	filename	varchar(50)	X			

Table: Information Page**Columns:**

Key	Column name	Data type	Not null	Unique	Index	Extra
Primary	pageID	integer	X	X	X	auto_increment
Foreign	courseID	integer	X	X	X	
	title	varchar(100)	X			
	content	text	X			

Table: Assignment**Columns:**

Key	Column name	Data type	Not null	Unique	Index	Extra
Primary	assignmentID	integer	X	X	X	auto_increment
Foreign	courseID	integer	X	X	X	
	title	varchar(100)	X			
	description	text	X			

Table: InCourse**Columns:**

Key	Column name	Data type	Not null	Unique	Index	Extra
Primary	userID	integer	X	X	X	
Primary	courseID	integer	X	X	X	
	status	enum	X			

Table: Result**Columns:**

Key	Column name	Data type	Not null	Unique	Index	Extra
Primary	userID	integer	X	X	X	
Primary	assignmentID	integer	X	X	X	
	grade	enum	X			

Table: Privilege**Columns:**

Key	Column name	Data type	Not null	Unique	Index	Extra
Primary	userID	integer	X	X	X	
	privilege	enum	X			

Classes

Class Activity

Field activityID

The ID that uniquely identifies the activity.

Type: Integer
Access level: Private

Field belongToCourse

The ID of the course that the activity belongs to.

Type: Integer
Access level: Private

Field description

The description of the activity.

Type: String
Access level: Private

Field endDateTime

The date and time that the activity ends.

Type: Date
Access level: Private

Field startDateTime

The date and time that the activity starts.

Type: Date
Access level: Private

Field title

The title of the activity.

Type: String
Access level: Private

Method getActivityID

Retrieves the value of the activityID field.

Requirements: 7.2-7.4
Parameters: N/A
Return: Integer (ID of the activity)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A

Post-conditions: N/A
Caller: - ActivityController::Update
Calls: N/A

Method getBelongToCourse

Retrieves the value of the belongToCourse field.

Requirements: 7.2
Parameters: N/A
Return: Integer (ID of the course that the activity belongs to)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - ActivityController::update
Calls: N/A

Method getDescription

Retrieves the value of the description field.

Requirements: 7.2
Parameters: N/A
Return: String (description of the activity)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - ActivityController::update
Calls: N/A

Method getEndTime

Retrieves the value of the endTime field.

Requirements: 7.2
Parameters: N/A
Return: Date (Date and time for the end of the activity)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - ActivityController::update
- Activity::setStartDateTime
Calls: N/A

Method getStartDateTime

Retrieves the value of the startDateTime field.

Requirements: 7.2
Parameters: N/A
Return: Date (Date and time for the start of the activity)
Data access: N/A

Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - ActivityController::update
- Activity::setEndDateTime
Calls: N/A

Method getTitle

Retrieves the value of the title field.

Requirements: 7.2
Parameters: N/A
Return: String (title of the activity)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - ActivityController::update
Calls: N/A

Method setBelongToCourse

Stores an integer in the belongToCourse field.

Requirements: 7.2
Parameters: - Integer (ID of the course that the activity belongs to)
Return: Boolean (true if the ID was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The course ID is an integer and identifies a course that exists
Post-conditions: N/A
Caller: - ActivityController::add
- ActivityController::update
Calls: N/A

Method setDescription

Stores a String in the description field.

Requirements: 7.2
Parameters: - String (description of the activity)
Return: Boolean (true if the description was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The description is a string no longer than 100 characters
Post-conditions: N/A
Caller: - ActivityController::add
- ActivityController::update
Calls: N/A

Method setEndTime

Stores a Date object in the endTime field.

Requirements:	7.2
Parameters:	- Timestamp (Date and time for the end of the activity)
Return:	Boolean (true if the date and time of an activity was successfully stored and validated)
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	- The endTime is a Date object, representing date and time where the year has to be at least 1900, and if startTime is defined then the endTime has to occur after startTime.
Post-conditions:	N/A
Caller:	- ActivityController::add - ActivityController::update - Activity::setStartTime
Calls:	- Activity::setStartTime

Method setStartTime

Stores a Date object in the startTime field.

Requirements:	7.2
Parameters:	- Date (Date and time for the start of the activity)
Return:	Boolean (true if the date and time of an activity was successfully stored and validated)
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	- The startTime is a Date object, representing date and time where the year has to be at least 1900, and if endTime is defined and startTime is defined to occur after endTime, the endTime is changed into the same as startTime.
Post-conditions:	N/A
Caller:	- ActivityController::add - ActivityController::update
Calls:	- Activity::getEndTime - Activity::setEndTime

Method setTitle

Stores a String in the title field.

Requirements:	7.2
Parameters:	- String (title of the activity)
Return:	Boolean (true if the title was successfully stored and validated)
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	- The title is a string no longer than 50 characters
Post-conditions:	N/A
Caller:	- ActivityController::add

Calls: - ActivityController::update
N/A

Class ActivityController

Method add

The method stores an activity in the database.

Requirements: 7.2
Parameters: - Activity (the activity to be stored)
Return: Boolean (true if activity was successfully stored in the database)
Data access: Inserts a row in the database table *Activity*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: - One new row is inserted into the *Activity* table in the database
Caller: - JSP page for adding a scheduled activity
Calls: - Activity get methods

Method get

The method gets an activity from the database.

Requirements: 7.2-7.3
Parameters: - Integer (the activity ID to be fetched)
Return: Activity (the activity corresponding to the activity ID)
Data access: Fetches a row in the database table *Activity*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for editing or viewing a scheduled activity
Calls: - Activity set methods

Method remove

The method removes an activity from the database.

Requirements: 7.2
Parameters: - Activity (the activity to be removed)
Return: Boolean (true if activity was successfully removed from the database)
Data access: Deletes a row in the database table *Activity*
Pre-conditions: - A connection to the database is established
- The activity exists in the database
Validity Check: N/A
Post-conditions: - The activity's row is removed from the *Activity* table in the database
Caller: - JSP page for removing a scheduled activity
Calls: N/A

Method update

The method updates an activity in the database.

Requirements: 7.2
Parameters: - Activity (the activity to be updated)
Return: Boolean (true if activity was successfully updated in the database)
Data access: Update a row in the database table *Activity*
Pre-conditions: - A connection to the database is established
- The activity exists in the database
Validity Check: N/A
Post-conditions: - One row is updated in the *Activity* table in the database
Caller: - JSP page for updating a scheduled activity
Calls: - Activity get methods

Class Assignment

Field assignmentID

The ID that uniquely identifies the assignment.

Type: Integer
Access level: Private

Field belongToCourse

The ID of the course that the assignment belongs to.

Type: Integer
Access level: Private

Field deadline

The deadline of the assignment.

Type: Deadline
Access level: Private

Field description

The description of the assignment.

Type: String
Access level: Private

Field title

The title of the assignment.

Type: String
Access level: Private

Method getAssignmentID

Retrieves the value of the assignmentID field.

Requirements: 10.1
Parameters: N/A
Return: Integer (ID of the assignment)

Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - AssignmentController::update
Calls: N/A

Method getBelongToCourse

Retrieves the value of the belongToCourse field.

Requirements: 10.1
Parameters: N/A
Return: String (name of the course that the activity belongs to)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - AssignmentController::update
Calls: N/A

Method getDeadline

Retrieves the Deadline object in the deadline field.

Requirements: 10.1
Parameters: N/A
Return: Deadline (the deadline of the activity)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - AssignmentController::update
Calls: N/A

Method getDescription

Retrieves the value of the description field.

Requirements: 10.1
Parameters: N/A
Return: String (description of the assignment)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - AssignmentController::update
Calls: N/A

Method getTitle

Retrieves the value of the title field.

Requirements: 10.1
Parameters: N/A

Return: String (title of the assignment)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - AssignmentController::update
Calls: N/A

Method setBelongToCourse

Stores an integer in the belongToCourse field.

Requirements: 7.2
Parameters: - Integer (ID of the course that the assignment belongs to)
Return: Boolean (true if the ID was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The course ID is an integer and identifies a course that exists
Post-conditions: N/A
Caller: - AssignmentController::add
- AssignmentController::update
Calls: N/A

Method setDeadline

Stores a Deadline object in the deadline field.

Requirements: 10.1
Parameters: - Deadline (the deadline of the activity)
Return: Boolean (true if the deadline was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The deadline is a Deadline object.
Post-conditions: N/A
Caller: - AssignmentController::update
Calls: N/A

Method setDescription

Stores a String in the description field.

Requirements: 7.2
Parameters: - String (description of the assignment)
Return: Boolean (true if the description was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The description is a string no longer than 100 characters
Post-conditions: N/A
Caller: - AssignmentController::add
- AssignmentController::update

Calls: N/A

Method setTitle

Stores a String in the title field.

Requirements: 7.2

Parameters: - String (title of the assignment)

Return: Boolean (true if the title was successfully stored and validated)

Data access: N/A

Pre-conditions: N/A

Validity Check: - The title is a string no longer than 50 characters

Post-conditions: N/A

Caller: - AssignmentController::add
- AssignmentController::update

Calls: N/A

Class AssignmentController

Method add

The method stores an assignment in the database.

Requirements: 10.1

Parameters: - Assignment (the assignment to be stored)

Return: Boolean (true if assignment was successfully stored in the database)

Data access: Inserts a row in the database table *Assignment*

Pre-conditions: - A connection to the database is established

Validity Check: N/A

Post-conditions: - One new row is inserted into the *Assignment* table in the database

Caller: - JSP page for adding an assignment

Calls: - Assignment get methods

Method get

The method gets an assignment from the database.

Requirements: 10.1-10.2

Parameters: - Integer (the assignment ID to be fetched)

Return: Assignment (the assignment corresponding to the assignment ID)

Data access: Fetches a row in the database table *Assignment*

Pre-conditions: - A connection to the database is established

Validity Check: N/A

Post-conditions: N/A

Caller: - JSP page for editing an assignment

Calls: - Assignment set methods

Method getAssignmentByCourse

The method generates a list of all assignments for a specific course.

Requirements: 10.1
Parameters: - Course (the course for which assignments shall be retrieved)
Return: ArrayList (containing all assignments for the course)
Data access: Retrieve rows from the database table *Assignment*
Pre-conditions: - A connection to the database is established
- The course exists in the database
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for displaying assignments for a course
- JSP page for displaying a list of existing assignments
Calls: - Assignment set methods

Method remove

The method removes an assignment from the database. It also removes all results associated to the assignment.

Requirements: 10.1
Parameters: - Assignment (the assignment to be removed)
Return: Boolean (true if assignment was successfully removed from the database)
Data access: Deletes a row in the database table *Assignment* and all rows in the database table *Result* that are associated to the deleted row in *Assignment*
Pre-conditions: - A connection to the database is established
- The assignment exists in the database
Validity Check: N/A
Post-conditions: - The assignment's row is removed from the *Assignment* table in the database and all rows in *Result* that were associated to that row has been removed
Caller: - JSP page for removing an assignment
Calls: N/A

Method update

The method updates an assignment in the database.

