

## **Final Report and Documentation**

**Name of the Project:** Autograder Frontend

### **Members of the Project:**

The following members contributed to the development of the project:

- Mainak Sarkar
- Max Smith
- Md Hasan Al Banna
- Riddhi Ghate
- Ryan Gonzalez
- Saksham Mehta
- Qinyao Hou

### **Table of Contents:**

Introduction	2
User Stories	3
Team Roles	39
Scrum Iterations	39
Member Contribution	41
Customer Meetings	42
BDD/TDD Process & Benefits/Problems	47
Configuration Management Approach	48
Heroku Deployment Issues	49
Tools/Gems	50
Repository Contents and Deployment Process	53
Links	54

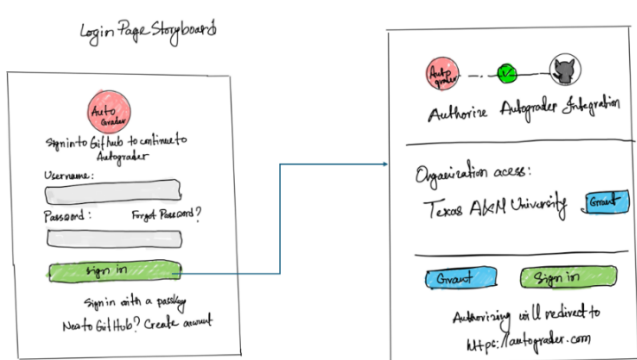
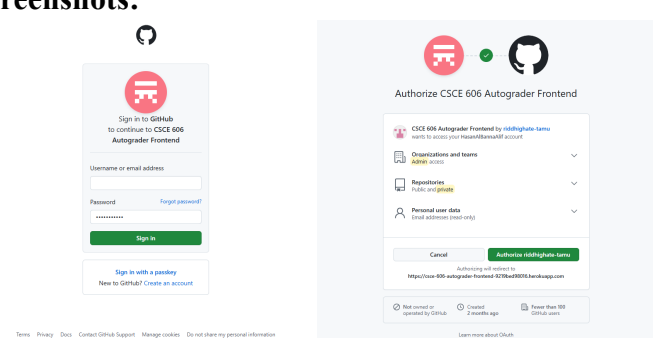
## Introduction

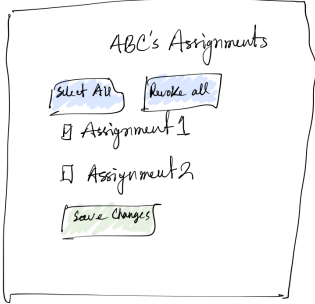
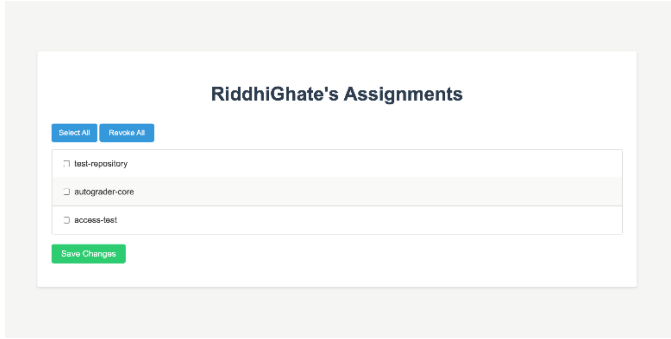
The “Autograder Frontend” is a project for building an interface to create and update assignments to upload them to the Autograder (gradescope). The project was built from scratch as it is not a legacy project. It uses the [autograder-core](#) repository created by Dr. Philip Ritchey to create the assignments. For user authentication, GitHub OAuth was used. The project followed Behavior-Driven Development (BDD) and Test-Driven Development (TDD). Rspec and Cucumber scenarios were used to test the project. Along with them to test the JavaScript functionalities, Capybara was used. The project is deployed on the Heroku server. The agile development process was followed for this project. Stand-up meetings, sprint planning meetings, and retrospective meetings were held at regular intervals. The team working agreements were updated after each sprint.

