

COMP 4770 Team Project Design Document
Team G
Tyler Bennett, Michelle Croke, Rebecca Price,
Michael Singleton, Scott Young
February 8th 2016

Contents:

User Stories	1
Use Cases	1
Data Dictionary	5
Domain Model	6
Collaboration Diagrams	6
Statechart	11
Wireframes	12
Test Plan	13
Technology Stack	14
Workplan	14

USER STORIES

1. As a student user, I need to edit code for assignments.
2. As a student user, I need to create new files on the system to work on them.
3. As a student user, I need to compile code to test my code.
4. As a student user, I need to run the output programs to test my program.
5. As a student user, I need to submit files to complete assignments.
6. As a student user, I need to view graded assignments to learn from my previous mistakes.
7. As a student user, I need to view my progress report/grades to gauge my success in the course.
8. As a student user, I need to download code in order to maintain a backup.
9. As a student user, I need to upload code in order to submit assignment completed on my local machine.
10. As a student user, I need to download (cross)compiled code to test programs on my local machine.
11. As a student user, I need to create test suites to unit test my code.
12. As a student user, I need to run test suites to unit test my code.
13. As a student user, I need to view all active/inactive assignments assigned by the professor
14. As a prof/TA user, I need to be able to access student's assignments to review/evaluate them.
15. As a prof/TA user, I need to be able to change grades to fix errors.
16. As a prof/TA user, I need to run test suites to mark student's code
17. As a prof user, I need to upload assignments for students to access.
18. As a prof user, I need to view my student list to see which students are in my class.
19. As a prof user, I need to view my TA list to see which TAs are in my class.
20. As a prof user, I need to upload grades for students to view.
21. As a prof user, I need to add comments on submissions for student feedback.
22. As a prof user, I need to create test suites to mark student's code
23. As a prof user, I need to assign TA's for each course that I have.
24. As an administrator, I need to be able to create prof accounts so professors can create courses.
25. As an administrator, I need to be able to add/remove student accounts in case the professor cannot.
26. As a TA user, I need to be able to view all uploaded assignments of the class.
27. As a TA user, I need to be able to compile and run uploaded student code.
28. As a TA user, I need to be able to assign a grade to each student assignment.
29. As a TA user, I need to be able to view the assigned assignments.
30. As a user, I need to log in to use the service.
31. As a user, I need to log out to stop using the service.

USE CASES

1) Name: Logging In

User Stories Addressed: 30

Actor: All

Preconditions: User info saved to the system server.

Basic Flow:

1. Input User information (username and password)
2. Click "Log In"
3. The system logs in with user info

Alternate Flow:

3.1 Invalid user info

Post condition: User is viewing the home page.

2) Name: Logging Out

User Stories Addressed: 31

Actor: All

Precondition: User is currently logged in.

Basic Flow:

1. Click "Log Out"
2. The system logs the user out

Postcondition: User is viewing the log in screen.

3) Name: Viewing Assignments

User Stories Addressed: 6, 13, 14, 26, 29

Actor: All

Preconditions: Assignment exists

Basic Flow:

1. Log in (see Logging in)
2. Click on relevant course
3. Click on view assignments
4. (Student Optional) Work on an assignment

Alternate Flow:

- 4.1 (Prof/TA Optional) View submitted assignments (see Using Systems Editor and Testing Code)
- 4.2 (Prof/TA Optional) Grade submitted assignments (See Testing Code and Uploading Grades)

Postconditions: The actor has viewed the assignment(s) and taken any necessary actions.

4) Name: Creating a File in Systems Editor

User Stories Addressed: 2

Actor: Students

Preconditions: Students must have an account and in class, user must be logged in, must be in editor, signed into class by Prof

Basic Flow:

1. Select assignment to work on
2. Open up in editor
3. Click "Create New"
4. New file is created and is ready to use

Alternate Flow:

- 1.1 No assignment selected; not opened

- 4.1 See: Use Case 5, log out

Postconditions: The file has been created which can now be edited, saved or submitted.

5) Name: Edit a File in Systems Editor

User Stories Addressed: 1

Actor: Students

Preconditions: Students must have an account and in class, user must be logged in, must be in editor, signed into class by Prof, file must already be created.

Basic Flow:

1. Select assignment to work on
2. Open up in editor
3. Click "Open File"
4. Select file to be opened

Alternate Flow:

- 1.1 No assignment selected; not opened
- 2.1 File is already opened once created, see 5.

5. File is opened and able to be edited

6. Save file or see: Use Case 6

6.1 Log out

Postconditions: The user has edited/added criteria and the file is saved or submitted.

6) Name: Submitting a File in Systems Editor

User Stories Addressed: 3, 4, 5, 27

Actor: All

Preconditions: Students must have an account and in class, user must be logged in, must be in editor, signed into class by Prof, file must already be created.

