Software Requirements Specification for ProofBuddy

Team Name	Group 3
Team Members	Nicole Itchon, Raphael Perez, Viet Pham, Iftekhar Rahman
Stakeholders	Steve Earth, Jeremy Johnson
Date Submitted	March 17, 2023

Contents

1. Introduction

1.1. System Purpose

ProofBuddy is an educational tool for teaching computer science students proof techniques and logical reasoning. The system is web-based and designed for use by both instructors and students. The system will leverage its additional functionalities to compete with existing natural deduction tools that do not exist within these tools.

1.2. System Scope

ProofBuddy will have the following capabilities for instructors, create/edit/delete courses, assignments, problems, and proofs, and view proofs by students. The system will have the following capabilities for students, join courses, view/submit assignments, work through problems, and create/edit/delete proofs. The ProofBuddy system requires users to create an account to utilize its functionalities. The system allows all users to report bugs and provide feedback to the team through its interface.

1.3. System Overview

ProofBuddy is a system developed by two different teams of graduate students at Drexel University. At the time of writing, the second team has adopted the initial version of ProofBuddy that was left behind by the first team. While the system will have the same requirements, the goal of the second team is to verify that requirements have been met and repair bugs and crashes that the system has experienced from both users and testing. While there are competitors to the ProofBuddy system, the overarching goal of the team is to sway instructors at not only Drexel but also other institutions to use the system. To do so, ProofBuddy must be intuitive, practical, secure, able to support instructors, able to guide student learning, limited on experiences with bugs and system crashes.

1.3.1. System Context

ProofBuddy implements additional functionalities that sets it aside from its competitors, such as allowing students to be enrolled in a course, allowing instructors to create custom assignments, grading assignments for instructors, and saving proofs. These capabilities support instructor administrative work as well as student learning, while other system function as proof tools.

1.3.2. System Function

ProofBuddy allows for user authentication, course creation and add students to course, assignment and problem creation, and solve proofs and obtain feedback as you solve. Currently, the system is web-based and at the time of writing will require being on Drexel network either by physically being on campus or through Drexel VPN.

1.3.3. User Characteristics

The intended use for ProofBuddy will be for instructors of natural deduction and students enrolled in those courses. Instructors can use ProofBuddy to store course assignments and student grades, while students can use ProofBuddy to complete assignments, request for an extension, and review proofs for upcoming exams.

1.4. Definitions

Natural deduction – technique where logical reasoning is expressed by inference rules closely related to the "natural" way of reasoning
Mathematical proof – inferential argument for a mathematical statement, showing that the stated assumptions logically guarantee the conclusion

2. System Requirements

2.1. Functional Requirements

Req 1	The system shall operate as a web-based platform.
Req 2	The system shall allow users to register for an instructor or student account.
Req 3	The system shall send a verification email to the user upon registering for an account.
Req 4	The system shall allow registration to users with a verified and unique email address.
Req 5	The system shall allow users to login to with their username and password.
Req 6	The system shall store user account data.
Req 7	The system shall allow users to view/edit their user profile.
Req 8	The system shall allow users to change their forgotten password via email.
Req 9	The system shall store courses created by instructors.
Req 10	The system shall associate students and assignments with courses.
Req 11	The system shall store assignments created by instructors.
Req 12	The system shall associate problems with assignments.
Req 13	The system shall store student assignment grades.
Req 14	The system shall store proofs saved by users.
Req 15	The system shall verify that proofs are correct.
Req 16	The system shall provide feedback when proofs contain an error or is incomplete.

2.2. Usability Requirements – Instructors

Req 17	The system shall allow instructors to create/edit/delete courses.
Req 18	The system shall allow instructors to add/remove students to a course
Req 19	The system shall allow instructors to create/edit/delete assignments.

Req 20	The system shall allow instructors to create/edit/delete problems for assignments.
Req 21	The system shall allow instructors to specify what rules students can use when working on a problem.
Req 22	The system shall allow instructors to create/edit/delete proofs.
Req 23	The system shall allow instructors to view student proofs.
Req 24	The system shall allow instructors to view assignment grades of students.
Req 25	The system shall allow instructors to bulk add students to a course.
Req 26	The system shall allow instructors to upload assignments.
Req 27	The system shall allow instructors to export grades.

2.3. Usability Requirements – Students

Req 28	The system shall allow students to join a course.
Req 29	The system shall allow students to view course details.
Req 30	The system shall allow students to view assignments for their enrolled course.
Req 31	The system shall allow students to view assignment details.
Req 32	The system shall allow students to view a problem for an assignment.
Req 33	The system shall allow students to complete a problem for an assignment.
Req 34	The system shall allow students to check their work upon clicking “Check Proof” button on the problem.
Req 35	The system shall allow students to save their work upon clicking “Save” button on the problem.
Req 36	The system shall allow students to submit an assignment for grading.
Req 37	The system shall allow students to request for an extension on an assignment when the due date has passed.
Req 38	The system shall allow students to create/edit/delete proofs.

2.4. System Interface

Req 39	When signing up for an account, the system shall present a form with fields username, email, password, and password confirmation.
Req 40	When logging in, the system shall present a form with fields username and password.
Req 41	When editing user profile, the system shall present a form with fields first name, last name, email address, current image, new image, and bio.
Req 42	When creating/editing a course, the system shall present a form with fields title, term, section, and students.
Req 43	When creating/editing an assignment, the system shall present a form with fields title, course, start date, due date.

Req 44	After creating an assignment, the system shall present an “Add Problem” button.
Req 45	When creating/editing a problem, the system shall present a form with fields question, points, target steps, lost points, rules, premises, and conclusion.
Req 46	When creating/editing a proof, the system shall present a form with fields name, rules, premises, and conclusion.
Req 47	After selecting a student to view their proofs, the system shall present a table of the selected student’s proofs.
Req 48	When submitting a bug or feedback, the system shall present a form with fields name, email, subject, details, and attachment.

2.5. Bug Fix Requirements

Req 49	The system shall compute student assignment scores accurately.
Req 50	The system shall send activation email successfully.
Req 51	The system shall constrain users from indenting a line more than once while working on a proof.
Req 52	The system shall report all mistakes in a proof upon clicking “Check Proof” button.
Req 53	The system shall prevent the cursor from jumping to the end of a line while working on a proof.
Req 54	The system shall present only the specified rules to students while working on a proof.
Req 55	The system shall prevent inserting a comma when a user puts their cursor in the middle of a rule, then enters either the + or x button.
Req 56	The system shall present line numbers as not editable.
Req 57	The system shall prevent the instructor from redirecting to login upon course creation.
Req 58	The system shall present buttons in a consistent order throughout the application.
Req 59	The system shall prevent crashing when the last line of proof is empty upon clicking “Check Proof” button.
Req 60	The system shall prevent crashing when no premises for specified.
Req 61	The system shall prevent crashing when =E is used in a proof.
Req 62	The development team shall incorporate panel dividers in the FOL rules per rule.
Req 63	The system shall allow instructors to set whether resubmissions are allowed on an assignment.
Req 64	The system shall allow instructors to set the time zone or default to 11:59pm of the local time zone.

Req 65	The system shall prevent showing warning about only saving proofs when logged in.
Req 66	The system shall automatically line numbers for sub-proofs accurately.

2.6. New Feature Requirements

Req 67	The system shall allow using a saved proof, only in cases where instructor has indicated that this functionality is allowed, and student has correct proof saved under certain name.
Req 68	The system shall allow proofs to be done for Equational Reasoning.
Req 69	The system shall present a button which inserts a comment line (i.e. the user can type any text they wish, but this line is fully ignored by the parser and does not affect the step count for the proof).
Req 70	The system shall allow the instructor to import a .csv of userids, email_address to bulk create student accounts and register for that course.
Req 71	The system shall allow a “disprove" mode that allows the user to enter in T/F for each variable, and ProofBuddy checks that all the premises evaluate to True, but the conclusion is False.
Req 72	The system shall allow exportation into LaTex, Jupyter.
Req 73	The system shall allow instructors to toggle the visibility of the number of target lines.
Req 74	The system shall allow students to access assignments after the due date.
Req 75	The system shall check premises for being well-formed before starting proof.
Req 76	The system shall autosave the save after every edit rather than requiring a “Save” button.
Req 77	The system shall warn the user if a line never gets used in a proof.
Req 78	The system shall ignore blank lines in proofs.
Req 79	The system shall present the “Toggle Rules” button only to instructors and not students.
Req 80	The development team shall improve testing procedures (corner cases and continuous testing).
Req 81	The development team shall develop a database storage system for student attempts/errors (for later analysis of common approaches/mistakes).
Req 82	The system shall allow the user should be able to click on the line number and it will auto-enter.
Req 83	The system shall allow users to extract a portion of a proof into a lemma.
Req 84	The development team shall improve robustness of assignment feature by implementing auto-grading and duplicating).
Req 85	The system shall allow instructors to set external rules either be (A) allowed unconditionally, or (B) only allowed if students proved it.

Req 86	The system shall allow users to configure counting lines to count all lines in a lemma or not.
Req 87	The system shall allow instructors to customize a grading penalty along with some canned feedback (e.g. if a student uses IP then -3 pts "Indirect Proof not needed") or at least a comment field for grading for some additional personalized feedback.
Req 88	The system shall allow for student resubmissions even before the due date has passed, which should be set by the instructor.
Req 89	The development team shall update the version log with most recent additions and bug fixes.

