Codex

Design Report

An iteration and improvement of the current version of Mooshak

A system for automatic judging of submitted programming solutions

03.05.2016

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

Hópur 31 - Authors

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

Teachers

Daníel Brandur Sigurgeirsson
Guðmundur Stefánsson

Contents

Overview

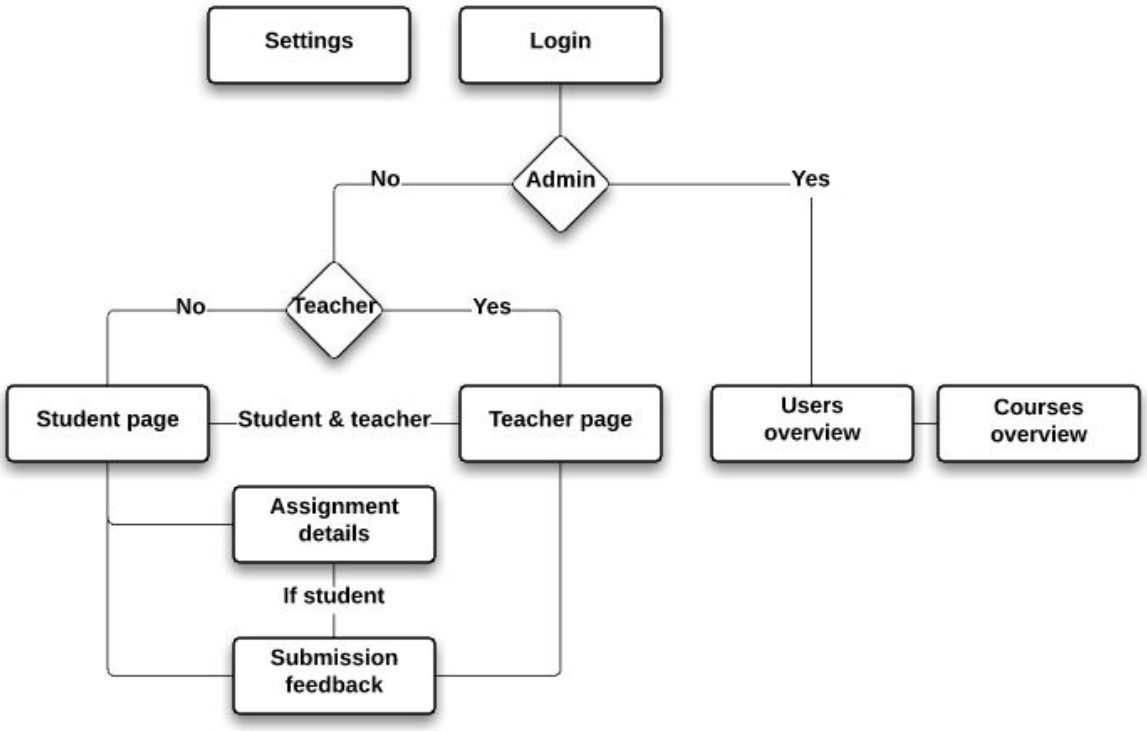
This design report contains information on how the front-end of *Codex* should be like and how the back-end of *Codex* should be coded.

For the front-end we included a navigational diagram that shows how to navigate *Codex* along with multiple mockups on the look and general structure of the *Codex* user interface.
We asked some people to be a part of low-fidelity prototyping so we could measure our usability goals from the requirement analysis report.

In the back-end we decided we needed an entity relation diagram and a table schema for the database, a class diagram for our C# classes and the coding rules for the various languages that *Codex* is comprised of. In the appendix we have also included an SQL definition for the database.

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.

Navigation diagram



Shared pages:

After logging in the user will be sent to their appropriate landing page (a user who is both student and teacher can swap between the student page and teacher page at will).

In the settings menu a user can change their password.

Using the navbar a user can logout, open their settings page, or go to their landing page regardless of the page they are currently on.

The submission feedback shows details on the submission. It can be viewed both by the submitting student (or assignment collaborator) and the course's teacher(s).

Student pages:

The student page will display a list of all assignments for that student. Each assignment in this list can be expanded to show more details, here the student can open the assignment details page for a selected assignment, or the submission feedback page for a selected submission.

The assignment details page will show more details on the assignment, including links to past submissions, which will take the user to the submission feedback page for that submission.

Teacher pages:

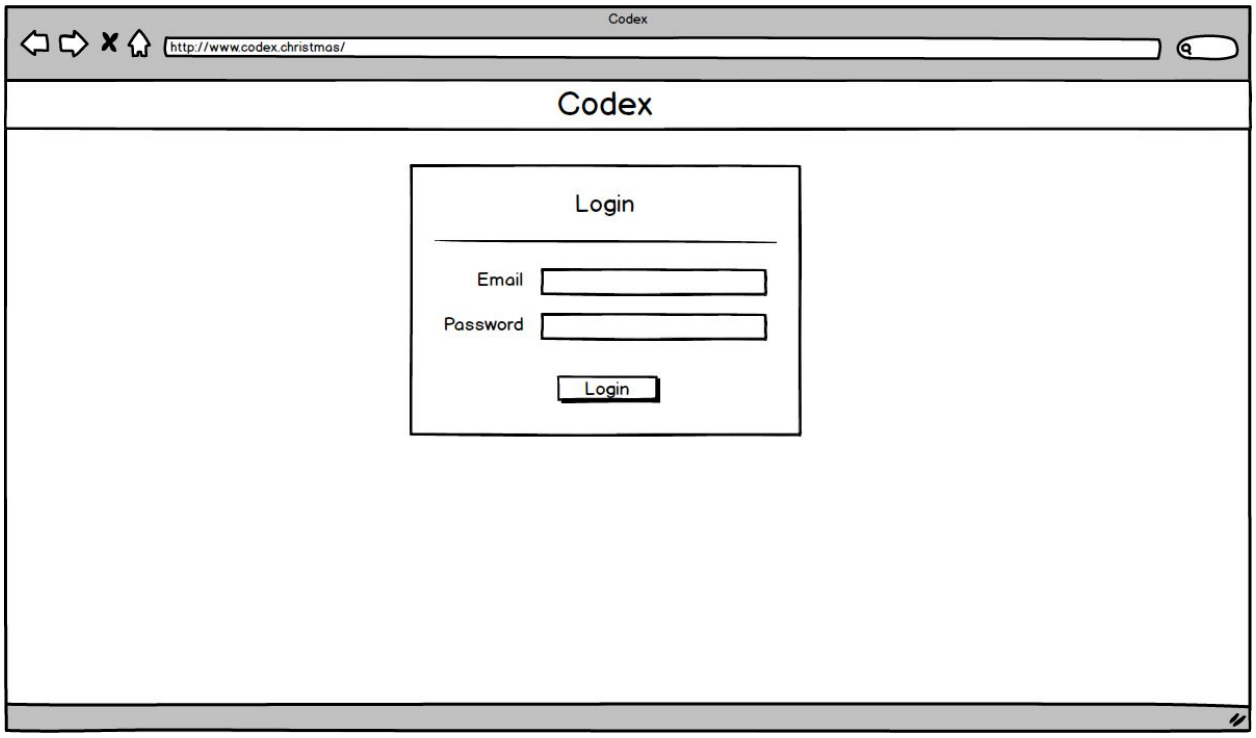
The teacher page will display a list of all assignments in the courses they are teaching in. Each assignment can be expanded for details about that assignment, including links to student submissions, which will take the teacher to the submission feedback page for that submission. On this page the teacher can also add new assignments/problems and view all course problems. When adding new assignments/problems a form will appear on screen.

Admin pages:

The landing page for admins is the user's overview page which will display a list of all users. They can switch between this page and the courses overview at will. These pages allow the admin to add/remove/edit users/courses, this includes adding students and teachers to courses.

Page prototypes

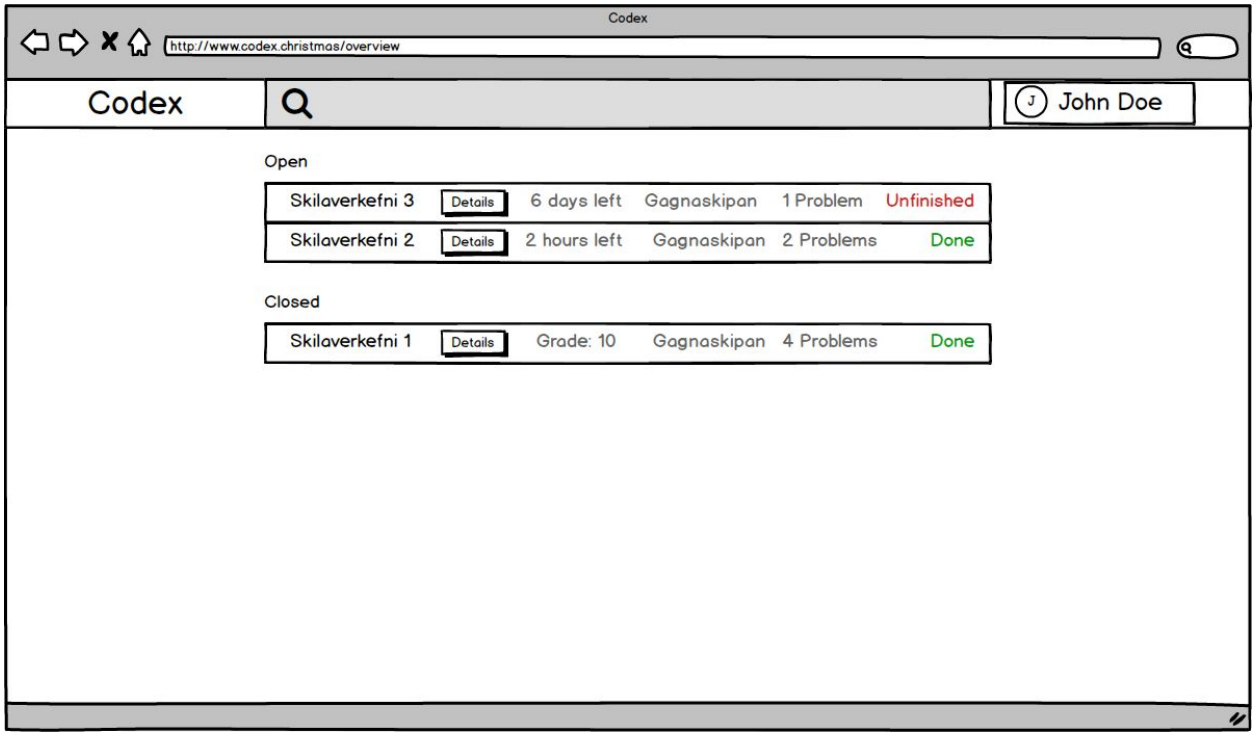
Login



When entering the side, the login page is the first page that appear in the system. The user enter his/her username (email) and password in separate input fields and presses login to initiate the login process. The system will automatically identify which type of user is trying to login and redirects the user to the relevant view based on the user type.

Student

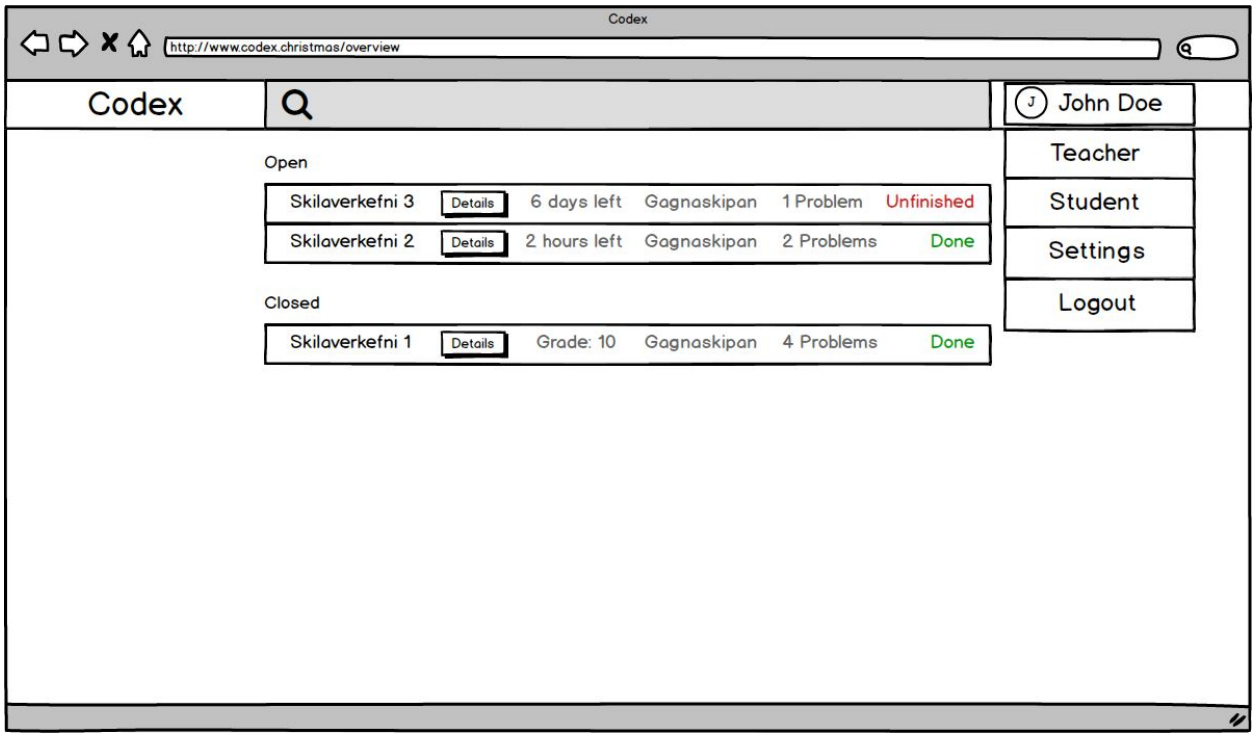
Student overview



This is the landing page for the students. It displays all the assignments for the student. Each assignment row can be clicked in order to expand it to view its problems. The details button can be clicked to view the assignment details.

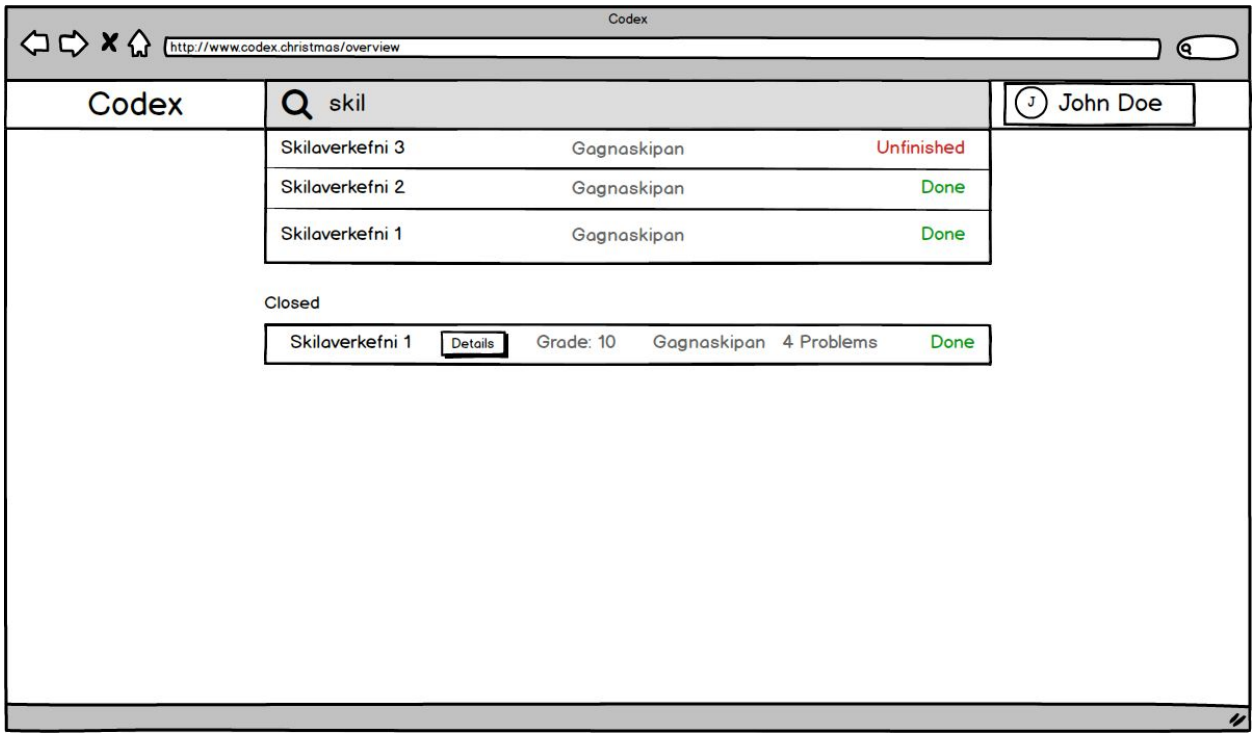
The student can search for assignments in with the search bar and the student's name can be clicked in the upper right hand corner in order to switch to teacher view, if the student is a teacher or a teaching assistant in a course, to access settings and to logout.

