Use Cases – Project Scetris

David Bialik, Julian Fleischer, Hagen Mahnke,
Konrad Reiche, André Zoufahl

March 4, 2011

Contents

Category: Basic interface methods

1. User logs in

Actors	Any user
Scope	GUI
Precondition	The user is not logged in, thus known to the system as an anonymous user.
Postcondition	-
Postcondition on Success	The user is logged in and a welcome-page is shown. From that point on, every page informs the user about his current login status (i.e. that she is logged in) and displays his user name.
Basic Course of Events	**1.** The user navigates to the login-form. **2.** The user enters his login-credentials, i.e. his username (or email-address) and his password **3.** The user submits the login-form.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	• A login form may be present on all pages, whilst not logged in. It may for example be reachable via a menu.
Implementation Status	• Implemented.

2. User edits his own password

Actors	Any user
Scope	GUI
Precondition	The user is logged in.
Postcondition	When the action is performed the user is presented another page which informs him about the success of the change of this password. The user is still logged in.
Postcondition on Success	The password is stored in the database, such that on next login only the new password will be accepted.
Basic Course of Events	**1.** The user opens his personal preferences (a link should be available in the main navigation). **2.** Within the preferences there is a single form, that asks the user for his current password and the new pasword (which should be entered twice). **3.** The user submits the form.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

3. User edits his profile

Actors	Any user
Scope	
Precondition	
Postcondition	
Postcondition on Success	
Basic Course of Events	-
Alternative Paths	-
Open Questions	-
Solved issues	**Q:** Some attributes (like "Matrikel-Number") may not be edited by a user himself. Do we have a mechanism for that? **A:** No, we do not. The data model should be update by an attribute "user-editable".
Implementation Notes	-
Implementation Status	-

4. User displays the lectures he currently is enrolled in.

Actors	Any registered user
Scope	
Precondition	The user is logged in.
Postcondition	Anyway a nicely formatted page is shown (on failure, an error explaining what went wrong).
Postcondition on Success	The page is displayed. No state is changed.
Basic Course of Events	**1.** The user clicks on a link, e.g. "my lectures" **2.** A list of the lectures the user currently is enrolled in is to be displayed. The user should be able to click on them for a detail view (see appropriate use case). People having special rights should special options they are allowed to do with these entities.
Alternative Paths	-
Open Questions	**Q:** Do we have a "leave" mechanism? If so, a "leave" link should also be displayed.
Solved issues	**Q:** Do we have the information about what is a running Academic Term? **A:** Yes, the information can be deduced from the start/end-times and the current date.
Implementation Notes	-
Implementation Status	-

5. User views his timetable

Actors	Student, Lecturer, Tutor, GUI, Database, Any registered user
Scope	GUI
Precondition	The system is running and the user is logged in.
Postcondition	The system is still running and the user is still logged in.
Postcondition on Success	The users timetable is displayed.
Basic Course of Events	**1.** User clicks on MyCourses. **2.** The GUI requests the database for the students courses which the student is enrolled in.
Alternative Paths	**2a.** The GUI requests the database for the lecturers courses which the lecturer gives. **2b.** The GUI requests the database for the tutor courses which the tutor is enrolled in and the courses which the tutor gives.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

Category: Course Management

1. Program manager creates a new Course

Actors	Program Manager
Scope	GUI
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	A new Course is created
Basic Course of Events	**1.** The user navigates to the admin list of all courses. **2.** The user clicks a "create new" button at the top of the list. **3.** The user sees a form to enter details about the course: name, Courses that are required for this Course (from a list of existing courses) with a button to add more, required features (from a list of existing features) and their quantity with a button to add more, CourseAttributes (from a list of existing CourseAttributes) and their value with a button to add more, a year. **4.** The user submits the form. **5.** The Course is created. The user sees an empty "new Course" form.
Alternative Paths	**1a.** The user does not have permission to edit courses and sees a message "Permission denied" **5a.** The Course is not created due to invalid values and the form is presented again, containing the data that was entered and information about which values are defective. **5b.** A database exception occurs. The same form including the data is presented again plus a message stating the problem.
Open Questions	-
Solved issues	-
Implementation Notes	• Values for CourseAttributes are always Strings.
Implementation Status	• not implemented

2. Program manager edits a course

Actors	Program Manager, GUI, Database
Scope	
Precondition	The system is running and the user is logged in with the rights of a program manager.
Postcondition	The system is still running, the user is still logged in and previous course data is not lost due to exceptions.
Postcondition on Success	The course information is updated into the database.
Basic Course of Events	**1.** The user navigates to the administrator list of all courses. **2.** The user clicks an edit button beside the course the user wishes to edit. **3.** The GUI displays a form with details about the course, e.g. name, course elements, course instances, courses that are required for this course with a button to add more, required features and their quantity with a button to add more, course attributes abd their value with a button to add more and a year. **4.** The user edits the information. **5.** The user submits the form. **6.** The GUI updates the altered course information to the databasse. The user sees the admin list of all courses.
Alternative Paths	**1a.** The user does not have permission to edit courses and the GUI displays the message "Permission denied". **6a.** The Course information is not altered due to invalid values and the form is presented again, containing the data that was entered and information about which values are defective **6b.** A database exception occurs. The same form including the data is presented again plus a message stating the problem
Open Questions	-
Solved issues	-
Implementation Notes	• Values for CourseAttributes are always Strings.
Implementation Status	• not implemented

3. Program manager views the admin list of Courses

Actors	Program Manager
Scope	
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	The user sees the admin list of Courses
Basic Course of Events	**1.** The user navigates to the admin list of all Courses. **2.** The user navigates to the admin list of all Courses for a certain program. The page contains a button to create a new course at the top and a list of single letters for navigation to Courses that start with this letter. Below that are the courses (ordered lexicographically). Beside each Course are buttons to delete and edit
Alternative Paths	**1a.** The user does not have permission to see a list of Courses and sees a message "Permission denied"
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

4. Program manager deletes a Course

Actors	Program Manager
Scope	
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	The Course is deleted.
Basic Course of Events	**1.** The user navigates to the admin list of all Courses. **2.** The user clicks the delete button beside the Course he wishes to delete. **3.** The user sees a prompt to choose what shall happen to associated CourseInstances, CourseElements and CourseElementInstances. The choices are "delete all", "keep all" and "delete selected" as radio buttons. When selecting "delete marked", a list with CI, CE, CEI and a check-box for each becomes available. The prompt can be terminated by pressing "ok" or "abort", and "abort" is the default selection. **4.** The user makes his decision and presses "ok". **5.** The user sees a prompt, to ensure he wants to delete the Course. The choices are "yes", "no" and "abort" The default choice is "no". **6.** The user chooses "yes". **7.** The Course (and CI,CE,CEI if desired) is tagged as deleted. The user sees the admin list of all Courses.
Alternative Paths	**1a.** The user does not have permission to edit courses and sees a message "Permission denied" **7a.** The Course is not tagged as deleted due to a database exception. The user sees a prompt stating the problem and a "retry" and a "abort" button. The default is "retry". Every subsequent failed attempt brings up the same prompt.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