3. Use Cases

3.1. Case Flow for Users

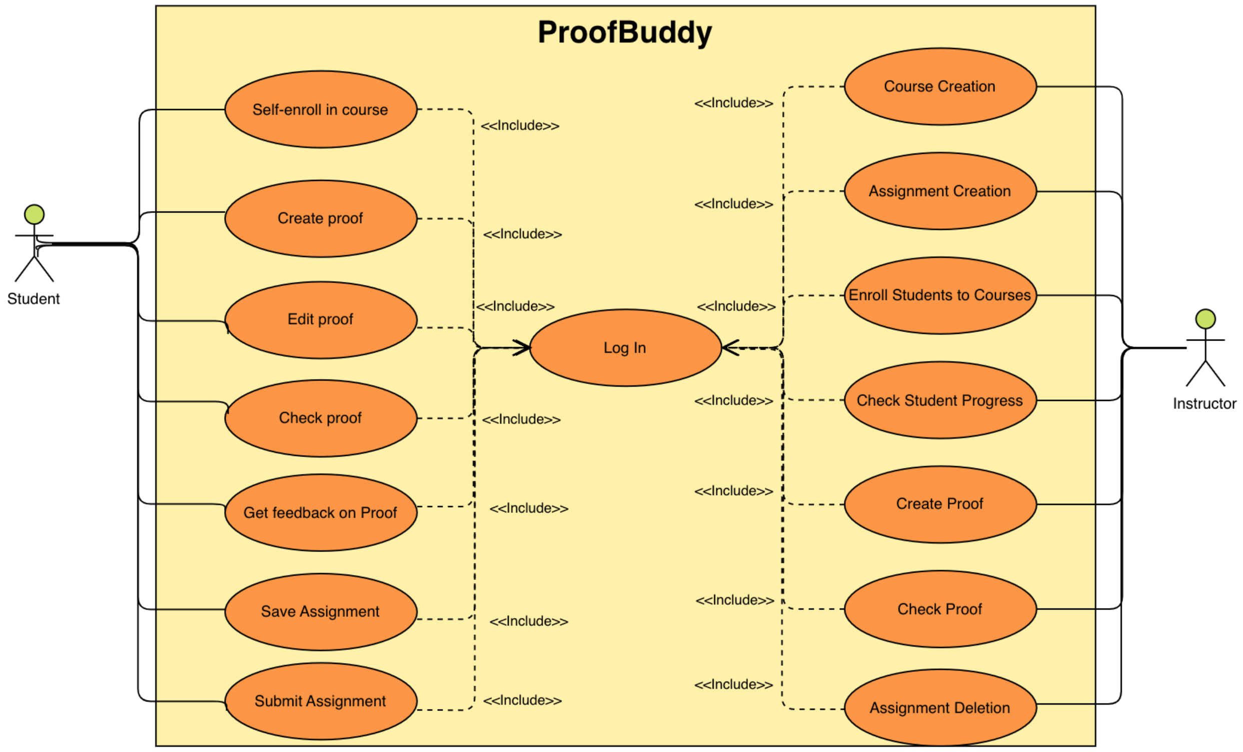
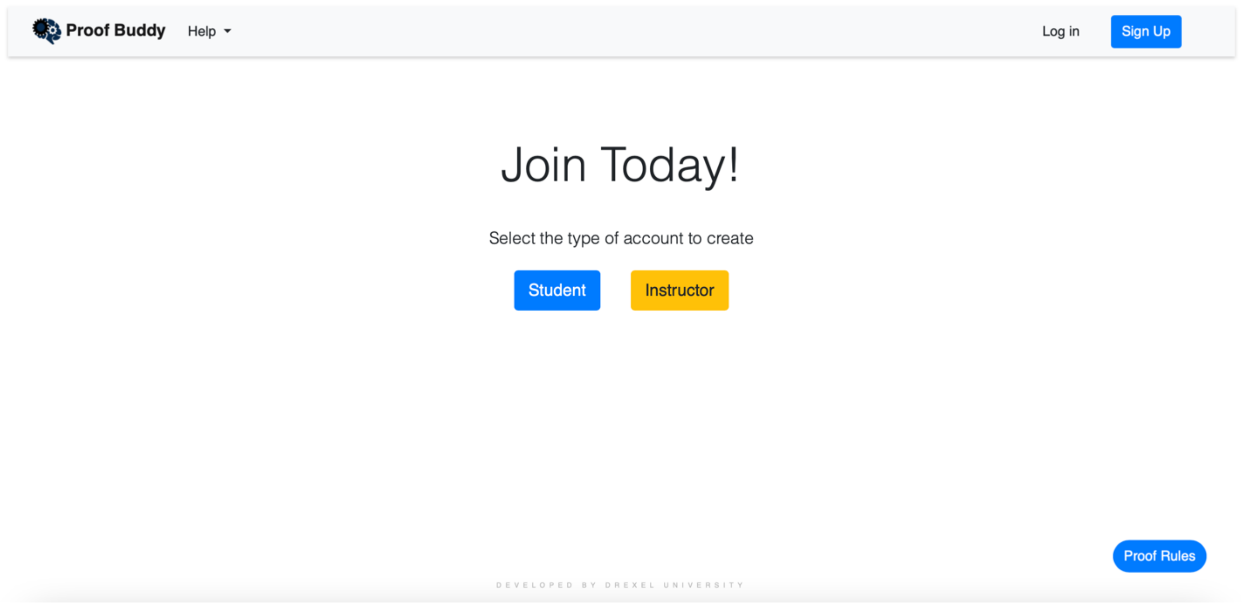
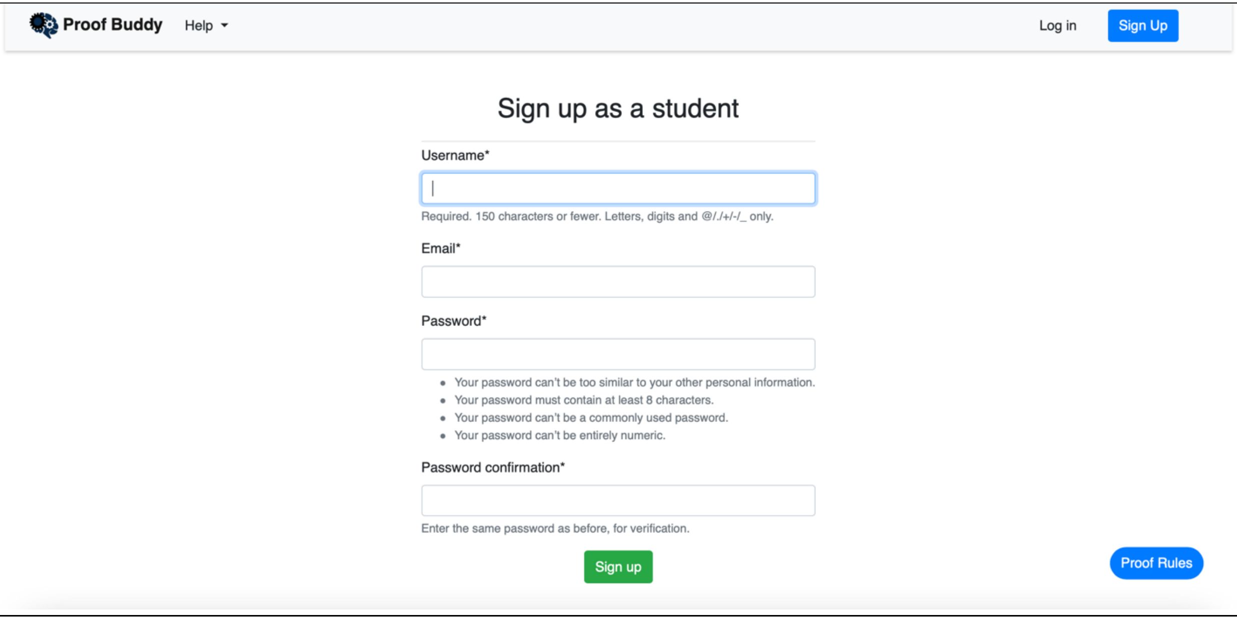


Figure 1: Case Flow for Users

When users visit the ProofBuddy website, users have the option to sign up for an account or log in. When signing up for an account, users have the option to register for a student account or instructor account. When logged in as a student, the user can self-enroll in a course, create proof, edit proof, check proof, get feedback on proof, save assignment, and submit assignment. When logged in as an instructor, the user can create a course, create an assignment, enroll students to courses, check student progress, create proof, edit proof, check proof, and delete assignments. The following use cases walks the reader through the flow of specific events along with pre-conditions, post-conditions, and screenshots.

3.2. Create Account

Use Case 1	Create an account
Actors	Visitors to ProofBuddy website
Description	This use case explains how a user creates an account and registers as either a student or an instructor.
Pre-conditions	N/A
Flow of events	1. User lands on ProofBuddy homepage 2. User clicks on "Sign Up" link on main screen or top navigation bar. 3. User selects type of account to create (student or instructor). 4. The system presents the sign-up form with fields username, email, password, and password confirmation. 5. User fills out form and clicks "Sign Up" button. 6. The system verifies that the form is complete, username and email are unique to the system and that password match. 7. The system notifies user that "Account created for [username]. Check Mail to activate the account". 8. User confirms registration by clicking the link provided in the email. 9. User is directed to login screen and the system notifies user that "Account activated for [username]".
Post-conditions	System message "Account activated for [username]".
Screenshots	 Proof Buddy Help Log in Sign Up Student Instructor Proof Rules DEVELOPED BY DREXEL UNIVERSITY Join Today! Select the type of account to create

