

MOOC Documentation

Cryptobionic

November 4, 2013

Cameron Fabbri

Dominic DeMarco

Erik Egge

Ajay Kumar

Contents

1.	Vision Statement.....	4
a.	Revision History.....	4
a.	Vision Statement.....	4
b.	Introduction	4
2.	Stakeholder Description.....	4
a.	Stakeholder Summary.....	4
3.	System Features.....	5
a.	Automatic account creation prevention.....	5
b.	Creation of a class	5
c.	Searching for a class.....	5
d.	Grading System	5
e.	File Upload System.....	5
f.	Registering for a class.....	5
4.	Use Cases.....	6
a.	Creating a class.....	6
1	Use Case Definition	6
1.1	Brief Description.....	6
1.2	Actors	6
1.3	Assumptions.....	6
1.4	Out-of-Scope	6
1.5	Pre-conditions	6
1.6	Post-conditions	6
1.7	Basic Flow.....	6
•	User specifies class details including but not limited to:	6
1.8	Sub Flows	7
1.9	Alternate Flows	7
2	Special Requirements.....	7
2.1	Business Rules	7
2.2	Data Requirements	7
2.3	Performance Requirements.....	7
2.4	Design Constraints	7
2.5	UML Activity Diagram	8
b.	Creating a user	9

c. Logging in	11
d. Updating profile.	13
e. Create Assignment	14
f. Grade Assignment	15
g. Update Grade	15
h. Submit Assignment	15
4. Supplementary Specification	16
a. Revision History	16
b. Introduction	16
c. Security	16
d. System Requirements	16
e. Usability	16
f. Performance	16
5. Domain Model	17
6. System Sequence Diagram	18
7. Database Structure	23
8. Test Cases	27
9. User Manual	29
Introduction:	31
Registering a New User/Logging In:	31
Student Functions:	32
Joining a Class:	32
Drop a Class:	32
Submit an Assignment:	33
Teacher Functions:	33
Creating a Class:	33
Approving Students:	34
Dropping Students:	34
Uploading Assignments:	34
Grading:	35
10. Glossary	36

1. Vision Statement

a. Revision History

The name of the group is Cryptobionic. The name of the product is MOOC, Massive Online Open Courses. It is to be implied that the authors were all of the group members, namely Cameron Fabbri, Dominic DeMarco, Erik Egge, and Ajay Kumar.

Version	Date	Description	Author(s)
Inception Draft	Oct. 26 th 2013	First draft.	
Revised Draft	Oct. 31 st 2013	Revised draft.	
Elaboration Draft	Nov. 4 th 2013	Draft for turn in.	
Final Paper	12/05/2013	Second Iteration	

a. Vision Statement

Our goal is to provide a friendly and easy to use website to provide the users with the opportunity to teach and enroll in various classes, free of charge.

b. Introduction

In our project, we develop a web application that organizes and manages a system of free online classes for anyone to join. Each user signs up as a student, and has the option to be a professor by creating a course. Classes are created, joined, and managed by the users. The professor (and creator) of each class is that class's administrator. File uploads will be allowed for notes and other documents pertaining to the class.

2. Stakeholder Description

a. Stakeholder Summary

Stakeholder	Description
Student	A student maintains their own profile as well as the classes they are enrolled in.
Professor	A professor maintains their own profile as well as the administrative services that go along with the class they teach.
Host	The person(s) responsible for hosting the server that runs the website.
User authentication service	Responsible for verifying the username and password and redirecting to the appropriate pages.

3. System Features

a. Automatic account creation prevention.

The system sends a verification email that enables the account. This prevents false accounts as well as accounts created by bots. It also verifies that the user has a working email for use of sending notifications.

b. Creation of a class

Every user is eligible to create their own class. When a user creates a class, they automatically become the professor of that class. This allows each user to have one account for teaching classes and joining the class, instead of the unnecessary need for multiple accounts.

c. Searching for a class

A user has the option to search the database for classes. The search results are links to the class pages which the user can click on. Upon reaching the class page, the user has the option to register for that class.

d. Grading System

After files are uploaded, the teacher can download them and then put in a grade in the system for it. Each student can view their grades on the grade page, which also shows their overall average based on the weighting for each type of assignment.

e. File Upload System

The file upload system allows professors to upload five types of files:

- Homework
- Quiz
- Test
- Project
- Materials

The homework, quiz, test, and project files can be downloaded by the students, completed, and then submitted by the students for grading. The class materials are in a separate section labeled "Class Materials" for students to download and view.

f. Registering for a class

Every user can register for any class of their choice. To do so, the user goes to that class page by searching for the class or going through the class directory, and clicking the register button. Upon registering, an email is sent to the professor with the user's information. The professor then has the option to do nothing (thereby not admitting the user), or to click the link given in the email to allow the student into the class. The user will receive an email notifying them when and if they have been admitted.

4. Use Cases

a. Creating a class

1 Use Case Definition

1.1 Brief Description

The purpose for this use case is to define the act of creating a class on mooc. Class creation can be from any of the current users.

1.2 Actors

Primary Actor(s):

Professor – The user who is creating the class. The user will be either already signed up as a student and have multiple roles or signed up as a professor, therefore maintaining the same role.

Secondary Actor(s):

Student(s) – The users that will be enrolling in the class.

Teacher's Assistant (TA) – Will have the roles of a normal teacher's assistant. Responsible for grading work and providing feedback to student users.

1.3 Assumptions

1. The class is created in a serious and responsible manner (no fake classes).
2. The class topic is appropriate.
3. Any user signed up as a student can create a class.

1.4 Out-of-Scope

- Checking for authenticity of a class before creation.

1.5 Pre-conditions

- A user is signed up either as a professor, student, or both.
- The intent of creating the class is honest.

1.6 Post-conditions

- Class is created.
- Students can join the class.

1.7 Basic Flow

- User specifies class details including but not limited to:
 - Class name.
 - Class description.

- Maximum size.
 - Class difficulty level.
 - Length of class.
- Class is created with a minimum start date of 1 week after creation.
- Students join the class.

1.8 Sub Flows

- At this stage N/A

1.9 Alternate Flows

- If a user wants to join a class after the deadline for joining, they must contact the professor for approval.