5. Program Manager or Main Lecturer views the detail view of a Course

Actors	Program Manager or Main Lecturer
Scope	
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	The user views the detail view of a Course.
Basic Course of Events	**1.** The user navigates to an admin list of Courses that contains the desired Course. **2.** The user clicks on the name of the Course. **3.** The user sees the detail view of that Course. It contains information: name, CourseElements, CourseInstances, Courses that are required for this course with a button to add more, required features and their quantity with a button to add more, CourseAttributes and their value with a button to add more, a year.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

6. Program manager creates a new CourseInstance

Actors	Program Manager
Scope	
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	A new CourseInstance is created
Basic Course of Events	**1.** The user navigates to the admin list of all CourseInstances. **2.** The user clicks a "create new" button at the top of the list. **3.** The user sees a form to enter details about the course: Course, program (from a list of programs), start and end date, main-lecturer (from a list of lecturers), CourseElementInstance (from a list of CEIs) with a button to add more. The form also contains a smart-copy option. Here a Course and a Program of which the Course was part can be chosen from a list and a button to copy those values into the form is present. **4.** The user enters the information manually or enters them via smart-copy (maybe adjusting manually). **5.** The user submits the form. **6.** The CourseInstance is created. The user sees an empty "new Course" form.
Alternative Paths	**1a.** The user does not have permission to edit CourseInstances and sees the public list of all CourseInstances. **6a.** The CourseInstance is not created due to invalid values and the form is presented again, containing the data that was entered and information about which values are defective **6b.** The CourseInstance is not created due to a database exception. The same form including the data is presented again plus a message stating the problem.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

7. Main lecturer or Program Manager edits a CourseInstance

Actors	Main Lecturer or Program Manager
Scope	
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	The information of a CourseInstance is altered.
Basic Course of Events	**1.** The user navigates to a (public or admin) list of all CourseInstances. **2.** The user clicks a CourseInstance. **3.** The user sees the detail view of that CourseInstance. **4.** The user clicks the the edit button. **5.** The user sees a form, containig the current values for that CourseInstance. **6.** The user edits the information and submits the form. **7.** The altered CourseInstance information is saved. The user sees the detail view of the CourseInstance.
Alternative Paths	**4a.** The user does not have permission to edit this CourseInstance and gets a view without the button. **5a.** The user does not have permission to edit this CourseInstance and sees a message stating that he lacks permission for that action. **7a.** The altered CourseInstance information is not saved due to invalid values and the form is presented again, containing the data that was entered and information about which values are defective **7b.** The altered CourseInstance information is not saved due to a database exception. The same form including the data is presented again plus a message stating the problem.
Open Questions	**Q:** Does our system support the restrictions stated above, that is especially a Main Lecturer may only edit his CourseInstances and a Program Manager may only edit CourseInstances of his program?
Solved issues	-
Implementation Notes	• A Program Manager may only edit CourseInstances from his Program. A Main Lecturer may only edit CourseInstances where he is assigned as Main Lecturer.
Implementation Status	• not implemented

8. A Program Manager deletes a CourseInstance

Actors	Program Manager
Scope	
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	The CourseInstance is tagged as deleted.
Basic Course of Events	**1.** The user navigates to the admin list of all CourseInstances. **2.** The user clicks the "delete" button right next to the CourseInstance she wishes to delete. **3.** The user sees a prompt asking whether associated CourseElementInstances shall be deleted as well. The Choices are "yes", "no" and "abort" and "yes" is the default. **4.** The user chooses "yes". **5.** The user sees a prompt asking to verify the deletion of the CourseInstance. The Choices are "yes", "no" and "abort" and "no" is the default. **6.** The user chooses "yes". **7.** The CourseInstance (and associated CEI if desired)is tagged as deleted. The user sees the admin list of all CourseInstances.
Alternative Paths	**3a.** The user does not have permission to edit that CourseInstance and sees a message stating that he lacks permission for that action. **7a.** The CourseInstance is not tagged as deleted due to a database exception. The user sees a prompt stating the problem and a "retry" and a "abort" button. The default is "retry". Every subsequent failed attempt brings up the same prompt.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

9. A user views a list of CourseInstances.

Actors	Any user
Scope	
Precondition	
Postcondition	
Postcondition on Success	The user sees the list of CourseInstances she desires.
Basic Course of Events	**1.** The user navigates to the public list of all CourseInstances. **2.** The user sees a list of all CourseInstances and (horizontal) lists for academic terms, departments and letters. Also a check-box for "My CourseInstances" is available. Beneath the lists is an "apply" button. **3.** The user makes his selection for a sub-list and presses "apply". **4.** The user sees the selected sub-list and the same selection possibilities as before, indicating the selection via color/bold font.
Alternative Paths	**1a.** The user is a Program Manager or Main Lecturer and sees the admin list of all CourseInstances. **2a.** >The user sees the public list and beside each CourseInstance are "edit" and "delete" buttons plus a "create new" button at the top of the page. **3a.** The user is satisfied with the list shown.
Open Questions	-
Solved issues	-
Implementation Notes	• Each entry in a list defines a subset of CourseInstances. For example applying the selection "Computer Science" will result in a list that contains all CourseInstances associated with computer science. Multiple selections are possible, selecting multiple values from one list will produce the union of the subsets. Selecting values from different lists will produce the intersection. Each list has "all" as an entry. All is the same as not selecting for that list. Selecting "My CourseInstances" will produce the subset of CourseInstances where the user is involved.
Implementation Status	• not implemented

10. A user views the detail view of a CourseInstance

Actors	Any user
Scope	
Precondition	
Postcondition	
Postcondition on Success	The user sees the detail view of the CourseInstance he is interested in.
Basic Course of Events	**1.** The user navigates to a list of CourseInstances that includes the desired CourseInstance (see use case "A user views a list of CourseInstances") **2.** The user clicks on the name of the CourseInstance **3.** The user sees the detail view of that CourseInstance, which includes: Main lecturer, academic term, department(s),recommended for year, description and (if already allocated) time and place or the associated CourseInstanceElements.
Alternative Paths	**3a.** The user has permission to edit this CourseInstance and also sees buttons to delete and edit.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