Upper right hand dropdown



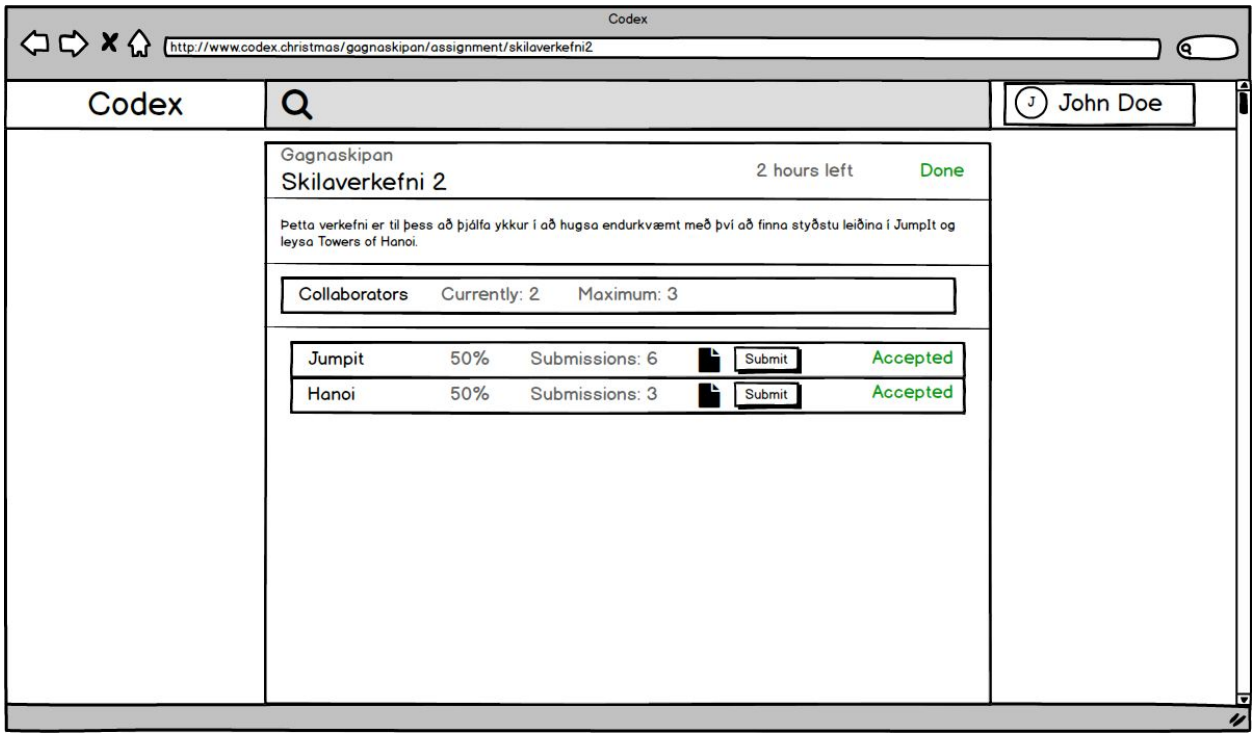
This dropdown can be accessed on every page. If the student is not a teacher or a teaching assistant only the Settings and Logout options will be displayed.

Search bar



The search bar is also on every page. Both students and teacher have access to this functionality of searching for assignments.

Assignment details

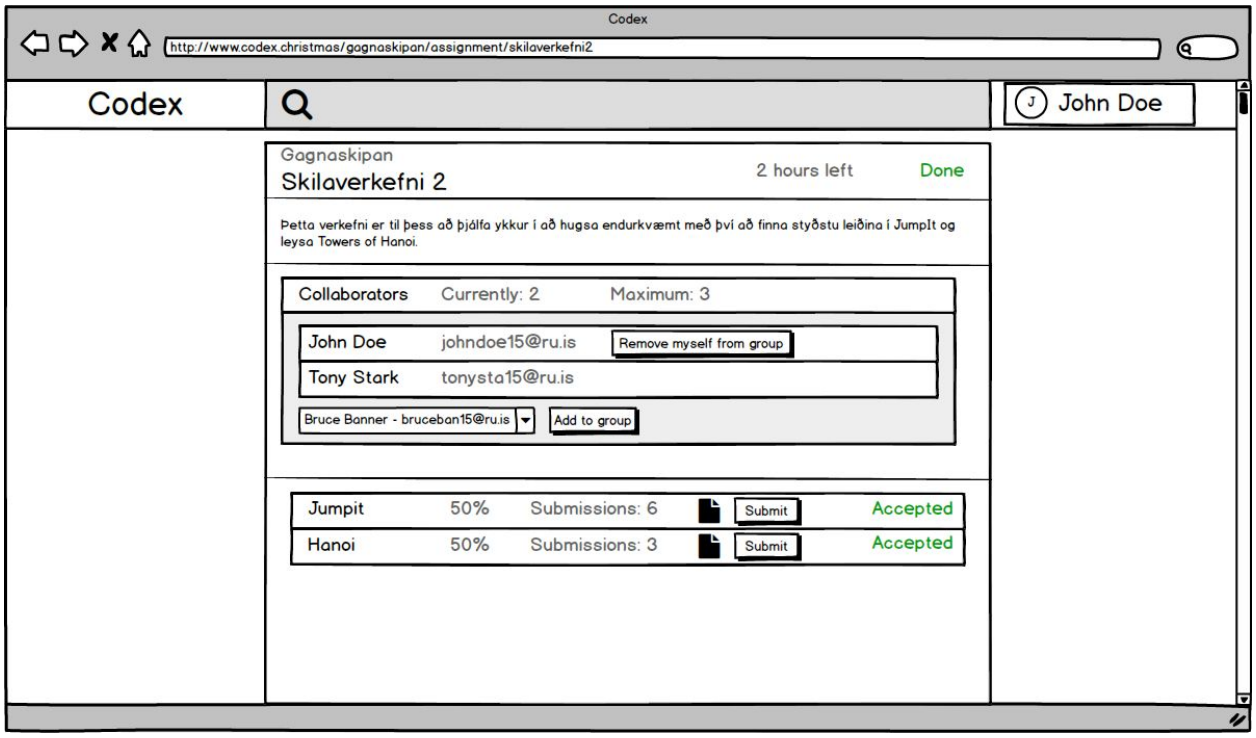


This is the view that is displayed when a student clicks the Details button on an assignment in the overview.

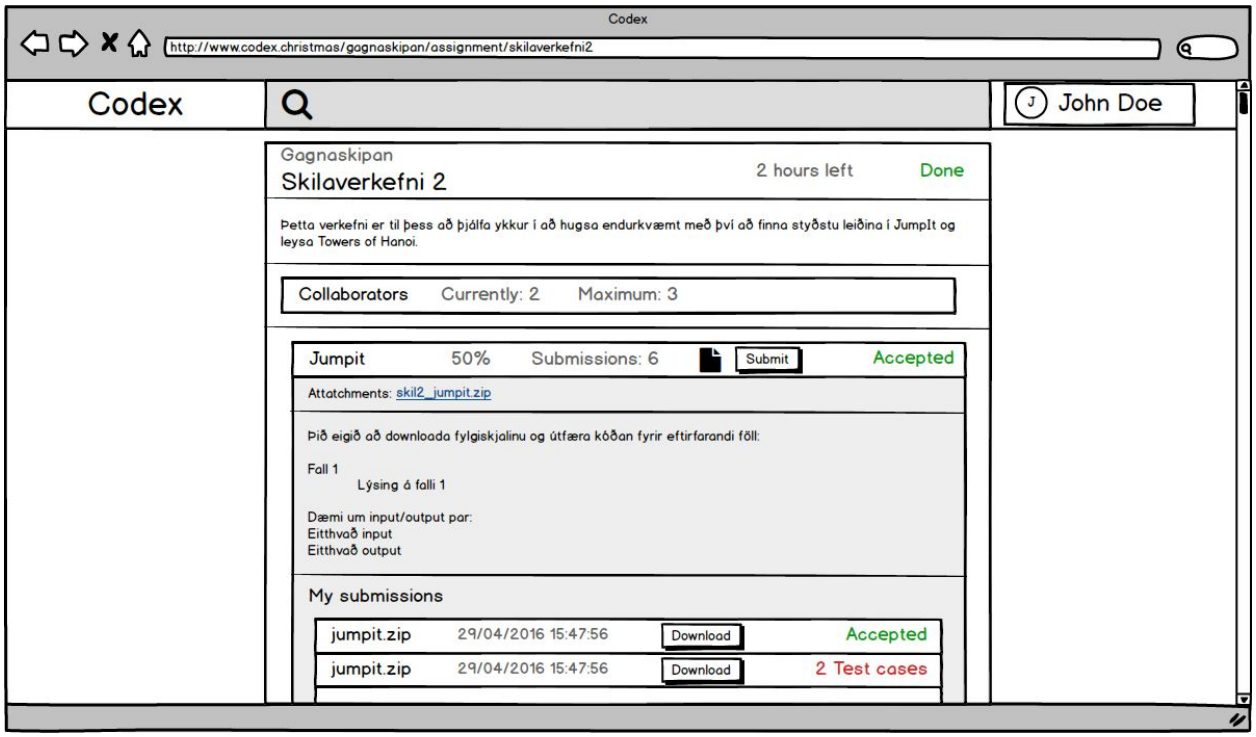
Here they can see the following information on the assignment:
In the header is the name of the assignment, the course it belongs to, how much time is left and the student's status on the assignment as a whole.

Below the header is the assignment description.

Underneath the description the Collaborators row. Clicking on it expands it.
Finally, the assignment problems are displayed. Clicking on a problem expands it.



Here the Collaborators row has been clicked. The student can remove themselves from the group, view and add their collaborators.



Here a problem has been clicked. Displaying what attachment is attached to the problem, the problem description and the student's submissions to the problem.

Assignment problems

When clicking on an assignment, a dropdown is displayed which shows the assignment's problems and there students can submit their solutions to the problems.

The following information is in a problem row: name of the problem, weight of the problem, number of submissions, choose file and submit button and, at the far right, the best result from the student's own submissions.

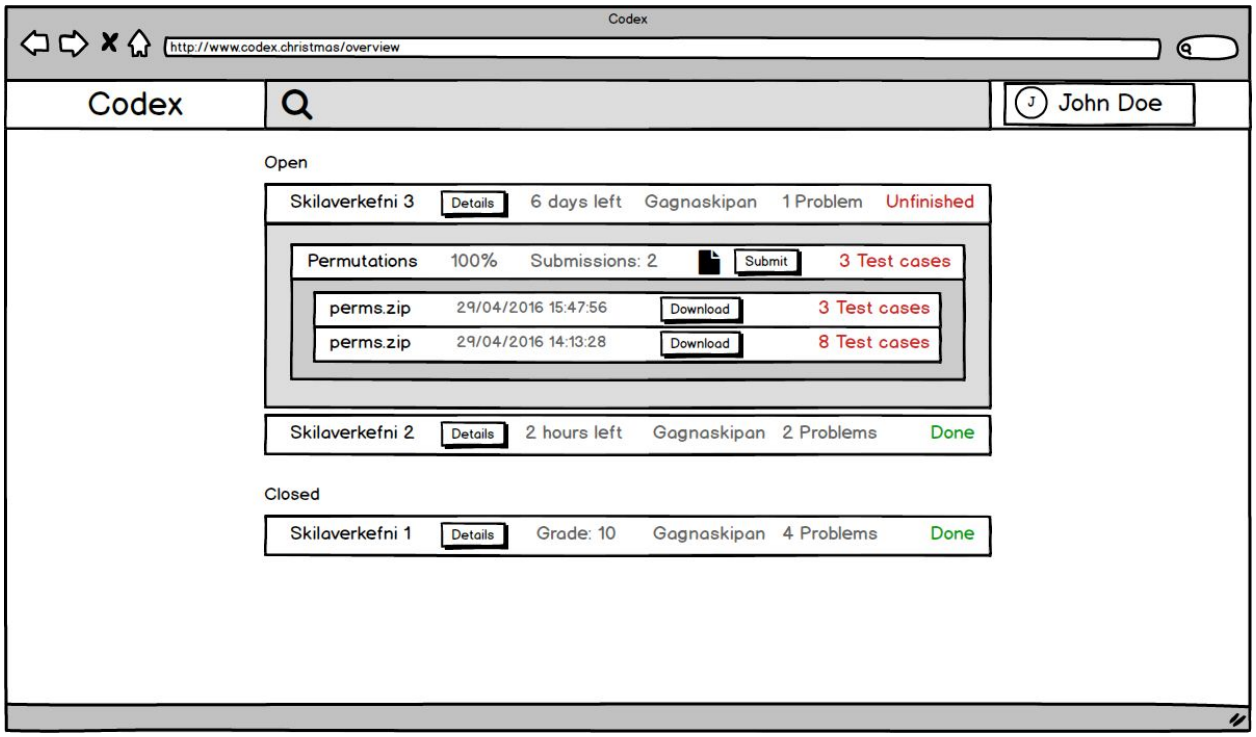
Submitting process

The student starts by choosing a file and pressing submit. A progress bar will be displayed.

When done uploading, the submission will be evaluated, displaying a spinner.

Finally, the result is displayed where the spinner was, regardless whether or not it's the student's best submission.

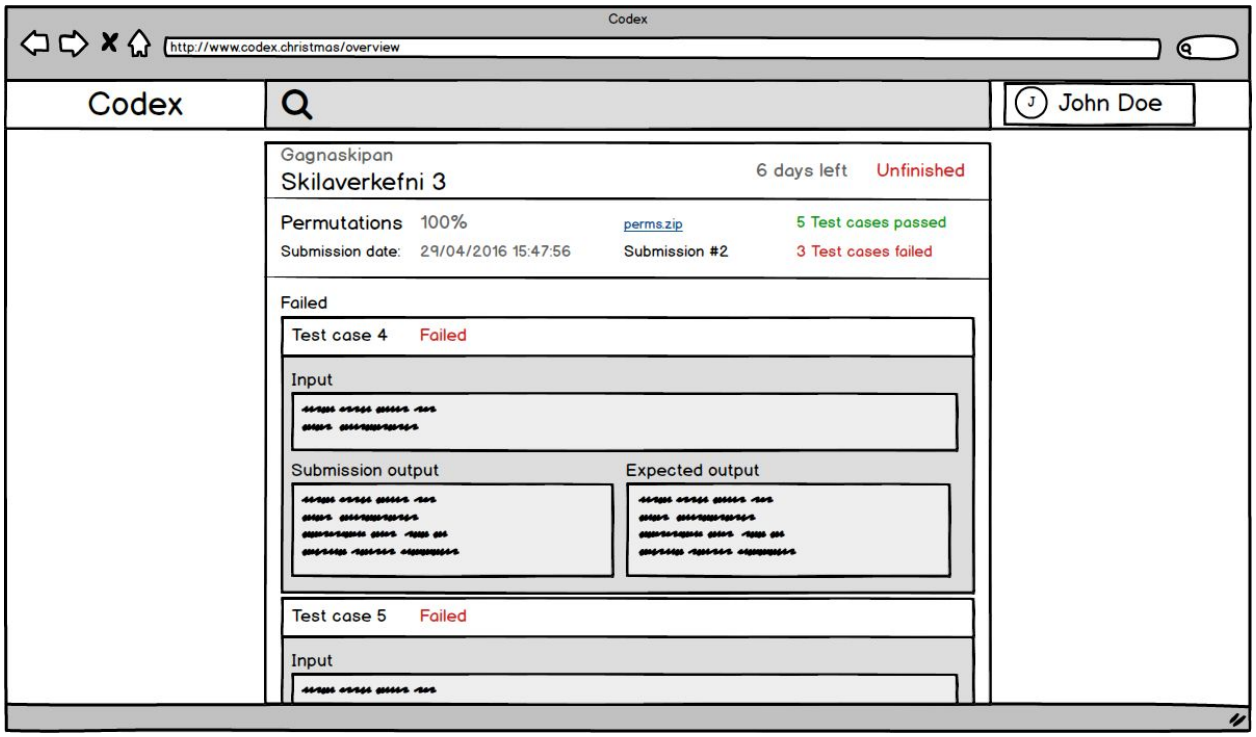
Submissions



Clicking on a problem displays all the submissions the student has submitted.

Here the student can see the name of the file he/she submitted, when it was submitted, a download button for the submission and the results of each submission.

Submission details



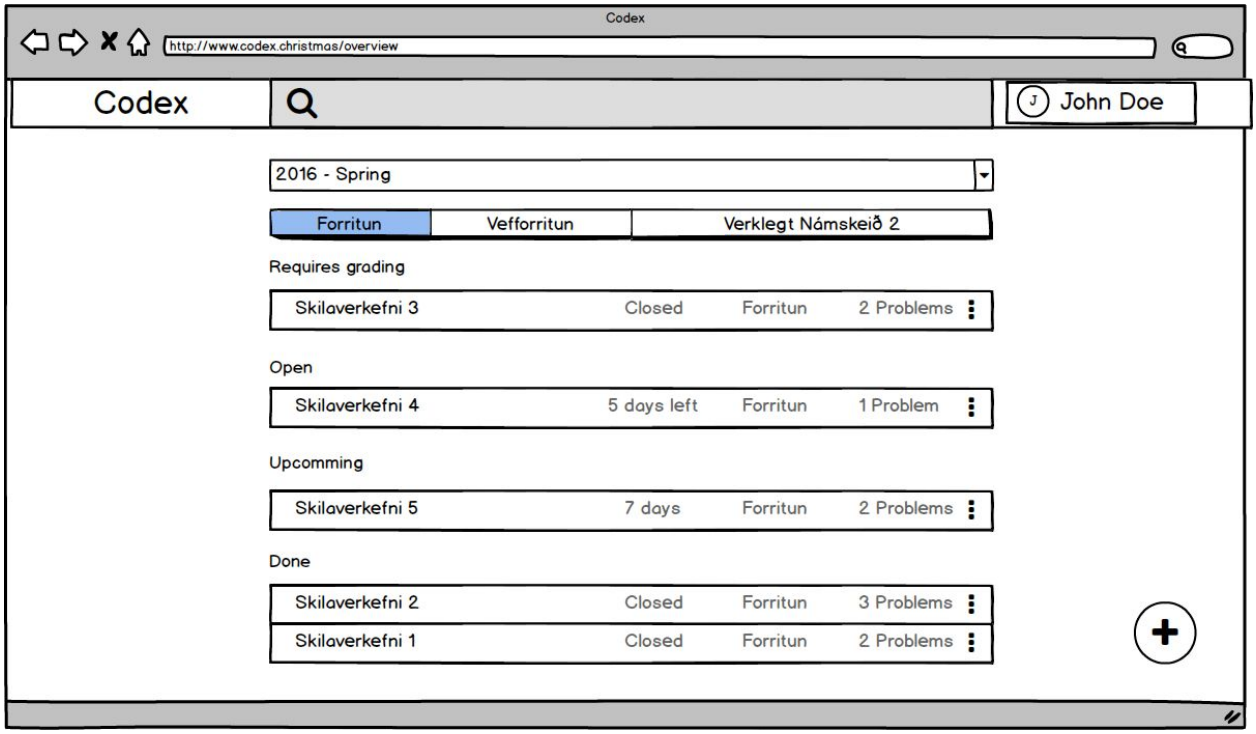
Clicking on a submission row directs to a new view where the student can see details about the submission.

The header displays a few details about the assignment, clicking on the name of the assignment directs to a new view displaying assignment details.

The area below the header displays information about the submission. The name of the problem it was submitted to, the weight of said problem, time of submission, name of submission, what number the submission is for the student and the results of the test cases.

Teacher

Teacher overview



This is the landing page for teachers.

The teacher can select a semester from the dropdown box below the search bar, the default is the current semester.

The teacher can select a course from the tabs below the semester dropdown.