2 Special Requirements

2.1 Business Rules

- At this stage N/A

2.2 Data Requirements

- Class name.
- Class information.
- Class start date.
- Class capacity.
- Class difficulty

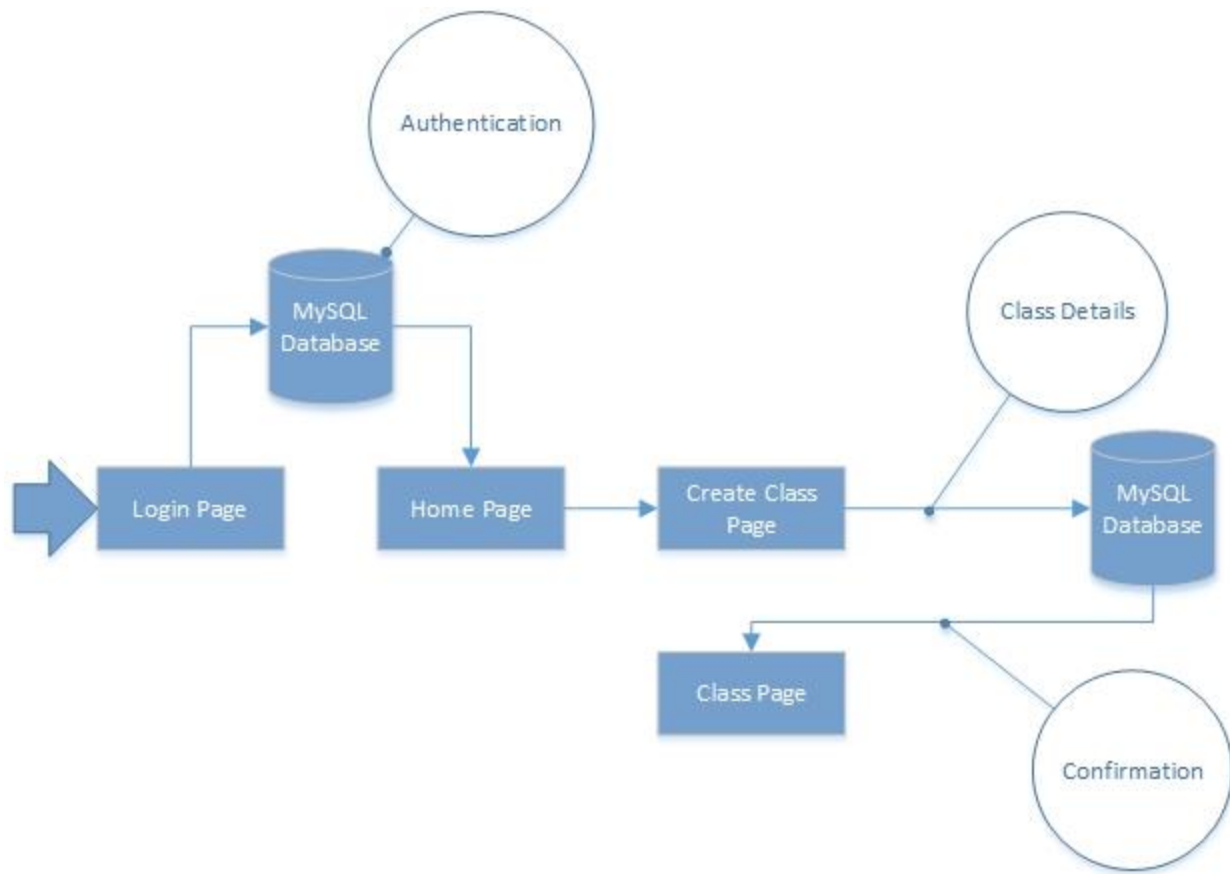
2.3 Performance Requirements

- Needs to be able to handle multiple class creations at one time.
- Needs to create the class in a reasonable amount of time.

2.4 Design Constraints

- Will not be running on a mobile device or platform.

2.5 UML Activity Diagram



The UML Diagram describes the steps a user would take to create a class:

1. Log in as either a student or teacher.
2. Go to the page to create a class.
3. Enter in class details, which are entered into the MySQL Database.
4. Brought to confirmation page (the class page itself).

b. Creating a user

Overview

Actor(s): User

Goal: Create a user in the website database

Trigger(s): Actor visits website and registers

1. Create New User

- User name, password, and email entry forms are displayed
- System checks existing user database for email address.
- If email address exists, go to **User Already Exists**
- User enters information.
- Information is checked for errors (goto **Errors**)
- New user is added to the database.

2. Check User Already Exists

- User is displayed with a password reset page and a confirmation button
- If button is pressed, reset user password in database and send email with new password

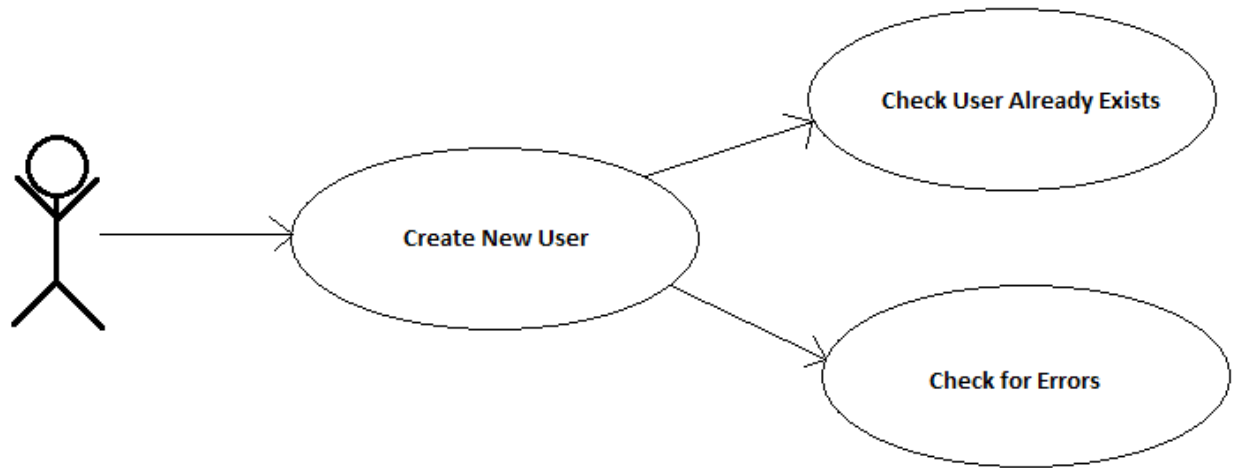
3. Check for Errors

- Check that user name does not already exist in existing users database.
- Check to make sure user name meets all requirements set (length, complexity, etc)
- Check to make sure password meets all requirements set (length, complexity, etc)
- If any errors are found, return to **Create New User** and display appropriate message.

4. Normal Flow:

1. Create new user.
2. Check User Already Exists.
3. Error check all inputs.

UML Diagram



c. Logging in

1. Introduction

In order the students to keep track of their own personal progress and enroll in new classes, and so the teachers using our system can post grades, assign work and upload content; we are making a system where all our different users can log in. In this particular part of our project, we will be using, PHP, JQuery and MySQL.

1.1. Purpose of the system

- We want to provide teachers a means to upload content, make courses, assign work, give tests and give grades to individual students. In order to do so we need to make sure that the person teaching a course has access to all these abilities and make sure only they have the power to access and change their own content.
- We want to provide a way for the students to get individual grades, enroll in courses and participate in the classes under their own alias.

1.2. Scope of the system

- The system will consist of a login screen with two text fields for entry of the username and the password. It will take the input and either give back an error if there is a mismatch or will bring the user to the correct screen if it's correct.