Requirements: 10.1
Parameters: - Assignment (the assignment to be updated)
Return: Boolean (true if assignment was successfully updated in the database)
Data access: Update a row in the database table *Assignment*
Pre-conditions: - A connection to the database is established
- The assignment exists in the database
Validity Check: N/A
Post-conditions: - One row is updated in the *Assignment* table in the database
Caller: - JSP page for updating an assignment
Calls: - Assignment get methods

Class Cache

Field cache

Stores the cached objects and the keys needed to retrieve them.

Type: HashMap<Object, BaseObject>

Access level: Private

Field timeout

The current cache timeout value.

Type: Integer

Access level: Private

Method add

Adds an object to the object cache for later retrieval.

Requirements: 14.1-14.3

Parameters: - Object (The key used to later retrieve the object)
- BaseObject (the object to cache)

Return: void

Data access: N/A

Pre-conditions: N/A

Validity Check: N/A

Post-conditions: - The object has been added to the object cache.

Caller: - Any page or object that loads BaseObject type objects.

Calls: N/A

Method get

Retrieves a previously cached object.

Requirements: 14.1-14.3

Parameters: - Object (The key to which the object is mapped)

Return: Object (Object mapped to the key, or null if no object found)

Data access: N/A

Pre-conditions: N/A

Validity Check: N/A

Post-conditions: - An object reference is returned, or null if there was no object found.

Caller: - Any page or object that loads a BaseObject type object.

Calls: N/A

Method getCacheTimeout

Gets the amount of time before cached objects become invalid.

Requirements: 14.1-14.3

Parameters: N/A

Return: Integer (representing the amount of minutes before cache items are considered invalid)

Data access: N/A

Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: N/A
Calls: N/A

Method remove

Removes an object from the object cache.

Requirements: 14.1-14.3
Parameters: - Key
Return: Boolean (true if the item was found and removed from the cache)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: - The object mapped to the specified key can no longer be retrieved.
Caller: - Any controller object.
Calls: N/A

Method setCacheTimeout

Sets the amount of minutes before cached items are considered invalid.

Requirements: 14.1-14.3
Parameters: - Integer (representing the amount of minutes before cache items are considered invalid)
Return: void
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: - The cache timeout time has been updated.
Caller: N/A
Calls: N/A

Class Course

Field assignments

A collection of Assignment objects representing assignments of this course.

Type: ArrayList<Assignment>
Access level: Private

Field courseCode

The course code of the course.

Type: String
Access level: Private

Field courseID

The ID that uniquely identifies the course.

Type: Integer
Access level: Private

Field deadlines

A collection of Deadline objects representing deadlines of this course.

Type: ArrayList<DeadLine>
Access level: Private

Field files

A collection of File objects representing files of this course.

Type: ArrayList<File>
Access level: Private

Field informationPages

A collection of InformationPage objects representing information pages of this course.

Type: ArrayList<InformationPage>
Access level: Private

Field news

A collection of News objects representing news of this course.

Type: ArrayList<News>
Access level: Private

Field results

A collection of Result objects representing results of this course.

Type: ArrayList<Result>
Access level: Private

Field users

A collection of User objects representing the students registered for this course.

Type: ArrayList<User>
Access level: Private

Method addAssignment

Stores an Assignment object in the assignments ArrayList.

Requirements: 10.1
Parameters: - Assignment (the assignment that's added to the course)
Return: Boolean (true if the Assignment object is successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The input has to be an Assignment object.

Post-conditions: N/A
Caller: N/A
Calls: N/A

Method addDeadline

Stores a deadline object in the deadlines ArrayList.

Requirements: 8.1
Parameters: - Deadline (the deadline that's added to the course)
Return: Boolean (true if the Deadline object is successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The input has to be a Deadline object.
Post-conditions: N/A
Caller: N/A
Calls: N/A

Method addFile

Stores a File object in the files ArrayList.

Requirements: 9.1
Parameters: - File (the file that's added to the course)
Return: Boolean (true if the Files object is successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The input has to be a File object.
Post-conditions: N/A
Caller: N/A
Calls: N/A

Method addInformationPage

Stores an InformationPage object in the InformationPage ArrayList.

Requirements: 6.1
Parameters: - InformationPage (the InformationPage that's added to the course)
Return: Boolean (true if the InformationPage object is successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The input has to be an InformationPage object.
Post-conditions: N/A
Caller: N/A
Calls: N/A

Method addNews

Stores a News object in the news ArrayList.

Requirements: 5.1

Parameters: - News (the news that's added to the course)
Return: Boolean (true if the News object is successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The input has to be a News object.
Post-conditions: N/A
Caller: N/A
Calls: N/A

Method addResult

Stores a Result object in the results ArrayList.

Requirements: 11.1
Parameters: - Result (the result that's added to the course)
Return: Boolean (true if the Result object is successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The input has to be a Result object.
Post-conditions: N/A
Caller: N/A
Calls: N/A

Method addUser

Stores a User object in the users ArrayList. Register a student for the course.

Requirements: 12.2
Parameters: - User (the student that's registered for the course)
Return: Boolean (true if the User object is successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The input is a User object. The user isn't already registered for the course.
Post-conditions: N/A
Caller: N/A
Calls: N/A

Method getAllAssignments

Retrieves all assignments for this course.

Requirements: 10.1-10.2
Parameters: N/A
Return: ArrayList<Deadline> (all deadlines for this course)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: N/A
Calls: N/A

Method getAllDeadlines

Retrieves all deadlines for this course.

Requirements:	8.1-8.3
Parameters:	N/A
Return:	ArrayList<Deadline> (all deadlines for this course)
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	N/A
Post-conditions:	N/A
Caller:	N/A
Calls:	N/A

Method getAllFiles

Retrieves all files for this course.

Requirements:	9.1-9.2
Parameters:	N/A
Return:	ArrayList<File> (all files for this course)
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	N/A
Post-conditions:	N/A
Caller:	N/A
Calls:	N/A

Method getAllInformationPages

Retrieves all information pages for this course.

Requirements:	6.1-6.2
Parameters:	N/A
Return:	ArrayList<InformationPages> (all information pages for this course)
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	N/A
Post-conditions:	N/A
Caller:	N/A
Calls:	N/A

Method getAllNews

Retrieves all news for this course.

Requirements:	5.1-5.2
Parameters:	N/A
Return:	ArrayList<News> (all the news for this course)
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	N/A
Post-conditions:	N/A
Caller:	N/A

Calls: N/A

Method getAllResults

Retrieves all results for this course.

Requirements: 11.1-11.2
Parameters: N/A
Return: ArrayList<Results> (all results for this course)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: N/A
Calls: N/A

Method getAllUsers

Retrieves all students that are registered for this course.

Requirements: 12.1
Parameters: N/A
Return: ArrayList<Users> (all students registered for this course)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: N/A
Calls: N/A

Method getCourseCode

Retrieves the value of the courseCode field.

Requirements: N/A
Parameters: N/A
Return: String (the course code for this course)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: N/A
Calls: N/A

Method getCourseID

Retrieves the value of the courseID field.

Requirements: 13.3
Parameters: N/A
Return: Integer (ID of the course)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A

Caller: - CourseController::update
Calls: N/A

Method setCourseCode

Updates the value of the course code field.

Requirements: 13.3
Parameters: - String (the course code for this course)
Return: N/A
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: N/A
Calls: N/A

Class CourseController

Method add

The method stores a course in the database.

Requirements: 13.3
Parameters: - Course (the course to be stored)
Return: Boolean (true if course was successfully stored in the database)
Data access: Inserts a row in the database table *Course*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: - One new row is inserted into the *Course* table in the database
Caller: - JSP page for adding a course
Calls: - Course get methods

Method GetDescription

The method gets a course description from the database.

Requirements: 4.1-4.2
Parameters: - Integer (the course ID of the course to be fetched)
Return: Course (the course corresponding to the course ID)
Data access: Fetches a row in the database table *Course*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for editing or viewing a course description
Calls: - Course set methods

Method getDescriptionByCourseCode

The method gets a course description from the database using a course code. The course code has to match exactly.

Requirements: 4.1

Parameters: - String (the course code of the course to be fetched)
Return: Course (the course corresponding to the course code)
Data access: Fetches a row in the database table *Course*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for editing user privileges for a course
Calls: - Course set methods

Method update

The method updates a course stored in the database.

Requirements: 13.3
Parameters: - Course (the course to be stored)
Return: Boolean (true if course was successfully stored in the database)
Data access: Update a row in the database table *Course*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: - One row is updated in the *Course* table in the database
Caller: - JSP page for editing a course
Calls: - Course get methods

Method updateDescription

The method updates a course with a course description in the database.

Requirements: 4.1, 13.3
Parameters: - Course (the course to be updated)
Return: Boolean (true if course description was successfully updated in the database)
Data access: Update a row in the database table *Course*
Pre-conditions: - A connection to the database is established
 - The course exists in the database
Validity Check: N/A
Post-conditions: - One row is updated in the *Course* table in the database
Caller: - JSP page for updating a course description
Calls: - Course get methods

Method Apply

The method applies a user for a course in the database. This is done by inserting a row in table *InCourse* with status field APPLYING.

Requirements: 12.3
Parameters: - Course (the course to which the user is applying for)
 - User (the user to apply)
Return: Boolean (true if the user was successfully applied for the course in the database)
Data access: Update a row in the database table *InCourse*
Pre-conditions: - A connection to the database is established
 - The course exists in the database

Validity Check: - The user exists in the database
 N/A
Post-conditions: - One row is inserted in the *InCourse* table in the database
Caller: - JSP page for applying for a course
Calls: N/A

Method register

The method registers a user for a course in the database. This is done by updating the status field in table *InCourse* to REGISTERED, from the previous state APPLYING.

Requirements: 12.2
Parameters: - Course (the course to which the user shall be registered for)
 - User (the user to register)
Return: Boolean (true if the user was successfully registered for the course in the database)
Data access: Update a row in the database table *InCourse*
Pre-conditions: - A connection to the database is established
 - The course exists in the database
 - The user exists in the database
 - The user has applied for the course
Validity Check: N/A
Post-conditions: - One row is updated in the *InCourse* table in the database
Caller: - JSP page for updating a course description
Calls: N/A

Method unregister

The method unregisters a user from a course in the database. This is done deleting the row in table *InCourse* corresponding to the given course and user.

Requirements: 12.4
Parameters: - Course (the course to which the user shall be unregistered from)
 - User (the user to unregister)
Return: Boolean (true if the user was successfully unregistered from the course in the database)
Data access: Delete a row in the database table *InCourse*
Pre-conditions: - A connection to the database is established
 - The course exists in the database
 - The user exists in the database
 - The user has been registered for the course
Validity Check: N/A
Post-conditions: - One row is deleted in the *InCourse* table in the database
Caller: - JSP page for updating a course description
Calls: N/A

Class Deadline

Field belongToCourse

The ID of the course that the deadline belongs to.