Proof Buddy Help Log in Sign Up

Sign up as a student

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

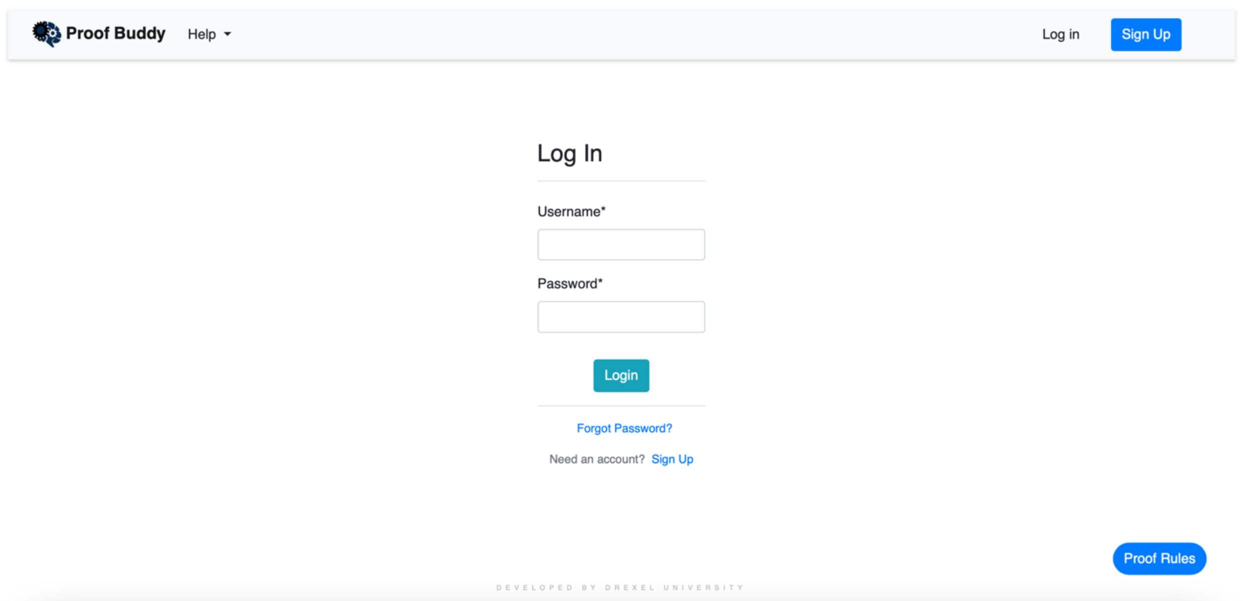
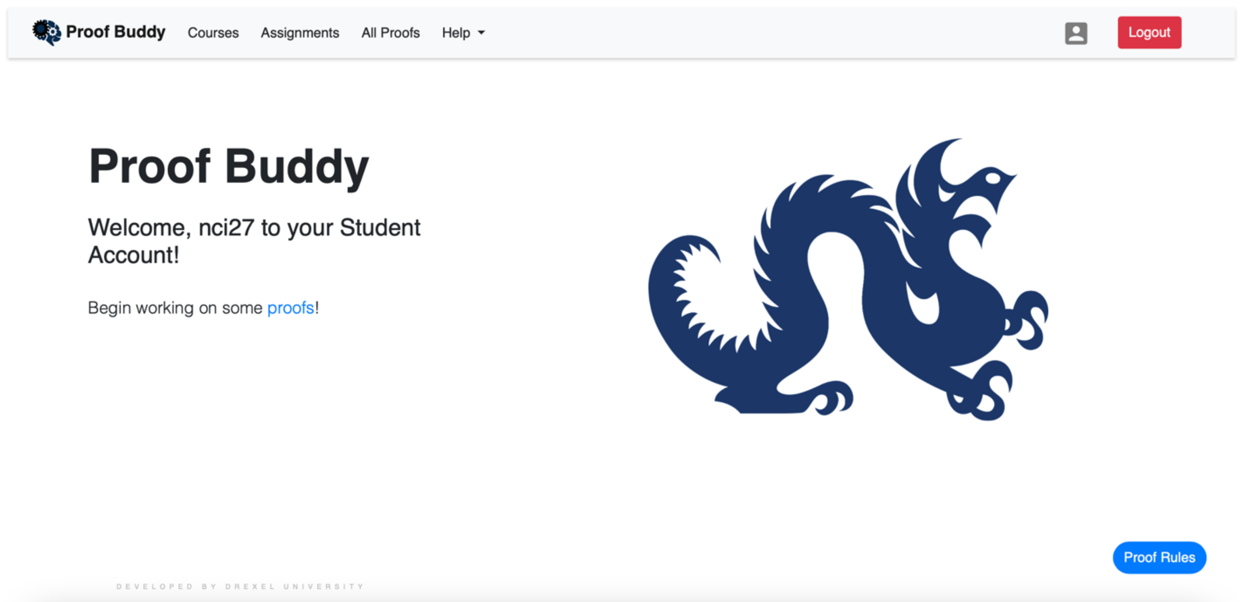
Enter the same password as before, for verification.

Sign up

Proof Rules

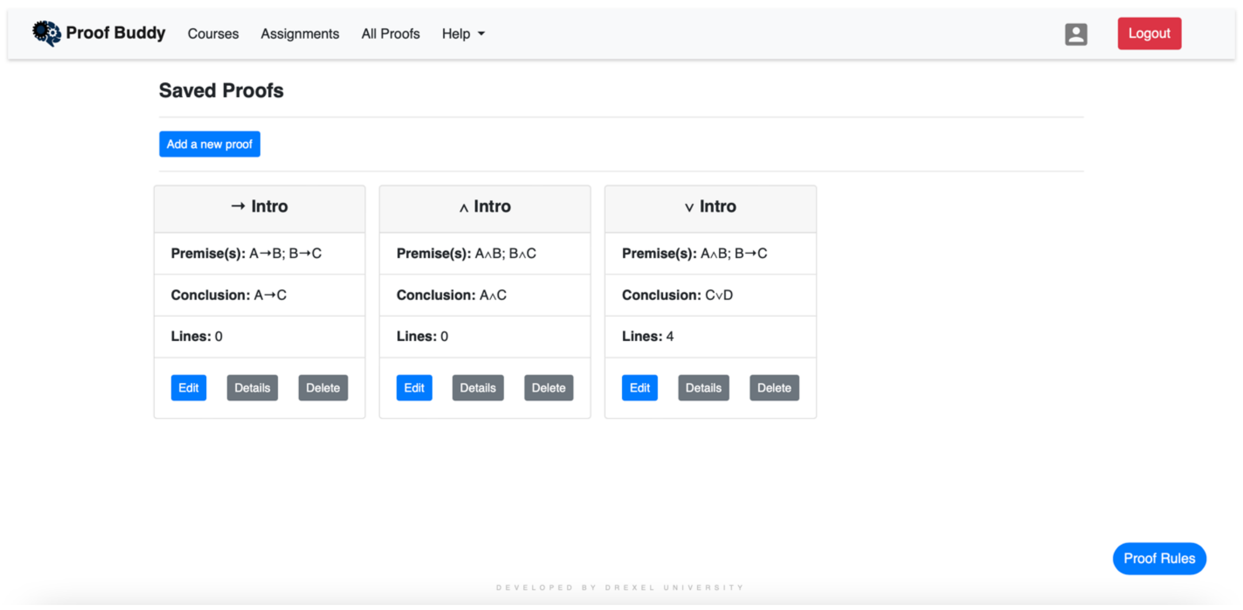
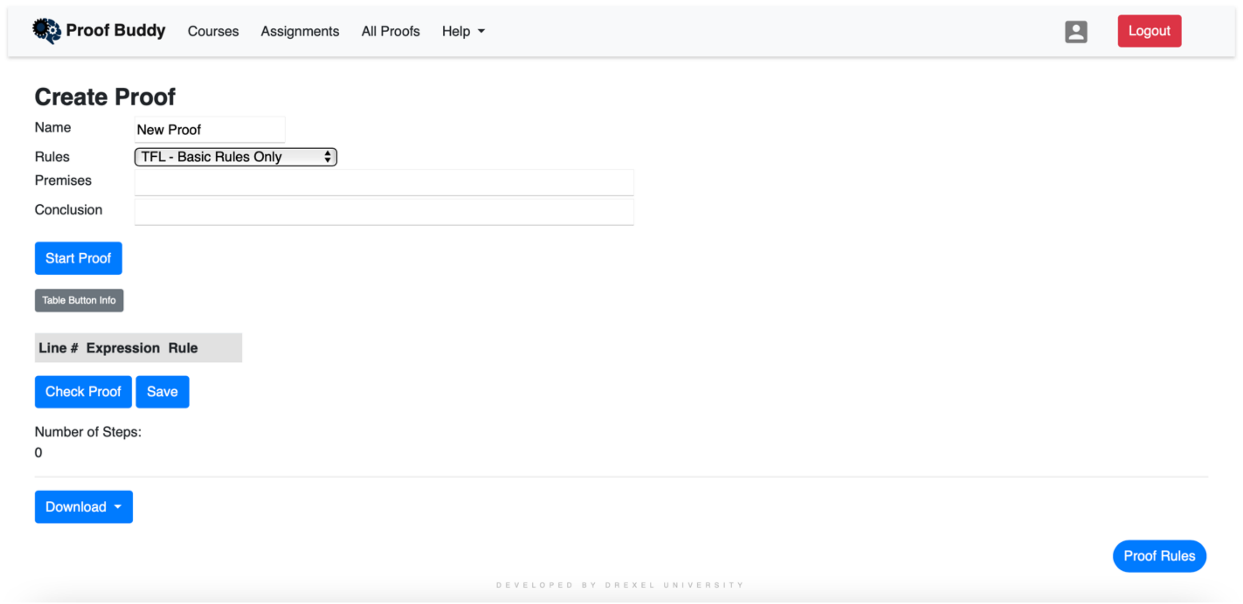
3.3. Login

Use Case 2	Login to website
Actors	Students, Instructors
Description	This use case describes how a user logs into their account.
Pre-conditions	The user has previously created an account.
Flow of events	1. The system presents the login form with fields username and password. 2. User enters username and password credentials and clicks "Login" button. 3. User is logged in and directed to ProofBuddy personalized homepage.
Post-conditions	User is able to login to ProofBuddy with username and password.

Screenshots	 Proof Buddy Help Log in Sign Up Log In Username* Password* Login Forgot Password? Need an account? Sign Up Proof Rules DEVELOPED BY DREXEL UNIVERSITY Proof Buddy Courses Assignments All Proofs Help Logout Proof Buddy Welcome, nci27 to your Student Account! Begin working on some proofs! Proof Rules DEVELOPED BY DREXEL UNIVERSITY

3.4. Create Proof

Use Case 3	Create a proof
Actors	Students, Instructors
Description	This use case describes how a user creates a proof.
Pre-conditions	The user has a registered account and is logged in to ProofBuddy.
Flow of events	1. User is logged in and directed to ProofBuddy personalized homepage. 2. User clicks “All Proofs” tab on top navigation bar. 3. User clicks “Add a new proof” button. 4. The system presents the create proof form with fields name, rules, premises, and conclusion.

