

Requirements Specification and Use Cases

ProofChecker - SE691

Rakhfa Amin, Thoman Andrews, Kersley Jatto, Aiasha Sattar, Colton Shoenberger

System Requirements:

1. The system shall operate as a web-based platform.
2. The system will have admin, instructors, and students as users.
3. The system shall allow each user to log into a unique account.
4. The system shall be capable of storing user account data.
5. The system shall be capable of storing proofs created by users.
6. The system shall be capable of storing problems created by users.
7. The system shall be capable of storing problem solutions.
8. The system shall be capable of storing assignments created by users.
9. The system shall be capable of storing assignment grades.
10. The system shall enable the testing of natural deduction.
11. The system shall send a verification email to the user upon registering.
12. The system shall allow users to report bugs and issues to the admin/developers.
13. The system shall allow users to send feedback to the admin/developers.
14. The system shall allow users to attach documents when reporting bugs and feedback.

Natural Deduction Tool Requirements:

15. The Natural Deduction Tool (i.e. "ProofChecker", herein "the tool") shall allow users to create proofs.
16. The tool shall allow users to edit proofs.
17. The tool shall allow users to save proofs.
18. The tool shall allow users to delete proofs.
19. The tool shall allow users to provide premises with which to begin a proof.
20. The tool shall allow users to provide a desired conclusion with which to end a proof.
21. The tool shall allow users to add new lines to a proof.
22. The tool shall allow users to add subproofs within a proof.
23. The tool shall allow users to conclude subproofs.
24. The tool shall allow users to delete a line within a proof.
25. The tool shall allow users to delete a subproof within a proof.
26. The tool shall allow users to edit lines within a proof
27. The tool shall allow users to edit subproofs within a proof
28. The tool shall number lines within a proof
29. The tool shall recognize the rules of Truth-Functional Logic (TFL)
30. The tool shall allow users to reference the rules of TFL to justify lines within a proof
31. The tool shall recognize the rules of First Order Logic (FOL)
32. The tool shall allow users to reference the rules of FOL to justify lines within a proof
33. The tool shall allow users to cite line numbers when justifying a line within a proof

34. The tool shall allow users to cite subproofs when justifying a line within a proof
35. The tool shall allow users to select what rules are permissible to reference within a proof
36. The tool shall be extensible such that additional rules from other forms of mathematical reasoning can be created and utilized within proofs
37. The tool shall be able to validate inputted expressions for proper formatting.
38. The tool shall be able to validate inputted lines for correctness.
39. The tool shall be able to validate a proof for correctness
40. The tool shall be able to validate a proof for completeness
41. The tool shall be able to display meaningful feedback to assist users with correcting errors.

User Operations - Student Requirements:

42. The system shall allow students to view assignments that have been assigned to them
43. The system shall allow students to view problems within an assignment
44. The system shall allow students to navigate between problems within an assignment
45. The system shall allow students to utilize the Natural Deduction Tool to complete problems
46. The tool shall provide static hints to students.
47. The tool shall provide dynamic hints to students.
48. The tool shall provide information for students to reference while working.
49. The tool shall allow students to check their inputted solutions for correctness.
50. The tool shall allow students to check their inputted solutions for completeness.
51. The tool shall allow students to save work in progress.
52. The tool shall allow students to submit an assignment for grading.
53. The tool shall be capable of automatically grade submitted work.

User Operations - Instructor Requirements:

54. The system shall allow instructors to create a course.
55. The system shall allow instructors to delete a course.
56. The system shall allow instructors to add students to a course.
57. The system shall allow instructors to remove students from a course.
58. The system shall allow instructors to create problems.
59. The system shall allow instructors to create assignments consisting of problems.
60. The system shall allow instructors to specify what rules students can use when answering a problem.
61. The system shall allow instructors to view assignments submitted by students.
62. The system shall allow instructors to view grades from submitted assignments.
63. The system shall allow instructors to upload assignments.
64. The system shall allow instructors to export assignments.
65. The system shall allow instructors to export grades.
66. The system shall allow instructors to edit grades.

