COMP 4770: Team Project
FINAL - Iteration 4

Team C, Winter 2016

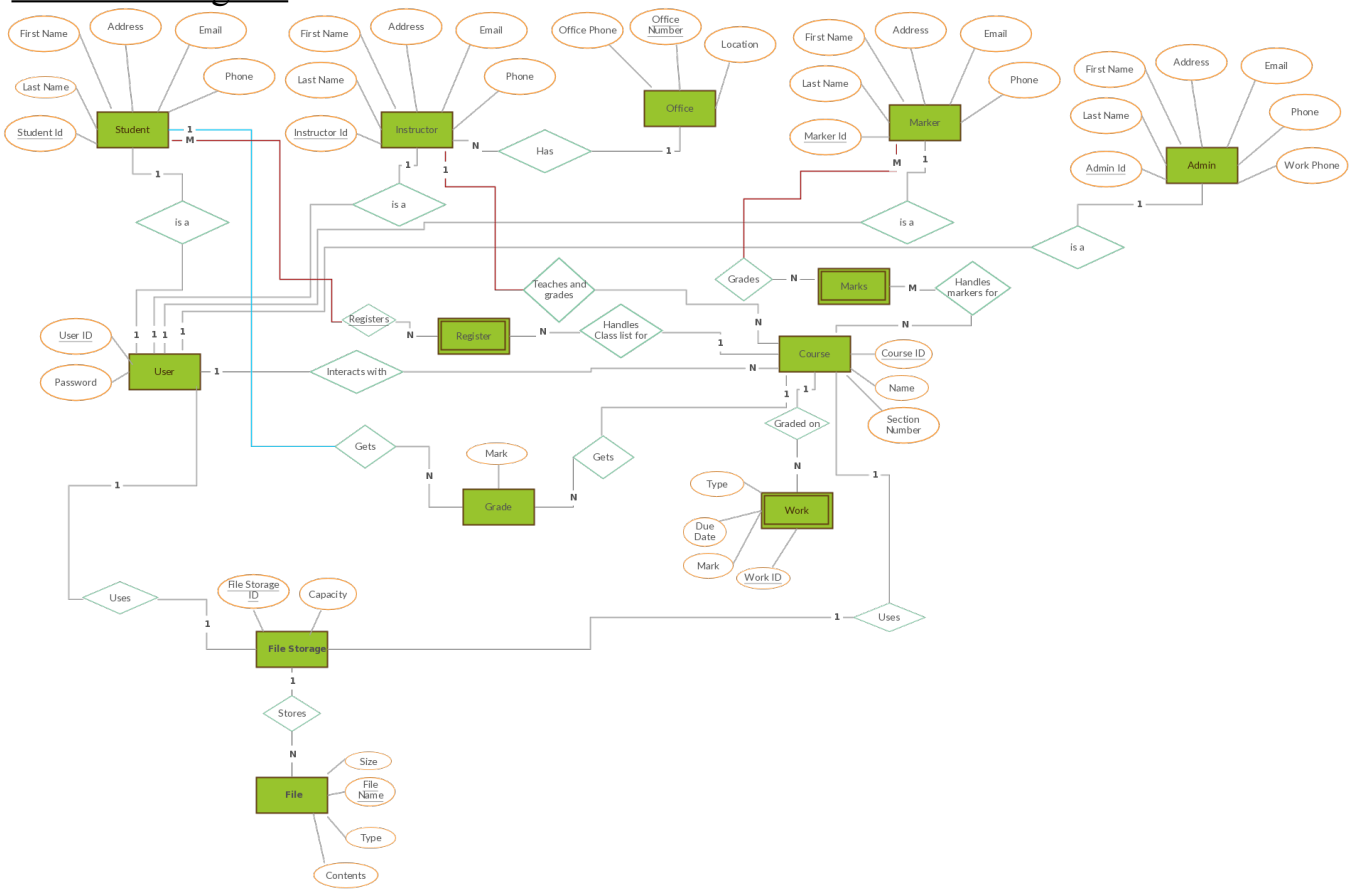
Tim Murray
Shane Butt
Dean Massecar
Philip St.Croix

Final Prototype

While the team had initially decided to use Java and Tomcat, after several weeks it was determined we were actually better off using NodeJS and other packages available through npm. In the end, we used Javascript as our language of choice, NodeJS as our server, SQLite3 as our database, as well as a number of other packages in npm.

We used CodeMirror, as was suggested, and this greatly simplified the text editor challenge on our site.

Database Diagram



Database Relations
See ./Diagrams/dbtablerelations.pdf

Domain Model
See ./Diagrams/domainmodel.vpp

Work Plan

Work Plan For CodeDrop Project January 8 - March 21

ID	Task Name	Duration
1.0	Iteration 0	2
1.1	- Requirement gathering	1
1.2	- Domain Model	1
1.3	- User Stories/Use cases	2
1.4	- State Chart	1
1.5	- Wire Frames	1
1.6	- Tech Selection	1
2.0	Iteration 1	3
2.1	- Database Schema	1
2.2	- APIs	3
2.3	- SoftWare Modeling	3
3.0	Iteration 2	3
3.1	- Adding Functionality	2
3.2	-Implementation	2
3.3	-testing	1
4.0	Iteration 3	3
4.1	- Further Implementation	3
4.2	- Testing	1
5.0	Iteration 4	4
5.1	- Finish implementation	4
5.2	- Update earlier iterations	1
5.3	- Testing	

This was the general plan, but as would be expected in a student project, was not actually followed all that accurately.