Basic Flow:

1. Select file to be submitted
2. Open up in editor
3. Click "Submit File"
4. Submit file to Prof for grading

Alternate Flow:

- 1.1 File is already open, see 3.
- 2.1 See: User Case 7, test before submission
- 4.1 Log out

Postconditions: The assignment should be submitted.

7) Name: Testing Code

User Stories Addressed: 11, 12, 16

Actor: All

Preconditions: Code has been written and needs to be tested against the assignment requirements.

Basic Flow:

1. Log in (see Logging in)
2. Click on relevant course
3. Click on "view assignments"
4. Click on an assignment
5. Click on "Run Against Tests"

Alternate Flow:

- 3.1 (Prof/TA) Click on a student
- 5.1 No test suite exists, create one

Postconditions: A list of passed and failed tests appears for the user to see.

8) Name: Upload Files to Assignment

User Stories Addressed: 9

Actor: Student, Prof

Preconditions: A file(s) was created offline and needs to be upload to the assignment.

Basic Flow:

1. Log in (See Logging in)
2. Click on course
3. Click on assignment
4. Click "Upload File"
5. Select File(s) to upload in file browser
6. Click "Ok"
7. Click "Upload"

Alternate Flow:

Postconditions: The file(s) are now in the online assignments directory.

9) Name: Uploading assignments for students

User Stories Addressed: 17,22

Actor: Prof

Preconditions: Prof must have an account.

Basic Flow:

1. Select course

Alternate Flow:

2. Create new assignment
3. Upload assignment (See "Upload Files to Assignment")

Postconditions: assignment will be uploaded. Students can access.

10) Name: Download Files from Assignment

User Stories Addressed: 8, 10

Actor: All

Preconditions: Assignment(s) exist on system and has one or more files in it that the user wishes to retrieve.

Basic Flow:

Alternate Flow:

1. Log in (See Logging in)
2. Click on course
3. Click on assignment
4. Select a file
5. Click "Download File"

5.1 Click "Cross-Compile" if C/C++ source file

Postconditions: The user has the file on their local machine .

11) Name: Viewing Students Grades

User Stories Addressed: 7

Actor: Students

Preconditions: Assignment must be given, TA must have uploaded student's grades/reports.

Basic Flow:

Alternate Flow:

1. Select Grades
2. View marks for each assignment
3. Select specific assignment to view feedback
- 4.1 Feedback unavailable

Postconditions: grades have been viewed.

12) Name: Assigning TA to course

User Stories Addressed: 23

Actor: Prof

Preconditions: TA must have an account, Prof must be in course.

Basic Flow:

Alternate Flow:

1. Select course
1. Select TA for course
2. Save

Postconditions: TA will have access to student in course assignment.

13) Name: Grading Students Assignments

User Stories Addressed: 15, 20,21,28

Actor: TA, Prof

Preconditions: Past due date of assignment, Student's submitted assignment to be graded, Prof submitted assignment and marking key.

Basic Flow:

Alternate Flow:

1. Select assignment
2. System Grabs the submitted assignments and displays them in submitted assignment page
3. Select assign to grade

4. Open in Marking editor
5. Add comments on assignment
6. Save comments

6.1 Assign already graded, go to submitted assign list

7. Grade students work
8. Save grading

8.1 Assign already graded, go to submitted assign list

9. Click the "Submit Grade" button
10. The System sends grades to Prof

9.1 Missing grade, remain on page

Postconditions: TA is viewing the "Submitted Assignments" page.

14) Name: Create Accounts

User Stories Addressed: 23, 27

Actor: All

Preconditions: Some user needs an account.

Basic Flow:

1. Connect to the server
2. Create account for user
3. Set up user credentials (username/password)
4. Add user's basic information
5. Add user's information specific to account type
6. Add class list for each course added
7. Complete account creation (save information)

Alternate Flow:

1.1 Cannot connect to server

6.1 Class list not submitted, not added to user

7.1 Insufficient info, account not created

Postconditions: User account ready for use.

DATA DICTIONARY

Account - An arrangement in which a person uses the Internet services of our software.

Admin - A person whose job is to manage the state of the software.

Assignment - The assigned work the prof gives the students.

Code - Program instructions.

Course - A number of lectures or other matter dealing with a subject.

Feedback - Helpful information or criticism that is given to someone to say what can be done to improve a performance.

File - A named collection of data or information.

Grades - A letter, number, or other symbol indicating the relative quality of a student's work in a course.

Prof - A teacher of the highest academic rank in a college or university, who has been awarded the title Professor in a particular branch of learning.

Role - The function assumed or part played by a person or thing in a particular situation.

