



Codex

Requirement Analysis Report

An iteration and improvement of the current version of Mooshak
A system for automatic judging of submitted programming solutions

28.04.2016

Reykjavik University - T-220-VLN2 - Practical Project II

Hópur 31 - Authors

Aron Práinn Sigurgrímsson
Birkir Brynjarsson
Hilmar Tryggvason
Jón Gunnar Hertervig
Jón Halldór Sigurbjörnsson
Sveinn Guðmundsson

Teachers

Daníel Brandur Sigurgeirsson
Guðmundur Stefánsson

Contents

Overview.....	2
General description.....	2
An evaluation of the current version of Mooshak.....	3
<i>Initial impression.....</i>	<i>3</i>
<i>Feedback for solutions.....</i>	<i>3</i>
<i>Mooshak summary.....</i>	<i>4</i>
User groups.....	5
Interviews.....	6
<i>Interview summary.....</i>	<i>6</i>
Requirement list.....	7
<i>Functional requirements.....</i>	<i>7</i>
<i>Non-functional requirements.....</i>	<i>9</i>
Usability goals.....	11
<i>Student submits a solution for a given problem.....</i>	<i>11</i>
<i>Teacher downloads a submitted solution to a problem.....</i>	<i>11</i>
<i>Administrator creates a new user.....</i>	<i>11</i>
<i>Student downloads his/her submission.....</i>	<i>12</i>
Use case diagram.....	13
<i>Student use case diagram.....</i>	<i>13</i>
<i>Teacher use case diagram.....</i>	<i>14</i>
<i>Administrator use case diagram.....</i>	<i>15</i>
Use cases.....	16
Appendix.....	28
<i>Interview questions and answers.....</i>	<i>28</i>
Administrator.....	28
Teacher.....	29
Student.....	31

Overview

This requirement analysis report contains the results from interviews with stakeholders, and evaluations of *Mooshak* to conclude the use cases, usability goals and requirements of a new improved system for the Reykjavík University. This proposed system has been codenamed *Codex* and will be referred to by that name in this report.

Codex is an iteration and improvement of the current version of *Mooshak*, a system used at the Reykjavík University and in programming competitions to judge and verify the correctness of submissions to programming problems. *Mooshak* still serves its purpose quite well at the university but it has its downsides. The flow of the user interface is unintuitive and overly complex, its main focus on contests has limited use within the university and there is definitely room for other enhancements more fitting to the university's needs.

General description

Since *Codex* will be entirely school focused, it has three user groups, administrators, teachers and students. The following is the basic idea and main goal for each of these three user groups:

- The administrators create users, courses, add users to courses and maintain the system.
- The teachers create assignments, problems and define test cases for the problems.
They also review and grade student submissions to problems.
- Students upload solutions to problems, get instant feedback on the correctness of their solutions and can possibly add collaborators to assignments if allowed.

Mooshak currently supports evaluation of multiple programming languages and other problems such as quizzes, regular expressions and string comparison. To begin with *Codex* will focus on supporting the evaluation of C++ programs which can then later be extended upon.

Another prospective enhancement for *Codex* is to support writing programming solutions in an editor within the system, this will make collaborations easier and teachers can get involved in the code to help out.

An evaluation of the current version of Mooshak

The following evaluation of *Mooshak* was based on our experience as students using the system as well as some further information gathered in release notes and other information online about *Mooshak*.

Initial impression

The first release of Mooshak (version 1.0) was originally released in 2001. Its latest version was released in 2015 but the system shows some obvious signs of being old such as using iframes and/or deprecated frameset and frame elements for templating (as of HTML5).

Its color scheme and layout is not very exciting or intuitive and the interface for logging in is overly complex (too many steps).

These are the steps taken to submit a solution as a student for a given programming assignment:

- Choose course.
- Choose the assignment/contest in that course and logging in with credentials.
- Upload solution.
- Getting feedback and possibly view the details of what went wrong.

These steps could possibly be shortened to just logging in and then uploading your solution. This is the main purpose of the system for a student so it should be easily accessible. This would also simplify the process of uploading and maintaining solutions for multiple assignments in multiple courses.

Feedback for solutions

One of the major positives of *Mooshak* is its instant feedback to students. After a student has submitted his/her solution he/she waits while the system finishes uploading, possibly compiling and then running the program (solution) with a given input. Then it compares the output of the program to the correct predetermined output for the problem. After that he/she instantly gets feedback if the solution was accepted (was correct), had compile time errors or had a wrong output (solution).

After getting his/her results the student can click to view the details of the evaluated solution. This takes him to the feedback page which lists all the checks that were done on the solution and their results.