Type: Integer
Access level: Private

Field deadlineDateTime

The date and time that of the deadline.

Type: Date
Access level: Private

Field deadlineID

The ID that uniquely identifies the deadline.

Type: Integer
Access level: Private

Field description

The description of the deadline.

Type: String
Access level: Private

Field title

The title of the deadline.

Type: String
Access level: Private

Method getBelongToCourse

Retrieves the value of the belongToCourse field.

Requirements: 8.1-8.3
Parameters: N/A
Return: Integer (ID of the course that the deadline belongs to)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller:
- DeadlineController::update
- DeadlineController::getCourseDeadlines
- DeadlineController::getUserDeadlines
Calls: N/A

Method getDeadlineDateTime

Retrieves the value of the deadlineDateTime field.

Requirements: 8.1-8.3
Parameters: N/A
Return: Date (Date and time for when the deadline expire)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A

Post-conditions: N/A
Caller: - DeadlineController::update
- DeadlineController::getCourseDeadlines
- DeadlineController::getUserDeadlines
Calls: N/A

Method getDeadlineID

Retrieves the value of the deadlineID field.

Requirements: 8.1
Parameters: N/A
Return: Integer (ID of the deadline)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - DeadlineController::update
Calls: N/A

Method getDescription

Retrieves the value of the description field.

Requirements: 8.1-8.3
Parameters: N/A
Return: String (description of the deadline)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - DeadlineController::update
- DeadlineController::getCourseDeadlines
- DeadlineController::getUserDeadlines
Calls: N/A

Method getTitle

Retrieves the value of the title field.

Requirements: 8.1-8.3
Parameters: N/A
Return: String (title of the deadline)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - DeadlineController::update
- DeadlineController::getCourseDeadlines
- DeadlineController::getUserDeadlines
Calls: N/A

Method setBelongToCourse

Stores an integer in the belongToCourse field.

Requirements: 8.1-8.3
Parameters: - Integer (ID of the course that the deadline belongs to)
Return: Boolean (true if the ID was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The course ID is an integer and identifies a course that exists
Post-conditions: N/A
Caller: - DeadlineController::add
- DeadlineController::update
Calls: N/A

Method setDeadlineDateTime

Stores a Date object in the deadlineDateTime field.

Requirements: 8.1-8.3
Parameters: - Date (Date and time for when the deadline expire)
Return: Boolean (true if the date and time of a deadline was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The input has to be a Date object and the year has to be at least 1900.
Post-conditions: N/A
Caller: - DeadlineController::update
- DeadlineController::add
Calls: N/A

Method setDescription

Stores a String in the description field.

Requirements: 8.1-8.3
Parameters: - String (description of the deadline)
Return: Boolean (true if the description was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The description is a string no longer than 100 characters
Post-conditions: N/A
Caller: - DeadlineController::add
- DeadlineController::update
Calls: N/A

Method setTitle

Stores a String in the title field.

Requirements: 8.1-8.3
Parameters: - String (title of the deadline)

Return: Boolean (true if the title was successfully stored and validated)

Data access: N/A

Pre-conditions: N/A

Validity Check: - The title is a string no longer than 50 characters

Post-conditions: N/A

Caller: - DeadlineController::add
- DeadlineController::update

Calls: N/A

Class DeadlineController

Method add

The method stores a deadline in the database.

Requirements: 8.1

Parameters: - Deadline (the deadline to be stored)

Return: Boolean (true if deadline was successfully stored in the database)

Data access: Inserts a row in the database table *Deadline*

Pre-conditions: - A connection to the database is established

Validity Check: N/A

Post-conditions: - One new row is inserted into the *Deadline* table in the database

Caller: - JSP page for adding a deadline

Calls: - Deadline get methods

Method get

The method gets a deadline from the database.

Requirements: 8.1-8.2

Parameters: - Integer (the deadline ID of the deadline to be fetched)

Return: Deadline (the deadline corresponding to the deadline ID)

Data access: Fetches a row in the database table *Deadline*

Pre-conditions: - A connection to the database is established

Validity Check: N/A

Post-conditions: N/A

Caller: - JSP page for editing or viewing a deadline

Calls: - Deadline set methods

Method getDeadlinesByCourse

The method generates a list of all deadlines for a specific course.

Requirements: 2.1-2.2

Parameters: - Course (the course for which deadlines shall be retrieved)

Return: ArrayList (containing all deadlines for the course)

Data access: Retrieve rows from the database table *Deadline*

Pre-conditions: - A connection to the database is established
- The deadline exists in the database

Validity Check: N/A

Post-conditions: N/A

Caller: - JSP page for displaying deadlines for a course
Calls: - Deadline set methods

Method getDeadlinesByUser

The method generates a list of all deadlines for courses that a specific user is registered for.

Requirements: 8.3
Parameters: - User (the user for which deadlines shall be retrieved)
Return: ArrayList (containing all deadlines for the user's courses)
Data access: Retrieves row from the database table *Deadline*
Pre-conditions: - A connection to the database is established
- The user exists in the database
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for displaying deadlines for a user
Calls: - Deadline set methods

Method remove

The method removes a deadline from the database.

Requirements: 8.1
Parameters: - Deadline (the deadline to be removed)
Return: Boolean (true if deadline was successfully removed from the database)
Data access: Deletes a row in the database table *Deadline*
Pre-conditions: - A connection to the database is established
- The deadline exists in the database
Validity Check: N/A
Post-conditions: - The deadline's row is removed from the *Deadline* table in the database
Caller: - JSP page for removing a deadline
Calls: N/A

Method update

The method updates a deadline in the database.

Requirements: 8.1
Parameters: - Deadline (the deadline to be updated)
Return: Boolean (true if deadline was successfully updated in the database)
Data access: Update a row in the database table *Deadline*
Pre-conditions: - A connection to the database is established
- The deadline exists in the database
Validity Check: N/A
Post-conditions: - One row is updated in the *Deadline* table in the database
Caller: - JSP page for updating a deadline
Calls: - Deadline get methods

Class File

Field belongToCourse

The ID of the course that the file belongs to.

Type: Integer

Access level: Private

Field description

The description of the file.

Type: String

Access level: Private

Field fileID

The ID that uniquely identifies the file.

Type: Integer

Access level: Private

Field filename

The name of the file.

Type: String

Access level: Private

Field title

The title of the file.

Type: String

Access level: Private

Method getBelongToCourse

Retrieves the value of the belongToCourse field.

Requirements: 9.1-9.2

Parameters: N/A

Return: Integer (ID of the course that the file belongs to)

Data access: N/A

Pre-conditions: N/A

Validity Check: N/A

Post-conditions: N/A

Caller: - FileController::update

Calls: N/A

Method getDescription

Retrieves the value of the description field.

Requirements: 9.1-9.2

Parameters: N/A

Return: String (description of the file)

Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - FileController::update
Calls: N/A

Method getFileID

Retrieves the value of the fileID field.

Requirements: 9.1
Parameters: N/A
Return: Integer (ID of the file)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - FileController::update
Calls: N/A

Method getFilename

Retrieves the value of the filename field.

Requirements: 9.1-9.2
Parameters: N/A
Return: String (name of the file)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - FileController::update
Calls: N/A

Method getTitle

Retrieves the value of the title field.

Requirements: 9.1-9.2
Parameters: N/A
Return: String (title of the file)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - FileController::update
Calls: N/A

Method setBelongToCourse

Stores an integer in the belongToCourse field.

Requirements: 9.1-9.2
Parameters: - Integer (ID of the course that the file belongs to)

Return: Boolean (true if the ID was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The course ID is an integer and identifies a course that exists
Post-conditions: N/A
Caller: - FileController::add
- FileController::update
Calls: N/A

Method setDescription

Stores a String in the description field.

Requirements: 9.1-9.2
Parameters: - String (description of the file)
Return: Boolean (true if the description was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The description is a string no longer than 100 characters
Post-conditions: N/A
Caller: - FileController::add
- FileController::update
Calls: N/A

Method setFilename

Stores a String in the filename field.

Requirements: 9.1-9.2
Parameters: - String (name of the file)
Return: Boolean (true if the name was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The name is a string no longer than 50 characters
Post-conditions: N/A
Caller: - FileController::add
- FileController::update
Calls: N/A

Method setTitle

Stores a String in the title field.

Requirements: 9.1-9.2
Parameters: - String (title of the file)
Return: Boolean (true if the title was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The title is a string no longer than 50 characters

Post-conditions: N/A
Caller: - FileController::add
- FileController::update
Calls: N/A

Class FileController

Method add

The method stores a file on the file system and its metadata in the database.

Requirements: 9.1
Parameters: - File (the file object to be stored)
Return: Boolean (true if file was successfully stored in the database)
Data access: Inserts a row in the database table *File*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: - One new row is inserted into the *File* table in the database and a new file is saved on the file system
Caller: - JSP page for adding a file
Calls: - File get methods

Method get

The method gets a file from the database.

Requirements: 9.1
Parameters: - Integer (the file ID of the file to be fetched)
Return: File (the file corresponding to the file ID)
Data access: Fetches a row in the database table *File*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for editing or viewing a file
Calls: - File set methods

Method getFileByCourse

The method generates a list of all files for a specific course.

Requirements: 9.1
Parameters: - Course (the course for which files shall be retrieved)
Return: ArrayList (containing all files for the course)
Data access: Retrieve rows from the database table *File*
Pre-conditions: - A connection to the database is established
- The course exists in the database
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for displaying files for a course
- JSP page for displaying a list of existing files
Calls: - File set methods

Method remove

The method removes a file from the file system and its metadata from the database.

Requirements: 9.1
Parameters: - File (the file object to be removed)
Return: Boolean (true if file was successfully removed from the database)
Data access: Deletes a row in the database table *File* and a file from the file system
Pre-conditions: - A connection to the database is established
- The file exists on the file system and its metadata exists in the database
Validity Check: N/A
Post-conditions: - The file's row is removed from the *File* table in the database and the file is removed from the file system
Caller: - JSP page for removing a file
Calls: N/A

Method update

The method updates a file on the file system and its metadata in the database.

Requirements: 9.1
Parameters: - File (the file object to be updated)
Return: Boolean (true if file was successfully updated in the database)
Data access: Update a row in the database table *File*
Pre-conditions: - A connection to the database is established
- The file exists in the database
Validity Check: N/A
Post-conditions: - One row is updated in the *File* table in the database
Caller: - JSP page for updating a file
Calls: - File get methods

Class InformationPage

Field belongToCourse

The ID of the course that the information page belongs to.

Type: Integer
Access level: Private

Field content

The content of the information page.

Type: String
Access level: Private

Field pageID

The ID that uniquely identifies the information page.

Type: Integer
Access level: Private

Field title

The title of the information page.