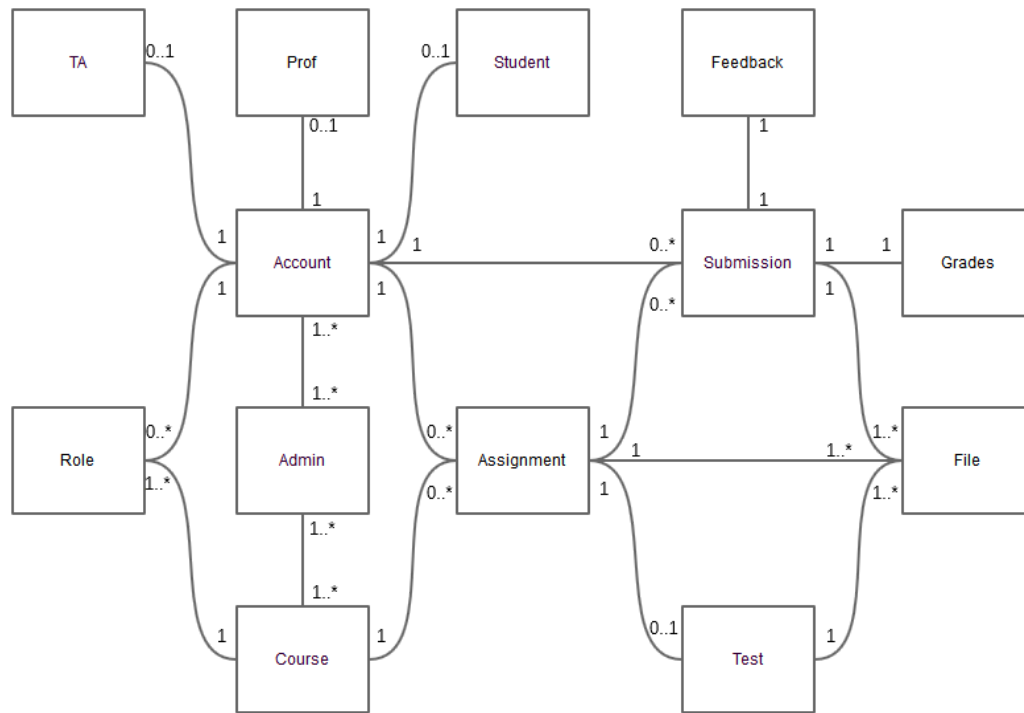
Student - A person who attends a school, college, or university

Submission - A presentation of code for consideration or judgment.

TA - An individual who assists a teacher with instructional responsibilities.

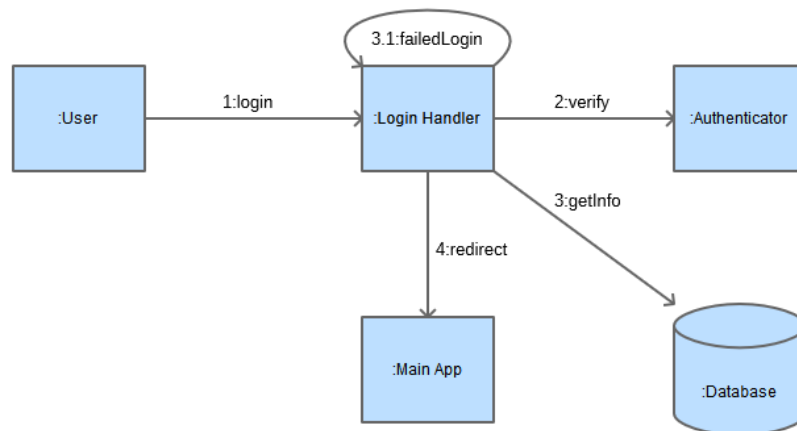
Test Suite - A collection of test cases that are intended to be used to test a software program to show that it has some specified set of behaviours.