Wireframes

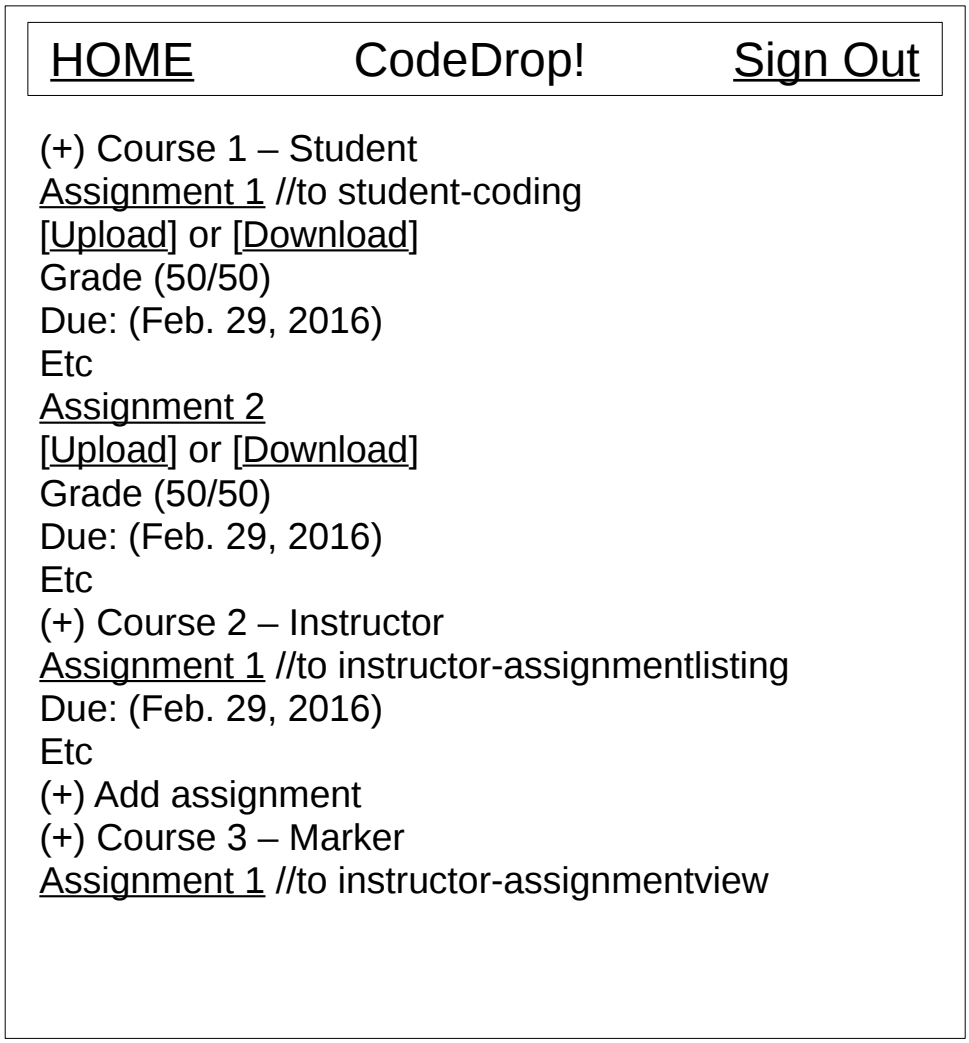
Below are wireframe diagrams of the individual screens that users will encounter in the system.

Universal Login page [universal-login]

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

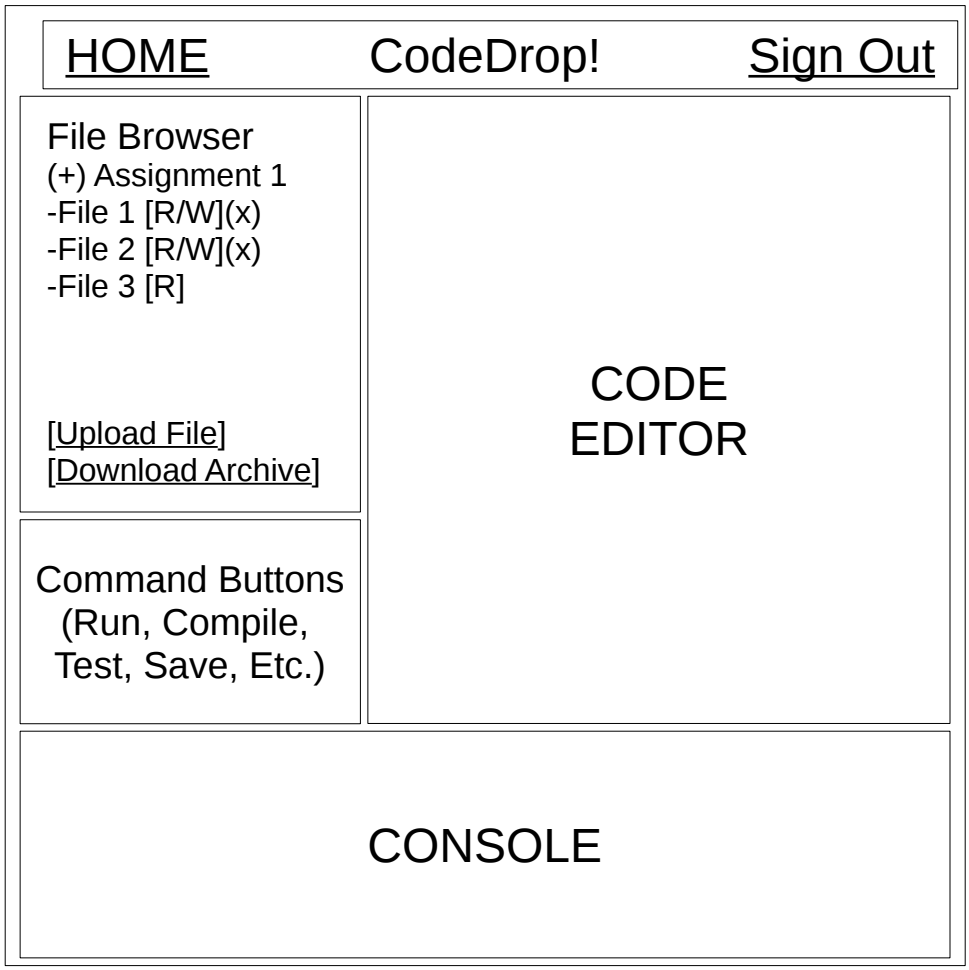
User Main Page: [user-mainpage]