Type: String
Access level: Private

Method getBelongToCourse

Retrieves the value of the belongToCourse field.

Requirements: 6.1-6.2
Parameters: N/A
Return: Integer (ID of the course that the information page belongs to)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - InformationPageController::update
Calls: N/A

Method getContent

Retrieves the value of the content field.

Requirements: 6.1-6.2
Parameters: N/A
Return: String (content of the information page)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - InformationPageController::update
Calls: N/A

Method getPagelD

Retrieves the value of the pagelD field.

Requirements: 6.1-6.2
Parameters: N/A
Return: Integer (ID of the information page)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - InformationPageController::update
Calls: N/A

Method getTitle

Retrieves the value of the title field.

Requirements: 6.1-6.2
Parameters: N/A

Return: String (title of the information page)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - InformationPageController::update
Calls: N/A

Method setBelongToCourse

Stores an integer in the belongToCourse field.

Requirements: 6.1-6.2
Parameters: - Integer (ID of the course that the information page belongs to)
Return: Boolean (true if the ID was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The course ID is an integer and identifies a course that exists
Post-conditions: N/A
Caller: - InformationPageController::add
- InformationPageController::update
Calls: N/A

Method setContent

Stores a String in the content field.

Requirements: 6.1-6.2
Parameters: - String (content of the information page)
Return: Boolean (true if the content was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The content is a string
Post-conditions: N/A
Caller: - FileController::add
- FileController::update
Calls: N/A

Method setTitle

Stores a String in the title field.

Requirements: 6.1-6.2
Parameters: - String (title of the information page)
Return: Boolean (true if the title was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The title is a string no longer than 50 characters
Post-conditions: N/A

Caller: - InformationPageController::add
- InformationPageController::update
Calls: N/A

Class InformationPageController

Method add

The method stores an information page in the database.

Requirements: 6.1
Parameters: - InformationPage (the information page to be stored)
Return: Boolean (true if information page was successfully stored in the database)
Data access: Inserts a row in the database table *InformationPage*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: - One new row is inserted into the *InformationPage* table in the database
Caller: - JSP page for adding an information page
Calls: - InformationPage get methods

Method Get

The method gets an information page from the database.

Requirements: 6.1-6.2
Parameters: - Integer (the information page ID of the information page to be fetched)
Return: InformationPage (the information page corresponding to the file ID)
Data access: Fetches a row in the database table *InformationPage*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for editing or viewing an information page
Calls: - InformationPage set methods

Method getInformationPageByCourse

The method generates a list of all information pages for a specific course.

Requirements: 6.1-6.2
Parameters: - Course (the course for which information pages shall be retrieved)
Return: ArrayList (containing all information pages for the course)
Data access: Retrieve rows from the database table *InformationPage*
Pre-conditions: - A connection to the database is established
- The course exists in the database
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for displaying a course website
- JSP page for displaying a list of existing information pages
Calls: - InformationPage set methods

Method remove

The method removes an information page from the database.

Requirements:	6.1
Parameters:	- InformationPage (the information page to be removed)
Return:	Boolean (true if information page was successfully removed from the database)
Data access:	Deletes a row in the database table <i>InformationPage</i>
Pre-conditions:	- A connection to the database is established - The information page exists in the database
Validity Check:	N/A
Post-conditions:	- The information page's row is removed from the <i>InformationPage</i> table in the database
Caller:	- JSP page for removing an information page
Calls:	N/A

Method update

The method updates an information page in the database.

Requirements:	6.1
Parameters:	- InformationPage (the information page to be updated)
Return:	Boolean (true if information page was successfully updated in the database)
Data access:	Update a row in the database table <i>InformationPage</i>
Pre-conditions:	- A connection to the database is established - The information page exists in the database
Validity Check:	N/A
Post-conditions:	- One row is updated in the <i>InformationPage</i> table in the database
Caller:	- JSP page for updating an information page
Calls:	- InformationPage get methods

Class News

Field belongToCourse

The ID of the course that the news belongs to.

Type:	Integer
Access level:	Private

Field content

The content of the news.

Type:	String
Access level:	Private

Field headline

The headline of the news.

Type:	String
Access level:	Private

Field newsID

The ID that uniquely identifies the news.

Type: Integer
Access level: Private

Method getBelongToCourse

Retrieves the value of the belongToCourse field.

Requirements: 5.1-5.2
Parameters: N/A
Return: Integer (ID of the course that the news belongs to)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller:
- NewsController::update
- NewsController::getCourseNews
- NewsController::getUserNews
Calls: N/A

Method getContent

Retrieves the value of the content field.

Requirements: 5.1-5.2
Parameters: N/A
Return: String (content of the news)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller:
- NewsController::update
- NewsController::getCourseNews
- NewsController::getUserNews
Calls: N/A

Method getHeadline

Retrieves the value of the headline field.

Requirements: 5.1-5.2
Parameters: N/A
Return: String (headline of the news)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller:
- NewsController::update
- NewsController::getCourseNews
- NewsController::getUserNews
Calls: N/A

Method getNewsID

Retrieves the value of the newsID field.

Requirements:	5.1
Parameters:	N/A
Return:	Integer (ID of the news)
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	N/A
Post-conditions:	N/A
Caller:	- NewsController::update
Calls:	N/A

Method setBelongToCourse

Stores an integer in the belongToCourse field.

Requirements:	5.1-5.2
Parameters:	- Integer (ID of the course that the news belongs to)
Return:	Boolean (true if the ID was successfully stored and validated)
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	- The course ID is an integer and identifies a course that exists
Post-conditions:	N/A
Caller:	- NewsController::add - NewsController::update
Calls:	N/A

Method setContent

Stores a String in the content field.

Requirements:	5.1-5.2
Parameters:	- String (content of the news)
Return:	Boolean (true if the content was successfully stored and validated)
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	- The description is a string
Post-conditions:	N/A
Caller:	- NewsController::add - NewsController::update
Calls:	N/A

Method setHeadline

Stores a String in the headline field.

Requirements:	5.1-5.2
Parameters:	- String (headline of the news)
Return:	Boolean (true if the headline was successfully stored and validated)

Data access: N/A
Pre-conditions: N/A
Validity Check: - The headline is a string no longer than 50 characters
Post-conditions: N/A
Caller: - NewsController::add
 - NewsController::update
Calls: N/A

Class NewsController

Method add

The method stores a news post in the database.

Requirements: 5.1
Parameters: - News (the news post to be stored)
Return: Boolean (true if news post was successfully stored in the database)
Data access: Inserts a row in the database table *News*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: - One new row is inserted into the *News* table in the database
Caller: - JSP page for adding a news post
Calls: - News get methods

Method get

The method gets a news post from the database.

Requirements: 5.1
Parameters: - Integer (the news ID of the news to be fetched)
Return: News (the news post corresponding to the news ID)
Data access: Fetches a row in the database table *News*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for editing or viewing a news post
Calls: - News set methods

Method getNewsByCourse

The method generates a list of all news for a specific course.

Requirements: 5.2
Parameters: - Course (the course for which news shall be retrieved)
Return: ArrayList (containing all news for the course)
Data access: Retrieve rows from the database table *News*
Pre-conditions: - A connection to the database is established
 - The course exists in the database
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for displaying news for a course
Calls: - News set methods

Method getNewsByUser

The method generates a list of all news for courses that a specific user is registered for.

Requirements:	2.1-2.3
Parameters:	- User (the user for which news shall be retrieved)
Return:	ArrayList (containing all news for the user's courses)
Data access:	Retrieves row from the database table <i>News</i>
Pre-conditions:	- A connection to the database is established - The user exists in the database
Validity Check:	N/A
Post-conditions:	N/A
Caller:	- JSP page for displaying news for a user
Calls:	- News get methods

Method remove

The method removes a news post from the database.

Requirements:	5.1
Parameters:	- News (the news post to be removed)
Return:	Boolean (true if news post was successfully removed from the database)
Data access:	Deletes a row in the database table <i>News</i>
Pre-conditions:	- A connection to the database is established - The news post exists in the database
Validity Check:	N/A
Post-conditions:	- The news post's row is removed from the <i>News</i> table in the database
Caller:	- JSP page for removing a news post
Calls:	N/A

Method update

The method updates a news post in the database.

Requirements:	5.1
Parameters:	- News (the news post to be updated)
Return:	Boolean (true if news post was successfully updated in the database)
Data access:	Update a row in the database table <i>News</i>
Pre-conditions:	- A connection to the database is established - The news post exists in the database
Validity Check:	N/A
Post-conditions:	- One row is updated in the <i>News</i> table in the database
Caller:	- JSP page for updating a news post
Calls:	- News set methods

Class Result

Field belongToAssignment

The assignment the result is registered for.

Type: Assignment
Access level: Private

Field belongToCourse

The ID of the course that the result belongs to.

Type: Integer
Access level: Private

Field grade

The grade of the assignment.

Type: String
Access level: Private

Field user

The user that the result belongs to.

Type: User
Access level: Private

Method getBelongToAssignment

Retrieves the value of the belongToAssignment field.

Requirements: 11.1-11.2
Parameters: N/A
Return: Assignment (assignment that the result belongs to)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - ResultController::update
Calls: N/A

Method getBelongToCourse

Retrieves the value of the belongToCourse field.

Requirements: 11.1-11.2
Parameters: N/A
Return: String (name of the course that the result belongs to)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - ResultController::update
Calls: N/A

Method getGrade

Retrieves the value in the grade field.

Requirements: 11.1-11.2

Parameters: N/A
Return: String (grade of an assignment for a specific course)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - AssignmentController::update
Calls: N/A

Method getUser

Retrieves the value of the user field.

Requirements: 11.1-11.2
Parameters: N/A
Return: User (user that the result belongs to)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - ResultController::update
Calls: N/A

Method setBelongToAssignment

Stores an Assignment object in the belongToAssignment field.

Requirements: 11.1-11.2
Parameters: - Assignment (Assignment the result belongs to)
Return: Boolean (true if the assignment was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The assignment is an Assignment object and identifies an existing assignment for the specified course in the belongToCourse field.
Post-conditions: N/A
Caller: - ResultController::add
- ResultController::update
Calls: N/A

Method setBelongToCourse

Stores an integer in the belongToCourse field.

Requirements: 11.1-11.2
Parameters: - Integer (ID of the course that the result belongs to)
Return: Boolean (true if the ID was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The course ID is an integer and identifies a course that exists

Post-conditions: N/A
Caller: - ResultController::add
- ResultController::update
Calls: N/A

Method setGrade

Stores a String in the grade field.

Requirements: 11.1-11.2
Parameters: - String (the grade of a specified assignment for a specified course)
Return: Boolean (true if the grade was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The grade has to be one of the letters A-F, the letter P or the string "Fx"
Post-conditions: N/A
Caller: - AssignmentController::add
- AssignmentController::update
Calls: N/A

Method setUser

Stores a User object in the user field.