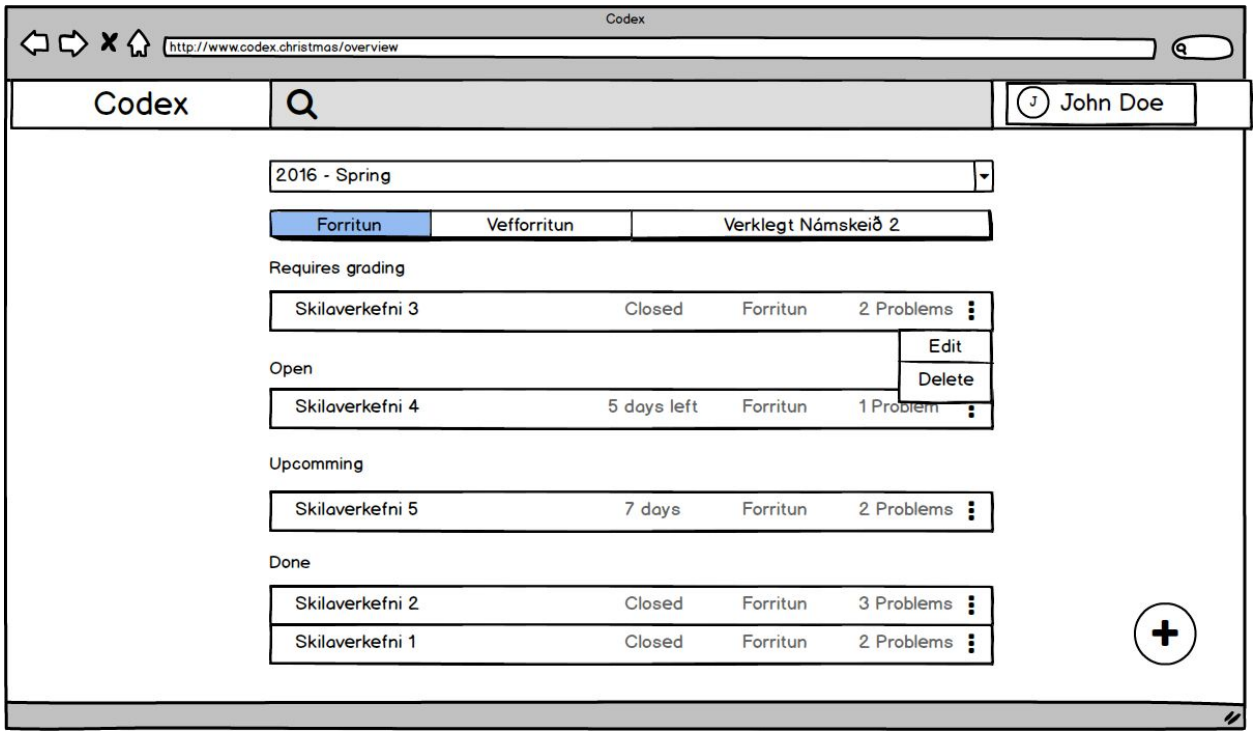
The assignments are categorized into four categories:
Requires grading: These assignments are passed their deadlines and contain students who have yet to receive their grades.
Open: Assignments that the students are solving at the moment.
Upcoming: Assignments that have not yet been open.
Done: These are past their deadlines and have been graded.

The assignments can be clicked to display the problems.
Each assignment has a vertical ellipsis icon. Clicking it displays a menu where the teacher can either edit or delete the assignment.

Clicking the plus button in the bottom right displays the operations to add a new assignment, a new problem or view the problems of the course.

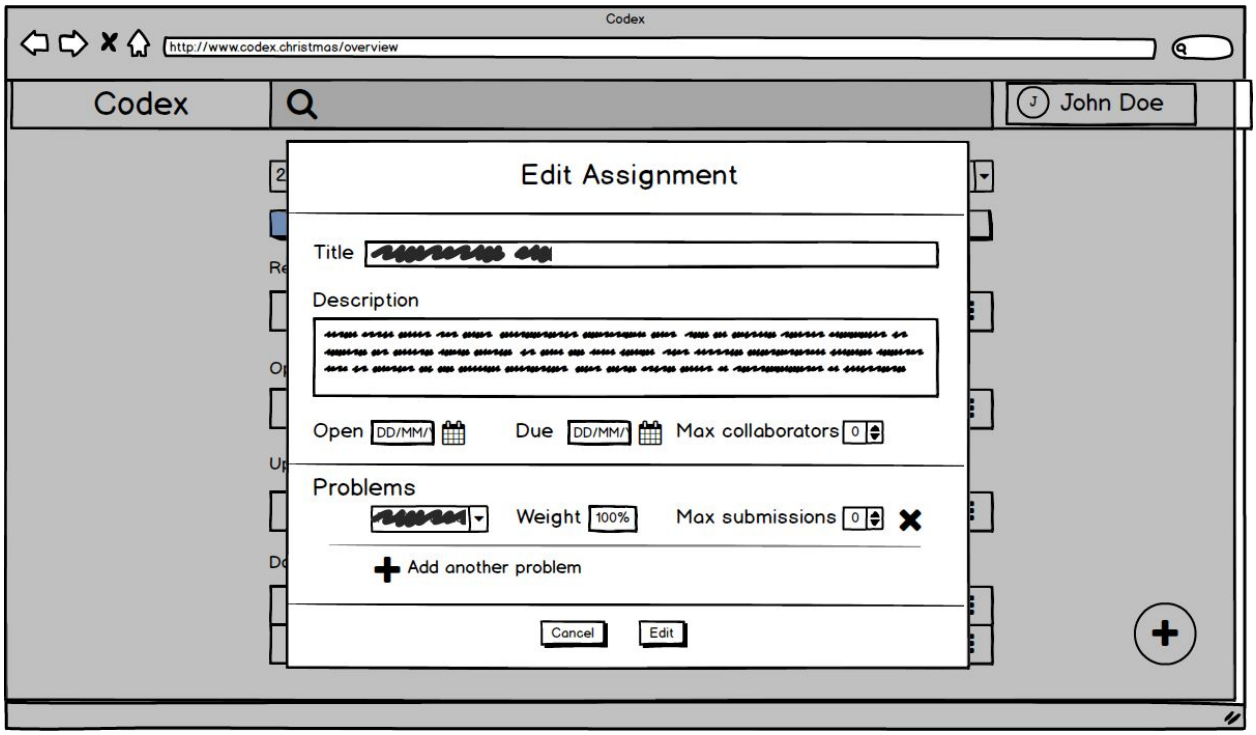
If the user is both a student and a teacher, the landing page is the teacher view. The user can switch between views through the dropdown in the upper right hand corner.

Assignment dropdown



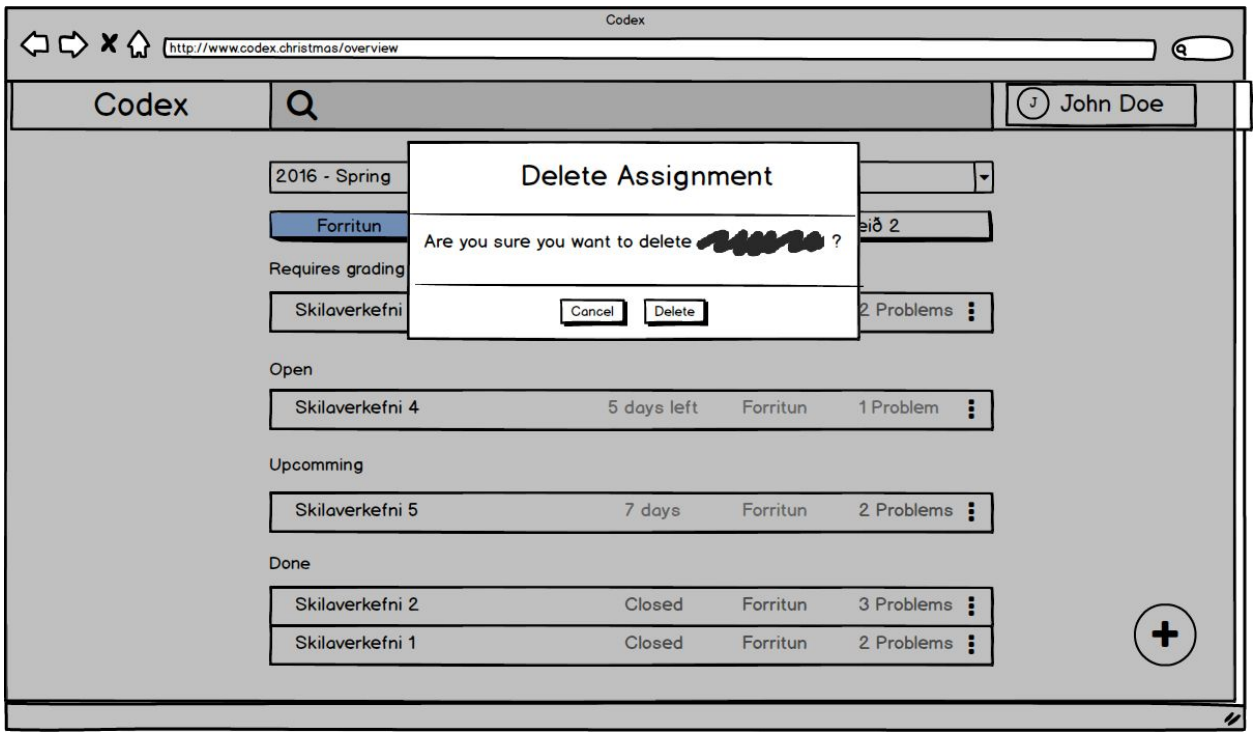
Here the vertical ellipsis has been clicked.

Edit assignment



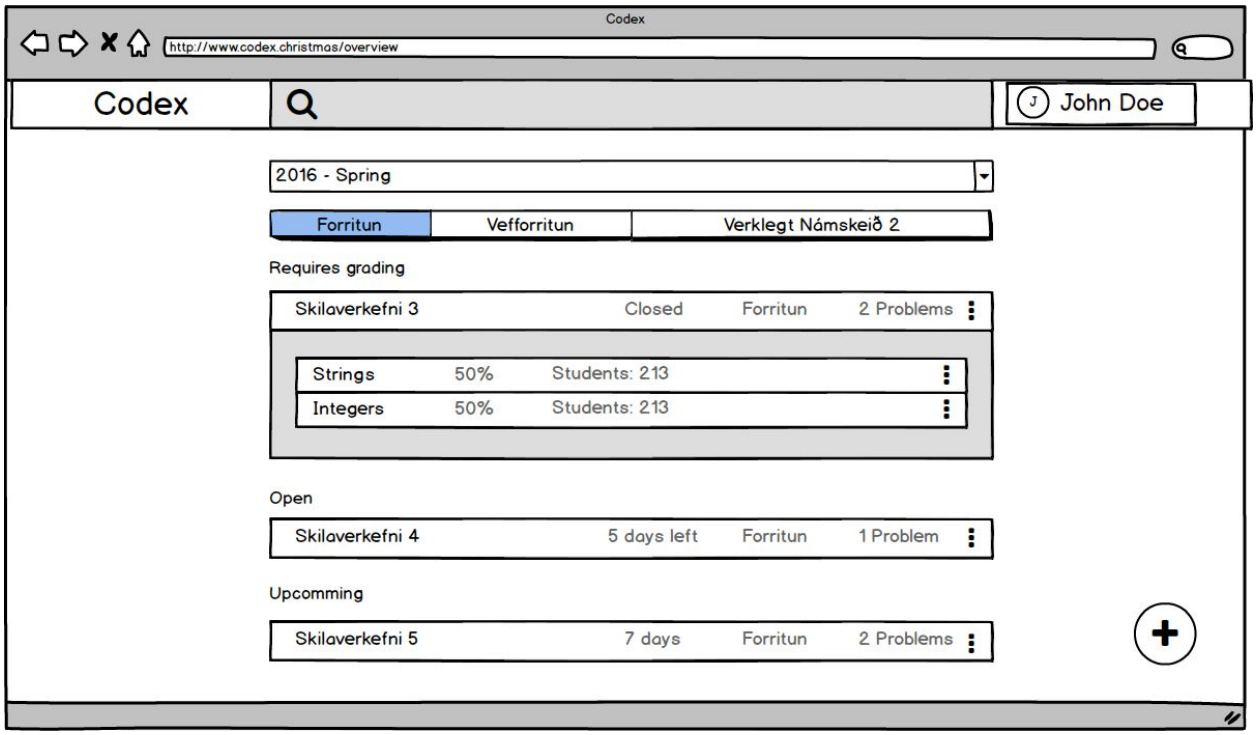
When editing an assignment a modal is displayed. Here the teacher can change the assignment's title, description, when the assignment opens and is due, maximum collaborators and which problems are in the assignment along with their weight and submission limit.

Delete assignment



When the teacher clicks Delete on an assignment the teacher is displayed a confirmation.

Assignment problems

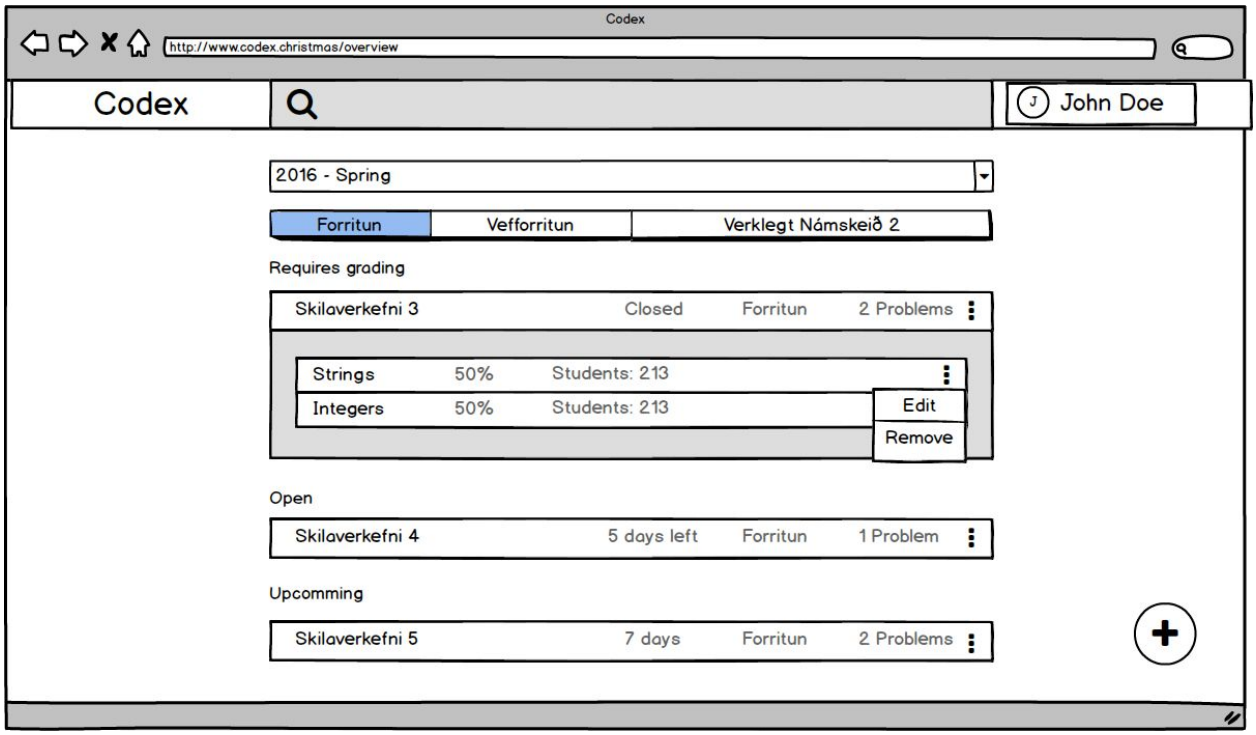


When the teacher clicks on an assignment he/she is shown the problems of said assignment.

Here they can click the problem to display the students for grading.

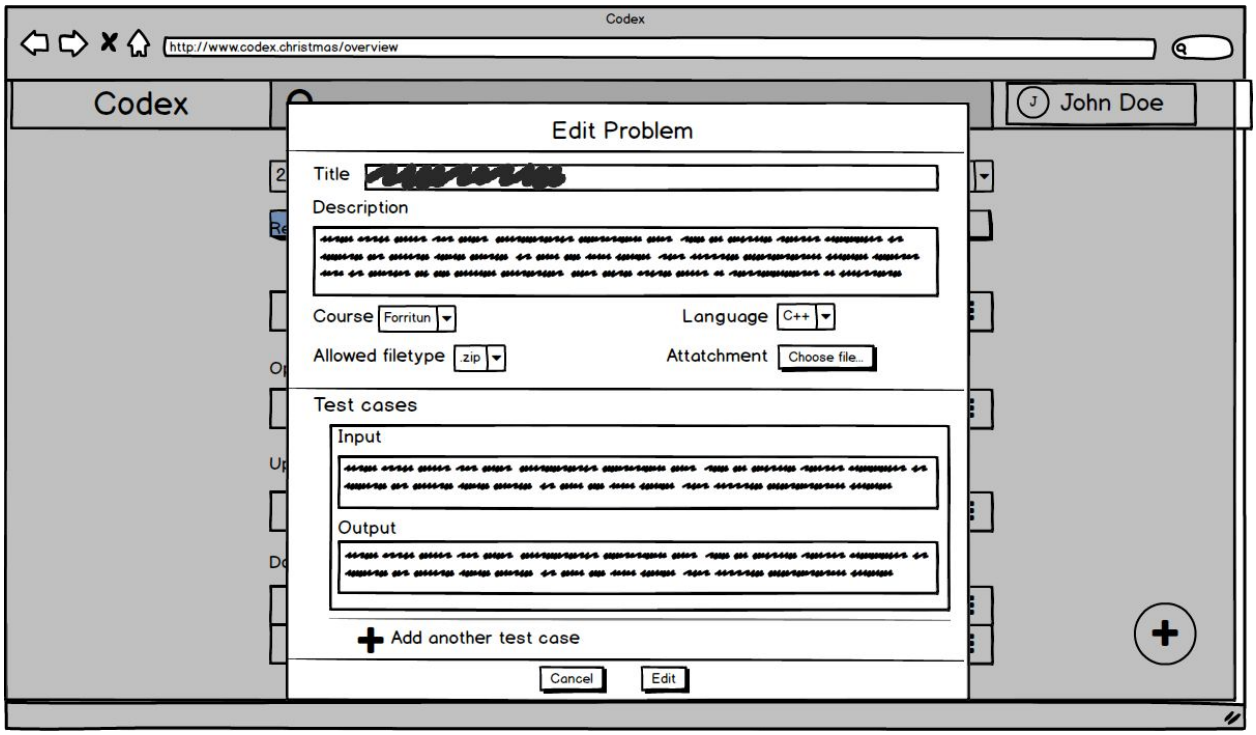
Each problem has a vertical ellipsis, just like the assignments. Clicking it displays a menu for either editing or removing the problem.