	5. User fills out form and clicks one of two buttons: a. "Start Proof" b. "Save" 6. User can click "Start Proof" button to display table of lines, expressions, and rules and begin solving the proof. 7. User can click "Save" button to work on the proof at a later time. 8. The system saves the proof in "All Proofs" tab upon clicking "Save" button.
Post-conditions	The new proof is displayed alongside previously saved proofs if there are any.
Screenshots	 

Saved Proofs

Add a new proof

→ Intro	∧ Intro	∨ Intro
Premise(s): A→B; B→C	**Premise(s):** A∧B; B∧C	**Premise(s):** A∧B; B→C
Conclusion: A→C	**Conclusion:** A∧C	**Conclusion:** C∨D
Lines: 0	**Lines:** 0	**Lines:** 4
Edit Details Delete	Edit Details Delete	Edit Details Delete

Create Proof

Name: New Proof
Rules: TFL - Basic Rules Only
Premises:
Conclusion:

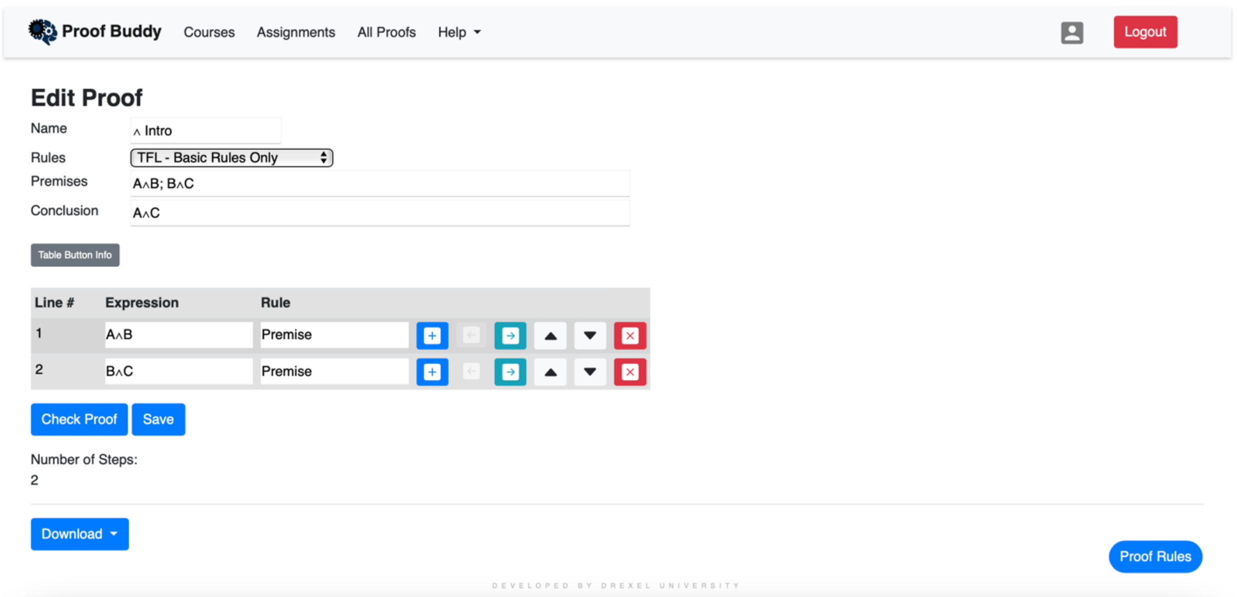
Line #	Expression	Rule

Number of Steps:
0

3.5. Validate Proof

Use Case 4	Validate a proof
Actors	Students, Instructors
Description	This use case describes how a user validates a proof.
Pre-conditions	The user has a registered account and is logged in to ProofBuddy. The user has a previously saved and unsolved proof and has navigated to this proof.
Flow of events	1. User clicks “Start Proof” button to display table of lines, expressions, and rules with premises filled in if there are any. 2. User can click + to insert a new line to proof. 3. User can click ← to push the line into a sub proof. 4. User can click → to pull the line out of the sub proof. 5. User can click ▲ to swap the current row with the one above if they’re sequential or pushes current row up into the previous row’s level. 6. User can click ▼ 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. 7. User can click ✕ to delete the line. 8. The system updates the “Number of Steps” count as the user adds or deletes lines in the proof. 9. User clicks “Check Proof” button to receive feedback from the system. 10. The system checks the validation of each proof line using TFL logic or FOL logic and displays a response to users. 11. The system displays error messages that also specify line number and error type if any line is invalid. 12. The system displays success message if all the lines are valid and proof is complete.
Post-conditions	System message “The proof is valid and complete!”.

Screenshots



Edit Proof

Name: ∧ Intro

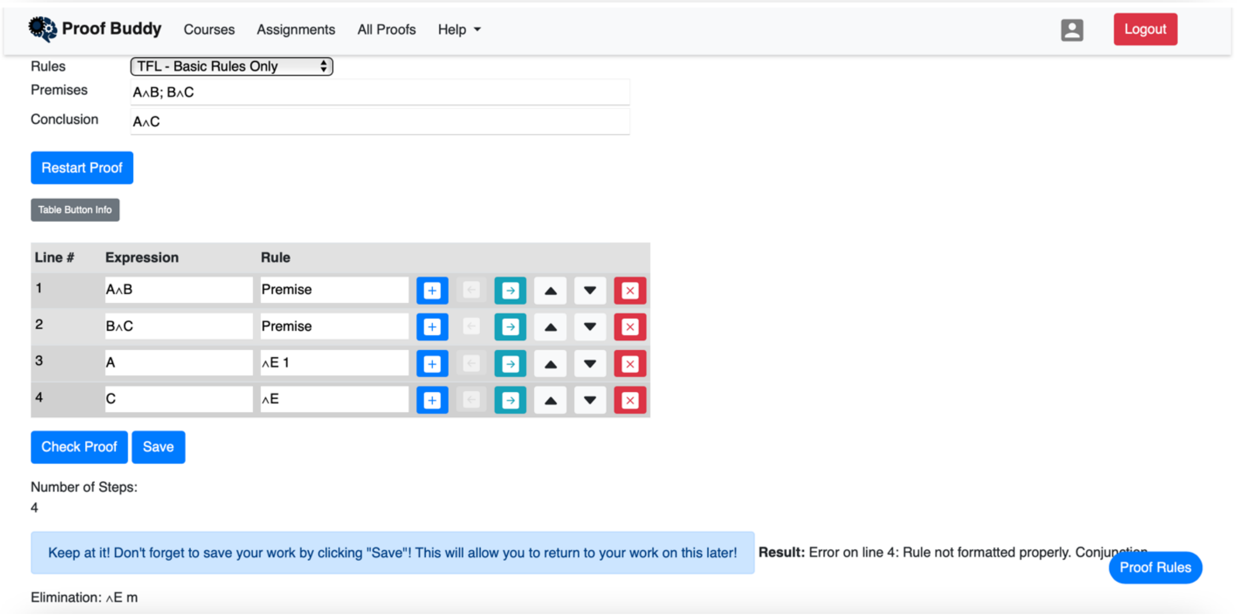
Rules: TFL - Basic Rules Only

Premises: A∧B; B∧C

Conclusion: A∧C

Line #	Expression	Rule
1	A∧B	Premise
2	B∧C	Premise

Number of Steps:
2



Rules: TFL - Basic Rules Only

Premises: A∧B; B∧C

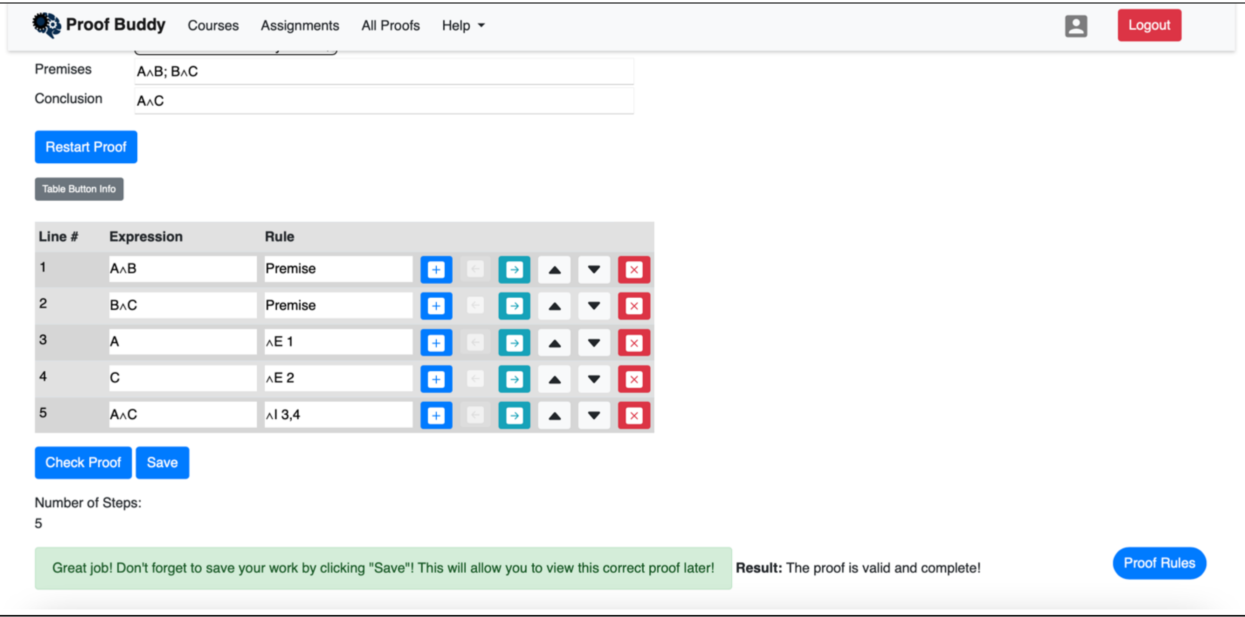
Conclusion: A∧C

Line #	Expression	Rule
1	A∧B	Premise
2	B∧C	Premise
3	A	∧E 1
4	C	∧E

Number of Steps:
4

Keep at it! Don't forget to save your work by clicking "Save"! This will allow you to return to your work on this later!

Result: Error on line 4: Rule not formatted properly. Conjunction Elimination: ∧E m

Premises A∧B; B∧C

Conclusion A∧C

Line #	Expression	Rule
1	A∧B	Premise
2	B∧C	Premise
3	A	∧E 1
4	C	∧E 2
5	A∧C	∧I 3,4

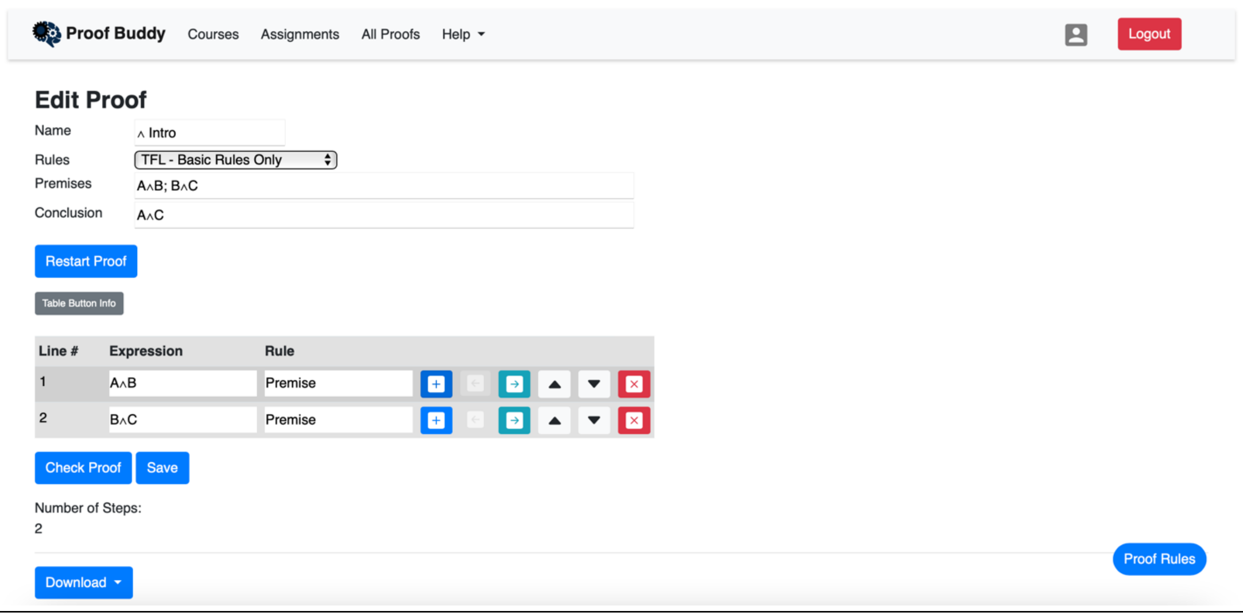
Number of Steps:
5

Great job! Don't forget to save your work by clicking "Save"! This will allow you to view this correct proof later!