Requirements: 11.1-11.2
Parameters: - User (user that the result belongs to)
Return: Boolean (true if the user was successfully stored and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The input is a User object and identifies a user that exists
Post-conditions: N/A
Caller: - ResultController::add
- ResultController::update
Calls: N/A

Class ResultController

Method getResultByUser

The method gets all results for a user from the database.

Requirements: 11.2
Parameters: - Integer (the user ID of the user whose results are to be fetched)
Return: ArrayList<Result> (the results corresponding to the user)
Data access: Fetches rows from the database table *Result*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for viewing results for a user

Calls: - Result set methods

Method getResultByUserAndCourse

The method gets all results for a user and a course from the database.

Requirements: 11.2
Parameters: - Integer (the user ID of the user whose results are to be fetched)
- Integer (the course ID of the course which's results are to be fetched)
Return: ArrayList<Result> (the results corresponding to the user and course)
Data access: Fetches rows from the database table *Result*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for viewing results for a user and course
Calls: - Result set methods

Method update

The method updates a user's result for an assignment in the database. If no result previously exists for the user, a new result is created. If no result is given, any previous result is deleted.

Requirements: 11.1
Parameters: - Result (the result to be saved, contains an assignment, a user and a grade)
Return: Boolean (true if the result for the assignment was successfully changed in the database)
Data access: Update, inserted or deleted a row in the database table *Result*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: - One row is updated, inserted or deleted in the *Result* table in the database
Caller: - JSP page for updating a result for an assignment
Calls: - Result get methods

Class Session

Method authenticate

Authenticates a user with the system.

Requirements: 1.1
Parameters: - String (username)
- String (password)
Return: Boolean (true if authentication was successful)
Data access: Retrieves values from database table *User*
Pre-conditions: - User is not logged in
- A connection to the database is established
Validity Check: N/A

Post-conditions: - The user state in the current session is set to logged in
Caller: - Any page that allows the user to log in.
Calls: - UserController::getUserByUsername
- Cache::add
- Cache::get

Method logout

Logs out a user from the system.

Requirements: 1.1
Parameters: N/A
Return: Void
Data access: N/A
Pre-conditions: - User is logged in.
Validity Check: N/A
Post-conditions: - The user state in the current session is set to logged out
- Any temporarily saved user data is removed.
Caller: - Any page.
Calls: N/A

Class Schedule

Field activities

Type: ArrayList<Activity>
Access level: Private

Field belongToCourse

The ID of the course that the schedule belongs to.

Type: Integer
Access level: Private

Method addActivity

The method adds an Activity object in the activities ArrayList.

Requirements: 7.1-7.5
Parameters: - Activity (activity to be added in the schedule that belongs to the specified course)
Return: Boolean (true if activity was successfully added and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The activity has to be an existing Activity object belonging to the course specified in the belongToCourse field.
Post-conditions: N/A
Caller: N/A
Calls: N/A

Method getAllActivities

The method retrieves all activities

Requirements: 7.1-7.5
Parameters: N/A
Return: ArrayList<Activity> (All the activities for a course)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - ScheduleController::getCourseSchedule
 - ScheduleController::getUserSchedule
Calls: N/A

Method getBelongToCourse

Retrieves the value of the belongToCourse field.

Requirements: 7.1-7.5
Parameters: N/A
Return: Integer (ID of the course that the schedule belongs to)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - ScheduleController::getCourseSchedule
 - ScheduleController::getUserSchedule
Calls: N/A

Method setBelongToCourse

Stores an integer in the belongToCourse field.

Requirements: 7.1-7.5
Parameters: - Integer (ID of the course that the schedule belongs to)
Return: Boolean (true if course ID was successfully added and validated)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The course ID is an integer and identifies a course that exists
Post-conditions: N/A
Caller: N/A
Calls: N/A

Class ScheduleController

Method export

The method exports a schedule into the iCalendar format.

Requirements: 7.5, 13.3
Parameters: N/A
Return: String (representing schedule in iCalendar format)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A

Post-conditions: - Scheduled activities are represented in iCalendar format
Caller: - JSP page for viewing a schedule
Calls: - Activity get methods.

Method `getScheduleByCourse`

The method generates a list of all activities for a specific course.

Requirements: 7.3
Parameters: - Course (the course for which news shall be retrieved)
Return: ArrayList (containing all scheduled activities for the course)
Data access: Retrieve rows from the database table *News*
Pre-conditions: - A connection to the database is established
- The course exists in the database
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for displaying a schedule for a course
Calls: - Activity get methods

Method `getScheduleByUser`

The method generates a list of all scheduled activities for courses that a specific user is registered for.

Requirements: 2.1-2.2
Parameters: - User (the user for which a schedule shall be retrieved)
Return: ArrayList (containing all scheduled activities for the user's courses)
Data access: Retrieves row from the database table *News*
Pre-conditions: - A connection to the database is established
- The user exists in the database
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for displaying a schedule for a user
Calls: - Activity get methods

Method `import`

The method imports a schedule and stores it in the database.

Requirements: 7.1
Parameters: - Schedule (represented in iCalendar format)
Return: Boolean (true if schedule was successfully stored in the database)
Data access: Inserts new rows into the *Activity* database table.
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: - Scheduled activities are inserted into the *Activity* table in the database
Caller: - JSP page for importing a schedule
Calls: - N/A

Method `removeScheduleByCourse`

The method removes all scheduled activities belonging to a course.

Requirements: 7.1, 13.3
Parameters: - String (containing course code)
Return: Void
Data access: All rows in table *Activity* containing activities for the course are removed.
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: - There are no activities in the database table *Activity* for the course
Caller: - JSP page for removing a schedule
Calls: N/A

Class User

Field courseAssistant

The course assistant privilege of a user. Contains course ID's for the courses for which the user has the privilege course assistant.

Type: HashSet<Integer>
Access level: Private

Field courseLeader

The course leader privilege of a user. Contains course ID's for the courses for which the user has the privilege course leader.

Type: HashSet<Integer>
Access level: Private

Field firstname

The firstname of a user.

Type: String
Access level: Private

Field lastname

The lastname of a user.

Type: String
Access level: Private

Field password

The password of a user.

Type: String
Access level: Private

Field sysadmin

The system administrator privilege of a user. If true then the user has the privilege "System Administrator".

Type: boolean
Access level: Private

Field userID

The ID that uniquely identifies the user.

Type: Integer
Access level: Private

Field username

The username of a user.

Type: String
Access level: Private

Method getCourseAssistantForCourse

The method returns true if the user has the course assistant privilege for the given course.

Requirements: 11.1
Parameters: - Course (the course to check privilege for)
Return: Boolean (true if the user has the privilege)
Data access: Retrieve rows from the table *Privilege* if the privileges haven't been previously fetched.
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: N/A
Caller: - Any JSP page where the course assistant privilege is sufficient
Calls: N/A

Method getCourseLeaderForCourse

The method returns true if the user has the course leader privilege for the given course.

Requirements: 3.1, 4.1, 5.1, 6.1, 7.1, 7.2, 8.1, 9.1, 10.1, 11.1, 12.1, 12.2, 12.4 and 13.2
Parameters: - Course (the course to check privilege for)
Return: Boolean (true if the user has the privilege)
Data access: Retrieve rows from the table *Privilege* if the privileges haven't been previously fetched.
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: N/A
Caller: - Any JSP page where the course leader privilege is sufficient
Calls: N/A

Method getFirstName

The method retrieves a first name.

Requirements: 13.4
Parameters: N/A
Return: String (first name of the user)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - UserController::update
Calls: N/A

Method `getLastName`

The method retrieves a last name.

Requirements: 13.4
Parameters: N/A
Return: String (Last name of the user)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - UserController::update
Calls: N/A

Method `getPassword`

The method retrieves a password.

Requirements: 13.4
Parameters: N/A
Return: Char (password of the user)
Data access: N/A
Pre-conditions: N/A
Validity Check: N/A
Post-conditions: N/A
Caller: - UserController::update
Calls: N/A

Method `getStudentInCourse`

The method returns true if the user is registered in the course.

Requirements: 9.2
Parameters: - Course (the course to check privilege for)
Return: Boolean (true if the user is registered for the course)
Data access: Retrieve rows from the table *InCourse* if the privileges haven't been previously fetched.
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: N/A
Caller: - Any JSP page where the being registered for a specific course is sufficient
Calls: N/A

Method `getSysAdmin`

The method retrieves the system administrator privilege for a user.

Requirements:	13.1, 13.3, 13.4
Parameters:	N/A
Return:	Boolean (true if the user is a system administrator)
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	N/A
Post-conditions:	N/A
Caller:	- <code>UserController::update</code>
Calls:	N/A

Method `getUserID`

Retrieves the value of the `userID` field.

Requirements:	12.2, 12.4, 13.1-13.2, 13.4
Parameters:	N/A
Return:	Integer (ID of the user)
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	N/A
Post-conditions:	N/A
Caller:	- <code>UserController::update</code>
Calls:	N/A

Method `getUsername`

The method retrieves a username.

Requirements:	13.4
Parameters:	N/A
Return:	String (username of the user)
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	N/A
Post-conditions:	N/A
Caller:	- <code>UserController::update</code>
Calls:	N/A

Method `setCourseAssistantForCourse`

Stores the course assistant privilege. Adds an integer representing the course (course ID) to the `courseAssistant` HashSet.

Requirements:	13.1
Parameters:	- Integer (course ID for the course to assign the privilege for)
Return:	N/A
Data access:	N/A
Pre-conditions:	N/A
Validity Check:	- The course has to exist.
Post-conditions:	N/A
Caller:	- <code>UserController::assignCourseAssistant</code>

Calls: N/A

Method setCourseLeaderForCourse

Stores the course leader privilege. Adds an integer representing the course (course ID) to the courseLeader HashSet.

Requirements: 13.1
Parameters: - Integer (course ID for the course to assign the privilege for)
Return: N/A
Data access: N/A
Pre-conditions: N/A
Validity Check: - The course has to exist.
Post-conditions: N/A
Caller: - UserController::assignCourseLeader
Calls: N/A

Method setFirstName

The method sets a first name.

Requirements: 13.4
Parameters: - Firstname (the first name to be stored)
Return: Boolean (true if first name was successfully stored in the database)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The username is a string
Post-conditions: N/A
Caller: - UserController::update
- UserController::add
Calls: N/A

Method setLastName

The method sets a last name.

Requirements: 13.4
Parameters: - Lastname (the last name to be stored)
Return: Boolean (true if last name was successfully stored in the database)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The lastname is a string
Post-conditions: N/A
Caller: - UserController::update
- UserController::add
Calls: N/A

Method setPassword

The method sets a password.

Requirements: 13.4
Parameters: - Password (the password to be stored)

Return: Boolean (true if password was successfully stored in the database)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The password is at least 5 chars
Post-conditions: N/A
Caller: - UserController::update
- UserController::add
Calls: N/A

Method setSysAdmin

Stores a Boolean in the system administrator privilege field.