11. Program Manager creates a CourseElement

Actors	Program Manager
Scope	
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	A new CourseElement is created.
Basic Course of Events	**1.** The user navigates to the admin list of CEs. **2.** The user presses the "create new" button at the top of the list. **3.** The user sees a form to enter the information about the new CE. The fields include: The Course that this CE is associated with, CourseElementType (from a list of existing CETs), name, duration, whether the CE is mandatory, the features that are required and their quantity (with a button to add more). Of these fields only the name and the type can be null. **4.** The user submits the form **5.** The CourseElement is created. The user sees an empty "new CourseElement" form.
Alternative Paths	**5a.** The CourseElement is not created due to invalid values and the form is presented again, containing the data that was entered and information about which values are defective. **5b.** A database exception occurs. The same form including the data is presented again plus a message stating the problem.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

12. Program Manager edits a CourseElement

Actors	Program Manager
Scope	
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	A CourseElement is altered.
Basic Course of Events	**1.** The user navigates to the admin list of CEs. **2.** The user presses the "edit" button beside the CE. **3.** The user sees a form containing the current information about the CE. The fields include: The Course that this CE is associated with, CourseElementType (from a list of existing CETs), name, duration, whether the CE is mandatory, the features that are required and their quantity (with a button to add more). Of these fields only the name and the type can be null. **4.** The user edits the values. **5.** The user submits the form. **6.** The CourseElement is altered. The user sees the detail view of the CE.
Alternative Paths	**6a.** The CourseElement is not altered due to invalid values and the form is presented again, containing the data that was entered and information about which values are defective. **6b.** A database exception occurs. The same form including the data is presented again plus a message stating the problem.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

13. Program Manager deletes a CourseElement

Actors	Program Manager
Scope	
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	A CourseElement is deleted.
Basic Course of Events	**1.** The user navigates to the admin list of CEs. **2.** The user presses the "delete" button beside the CE. **3.** The user sees a prompt asking whether associated CourseElementInstances shall be deleted. The choices are "yes", "no" and "abort". Default is yes. **4.** The user chooses "yes". **5.** The user sees a prompt to confirm the deletion of the CE. The choices are "yes" and "no", "no" being default. **6.** The user chooses "yes". **7.** The CourseElement (and associated CEI if desired)is tagged as deleted. The user sees the admin list of all CourseElements.
Alternative Paths	**7a.** The CourseElement is not tagged as deleted due to a database exception. The user sees a prompt stating the problem and a "retry" and a "abort" button. The default is "retry". Every subsequent failed attempt brings up the same prompt.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

14. A Program Manager views an admin list of CourseElements.

Actors	Program Manager
Scope	
Precondition	
Postcondition	
Postcondition on Success	The user sees the admin list of CourseElements she wants.
Basic Course of Events	**1.** The user navigates to the admin list of all CourseElements. **2.** The user sees a list of all CourseElements and (horizontal) lists for departments and letters. Also a textfield to search for CEs belonging to a specific Course is available. Beneath the lists is an "apply" button. Beside each CourseElement are "edit" and "delete" buttons plus a "create new" button at the top of the page. **3.** The user makes his selection for a sub-list and presses "apply". **4.** The user sees the selected sub-list and the same selection possibilities as before, indicating the selection via color/bold font.
Alternative Paths	**3a.** The user is satisfied with the list shown.
Open Questions	-
Solved issues	-
Implementation Notes	• Each entry in a list defines a subset of CourseElements. For example applying the selection "Computer Science" will result in a list that contains all CourseElementss associated with computer science. Multiple selections are possible, selecting multiple values from one list will produce the union of the subsets. Selecting values from different lists will produce the intersection. Each list has "all" as an entry. All is the same as not selecting for that list. Entering a name of a Course in the textfield will produce the subset of CourseElements that are asscociated with that Course.
Implementation Status	• not implemented

15. A Program Manager views the detail view of a CourseElement

Actors	Program Manager
Scope	
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	The user views the detail view of a CourseElement.
Basic Course of Events	**1.** The user navigates to an admin list of CourseElements containing the desired CE. **2.** The user clicks the name of the CE. **3.** The user sees the detail view of the CE. Information about the Course that this CE is associated with, CourseElementType (from a list of existing CETs), name, duration, whether the CE is mandatory, the features that are required and their quantity (with a button to add more) is displayed. Buttons to "edit" or "delete" this CE are also available.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

16. Main lecturer creates a CourseElementInstance

Actors	Main lecturer
Scope	
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	A new CourseElementInstance is created.
Basic Course of Events	**1.** The user navigates to the detail view of a Course. **2.** The user presses the "create new instance" button beside a CourseElement. **3.** The user sees a form to enter the information about the new CEI. The fields include: The CourseInstance that this CEI is associated with from a list of CIs (default is the most recent CourseInstance), lecturer, duration. Of these fields only the lecturer can be null. **4.** The user submits the form **5.** The CourseElementInstance is created. The user sees the detail view of the Course.
Alternative Paths	**5a.** The CourseElementInstance is not created due to invalid values and the form is presented again, containing the data that was entered and information about which values are defective. **5b.** A database exception occurs. The same form including the data is presented again plus a message stating the problem.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

17. Main Lecturer edits a CourseElementInstance

Actors	Main Lecturer
Scope	
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	The CEI is altered.
Basic Course of Events	**1.** The user navigates to an admin list of CEIs. **2.** The user presses the "edit" button beside the CEI. **3.** The user sees a form containing the current information about the CEI. The fields include: The CourseInstance that this CEI is associated with, the CourseElement that this CEI is an instance of, lecturer, duration. Of these fields only the lecturer can be null. **4.** The user edits the values. **5.** The user submits the form. **6.** The CourseElementInstance is altered. The user sees an admin list of CEIs.
Alternative Paths	**6a.** The CourseElementInstance is not altered due to invalid values and the form is presented again, containing the data that was entered and information about which values are defective. **6b.** A database exception occurs. The same form including the data is presented again plus a message stating the problem.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

18. Main Lecturer deletes a CourseElementInstance

Actors	Main Lecturer
Scope	
Precondition	The user is logged in.
Postcondition	
Postcondition on Success	A CourseElementInstance is deleted.
Basic Course of Events	**1.** The user navigates to an admin list of CEIs containing the desired CEI. **2.** The user presses the "delete" button beside the CEI. **3.** The user sees a prompt to confirm the deletion of the CEI. The choices are "yes" and "no", "no" being default. **4.** The user chooses "yes". **5.** The CourseElement is tagged as deleted. The user sees the admin list of all CEI.
Alternative Paths	**5a.** The CEI is not tagged as deleted due to a database exception. The user sees a prompt stating the problem and a "retry" and a "abort" button. The default is "retry". Every subsequent failed attempt brings up the same prompt.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