Result: The proof is valid and complete!

3.6. Edit Proof

Use Case 5	Edit a proof
Actors	Students, Instructors
Description	This use case describes how a user edits a proof.
Pre-conditions	The user has a registered account and is logged in to ProofBuddy. The user has a previously saved proof.
Flow of events	1. User clicks “All Proofs” tab on top navigation bar. 2. User clicks “Edit” button on a proof they want to edit. 3. The system presents the edit proof form with prepopulated fields name, rules, premises, and conclusion. 4. User edits form and clicks one of two buttons: a. “Restart Proof” b. “Save” 5. User can click “Restart Proof” button to display table of lines, expressions, and rules and begin solving the proof. 6. User can click “Save” button to complete their changes and/or work on the proof at a later time. 7. The system updates the proof in “All Proofs” tab upon clicking “Save” button.
Post-conditions	The updated proof is displayed alongside previously saved proofs if there are any.

Screenshots	

Edit Proof

Name ∧ Intro

Rules TFL - Basic Rules Only

Premises A∧B; B∧C

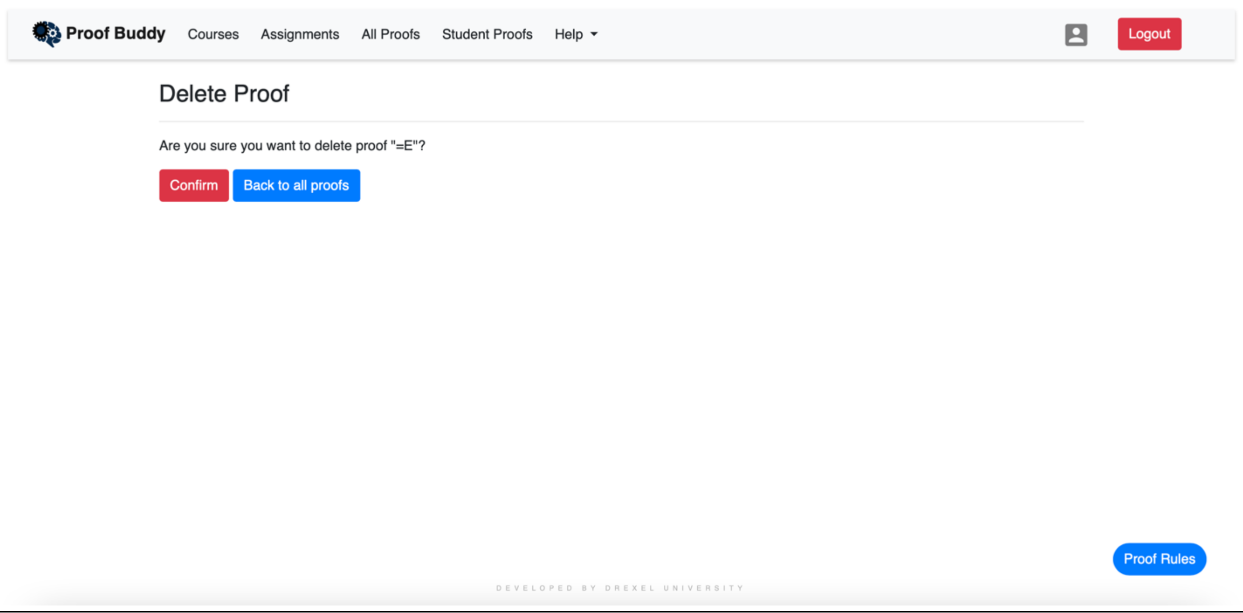
Conclusion A∧C

Line #	Expression	Rule
1	A∧B	Premise
2	B∧C	Premise

Number of Steps:
2

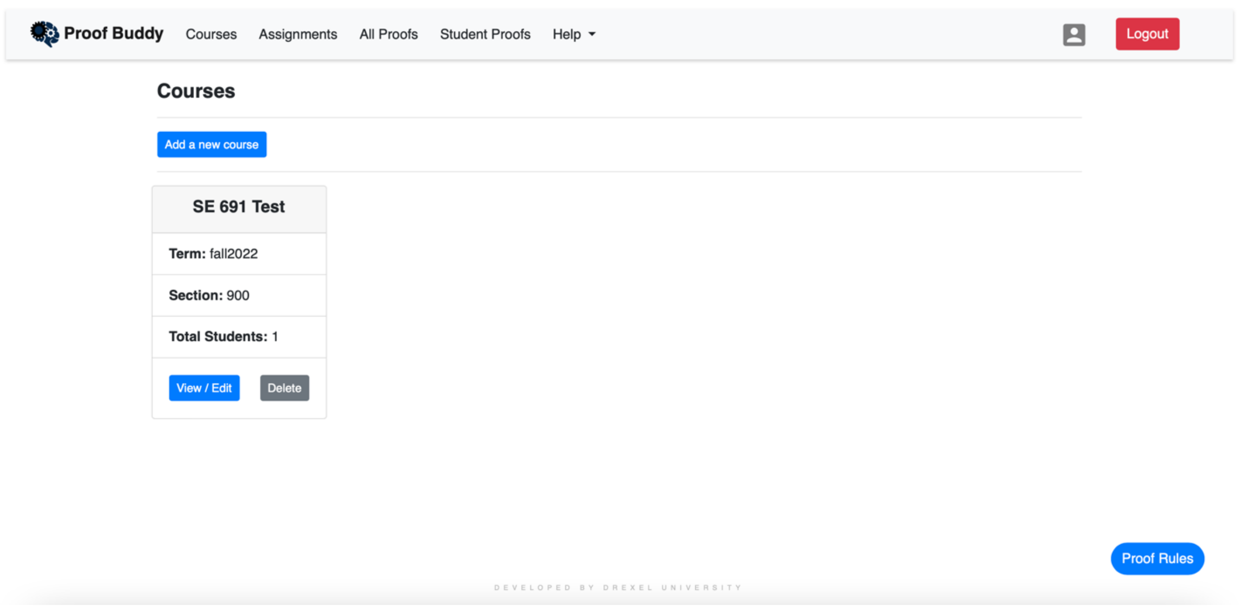
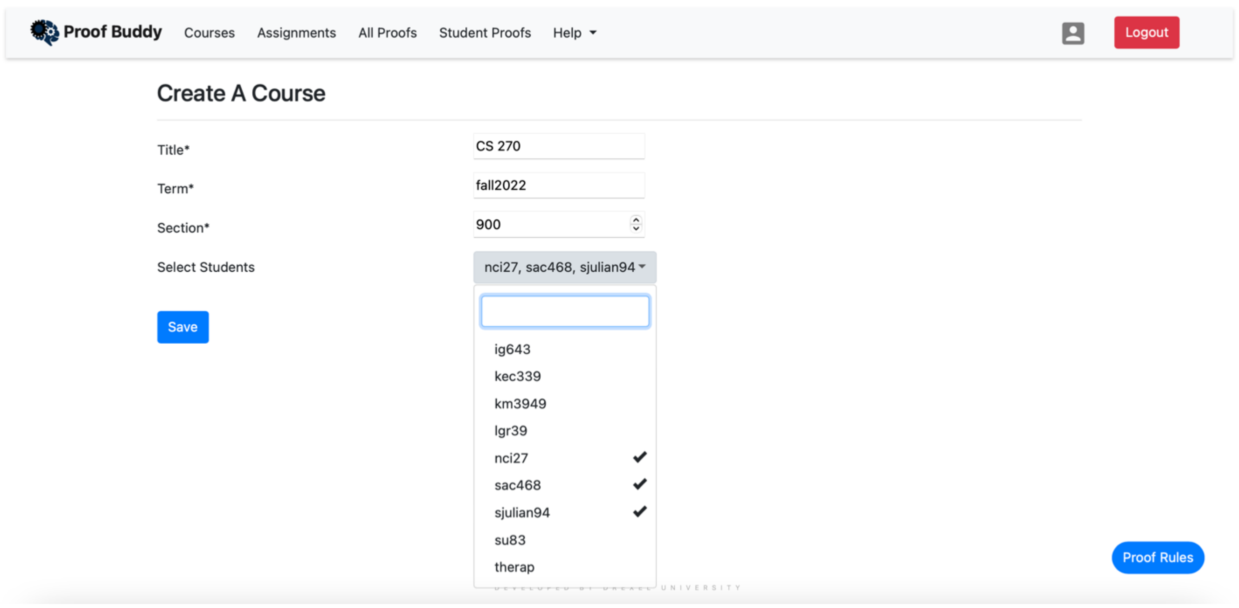
3.7. Delete Proof

Use Case 6	Delete a proof
Actors	Students, Instructors
Description	This use case describes how a user deletes a proof.
Pre-conditions	The user has a registered account and is logged in to ProofBuddy. The user has a previously saved proof.
Flow of events	1. User clicks “All Proofs” tab on top navigation bar. 2. User clicks “Delete” button on a proof they want to delete. 3. The system presents “Are you sure you want to delete proof” message to user. 4. User clicks one of two buttons: a. “Confirm” b. “Back to all proofs” 5. User can click “Confirm” button to delete proof. 6. User can click “Back to all proofs” button to cancel deletion and return to all proofs page.
Post-conditions	The proof no longer appears in the user’s saved proofs page.