Problem dropdown



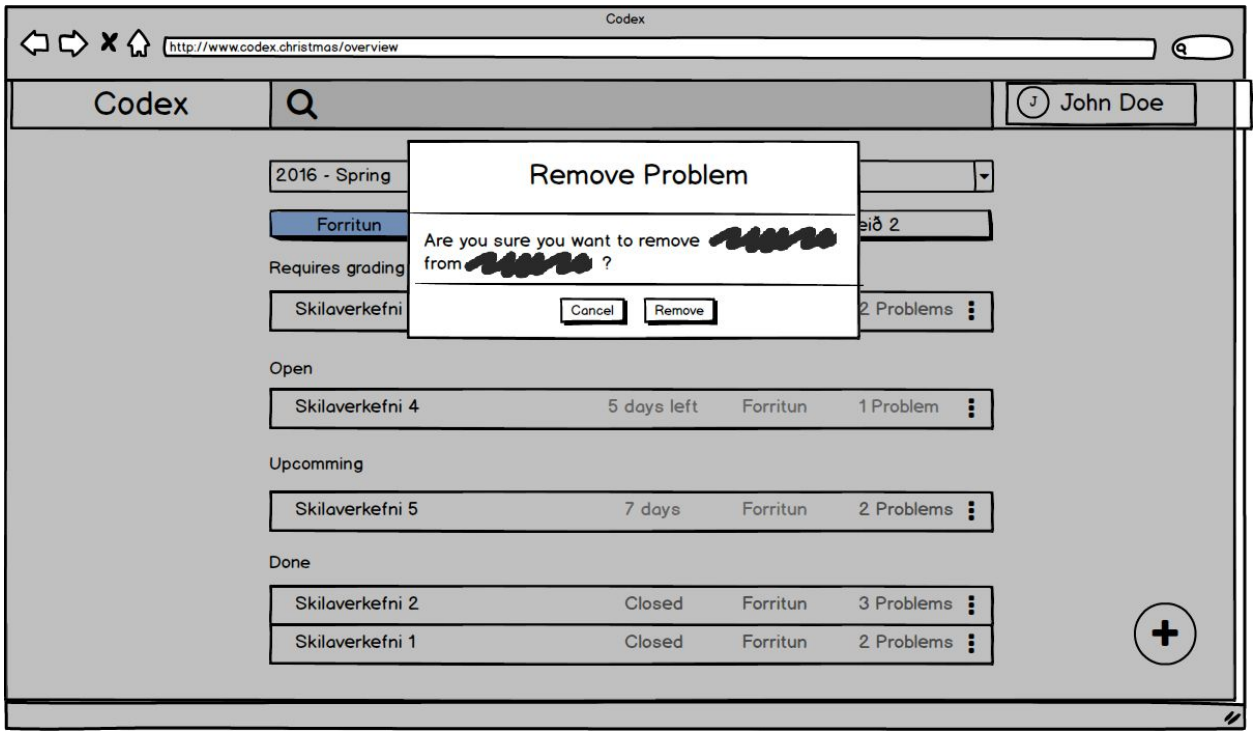
The vertical ellipsis for the problem Strings has been clicked.

Edit problem



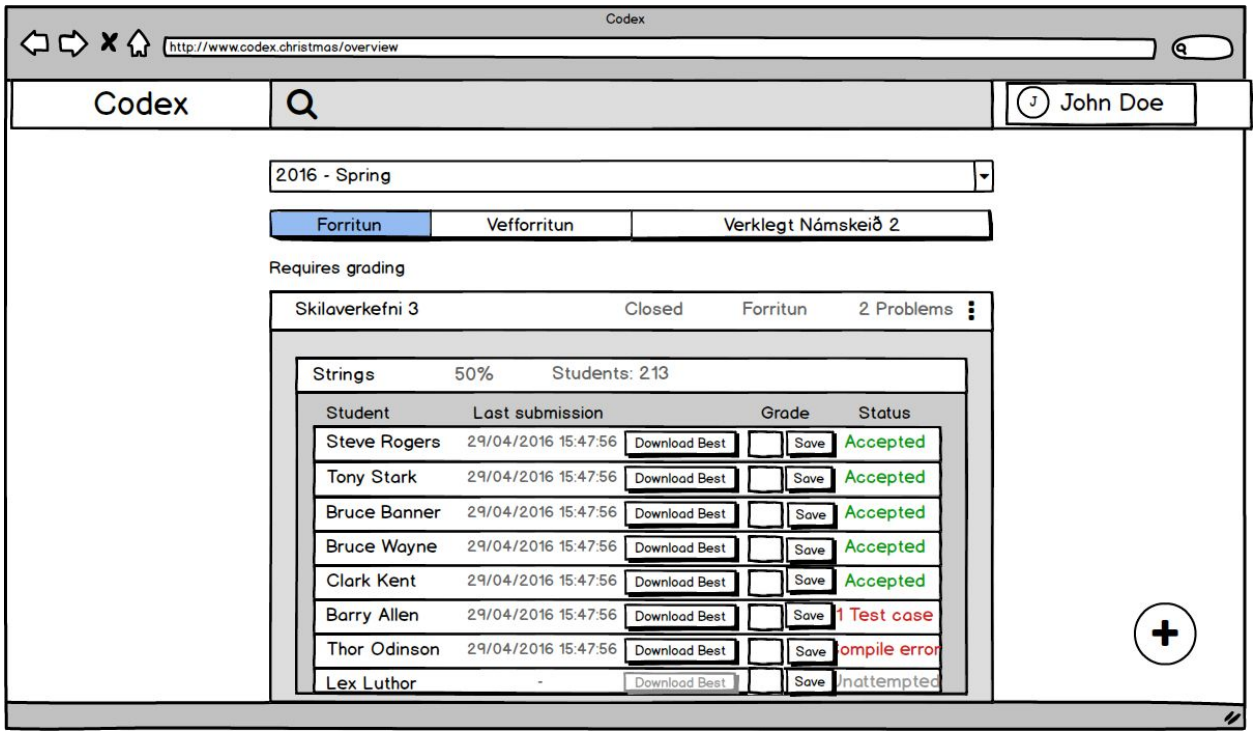
The teacher can change the following for a problem: title, description, course, language, allowed filetype, the problem's attachments and test cases.

Remove problem



When removing a problem a confirmation is displayed.

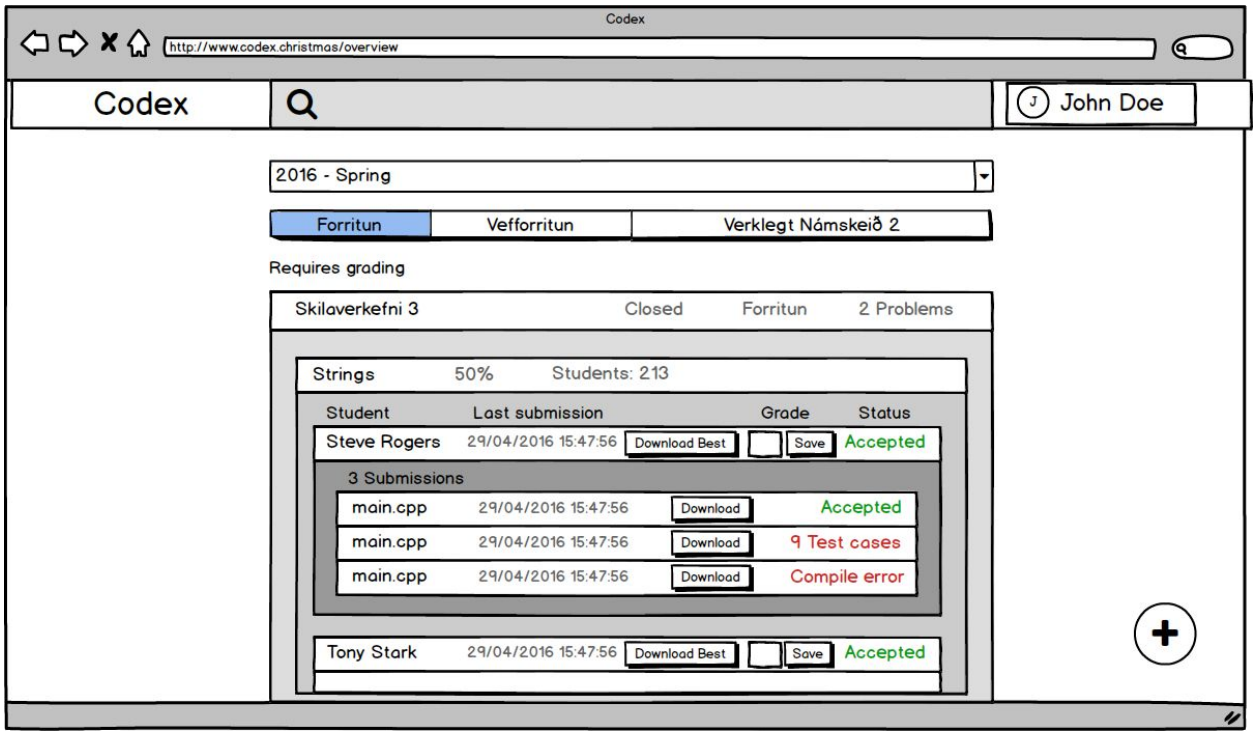
Grading



Here the problem Strings has been clicked, displaying a list of the students who need to be graded.

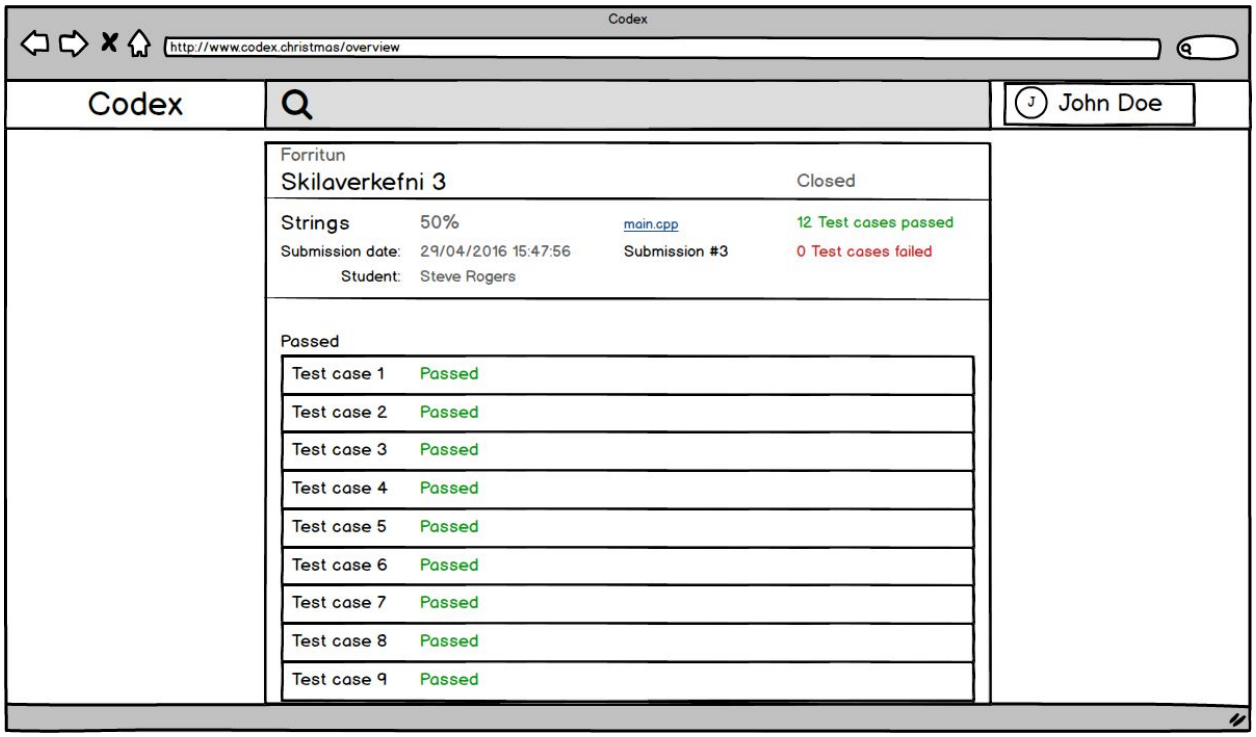
The teacher can see the student's name, when the student submitted last, download the student's best submission, grade the student and see the result of the best submission.

Student submissions



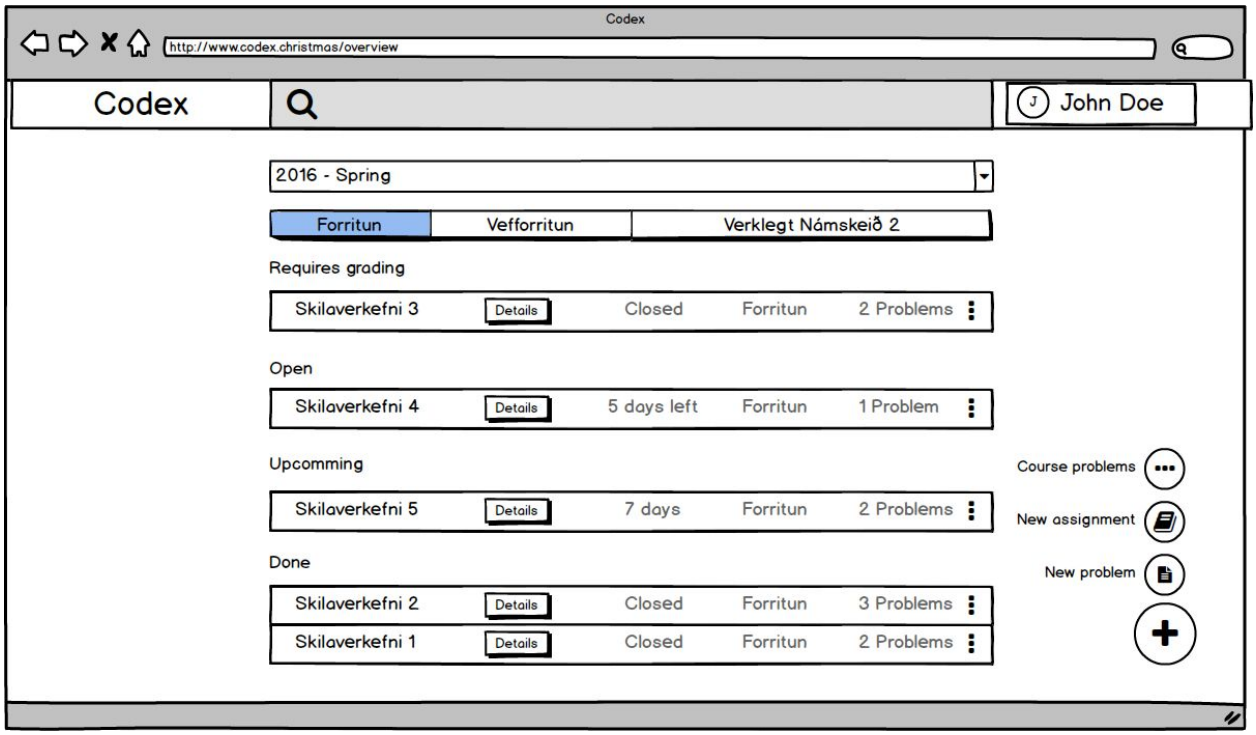
The teacher can click a student in order to see the all of the student's submissions. Clicking on a submission directs the teacher to a the submission details view.

Submission details



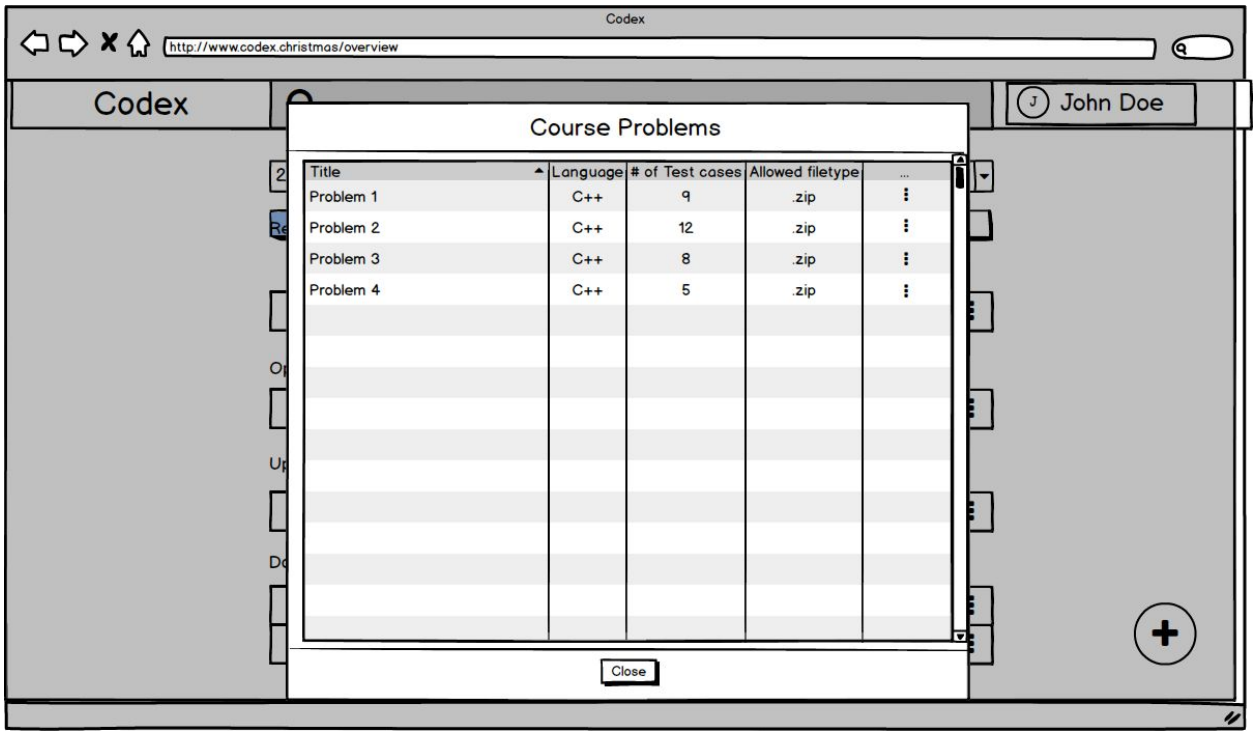
The submissions details view is identical to the student's submissions details view, except for the student's name .