1.3. Objectives and success criteria of the project

- Users can access their accounts through our site

1.4. References

- <http://tutorialzine.com/2009/10/cool-login-system-php-jquery/>

1.5. Overview

- The log in system will allow for users to access their own specific accounts and will test to see if their credentials are correct before bringing them to their specific.

2. Proposed System

2.1. Overview

- The proposed system will give the user to choose between a student or teacher login, then the page will give the user name and password fields that the user must put in their specific data.

2.2. Functional Requirements

- The functional requirements are that the system needs to read the username and password given by the user and let them into the system if the fields match up to the user's settings.

2.3. Nonfunctional Requirements

2.3.1. Usability

- The user should be able to find the login section easily and should be able to use it easily
- The system should be able to bring the user to the correct user page after log in

2.3.3. Performance

- Log in time shouldn't take too long

2.3.4. Supportability

- Possibly a system where a user can request a lost username or password but this is a goal for later

2.3.5. Implementation

- The system will use a combination of PHP, MySQL and JQuery to log users into our software

2.3.6. Interface

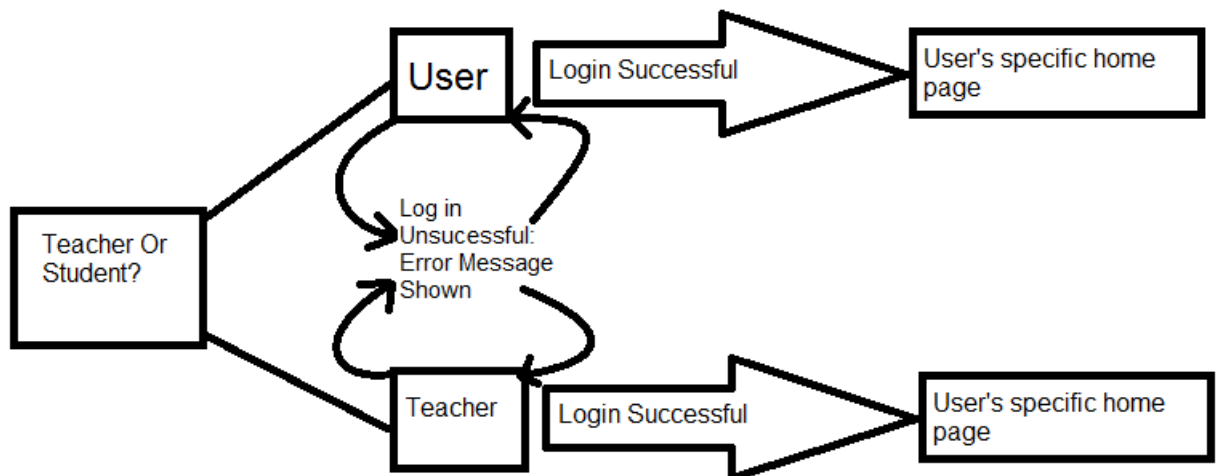
- The log in section should be easy to find and use for the user

2.4. System Models

2.4.1. Scenarios

- If the log in is unsuccessful the user will be prompted with an error on the screen telling them either the user name or password is wrong
- If the log in is successful the user will be taken to their specific home page

2.4.2. Use Case Model



d. Updating profile.

Use Case Name: Update Profile

Scope:

- User Interface and database and interaction between them

Level:

- User Interface and database level

Primary actor:

- User (Teacher / TA/ Student)

Stakeholders and interests:

- User: wants to update his profile. For example, in case of any changes in basic information, enter information is wrong, typos, etc.
- Administrator: wants only eligible users to update profile

Preconditions:

- User has an account in the system

Success guarantees / Post conditions:

- User's account has been updated. The database and server is running normally.

Main success scenario:

- User modifies and submits the profile data
- System Validates data
- System updates user profile/account
- System notifies the user

Extensions (or alternative flows):

- In the event the system could not validate the data, then the user will be notified and instructed

Special requirement:

- Confirmation of updated profile (or reason for failure) to be provided to the user within the time bound

Technology and data variations list:

- Data entered by keyboard

Frequency of occurrence:

- Need not be continuous

e. Create Assignment

Use Case Name: Create Assignment

Scope:

- User Interface and database and interaction between them

Level:

- User-goal level

Primary actor:

- User (Teacher /TA)

Stakeholders and interests:

- User: wants to create assignment
- Administrator: wants only eligible users to create assignment

Preconditions:

- User has an account in the system

Success guarantees / Post conditions:

- Assignment has been created. The database and server is running normally.

Main Success scenario:

- User specifies the following details but are not limited
 - Assignment name
 - Assignment number
- System validates the details.
- System checks whether the assignment already exists
- System creates the assignment
- System notifies the user

Extensions (or alternative flows):

- In the event the system could not validate the details and/or assignment already exists then the user will be notified and instructed

Special Requirements:

- Confirmation of created assignment (or reason for failure) to be provided to the user within the time bound

f. Grade Assignment

Use Case Name: Grade Assignment

Main Success Scenario:

The teacher checks submissions for the assignments and grades them accordingly.

Alternate Scenario:

The teacher finds that the submitted assignment is not in the correct format or is corrupted and emails the student asking them to resubmit it.

g. Update Grade

Use Case Name: Update Grade

Main Success Scenario:

The teacher re-grades the assignment and updates the grade for the student accordingly.

Alternate Scenario:

If there is a typo in the grade then the teacher can update the grade.

h. Submit Assignment

Use Case Name: Submit Assignment

Main Success Scenario:

The student uploads their homework for the correct assignment.

Alternate Scenario:

If the file size exceeds the maximum upload size then the student must inform the teacher.

If the assignment due date has passed then the assignment is rejected.

4. Supplementary Specification

a. Revision History

Version	Date	Description	Author(s)
Inception Draft	Oct. 26 th 2013	First draft.	
Revised Draft	Oct. 31 st 2013	Revised draft.	
Elaboration Draft	Nov. 4 th 2013	Final draft.	
Final Paper	Dec. 5 th 2013	Final Copy	

b. Introduction

This document captures the system requirements that are not stated in the use cases.

c. Security

The server uses permissions specific to each user to limit write access to avoid overwriting of code and malicious file uploads. Each user also has their own database in the MySQL system to avoid overwriting of tables and data. There is a central database and central folder for use of the final versions of code.

d. System Requirements

The basic system requirements of hosting a website of this nature are:

- Apache Web Server
- PHP 5.x.x
- MySQL/MariaDB

e. Usability

Each page will have a consistent feel and look, with a global navigation header. Each form will validate the information being inputted.

f. Performance

Each function will to the best of their ability perform in the most efficient way possible. An example would be a search function searching for a user's phone number in a table:

using:

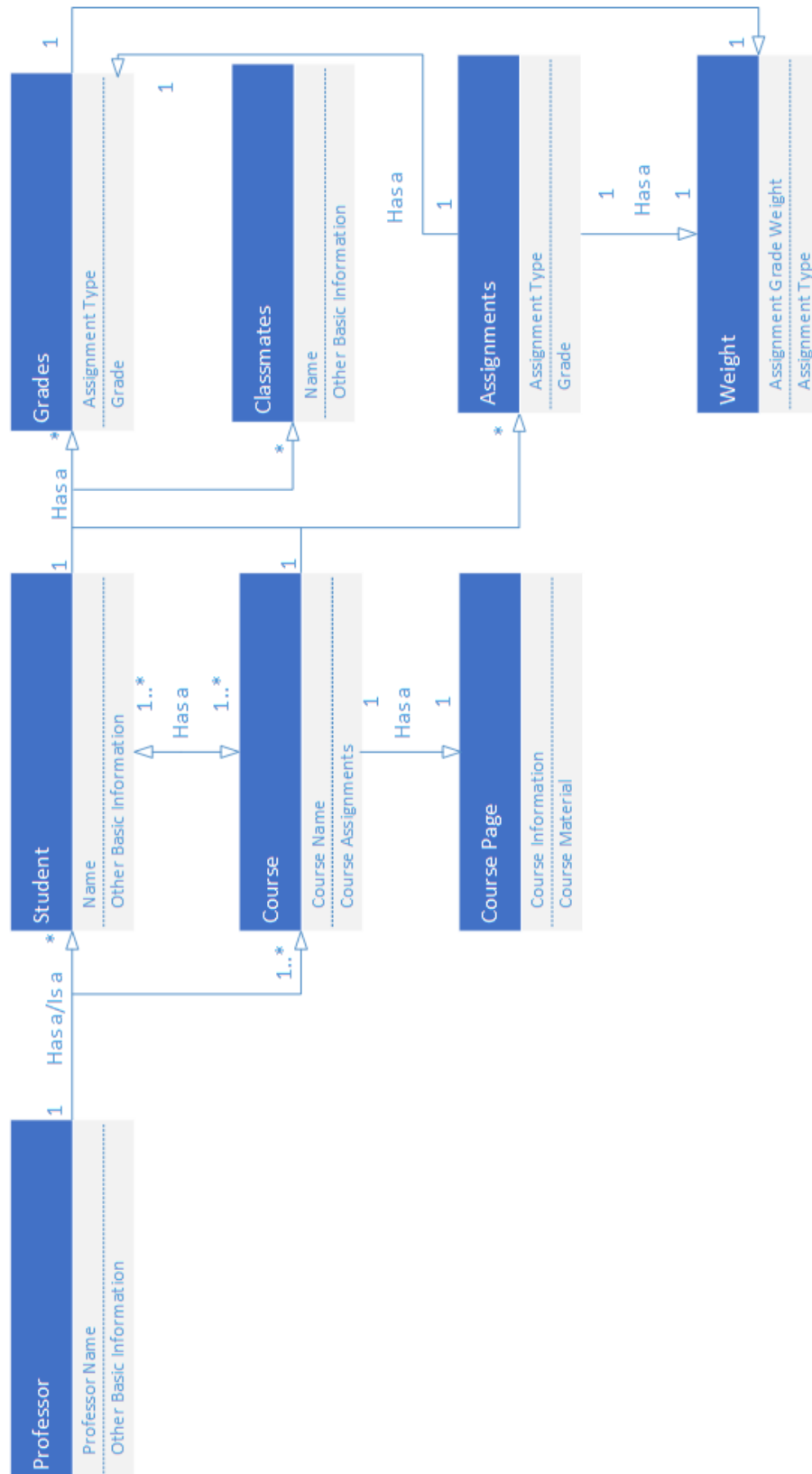
```
SELECT Phone FROM Users WHERE Username='$username';
```

instead of:

```
SELECT * FROM Users WHERE Username='$username';
```

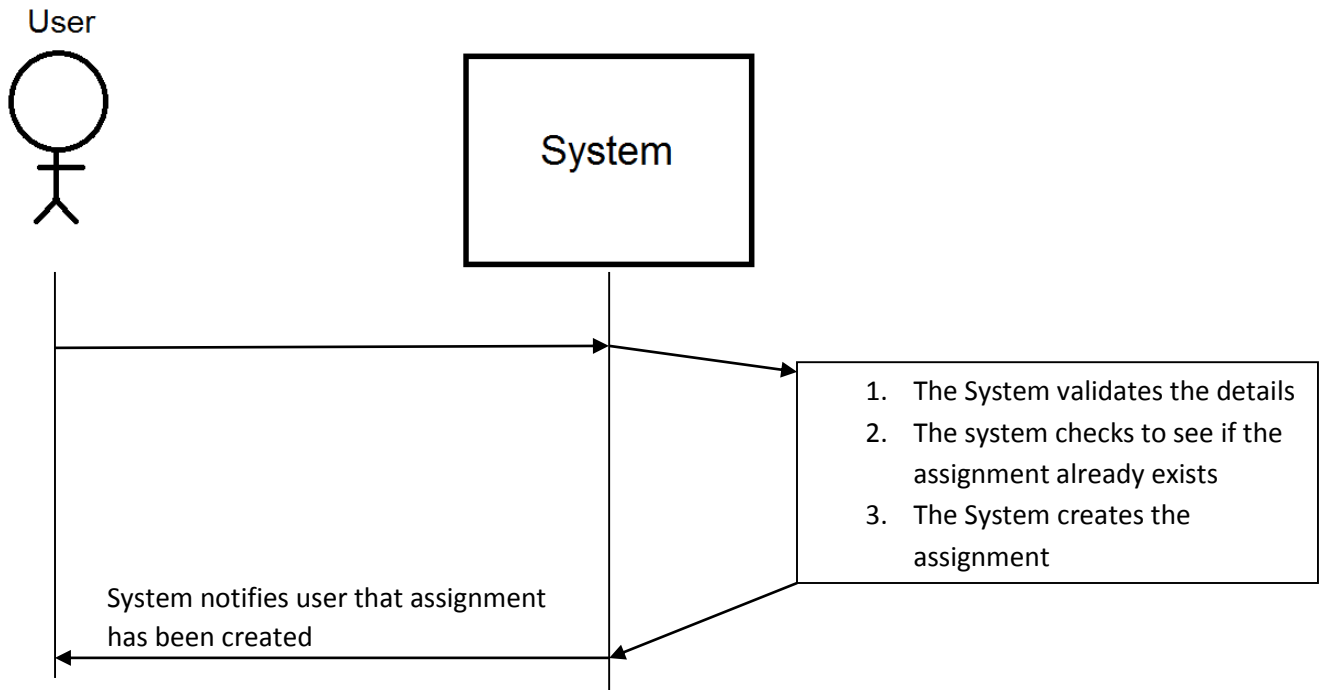
will improve the search time by only searching for the **Phone** field instead of every field. With a large number of entries this will make a considerable difference.