This is the first page that all users see when they log in. What is shown depends on what roles in what courses each user has. This page should provide the primary access point into all required activities (for non-admin tasks)



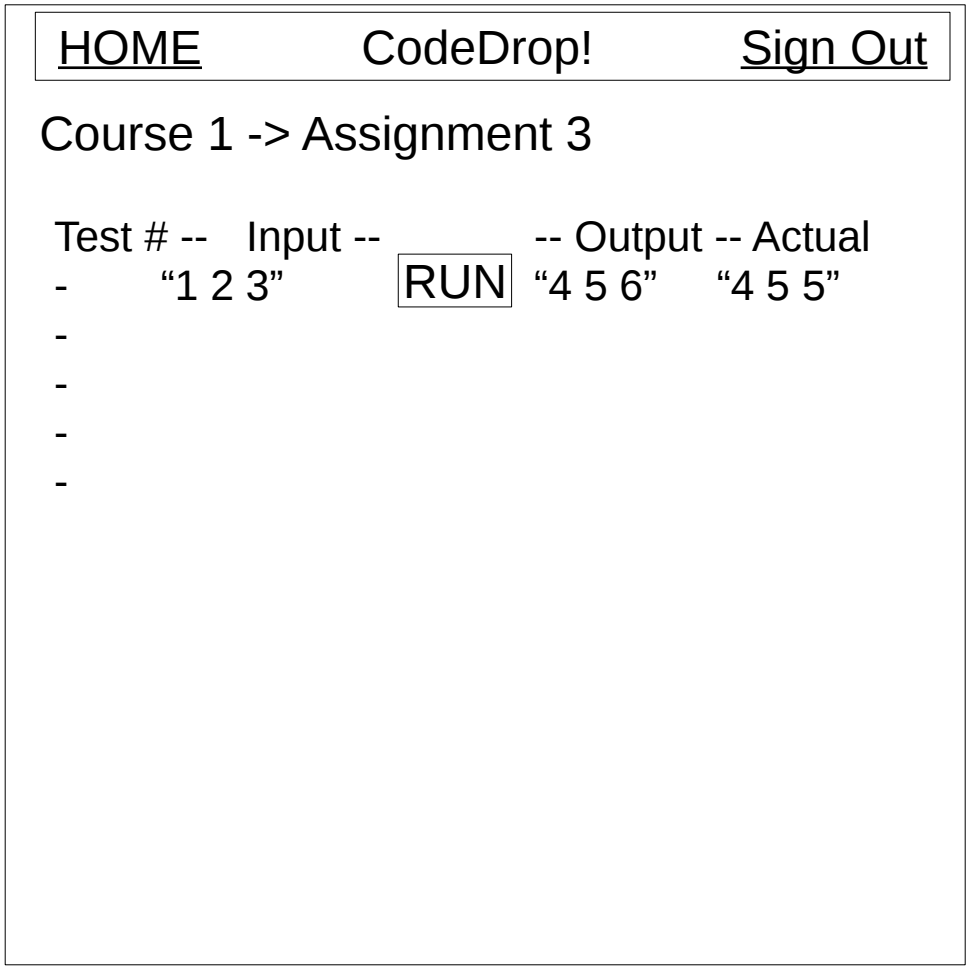
Student Coding Page: [student-coding]

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.



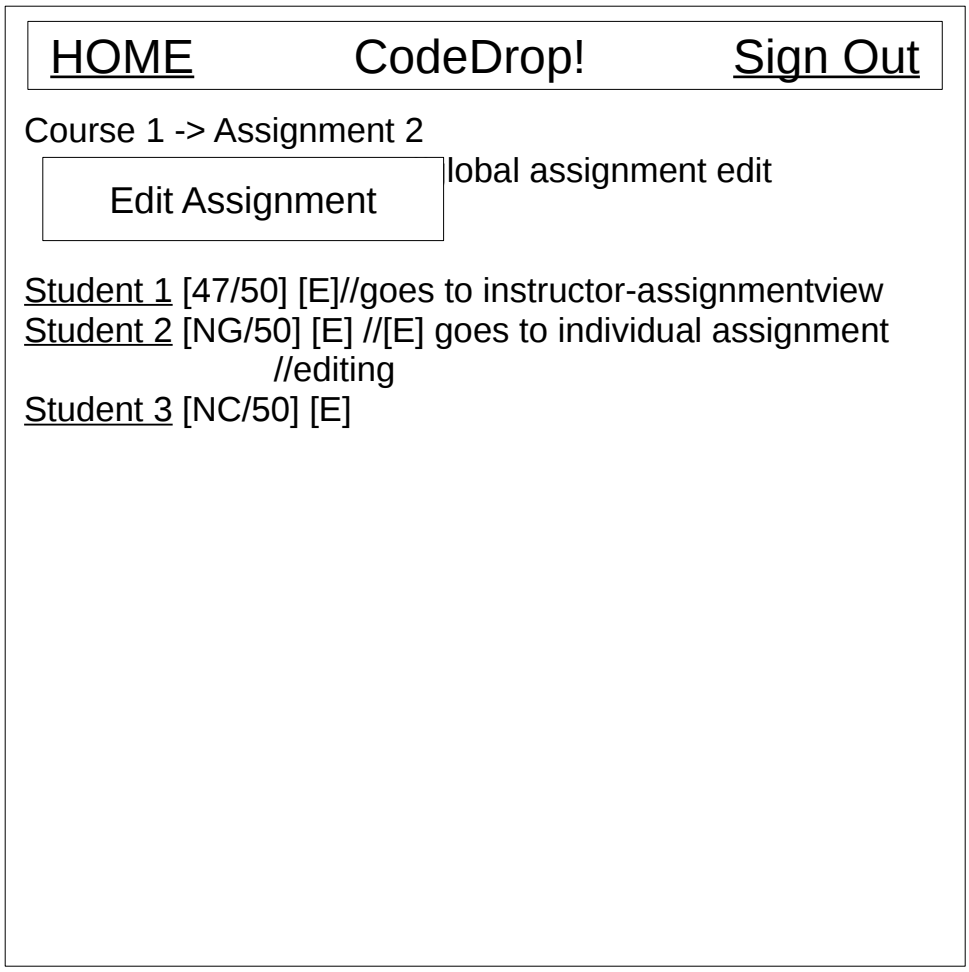
Student Testing Page: [student-testing]