Plus button



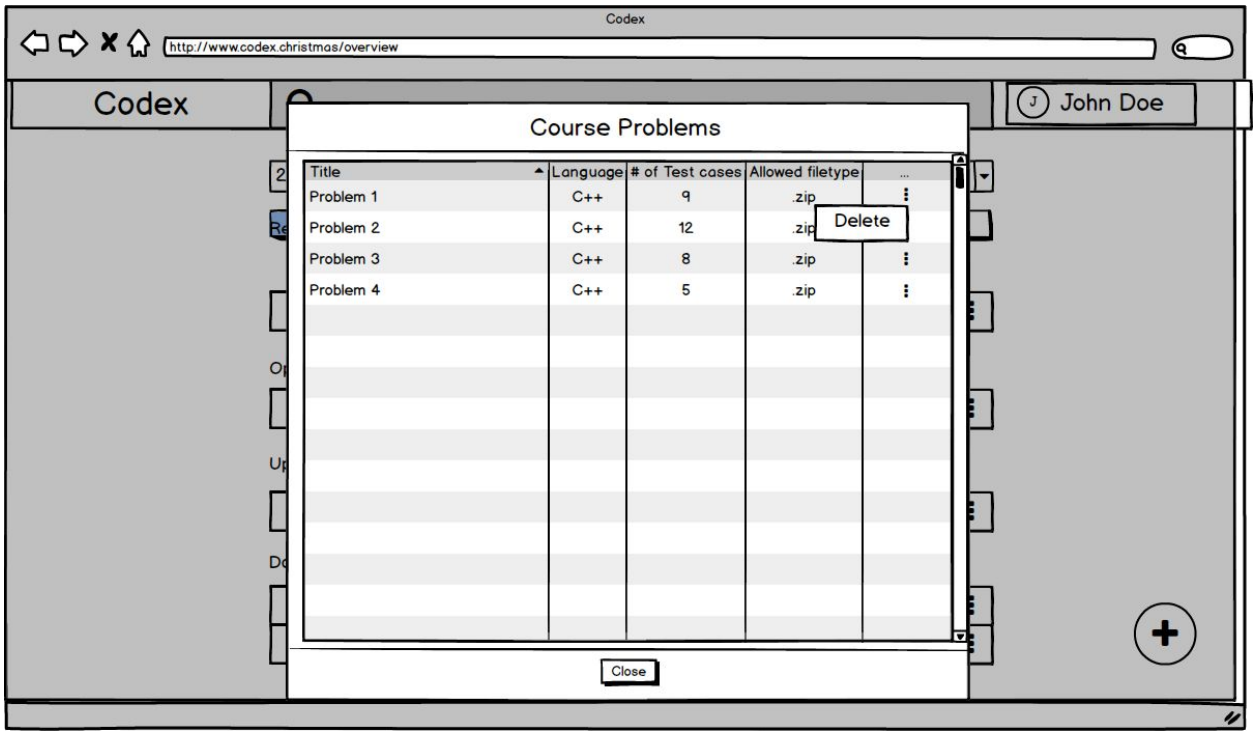
Clicking the plus icon displays three options:
Course problems: Display what problems exist for the selected course.
New assignment: Create a new assignment for the selected course.
New problem: Create a new problem.

Course problems

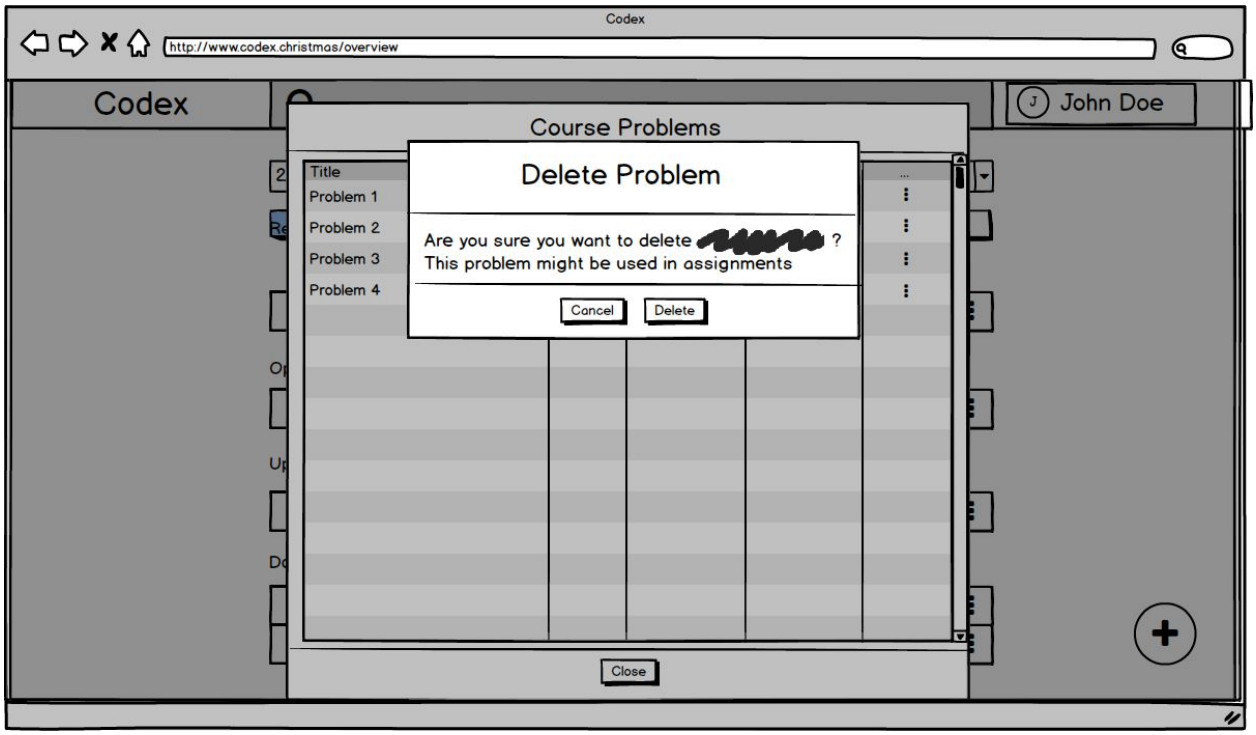


Here the teacher can view the available problems for the course.
If the teacher clicks on a problem, he/she will begin editing it.
Clicking the vertical ellipsis displays the Delete button if the teacher wishes to delete the problem.

Course problems dropdown

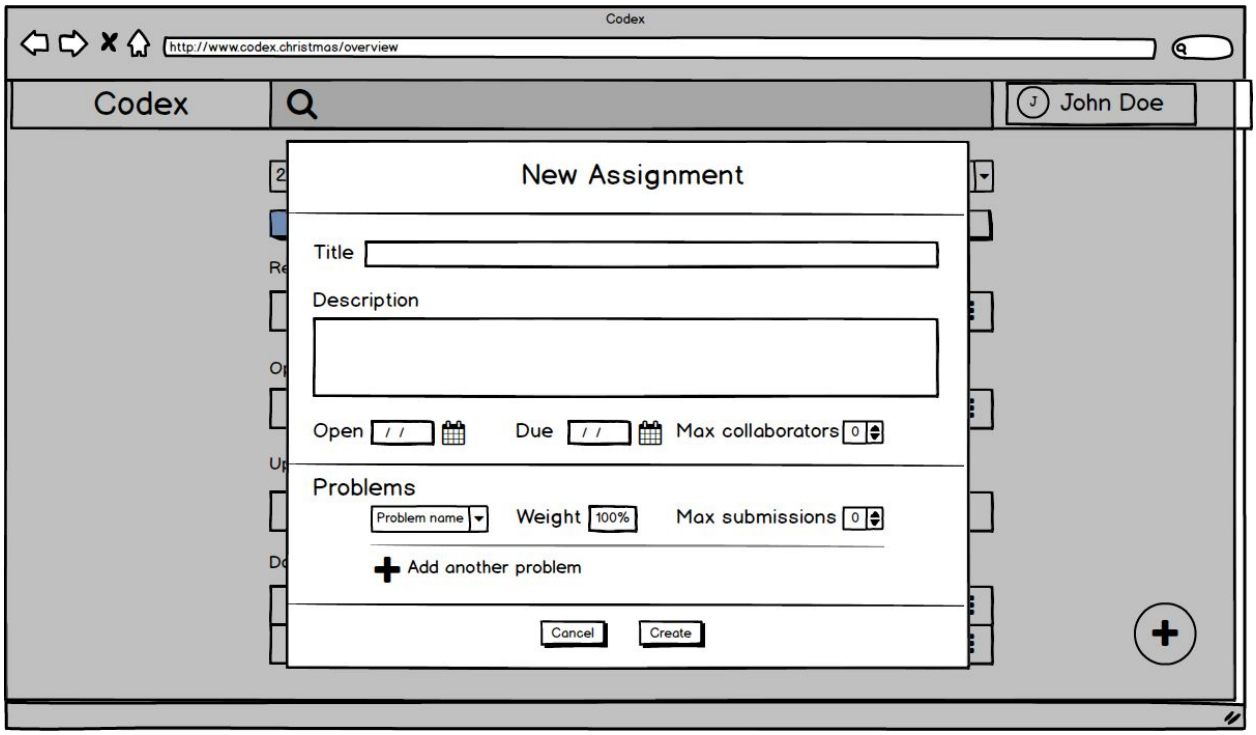


Here the ellipsis for Problem 1 has been clicked.



Clicking the delete button displays a confirmation.

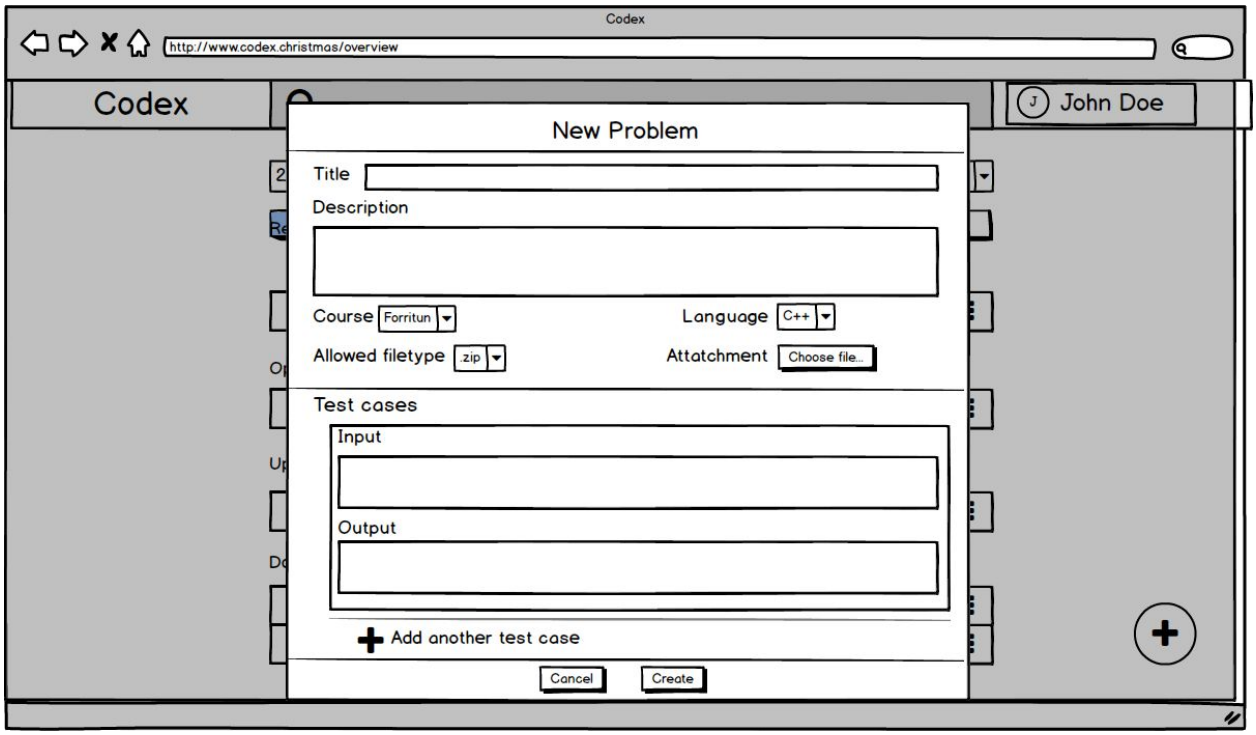
New assignment



When creating a new assignment a modal appears.

The teacher has to specify the following when creating a new assignment: the title, description, when the assignment should open, when the assignment is due, how many can collaborate (0 means it's a solo assignment) and what problems are in the assignment along with their weight and submission limit (0 means no limit).

New problem

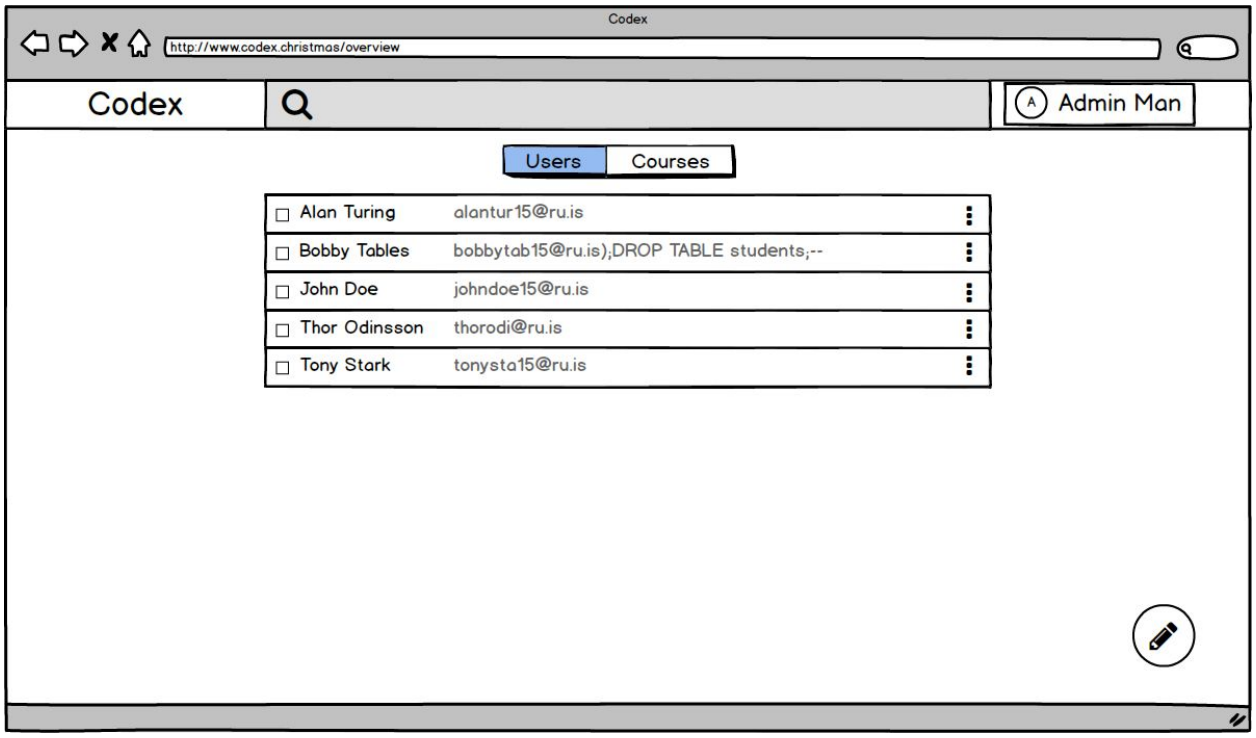


When creating a new problem a modal appears.

The teacher has to specify the following the creating a new problem: the title, description, what course the problem belongs to, language, allowed file type, the problem's attachment and at least one test case.

Admin

Admin overview



The admin's landing page is split in two; the user list and the course list.

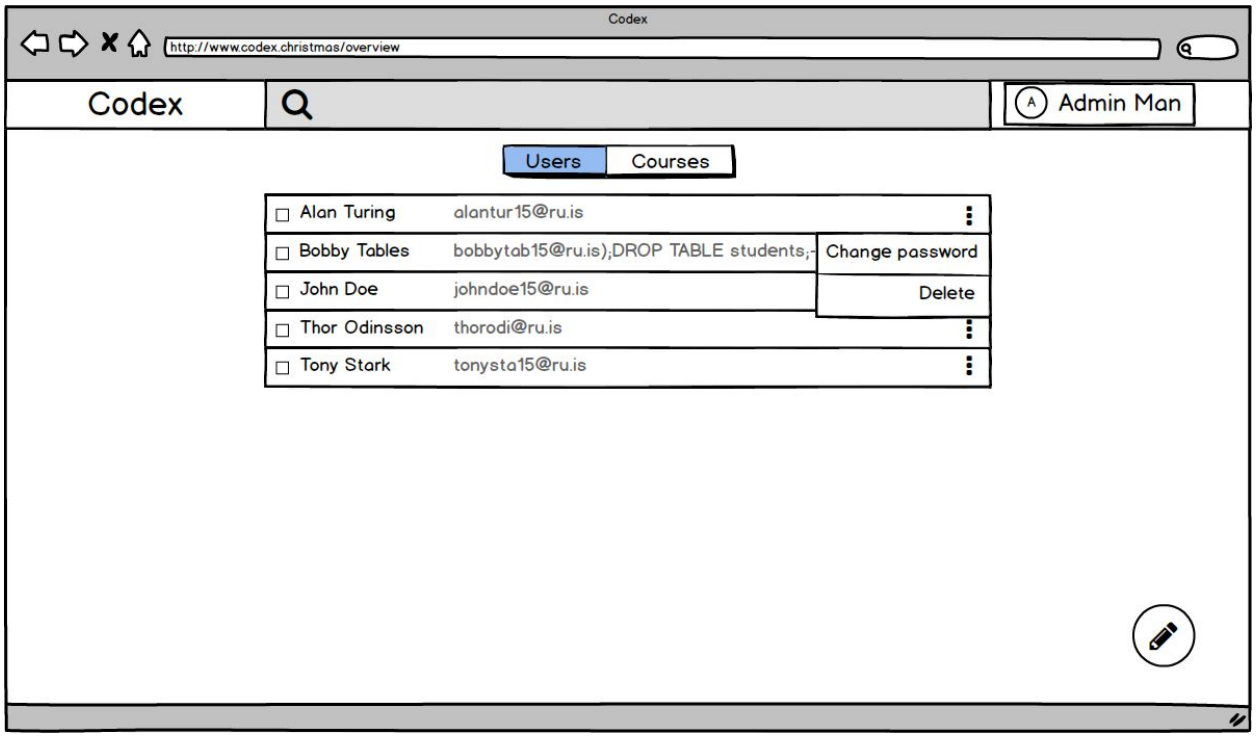
The admin can click on a user to edit the user and click the vertical ellipsis to view the Change password and Delete option.

The checkboxes are used to select multiple users and use the Delete Selected Users option which can be found by clicking the pencil icon in the lower right hand corner.

The search field can search for both users and courses.