5. Domain Model



6. System Sequence Diagram

System Sequence Diagram: Create Assignment



Operation Contract: Create Assignment

Operation: CreateAssignment(Assignment Name, Assignment Number, Assignment Details)

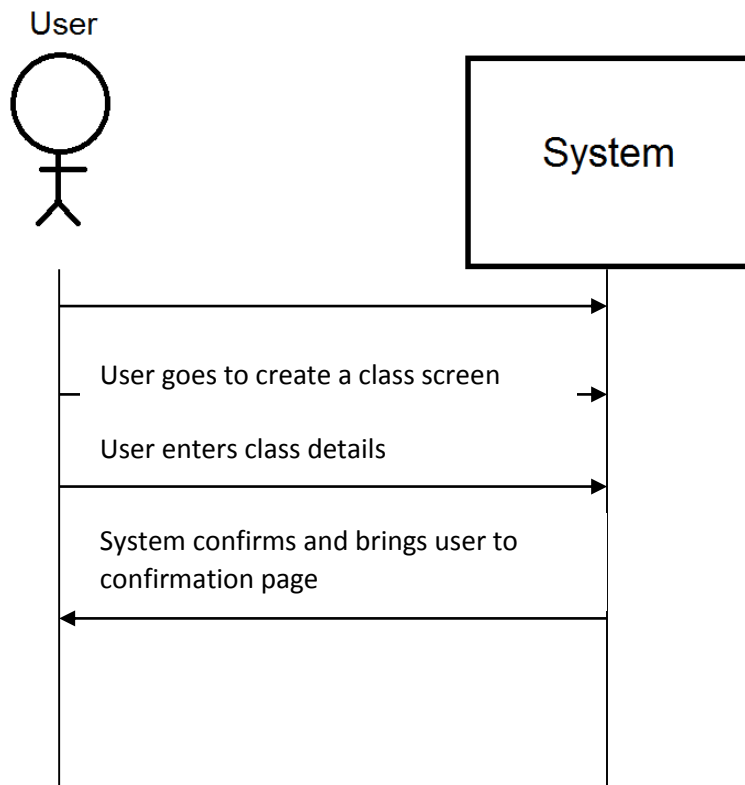
Cross References: UC Create Assignment

Pre Conditions: The user is registered to the system and logged in

Post Conditions:

- The assignment is created
- User is notified that their assignment has been created

System Sequence Diagram: Create a Class



Operation Contract: Create a Class

Operation: CreateAClass(Class Name, Class Details, Class Syllabus)

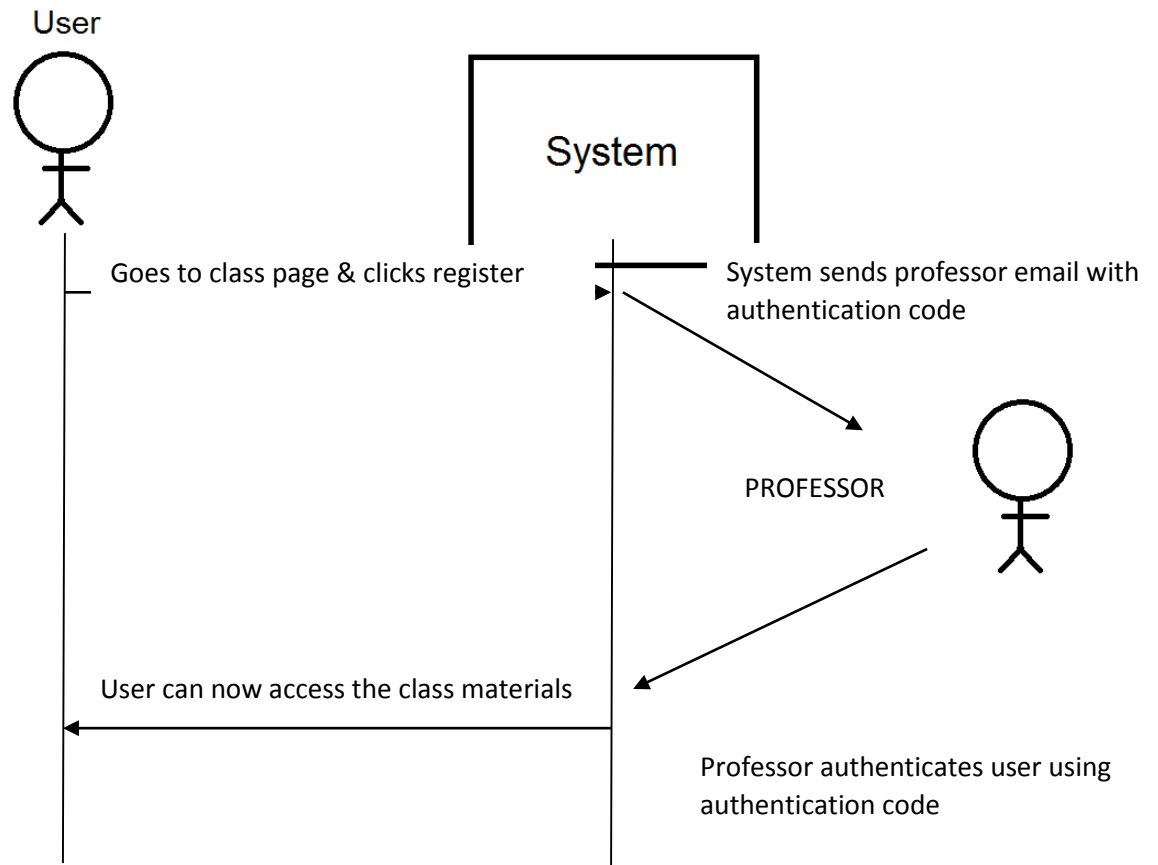
Cross References: UC Create A Class

Pre Conditions: The user is registered to the system

Post Conditions:

- The class is made with the class details
- System confirms the class and makes it available to be seen by all