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



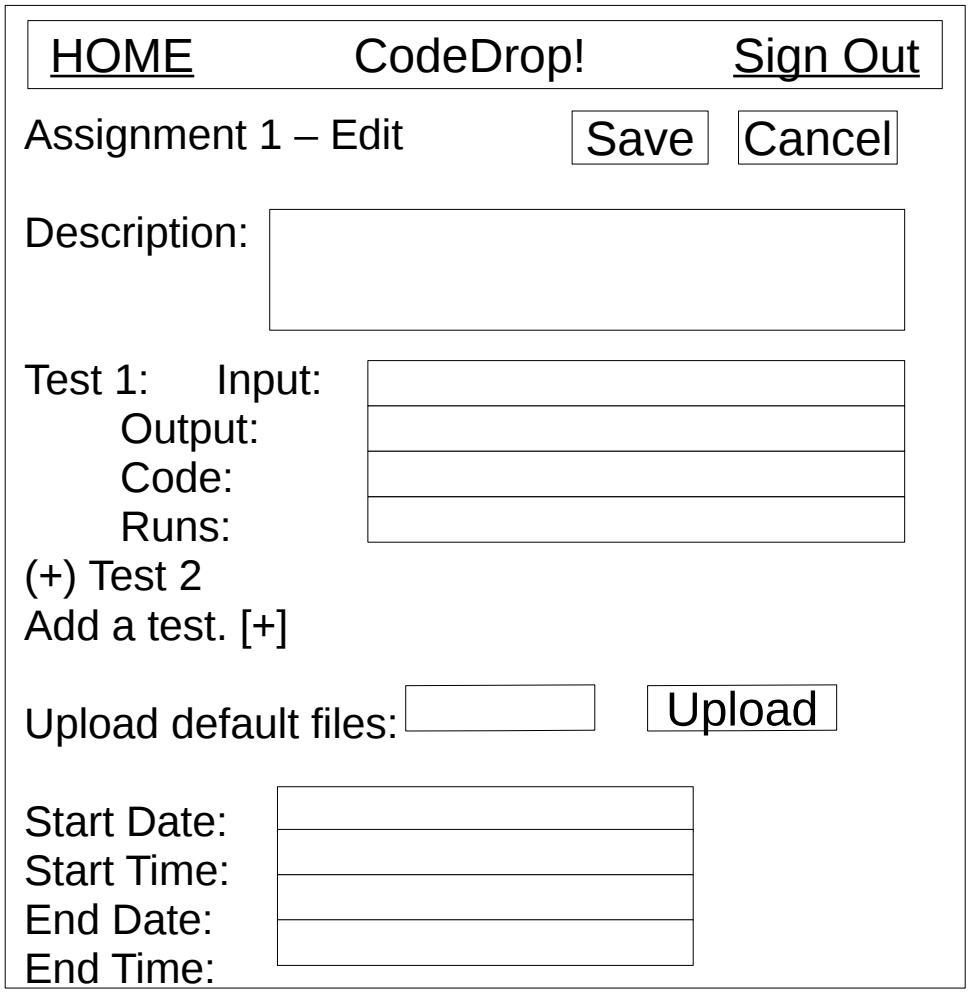
Instructor/Marker Course Management [instructor-assignmentlisting]

Allows Instructors or Markers to view and navigate through a course and all of its assignments and view all students assigned and their work. Instructors can edit assignments, globally and individually (say to give a certain student an extended submission time)



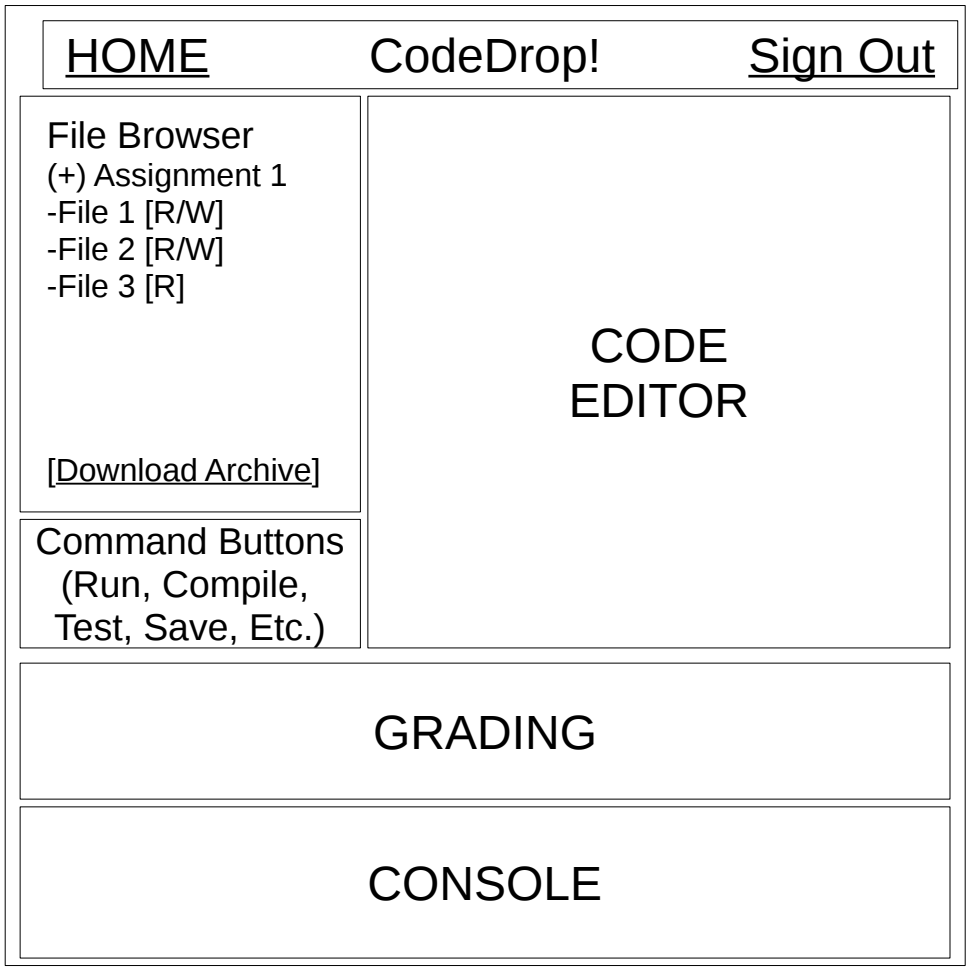
Instructor Assignment Edit Page: [instructor-assignmentedit]