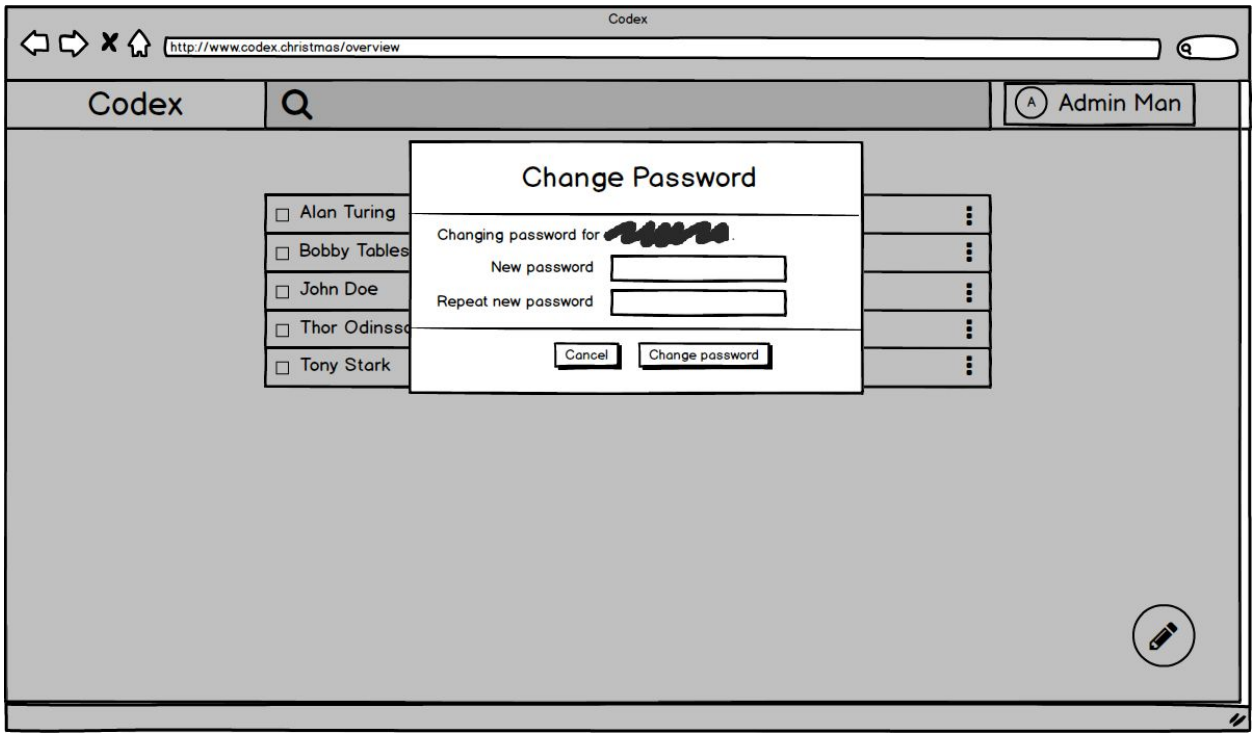
The admin also has a dropdown menu when clicking his/her name in the upper right hand corner. This displays the Settings and Logout options.

User dropdown



Here the vertical ellipsis for Alan Turing has been clicked.

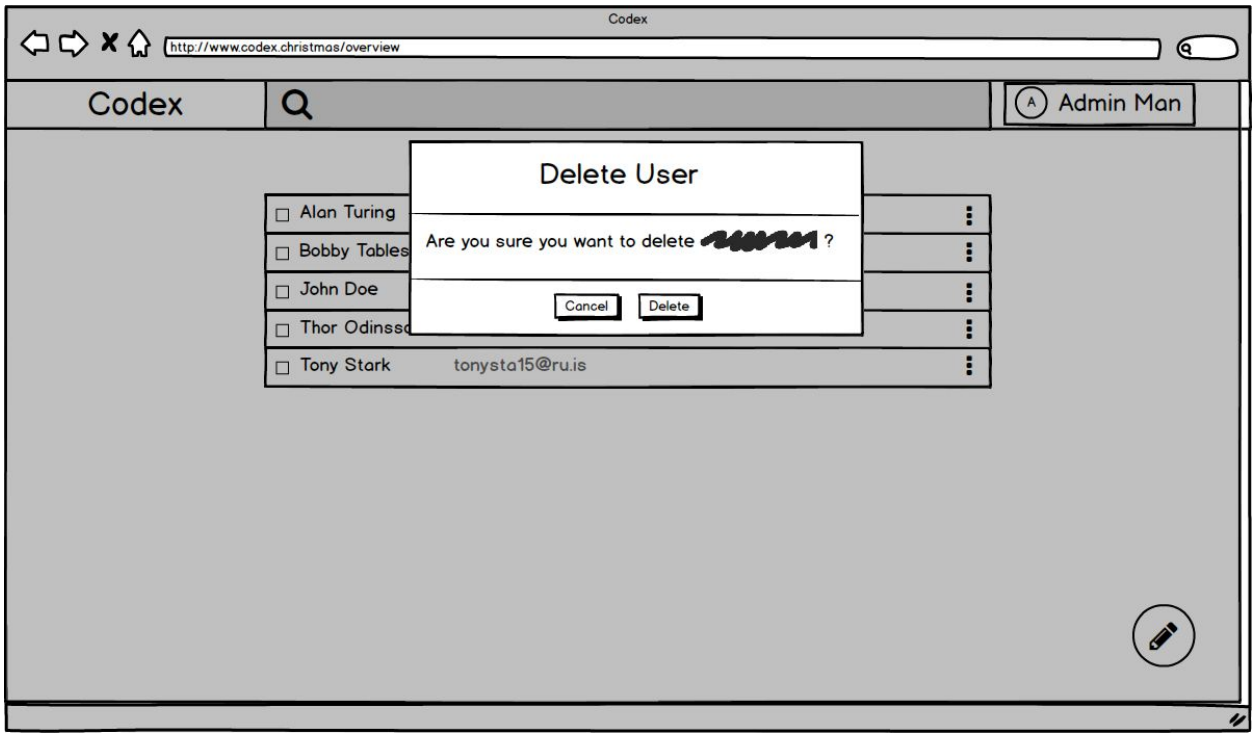
Change password



When the Change password option is selected a modal appears.

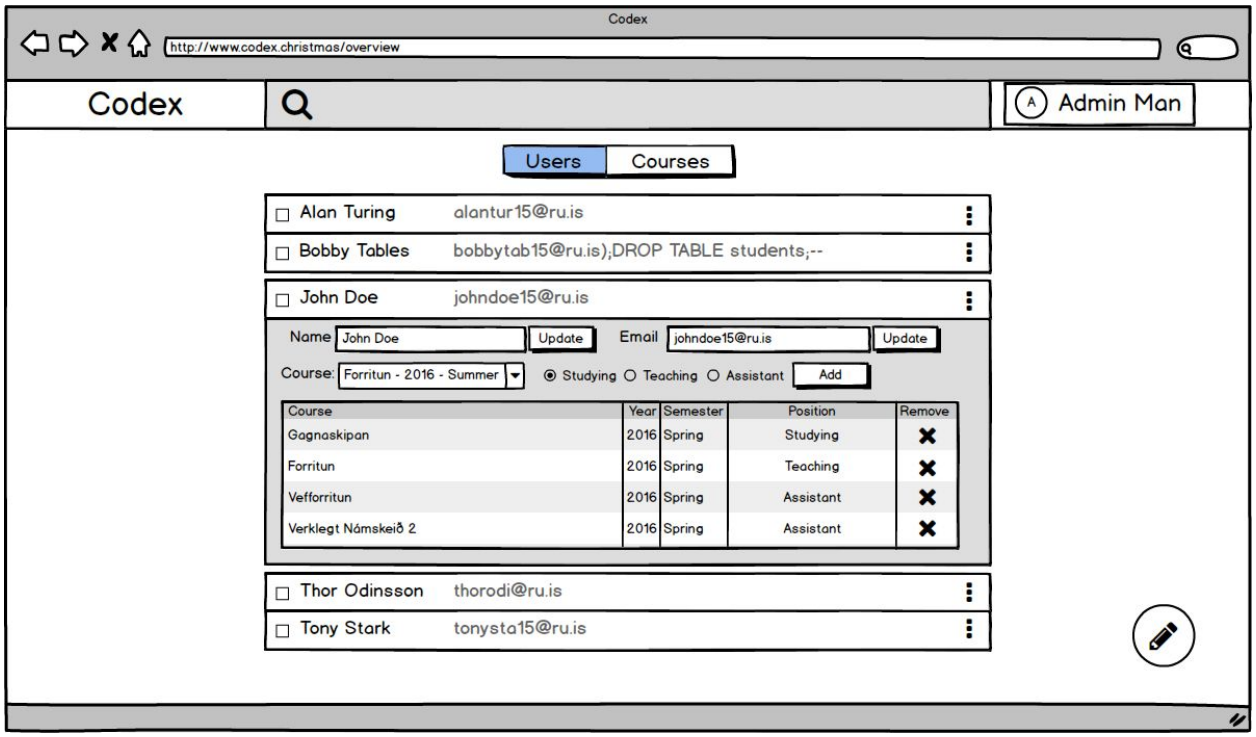
The admin has to input the new password twice.

Delete user



When deleting a user a confirmation is displayed.

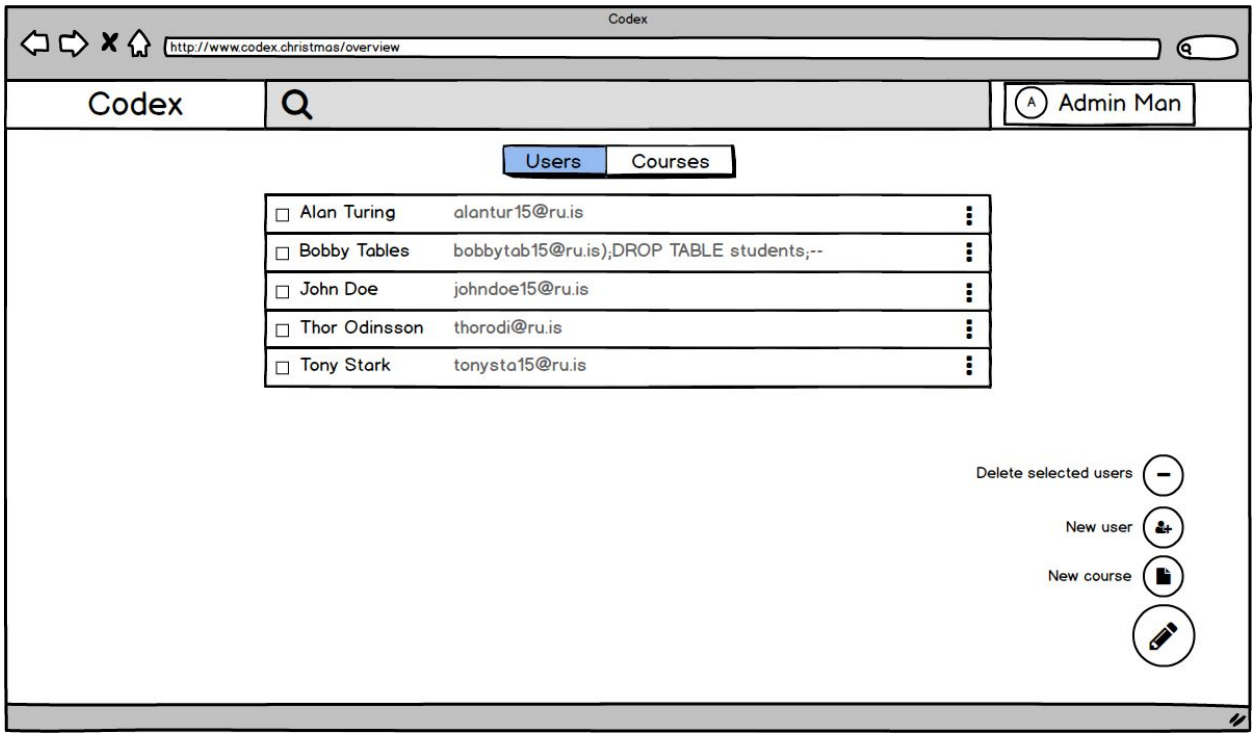
Edit user



Here John Doe has been clicked.

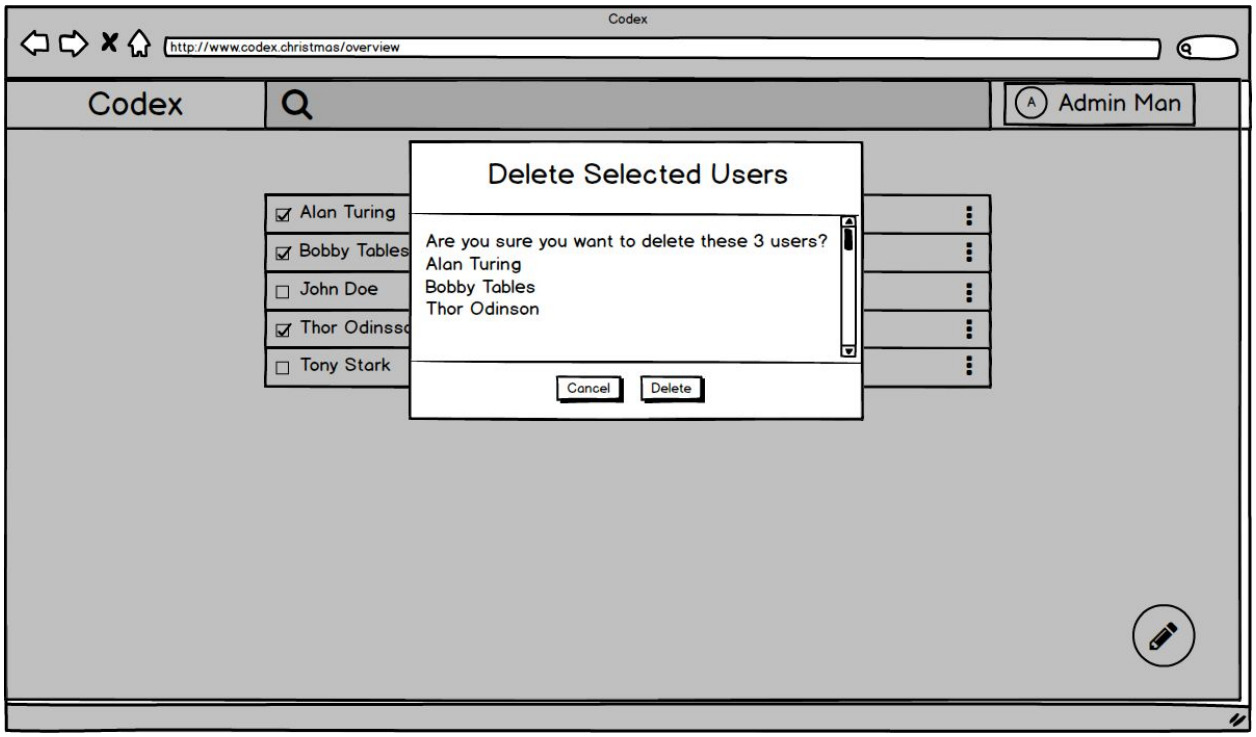
The admin can change the user's name, email and which courses the user is in.

Pencil button



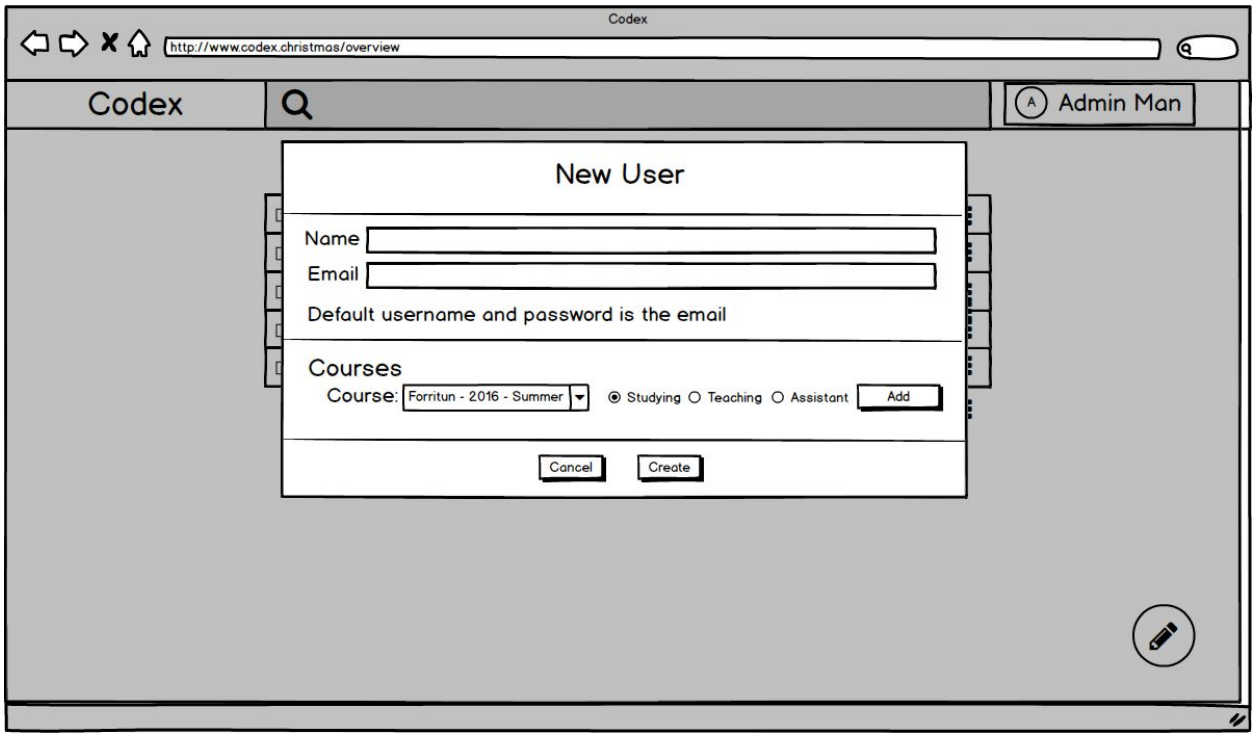
When the admin clicks the pencil button, three options appear:
Delete selected users: Displays a prompt to delete the selected users.
New user: Create a new user
New course: Create a new course

Delete selected users



When selecting multiple users and deleting them a confirmation is displayed.