DOMAIN MODEL

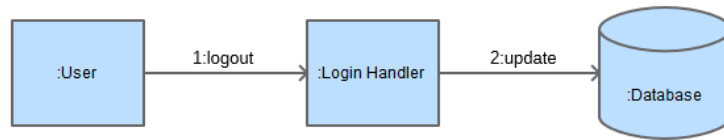


COLLABORATION DIAGRAMS

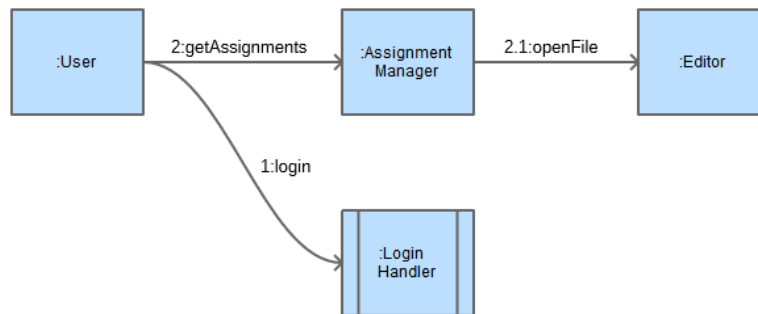
Logging In



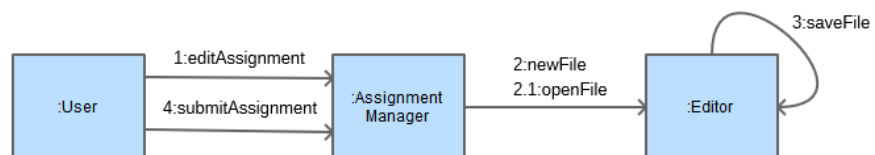
Logging Out



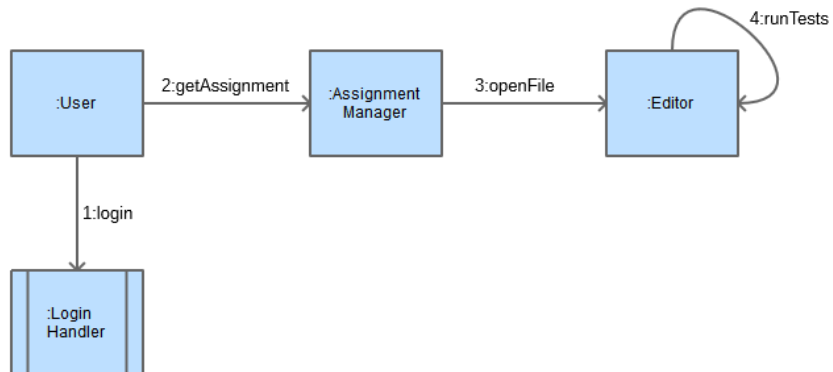
Viewing Assignments



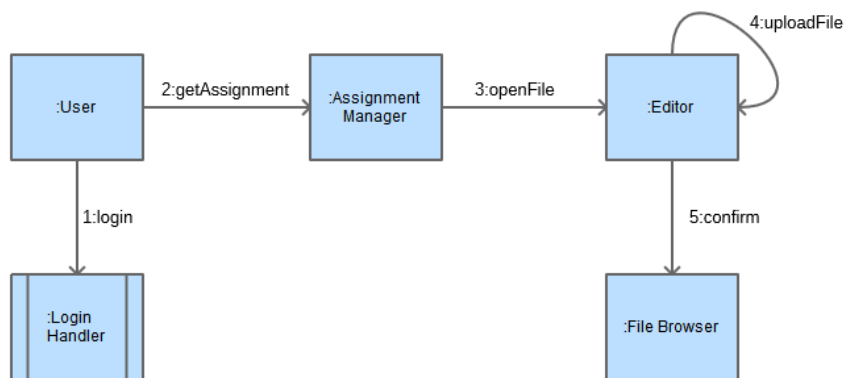
Using Systems Editor