Screenshots	 Proof Buddy Courses Assignments All Proofs Student Proofs Help Logout Delete Proof Are you sure you want to delete proof "=E"? Confirm Back to all proofs Proof Rules DEVELOPED BY DREXEL UNIVERSITY

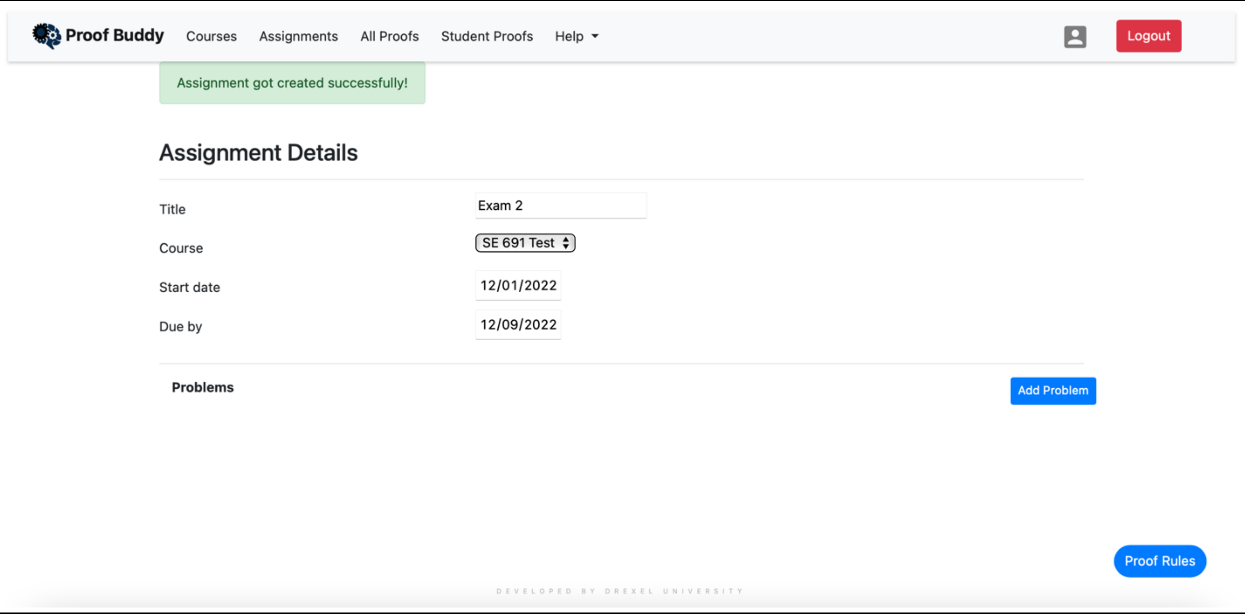
3.8. Create Course

Use Case 7	Create a course
Actors	Instructors
Description	This use case describes how an instructor creates a course.
Pre-conditions	The user has a registered instructor account and is logged in to ProofBuddy.
Flow of events	1. User clicks "Courses" tab on top navigation bar. 2. User clicks "Add a new course" button. 3. The system presents the create a course form with fields title, term, section, and select students. 4. User fills out form and clicks "Save" button to create the course. 5. The system saves the course in "Courses" tab upon clicking "Save" button.
Post-conditions	The new course is displayed alongside previously saved courses if there are any.

Screenshots	 Proof Buddy Courses Assignments All Proofs Student Proofs Help Logout Courses Add a new course View / Edit Delete Proof Rules DEVELOPED BY DREXEL UNIVERSITY SE 691 Test **Term:** fall2022 **Section:** 900 **Total Students:** 1  Proof Buddy Courses Assignments All Proofs Student Proofs Help Logout Create A Course Save Proof Rules Title* CS 270 Term* fall2022 Section* 900 Select Students nci27, sac468, sjulian94 ig643 kec339 km3949 lgr39 nci27 ✓ sac468 ✓ sjulian94 ✓ su83 therap

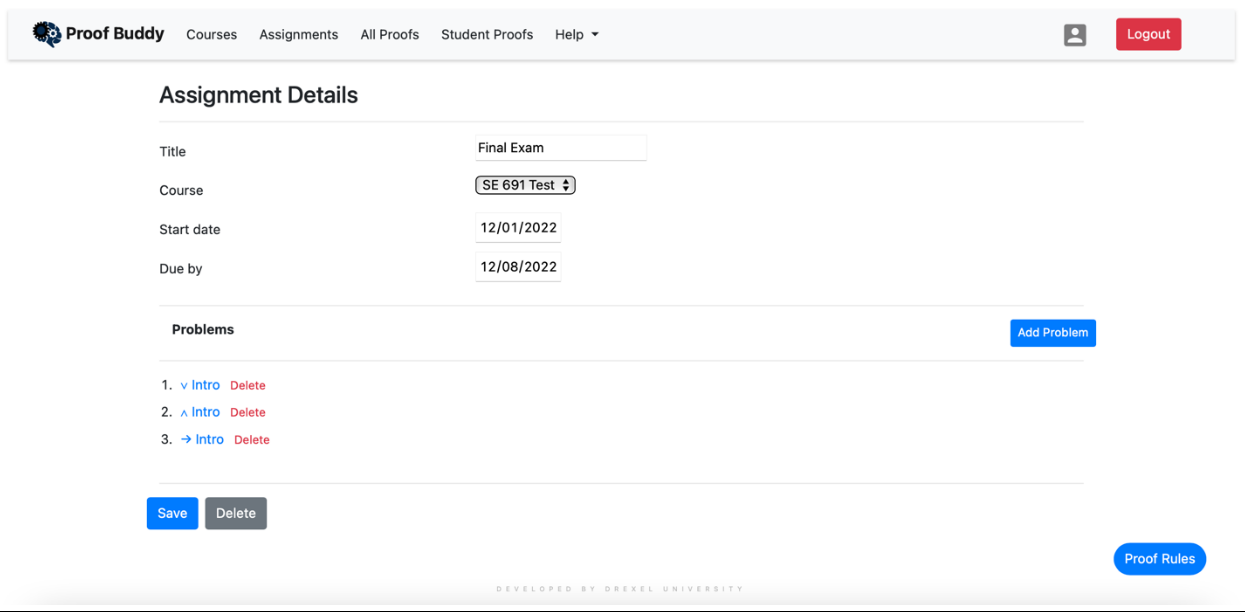
3.9. Create Assignment

Use Case 8	Create an assignment
Actors	Instructors
Description	This use case describes how an instructor creates an assignment.
Pre-conditions	The user has a registered instructor account and is logged in to ProofBuddy.
Flow of events	1. User clicks "Assignments" tab on top navigation bar. 2. User clicks "Add a new assignment" button. 3. The system presents the create an assignment form with fields title, course, start date, and due date.

	4. User fills out form and clicks “Save” button to create the assignment. 5. The system saves the assignment in “Assignments” tab upon clicking “Save” button. 6. The system presents success message “Assignment got created successfully!” and allows the instructor to start adding problems to assignment.
Post-conditions	The new assignment is displayed alongside previously saved assignments if there are any and the instructor can add problems to assignment. The student’s enrolled in course will be able to view assignment details.
Screenshots	 Proof Buddy Courses Assignments All Proofs Student Proofs Help Logout Add Problem Proof Rules DEVELOPED BY DREXEL UNIVERSITY Assignment got created successfully! Assignment Details Title: Exam 2 Course: SE 691 Test Start date: 12/01/2022 Due by: 12/09/2022 Problems

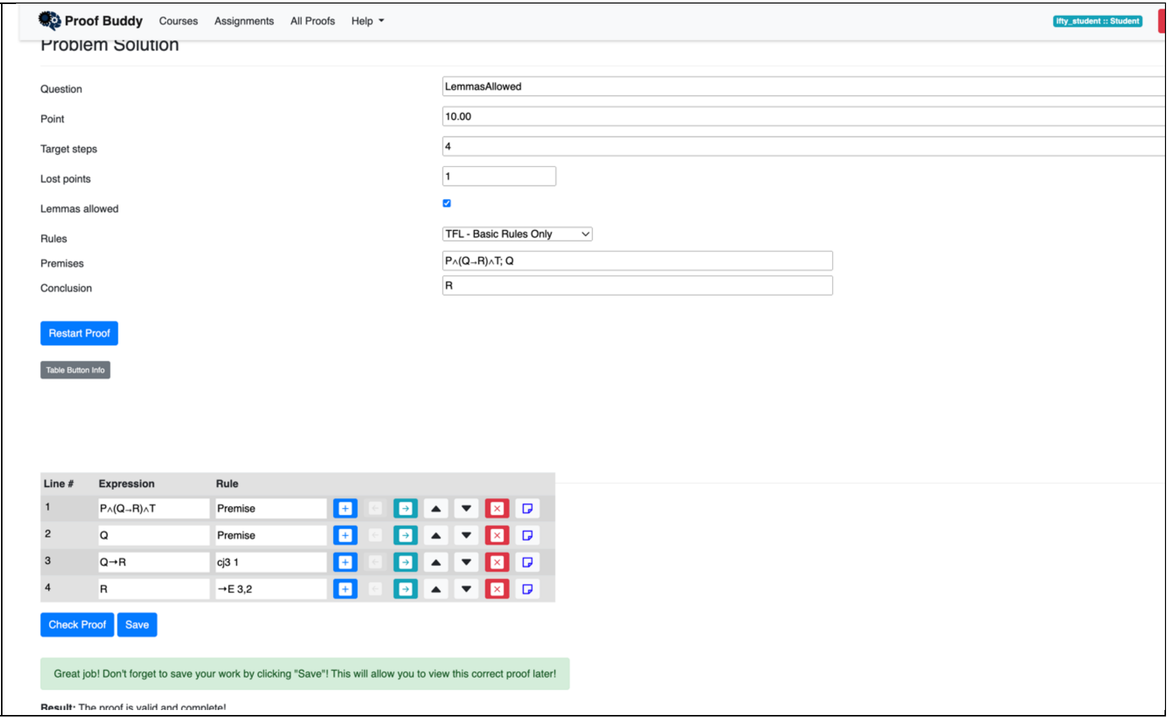
3.10. Add Problem to Assignment

Use Case 9	Add problem to assignment
Actors	Instructors
Description	This use case describes how an instructor adds a problem to an assignment.
Pre-conditions	The user has a registered instructor account and is logged in to ProofBuddy. The user has a previously saved assignment and has navigated to this assignment.
Flow of events	1. User clicks “Add Problem” button. 2. The system presents the create problem form with fields question, point, target steps, lost points, rules, premises and conclusion. 3. User fills out form and clicks “Save” button to create the problem. 4. The system saves the problem to the specific assignment upon clicking “Save” button. 5. The system presents success message “Problem saved successfully!”.
Post-conditions	Instructor is director to assignment details page and the new problem is displayed alongside previously saved problems if there are any.