System Sequence Diagram: Register for Class



Operation Contract: Register for Class

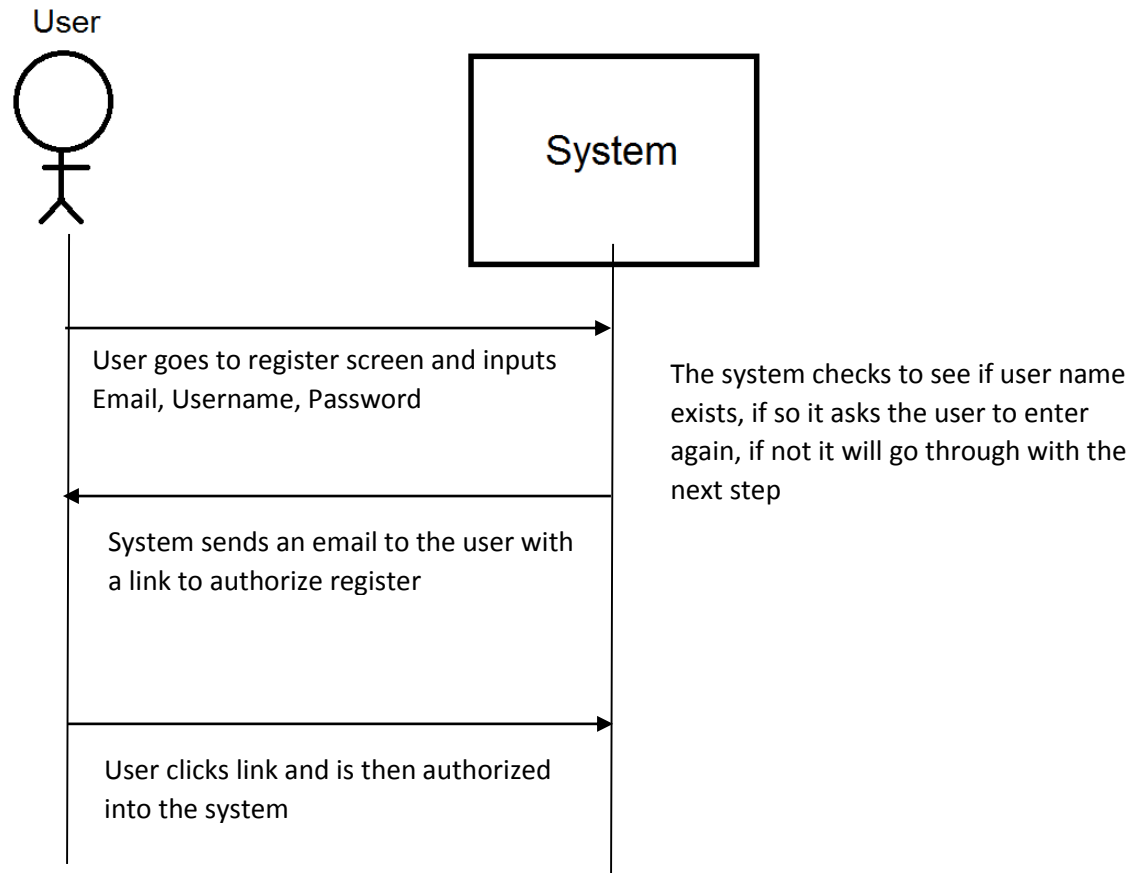
Operation: RegisterforClass()

Cross References: UC for Register for Class

Pre Conditions: The user is registered for the system

Post Conditions: The student is now in the system as registered for the class, this student can now submit assignments, get grades and access course materials.

System Sequence Diagram: Create a User



Operation Contract: Create a User

Operation: CreateAUser(Email,Username>Password,RepeatPassword)

Cross References: UC for Create a User

Pre Conditions: The user goes to the site and wishes to join MOOC

Post Conditions: The user is now part of the system and can start using MOOC

7. Database Structure

```
MariaDB [fabbricm]> show tables;
```

```
+-----+
| Tables_in_fabbricm |
+-----+
| assignments         |
| classes             |
| comments            |
| files               |
| final_grades        |
| grades              |
| junction_student    |
| posts               |
| submissions         |
| temp_class          |
| temp_users          |
| user_info           |
| users               |
| weights             |
+-----+
```

```
MariaDB [fabbricm]> show columns from assignments;
```

```
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| class_id       | varchar(100)  | YES  |     | NULL    |       |
| assignment_name | varchar(100)  | YES  |     | NULL    |       |
| assignment_type | varchar(100)  | YES  |     | NULL    |       |
| grade          | float         | YES  |     | NULL    |       |
| due_date       | date          | YES  |     | NULL    |       |
| file_id        | int(11)       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
```

```
MariaDB [fabbricm]> show columns from classes;
```

```
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| class_id       | int(11)       | NO   | PRI | NULL    | auto_increment |
| class_name     | varchar(100)  | YES  |     | NULL    |       |
| professor      | varchar(100)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
```

MariaDB [fabbricm]> show columns from comments;

Field	Type	Null	Key	Default	Extra
comment_id	int(11)	NO		NULL	
class_id	varchar(100)	YES		NULL	
Username	varchar(100)	YES		NULL	
comment	mediumtext	YES		NULL	
comment_date	date	YES		NULL	
comment_time	varchar(20)	YES		NULL	

MariaDB [fabbricm]> show columns from files;

Field	Type	Null	Key	Default	Extra
file_id	int(11)	NO	PRI	NULL	auto_increment
class_id	int(11)	YES		NULL	
filename	varchar(100)	YES		NULL	
filetype	varchar(100)	YES		NULL	

MariaDB [fabbricm]> show columns from final_grades;

Field	Type	Null	Key	Default	Extra
username	varchar(100)	YES		NULL	
class_id	varchar(100)	YES		NULL	
grade	varchar(100)	YES		NULL	

MariaDB [fabbricm]> show columns from grades;

Field	Type	Null	Key	Default	Extra
class_id	varchar(20)	YES		NULL	
file_id	int(11)	YES		NULL	
grade	int(11)	YES		NULL	

MariaDB [fabbricm]> show columns from junction_student;

Field	Type	Null	Key	Default	Extra
Student	varchar(100)	YES		NULL	
class_id	varchar(10)	YES		NULL	

MariaDB [fabbricm]> show columns from posts;

Field	Type	Null	Key	Default	Extra
post_id	int(11)	NO	PRI	NULL	auto_increment
class_id	varchar(100)	YES		NULL	
Username	varchar(100)	YES		NULL	
post	mediumtext	YES		NULL	
post_date	date	YES		NULL	
post_time	varchar(20)	YES		NULL	

MariaDB [fabbricm]> show columns from submissions;

Field	Type	Null	Key	Default	Extra
class_id	varchar(20)	YES		NULL	
assignment_id	int(11)	YES		NULL	
file_id	int(11)	YES		NULL	
username	varchar(100)	YES		NULL	
filename	varchar(100)	YES		NULL	
submit_date	date	YES		NULL	

MariaDB [fabbricm]> show columns from temp_class;

Field	Type	Null	Key	Default	Extra
class_id	varchar(100)	YES		NULL	
username	varchar(100)	YES		NULL	
professor	varchar(100)	YES		NULL	
rand_key	varchar(4)	YES		NULL	

MariaDB [fabbricm]> show columns from temp_users;

Field	Type	Null	Key	Default	Extra
Email	varchar(100)	YES		NULL	
Username	varchar(100)	YES		NULL	
Password	varchar(100)	YES		NULL	
auth_key	varchar(100)	YES		NULL	

MariaDB [fabbricm]> show columns from user_info;

Field	Type	Null	Key	Default	Extra
FirstName	varchar(100)	YES		NULL	
LastName	varchar(100)	YES		NULL	
Username	varchar(100)	YES		NULL	
Email	varchar(100)	YES		NULL	
Phone	varchar(15)	YES		NULL	
Age	varchar(3)	YES		NULL	
Type	varchar(9)	YES		NULL	

MariaDB [fabbricm]> show columns from users;