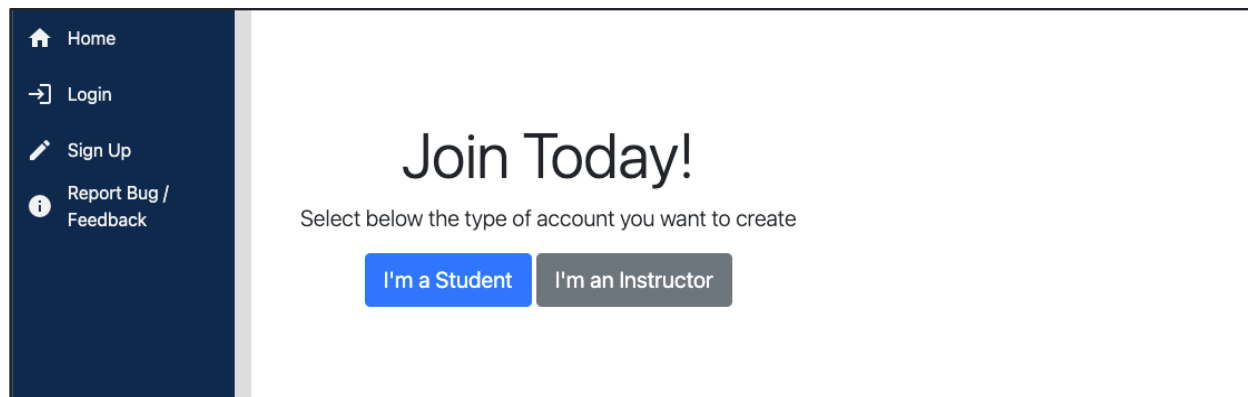
Account Requirements:

- 67. The system shall allow users to create an account with a username and password
- 68. The system shall allow users to log into their account with their username and password
- 69. The system shall be able to distinguish between accounts of students and accounts of instructors
- 70. The system shall allow users to view their forgotten username via email.
- 71. The system shall allow users to change their password via email.
- 72. The system shall allow users to have a user profile.
- 73. The system shall allow users to upload a photo for their user profile.
- 74. The system shall allow users to have a bio for their user profile.
- 75. The system shall allow users to edit their user profile.