Requirements: 13.1
Parameters: - Boolean (true if the user is a system administrator)
Return: N/A
Data access: N/A
Pre-conditions: N/A
Validity Check: - The privilege has to be a Boolean.
Post-conditions: N/A
Caller: - UserController::add
- UserController::update
Calls: N/A

Method setUsername

The method sets a username.

Requirements: 13.4
Parameters: - Username (the username to be stored)
Return: Boolean (true if username was successfully stored in the database)
Data access: N/A
Pre-conditions: N/A
Validity Check: - The username is a string and is at least 5 chars long
Post-conditions: N/A
Caller: - UserController::update
- UserController::add
Calls: N/A

Class UserController

Method add

The method stores a user in the database.

Requirements: 13.4
Parameters: - User (the user to be stored)
Return: Boolean (true if user was successfully stored in the database)
Data access: Inserts a row in the database table *User*
Pre-conditions: - A connection to the database is established
Validity Check: N/A

Post-conditions: - One new row is inserted into the *User* table in the database
Caller: - JSP page for adding a user
Calls: - User get methods

Method assignCourseAssistant

The method assigns a user the course assistant privilege in the database by updating the status field of the *InCourse* table to ASSISTANT.

Requirements: 13.1
Parameters: - User (the user to assign the privilege to)
- Course (the course to assign the privilege to)
Return: Boolean (true if privilege was successfully assigned in the database)
Data access: Update a row in the database table *InCourse*
Pre-conditions: - A connection to the database is established
- The user exists in the database
- The course exists in the database
Validity Check: N/A
Post-conditions: - One row is updated or inserted in the *InCourse* table in the database with value of the status field set to ASSISTANT
Caller: - JSP page for assigning the course assistant privilege
Calls: N/A

Method assignCourseLeader

The method assigns a user the course leader privilege in the database by updating the status field of the *InCourse* table to LEADER.

Requirements: 13.1
Parameters: - User (the user to assign the privilege to)
- Course (the course to assign the privilege to)
Return: Boolean (true if privilege was successfully assigned in the database)
Data access: Update a row in the database table *InCourse*
Pre-conditions: - A connection to the database is established
- The user exists in the database
- The course exists in the database
Validity Check: N/A
Post-conditions: - One row is updated or inserted in the *InCourse* table in the database with value of the status field set to LEADER
Caller: - JSP page for assigning the course leader privilege
Calls: N/A

Method assignSysAdmin

The method assigns a user the system administrator privilege in the database.

Requirements: 13.1
Parameters: - User (the user to assign the privilege to)
Return: Boolean (true if privilege was successfully assigned in the database)
Data access: Update a row in the database table *Privilege*

Pre-conditions: - A connection to the database is established
- The user exists in the database
Validity Check: N/A
Post-conditions: - One row is inserted in the *Privilege* table in the database
Caller: - JSP page for assigning the system administrator privilege
Calls: N/A

Method get

The method gets a user from the database.

Requirements: 13.4
Parameters: - Integer (the user ID of the user to be fetched)
Return: User (the user corresponding to the user ID)
Data access: Fetches a row in the database table *User*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: N/A
Caller: - JSP page for editing or viewing a user
Calls: - User set methods

Method getUserByUsername

The method gets a user from the database using a username. The username has to match exactly.

Requirements: 1.1
Parameters: - String (the username of the user to be fetched)
Return: User (the user corresponding to the username)
Data access: Fetches a row in the database table *User*
Pre-conditions: - A connection to the database is established
Validity Check: N/A
Post-conditions: N/A
Caller: - Session::authenticate
Calls: - User set methods

Method remove

The method removes a user from the database.

Requirements: 13.4
Parameters: - User (the user to be removed)
Return: Boolean (true if user was successfully removed from the database)
Data access: Deletes a row in the database table *User*
Pre-conditions: - A connection to the database is established
- The user exists in the database
Validity Check: N/A
Post-conditions: - The user's row is removed from the *User* table in the database
Caller: - JSP page for removing a user
Calls: N/A

Method revokeCourseAssistant

The method revokes the course assistant privilege for a user in the database by removing the row for the user and course in the *InCourse* database table.

Requirements:	13.1
Parameters:	- User (the user to revoke the privilege from) - Course (the course to revoke the privilege from)
Return:	Boolean (true if privilege was successfully revoked in the database)
Data access:	Delete a row in the database table <i>InCourse</i>
Pre-conditions:	- A connection to the database is established - The user exists in the database - The course exists in the database
Validity Check:	N/A
Post-conditions:	- The user is no longer course assistant for the specified course
Caller:	- JSP page for revoking the course assistant privilege
Calls:	N/A

Method revokeCourseLeader

The method revokes the course assistant privilege for a user in the database by removing the row for the user and course in the *InCourse* database table.

Requirements:	13.1
Parameters:	- User (the user to revoke the privilege from) - Course (the course to revoke the privilege from)
Return:	Boolean (true if privilege was successfully revoked in the database)
Data access:	Delete a row in the database table <i>InCourse</i>
Pre-conditions:	- A connection to the database is established - The user exists in the database - The course exists in the database
Validity Check:	N/A
Post-conditions:	- The user is no longer course leader for the specified course
Caller:	- JSP page for revoking the course leader privilege
Calls:	N/A

Method revokeSysAdmin

The method revokes a user the system administrator privilege in the database.

Requirements:	13.1
Parameters:	- User (the user to revoke the privilege from)
Return:	Boolean (true if privilege was successfully revoked in the database)
Data access:	Delete a row in the database table <i>Privilege</i>
Pre-conditions:	- A connection to the database is established - The user exists in the database
Validity Check:	N/A
Post-conditions:	- The user is no longer system administrator
Caller:	- JSP page for revoking the system administrator privilege
Calls:	N/A

Method update

The method updates a user in the database.

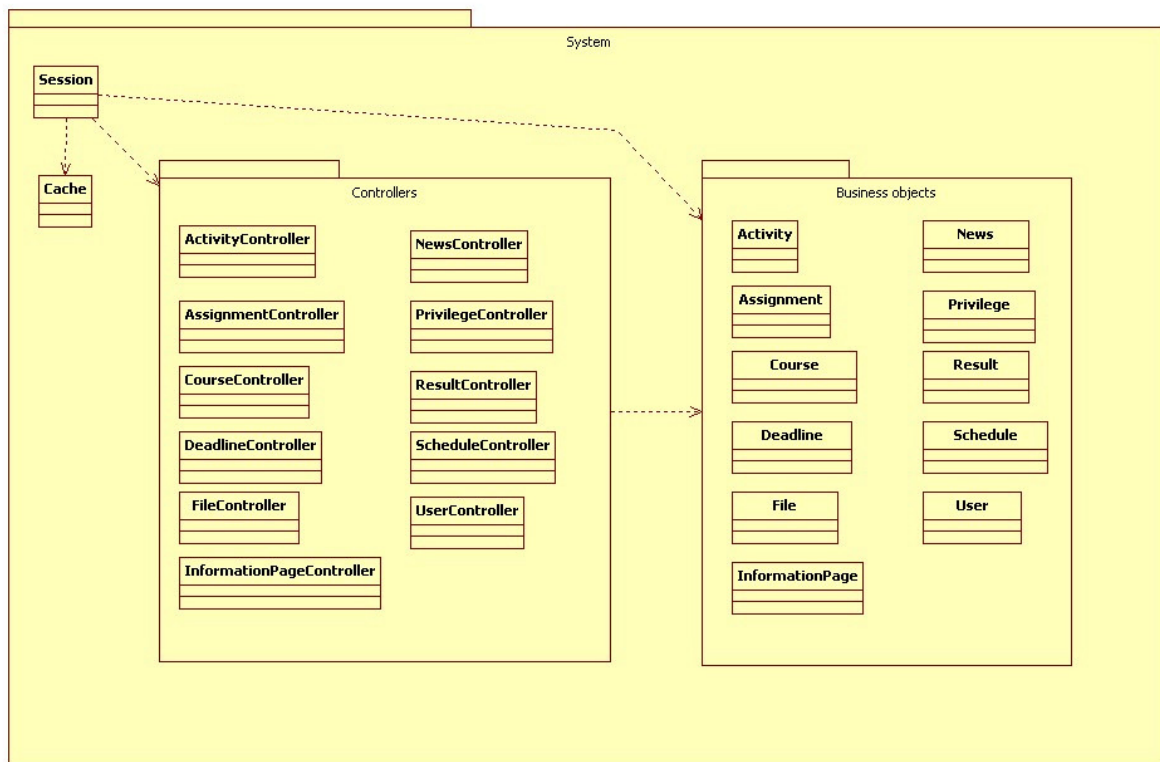
Requirements:	13.4
Parameters:	- User (the user to be updated)
Return:	Boolean (true if user was successfully updated in the database)
Data access:	Update a row in the database table <i>User</i>
Pre-conditions:	- A connection to the database is established - The user exists in the database
Validity Check:	N/A
Post-conditions:	- One row is updated in the <i>User</i> table in the database
Caller:	- JSP page for updating a user
Calls:	- User get methods

Implementation Index of Requirements

Requirement	Implemented in
1.1	Session::Authenticate Session::Logout
2.1	DeadlineController::GetDeadlinesByUser NewsController::GetNewsByUser ScheduleController::GetScheduleByUser ResultController::GetResultByUser
2.2	Display implemented in JSP page DeadlineController::GetDeadlinesByUser NewsController::GetNewsByUser ScheduleController::GetScheduleByUser
2.3	NewsController::GetNewsByUser
3.1	JSP Page for the creation guide
4.1	CourseController::UpdateDescription
4.2	CourseController::GetDescription
5.1	NewsController::Add NewsController::GetNewsByCourse NewsController::Update NewsController::Remove
5.2	NewsController::GetNewsByCourse
6.1	InformationPageController::Add InformationPageController::Update InformationPageController::Remove InformationPageController::GetInformationPageByCourse
6.2	InformationPageController::Get
7.1	ScheduleController::Import
7.1	ScheduleController::RemoveScheduleByCourse
7.2	ActivityController::Add ActivityController::Update ActivityController::Remove
7.3	ScheduleController::GetScheduleByCourse

ActivityController::Get
 7.4 ScheduleController::GetScheduleByUser
 7.5 ScheduleController::Export
 8.1 DeadlineController::Add
 DeadlineController::Update
 DeadlineController::Remove
 8.2 DeadlineController::Get
 8.3 DeadlineController::GetDeadlinesByUser
 9.1 FileController::Add
 FileController::Update
 FileController::Remove
 9.2 FileController::Get
 Course::GetAllUsers
 10.1 AssignmentController::Add
 AssignmentController::Update
 AssignmentController::Remove
 10.2 AssignmentController::GetAssignmentByCourse
 11.1 ResultController::Add
 11.2 ResultController::GetResultByUser
 12.1 Course::GetAllUsers
 12.2 CourseController::Register
 Course::AddUser
 12.3 CourseController::Apply
 12.4 CourseController::Unregister
 13.1 UserController::AssignCourseLeader
 UserController::AssignCourseAssistant
 UserController::AssignSysAdmin
 13.2 UserController::AssignCourseAssistant
 UserController::RevokeCourseAssistant
 13.3 CourseController::Add
 CourseController::Update
 13.4 UserController::Add
 UserController::Update
 UserController::Remove
 14.1 Cache::Add
 Cache::Get
 14.2 Cache::Add
 Cache::Get
 14.3 Cache::Add
 Cache::Get

5.6 Package diagram



The system can be roughly divided into two packages, one being the controller package. These classes handle loading and saving of information to the database. The other package is the business object package, where the classes represent relevant domain entities such as news, assignments etc. These objects encapsulate all the data of the related entities and are also responsible for all data validation.