There are custom tests made on solutions defined by the problem author but common tests are time limit checking, comparison of obtained and expected output and *Valgrind* to check for memory errors.

These checks can both be static and or dynamic where static tests are invoked immediately after compilation (if compiled) before any execution and takes as input the code of the program being evaluated, and dynamic checks are invoked after each execution with a test case. Special checks such as *Valgrind*, are configured separately.

Mooshak has these (and probably more) tools to evaluate the following programming languages and problems:

- C++
- Pascal
- Python
- Java
- C# (Running Mono)
- Text files (string comparison)
- Regular Expressions
- Quizzes

To run Mooshak you need a *Linux* box with *Tcl* and an *Apache* server. You'll also need *gcc* (C compiler) and *make* (to generate executables). *Apache* needs to be configured to be allowed and able to run/execute CGI scripts.

Mooshak summary

Our summary lists what we see as pros and the cons within *Mooshak*.

Pros:

- Extensive tools for evaluation
- Instant feedback on solutions
- The ability to download previous submissions

Cons:

- Unintuitive user interface
- Strange color scheming
- Long log in process (select course, select assignment, enter credentials)
- Hard to navigate from within a problem
- Having to log out to get to other assignments and courses
- Downloading previous submissions is a tedious process and has terrible file naming
- Everything is using a contest terminology

User groups

User group	Background	Use of system	Context	Main tasks
Administrators	Age: Over 20 Education: At least a Bachelor degree in Computer Science Computer Skills: Very good	How often: As often as needed Number of users: 1-10	Real: School Technical: Desktop and laptop	- Managing the user-base
Teachers	Age: Over 20 Education: At least a Bachelor degree. in Computer Science. Assistants often students. Computer Skills: Very good	How often: Weekly Number of users: 5-50	Real: School and home Technical: Desktop and laptop	- Create assignments and problems - Grade student submissions
Students	Age: At least 18 Education: Secondary education Computer Skills: Good	How often: Weekly Number of users: 500-1500	Real: School and home Technical: Desktop and laptop	- Submit solutions and getting feedback on submissions

Interviews

We conducted interviews with 1 teacher and 4 students. Unfortunately we didn't get hold of anyone who had extensively maintained the back end as an administrator of *Mooshak* and created assignments, we were nonetheless able to gather valuable insight and information from our interviewees with students and teachers. The interviews can be found in the appendix.

Interview summary

The students we interviewed were all mobile users, using laptops as their main computer both at school/work and at home. They have been using *Mooshak* at least once a week during school semesters.

The students found that:

- *Mooshak* is too complex for its simple purpose.
- It's overly focused on the contest part while they just want their solutions validated.
- The navigation is bad, it's not possible to go back to the main page when viewing more information on a problem and it's impossible to navigate between assignments and courses without logging out.
- They don't like the color scheme.
- They want to be able to combine aspects of *MySchool* and *Mooshak*, as they find it redundant to have to go between these two applications for a task that could be solved within either one.

Today this has to be done to assign collaborators for assignments.

- The Centris API from the university could be used to solve this.

The teacher found that:

- There should be better flow in all operations.
- The competitive aspect of *Mooshak* is redundant.
- The grading and creation of assignments should be more intuitive.
- It should be easy to collaborate amongst students and assist as a teacher. But this would probably require some sort of code editor within the system.
 - He suggested using ACE editor, an embeddable code editor written in JavaScript.

Both students and teacher like the fact that *Mooshak* works, but it's almost the only thing they like about it.

Requirement list

The following is the list of the requirements for *Codex*, based on our research and interviews with stakeholders and what we've learned from *Mooshak*, which we are extending upon.

There are some words used in the requirement that need further explanation:

Problem: A problem is part of an assignment, it is what the students hand in solutions for. An assignment can have 1 or more problems.

Input-output pair: The teacher defines the input to run with a student solution and the corresponding expected output.

Solution/Submission: The hand in from a student to a problem.

Functional requirements

ID	Description	Priority	Use case	User group
1	Log in to Codex	A	7	Everyone
2	Create user	A	1	Administrators
3	Edit user	A	9	Administrators
4	Delete user	A	19	Administrators
5	Assign user role to user	A	1,9	Administrators
6	View all users	A	22	Administrators
7	Create course	A	4	Administrators
8	Edit course	A	18	Administrators
9	Delete course	A	20	Administrators
10	Add user to course	A	1,9	Administrators
11	Remove user from course	A	9	Administrators
12	View all courses	A	23	Administrators
13	Create assignment	A	2	Teachers