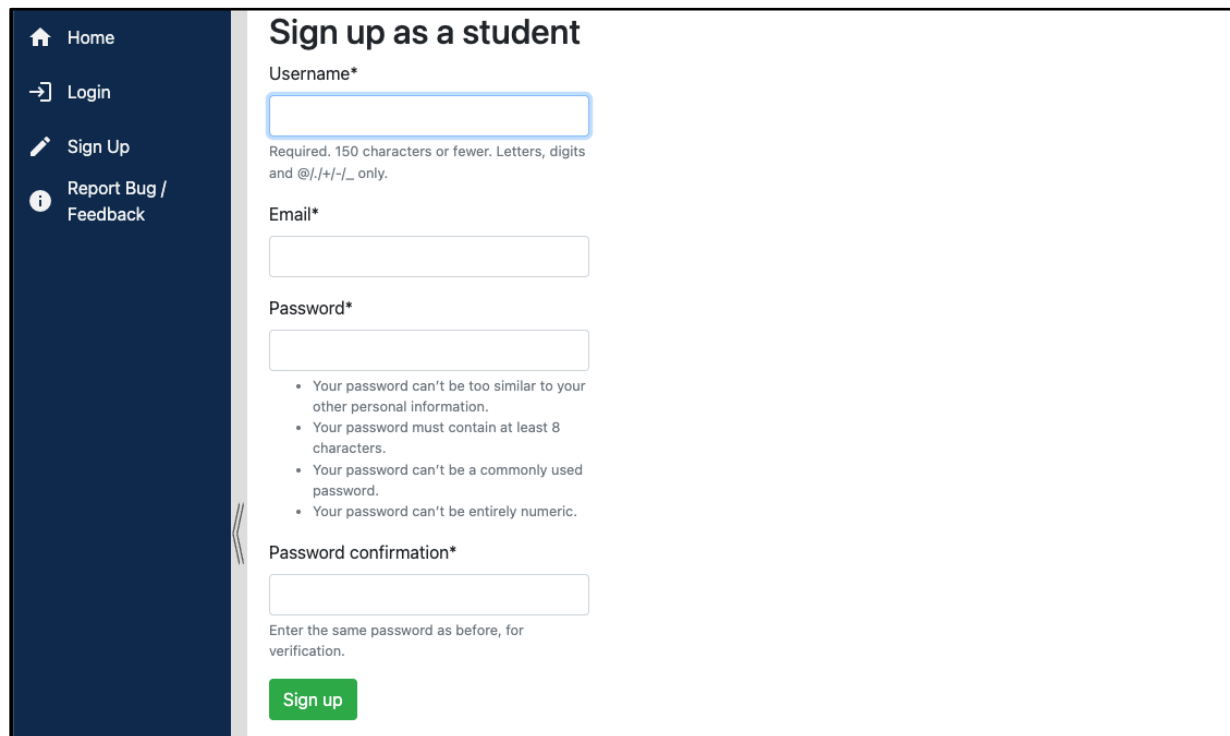
Use Cases

Use Case 1	Create an Account
Actors	Guests
Brief Description	This use case explains how a user creates an account and register as either a student or an instructor.
Basic Flow of Events	<p>Basic flow begins when a guest user visits the web-based proof checker tool. Side Menu bar should be displayed.</p> <ol style="list-style-type: none">1. Guest User clicks on the 'Signup' link on the menu bar to access the sign up page.2. Sign up landing page prompts the guest to sign up as a student or as an instructor.3. Guest user clicks on respective roles.4. Sign up form is displayed, and the guest user enters required information in the fields and clicks on the sign-up button.5. The system validates the information that the guest user has entered.6. User information is stored in the user's account and the system sends out a verification email to the email address provided.7. The system notifies the user that a verification email has been sent.8. User account is activated once he/she confirms registration by clicking on the link provided in the email.9. This ends the use case.
Alternative Flow of events	<p>During the sign-up process, if the guest user enters invalid information, the following occurs:</p> <ol style="list-style-type: none">1. The system described which information was valid.2. The system gives the user suggestions on entering valid data.3. User re-enters the information and the system verifies them again.4. If the user information fails the validation again, the alternate use case for invalid user information occurs again. This continues until the user enters valid information.
Pre-conditions	N/A

Post-condition(s)	Success message 'Account created for (username), check mail to activate the account' is displayed.
-------------------	--



Sign up page



Student Sign up Form

Use Case 2	A User Logs In
Actors	Students, Instructors and Admins

Brief Description	This use case describes how a user logs into their account.
Basic Flow of Events	<p>The basic flow starts when the user arrives at the proof tool's main home page.</p> <ol style="list-style-type: none"> 1. User clicks the Login field on the left bar 2. User fills in their username and password into the respective fields 3. User clicks the Login button 4. The user is signed into their account
Alternative Flow of Events	<p>When the user attempts login and enters invalid login information:</p> <ol style="list-style-type: none"> 1. The system will detail the issue with the user's provided credentials. 2. If the user has an account, they correct the login errors and proceed with signing in according to the basic flow of events. 3. If the user does not have an account, the user decides to create an account by clicking the Sign Up which redirects them to the Sign Up page. 4. Once there the user creates an account and then proceeds to sign in after registering.
Pre-conditions	The user has previously created an account.
Post-condition(s)	User is successfully signed in.

Login Page

Use Case 3	Creating a Proof
Actors	Users
Brief Description	This use case explains how a user creates a proof in the system
Basic flow of events	<p>Basic flow begins when need is realized to create a proof that the user would revisit later.</p> <ol style="list-style-type: none"> 1. User logs into the web-based proof checker system. The side nav bar is displayed. 2. User clicks on the 'My proofs' link from the side nav bar. 3. A new page loads with 'Add a new proof button' on the top and the user clicks on it. 4. New page loads with 'Name', 'Premises' and 'Conclusion' fields and a drop-down list for 'rules'. User fills out the fields with those arguments and selects the rule from that to be applied within the proof. 5. User clicks on the start proof that displays Line, expression, rule fields to work on the proof. 6. If a user wants to check verification for proof statements, the basic flow of events follows from 'checking proof for validations' use case. 7. At any time if the user wants to save the proof, he/she clicks on the save button. 8. The proof is saved in 'proofs' collections for the user, and this concludes this use case.
Pre-conditions	User account is registered in the system and the user is logged in.
Post-condition(s)	The proof is saved under the 'proofs' collection for each registered user. The new created proof is displayed alongside the previously saved proofs if there are any.

