COMP 4770: Team Project
Iteration 0: Requirements Gathering

Team "C"omputer
Science
Students
Taking
Computer
Science
4770:
Team
Project
As
Part of
Their
2016
Winter
Semester
At
Memorial
University of
Newfounland,
Winter 2016.

<u>**aka Team CSSTCS4770TPAPT2016WSAMUN Winter2016:**</u>

Tim Murray
Shane Butt
Dean Massecar
Philip St.Croix

Iteration 0:

Outlined below is the first iteration of the CodeDrop (name pending) project from our team. This iteration comprises of a selection of User Stories and Use Cases, some wireframe sketches which showcase a potential UI for the system, and a UML State Chart describing the logical flow of the program.

Before continuing here is a quick run down of the technologies that we have put into consideration for use with CodeDrop (all pending approval):

The project will be coded primarily -- if not fully -- in Java on the server side, with Javascript/HTML on the front end.

Tomcat will be used as our application server.

Spring is in consideration as a Java web framework.

PostgreSQL will be used for our database needs.

CodeMirror has been suggested, for use as the in-browser text editor that the Users would interact with.

Docker provides functionality to sandbox our compilation and testing environments as we will be running arbitrary code on the server side.

Wireframes

Below are wireframe diagrams of the individual screens that users will encounter in the system.

Universal Login page (left) [universal-login]

This is the page encountered by all users when they first enter the system.

Student Main Page: (right) [student-assignments]

This is the first page the Student sees when they log in, which allows them to select assignments to work on from a list of assignments and courses.

Student Coding Page: (left) [student-coding]

This is the page on which the Student writes/uploads code. It has a coding window as well as a File browser and console window.

Student Testing Page: (right) [student-testing]

This page allows a student to run the tests for their assignments after they have written code.

Instructor/Marker Course Management (left) [instructor-management, marker-management]

Allows Instructors or Markers to view and navigate through all of their courses/ assignments and view all students assigned. Instructors can *add* assignments.

Instructor/Marker Student Management: (right) [instructor-studentmanagement]

Allows the Instructor or Marker to view assignment submissions for the Students registered in a particular course. Opening an assignment here opens up the Instructor Assignment View page.

```
┌─────────────────────────────┐   ┌─────────────────────────────┐
│ HOME               Sign Out │   │ HOME               Sign Out │
│                             │   │                             │
│ (+) Course 1  - (Open Course)│  │ COURSE 1                    │
│      - Assignment 1 (x)     │   │ (+) Student 1 (STUDENT NUM) │
│      - Assignment 2 (x)     │   │      - Assignment 1 (grade) │
│      - Assignment 3 (x)     │   │      - Assignment 2 (grade) │
│      - [+] Add Assignment   │   │      - Assignment 3 (NC)    │
│ (+) Course 2                │   │ (+) Student 2 (NUM)         │
│ (+) Course 3                │   │ (+) Student 3 (NUM)         │
│                             │   │                             │
└─────────────────────────────┘   └─────────────────────────────┘
```

Instructor Assignment Edit Page: (right) [instructor-assignmentedit]

Where the instructor edits assignments. This allows the user to add in multiple tests, as well as descriptions, and start/end dates.

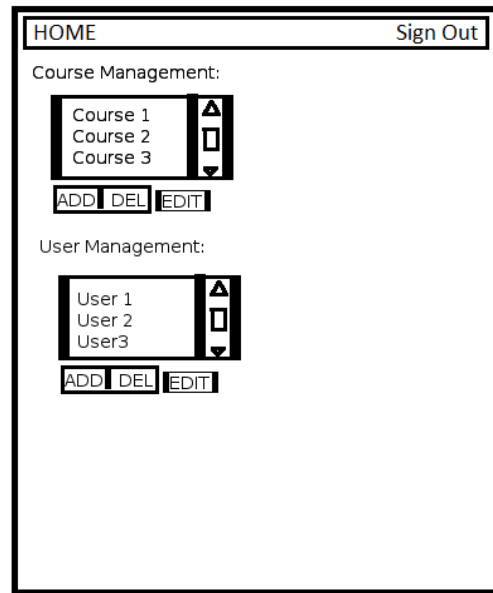**Assignment Viewing Page:** (right) [instructor-assignmentview]