6. Functional Test Cases

Test Case TC1: Authenticate to the System

Functionality to Test: The user shall be able to log in.

Functional Requirement: 1.1

Inputs:

- Username
- Password

Expected Outputs:

- User session
- Confirmation

Instructions for Tester

1. Input username and password.
2. Select log in.
3. Verify that the page has a log out button.

Test Case TC2: View Personal Page

Functionality to Test: The user shall be able to view his or hers personal page.

Functional Requirement: 2.1

Inputs: None
Expected Outputs: - Personal Page

Instructions for Tester

1. Navigate to the "News" under "Personal Links" section of the website.
2. Verify that the personal page is displayed.

Test Case TC3: View Overview of Course News

Functionality to Test: The user shall be able to view an overview of the user's courses news.

Functional Requirement: 2.3

Inputs: None

Expected Outputs: - List of course news for the courses the student is registered for

Instructions for Tester

1. Navigate to the "News" under "Personal Links" section of the website.
2. Verify that the course news for the courses the user is registered for is displayed.

Test Case TC4: Create Course Website

Functionality to Test: The user shall, if assigned as course leader for a course, be able to create a course website with the help of a guide.

Functional Requirement: 3.1

Inputs:

- Course name
- Credits
- Start period
- End period
- Description of course
- Schedule
- Title of information page
- Content of information page
- Title of deadline
- Time of deadline
- Description of deadline

Expected Outputs: - Course website and confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website creation guide.
3. Input course name, credits, start period, end period and description.
4. Select Continue.
5. Select a schedule to import.
6. Select Import.
7. Select Continue.
8. Input title and content of an information page.
9. Select Preview.

10. Select Save.
11. Select Continue.
12. Input title, time and description of a deadline.
13. Select Preview.
14. Select Save.
15. Select Continue.
16. Select Create Course Website.
17. Verify that the course website is available by selecting to the course website.

Test Case TC5: Edit Existing Course Description

Functionality to Test: The user shall, if assigned course leader for the course, be able to edit the existing course description.

Functional Requirement: 4.1

Inputs:

- Course Name
- Credits
- Begin Period
- End Period
- Description

Expected Outputs:

- Course description
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to a course website.
3. Navigate to the "Course Description" under "Course Leader Links" section of a course website.
4. Edit the course description.
5. Select Save.
6. Verify that the course description is available from the course description page (TC6).

Test Case TC6: View Course Description

Functionality to Test: The user shall be able to view a course description.

Functional Requirement: 4.2

Inputs: None

Expected Outputs: - Course description

Instructions for Tester

1. Navigate to a course website.
2. Navigate to the "Course Description" page of a course website.
3. Verify that the course description for the course is displayed.

Test Case TC7: Add Course News

Functionality to Test: The user shall, if assigned course leader for the course, be able to add course news to the course.

Functional Requirement: 5.1

Inputs: - Headline

Expected Outputs:

- Content
- Course news
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to a course website.
3. Navigate to the "News" under "Course Leader Links" section of a course website.
4. Input headline and content of a news.
5. Select Preview.
6. Select Save.
7. Verify that the news is available from the news page (TC10).

Test Case TC8: Edit Existing Course News

Functionality to Test: The user shall, if assigned course leader for the course, be able to edit existing course news.

Functional Requirement: 5.1

Inputs:

- Headline
- Content

Expected Outputs:

- Course news
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to a course website.
3. Navigate to the "News" under "Course Leader Links" section of a course website.
4. Select news to edit.
5. Edit the news.
6. Select Preview.
7. Select Save.
8. Verify that the news is available from the news page (TC10).

Test Case TC9: Remove Existing Course News

Functionality to Test: The user shall, if assigned course leader for the course, be able to remove existing course news.

Functional Requirement: 5.1

Inputs:

- Course news to be removed

Expected Outputs:

- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to a course website.
3. Navigate to the "News" under "Course Leader Links" section of a course website.
4. Select news to remove.

5. Verify that the news is removed from the news page (TC10).

Test Case TC10: View Course News

Functionality to Test: The user shall be able to view course news.
Functional Requirement: 5.2
Inputs: None
Expected Outputs: - The course news for a course order by date in descending order

Instructions for Tester

1. Navigate to a course website.
2. Navigate to the "News" section of a course website.
3. Verify that the course news for the course is displayed.

Test Case TC11: Add Information Page

Functionality to Test: The user shall be able to add an information page to the course website for courses he or she is assigned the privilege course leader if he or she is authenticated.
Functional Requirement: 6.1
Inputs: - Title
- Content
Expected Outputs: - Information page
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the "Information pages" management section for a course website.
3. Input title and content where requested.
4. Select Preview.
5. Select Save.
6. Verify that the information page has been added to the list of information pages in the "Information page" management section, and view the information page (TC14).

Test Case TC12: Edit Existing Information Page

Functionality to Test: The user shall be able to edit an existing information page for courses he or she is assigned the privilege course leader if he or she is authenticated.
Functional Requirement: 6.1
Inputs: - The information page to edit
- Title
- Content
Expected Outputs: - Information page
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the "Information pages" management section for a course website.
3. Select an existing information page to edit.
4. Input updated title and content where requested.
5. Select Preview.
6. Select Save.
7. Verify that the information page has been edited accordingly (TC14).

Test Case TC13: Remove Existing Information Page

Functionality to Test: The user shall be able to remove an existing information page for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 6.1

Inputs: - Information page to be removed

Expected Outputs: - Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the "Information pages" management section for a course website.
3. Select an existing information page to remove.
4. Confirm removal of the selected information page.
5. Verify that the information page has been removed from the list of information pages in the "Information page" management section.

Test Case TC14: View Information Page

Functionality to Test: The user shall be able to view an information page.

Functional Requirement: 6.2

Inputs: - Selected information page

Expected Outputs: - Information page for viewing

Instructions for Tester

1. Navigate to a course website.
2. Select an information page to view.
3. Verify that the selected information page is displayed.

Test Case TC15: Import Course Schedule

Functionality to Test: The user shall be able to import a course schedule in the iCalendar format for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 7.1

Inputs: - Schedule in the iCalendar format

Expected Outputs: - Schedule
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the "Schedule" management section for a course website.
3. Input the location of the iCalendar file, where requested.
4. Select Import.
5. Verify that the schedule has been added correctly (TC20).

Test Case TC16: Remove Existing Course Schedule

Functionality to Test: The user shall be able to remove an existing course schedule for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 7.1

Inputs: None

Expected Outputs: - Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the "Schedule" management section for a course website.
3. Select remove schedule.
4. Confirm removal of schedule.
5. Verify that all the scheduled activities have been removed from the list of scheduled activities in the "Schedule" management section.

Test Case TC17: Add Scheduled Activity

Functionality to Test: The user shall be able to add a scheduled activity for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 7.2

Inputs:

- Title
- Description
- Starting date
- Starting time
- Ending date
- Ending time

Expected Outputs:

- Scheduled activity
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the "Schedule" management section for a course website.
3. Input title, starting date, starting time, ending date, ending time and description where requested.
4. Select Preview.
5. Select Save.
6. Verify that the scheduled activity has been added to the course schedule (TC20) and the compiled schedule (TC21).

Test Case TC18: Edit Existing Scheduled Activity

Functionality to Test: The user shall be able to edit an existing scheduled activity for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 7.2

Inputs:

- The scheduled activity to edit
- Title
- Description
- Starting date
- Starting time
- Ending date
- Ending time

Expected Outputs:

- Scheduled activity
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the "Schedule" management section for a course website.
3. Select the existing scheduled activity to edit.
4. Input title, starting date, starting time, ending date, ending time and description where requested.
5. Select Preview.
6. Select Save.
7. Verify that the scheduled activity has been edited accordingly (TC20 and TC21).

Test Case TC19: Remove Existing Scheduled Activity

Functionality to Test: The user shall be able to remove an existing scheduled activity for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 7.2

Inputs:

- The scheduled activity to remove

Expected Outputs:

- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the "Schedule" management section for a course website.
3. Select the existing scheduled activity to remove.
4. Confirm removal of selected scheduled activity.
5. Verify that the scheduled activity has been removed from the list of scheduled activities in the "Schedule" management section.

Test Case TC20: View Scheduled Activity from Course Schedule

Functionality to Test: The user shall be able to view a scheduled activity from the course schedule.

Functional Requirement: 7.3

Inputs:

- The selected scheduled activity to view

Expected Outputs:

- Detailed information for the selected scheduled activity

Instructions for Tester

1. Navigate to a course website.
2. Select to view the course schedule.
3. Select a scheduled activity from the course schedule.
4. Verify that the details for the activity are displayed.

Test Case TC21: View Scheduled Activity from Compiled Schedule

Functionality to Test: The user shall be able to view scheduled activities from the compiled schedule, which can be accessed through the personal page, and the details of the activities.

Functional Requirement: 7.4

Inputs: - Scheduled activity

Expected Outputs: - The description of the selected activity

Instructions for Tester

1. Navigate to the personal page.
2. Navigate to the "Schedule" section under "Personal Links".
3. Select a schedule activity to view the description of the activity.
4. Verify that the details for the activity are displayed.

Test Case TC22: Export Schedule in iCalendar Format

Functionality to Test: The user shall be able to export the compiled schedule and the course schedule on the course website.

Functional Requirement: 7.5

Inputs: - The schedule

Expected Outputs: - The schedule in iCalendar format

Instructions for Tester

1. Navigate to the personal page or the course website.
2. Navigate to the "Schedule" section.
3. Select Export this schedule to iCalendar format.
4. Verify that the schedule was exported by importing it to Google Calendar and comparing the activities' times and descriptions.

Test Case TC23: Add Deadline

Functionality to Test: The user shall be able to add deadlines for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 8.1

Inputs:
- Title of the deadline
- Date of the deadline
- Description of the deadline

Expected Outputs:
- The added deadline
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website.
3. Navigate to the "Deadline" section under "Course Leader Links".
4. Input title, date and description for the deadline.
5. Select Preview.
6. Select Save.
7. Verify that the deadline is available from the deadline page (TC26).