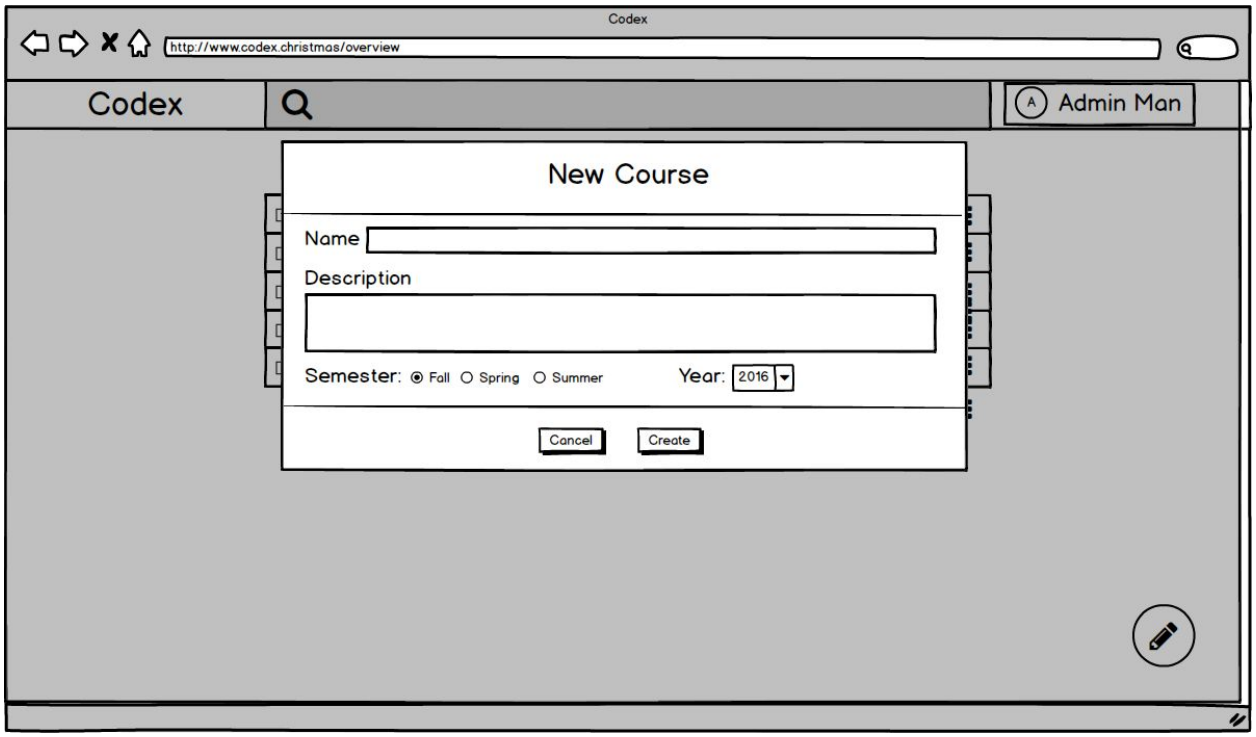
New user



When creating a new user a modal appears.

The admin has to specify the new user's name, email and possibly what courses the user is in and the user's position in the course.

New course



When creating a new course a modal appears

The admin has to specify the new course's name, possibly a description, a semester and year.

Courses

In the Courses tab a list of all courses is displayed.

For each course the admin can see the course's name, year, semester and the number of teachers, assistants and students in the course.

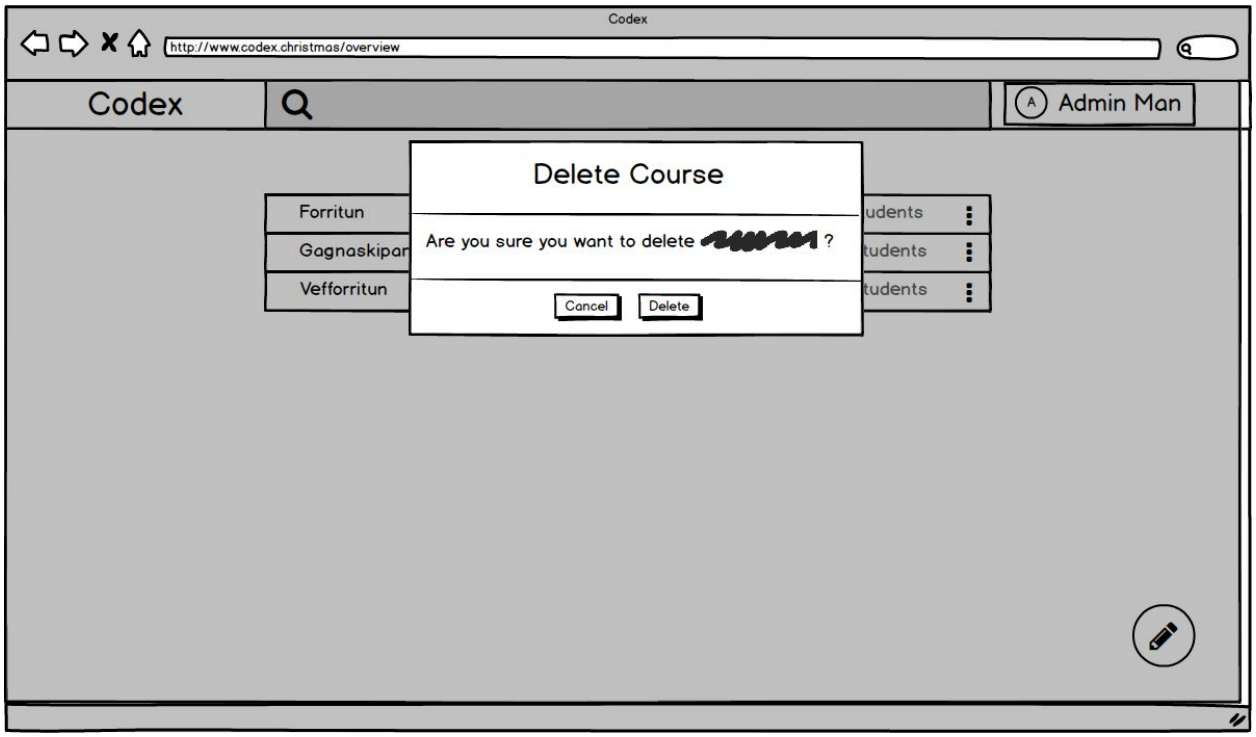
Clicking on a course allows the admin to edit said course.

The vertical ellipsis can be clicked to display the Delete button.

Course dropdown

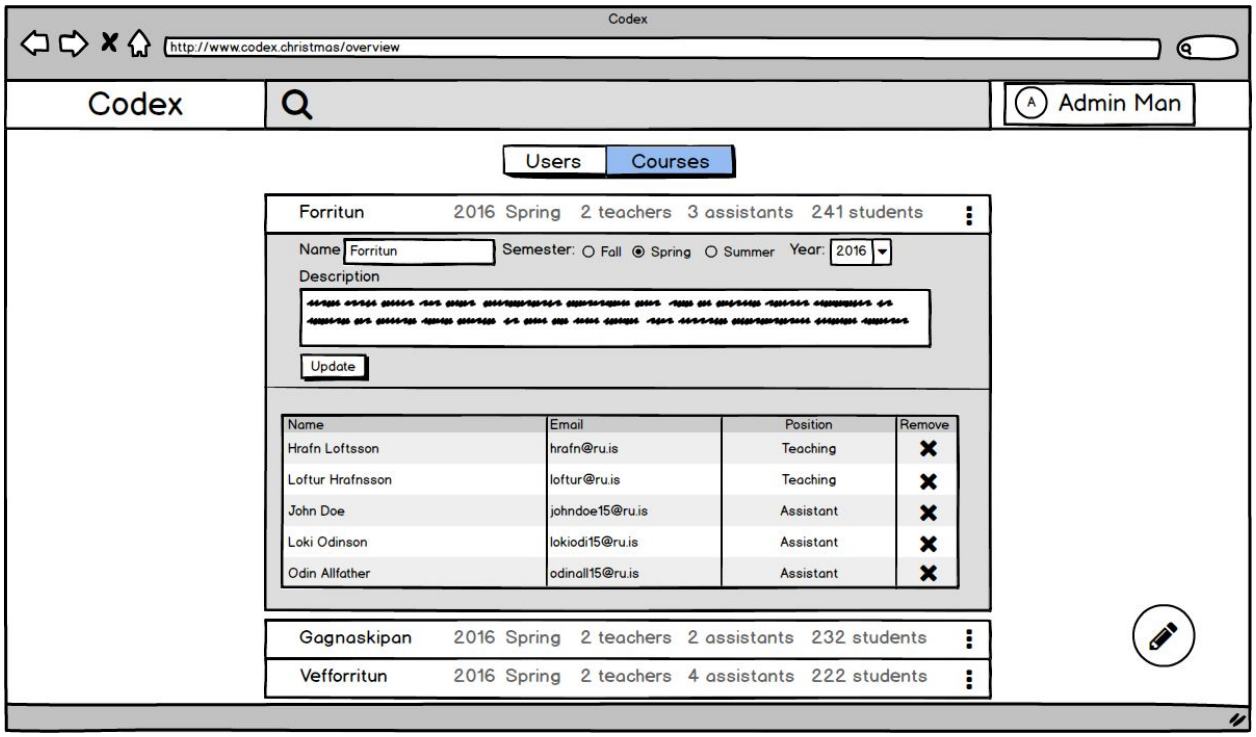
Here the vertical ellipsis for the course Forritun has been clicked.

Delete course



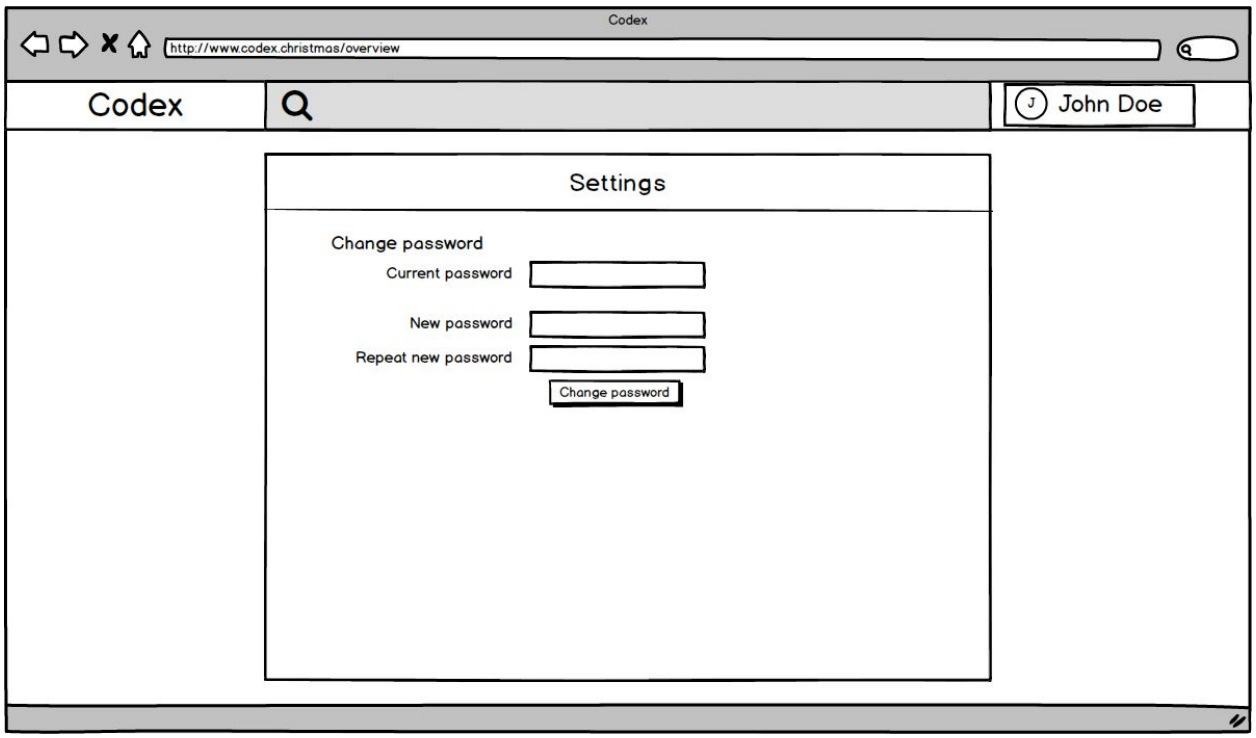
When deleting a course a confirmation is displayed.

Edit course



The admin can edit the course's name, what semester and year the course is on, the course's description and can remove teachers and assistant from the course.

Settings



The settings page, which is accessed from the dropdown menu in the upper right hand corner, is available to everyone.

Here the users of *Codex* can change their passwords.

Prototype evaluation

Having built our prototypes with the wireframing software Balsamiq it enabled us to have students and teachers easily test it. That way we could measure them against our desired usability goals and get their input on the prototypes.

We conducted the tests by giving each participant a few different tasks according to our usability goals but specifying details when needed. Below we have added the usability goal tables from the requirement analysis report and updated them with the test results.

Student tasks

We got 4 students to try out each task.

Student	Age	Education	Computer Skills
Student 1	21	1st year at HR	Good
Student 2	24	1st year at HR	Good
Student 3	19	1st year at HR	Very Good
Student 4	33	1st year at HR	Good

Student submits a solution to the *Permutations* problem in *Skilaverkefni 3*

Effectiveness: All students finished the task

Efficiency: 10sec, 14sec, 10sec, 21sec (Average: 13,75)

Satisfaction: Very Good, Very Good, Extremely Good, Extremely Good (4.5)

Usability factor	Data collected	Worst case	Preferred case	Best case	Value now
Effectiveness	Finished task	70%	100%	100%	100%
Efficiency	Time to submit a solution	40 seconds	20 seconds	10 seconds	13,75
Satisfaction	The look and feel of website	Not good	Very good	Extremely good	4.5

Results

When users reach their home view, submitting a solution to a problem in an open assignment is only 2-clicks away (Browse for file + submit file). The students were generally very happy with the interface and quicker than we expected to finish the task.

Student downloads his/her submission to the *Permutations* problem in *Skilaverkefni 3*

Efficiency: All students finished the task

Effectiveness: 5sec, 6sec, 11sec, 7sec (Average: 7,25)

Satisfaction: Extremely Good, Extremely Good, Extremely Good, Extremely Good (5)

Usability factor	Data collected	Worst case	Preferred case	Best case	Value now
Effectiveness	Finished task	70%	100%	100%	100%
Efficiency	Time to download the submission	40 seconds	20 seconds	10 seconds	7,25
Satisfaction	The look and feel of website	Not good	Very good	Extremely good	5

Results

Having already seen some of the interface the students were extremely quick finishing this task. And they we're all extremely happy with the interface.

Teacher tasks

We found 1 student to impersonate a teacher.

Teacher	Age	Education	Computer Skills
Teacher 1	28	3rd year at HR	Very Good

Teachers downloads the best solution from *Barry Allen* for *Strings* problem in *Skilaverkefni 3*

Effectiveness: 100%

Efficiency: 16sec

Satisfaction: Very good (4)

Usability factor	Data collected	Worst case	Preferred case	Best case	Value now
Effectiveness	Finished task	80%	100%	100%	100%
Efficiency	Time to download a submission	50 seconds	20 seconds	10 seconds	16
Satisfaction	The look and feel of website	Not good	Very good	Extremely good	4

Results

The teacher was relatively quick finishing the task, his comments were positive. He liked the interface and especially the ordering of assignments on the landing page.

Administrator tasks

We found a network administrator to impersonate an administrator

Administrator	Age	Education	Computer Skills
Administrator 1	21	1st year at HR	Very Good

Administrator creates a new user

Effectiveness: 100%

Efficiency: 43sec

Satisfaction: Good (3)

Usability factor	Data collected	Worst case	Preferred case	Best case	Value now
Effectiveness	Finished task	70%	100%	100%	100%
Efficiency	Time to create a new user	1.5 minutes	40 seconds	20 seconds	43
Satisfaction	The look and feel of website	Not good	Very good	Extremely good	3

Results

The test subject was a bit longer to finish the task than we expected. However after finishing it he was very happy with the interface and blamed the mockup for not figuring out the router faster. When inside the view for all students there's an edit menu at the bottom right just like in every other android app, which most users are familiar with.

Database entity relation diagram

This is the ERD for the database that helped us when designing the database for Codex. The entities are displayed as rectangles and the diamonds are the relations between entities. Both entities and relations are denoted as weak if they have double lined edges. The relations that are many-to-many have names which is their table name in the database.

Weak entities cannot exist without the stronger entities it is related to and they hold a foreign key to those entities.

The database can be more thoroughly understood by looking at the table schema / class diagram in the next chapter.