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



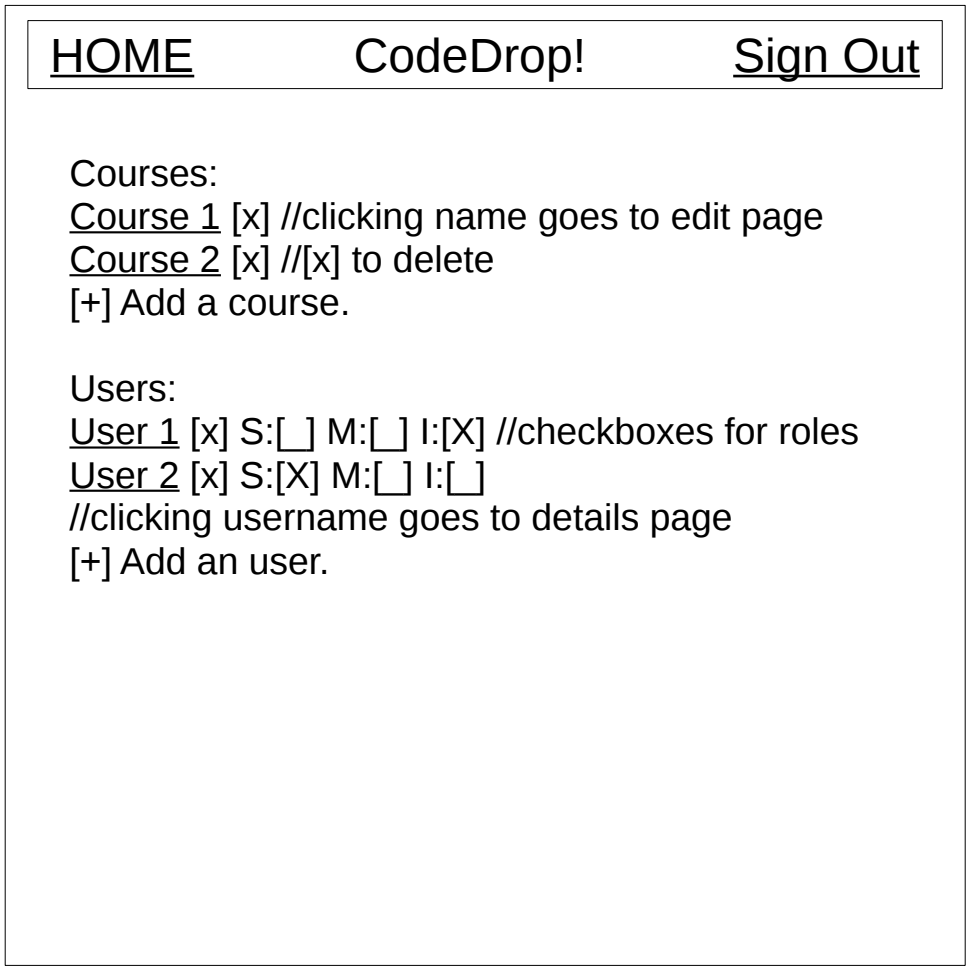
Assignment Viewing Page: [instructor-assignmentview]

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.