Test Case TC24: Edit Existing Deadline

Functionality to Test: The user shall be able to edit existing deadlines for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 8.1

Inputs:

- The selected deadline
- Title of the deadline
- Date of the deadline
- Description of the deadline

Expected Outputs:

- The edited deadline
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website.
3. Navigate to the "Deadline" section under "Course Leader Links".
4. Select deadline to edit.
5. Edit title, date or description for the deadline.
6. Select Preview.
7. Select Save.
8. Verify that the deadline is available from the deadline page (TC26).

Test Case TC25: Remove Existing Deadline

Functionality to Test: The user shall be able to remove existing deadlines for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 8.1

Inputs:

- The selected deadline

Expected Outputs:

- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website.
3. Navigate to the "Deadline" section under "Course Leader Links".
4. Select deadline to remove.
5. Select Confirm.
6. Verify that the deadline is removed from the deadline page (TC26).

Test Case TC26: View Deadlines

Functionality to Test: The user shall be able to view existing deadlines for a course.

Functional Requirement: 8.2

Inputs: - The ID of a deadline

Expected Outputs: - The selected deadline

Instructions for Tester

1. Navigate to the course website.
2. Navigate to the "Deadline" section under "Courses".
3. Select deadline to view.
4. Verify that the selected deadline is displayed.

Test Case TC27: View Overview of Deadlines

Functionality to Test: The user shall be able to view the overview of deadlines for courses that the user is registered for.

Functional Requirement: 8.3

Inputs: None

Expected Outputs: - A list of deadlines for courses the user is registered for

Instructions for Tester

1. Navigate to the personal page.
2. Navigate to the "Deadline" section under "Personal Links".
3. Verify that the deadlines for the courses the user is registered for are displayed.

Test Case TC28: Upload File

Functionality to Test: The user shall be able to upload files for courses that the user is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 9.1

Inputs: - Title of the file
- Description of the file
- File to upload

Expected Outputs: - Uploaded file
- File description
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website.
3. Navigate to the "Files" section under "Course Leader Links".
4. Input title, description and file to upload.
5. Select Save.
6. Verify that the uploaded file is available from the file page (TC31).

Test Case TC29: Edit Existing File

Functionality to Test: The user shall be able to edit existing uploaded files for courses that the user is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 9.1

Inputs:

- File to be edited
- Title of the file
- Description of the file
- File to upload

Expected Outputs:

- Uploaded file
- File description
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website.
3. Navigate to the "Files" section under "Course Leader Links".
4. Select file to be edited.
5. Edit title, description or file to upload.
6. Select Save.
7. Verify that the uploaded file is available from the file page (TC31).

Test Case TC30: Remove Existing File

Functionality to Test: The user shall be able to remove existing uploaded files for courses that the user is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 9.1

Inputs:

- File to be removed

Expected Outputs:

- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website.
3. Navigate to the "Files" section under "Course Leader Links".
4. Select file to be removed.
5. Select Confirm.
6. Verify that the uploaded file is removed from the file page (TC31).

Test Case TC31: Download File

Functionality to Test: The user shall be able to download files for courses he or she is registered for if he or she is authenticated.

Functional Requirement: 9.2

Inputs:

- Username
- Password

Expected Outputs:

- The selected file

Instructions for Tester

1. Authenticate to the system (TC1).

2. Navigate to a course website.
3. Navigate to the "Uploaded Files" section of a course website.
4. Select a file to download.
5. Verify that the file is downloaded.

Test Case TC32: Add Course Assignment

Functionality to Test: The user shall be able to add course assignments for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 10.1

Inputs:

- Title of the assignment
- Description of the assignment

Expected Outputs: - The added assignment and confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website.
3. Navigate to the "Assignment" section under the "Course Leader Links".
4. Input title and description for the course assignment.
5. Select Preview.
6. Select Save.
7. Verify that the assignment is available from the assignment page (TC35).

Test Case TC33: Edit Existing Course Assignment

Functionality to Test: The user shall be able to edit existing course assignments for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 10.1

Inputs:

- Title of the assignment
- Description of the assignment

Expected Outputs: - The edited assignment and confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website.
3. Navigate to the "Assignment" section under the "Course Leader Links".
4. Select assignment to edit.
5. Edit title and description for the course assignment.
6. Select Preview.
7. Select Save.
8. Verify that the assignment is available from the assignment page (TC35).

Test Case TC34: Remove Existing Course Assignment

Functionality to Test: The user shall be able to remove existing course assignments for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 10.1

- Inputs:**
- Title of the assignment
 - Description of the assignment
- Expected Outputs:**
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website.
3. Navigate to the "Assignment" section under the "Course Leader Links".
4. Select course assignment to remove.
5. Select Confirm.
6. Verify that the assignment is removed from the assignment page (TC35).

Test Case TC35: View Course Assignments

Functionality to Test: The user shall be able to view existing course assignments for a course.

Functional Requirement: 10.2

Inputs: None

Expected Outputs: - List of course assignments for the course

Instructions for Tester

1. Navigate to the course website.
2. Navigate to the "Assignment" section.
3. Verify that the assignments for the course are displayed.

Test Case TC36: Register Results

Functionality to Test: The user shall be able to register results for existing course assignments for courses he or she is assigned the privilege course leader or course assistant if he or she is authenticated.

Functional Requirement: 11.1

Inputs:

- Grade
- Course assignment to assign grade to
- User to assign grade to

Expected Outputs: - Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website.
3. Navigate to the "Results" section under the "Course Leader Links".
4. Enter grades for the users and assignments.
5. Select Save.
6. Verify that the grade is available from the register results page (TC36).

Test Case TC37: View Results

Functionality to Test: The user shall be able to view existing results for a course he or she is registered for if he or she is authenticated.

Functional Requirement: 11.2
Inputs: None
Expected Outputs: - List of results for the course

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website.
3. Navigate to the "Results" section.
4. Verify that the user's results for the course are displayed.

Test Case TC38: View Registered Students

Functionality to Test: The user shall be able to view registered users for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 12.1
Inputs: None
Expected Outputs: - List of students registered for the course

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website.
3. Navigate to the "Registrations" section under the "Course Leader Links".
4. Verify that the registered users for the course are displayed.

Test Case TC39: Confirm Application to Get Registered for Course

Functionality to Test: The user shall be able to accept applying users for courses he or she is assigned the privilege course leader if he or she is authenticated.

Functional Requirement: 12.2
Inputs: - Users
Expected Outputs: - List of students for the course

Instructions for Tester

1. Authenticate to the system (TC1).
2. Navigate to the course website.
3. Navigate to the "Registrations" section under the "Course Leader Links".
4. Select student applications to accept.
5. Select Verify the Selected Users.
6. Verify that the user is registered for the course (TC38).

Test Case TC40: Apply for Course

Functionality to Test: The user shall be able to apply for a course which he or she isn't already registered in.

Functional Requirement: 12.3
Inputs: None
Expected Outputs: - Updated student status

Inputs: - Username
- Password
Expected Outputs: - Updated User
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Select Users from the System Administrator submenu.
3. Input the username into the username field under Edit Existing User.
4. Select the Edit Password link corresponding to the user.
5. Input the new password.
6. Press the save button.
7. Verify that the user can log in using the new password (TC1).

Test Case TC44: Remove User Account

Functionality to Test: A system administrator shall be able to remove user accounts.

Functional Requirement: 13.4

Inputs: - Username
Expected Outputs: - Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Select Users from the System Administrator submenu.
3. Input the username into the username field under Edit Existing User.
4. Select the Remove link corresponding to the user.
5. Select Confirm.
6. Verify that the user can no longer log in (TC1).

Test Case TC45a: Edit User Privileges

Functionality to Test: A system administrator shall be able to edit user privileges to make other users System Administrators.

Functional Requirement: 13.1

Inputs: - Username
Expected Outputs: - Updated user
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Select Users from the System Administrator submenu.
3. Input the username into the username field under Edit Existing User.
4. Select the Edit Privileges link corresponding to the user.
5. Select the System Administrator checkbox.
6. Select Save.
7. Select Confirm.
8. Verify that the user has access to the System Administrator functions of the system such as editing user privileges (TC45a).

Test Case TC45b: Edit User Privileges

Functionality to Test: A system administrator shall be able to edit user privileges to make other users Course Leader or Course Assistant.

Functional Requirement: 13.1

Inputs: - Username
- Course Code

Expected Outputs: - Updated user
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Select Users from the System Administrator submenu.
3. Input the username into the username field under Edit Existing User.
4. Select the Edit Privileges link corresponding to the user.
5. Select the Course Leader or Course Assistant checkbox.
6. Input course code.
7. Select Save.
8. Select Confirm.
9. Verify that the user has access to Course Leader/Course Assistant functions such as registering results (TC36).

Test Case TC46: Add Course Assistant

Functionality to Test: A course leader shall be able to assign course assistants to courses they are responsible for.

Functional Requirement: 13.2

Inputs: - Username

Expected Outputs: - Added Course Assistant
- Confirmation

Instructions for Tester

1. Authenticate to the system (TC1).
2. Authenticate with the system.
3. Select Course Assistants from the Course Leader Links submenu.
4. Input the username.
5. Select Continue.
6. Select Confirm.
7. Verify that the user can now add results for students registered for the course (TC36).

Test Case TC47: Remove Course Assistant

Functionality to Test: A course leader shall be able to remove course assistants to courses they are responsible for.

Functional Requirement: 13.2

Inputs: - Course
- Username

Expected Outputs: - Confirmation

Instructions for Tester

1. Authenticate with the system.
2. Select Course Assistants from the Course Leader Links submenu.
3. Select the Remove link corresponding to the Course leader to remove.
4. Select Confirm.
5. Verify that the user no longer has access to Course Leader functions for the course, such as adding course assistants (TC46).

Test Case TC48: Add Course

Functionality to Test: A System Administrator shall be able to add courses to the system.

Functional Requirement: 13.3

Inputs: - Course Code

Expected Outputs: - New Course and Confirmation

Instructions for Tester

1. Authenticate with the system.
2. Select Courses from the System Administrator submenu.
3. Input the course code in the field corresponding to Add Course Code.
4. Select Add.
5. Select Confirm.
6. Verify that a Course Leader can now be assigned to the course (TC45b).

Test Case TC49: Edit Existing Course

Functionality to Test: A System Administrator shall be able to edit existing courses in the system.

Functional Requirement: 13.3

Inputs: - Course Code (old)
- Course Code (new)

Expected Outputs: - Updated Course and Confirmation

Instructions for Tester

1. Authenticate with the system.
2. Select Courses from the System Administrator submenu.
3. Input the course code in the field corresponding to Edit Course Code.
4. Select Search.
5. Select the Edit link corresponding to the course to be edited.
6. Input the new Course Code
7. Select Save.
8. Select Confirm.
9. Verify that a Course Leader can be assigned to the course using the new course code but not the old one (TC45b).