This page shows the instructor/marker details about an individual assignment submission from a student, and allows them to run tests on it/ leave comments and grade.

```
┌─────────────────────────────┐   ┌─────────────────────────────┐
│ HOME               Sign Out │   │ HOME               Sign Out │
│ Assignment 1 -- Edit        │   │                             │
│ Description: ┌─────────────┐│   │ ┌─────────┐ ┌─────────────┐ │
│             │             ││   │ │File     │ │Code +       │ │
│             │             ││   │ │browser  │ │comments     │ │
│             └─────────────┘│   │ │         │ │             │ │
│ Test 1:                     │   │ └─────────┘ │             │ │
│    Input:     ┌──────────┐ │   │ ┌─────────┐ │             │ │
│    Output:    │          │ │   │ │Command  │ │             │ │
│    Test Code (opt):│     │ │   │ │buttons  │ │             │ │
│    Test Runs: │          │ │   │ └─────────┘ └─────────────┘ │
│ (+) Test 2    └──────────┘ │   │ ┌─────────────────────────┐ │
│ Add a test. [+]            │   │ │Marking criteria and     │ │
│                            │   │ │grading interface        │ │
│ Start Date: ┌────────────┐ │   │ └─────────────────────────┘ │
│ Start Time: │            │ │   │ ┌─────────────────────────┐ │
│ End Date:   │            │ │   │ │Console interface        │ │
│ End Time:   └────────────┘ │   │ │                         │ │
│                            │   │ │                         │ │
│ [SAVE]      [ CANCEL ]     │   │ └─────────────────────────┘ │
└─────────────────────────────┘   └─────────────────────────────┘
```

Admin Management Page:

[administrator-management]

This page shows the Administrative user a list of courses and of Users in the system. It provides links for addition, deletion and editing of these entities.

```
HOME                          Sign Out
Course Management:

  Course 1    △
  Course 2    □
  Course 3    ▽

ADD  DEL  EDIT

User Management:

  User 1      △
  User 2      □
  User3       ▽

ADD  DEL  EDIT
```
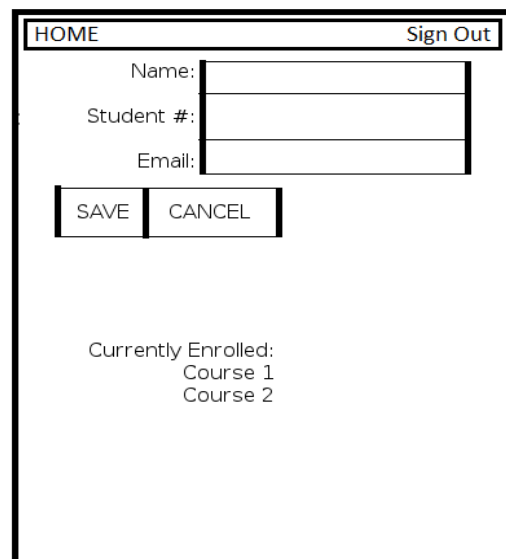
Admin User Management Screen:

[administrator-usermanagement]

This page lets an Admin user change details about system users, such as their names, emails, and permissions.

```
HOME                          Sign Out
       Name: [            ]
   Student #: [            ]
      Email: [            ]

   SAVE    CANCEL




       Currently Enrolled:
            Course 1
            Course 2
```
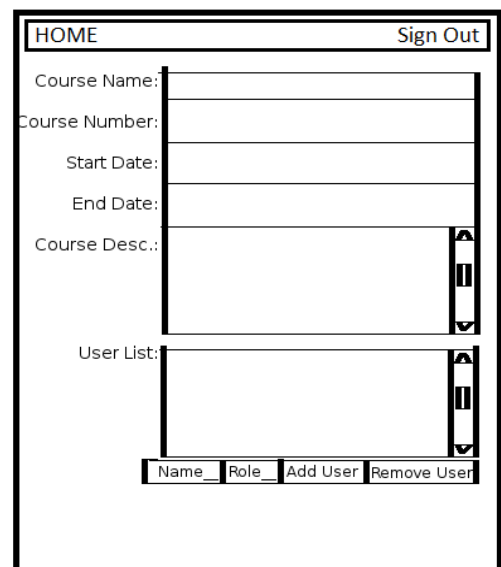
Admin Course Management Screen:

[administator-coursemanagement]

This page allows Administrators to set the details of courses, such as name, number, and also to add individual students, and instructors to the course being edited..

```
HOME                          Sign Out
 Course Name: [            ]
Course Number: [            ]
  Start Date: [            ]
   End Date: [            ]
 Course Desc.: [          ] △
              [          ] □
              [          ] ▽

   User List: [          ] △
              [          ] □
              [          ] ▽

         Name_  Role_  Add User  Remove User
```
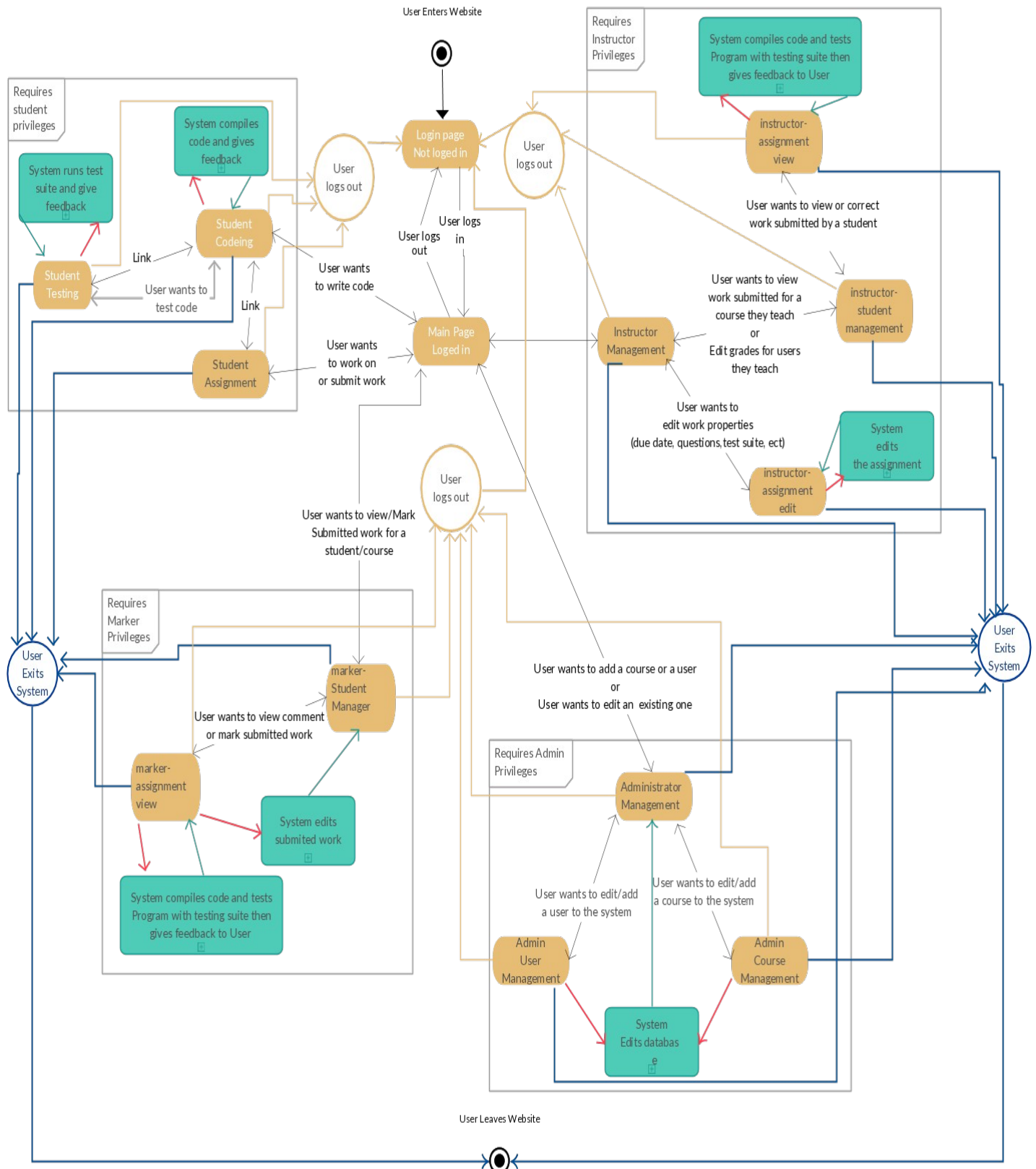
State Chart: This chart provides a view of the states of the *CodeDrop* system, and the actions which Users can take to transition between them.