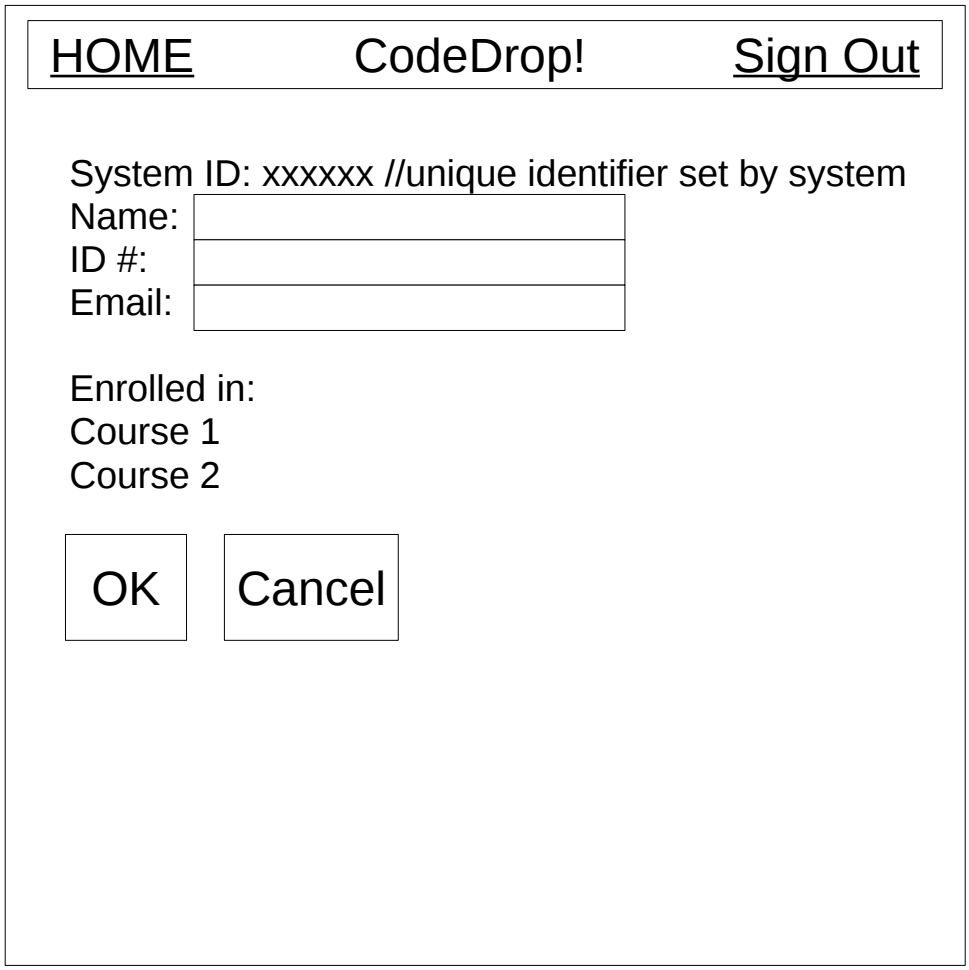
Admin Management Page: [administrator-management]

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.



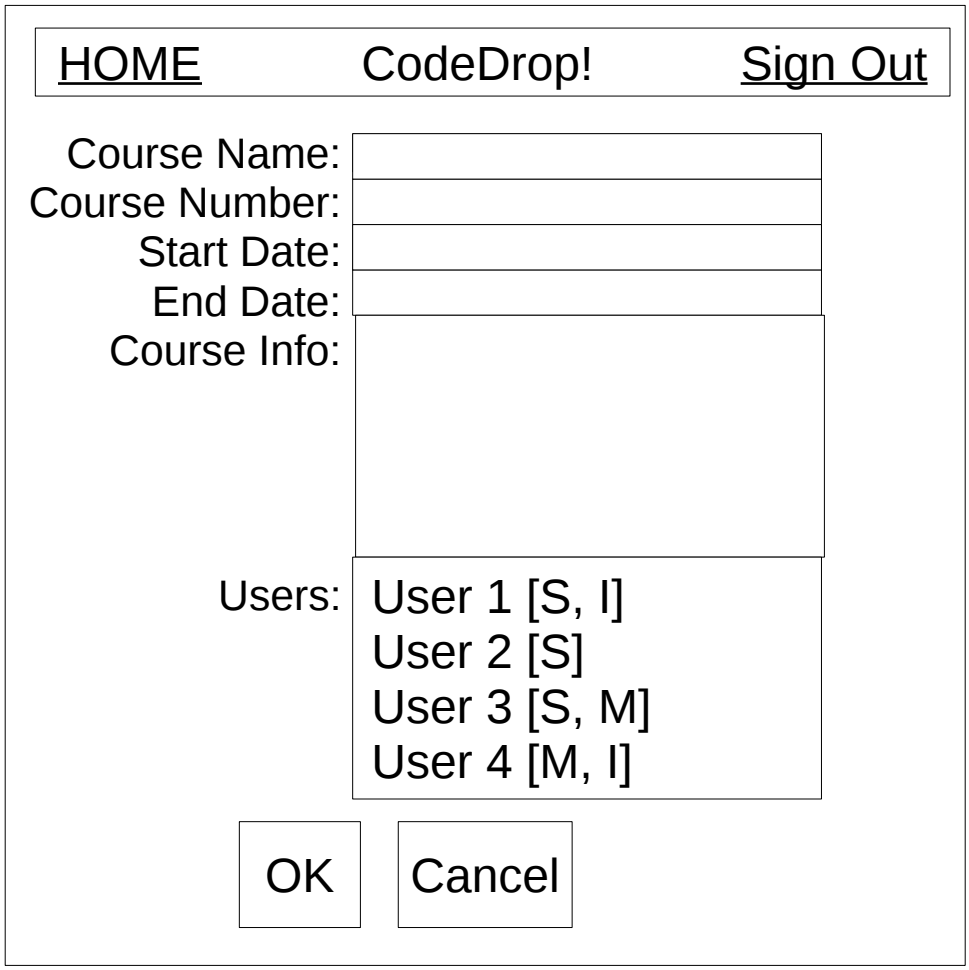
Admin User Management Screen: [administrator-usermanagement]