14	Edit assignment	A	8	Teachers
15	Delete assignment	A	8	Teachers
16	Create problem	A	17	Teachers
17	Edit problem	A	8	Teachers
18	Delete problem	A	8	Teachers
19	Edit/Set % of grade to problem	A	17	Teachers
20	Create/Define input-output pairs for problem	A	2	Teachers
21	Edit input-output pairs for problem	A	8	Teachers
22	Submit solution for problem	A	3	Students
23	View all submissions for problem	A	12,21	Teachers & Students
24	View the most correct submission for each student	A	12	Teachers
25	View all assignments	A	3,10	Teachers & Students
26	View all problems within assignment	A	3,11	Teachers & Students
27	Search/Filter assignments	A	10	Teachers & Students
28	Grade problems/assignments	A	14	Teachers
29	Download code from a submission	A	15, 16	Teachers & Students
30	Specify allowed file types and maximum file size for each problem	B	17	Teachers
31	Edit/Set the amount of allowed collaborators to an assignment	B	2	Teachers
32	Add student as collaborator to assignment	B	5	Students
33	Remove yourself as collaborator to assignment	B	6	Students

34	View submissions from all collaborators	B		Teachers & Students
35	Set submission limits for problem	B	17	Teachers
36	Settings panel	B		Teachers & Students
37	Notification center	B		Teachers & Students
38	Evaluate C++ submissions	B		Teachers & Students
39	In-application editor (Ace editor)	C		Teachers & Students
40	Students can invite Teachers to view their code in an in-application editor	C		Teachers & Students
41	Leaderboard of each assignment	C		Teachers & Students

Non-functional requirements

ID	Description	Priority
1	Application is web-based	A
2	Sanitize all user input	A
3	Support latest version of Chrome, Firefox and Safari	A
4	Maximum allowed file size for upload is 50mb	A
5	System should be non-dependent on JavaScript	B
6	Responsive layout design	B
7	Login expires after 24 hours	B
8	Centris login	C
9	Centris synchronized courses	C
10	Centris synchronized assignments	C
11	Backup data on server every 24 hours	C
12	Icelandic support	C

Usability goals

The following are the usability goals that we've set for *Codex*.

Student submits a solution for a given problem

Usability factor	Data collected	Worst case	Preferred case	Best case	Value now
Effectiveness	Finished task	70%	100%	100%	N/A
Efficiency	Time to submit a solution	40 seconds	20 seconds	10 seconds	N/A
Satisfaction	The look and feel of website	Not good	Very good	Extremely good	N/A

Teacher downloads a submitted solution to a problem

Usability factor	Data collected	Worst case	Preferred case	Best case	Value now
Effectiveness	Finished task	80%	100%	100%	N/A
Efficiency	Time to download a submission	50 seconds	20 seconds	10 seconds	N/A
Satisfaction	The look and feel of website	Not good	Very good	Extremely good	N/A

Administrator creates a new user

Usability factor	Data collected	Worst case	Preferred case	Best case	Value now
Effectiveness	Finished task	70%	100%	100%	N/A
Efficiency	Time to create a new user	1.5 minutes	40 seconds	20 seconds	N/A
Satisfaction	The look and feel of website	Not good	Very good	Extremely good	N/A