Home

Proofs

Logout

Create Proof

Name

Homework question

Rules

TFL - Basic Rules Only

Premises

$A \wedge B$

Conclusion

A

[Click here to understand what each button does!](#)

Line #	Expression	Rule					
1	$A \wedge B$	Premise	+	-	→	↕	×
2	A	$\wedge E$ 1	+	-	→	↕	×

Check Proof







Save

Number of Steps: 2

Download

Add a new proof page

Use Case 4	Checking Mathematical Proof for Validation
Actors	Users
Brief Description	This use case explains how a user validates a mathematical proof

Basic flow of events	<p>Basic flow begins when a user visits the web-based proof checker tool and the home page is displayed.</p> <ol style="list-style-type: none"> 1. User clicks on the 'ProofChecker' link in the homepage. 2. The proof checker page is displayed. There are fields to enter name, premises, conclusion and drop down list to select rules to be referenced. 3. User enters those arguments and clicks the "Start Proof" button. 4. A table with Line no, Expression and Rule columns appears. The line no is automatically generated. For the first row, if there is a premise, the tool also pre-fills expression and rule fields. Users are also provided with following buttons for each row: <ol style="list-style-type: none"> a.  To insert a new line to the proof after the current row. b.  To push the current line into a subproof. c.  To pull the current line out of the subproof. d.  To swap the current row with the one above if they're sequential or pushes current row up into the previous row's level e.  To swap the current row with the one below it if they're sequential or pushes the current row down into the next row's level. f.  To delete the current row. 5. As the user continues to add/delete lines, the line numbers are updated accordingly. 6. Users can check the validation of the proof by clicking on the 'Check Proof' button. 7. The tool checks the validation of each proof line using TFL logic or FOL logic and displays a response to users. If the statement is valid the response 8. If any line is invalid, the system displays error messages that also specify line number and error type. 9. Users re-enters proof statements for validation. 10. If all the lines are valid and proof is complete, this use case concludes.
Pre-conditions	N/A
Post-condition(s)	Success message 'The proof is valid and complete!' is displayed.

Name: Homework question
 Rules: TFL - Basic Rules Only
 Premises: $A \vee B; A \rightarrow C; B \rightarrow C$
 Conclusion: C

[Click here to understand what each button does!](#)

Line #	Expression	Rule
1	$A \vee B$	Premise
2	$A \rightarrow C$	Premise
3	$B \rightarrow C$	Premise
4.1	A	Assumption
4.2	C	$\rightarrow E$ 2

Number of Steps: 5

Result: Error on line 4.2: Line numbers are not specified correctly. Conditional Elimination (Modus Ponens): $\rightarrow E$ m, n

ProofChecker tool with error message

Name: Homework question
 Rules: TFL - Basic Rules Only
 Premises: $A \vee B; A \rightarrow C; B \rightarrow C$
 Conclusion: C

[Click here to understand what each button does!](#)

Line #	Expression	Rule
1	$A \vee B$	Premise
2	$A \rightarrow C$	Premise
3	$B \rightarrow C$	Premise
4.1	A	Assumption
4.2	C	$\rightarrow E$ 2, 4.1
5.1	B	Assumption
5.2	C	$\rightarrow E$ 3, 5.1
6	C	$\vee E$ 1,4,5

Number of Steps: 8

Result: The proof is valid and complete!

ProofChecker tool response after a valid and completed proof

Use Case 5	Viewing Existing Proofs
Actors	Users
Brief Description	This use case explains how a user access the proofs he/she previously saved
Basic flow of events	<p>Basic flow begins when need is realized to access a proof user had previously saved.</p> <ol style="list-style-type: none"> 1. User logs into the web-based proof checker system. The side nav bar is displayed. 2. User clicks on the 'My proofs' link from the side nav bar. 3. A new page loads with all the proofs he has previously saved. 4. This ends the use case
Pre-conditions	User account is registered in the system and the user has previously stored proof while logged in.