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

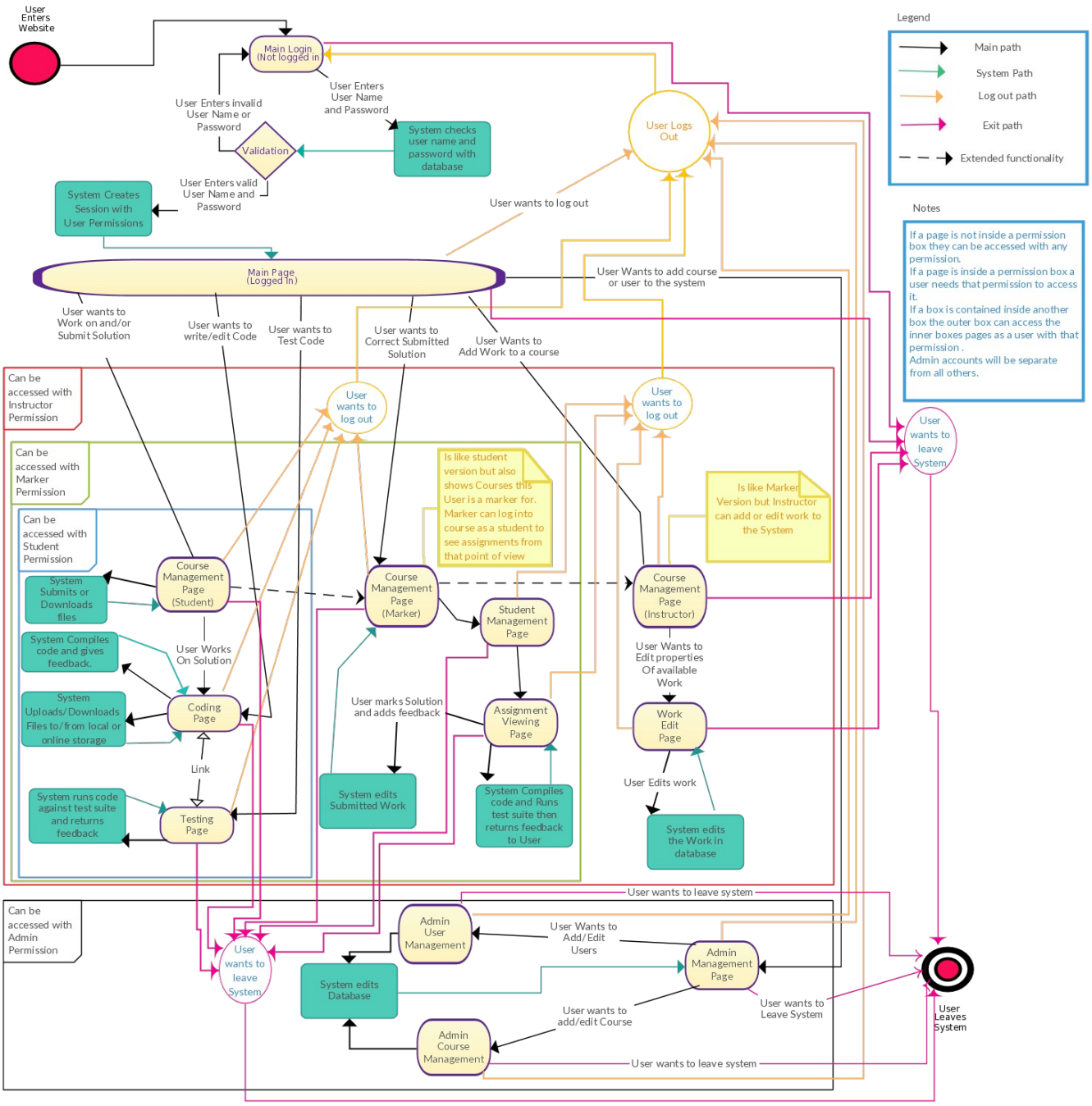


Admin Course Management Screen: [administator-coursemanagement]

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.



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

Main Path:

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*

Alternate Paths:

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

Test Plan for Code Drop

Overview:

This test plan is made up of Test cases that closely follow the Use Cases already provided, but they will possibly include more steps and details to ensure proper testing of all the functionality of each feature of the system. In addition, there will be more cases added to test each individual interactive UI element or any other function of CodeDrop™. There test cases are intended to be used when the testing begins, and some of the details will be filled out then, as needed, such as bugs and status.

Test Case Format:

Name:

Description of Feature to Test:

Preconditions (if any):

Steps to Follow:

The following happens either during or after the step execution:

Bugs Found:

Status:

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Estimated time to to complete all tests: *TBD*

Test Cases:

Name: Log in as a user

Description of Feature to Test: Logging in as a specific user, different users have different roles and permissions depending on what was set when the account was created.

Preconditions (if any): N/A

Steps to Follow:

1. Type in User name and password for the User.
2. Click the “Login” button.

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Add an assignment to a course

Description of Feature to Test: Instructors have the ability to add assignments from the instructor-management page which can be seen and completed by users enrolled in the course the assignment is added to.

Preconditions (if any): Logged in as Instructor and navigated to instructor-management page.

Steps to Follow:

1. Click “expand” (the plus symbol) for the desired course.
3. Click “Add”
4. Edit all appropriate information
5. Add tests (by clicking the plus/add button) with appropriate information for each one
6. Click “save”

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Grade a submitted assignment as Instructor

Description of Feature to Test: Users set as Instructor can grade submitted assignments from students from the assignment-grading/viewing page.

Preconditions (if any): Logged in as Instructor, navigated to instructor-management page.

Steps to Follow:

1. Click on “Expand” (the plus symbol) for the desired course.
7. Click “Expand” on the desired student.
8. Click on the most recently submitted assignment.
9. Expected and given results for the tests are displayed.
10. Leave comments in the comment text box for each test result.
11. Submit the assignment grade by clicking “save”.

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Grade a submitted assignment as a Marker or Instructor

Description of Feature to Test: Users set as Marker or instructor can grade submitted assignments from students from the assignment-grading/viewing page.

Preconditions (if any): Logged in as Marker or instructor

Steps to Follow:

1. Click on “Expand” (the plus symbol) for the desired course.
12. Click “Expand” on the desired student.
13. Click on the most recently submitted assignment.
14. Expected and given results for the tests are displayed.
15. Leave comments in the comment text box for each test result.
16. Submit the assignment grade by clicking “save”.

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Make an edit to an individual student’s assignment as an Instructor.

Description of Feature to Test: Instructors have the ability to edit an existing assignment belonging to an individual student from the assignment-listing page.

Preconditions (if any): Logged in as Instructor

Steps to Follow:

1. Click “expand” (the plus symbol) for the desired course
17. Click on the desired assignment.
18. Click on “edit” for the student who’s assignment is to be edited.
19. Make edit to assignment details (test all fields)

20. Click “save”

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Make an edit to an assignment (globally) as an Instructor.