19. A user views a list of CourseElementInstances.

Actors	Any user
Scope	
Precondition	
Postcondition	
Postcondition on Success	The user sees list of CEIs she wants.
Basic Course of Events	**1.** The user navigates to the list of all CEIs. **2.** The user sees a list of all CEIS and (horizontal) lists for academic terms, departments and letters. Also a textfield to search for CEIs belonging to a specific Course is available. Beneath the lists is an "apply" button. If the user has permission to edit CEIs there are "edit" and "delete" buttons beside each CEI plus a "create new" button at the top of the page. **3.** The user makes his selection for a sub-list and presses "apply". **4.** The user sees the selected sub-list and the same selection possibilities as before, indicating the selection via color/bold font.
Alternative Paths	**3a.** The user is satisfied with the list shown.
Open Questions	-
Solved issues	-
Implementation Notes	• Each entry in a list defines a subset of CEIs. For example applying the selection "Computer Science" will result in a list that contains all CEIs associated with computer science. Multiple selections are possible, selecting multiple values from one list will produce the union of the subsets. Selecting values from different lists will produce the intersection. Each list has "all" as an entry. All is the same as not selecting for that list. Entering a name of a Course in the textfield will produce the subset of CEIs that are asscociated with that Course.
Implementation Status	• not implemented

20. A user views the detail view of a CourseElementInstance

Actors	Any user
Scope	
Precondition	
Postcondition	
Postcondition on Success	The user views the detail view of a CEI.
Basic Course of Events	**1.** The user navigates to a list of CEIs containing the desired CEI. **2.** The user clicks the name of the CEI. **3.** The user sees the detail view of the CEI. Information includes: The CourseInstance that this CEI is associated with, the CourseElement that this CEI is an instance of, duration. If the user has permisson to edit CEIs, buttons to "edit" or "delete" this CEI are also available.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

21. A course element is removed from a course

Actors	Program Manager, GUI, Database
Scope	GUI
Precondition	The system is running, the user is still logged in.
Postcondition	
Postcondition on Success	The course element is deleted from the database.
Basic Course of Events	**1.** User opens the course element panel. **2.** User selects a course elements. **3.** User clicks on submit. **4.** GUI deletes the course element from the datbase.
Alternative Paths	-
Open Questions	**Q:** What happens with already instantiated course elements (which we call course element instance)?
Solved issues	-
Implementation Notes	-
Implementation Status	-

22. Create a Course Element Types

Actors	Program Manager, GUI, Database
Scope	GUI
Precondition	The system is running, the user is still logged in.
Postcondition	
Postcondition on Success	The new course element type is insirted into the database.
Basic Course of Events	**1.** User opens the course element type panel. **2.** User enters the name of the course element type. **3.** User clicks on submit **4.** GUI inserts the new course element type into the database.
Alternative Paths	**4a.** GUI tries to insert the new course element type into the database. **4b.** Database throws a conflict on the insertion, because a course element type of this name already exists. **4c.** GUI displays the exception.
Open Questions	-
Solved issues	**Q:** What's the use for course element types? Do we need or want them at all? **A:** Course Element types have no functional use, for instance for the scheduler or similiar. Instead it is just an information about the type of the course element.
Implementation Notes	-
Implementation Status	-

23. Program Manager creates a new program

Actors	Program Manager, Database, GUI
Scope	GUI
Precondition	The System is running and the user is logged in.
Postcondition	The System is still running and the user is still logged in.
Postcondition on Success	A new program is inserted into the database.
Basic Course of Events	**1.** User opens the program panel. **2.** User chooses to create a new program. **3.** User selects the department of the new program. **4.** User selects the academic term of the new program. **5.** User clicks on submit. **6.** GUI inserts the new program.
Alternative Paths	-
Open Questions	-
Solved issues	**Q:** What is the difference between the creation of a program and an academic term? Should a program automatically be generated per department when an Academic Term is created? **A:** Yes, they should. It might be an unnecessary step.
Implementation Notes	• The program creation was implemented very differently. We have come to realize the program creation is a starting point for creating alot of entities which are dependent to the program, for instance course element instances. Therefore the program creation is now a guided process of chained forms.
Implementation Status	-

24. Admin creates a new Academic Term

Actors	Program Manager, Database, GUI
Scope	GUI
Precondition	The System is running and the user is logged in.
Postcondition	The System is still running and the user is still logged in.
Postcondition on Success	A new academic term is inserted into the database.
Basic Course of Events	**1.** User opens the academic term panel. **2.** User chooses to create a new academic term. **3.** User the name of the new academic term. **4.** User clicks on submit. **5.** GUI inserts the new academic term into the database.
Alternative Paths	-
Open Questions	**Q:** How exactly does smarty copy work? How are starting/ending dates resolved? How Course Element Instances and main-lecturers? **Q:** Should this use case be splitted up into two use cases: "create AT", "smarty create AT"?
Solved issues	-
Implementation Notes	• Smarty copy.
Implementation Status	-

25. Edit an Academic Term

Actors	Program Manager, Database, GUI
Scope	GUI
Precondition	The System is running and the user is logged in.
Postcondition	The System is still running and the user is still logged in.
Postcondition on Success	The academic is altered.
Basic Course of Events	**1.** User opens the academic term panel. **2.** User chooses to edit an academic term. **3.** User changes the name of the new academic term. **4.** User clicks on submit. **5.** GUI updates the new academic term into the database..
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

Category: Enrollment

1. User enrolls in a course

Actors	Student, GUI, Database
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of a student.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The user is enrolled in the chosen courses.
Basic Course of Events	**1.** User open course panel. **2.** User selects a course instance from a list of course instances. **3.** User clicks on submit. **4.** GUI displays details about the course instance data, e.g. main lecturer, lesson period, different course element instances **5.** User selects varies course element instances. **6.** User clicks on submit. **7.** GUI inserts data into the database with the association between person to course instance and person to course element instance.
Alternative Paths	-
Open Questions	-
Solved issues	**Q:** Do we provide a grading system, e.g. checking if enough course element instances were chosen? **A:** No, but the relational schema allows the implementation of checks. A complete grading system could be implemented in
Implementation Notes	• The implementation is based on the relation CourseElementInstance and not on the ProposedScheduling.
Implementation Status	-