Proofs Collection:

- homework 1.3**
 - Premise(s): $A \wedge B$
 - Conclusion: A
 - Lines: 1
- New Proof**
 - Premise(s): $A \wedge C, B \wedge X$
 - Conclusion: $(A \wedge B) \vee Q$
 - Lines: 5
- New Proof**
 - Premise(s): None
 - Conclusion: $\neg C$
 - Lines: 2
- hw 1**
 - Premise(s): $(A \wedge B) \rightarrow C$
 - Conclusion: $A \rightarrow B \rightarrow C$
 - Lines: 3

'Proofs' collection for an instructor account

Use Case 6	Editing Existing Proofs
Actors	Users
Brief Description	This use case explains how a user edits a proof in the system
Basic flow of events	<ol style="list-style-type: none"> 1. The user signs into the Proof Tool 2. The user clicks Proofs which will direct them to a page displaying all their existing proofs 3. The user clicks the Edit Button on the proof they want to adjust. 4. The user is brought to a page where the proof they selected has all its fields displayed 5. The user makes adjustments to the values in one or more fields in the proof; or, the user adds or removes lines to the proof 6. The user completes their changes and hits the save button 7. The updates to the proof are stored in the database
Pre-conditions	The user already has proofs saved to their account.
Post-condition(s)	Upon clicking save, the page will update the user on the proof's status on whether it's complete or still has issues.

[Home](#)
[Profile](#)
[Courses](#)
[Assignments](#)
[Proofs](#)
[Proofs By Students](#)
[Logout](#)

Edit Proof

Name
Rules
Premises
Conclusion

Restart Proof

[Click here to understand what each button does!](#)

Line #	Expression	Rule	
1	<input type="text" value="A∧C"/>	<input type="text" value="Premise"/>	<input type="button" value="+"/> <input type="button" value="←"/> <input type="button" value="→"/> <input type="button" value="↕"/> <input type="button" value="×"/>
2	<input type="text" value="B∧X"/>	<input type="text" value="Premise"/>	<input type="button" value="+"/> <input type="button" value="←"/> <input type="button" value="→"/> <input type="button" value="↕"/> <input type="button" value="×"/>
3	<input type="text"/>	<input type="text"/>	<input type="button" value="+"/> <input type="button" value="←"/> <input type="button" value="→"/> <input type="button" value="↕"/> <input type="button" value="×"/>