Description of Feature to Test: Instructors have the ability to edit an existing assignment belonging to an entire course.

Preconditions (if any): Logged in as Instructor

Steps to Follow:

1. Click “expand” (the plus symbol) for the desired course
21. Click on the desired assignment.
22. Click on “edit assignment”
23. Make edit to assignment details (test all fields)
24. Click “save”

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Compile and test code as Student

Description of Feature to Test: A student has the ability to write, compile and test code in the student-testing page.

Preconditions (if any): Logged in as student, navigated to student-coding page.

Steps to Follow:

1. Type simple code in text box provided.
25. Click “compile”
26. Click “Run”
27. View results in the console window.

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Test code against expected results for an assignment as a Student.

Description of Feature to Test: A student has the ability to, after writing code, to test against the expected results and then submit their code for the assignment. Note: The system saves automatically every [insert interval of tim here] so there is no need to click a “submit” button, once the sue date is reached the assignment editing locks for the student and the latest version is the one submitted for grading.

Preconditions (if any): Logged in as student, navigated to student-testing page, have code written in the student-coding page.

Steps to Follow:

1. Click on “Test”
28. Run each test against the expected results by clicking “run”

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Download archive with each type of permissions (excluding admin)

Description of Feature to Test: The ability to download archive of assignment files is available for each kind of user permissions (excluding admin) for a given course.

Preconditions (if any): Logged in as either student, instructor or marker, navigated to instructor-assignmentview or student-coding page respectively. There is at least one file available to download.

Steps to Follow:

1. Click “download archive”

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Upload file containing pre-written code as a Student from either main page or coding page

Description of Feature to Test: A student can choose to upload a file containing code that they have made elsewhere and use it as code for their assignment.

Preconditions (if any): Logged in as a student, navigated to the student-coding page OR the User-mainpage.

Steps to Follow:

1. Click “Upload”
29. Choose desired file

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Create user as Administrator

Description of Feature to Test: The administrator can create users and add them to the system.

Preconditions (if any): Logged in as Administrator

Steps to Follow:

1. Click “add user”
30. Fill out required information about user
31. click “OK”

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Create course as an Administrator

Description of Feature to Test: The administrator can create courses and add them to the system

Preconditions (if any): Logged in as an administrator, at least one user created

Steps to Follow:

1. Click “add a course”
32. Fill out required information about course
33. Choose at least one user to add to the course and assign one or more permissions
34. Click “OK”

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Delete course as an Administrator

Description of Feature to Test: The administrator can delete created courses from the system.

Preconditions (if any): Logged in as an administrator, at least one course created

Steps to Follow:

1. Press the “X” button next to the desired course
35. The course is deleted

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Delete User as an Administrator

Description of Feature to Test: The administrator can delete created users from the system.

Preconditions (if any): Logged in as an administrator, at least one course created

Steps to Follow:

1. Press the “X” button next to the desired user
36. The user is deleted

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: Delete a file for an assignment as a Student

Description of Feature to Test: The students can delete files from the system that they have uploaded for an assignment from the student-coding page.

Preconditions (if any): Logged in as an administrator, navigated to student-coding page, at least one file uploaded for an assignment

Steps to Follow:

1. Press the “X” button
37. The file is deleted

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Name: View an assignment as Student, marker or instructor

Description of Feature to Test: The users can view assignments by clicking them on the user-mainpage.

Preconditions (if any): Logged in as student, instructor or marker

Steps to Follow:

1. Click the desired assignment
38. Make sure the appropriate assignment has loaded with all details

The following happens either during or after the step execution:

Bugs Found: To be filled out if applicable when test is run.

Status: Not started.

☐ *Passed*

☐ *Failed (Reason: _____)*

☐ *Paused (Reason: _____)*

Additional Notes:

Data Dictionary

CodeDrop:
Name given to the project.

System:
This refers in general to all aspects of the project such as the web pages ,server , databases etc, etc.

Web Pages:
The interface that Users will use to interact with the Server.

Database:
The store of information used in the system.

Server:
The Link that allows users to access system from anywhere. Such that a user sends a request using the web page, this sends the request to the server, server sends request to database, database sends results to server and server returns the results via web page to the user.

Log in:
To gain access to system.

Course:
A group of students with an instructor that may have one or more markers, in which students complete similar work for a grade.

Work:
Any gradable object such as tests, quizzes, assignments, projects, etc etc.

User:
Anyone who can log in and use the system.

Privilege:
Sets what a user is and is not allowed to do on the system.

Student:
A user Privilege that can submit work to the system to be corrected.

Marker:
A user Privilege that can correct submitted work for a course, extends Student.

Instructor:
A user Privilege that can assign work, correct work, edit work for a course, extends Marker.

Administrator(Admin):
A user that has control of the system. They add/remove users and courses to the system, as well as assign Students/Instructors/Markers to courses.

Test Suite:
A set of tests to be run against code to determine it's functionality.

Compile:
Means to turn written code into program computer can read.

Feedback(regards to code):
Message returned when compiling/running code. Could be an error message that occurs at compile/run time or results of running a test.

Feedback(regards to Work):
This is the the general term for things like Marks and comments on work telling students what they did wrong.

Due Date:
The date some work is due by.

Solution:
A students answer to some work.

Submission:
A solution a student presents for grading.

Upload:
To take a set of files from a user's computer and adding them to the system.

Download:
To take a set of files from the system and add them to a user's computer.

Console:
A window that allows the user to run commands. This will mainly be used by students that want to use a Linux like environment to code. Not as powerful as a Linux console.

Cloud:
The System space for uploaded files.

Errors:
Two types Runtime, a logical error that occurs when running code and error occurs ex (1/0), and Compiletime error , An error in the code when compiling such as variable type mismatching.