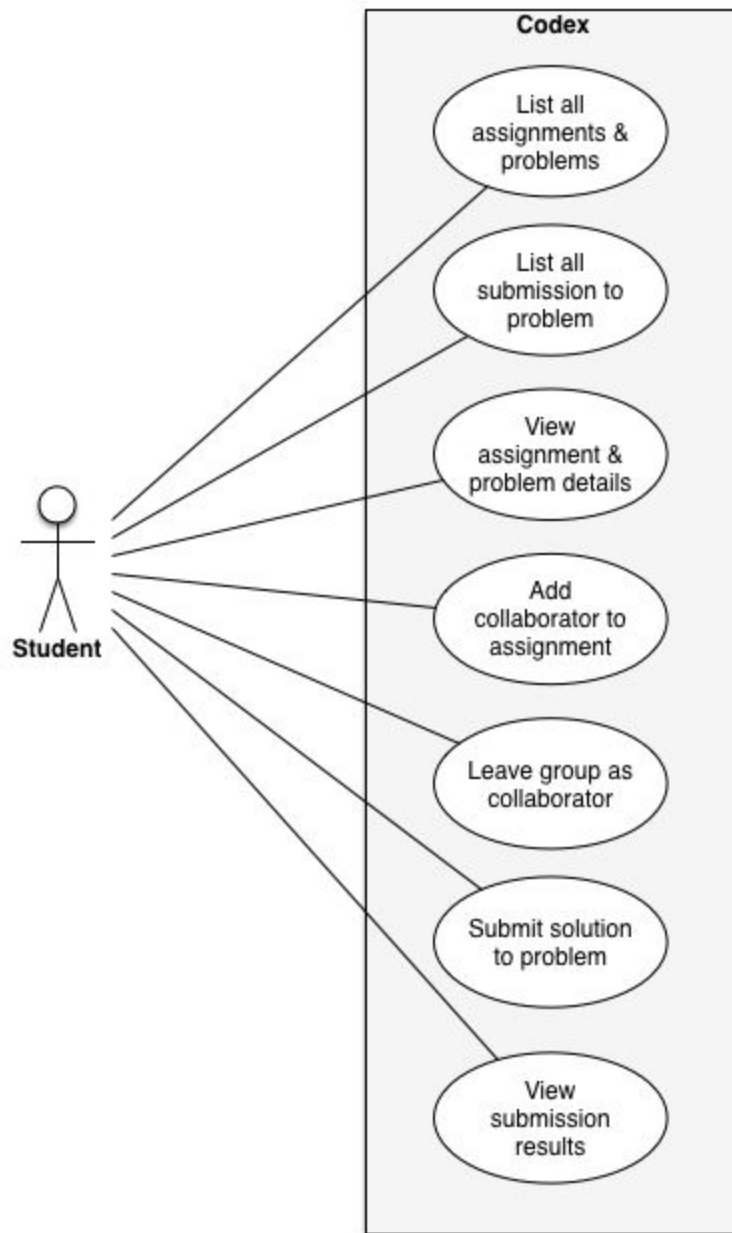
Student downloads his/her submission

Usability factor	Data collected	Worst case	Preferred case	Best case	Value now
Effectiveness	Finished task	70%	100%	100%	N/A
Efficiency	Time to download the submission	40 seconds	20 seconds	10 seconds	N/A
Satisfaction	The look and feel of website	Not good	Very good	Extremely good	N/A

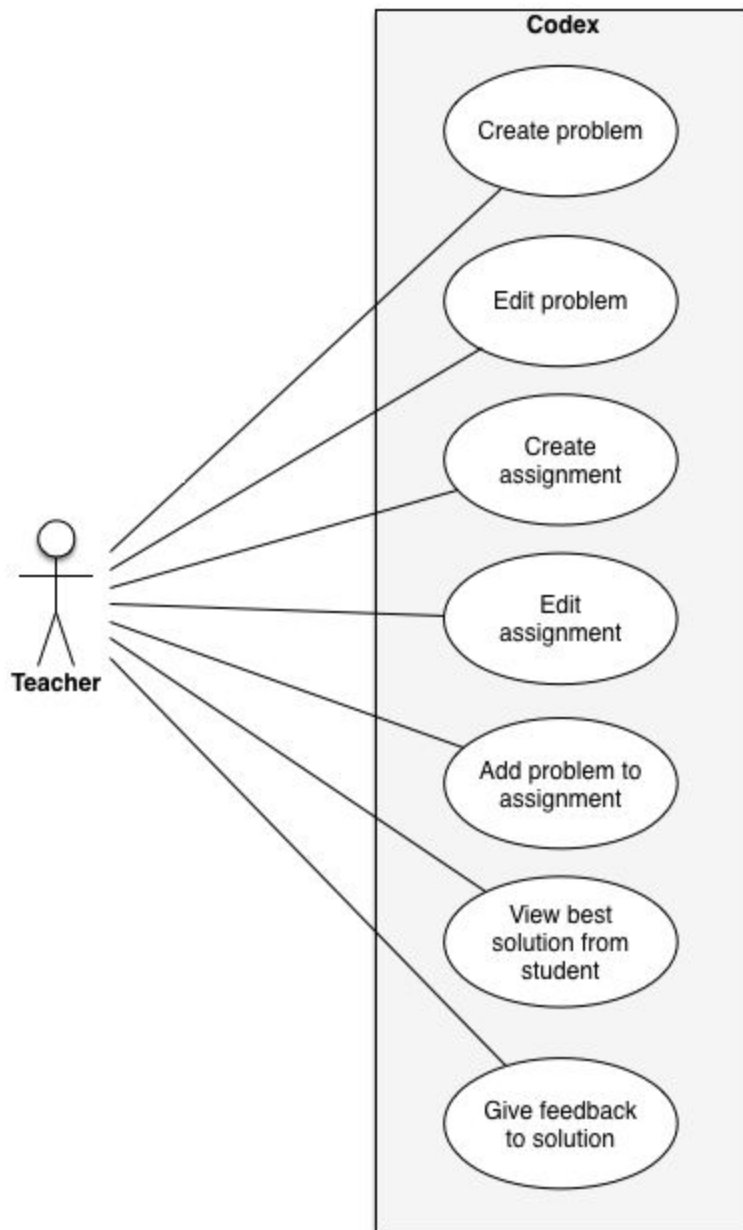
Use case diagrams

The following use case diagrams are for use cases of high priority only.