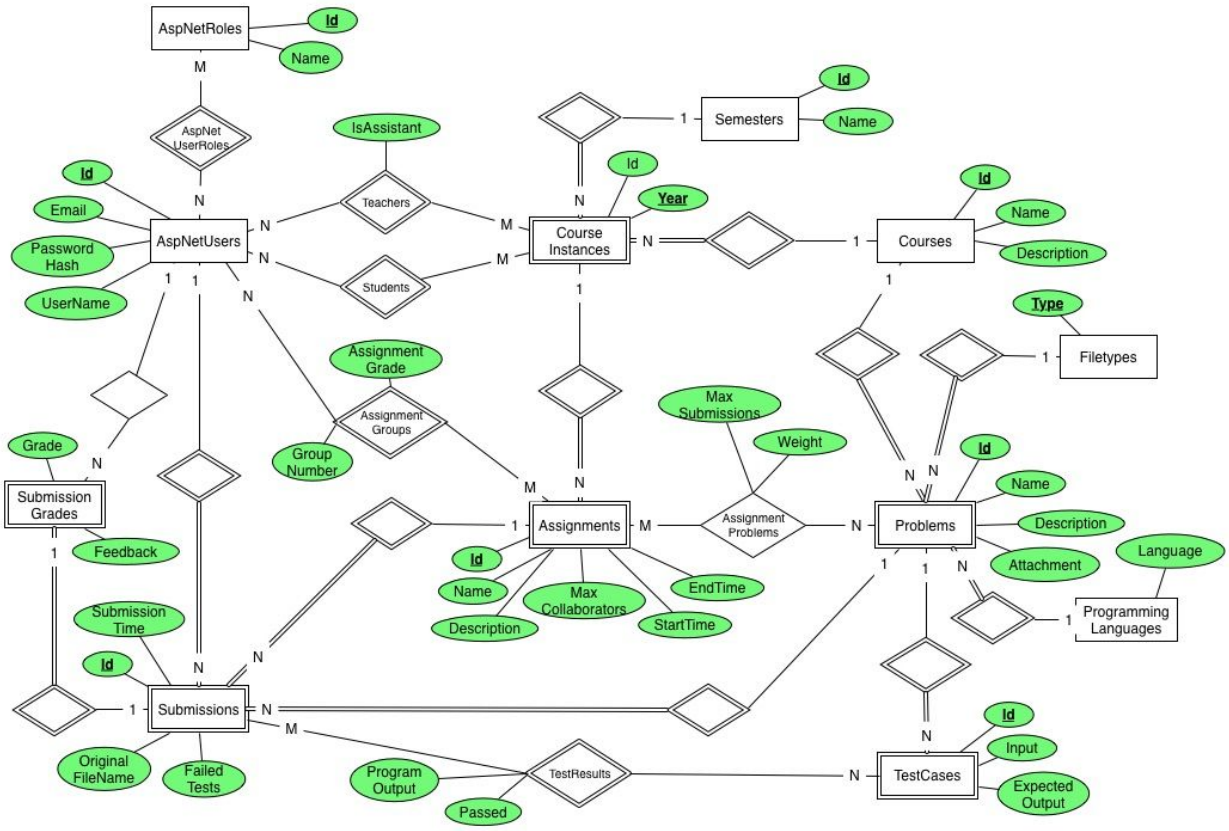
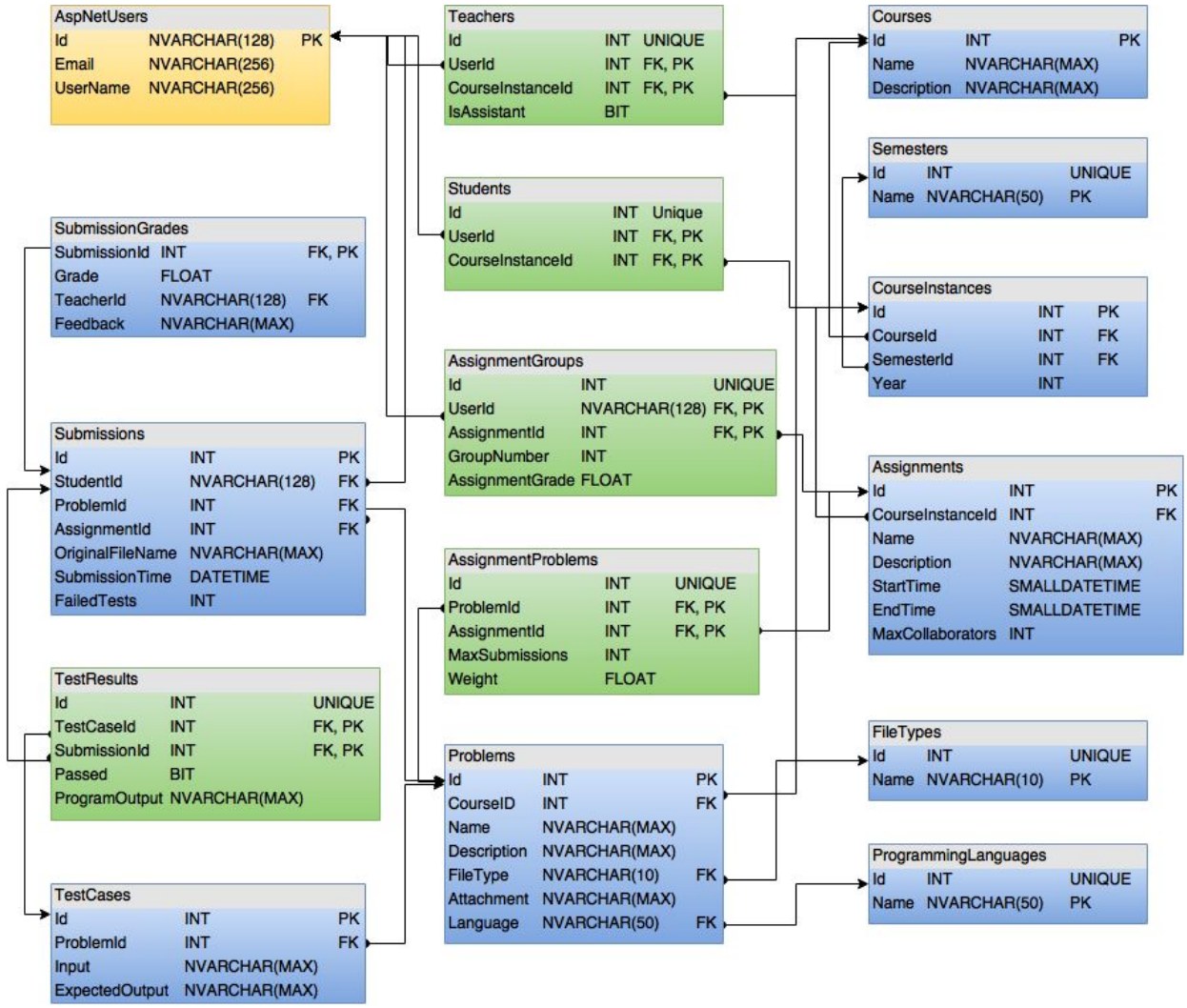


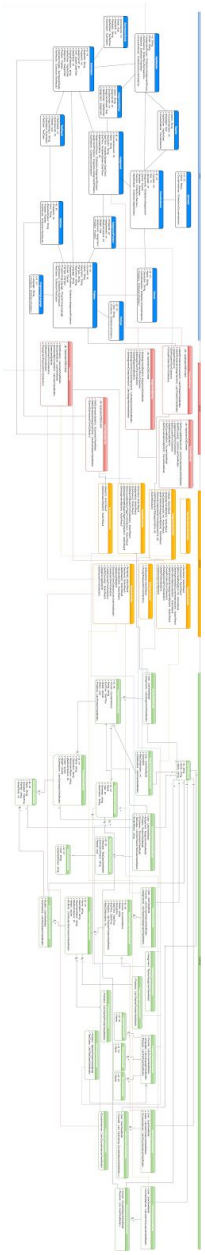
Table schema

When building the database we implemented the ASP.NET identity and built from that. The AspNetUsers table is a part of the Identity. The blue tables are the entities from the ERD diagram while the green tables are the many-to-many relation tables between them. The relation tables have a primary key constraint between the foreign keys that come from the tables they are connecting together. We also added a unique id that we thought might come in handy when coding. The connections in the schema are to visualize where the foreign key comes from.



Class diagram

The class diagram is also in a more readable format at the bottom of the document in the appendix.



Coding rules

The source code of Codex should be written with the following rules.

C#

- PascalCasing should be used as the default naming convention.
- camelCasing should be used for local variables and parameters.
  This does not apply to C# properties.
- Member variables should use camelCasing in addition to having an underscore prefix.
- Constant variables should be written in uppercase.
- Place one space between variables and operators.
- The placement of curly braces will use Stroustrup style.
- The indent style will use tabs, which are 4 characters wide.
- When possible and appropriate prefer using a foreach loop.
- Write an XML summary block for every public class, method and property.
- Write a comment for every private member variable.

JavaScript

- camelCase should be used as the default naming convention.
- Constant and global variables should be written in uppercase.
- The placement of curly braces will use Stroustrup style.
- The indent style will use tabs, which are 4 characters wide.
- Place one space between variables and operators.

SQL

- PascalCase should be used for table names
- camelCase should be used for column names
- Every table must have an integer Id field.

HTML

- Document doctype should be HTML5.
- The indent style will use tabs, which are 4 characters wide.
- Use only lowercase letters when applying html elements.
- Remove all trailing white spaces.
- Use a new line for every block, list, or table element, and indent every such child element.
- When quoting attributes values, use double quotation marks.

CSS

- The placement of curly braces will use Stroustrup style.
- Use meaningful or generic ID and class names.
- Separate words in ID and class names by a hyphen.
- Use shorthand properties where possible and when it applies.
- Place leading “0's” in values. (Example - padding: 0.5em;)
- Use 3 character hexadecimal notation where possible.
- Always separate rules by adding a new line (two line breaks) between rules.

Appendix

Database definition

```
/* The predefined AspNetUsers Table */
CREATE TABLE [dbo].[AspNetUsers] (
    [Id]                   NVARCHAR (128) NOT NULL,
    [Email]                NVARCHAR (256) NULL,
    [EmailConfirmed]       BIT            NOT NULL,
    [PasswordHash]         NVARCHAR (MAX) NULL,
    [SecurityStamp]        NVARCHAR (MAX) NULL,
    [PhoneNumber]          NVARCHAR (MAX) NULL,
    [PhoneNumberConfirmed] BIT            NOT NULL,
    [TwoFactorEnabled]     BIT            NOT NULL,
    [LockoutEndDateUtc]    DATETIME       NULL,
    [LockoutEnabled]       BIT            NOT NULL,
    [AccessFailedCount]    INT            NOT NULL,
    [UserName]             NVARCHAR (256) NOT NULL,
    CONSTRAINT [PK_dbo.AspNetUsers] PRIMARY KEY CLUSTERED ([Id] ASC)
);

CREATE TABLE [dbo].[Courses]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [Name] NVARCHAR(MAX) NOT NULL,
    [Description] NVARCHAR(MAX)
)

CREATE TABLE [dbo].[Semesters]
(
    [Id] INT NOT NULL UNIQUE IDENTITY,
    [Name] NVARCHAR(50) NOT NULL PRIMARY KEY
)

CREATE TABLE [dbo].[CourseInstances]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [CourseId] INT NOT NULL,
    [Year] INT NOT NULL,
    [SemesterId] INT NOT NULL,
    CONSTRAINT [FK_CourseInstances_Courses] FOREIGN KEY ([CourseId]) REFERENCES [Courses]([Id]),
    CONSTRAINT [FK_CourseInstances_Semesters] FOREIGN KEY ([SemesterId]) REFERENCES [Semesters]([Id])
)

/* Teachers many-to-many relation between AspNetUsers and CourseInstances with Assistant Flag */
CREATE TABLE [dbo].[Teachers]
(
    [Id] INT NOT NULL UNIQUE IDENTITY,
    [UserId] NVARCHAR(128) NOT NULL,
    [CourseInstanceId] INT NOT NULL,
    [IsAssistant] BIT DEFAULT 0,
    CONSTRAINT [FK_Teachers_AspNetUsers] FOREIGN KEY ([UserId]) REFERENCES [AspNetUsers]([Id]),
    CONSTRAINT [FK_Teachers_CourseInstances] FOREIGN KEY ([CourseInstanceId]) REFERENCES [CourseInstances]([Id]),
    CONSTRAINT [PK_Teachers] PRIMARY KEY ([UserId], [CourseInstanceId])
)

/* The relation between AspNetUsers and CourseInstances */
CREATE TABLE [dbo].[Students]
(
    [Id] INT NOT NULL UNIQUE IDENTITY,
    [UserId] NVARCHAR(128) NOT NULL,
    [CourseInstanceId] INT NOT NULL,
    CONSTRAINT [FK_Students_AspNetUsers] FOREIGN KEY ([UserId]) REFERENCES [AspNetUsers]([Id]),
    CONSTRAINT [FK_Students_CourseInstances] FOREIGN KEY ([CourseInstanceId]) REFERENCES [CourseInstances]([Id]),
    CONSTRAINT [PK_Students] PRIMARY KEY ([UserId], [CourseInstanceId])
)

```

```
CREATE TABLE [dbo].[Filetypes]
(
    [Id] INT NOT NULL UNIQUE IDENTITY,
    [Type] NVARCHAR(10) NOT NULL PRIMARY KEY
)

CREATE TABLE [dbo].[ProgrammingLanguages]
(
    [Id] INT NOT NULL UNIQUE IDENTITY,
    [Language] NVARCHAR(50) NOT NULL PRIMARY KEY
)

/* Problem is part of an Assignment, what is called Milestones in Centris.
Problems can be reused and belong to multiple Assignments */
CREATE TABLE [dbo].[Problems]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [CourseId] INT NOT NULL,
    [Name] NVARCHAR(MAX) NOT NULL,
    [Description] NVARCHAR(MAX),
    [Filetype] NVARCHAR(10) NOT NULL,
    [Attachment] NVARCHAR(MAX),
    [Language] NVARCHAR(50) NOT NULL,
    CONSTRAINT [FK_Problems_Courses] FOREIGN KEY ([CourseId]) REFERENCES [Courses]([Id]),
    CONSTRAINT [FK_Problems_Filetypes] FOREIGN KEY ([Filetype]) REFERENCES [Filetypes]([Type]),
    CONSTRAINT [FK_Problems_ProgrammingLanguages] FOREIGN KEY ([Language]) REFERENCES [ProgrammingLanguages]([Language])
)

/* SmallDateTime stores YYYY-MM-DD HH:MM */
CREATE TABLE [dbo].[Assignments]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [CourseInstanceId] INT NOT NULL,
    [Name] NVARCHAR(MAX) NOT NULL,
    [Description] NVARCHAR(MAX),
    [StartTime] SMALLDATETIME,
    [EndTime] SMALLDATETIME,
    [MaxCollaborators] INT NOT NULL DEFAULT 1,
    CONSTRAINT [FK_Assignments_CourseInstances] FOREIGN KEY ([CourseInstanceId]) REFERENCES [CourseInstances]([Id])
)

/* The many-to-many relation between Assignments and Problems */
CREATE TABLE [dbo].[AssignmentProblems]
(
    [Id] INT NOT NULL UNIQUE IDENTITY,
    [ProblemId] INT NOT NULL,
    [AssignmentId] INT NOT NULL,
    [MaxSubmissions] INT NOT NULL DEFAULT 0,
    [Weight] FLOAT NOT NULL DEFAULT 0,
    CONSTRAINT [FK_AssignmentProblems_Problems] FOREIGN KEY ([ProblemId]) REFERENCES [Problems]([Id]),
    CONSTRAINT [FK_AssignmentProblems_Assignments] FOREIGN KEY ([AssignmentId]) REFERENCES [Assignments]([Id]),
    CONSTRAINT [PK_AssignmentProblems] PRIMARY KEY ([ProblemId], [AssignmentId])
)

/* When assignments are created, all Students in that course instance are assigned a group.
Students can then be joined together in a group by updating the groupNumber so it matches the students. */
CREATE TABLE [dbo].[AssignmentGroups]
(
    [Id] INT NOT NULL UNIQUE IDENTITY,
    [UserId] NVARCHAR(128) NOT NULL,
    [AssignmentId] INT NOT NULL,
    [GroupNumber] INT NOT NULL IDENTITY,
    [AssignmentGrade] FLOAT,
    CONSTRAINT [FK_AssignmentGroups_AspNetUsers] FOREIGN KEY ([UserId]) REFERENCES [AspNetUsers]([Id]),
    CONSTRAINT [FK_AssignmentGroups_Assignments] FOREIGN KEY ([AssignmentId]) REFERENCES [Assignments]([Id]),
    CONSTRAINT [PK_AssignmentGroups] PRIMARY KEY ([UserId], [AssignmentId])
)
```

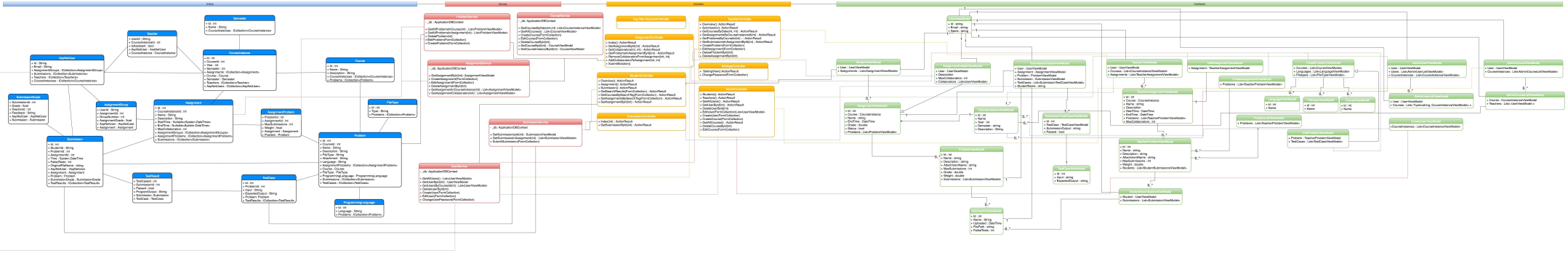
```
/* A student can submit a solution to a Problem in an Assignment, these are called Submissions.
The amount of possible Submissions can be limited in the AssignmentProblems table */
CREATE TABLE [dbo].[Submissions]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [StudentId] NVARCHAR(128) NOT NULL,
    [ProblemId] INT NOT NULL,
    [AssignmentId] INT NOT NULL,
    [SubmissionTime] DATETIME NOT NULL,
    [OriginalFileName] NVARCHAR(256) NOT NULL,
    [FailedTests] INT,
    CONSTRAINT [FK_Submissions_AspNetUsers] FOREIGN KEY ([StudentId]) REFERENCES [AspNetUsers]([Id]),
    CONSTRAINT [FK_Submissions_Problems] FOREIGN KEY ([ProblemId]) REFERENCES [Problems]([Id]),
    CONSTRAINT [FK_Submissions_Assignments] FOREIGN KEY ([AssignmentId]) REFERENCES [Assignments]([Id]),
)

CREATE TABLE [dbo].[SubmissionGrades]
(
    [SubmissionId] INT NOT NULL,
    [Grade] FLOAT,
    [TeacherId] NVARCHAR(128) NOT NULL,
    [Feedback] NVARCHAR(MAX),
    CONSTRAINT [FK_SubmissionGrades_Submissions] FOREIGN KEY ([SubmissionId]) REFERENCES [Submissions]([Id]),
    CONSTRAINT [FK_SubmissionGrades_AspNetUsers] FOREIGN KEY ([TeacherId]) REFERENCES [AspNetUsers]([Id]),
    PRIMARY KEY ([SubmissionId])
)

CREATE TABLE [dbo].[TestCases]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [ProblemId] INT NOT NULL,
    [Input] NVARCHAR(MAX),
    [ExpectedOutput] NVARCHAR(MAX),
    CONSTRAINT [FK_TestCases_Problems] FOREIGN KEY ([ProblemId]) REFERENCES [Problems]([Id])
)

CREATE TABLE [dbo].[TestResults]
(
    [Id] INT NOT NULL UNIQUE IDENTITY,
    [TestCaseId] INT NOT NULL,
    [SubmissionId] INT NOT NULL,
    [Passed] BIT DEFAULT 0,
    [ProgramOutput] NVARCHAR(MAX),
    CONSTRAINT [FK_TestResults_TestCases] FOREIGN KEY ([TestCaseId]) REFERENCES [TestCases]([Id]),
    CONSTRAINT [FK_TestResults_Submissions] FOREIGN KEY ([SubmissionId]) REFERENCES [Submissions]([Id]),
    CONSTRAINT [PK_TestResults] PRIMARY KEY ([TestCaseId], [SubmissionId])
)

```


Entities
Services
Controllers
ViewModels