Field	Type	Null	Key	Default	Extra
Username	varchar(100)	YES		NULL	
Password	varchar(100)	YES		NULL	

MariaDB [fabbricm]> show columns from weights;

Field	Type	Null	Key	Default	Extra
class_id	varchar(10)	YES		NULL	
Test	float	YES		NULL	
Quiz	float	YES		NULL	
Homework	float	YES		NULL	
Project	float	YES		NULL	
Total	float	YES		NULL	

8. Test Cases

Black Box Test Cases

Editing Grade Test Case

Step	Action	Expected Result	Actual Result	Status
1	Edit Grade Button is pressed	Text box awaiting grade appears	Text box appears	Pass
2	Enter grade into text box	Grade should appear correctly inside of the text box	Grade displays in text box	Pass
3	Update Grades button pressed	Updated grade is displayed	Correct grade is displayed	Pass

Add a File Test Case

Step	Action	Expected Result	Actual Result	Status
1	Browse button is clicked	File browser to select file should appear	File browser appears	Pass
2	File is selected and confirmed	File appears in text box	File appears in text box	Pass
3	Due date is entered	Calendar drops down and allows date to be selected	Calendar drops down on the chrome browser	Pass?
4	File type is selected from a drop down menu	The file type is displayed correctly in the selection field	File type is displayed correctly	Pass
5	Submit button is pressed	File is displayed in file table with the correct supplementary information	File and information are both displayed correctly	Pass

Delete File Text Case

Step	Action	Expected Result	Actual Result	Status
1	Delete button is pressed	File table refreshes showing file has been deleted	File table refreshes and file is no longer apparent	Pass
2	Delete file from sql tables	File entries and traces within the database are deleted	Files and traces of said file no longer exist in the database	Pass

White Box Test Cases

Drop Class Test Case

Step	Action	Expected Result	Actual Result	Status
1	Drop class button is pressed	Delete from table query is run with student ID	Delete from table query is run	Pass
2	Drop action is run	Student's entry removed from sql table	Student entry no longer exists in classes table	Pass

Submit Assignment Test Case

Step	Action	Expected Result	Actual Result	Status
1	Submit button is pressed	User is brought to an upload file page	User is brought to an upload file page	Pass
2	User selects a file	File is selected and displayed correctly	File is selected and displayed correctly	Pass
3	Submit button is pressed	Mysql queries are created ready to be executed	Mysql queries are generated	Pass
4	Insert into table files	An entry in the table files should be created with it's own unique fileid	Entry is successfully created	Pass
5	File is uploaded	File should end up in “uploads folder”	File is successfully uploaded	Pass
6	Insert into submissions	A table entry with a corresponding file ID is created associating with the entry in the files table	Entry is successfully created	Pass
7	Page redirection	User is brought to new files page	User is redirected successfully	Pass

SOFTWARE DESIGN AND DEVELOPMENT

User Manual

Mooc

Team Cryptobionic

12/5/2013

Contents

Introduction:	30
Registering a New User/Logging In:	31
Student Functions:	32
Joining a Class:	32
Drop a Class:.....	32
Submit an Assignment:	33
Teacher Functions:	33
Creating a Class:	33
Approving Students:.....	34
Dropping Students:	34
Uploading Assignments:.....	34
Grading:.....	35

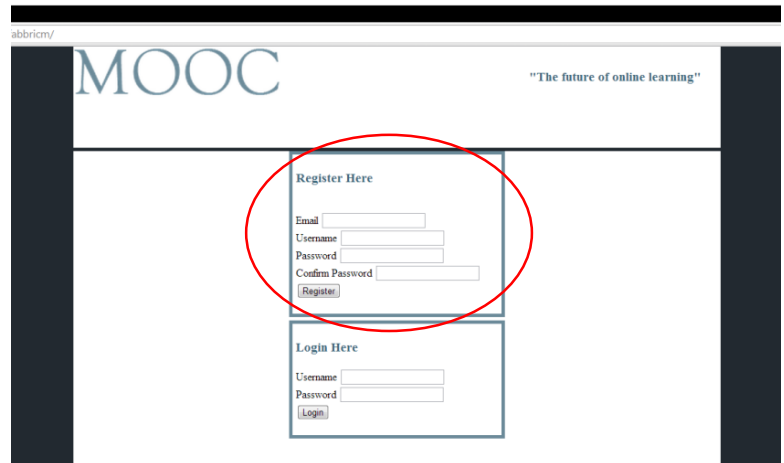
Introduction:

Welcome to MOOC! (Massive Online Open Courses!). Soon enough you will be learning anything you could possibly desire to know and/or teaching people the great knowledge that you possess! But to first get started using MOOC you must make a new user, and whether you plan to teach/learn/or both, it is all the same starting out!

Registering a New User/Logging In:

To register a new user, start by accessing the site. The first page that pops up is the log-in/register screen. To register a new user, type the following user information into the “Register Here” box.

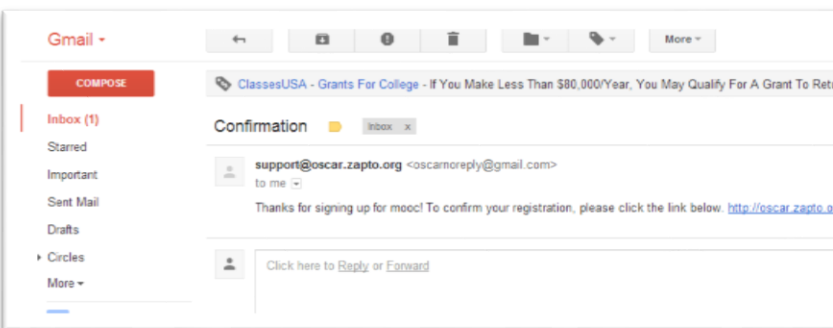
- Your Email (This will be the email you receive confirmation emails to)
- Your desired username
- Your desired password
- Repeat that desired password



The screenshot shows the MOOC website's registration and login interface. The 'Register Here' section is highlighted with a red circle. It contains four input fields: Email, Username, Password, and Confirm Password, followed by a 'Register' button. Below this is the 'Login Here' section with fields for Username and Password, and a 'Login' button. The website header includes the MOOC logo and the tagline 'The future of online learning'.

And press register to finish registering for the site. Now the site will send you a confirmation email to the account that you entered. So go check that email address for the link.

(Note: That this email may be in your junk mail)



depending on your settings)

Once you have found the email, click the link the email has sent you and it will bring you to a page where you can input some more information about yourself before heading to the rest of the site. Once you are done with that press “Update” and it will bring you to the login screen again. This time, enter your credentials into the “Loggin Here” box and then press login. And by doing this, you will be brought to your homepage!

Student Functions:

Joining a Class:

You are looking to join a class. You can either use the search bar located on the middle of the home screen or on the top navigation bar you can click on classes.