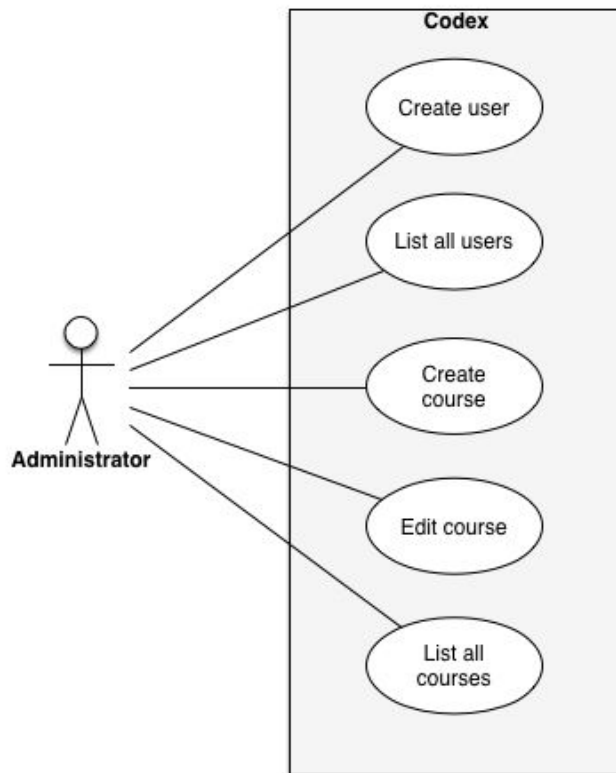
Student use case diagram



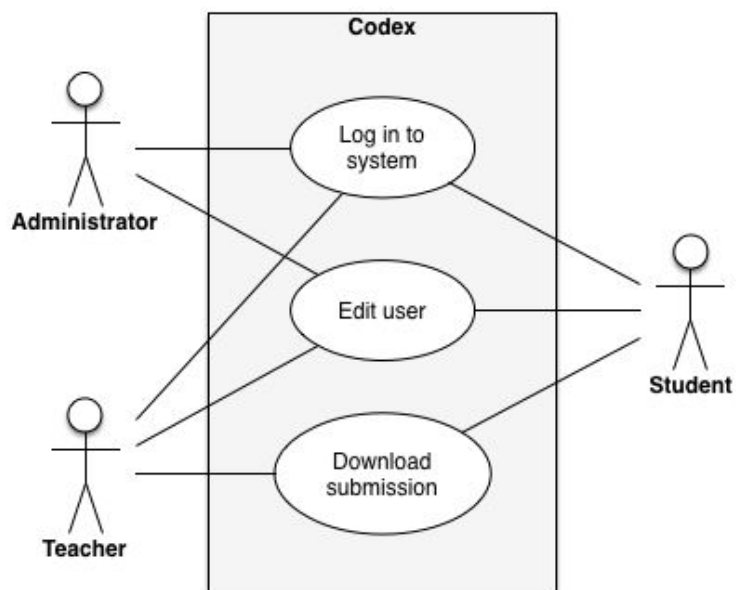
Teacher use case diagram



Administrator use case diagram



Mixed use case diagram



Use cases

Name:	Administrator creates a user
Number:	1
Priority	High
Precondition:	User must not already exist
Description:	An administrator clicks a button that corresponds to creating a new user. The administrator is presented with a form that needs to be filled in order to create a user. In the form the administrator, among other things, defines the user's role and links the user to a course. Finally, the administrator clicks the create button.
Alternative:	-If any required information is missing/invalid, the administrator is immediately notified and must fill in the missing/invalid information before the user can be created
Postcondition	The new user now exists in the system
Source	2, 5, 10
Actor	Administrator

Name:	Teacher creates an assignment within a course
Number:	2
Priority	High
Precondition:	The course exists in the system
Description:	The teacher clicks the button that corresponds to creating an assignment. The Teacher inputs the necessary information regarding an assignment and can then create as many problems within the assignment as the teacher wants (See use case 17).
Alternative:	-If any required information is missing/invalid, the teacher is immediately notified and must fill in the missing/invalid information before the assignment can be created
Postcondition	The assignment now exists in the course
Source	13, 31
Actor	Teacher