User Stories:

In our vision of the system, there is 4 unique users, *Students, Instructors, Markers* and *Administrators*. They each have various user stories which are outlined below:

Instructor:

[I-01] As an Instructor I want to add assignments to a course, so that my Students can view it.

-Assignments have due dates for when they can have solutions upload.

-[Query: should students be informed when assignment is uploaded?]

[I-02] As an Instructor I want to edit assignments.

[I-03] As an instructor, I want to group message students, so that I can provide new information/assignments.

  -[Query: Will groups be contacted by external email or an internal chat/messaging system?]

[I-04] As an instructor I want to message markers, so that I can provide new information/assignments.

[Q: external email or internal system?]

[I-05] As an instructor, I want to set tests, so that code can be automatically graded.

       [Q: What sort of tests? Basic expected string output comparison? Other?]

[I-06] As an instructor, I want to leave feedback, so I can comment on how students can improve submitted work.

       [Q: Should this be inserted directly into the file? Or linked in a separate file by 'line number'?]

[I-07] As an Instructor, I want to set due dates, so that assignment submission windows close at a certain time.

  -a non-writeable version should become available for reference

[I-08] As an Instructor I want to be able to view the courses I am assigned to.

[I-09] As an Instructor, I want to be able to set grading criteria.

[I-10] As an Instructor I want to return assignments to Students with comments and marks, so that the

Student can have feedback.

[I-11] As an Instructor I want to be able to change due dates for upcoming work.

Marker:

[M-01] As a Marker I want to be able to view the courses I am assigned to.

[M-02] As a marker, I want to be able to set grades on assignments.
Query: [how are students notified about updated marks on their submissions?]

[M-03] As a Marker, I want to view submissions, so that I may mark them.

Student:

[S-01] As a student, I want to upload files, so that I can code on my own computer.

[S-02] As a student, I want to download files, so that I can retrieve them from the server.
- as well as archives (JAR files) for Java code so that they can run it on their own computer, and the compiled code. [Q: how should this be done for C++? so many different environments need to be compiled for in order to make a proper executable.]

[S-03] As a student, I want to code in the browser, so that I am not constrained to any specific computer.
  -Java and C++ languages should be supported, with extensibility for more (eg Python).

[S-04] As a student, I want to be able to receive feedback, so that I can see how my work can improve.

[S-05] As a student, I want to automatically run tests on compiled code.

[S-06] As a student, I want to see any compile time errors.
  • With syntax highlighting for errors.

[S-07] As a student, I want to interact with my code via console, so that I can have a similar experience

to what I would have on my own computer.

[S-08] As a Student I want to be able to view the courses I am assigned to.

[S-09] As a student I want to compile code to see if it works.

[S-10] As a student I want to view my Assignments for courses I am assigned to.

Administrator:

[A-01] As an Administrator I want to be able to assign Instructors to courses.

[A-02] As an Administrator I want to be able to assign Students to courses.