- Search Bar: By entering any input into the search bar, it will find the classes that contain that input. For example: If you type in “a” it will bring up all classes that start with “a”.
- Top Navigation Bar: By clicking the “Classes” tab on the top Navigation bar it will bring you to your classes’ page. This page shows you all the classes that you are in, and all the classes you

teach. It also gives you another search bar to search for new classes with that works the same as the one on the home page. It gives you a link to create your own class which we will go over in

the section: Creating a Class. Last but not least it also gives us a link to view all available classes. This is good if you are new to MOOC and want to see all the courses we offer, or if you are stuck on what you would like to learn about next and need some inspiration!



To register for the class, click on the link for the class to go to that class’s specific page. From there you want to click “Register”. (Note: If you are already registered for the course “Register” doesn’t show up.)

By clicking “Register” this will send a confirmation email to the professor, allowing them to either accept you into the course or not allow you into the course for now if they don’t feel you have the right credentials. You will receive a confirmation email if they accept you into the course. Once you are in the course it will show up under “Classes you are in” on your home page! Congrats! You joined your first course!

Drop a Class:

To drop a class, go to your homepage, under classes you are in, there is a link next to the class you wish to drop “Drop Class” by clicking this it will drop you from that course.

Submit an Assignment:

To submit an assignment for a course go to the class home page and then click assignments. From there it will give a table with all the assignments for that course, the assignment might include a description of the assignment or maybe even a file you need to work on or see to do the assignment.

Name	Type	Due date	Submit
	Test	2013-12-12	Submit

Once you have completed the assignment, click on the submit button for that assignment. It will then bring you to a page where you can choose the file you want to upload. Once you have chosen that file it will bring you back to the page where you can then press "Submit".

Teacher Functions:

Creating a Class:

Congrats! You are a professor to be! And you are ready to share your knowledge of the world! You can create your class by going to "Classes" link in the header. In the Classes page there is a link to "Create your own class", click on that link. In the window that is returned, there are

Class Name

Here you specify the weights of how your class will be graded.
Each field can have any value (including 0) but the total must add up to 100.

Tests %

Quizzes %

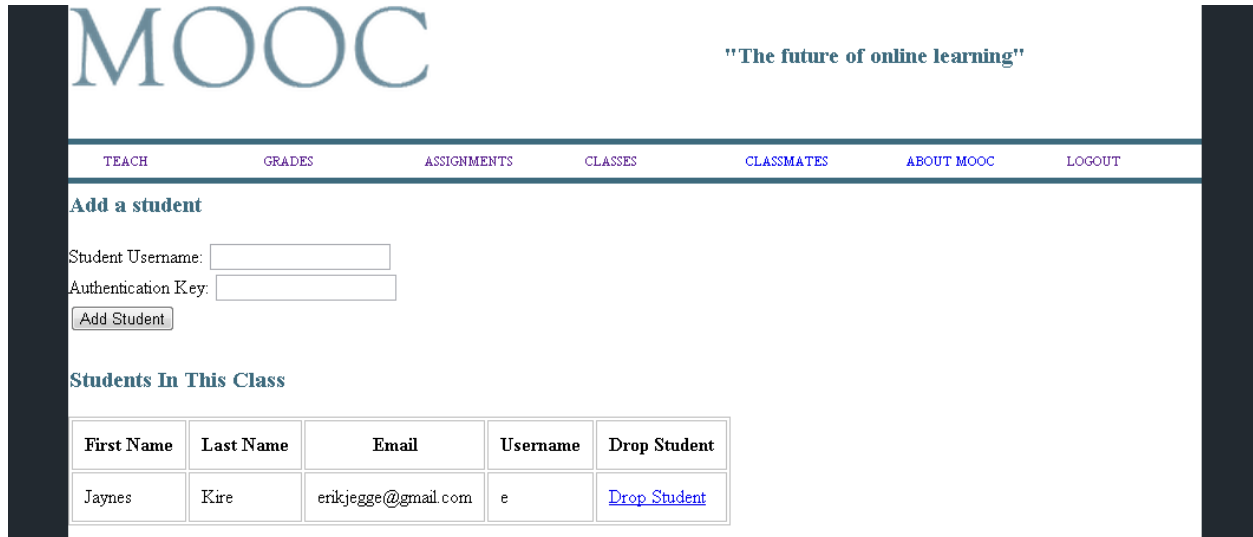
Homework %

Projects %

options to name your course, and specifying the weight of the different categories of assignments. When you are done click "Create Class" and done! You have created your new class!

Approving Students:

Now that you have a class, you now need to accept students who apply to learn your knowledge! To do this, once students sign up for your course you will get an email with an authentication code and a link. Click on the link in the email and it will bring you to the correct page. Type in the student's username and the authentication code into the forms and then click



The screenshot shows the MOOC interface. At the top, the MOOC logo is on the left and the tagline "The future of online learning" is on the right. Below the header is a navigation bar with links: TEACH, GRADES, ASSIGNMENTS, CLASSES, CLASSMATES, ABOUT MOOC, and LOGOUT. The main content area is titled "Add a student". It contains two input fields: "Student Username:" and "Authentication Key:". Below these fields is a button labeled "Add Student". Below the form is a section titled "Students In This Class" which contains a table with the following data:

First Name	Last Name	Email	Username	Drop Student
Jaynes	Kire	erikjegge@gmail.com	e	Drop Student

"Add Student". Once that is done the screen will refresh and that student should be added to the students table on that page.

Dropping Students:

On the "Manage Students" page, in the students table there is an option next to each student that says "Drop Student" to drop that student just click the link and he/she will disappear from the table.

Uploading Assignments:

To upload an assignment to your class to give to your students go to the class page and click "Files". It will give you options to upload a file, give that assignment a name and give a due date and put it in a specific category for weighting of the assignment in the student's final grade. By clicking "Submit" the assignment will be posted, it will show you up in the assignments table for your class.

MOOC

"The future of online learning"

JAYNES
GRADES
ASSIGNMENTS
CLASSES
CLASSMATES
ABOUT MOOC
LOGOUT

Id	file	type	Delete
----	------	------	--------

Filename: AddClass.PNG
 File type: Assignment Name
 Due Date

Grading:

To grade assignments go to the class page and click on “Submissions”. It will bring you to the page below. Under each submitted assignment in the table there will be an “Edit Grade” button. Click on that link and then input the grade into the form given and then click, “Update Grades”. The grades will be updated in the table.

MOOC

"The future of online learning"

TEACH
GRADES
ASSIGNMENTS
CLASSES
CLASSMATES
ABOUT MOOC
LOGOUT

Student	FileID	Assignment	File Name	Date Submitted	Grade
e	13		AddAssignment.PNG	2013-12-04	<input type="text" value="90"/>

10. Glossary

MySQL/MariaDB: Database system used for managing large quantities of data

Header: Common navigation pane displayed at the top of all subsequent pages of the website.

PHP: Programming language used to communicate between the website and the MySQL database.

Apache: Web server used to serve http pages in a Linux environment.