Screenshots	 Proof Buddy Courses Assignments All Proofs Student Proofs Help Logout Assignment Details Title: Final Exam Course: SE 691 Test Start date: 12/01/2022 Due by: 12/08/2022 Problems Add Problem 1. ∨ Intro Delete 2. ∧ Intro Delete 3. → Intro Delete Save Delete Proof Rules DEVELOPED BY DREXEL UNIVERSITY

3.11 Using a Lemma in a Proof

Use Case 10	Using a lemma in a proof
Actors	Instructors, Students
Description	This use case describes how an instructor or student can use a lemma in a proof. The proof can be one done stand alone or as part of a problem on an assignment.
Pre-conditions	The user has the lemma's proof correctly proven and saved in their account.
Flow of events	1. The user solves a proof correctly and confirms this using the check proof button. 2. The user saves the proof to their account using the save proof button. 3. The user uses the saved proof as a lemma in a subsequent proof. 4. The user gets the appropriate results message when checking the main proof.
Post-conditions	Proof is either correct or incorrect (lemma was used correctly). The lemma was not valid due to the specified reason.

Screenshots	

Problem Solution

Question	LemmasAllowed
Point	10.00
Target steps	4
Lost points	1
Lemmas allowed	✓
Rules	TFL - Basic Rules Only
Premises	P∧(Q→R)∧T; Q
Conclusion	R

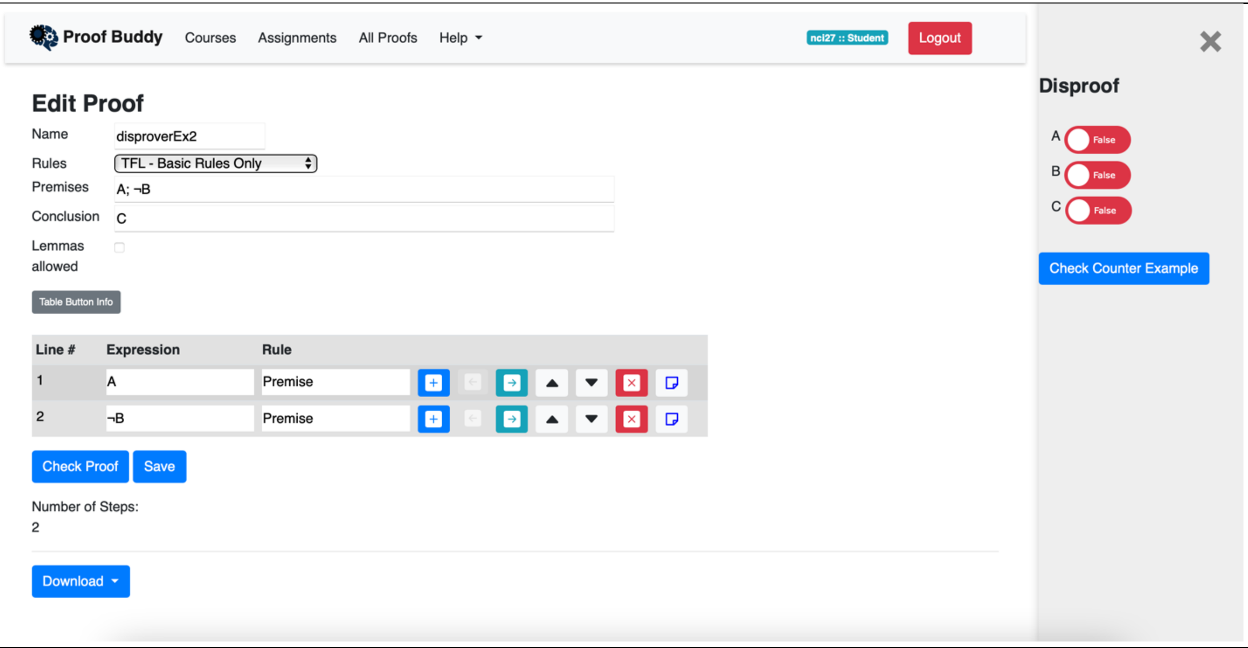
Line #	Expression	Rule
1	P∧(Q→R)∧T	Premise
2	Q	Premise
3	Q→R	cj3 1
4	R	→E 3,2

Great job! Don't forget to save your work by clicking "Save"! This will allow you to view this correct proof later!

Result: The proof is valid and complete!

3.12 Disprove a Proof

Use Case 11	Disprove a proof
Actors	Instructors, Students
Description	This use case describes how a user can use the disprove functionality of ProofBuddy.
Pre-conditions	The user has a registered account and is logged in to ProofBuddy. The user has a previously saved proof and has navigated to this proof.
Flow of events	1. User clicks "Start Counter Example" button. 2. The system presents the disproof form with toggle switches for each variable in the premise and conclusion all initially evaluating to false. 3. User toggles each switch for premise variables to evaluate to true and conclusion variables to evaluate to false. 4. User clicks "Check Counter Example" button.
Post-conditions	The system presents feedback to the user on whether of counter example is valid or not as well as provide which premise variables were not satisfied.

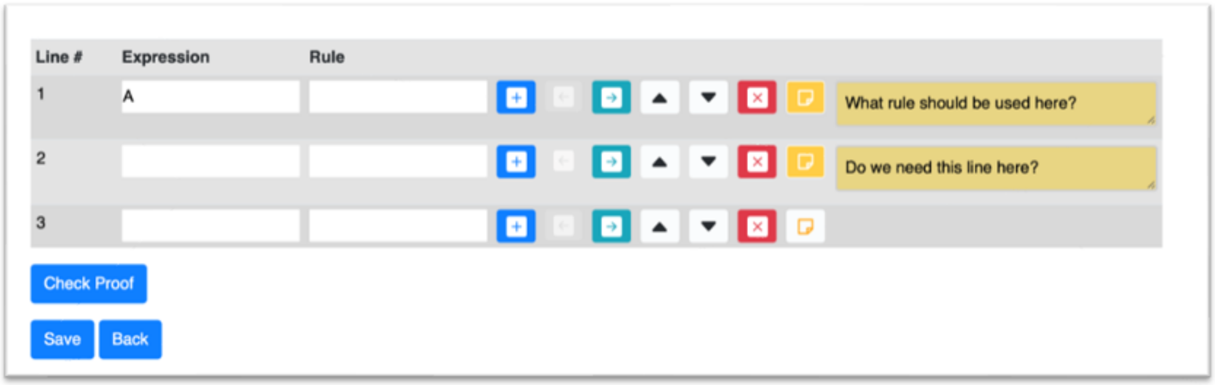
Screenshots	 Edit Proof Name: disproverEx2 Rules: TFL - Basic Rules Only Premises: A; ¬B Conclusion: C Lemmas allowed Line # | Expression | Rule 1 | A | Premise 2 | ¬B | Premise Number of Steps: 2 Disproof A False B False C False

3.13 Using Comments/Responses in a Proof

Use Case 12	Using comments/responses in a proof
Actors	Instructors, Students
Description	This use case describes how an instructor or students can use comment/response text box as a mean of communication or note-taking.
Pre-conditions	The instructor or student has a registered account and is logged in to ProofBuddy.
Flow of events	1. Instructor creates a problem. 2. Instructor clicks “Start Proof” to start creating proof lines for the problem 3. On each proof line, the instructor clicks the yellow comment button to create a text box on the same line. 4. Instructor adds instructions to the text boxes. 5. Once finished, the instructor hits “Save.” 6. A student logs into their account and open the assignment 7. They click on the problem and click “Start Proof.” 8. They see the incomplete proof with instructions on each line 9. They finish the proof using the instructions and hit “Submit”
Post-conditions	1. Instructor and student can see each other’s instructions and responses. 2. Comments and responses don’t affect the validity of the proof when “Check Proof” is clicked

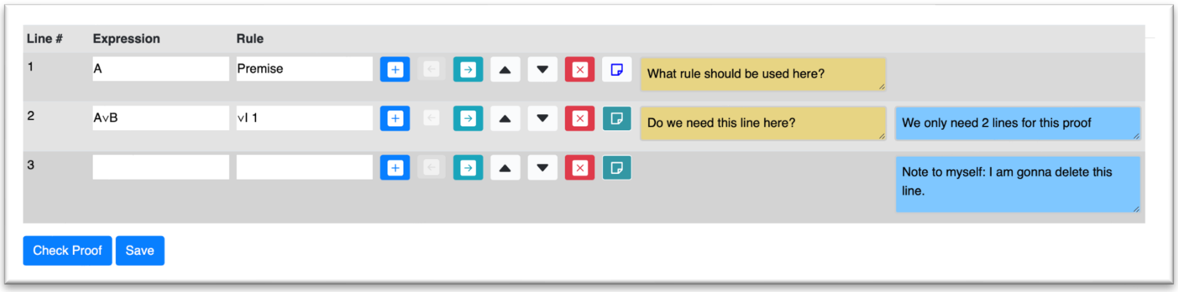
Screenshots

Instructor's View:



Line #	Expression	Rule	
1	A		What rule should be used here?
2			Do we need this line here?
3			