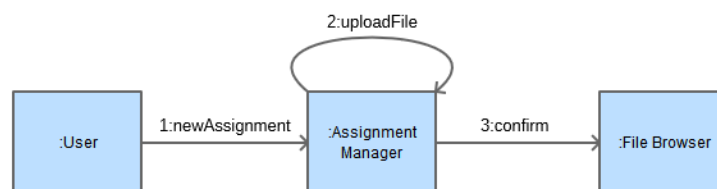
Testing Code



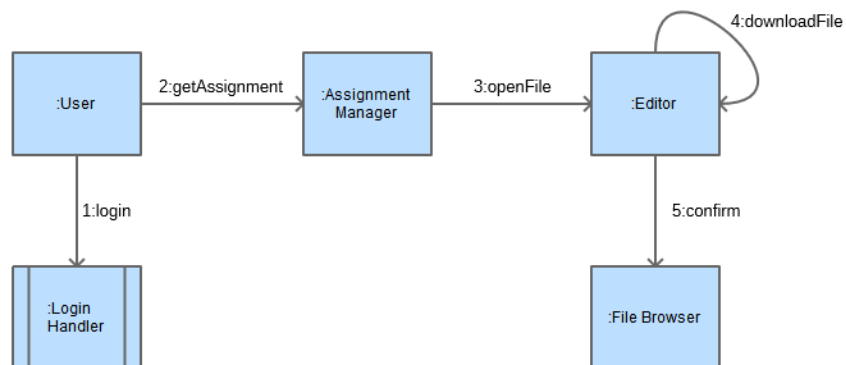
Uploading Files to Assignment



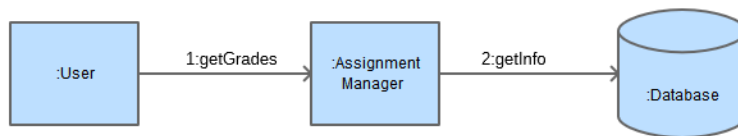
Uploading Assignments for Students



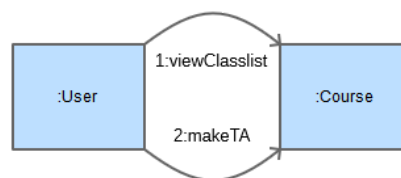
Downloading Files From Assignment



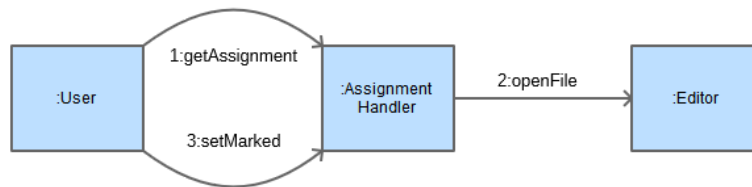