Name	Student submits a solution for a problem
Number	3
Priority	High
Precondition	The problem must exist in the system
Description	<p>The student starts by viewing all assignments.</p> <p>The student selects an assignment, which displays the problems for the assignment, followed by selecting the problem he/she wishes to submit a solution for.</p> <p>The student clicks the button that corresponds to selecting a file from his/her computer.</p> <p>The student then clicks the upload button.</p>
Alternative	-If the student uploads nothing, file(s) that are not allowed or if the file(s) are greater than the allowed maximum size, nothing is uploaded and the student is notified.
Postcondition	<p>-if the solution was accepted it registers itself as accepted</p> <p>-if the solution was denied, in case it didn't compile or didn't output correctly the solution registers itself as failed.</p> <p>-In either case the user gets detailed information about the run process.</p>
Source	22, 25, 26
Actor	Student

Name	Administrator creates a course
Number	4
Priority	High
Precondition	The course doesn't already exist in the system
Description	<p>The administrator clicks the button corresponding to creating a new course.</p> <p>Administrator inputs the required information and clicks the create button.</p>
Alternative	-If any required information is missing/invalid, the administrator is immediately notified and must fill in the missing/invalid information before the course can be created
Postcondition	The course now exists in the system.
Source	7
Actor	Administrator

Name	Student adds another student as a collaborator in an assignment
Number	5
Priority	Medium
Precondition	The student which is to be added is not a collaborator of the assignment
Description	The student views his/her assignments. The student clicks the button corresponding adding a student as a collaborator. The student navigates through a list of available students. The student selects the desired student. Finally, the student clicks the add button.
Alternative	-If the add button is clicked without selecting a student, the student is notified.
Postcondition	The selected student is added as a collaborator to the assignment. The student that was added is sent a notification.
Source	32
Actor	Student

Name	Student removes themselves as collaborator in an assignment
Number	6
Priority	Medium
Precondition	The student is a collaborator in the assignment
Description	The student starts by viewing all assignments and then selects the assignment. The student presses the button that corresponds to removing him/herself as a collaborator in the assignment
Alternative	None
Postcondition	The student is no longer a collaborator to the assignment, any submissions done in the group will remain there, the student who left will need to submit their solution again
Source	33
Actor	Student

Name	User logs in to Codex
Number	7
Priority	High
Precondition	None
Description	The user inputs password and username to the appropriate input slots and clicks log-in.
Alternative	-If the system cannot authenticate the username/password pair, the user will be notified
Postcondition	The user is now logged in
Source	1
Actor	Student, Teacher, Administrator

Name	Teacher edits/delete an assignment from a course
Number	8
Priority	High
Precondition	There is an assignment in the course
Description	The teacher chooses an assignment from a list of relevant assignments, they will then be presented with a form where they can edit or delete the assignment and/or its problems
Alternative	-If there is no relevant assignment the teacher is presented with a message
Postcondition	The assignment has been altered/removed as the teacher intended
Source	14, 15, 17, 18, 21
Actor	Teacher

Name:	Administrator edits a user
Number:	9
Priority	High
Precondition:	User must already be part of the system
Description:	The administrator starts by viewing a list of users and selects the one he/she wished to edit. A form is now displayed with the current information already filled in. The administrator can make any changes to the form and click the submit button when finished.
Alternative:	-If any required information is missing/invalid, the administrator is immediately notified and must fill in the missing/invalid information before the course can be created
Postcondition	The selected user has been edited according to changes made
Source	3, 5, 10, 11
Actor	Administrator

Name	User (Teacher/Student) views assignments relevant to them
Number	10
Priority	High
Precondition	The user is teaching/studying in a course
Description	The user clicks the button that corresponds to displaying his/her assignments. The user is presented with a list of assignments relevant to them. The assignments should be sorted by date (newest first) and can be filtered.
Alternative	-If there is no relevant assignment the user is presented with a message -if the user is both a teacher and a student they are presented with two tabs, one for assignments they are teaching, the other for assignments they are studying
Postcondition	The user can view assignments relevant to them
Source	25, 27
Actor	Student, Teacher

Name	User (Teacher/Student) views an assignment and it's problems
Number	11
Priority	High
Precondition	The user is teaching/studying in a course with at least one assignment
Description	The user chooses an assignment from a list of relevant assignments (See use case 10), they will then be presented with information displaying the properties and problems of the selected assignment
Alternative	-If there is no relevant assignment the user is presented with a message -if the user is both a teacher and a student they are presented with two tabs, one for assignments they are teaching, the other for assignments they are studying
Postcondition	The user can view the selected assignment
Source	26
Actor	Student, Teacher