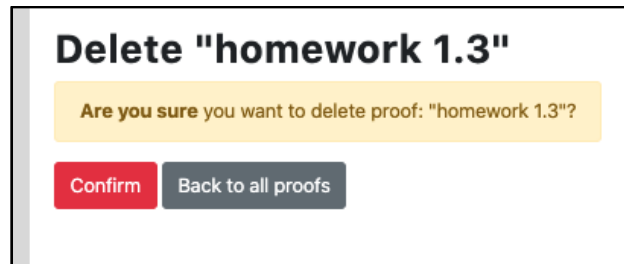
Check Proof Save

Number of Steps: 3

Proof opened in edit mode

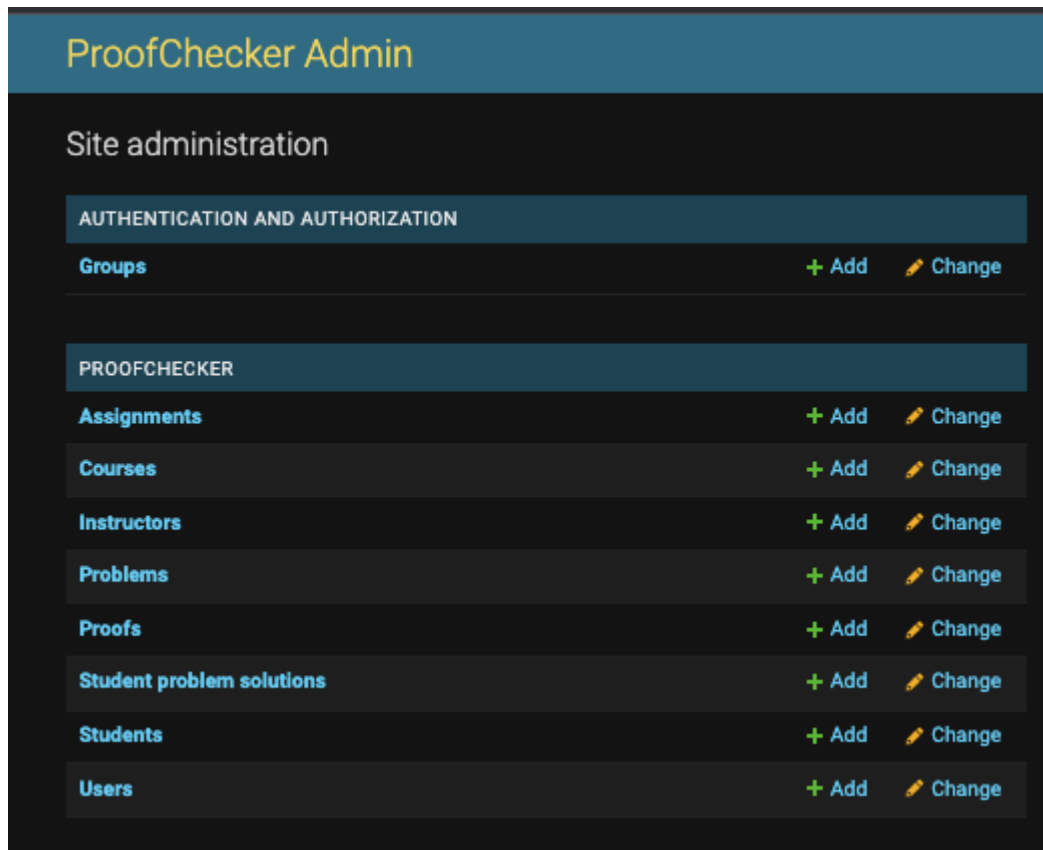
Use Case 7	Deleting Existing Proofs
Actors	Users
Brief Description	This use case explains how a user deletes a proof from the system
Basic flow of events	<p>Basic flow begins when the user has a need for removing a proof from the system.</p> <ol style="list-style-type: none"> 1. User logs into the web-based proof checker system. The side nav bar is displayed. 2. User clicks on the 'My proofs' link from the side nav bar. 3. The user's proofs will be displayed with a Delete button on the lower right of each 4. The user clicks 'Delete' 5. The user is prompted for reassurance that they want to delete their proof 6. The user clicks confirm 7. The proof is deleted from their account
Pre-conditions	The user already has proofs saved to their account.

Post-condition(s)	The proof will no longer appear in the user's proof list.
-------------------	---



The user is prompted to make sure they want to delete their proof

Use Case 8	Managing the Database
Actors	Admins
Brief Description	This use case explains how a user gets the Site Administration page to use various admin functionalities.
Basic Flow of Events	<ol style="list-style-type: none"> 1. The user should open the admin page 2. The user logs in to the admin page with a superuser account 3. The user arrives at Site Administration and can directly add, edit, or remove various fields in the Proof Tool database like Assignments, Problems, Proofs, and Users
Pre-conditions	N/A
Post-condition(s)	The database will be directly changed



A page showing the various admin functions

Use Case 10	Creating a Course
Actors	Instructors
Brief Description	This use case explains how an instructor creates a course.
Basic flow of events	<p>Basic flow begins when an instructor wants to create a new course.</p> <ol style="list-style-type: none"> 1. The instructor signs into the Proof Tool. 2. The instructor clicks Courses on the left menu. 3. The user is brought a page where they can create courses. 4. The instructor clicks the 'Add a new course' button and a page is displayed with a title, term, and section fields and a drop-down menu to select students.. 5. Instructor fills out the fields and selects students to be added to the course from the drop-down menu. 6. Instructor clicks on the save button to create the course.

Pre-conditions	User account is registered as an instructor
Post-condition(s)	Course is now stored in the instructor's 'courses' collection. The newly created course will appear next to any previously created courses.