Category: Features

1. Create feature

Actors	user with the right to create a feature
Scope	
Precondition	The system is up and running, the user who wants to create a new feature is logged in.
Postcondition	
Postcondition on Success	The new feature is created
Basic Course of Events	**1.** User navigates to the "create feature" page. **2.** A form will be presented to the user in which the user can enter a name for the new feature. **3.** The form will be submitted. **4.** The new feature is created. A new empty form will be presented to the user in order to enter another new feature.
Alternative Paths	**1a.** Without needed rights, the user wont get to this page. **4a.** The new feature is not created, due to malformed data (bad syntax, invalid values). In this case the same form with hints on what went wrong should be displayed again. **4b.** A database exception occurs. The user should be informed about it, the data should not be lost, i.e. the same form should be shown again (maybe in a way that it can easily be retried).
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

2. Edit feature

Actors	user with the right to edit features
Scope	
Precondition	The system is up and running, the user who wants to edit a feature is logged in.
Postcondition	The feature will not be lost or deleted in case of an error while changing the data of the feature.
Postcondition on Success	The feature will have its data legally changed.
Basic Course of Events	**1.** The user selects a feature for editing. From the feature list. **2.** The edit-feature form is shown, with the current name filled in. **3.** The user changes the name of the feature. **4.** The entered name is checked whether it already exists before submitting. **5.** If the renaming was legal (e.g. no newer record has been saved in between) the new name will be written to the database.
Alternative Paths	**1a.** The user does not has sufficient rights for editing. In lists the entry edit feature should be missing. **2a.** Show an error message, explaining the missing rights, if the user does not have the needed rights.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

3. Delete feature

Actors	User with the right to delete features
Scope	
Precondition	The system is up and running, the user who wants to delete a feature is logged in.
Postcondition	
Postcondition on Success	The feature will be deleted.
Basic Course of Events	**1.** The user selects a feature to delete. From list view or similar. **2.** A confirmation page is shown, asking if the room should be really deleted. **3.** The feature will be deleted. **4.** The user will land where he left off for his deleting action.
Alternative Paths	**2a.** A message page is shown indicating that and where the feature is currently in use.
Open Questions	**Q:** maybe it would be nice to display the delete option only on features which *are* currently deletable?
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

4. List features

Actors	User with rights to change features
Scope	
Precondition	The system is up and running, the user is logged in.
Postcondition	no changes will be made to the database in this view
Postcondition on Success	presentation of features
Basic Course of Events	**1.** User navigates to the "list features" page **2.** User can specify his search for features by name. **3.** User gets a list of features fulfilling his search criteria.
Alternative Paths	-
Open Questions	**Q:** Should there be a puplic view of features?
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

Category: Public listings

1. A User displays list of lectures

Actors	Any user (anonymous & logged in)
Scope	
Precondition	The system is running.
Postcondition	-
Postcondition on Success	A page displaying an alphabetically sorted list, grouped by the first letter, is displayed.
Basic Course of Events	**1.** User clicks on a link "lectures". **2.** An alphabetically sorted list of all lectures is shown. **3.** Using a form on that page, a lecturer can be searched or the current view narrowed (filtered). **4.** The filter-form is submitted.
Alternative Paths	**2a.** If too many lectures were to be shown, a multi-page is generated. **2b.** If no lectures match the given filters (or there are none), a message like "no lectures match the given filters" is shown. **2c.** The user may equally well stop here. **3a.** A preset filter may be selected: "Display lectures recommended for my Year" (if the user is logged in). This should only display courses which are available (i.e. there are Course Instances in the running Academic Term) and subject of "recommendedForYear" with the Year of the currently logged in user. **3b.** A preset filter may be selected: "Display lectures at my home institute only" (if the user is logged in).
Open Questions	**Q:** Is the information about the current year of a student stored somewhere? **Q:** Is the information about a "home institute" stored somewhere?
Solved issues	-
Implementation Notes	-
Implementation Status	-

2. Detail view for a lecture

Actors	Any user
Scope	GUI
Precondition	The system is running and the user is logged in.
Postcondition	The system is still running and the user is still logged in.
Postcondition on Success	A page displaying information about a lecture is shown. This includes which Course Element Instances happen when & where, who is the main lecturer, and Course Attributes. If the currently logged in user has sufficient rights, links for editing/deleting the currently shown resource may be displayed, too.
Basic Course of Events	**1.** User opens the user panel. **2.** User enters the name of a lecturer. **3.** User clicks on submit. **4.** GUI displays a link to the lecturer. **5.** User clicks on the link. **6.** GUI requests user data from the database. **7.** GUI displays the lecturer data.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

3. Anonymous user displays list of lecturers

Actors	Any user (anonymous & logged in)
Scope	
Precondition	The system is running.
Postcondition	-
Postcondition on Success	A page displaying an alphabetically sorted list, grouped by the first letter, is displayed.
Basic Course of Events	**1.** User clicks on a link "people". **2.** An alphabetically sorted list of all lecturers is shown. **3.** Using a form on that page, a lecturer can be searched or the current view narrowed (filtered), e.g. "view only lecturers from the institute for computer science". **4.** The filter-form is submitted.
Alternative Paths	**2a.** If no lecturers match the given filters (or there are none), a message like "no data set matches the given filters" is shown. **2b.** If too many lecturers were to be shown, a multi-page is generated. **2c.** The user may equally well stop here.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

Category: Rooms

1. Create room

Actors	administrator or program administrator
Scope	
Precondition	The system is up and running, the user who wants to create a new room is logged in.
Postcondition	
Postcondition on Success	The new room is created
Basic Course of Events	**1.** User navigates to the "create room" page. **2.** A form will be presented to the user in which the user can enter data about the room to create: name, number, building(from list of existent buildings), preferred timeslot, provided features(from list or new one) together with quantity via form field that only accepts positive integers **3.** The form will be submitted. **4.** The new room is created. A new empty form will be presented to enter data about the next room.
Alternative Paths	**1a.** Without needed rights, the user should be sent to the "public room list". **4a.** The new room is not created, due to malformed data (bad syntax, invalid values). In this case the same form with hints on what went wrong should be displayed again. **4b.** A database exception occurs. The user should be informed about it, the data should not be lost, i.e. the same form should be shown again (maybe in a way that it can easily be retried).
Open Questions	**Q:** how to select the preferred timeslot(list, ...)? **Q:** how are the quantities of features saved? It is possible that a feature is restricted to have only the values 0 and 1(then boolean checkboxed would be preferable for this kind of feature)?
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