[A-03] As an Administrator I want to be able to assign Markers to courses.

[A-04] As an Administrator, I want to deploy new language compilation and test suites.

[A-05] As an Administrator, I want to be able to create courses
  -Courses have an instructor , markers and a classlist.

[A-06] As an Administrator I need to create accounts for users
  -There are four types of users admin, instructor, marker and student.

Use Cases

Name: Instructor grades a Submitted Assignment

User stories addressed: [I-06, I-10, M-02, M-03]

Precondition: logged in

Main Scenario:

1. The User clicks on a student's name
2. The User clicks on the newest submitted assignment for the selected student
3. *The system loads an assignment-grading/viewing page containing the expected results vs the given results from the assignment*
4. The user optionally adds comments next to each result
5. The user clicks "save"
6. *The system creates an edit and applies it*
7. *The system reloads the assignment-view page*
8. The user submits the graded assignment

Alternate Scenarios:

5.1. The user clicks on "cancel"

      6.1. The system displays a confirmation dialog: "

Are you sure you wish to cancel?"

      7.1. If the user clicks no, back to step 4, otherwise on to 8.1

8.1. *The system reloads the assignment-view page*

Postcondition: logged in, navigated to instructor-studentmanagement page

Name: Instructor add/edit a new assignment

User stories addressed: [I-01, I-02, I-05, I-07, I-08, M-02, M-03]

Precondition: logged in, navigated to instructor-management page

Main Path:

1. User clicks expand for a course

2. User clicks add assignment

3. *The system loads the instructor-assignmentedit page*

4. User edits assignment value and due date

5. User adds a test with appropriate description, expected results, individual value and number of allowed runs.

6. User keeps adding tests as described in 4. until the desired amount is reached.

7. User clicks "Save"

8. *System saves the assignment and updates the assignment page for each student in the course.*

9. *The system reloads the instructor-assignmentedit page*

Alternate Paths:

6.1. User clicks "cancel"

7.1. *The system displays a confirmation dialog: "*

Are you sure you wish to cancel?"

8.1. If the user clicks no, back to step 5, otherwise on to 9.1

9.1. *The system reloads the instructor-management page*

Postcondition: logged in, navigated to instructor-assignmentedit page

Name: Instructor messages another user

User stories addressed: [I-03, I-04]

Precondition: logged in, navigated to instructor-studentmanagement page

Main Path:

1. User clicks on a "send message" icon
2. *System loads a send-message page*
3. User selects other users desired to receive message from a drop down menu
4. User edits the message space
5. User clicks "send"
6. *The system sends the message to all recipients*

Alternate Paths:

5.1. The user clicks "cancel"

6.1. T*he system displays a confirmation dialog: "*

Are you sure you wish to cancel?"

7.1. If the user clicks no, back to step 4, otherwise on to 8.1

8.1. *The system reloads the start page.*

3.2. The user does not select any recipients

6.2. *The system displays a message saying no recipient has been selected, the message does not send and page reloads.*

Postcondition: logged in, navigated to instructor-studentmanagement page

Name: Instructor makes an edit to an assignment

User stories addressed: [I-02, I-11, M-02, M-03]

Precondition: logged in, navigated to instructor-assignmentedit page

Main Path:

1. User clicks "edit" for an assignment

2. User edits some or all of the assignment details

3. User clicks "save"

4. *System reloads instuctor-assignmentedit page with updated details*

Alternate Paths:

3.1. User clicks "cancel"

4.1. *The system displays a confirmation dialog: "*

Are you sure you wish to cancel?"

5.1. If the user clicks no, back to step 2, otherwise on to 6.1

6.1. *The system reloads the instructor-assignmentedit page without change*

Postcondition: logged in, navigated to the instructor-assignmentedit page

Name: Student compiles and tests submitted code

User stories addressed:[S01,S03,S05,S06,S09]

Precondition: logged in, navigated to student-coding

Main Path:

      1. User enters code

      2. User selects language to compile in

      3. *System compiles code with respective compiler*

      4. User gets feedback

      5. User runs tests

      6. User saves code

Alternate Paths:

1.1. User uploads code

1.2. User enters code into browser

4.1. Compile errors go to step 1

4.2. Successful compilation go to step 4

5.1. No runtime errors expected results go to step 6

5.2. Runtime error goto step 1

5.3. unexpected results goto step 1

6.1. To cloud

6.2. To computer

Postcondition: logged in, navigated to student-coding

Name: Student submits code.