Name	A teacher views the student submissions to a problem
Number	12
Priority	High
Precondition	The assignment and problem both exist in the system
Description	The teacher clicks the button that corresponds to viewing his/her assignments (see use case 10). The teacher click the assignment and then the desired problem, which displays a list of students that have submitted a solution for the problem. Clicking a student displays a list of all submissions made by that student for the problem.
Alternative	-if a student has not submitted a solution, it will show a message instead of a list of student submissions
Postcondition	The teacher can view submissions from their students
Source	23, 24
Actor	Teacher

Name	A teacher views a student submission
Number	13
Priority	High
Precondition	The submission exists
Description	The teacher starts by viewing the submissions (See use case 12). Then the teacher clicks the desired submission.
Alternative	None
Postcondition	The teacher is presented detailed information on the submission
Source	23, 24
Actor	Teacher

Name	A teacher grades a submission
Number	14
Priority	High
Precondition	The teacher is viewing the problem
Description	The teacher clicks the button that corresponds to viewing his/her assignments (see use case 10). Then the teacher filters the submissions by best submission. (See use case 13) The teacher inputs the grade for the selected student, and presses the submit button
Alternative	-If the grade is invalid, the teacher will be notified
Postcondition	A grade will be saved to the student's submission
Source	28
Actor	Teacher

Name	A teacher downloads a submission
Number	15
Priority	High
Precondition	The submission exists
Description	The teacher starts by viewing the submission (See use case 12) The teacher clicks the download button on the selected submission
Alternative	None
Postcondition	The teacher downloads a copy of the submission onto their computer
Source	29
Actor	Teacher

Name	A student downloads their own submission
Number	16
Priority	High
Precondition	The submission exists
Description	The student starts by viewing his/her assignments (See use case 11). The student clicks on a problem expander button, which displays all of his/her submissions for the problem. The student selects the "Download" button on the selected submission
Alternative	None
Postcondition	The student downloads a copy of the submission onto their computer
Source	29
Actor	Student

Name:	Teacher creates problems within an assignment
Number:	17
Priority	High
Precondition:	None
Description:	The teacher specifies input and output for testing, defines max number of students in a group for a handin and sets the limit of submissions.
Alternative:	-if the teacher creates more than one problem they must give it a % value if the teacher does not it, will be split even. -if no output and/or input are set, then the teacher is notified that these must be defined.
Postcondition	The problems are now a part of the assignment
Source	16
Actor	Teacher

Name:	Administrator edits a course
Number:	18
Priority	High
Precondition:	The course must exist in system
Description:	The administrator starts by viewing all courses (See use case 23). The administrator clicks on the course and a form is displayed with pre-filled fields with current information.
Alternative:	-If any required information is missing/invalid, the administrator is immediately notified and must fill in the missing/invalid information before the course can be edited
Postcondition	The course has now been changed
Source	8
Actor	Administrator

Name:	Administrator deletes a user
Number:	19
Priority	High
Precondition:	The user must exist in the system
Description:	The administrator starts by viewing all users (See use case 22) The administrator locates the user to be deleted, by either traversing the list of users or searching for a user. Once the administrator has selected the user, a delete button will become active and the administrator presses said button.
Alternative:	None
Postcondition	User is no longer in the system
Source	4
Actor	Administrator

Name:	Administrator deletes courses
Number:	20
Priority	High
Precondition:	The course must exist in the system
Description:	The administrator starts by viewing all courses (See use case 23). The administrator locates the course to be deleted, by either traversing the list of courses or searching for a course. Once the administrator has selected the course, a delete button will become active and the administrator presses said button.
Alternative:	None
Postcondition	The course is no longer in the system
Source	9
Actor	Administrator

Name:	User views the history of all submissions for a problem
Number:	21
Priority	High
Precondition:	Assignment problem must exist
Description:	From the assignment viewer the user navigates to the assignment he wishes to see all of the submissions for a problem. Once the user has clicked on the problem, a list of submissions appears.
Alternative:	None
Postcondition	User is able to view a list of submissions
Source	23
Actor	Teacher and Student

Name:	Administrator views all users
Number:	22
Priority	High
Precondition:	None
Description:	The administrator clicks the button corresponding to viewing all users.
Alternative:	None
Postcondition	A list of all users is displayed
Source	6
Actor	Administrator

Name:	Administrator views all courses
Number:	23
Priority	High
Precondition:	None
Description:	The administrator clicks the button corresponding to viewing all courses.
Alternative:	None
Postcondition	A list of all courses is displayed
Source	12
Actor	Administrator

Appendix

Interview questions and answers

As mentioned earlier in the report we did not find an administrator that has extensively used the backend of Mooshak, but we interviewed a teacher that gave us some insight.