2. Edit a room

Actors	administrator or program administrator
Scope	
Precondition	The system is up and running, the user who wants to edit a room is logged in.
Postcondition	The room will not be lost or deleted in case of an error while changing the data of the room.
Postcondition on Success	The room will have its data legally changed.
Basic Course of Events	**1.** The user selects a room for editing. From list or similar. **2.** The edit-room form is shown, with the current values of the room filled in. **3.** The user changes the values he wish to change. May add features from a list of existing ones or create a new feature. **4.** The entered data are checked before they are submitted. If the quantity of a feature was changed to zero it should be deleted from the room. **5.** If the informations are not outdated (e.g. a newer record has been saved in between), the new values are written to the database.
Alternative Paths	**1a.** The user does not has sufficient rights for editing. In lists the entry edit should be missing. **2a.** Show an error message, explaining the missing rights, if the user does not have the needed rights. **5a.** If the informations *are* outdated, they should not be written to the database. However, the information should not get lost. Thus is would be best to display the form whose values could not be updated again. **5b.** It would be even better to show the differences and assist in merging the two (Wikipedia does have such a feature for so-called "edit conflicts").
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

3. Delete a room

Actors	administrator or program administrator
Scope	
Precondition	The system is up and running, the user who wants to delete a room is logged in.
Postcondition	The room will not be deleted in parts and only if not referred to somewhere.
Postcondition on Success	The room will be deleted.
Basic Course of Events	**1.** The user selects a room to delete. From list or similar. **2.** A confirmation page is shown, asking if the room should be really deleted. **3.** The room will be deleted. **4.** The user will be shown where he came from. Probably a list view of rooms.
Alternative Paths	**1a.** The user does not has sufficient rights for deleting. In lists the entry delete should be missing. **2a.** A message page is shown, that the page is referred to somewhere and where.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

4. List rooms

Actors	User with or without the right to change rooms
Scope	
Precondition	The system is up and running, the user with the right to change rooms is logged in.
Postcondition	no changes will be made to the database
Postcondition on Success	presentation of rooms
Basic Course of Events	**1.** User navigates to the "list rooms" page. **2.** User can specify his search for rooms by using different things. **3.** User will get a list of rooms fulfilling his search.
Alternative Paths	**3a.** User with rights to change rooms will get a list of rooms fulfilling his search with links to edit and delete
Open Questions	**Q:** What should be supported here? All features, name, department, number, ... **Q:** Should all the data of the searched rooms be shown in this table, would make a detail view unnecessary?
Solved issues	-
Implementation Notes	-
Implementation Status	• not implemented

Category: Scheduling (both collaborative and automatic)

1. Program admin starts scheduling

Actors	Program administrator, GUI, Scheduler, Database
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of a program administrator.
Postcondition	The system is still running, the user is still logged in and if there was a previous proposed scheduling it is not lost due to exceptions.
Postcondition on Success	The proposed schedule is inserted or updated into the database.
Basic Course of Events	**1.** User opens schedule panel. **2.** User defines a program to schedule. **3.** User clicks on start. **4.** Scheduler launches the scheduling of the defined program. **5.** Scheduler changes its status to running. **6.** User waits until the scheduling is finished by itself. **7.** Scheduler inserts the proposed schedule into the database. **8.** Scheduler changes its status to ready.
Alternative Paths	**6a.** User clicks on stop. **6b.** Scheduler changes its status to stopping. **7a.** Scheduler updates the proposed schedule in the database.
Open Questions	**Q:** Should a program administrator only be enabled to schedule his own program? **Q:** Should an administrator also be enabled to schedule a program?
Solved issues	**Q:** Should the program administrator get a notifaction on his next login? **A:** Yes. **Q:** Will there be scheduling proposals for conctlict resolutions? **A:** No, but the user will be informed about the reasons of the conflicts
Implementation Notes	-
Implementation Status	• Program administrator notification is missing.

2. Pause the scheduling process

Actors	Program administrator, GUI, Scheduler, Database
Scope	GUI
Precondition	The system is running, the scheduling process has been started, it is running and the user is still logged in with the rights of a program administrator.
Postcondition	The system is still running, the user is still logged in and if the pausing did not take action yet the scheduling status is set to stopping.
Postcondition on Success	The scheduling process is paused.
Basic Course of Events	**1.** User opens schedule panel. **2.** User clicks on stop. **3.** Scheduler changes its status to stopping. **4.** Schedulers inserts current proposed schedule into the database. **5.** Scheduler changes its status to ready.
Alternative Paths	**4a.** Scheduler updates current proposed schedule in the database.
Open Questions	-
Solved issues	-
Implementation Notes	• The Java thread interruption policy is used for implementation.
Implementation Status	-

3. Resume a scheduling process

Actors	Program administrator, GUI, Scheduler, Database
Scope	GUI
Precondition	The system is running, the scheduling process is ready and the user is still logged in with the rights of a program administrator.
Postcondition	The system is still running, the user is still logged in and if there was no proposed schedule the scheduling process starts as a new scheduling.
Postcondition on Success	The resumed scheduling starts with the previous number of resolved constraints.
Basic Course of Events	**1.** User opens schedule panel. **2.** User clicks on resume. **3.** Scheduler launches the scheduling of the defined program. **4.** Scheduler changes its status to running.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	• Resume is implemented by using the current proposed schedule of each course to reallocate their position in the scheduler internal presentation of rooms, time slots and courses.
Implementation Status	-

4. A conflict occurs

Actors	Program administrator, GUI, Scheduler, Database
Scope	GUI
Precondition	The system is running, the user is logged in with the rights of a program administrator and the scheduling was started.
Postcondition	The system is still running, the user is still logged in and if there was a previous proposed scheduling it is not lost due to the conflict.
Postcondition on Success	The reasons for the conflicts are displayed.
Basic Course of Events	**1.** Scheduler spots a non-solveable conflict while scheduling. **2.** Scheduler notfies the systems about the conflict. **3.** Scheduler terminates the scheduling process. **4.** User opens schedule panel. **5.** GUI displays the reason of the conflict.
Alternative Paths	-
Open Questions	**Q:** The scheduler uses exceptions to inform the system about conflicts. Will the exceptions be saved to display them until the user opens the schedule panel again?
Solved issues	**Q:** Should the current proposed schedule of the scheduling process be inserted or updated into the database? **A:** No, as the current proposed schedule is incomplete.
Implementation Notes	-
Implementation Status	-