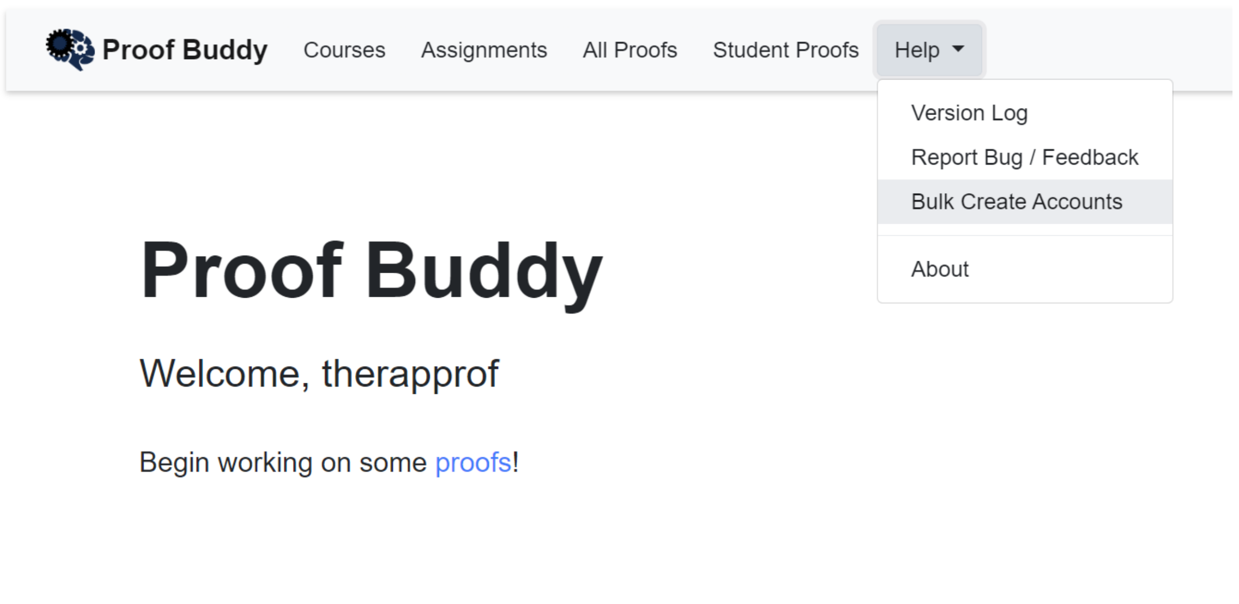
Student's View:



Line #	Expression	Rule		
1	A	Premise	What rule should be used here?	
2	AvB	vI 1	Do we need this line here?	We only need 2 lines for this proof
3				Note to myself: I am gonna delete this line.

3.14 Bulk Create Accounts by Importing CSV of Email Addresses

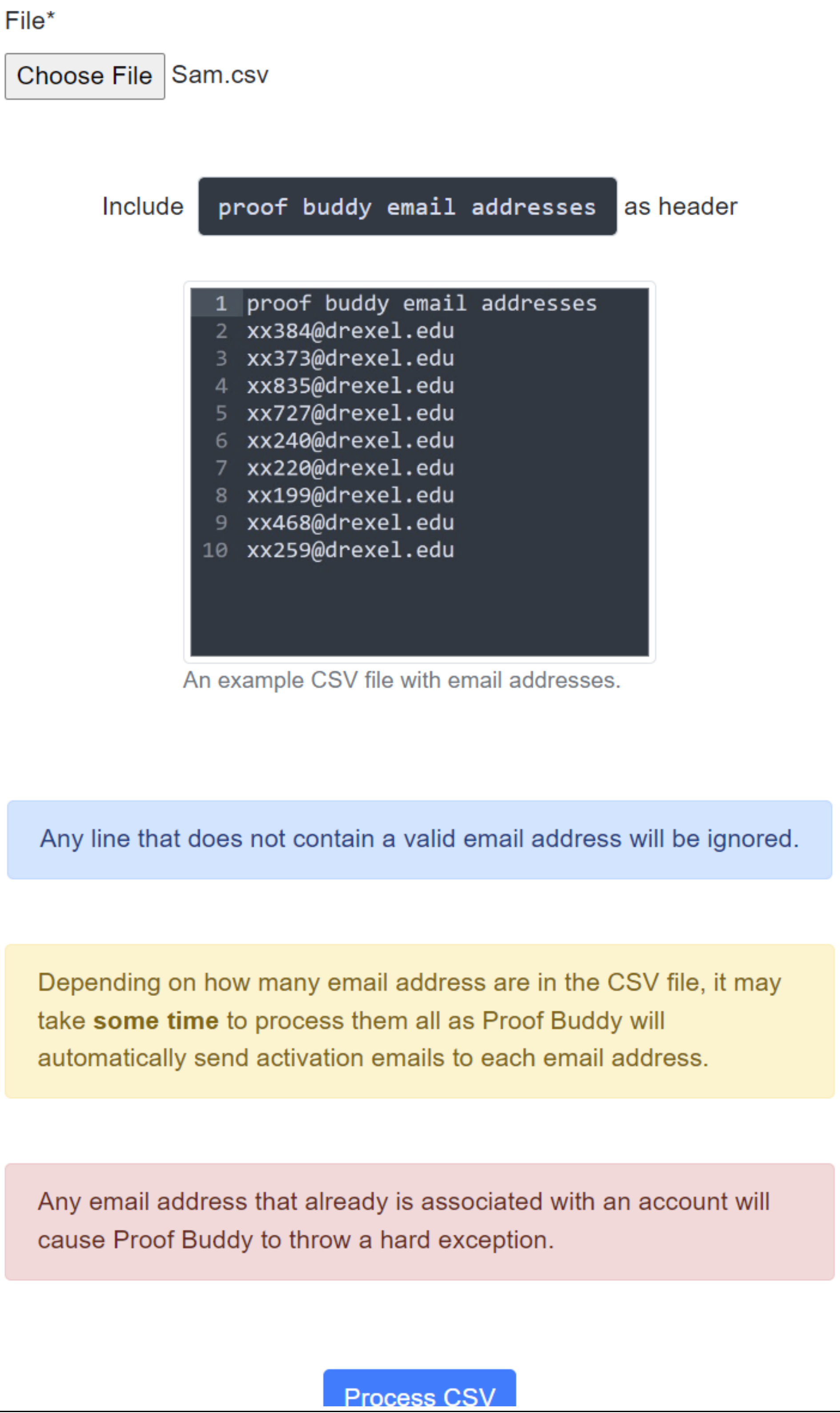
Use Case 13	Bulk create accounts by importing CSV of email addresses
Actors	Instructors
Description	This use case describes how an instructor can easily upload an entire class of students with all their email addresses in one CSV file with one button click.
Pre-conditions	The instructor must have an instructor account. Then the students account can be created.
Flow of events	1. Instructor will navigate to the help link on the navbar (top bar). 2. They will click on bulk create accounts. 3. They will have a .CSV file prepared with the student email addresses. 4. Once the .CSV is uploaded, the application will begin processing each email, creating a student account with the first part of the email address as their username and a random 16-character password. 5. Once it is completed, there will be a screen stating the upload is complete. 6. Every email address on the .CSV file will have an activation link to activate the student account.
Post-conditions	1. Student will be able to click to activate account and change password 2. Then student will be able to work in ProofBuddy right away.

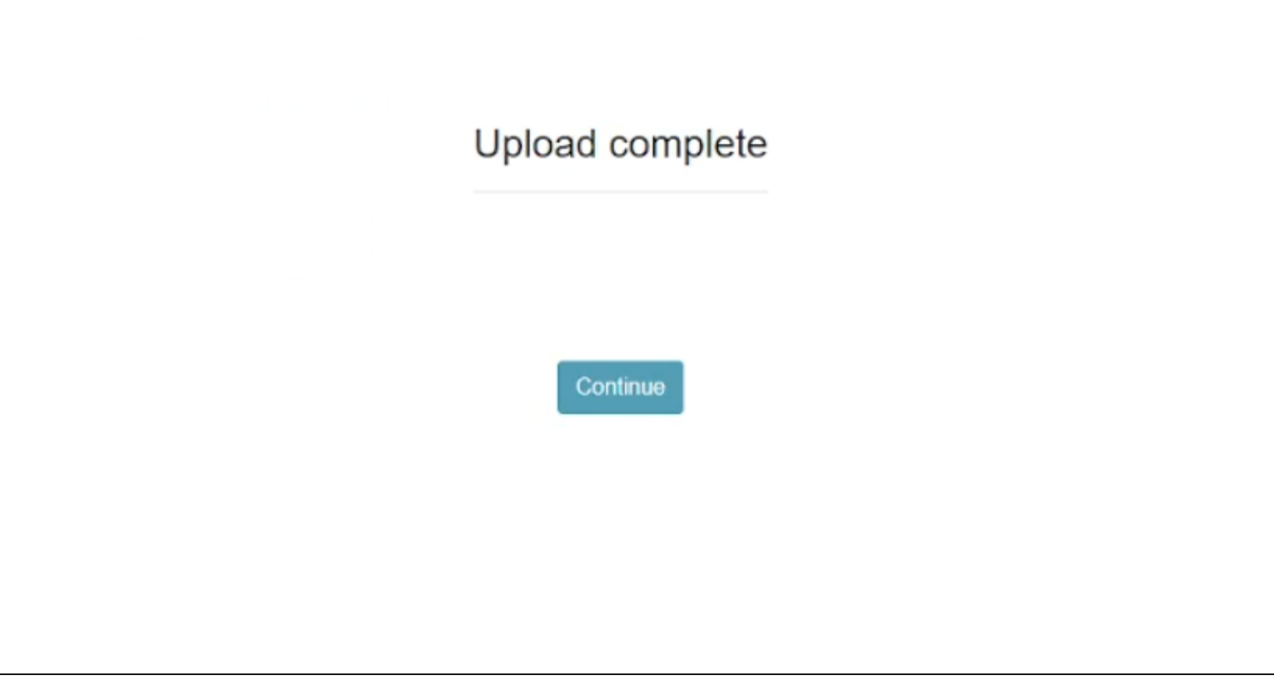
Screenshots	

Proof Buddy

Welcome, therapprof

Begin working on some proofs!

	 File*
	 Choose File Sam.csv
	 Include `proof buddy email addresses` as header
	 `proof buddy email addresses`
	 `xx384@drexel.edu`
	 `xx373@drexel.edu`
	 `xx835@drexel.edu`
	 `xx727@drexel.edu`
	 `xx240@drexel.edu`
	 `xx220@drexel.edu`
	 `xx199@drexel.edu`
	 `xx468@drexel.edu`
	 `xx259@drexel.edu`
	 An example CSV file with email addresses.
	 Any line that does not contain a valid email address will be ignored.
	 Depending on how many email address are in the CSV file, it may take **some time** to process them all as Proof Buddy will automatically send activation emails to each email address.
	 Any email address that already is associated with an account will cause Proof Buddy to throw a hard exception.
	 Process CSV

	 Upload complete

4. References

Requirements Specification and Use Cases from Team 1
https://en.wikipedia.org/wiki/Natural_deduction
https://en.wikipedia.org/wiki/Mathematical_proof