[Home](#)
[Profile](#)
[Courses](#)
[Assignments](#)
[Proofs](#)
[Proofs By Students](#)
[Logout](#)

Create A Course

Title*

CS270 : Formal Logic

Term*

Winter

Section*

900

Select Students

student2, student3, studi

Save

student1

student2 ✓

student3 ✓

student4 ✓

student5

Add A course page

Use Case 11	Creating an Assignment
Actors	Instructor
Brief Description	This use case explains how an Instructor can create an assignment for a course

Basic flow of events	<p>Basic flow begins when an instructor wants to create an assignment for a course.</p> <ol style="list-style-type: none"> 1. The instructor signs into the Proof Tool. 2. Instructor clicks on the Assignments link from the left menu. 3. A new page loads that allows them to add assignments. 4. The instructor clicks the 'Add a New Assignment' button which displays a page with title, due by fields and course drop down menu. The page also informs the instructor that assignment has to be saved before adding problems to it. 5. Instructor fills out the fields and selects the course from the drop-down menu. 6. Instructor clicks on the save button 7. Success message appears on top of the page saying an assignment is created successfully. 8. Assignment details page with add problem button is displayed. 9. This concludes the use case for creating an assignment.
Pre-conditions	User account is registered as an instructor
Post-condition(s)	<p>An assignment is created and an instructor can add problems to it.</p> <p>The instructor's students will be able to view its details.</p>

Home

Profile

Courses

Assignments

Proofs

Proofs By Students

Logout

Create An Assignment

Please save the assignment before adding Problems

Title

HomeWork 1

Course

CS 270 : Formal Logic

Due by

03/14/2022

Save

Assignment creation page

Use Case 12	Adding a problem to an assignment
Actors	Instructor
Brief Description	This use case explains how an Instructor can add problems to an assignment he has created.
Basic flow of events	<p>Basic flow begins when an instructor wants to add problems to an assignment he created.</p> <ol style="list-style-type: none"> 1. The instructor signs into the Proof Tool. 2. The basic flow of events follows from creating an assignment use case. 3. Assignment details page displays a button to add problems to it. 4. The instructor clicks the 'Add problemt' button which loads a page with fields to enter Question, point, target steps, rules to be used, premises and conclusion. Instructor fills out the fields.

	<p>6. Instructor clicks on save button.</p> <p>7. Instructor is redirected to the assignment details page with a success message saying 'problem saved successfully' on top of the page.</p> <p>8.</p>
Alternative Flow of Events	<p>The need of this use cases is realized When the instructor wants to create a problem that is partially completed and the students have to fill out empty proof lines to solve them:</p> <ol style="list-style-type: none"> 5. The basic flow of events for adding a problem follows from step 1-4 . 6. If the instructor wants to create a problem that would be partially solved, He/She begins the proof by clicking start proof. 7. Instructors can use the available buttons to create a partially completed proof. 8. Instructor clicks on save to add the problem to the assignment.
Pre-conditions	Instructor has created an assignment.
Post-condition(s)	Instructor is redirected to the assignment details page where other problems are also displayed in a list.

Home

Profile

Courses

Assignments

Proofs

Proofs By Students

Logout

Create Problem

Question

Solve the following problem

Point

10

Target steps

7

Rules

TFL - Basic Rules Only

Premises

$A \wedge C, B \wedge X$

Conclusion

$(A \wedge B) \vee Q$

Start Proof

[Click here to understand what each button does!](#)

Line #	Expression	Rule

Save

Back

Problem consists of premises and conclusion

Home

Profile

Courses

Assignments

Proofs

Proofs By Students

Logout

Create Problem

Question

Fill out the rules and complete the proof

Point

10

Target steps

6

Rules

TFL - Basic Rules Only

Premises

$A \wedge C, B \wedge X$

Conclusion

$(A \wedge B) \vee Q$

[Click here to understand what each button does!](#)

Line #	Expression	Rule					
1	$A \wedge C$	Premise	+	←	→	↕	×
2	$B \wedge X$	Premise	+	←	→	↕	×
3	A		+	←	→	↕	×
4	B		+	←	→	↕	×
5	$A \wedge B$		+	←	→	↕	×

Save

Back

Partially completed proof as a problem

ProofChecker Use Case Diagram