5. Change the proposed schedule of a Course Element Instance

Actors	Program administrator, Main lecturer, GUI, Scheduler, Database
Scope	GUI
Precondition	The system is running and user is logged in with the rights of a program administrator or main lecturer.
Postcondition	The system is still running, the user is still logged in and if there was a previous proposed schedule it is not lost due to exceptions.
Postcondition on Success	The proposed schedule is inserted or updated into the database.
Basic Course of Events	**1.** User opens the timetable of a room. **2.** User moves the Course Element Instance to its new time slot. **3.** User clicks on submit. **4.** Scheduler calculates the score and potential conflicts. **5.** GUI displays potential conflicts. **6.** User clicks on accept. **7.** Scheduler inserts the new proposed schedule into the database.
Alternative Paths	**3a.** User clicks on discard. **3b.** GUI reverses the changes and displays the original state. **6a.** User clicks on discard. **6b.** GUI reverses the changes and displays the original state. **7a.** Scheduler updates the existing proposed schedule in the database.
Open Questions	-
Solved issues	**Q:** Should the program administrator get a notification on his next login? **A:** Yes. **Q:** Will there be scheduling proposals for conflict resolutions? **A:** No, but the user will be informed about the reasons of the conflicts.
Implementation Notes	-
Implementation Status	-

6. Automatically reschedule a subset of courses

Actors	Program administrator, GUI, Scheduler, Database
Scope	GUI
Precondition	The system is running, the user is logged in with the rights of a program administrator and there is no running scheduling process.
Postcondition	The system is still running, the user is still logged in and if there was a previous proposed schedule it is not lost due to exceptions.
Postcondition on Success	The proposed schedule is inserted or updated into the database.
Basic Course of Events	**1.** User opens the course panel. **2.** User defines a program **3.** User selects the course element instances which belong to the subset of courses. **4.** User clicks on schedule. **5.** System updates the course element instances attribute scheduleable lesson of those course element instances which were not selected but belong to the same program to false. **6.** Scheduler launches the scheduling of the defined program. **7.** Scheduler changes its status to running. **8.** User waits until the scheduling is finished by itself. **9.** Scheduler inserts the proposed schedule into the database. **10.** Scheduler changes its status to ready.
Alternative Paths	**8a.** User clicks on stop. **8b.** Scheduler changes its status to stopping. **9a.** Scheduler updates the proposed schedule in the database.
Open Questions	**Q:** Should the subset scheduling of courses be offered as a method of the scheduler?
Solved issues	-
Implementation Notes	-
Implementation Status	-

7. Define a program

Actors	Program administrator, GUI, Scheduler, Database
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of a program administrator.
Postcondition	The system is still running and the user is still logged in.
Postcondition on Success	The system finds a program fitting for the specified academic term and department.
Basic Course of Events	**1.** User opens the schedule panel. **2.** User selects an academic term from the available academic terms. **3.** User selects a department from the available departments. **4.** User clicks on start. **5.** System queries the database for a program with the given academic term and the given department. **6.** Database returns the required program.
Alternative Paths	**4a.** User clicks on resume.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

Category: System management

1. Create privileges

Actors	Program Administrator, GUI
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	A new privilege is inserted into the database.
Basic Course of Events	**1.** User opens the privilege panel. **2.** User clicks on new privilege. **3.** User enters the name for the new privilege. **4.** User clicks on submit. **5.** GUI inserts the new privilege into the database.
Alternative Paths	**3a.** User enters many privileges in the extended text field.
Open Questions	-
Solved issues	-
Implementation Notes	• The privileges can be created in the web interface, however only privileges created by the installer have an effect..
Implementation Status	• Implemented.

2. Delete privileges

Actors	Program Administrator, GUI
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The selected privilege is deleted from the database.
Basic Course of Events	**1.** User opens the privilege panel. **2.** User clicks on delete privilege. **3.** User selects a privilege. **4.** User clicks on submit. **5.** GUI deletes the privilege from the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

3. Privileges detail view

Actors	Program Administrator, GUI
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The GUI displays the privilege and its details.
Basic Course of Events	**1.** User opens the privilege panel. **2.** User clicks on "all privileges" **3.** GUI requests data from the database **4.** GUI displays the data from the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

4. Edit privileges

Actors	Program Administrator, GUI
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The privilege is altered.
Basic Course of Events	**1.** User opens the privilege panel. **2.** User clicks on edit privilege. **3.** User selects a privilege. **4.** User changes the privilege name. **5.** User clicks on submit. **6.** GUI updates the privilege into the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

5. Create Attribute regarding Persons

Actors	Program Administrator, GUI
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The attribute is created.
Basic Course of Events	**1.** User opens the attribute panel. **2.** User clicks on create attribute. **3.** User enters the name of the attribute. **4.** User enters the type of the attribute. **5.** User chooses the attribute to be a unique attribute. **6.** GUI inserts the attribute into the database.
Alternative Paths	**5a.** User chooses the attribute to be not a unique attribute.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

6. Edit Attribute regarding Persons

Actors	Program Administrator, GUI
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The attribute is altered.
Basic Course of Events	**1.** User opens the attribute panel. **2.** User clicks on edit attribute. **3.** User changes the data of the attribute. **4.** User clicks on submit. **5.** GUI updates the attribute into the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

7. Delete Attribute regarding Persons

Actors	Program Administrator, GUI
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The selected attribute is deleted from the database.
Basic Course of Events	**1.** User opens the attribute panel. **2.** User clicks on delete attribute. **3.** User selects an attribute. **4.** User clicks on submit. **5.** GUI deletes the attribute from the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

8. Create Course Attribute

Actors	Program Administrator, GUI
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The course attribute is created.
Basic Course of Events	**1.** User opens the course attribute panel. **2.** User clicks on create course attribute. **3.** User enters the name of the course attribute. **4.** User enters the type of the course attribute. **5.** User chooses the course attribute to be a unique attribute. **6.** GUI inserts the course attribute into the database.
Alternative Paths	**5a.** User chooses the course attribute to be not a unique attribute.
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

9. Edit Course Attribute

Actors	
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The course attribute is altered.
Basic Course of Events	**1.** User opens the course attribute panel. **2.** User clicks on edit course attribute. **3.** User changes the data of the course attribute. **4.** User clicks on submit. **5.** GUI updates the attribute into the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

10. Delete Course Attribute

Actors	Program Administrator, GUI
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The selected attribute is deleted from the database.
Basic Course of Events	**1.** User opens the course attribute panel. **2.** User clicks on delete course attribute. **3.** User selects a course attribute. **4.** User clicks on submit. **5.** GUI deletes the course attribute from the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

Category: User management

1. Admin creates a new user account

Actors	A user with sufficient rights (typically called admin)
Scope	
Precondition	The system is up and running, the user who wants to create a new user account is logged in
Postcondition	
Postcondition on Success	The new user account is created
Basic Course of Events	**1.** The user navigates to the "create user" page. **2.** A form is displayed which offers to create a new user, including first-name, middle-name, last-name, login-name, email-address, password, whether it is a student or a lecturer, etc. Most importantly it should also be possible to select roles and privileges from a list to assign to the user. **3.** The form is submitted. **4.** The new user account is created. The user is shown another empty form for the creation of another user.
Alternative Paths	**1a.** With insufficient rights, the user should not be able to navigate there at all. **4a.** The new account is not created, due to malformed data (bad syntax, invalid values). In this case the same form with hints on what went wrong should be displayed again. **4b.** A database exception occurs. The user should be informed about it, the data should not be lost, i.e. the same form should be shown again (maybe in a way that it can easily be retried).
Open Questions	**Q:** Should we extend a user account by a "home department"?
Solved issues	**Q:** Should we extend a user account by the option of making him a super-user (super-user do have all rights) **A:** Yes.
Implementation Notes	-
Implementation Status	• Implemented, except for checking of rights.