These are the questions that we had in mind for an administrator.

Administrator

- **Background**
 1. How old are you?
 2. What is your education?
 3. How would you rate your computer skills?
- **Usage**
 1. How often would you use Codex?
- **Context**
 1. Where would you use Codex?
 2. On what devices would you use Codex?
- **User goals**
 1. What would you like to be able to achieve with Codex?
- ***If interviewee has been an administrator for Mooshak before***
 1. Talk us through the process of creating/adding a user.
 2. What are the most time consuming tasks in Mooshak?
 3. What are the things you like most about Mooshak?
 4. What are the things you would like to see improved in Mooshak?

Teacher

- **Background**

1. How old are you?

a. 43

2. What is your education?

a. Bachelor degree in computer science

3. How would you rate your computer skills?

a. Very good

- **Usage**

1. How often would you use Codex?

a. 1-2 times per week

- **Context**

1. Where would you use Codex?

a. At school and maybe at home


2. On what devices would you use Codex?

a. Almost entirely on laptop.

- **User goals**

1. What would you like to be able to achieve with Codex?

a. Better flow in all operations, designed with the needs of the teachers and students in mind and with less or without focus on the competitive aspect of the older version. Make it simple and intuitive for teachers to create and grade assignments. Make it easy to collaborate amongst students and simple to assist as a teacher. This would probably need some sort of editor within the system to view, edit and submit code (solutions).

- 
- *If interviewee has been a teacher using Mooshak before*
 1. **Talk us through the process of creating an assignment**
 - a. I have too little experience of that procedure.
 2. **What are the most time consuming tasks in Mooshak?**
 - a. Reviewing code feedback.
 3. **What are the things you like most about Mooshak?**
 - a. The fact that it works and students get instant feedback when submitting their solutions.
 4. **What are the things you would like to see improved in Mooshak?**
 - a. This was answered in the user goals section above.

Student

- **Background**

1. How old are you?

- a. 27
- b. 21
- c. 20
- d. 20

2. What is your education?

- a. Stúdentspróf
- b. Stúdentspróf
- c. Studentspróf
- d. Stúdentspróf

3. How would you rate your computer skills?

- a. Extremely good
- b. Very good
- c. Average
- d. Good

- **Usage**

1. How often would you use Codex?

- a. Every week, autumn and spring.
- b. As often as new assignment is available.
- c. As much as the assignments
- d. Approximately once per week.

- **Context**

1. Where would you use Codex?

- a. At home and in school.
- b. In school.
- c. In school and at home
- d. In school and at home.

2. On what devices would you use Codex?

- a. Laptop
- b. Laptop
- c. Laptop
- d. Laptop

- **User goals**

1. What would you like to be able to achieve with Codex?

- a. Hand in assignments, get instant feedback.
- b. Good login, "easy to use" layout. Hand in assignments, check if my code works, get error messages as feedback.
- c. Get Accepted even if the output does not contain equal number of invisible spaces
- d. I would like to work in user friendlier interface and improved layout.

- **If interviewee has been a student using Mooshak before**

1. Talk us through the process of submitting a solution to an assignment.

- a. I go to the homepage of the website, choose course, choose project and log-in (finally!). Press "choose file" and submit and get results.
- b. Choose course, press log-in, choose assignment, enter my log-in info, choose project within a single assignment, press submit file and wait for a feedback.

- c. Choose a course then assignment then log in then i submit file and then get a feedback.
- d. Its very time consuming and boring to log-in to the system. First, I choose the course, then, I choose the project within chosen course, and then I can finally log in. When I'm logged-in, I can choose which part of the assignment i'm going to hand-in, I upload my code and submit it. If I receive an error message(s), I try to read through the message and fix my project based on my error findings.

2. What are the most time consuming tasks in Mooshak?

- a. Log-in and get former hand-ins.
- b. Look for the problem (if it occurs).
- c. Understanding what and where the error is
- d. To read the error message(s), especially if there are some memory errors/leaks.

3. What are the things you like most about Mooshak?

- a. Instant feedback (but needs better readability, hard for eyes to read error messages).
- b. Fast response time.
- c. Get some feedback for the code
- d. It's good to get an instant feedback.

4. What are the things you would like to see improved in Mooshak?

- a. Better layout, snappier, fewer options to do/complete tasks.
- b. No information about other people hand-ins. Don't want to search for my submission if many people are submitting at the same time. Less information in error messages. Only want to see what is wrong, no need to see what is correct.
- c. Don't be annoying with the accepting and include it with MySchool.
- d. I would like to see improved layout so the user can navigate more easily around the system. I would be nice to have to option to switch between assignments without having to log-out.