Viewing Student Grades



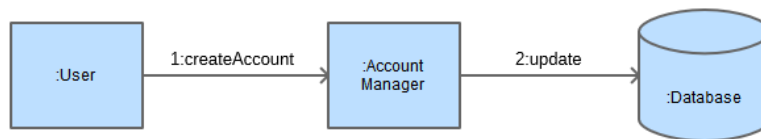
Assigning TA to Course



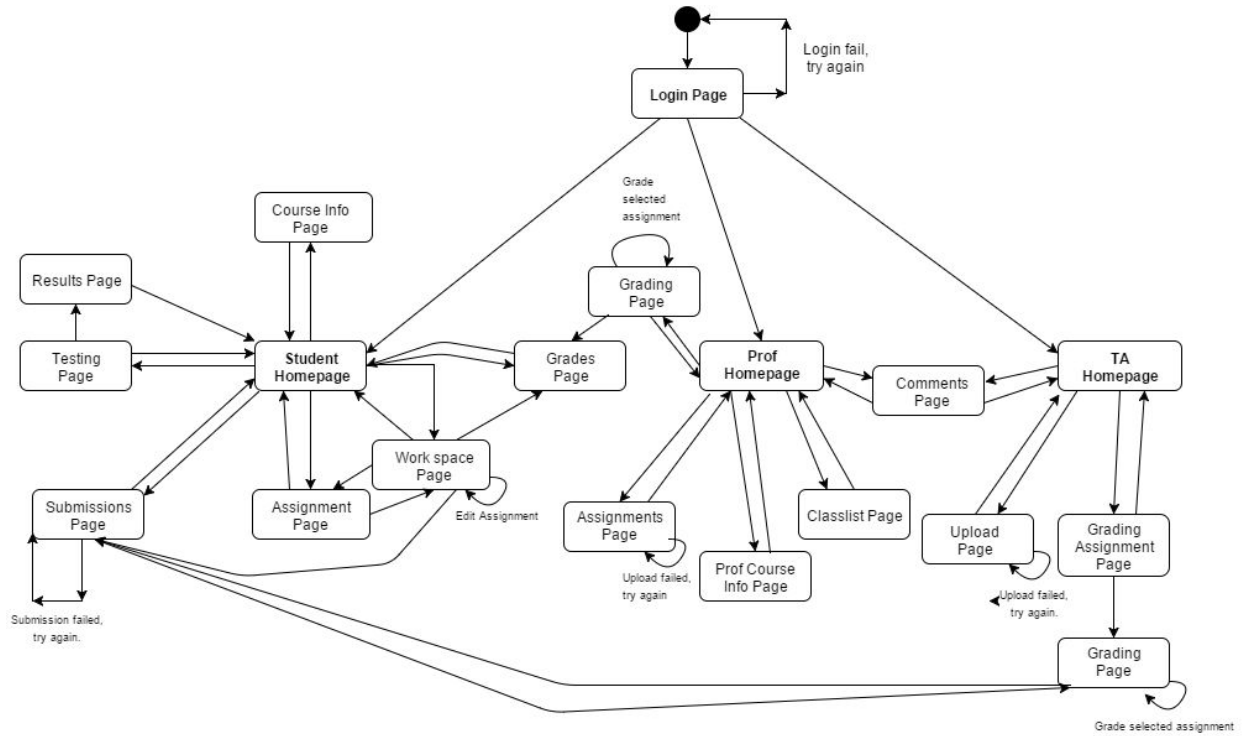
Grading Student Assignments



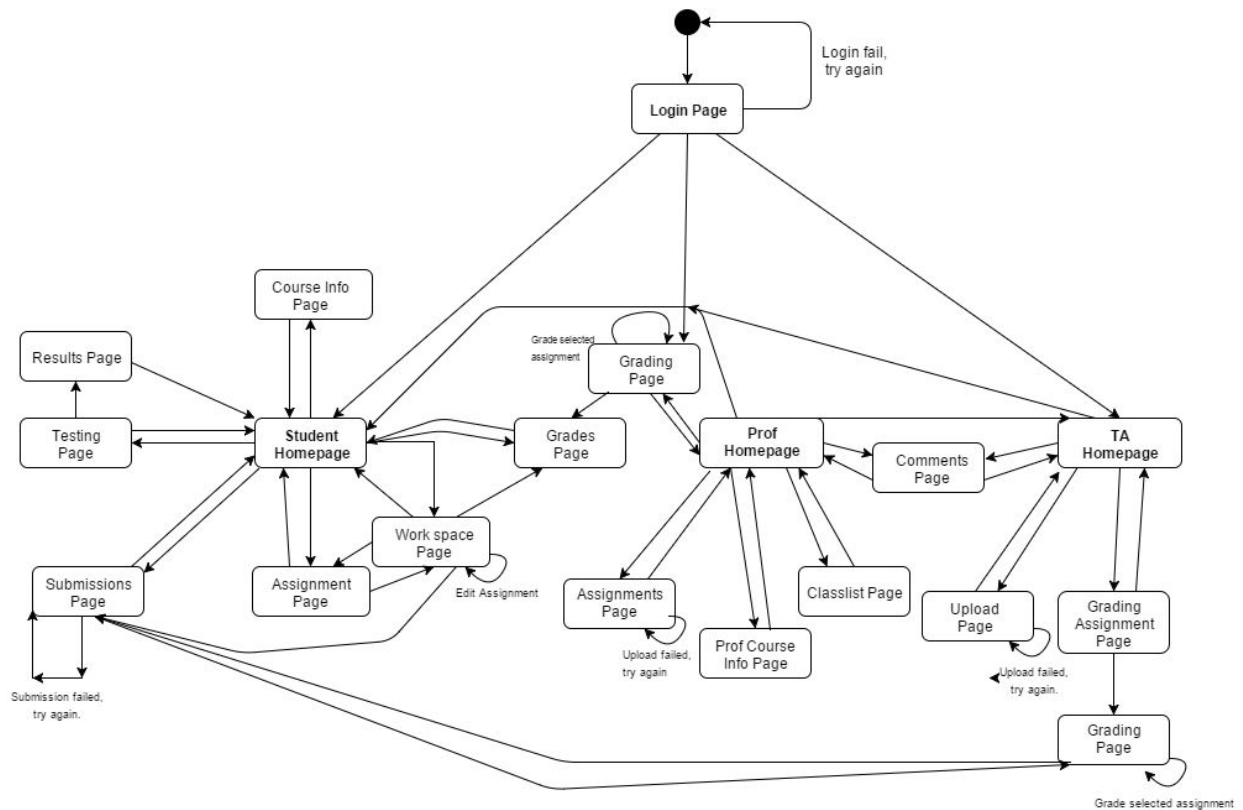
Creating Accounts



STATECHART

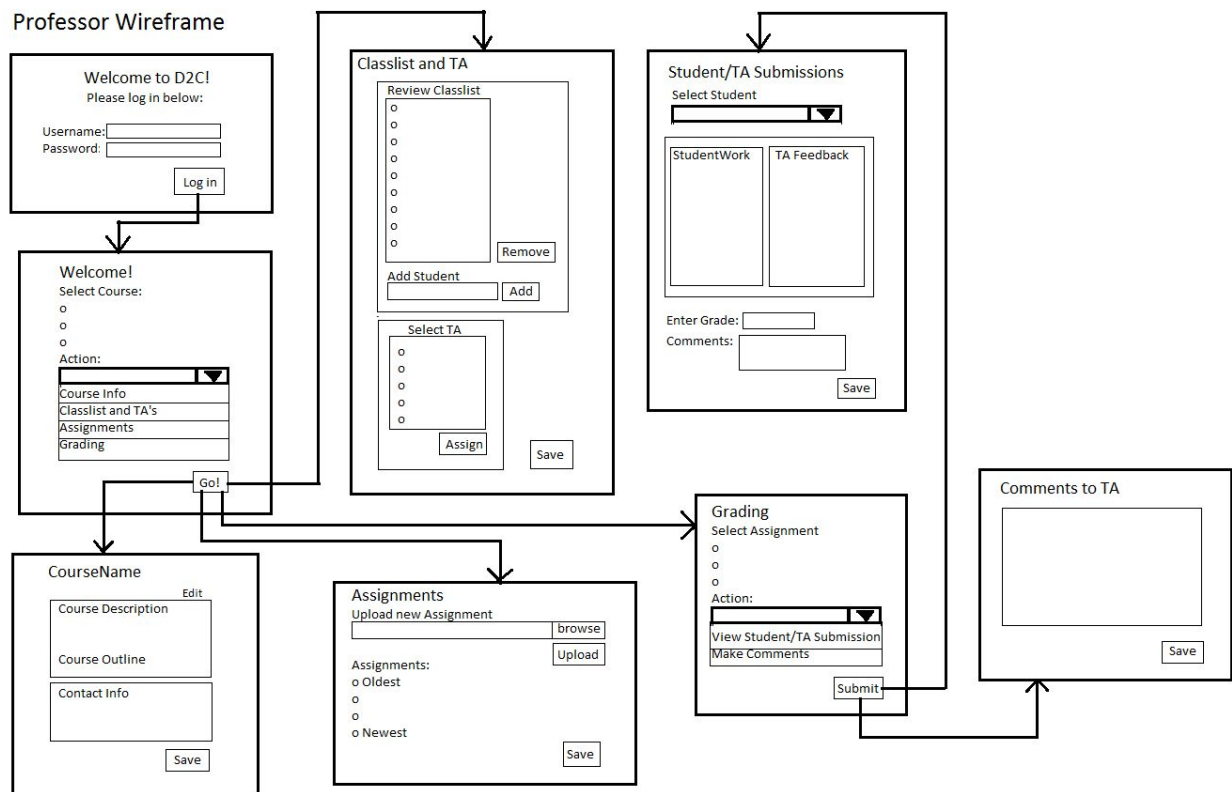


Revised Statechart:

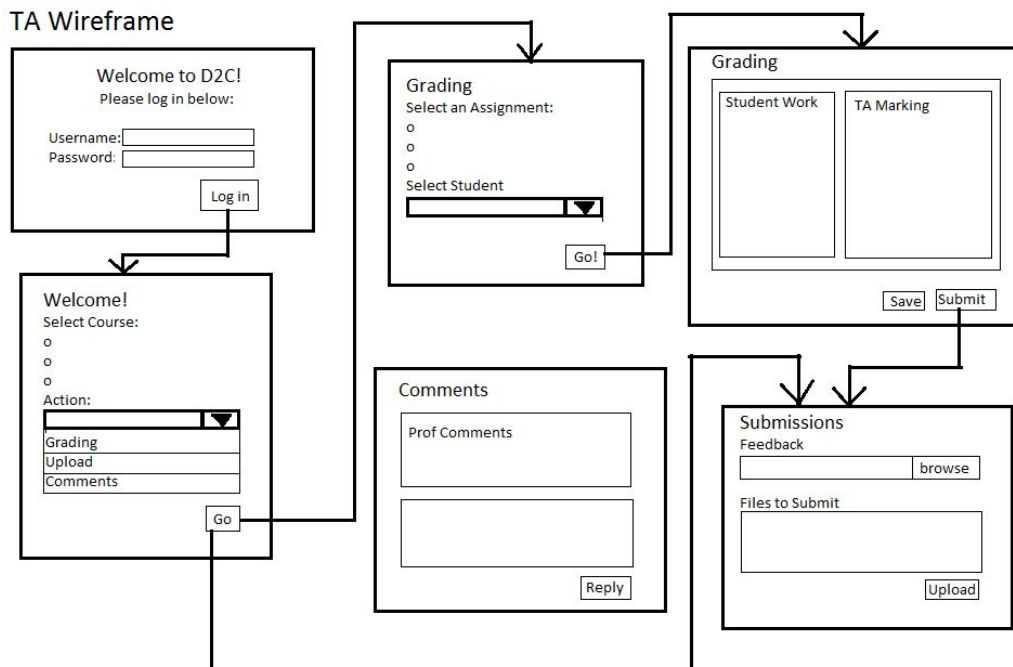


WIREFRAMES

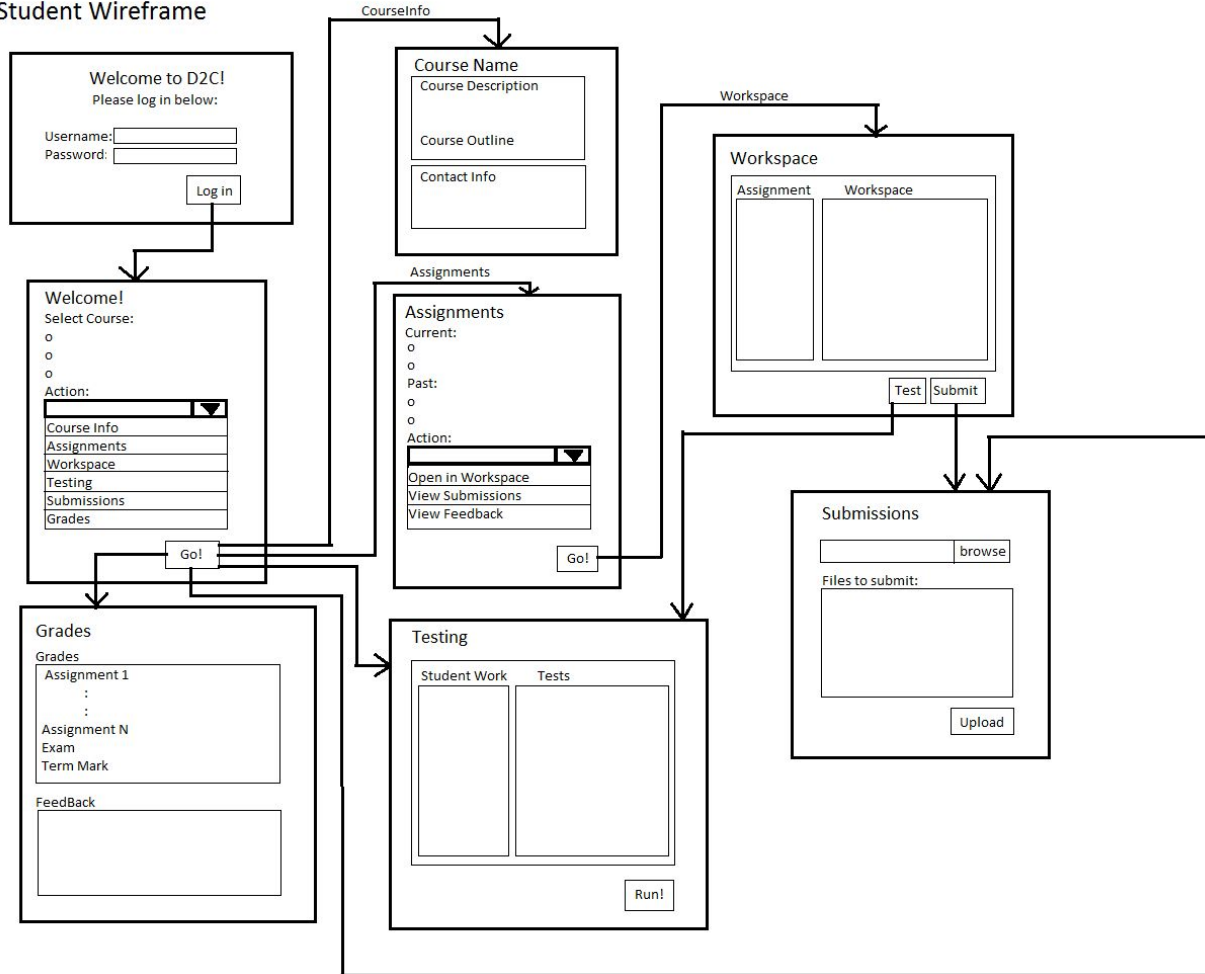
Professor Wireframe



TA Wireframe



Student Wireframe



TEST PLAN

1. Unit test classes (JUnit) to test each server side resource (JAX-RS).
 - a. Tests to verify server side compilation of java/c++
 - b. Tests to verify server side running of java classes
 - c. Tests to verify server side cross-compilation of c++
 - d. Tests to verify server side web resource supplementation
 - e. Tests to verify all prepared SQL statements
 - f. Tests to check prepared SQL statements security
 - g. Tests to check each call to the RESTful API
2. Unit test collections (JUnit) to test each client side javascript function.
 - a. Tests to verify each button on the page that calls the RESTful API, creates the appropriate http request
 - b. Tests to verify calls to Database return properly
 - c. Tests to verify correctness of assignment state (open, closed, graded, ect.)
 - d. Tests to insure security of client side application

TECHNOLOGY STACK

Server Side:

- Tomcat/Jersey (RESTful/Java 8)
- MySQL
- Eclipse for java EE developers
- JUnit (java test suite)
- OpenSSL

Client Side:

- AngularJS/D3 (Javascript/Typescript)
- CSS3/HTML5
- QUnit
- Basic Authentication

WORKPLAN

				Time Estimates			
ID	Activity	Predecessor	Person Assigned	Opt(O)	Normal(M)	Pessimistic(P)	Expected Time
1	Software Modeling	-		1	2	4	3
2	Specification of Internal Interfaces	1		1	2	4	3
3	Database Schemata	1		0.5	1	3	2
4	Build Database	3		1	4	7	4
5	End-to-end Functionality	2,4		0.25	0.5	1	1
6	User Accounts	5		1	2	3	2
7	System Editor	5		1	3	4	3
8	Testing Suite	5		2	5	7	5
9	Creating Assignments	5		1	3	3.5	3
10	Grading System	5		1	2	4	3
11	Polish Product	all		3	5	7	5
						Total Work Days	34 per feature