2. Edit another user account

Actors	A user with sufficient rights
Scope	
Precondition	The system is up and running, the user who wants to edit is logged in.
Postcondition	
Postcondition on Success	The user account to be edited gets updated by the values provided by the editing user. A page informaing about the success is shown.
Basic Course of Events	**1.** The user selects a user account for editing. This is typically achived by clicking an "edit"-link in an appropriate list (search result, user listing, etc.). **2.** The edit-user form is shown, with the current account properties filled in. However, certain special fields, i.e. "password" are specially protected. If the user does also have the right to edit those fields, these fields should also be shown. If the user does not, these should simply not be shown at all. **3.** The user changes the values. **4.** The form is checked by the application according to the definition set up for it (see Forms). **5.** If the information is not outdated (e.g. a newer record has been saved in between), the new values are to be written to the database.
Alternative Paths	**1a.** The user does not have sufficient rights to edit another users account. Edit links should not be displayed at all then. **2a.** If the user has gotten *somehow* to that page (e.g. by copying a link) and does not have sufficient rights to do so, a page that explains that the user does not have sufficient rights should be shown. **5a.** If the information *is* outdated, it should not be written to the database. However, the information should not get lost. Thus is would be best to display the form whose values could not be updated again. **5b.** It would be even better to show the differences and assist in merging the two (Wikipedia does have such a feature for so-called "edit conflicts").
Open Questions	-
Solved issues	-
Implementation Notes	• A transaction should be used to save the values. • The "timekey" attribte has special support to identify outdated records. • Currently there are two privileges "edit_user_accounts", "edit_user_passwords".
Implementation Status	• Implement, rights are party respected already.

3. Create a Role

Actors	A user with sufficient rights
Scope	
Precondition	The system is up and running, the user who wants to create the role is logged in.
Postcondition	
Postcondition on Success	The new role is created.
Basic Course of Events	**1.** The user navigates to the "create role" form. **2.** The user enters the name of the role and selects privileges which should be associated with this role. **3.** The user submits the form. **4.** The system displays the successful creation of the role and association of privileges.
Alternative Paths	**4a.** If a role with the same name already exists, the role is not created but a hint on the issue is shown. **4b.** If some error occurs (like connection to the database is lost), a message informing the user about the error is shown. **4c.** The system detects that the user does not have sufficient rights to create a role. While a user should not be able to navigate to pages he does not have the right to access, it is (in corner-cases) possible to happen so; for example if a user navigated to such a page and the right to use that page is revoked while he is working on that form.
Open Questions	-
Solved issues	-
Implementation Notes	• Creation of the role and creation of the roleImpliesPrivilege-relations should happen within a single transaction. If one of these fails, the users request can not be completed as a whole and therefor nothing should be done at all. This way we do not have half-baked relations within the database.
Implementation Status	-

4. Edit a Role

Actors	
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The role is altered.
Basic Course of Events	**1.** User opens the role panel. **2.** User clicks on edit role. **3.** User changes the data of the role. **4.** User clicks on submit. **5.** GUI updates the role into the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

5. Assign a Privilege to a Role

Actors	Administrator
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The role has a new privilege assigned.
Basic Course of Events	**1.** User opens the role panel. **2.** User clicks on edit role. **3.** User selects a privilege. **4.** User clicks on submit. **5.** GUI inserts the new association between privilege and role into the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

6. Assign an Attribute to a Role

Actors	Administrator, GUI, Database
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The role has a new attribute assigned.
Basic Course of Events	**1.** User opens the role panel. **2.** User clicks on edit role. **3.** User selects an attribute. **4.** User clicks on submit. **5.** GUI inserts the new association between attribute and role into the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

7. Assign a Role to a User

Actors	Administrator, GUI, Database
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The user has a new role assigned.
Basic Course of Events	**1.** User opens the user panel. **2.** User clicks on edit user. **3.** User selects a role. **4.** User clicks on submit. **5.** GUI inserts the new association between user and role into the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

8. Remove a Role form a User

Actors	Administrator, GUI, Database
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The user has a new role assigned.
Basic Course of Events	**1.** User opens the user panel. **2.** User clicks on edit user. **3.** User clicks on delete role for a defined role. **4.** GUI deletes the association between user and role into the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

9. Delete a Role

Actors	Administrator, GUI, Database
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The selected role is deleted.
Basic Course of Events	**1.** User opens the role panel. **2.** User clicks on delete role. **3.** User selects a role. **4.** User clicks on submit. **5.** GUI deletes the role from the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

10. Assign a Privilege to a User

Actors	Administrator, GUI, Database
Scope	GUI
Precondition	The system is running and the user is logged in with the rights of an administrator.
Postcondition	The system is still running, the user is still logged in.
Postcondition on Success	The user has a new privilege assigned.
Basic Course of Events	**1.** User opens the user panel. **2.** User clicks on edit user. **3.** User selects an privilege. **4.** User clicks on submit. **5.** GUI inserts the new association between user and privilege into the database.
Alternative Paths	-
Open Questions	-
Solved issues	-
Implementation Notes	-
Implementation Status	-

Uncategorized use cases

1. Something happens

Actors	Actor 1, Actor 2
Scope	Scope of this use-case.
Precondition	Pre-condition.
Postcondition	Post-condition (anyway).
Postcondition on Success	Special conditions if successfull.
Basic Course of Events	**1.** Actor 1 does something **2.** Actor 2 does something **3.** Actor 1 does something smart.
Alternative Paths	**2a.** Actor 2 does something not. **3a.** Actor 2 does something stupid. **3b.** Actor 2 does nothing.
Open Questions	**Q:** What happens if z? **Q:** What happens if z-prime?
Solved issues	**Q:** What happens if x? **A:** Then y.
Implementation Notes	• An implementation note. • Yet another note... • Give us more notes! • Math-mode: $\sqrt[7]{45}$
Implementation Status	• May **contain** *slanted* and *italic* `formattings`. Also SMALL CAPS if you want to.