User Stories addressed: [S-01, S-03, S-07]

Precondition: Student is on student-assignments page

Main Path:

1. Student selects assignment to work on.
2. Student enters code into the in browser editor.
3. Student hits "Save" to save his code.
4. Student chooses to save his code to the system.
5. *System saves the entered code and returns a message verifying that all has gone well.*

Alternate Paths:

2.1. Student selects "Upload" and then chooses a file in his local computer to use for the assignment.

3.1. Student uses the console provided to test the code themself before submitting

4.1. Student chooses to save his code to his local machine.

Postcondition: Student is student-coding.

Name: Student checks feedback on assignment

User Stories addressed: [S-04]

Precondition: logged in, on student-assignments page

Main Path:

1. Student clicks on the course expand
2. Student clicks on the assignment expand
3. Student clicks on "Grades"
4. *The system loads feedback for each test*

Postcondition: logged in, on student-assignment page

Name: User selects an assignment for a course.

User Stories addressed: [S08, S10, M-01]

Precondition: Student is logged in.

Main Path:

1. User selects a course by clicking "Expand" on the course header.

2. *System shows the student a list of assignments for the selected course.*

3. User selects an assignment.

Alternate Paths:

1.1 User selects a different course by repeating Step 1.

3.1 Student selects a different course by repeating Step 1.

Postcondition: Student is on the student-assignment page

Name: Student downloads files from server

User stories addressed: [S-02]

Precondition: logged in, on student-coding page

Main Path:

1. Student selects the files from the list

2. Student clicks "download" from the commands

3. *System sends file to client*

Postcondition: *Still in student-coding*

Name: Administrator creates a user

User stories addressed: [A-06]

Precondition: Admin is logged in and on administrator-usermanagement page

Main Path:

1. Admin fills out the user information
2. Admin clicks "Save"
3. *The system saves the user and allows them to be added to courses and given roles*

Alternate Paths:

2.1. Admin clicks "cancel"

3.1. *The system displays a confirmation dialog: "*

Are you sure you wish to cancel?"

4.1. If the user clicks no, back to step 1, otherwise on to 5.1

5.1. *System reloads the administrator-usermanagement page*

Postcondition: logged in, still on administrator-usermanagement page

Name: Administrator assigns a user to a course

User stories addressed: [A-01, A-02, A-03]

Precondition: Admin is logged in and on administrator-management page

Main Path:

1. Admin clicks on a course name (which highlights it)
2. Admin clicks "Edit"
3. *System loads administrator-coursemanagment page*
4. Admin adds users to the course by selecting them, then and their role and then clicking "Add"

Alternate Path:

User already in class:

4.1. *System displays a dialog: "User already in class"*

Postcondition: Logged in, navigated to administrator-coursemanagement page

Name: Administrator creates a course

User Stories addressed: [A-05, A-06]

Precondition: logged in, on administrator-management page

<u>**Main Path:**</u>

1. Admin clicks "Add" under "Course Management".
2. *System loads the administrator-coursemanagement page*
3. Admin fills out the information for the course in the text boxes, then elects the different users for instructor, markers and students
4. Admin clicks "save"
5. *The system reloads the administrator-coursemanagement page*

<u>**Alternate Paths:**</u>

4.1. User clicks "cancel"

5.1. *The system displays a confirmation dialog: "*

Are you sure you wish to cancel?"

6.1. If the user clicks no, back to step 3, otherwise on to 7.1

7.1. *System reloads administrator-coursemanagement page*

Postcondition: logged in, navigated to administrator-coursemanagement page