The main customer need was a usable, accessible front-end for the CSCE 120 auto-grading system that allows for easier creation and maintenance of assignments and their constituent tests. The admin(s) of CSCE 120’s GitHub organization will have the ability to add new users and manage permissions, controlling who can view and edit tests within each assignment. Also, the user should be able to download the repository for the assignment. Other demanding features were assignment searching, test grouping, file uploading to GitHub, visualizing the file structure of the GitHub repository for the assignments, and ensuring different interfaces for different test types. The user Authentication system allows users to log in to the system if they are under the GitHub organization. The “Manage Users” page allows for user management-related permissions (read or write). The “Course Dashboard” page allows the creation of a new assignment which creates a repository in the GitHub. You can search assignments and also download the assignment as a zip file using the “Export to Gradescope” button. The persons who can manage the assignment can be changed from here as well. After the creation of assignments, the tests for the assignments can be accessed. The interface allows one to create new tests, update them, and delete them. Also, the tests can be assigned to a specific group. You can see the file tree from this page. While creating the tests, based on the selected type, this page loads a different interface related to that test. You can select a specific file path to upload files through this interface. This way the project meets the major user requirements. For this project, stakeholders include CSCE 120 course instructors and teaching assistants, who need a simpler and more automated way to create, modify, and view auto-graded assignments, as well as students, who benefit from having high-quality tests for their assignments. The ability to run test cases independently within the interface will ensure that tests are behaving as expected and allow for rapid assignment development.

## User Stories

Here, we will describe each user story. We have used the Lo-fi mockups and screenshots to explain the user stories.

Sprint 1	
Story 1	<b>Story Description:</b> Feature: Login system As an instructor or TA So that I can create a new assignment or modify a previous one I want to authenticate with a third-party service and login to the system.
	<b>Points:</b> 4
	<b>Implementation Status:</b> Completed
	<b>Changes:</b> None
	<b>Mockups:</b> 
	<b>Screenshots:</b> 

<b>Story 2</b>	<p><b>Story Description:</b>  Feature: Access control  As an administrator of the CSCE 120 GitHub organization  So that I can manage who can interact with the application and how  I want to be able to do so through modifying users' roles within the GitHub organization through the application.</p> <p><b>Points:</b> 3</p> <p><b>Implementation Status:</b> Completed</p> <p><b>Changes:</b> None</p> <p><b>Mockups:</b></p>  <p><b>Screenshots:</b></p> 
<b>Story 3</b>	<p><b>Story Description:</b>  Feature: Create a new assignment  As an instructor  So that I can generate tests for a new autograded assignment  I want to create a new Git repository under the CSCE 120 GitHub organization from the autograded-assignment-template repository.</p> <p><b>Points:</b> 3</p> <p><b>Implementation Status:</b> Completed</p> <p><b>Changes:</b> None</p>

### Mockups:

A hand-drawn sketch of a web form titled "New Assignment". It contains two text input fields, both with "hw15" entered. The first field is labeled "Assignment name" and the second is labeled "Repository name". Below the inputs is a "Submit" button and a link labeled "Back to assignments".

### Screenshots:

A screenshot of a web page titled "New assignment". It features two input fields labeled "Assignment name" and "Repository name", both containing the text "hw15". Below the inputs is a "Submit" button and a link labeled "Back to assignments".

### Story 4

#### Story Description:

Feature: Add new test

As an instructor (or TA)

So that I can test some aspect of student code

I want to add a new test case to an assignment.

**Points:** 1

**Implementation Status:** Completed

**Changes:** None

Mockups:

Assignment 1

Tests for Assignment 1

• test 1

Admin tests: [Create and Download ZIP](#)

New Test for Assignment 1

Name

Points

Test type Coverage ▾

Include

Number

Show output ☐

Skip ☐

Timeout

Visibility Visible ▾

Actual Test

[Create Test](#)

[Back to assignment](#)

Screenshots:

Assignment: aa

Tests for aa

[Add new test](#) [Create and Download ZIP](#)

New Test for Assignment aa

Name

Points

Test type coverage ▾

Target

Include

Number

Show output ☐

Skip ☐

Timeout

Visibility Visible ▾

Actual test

[Create Test](#)

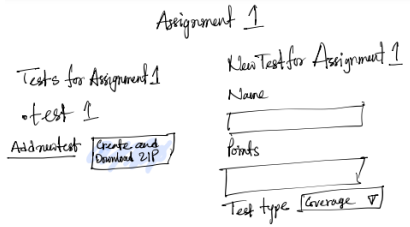
[Back to Assignment](#)

Story 5	<b>Story Description:</b> Feature: Review existing test cases As an instructor or TA So that I can review details of the existing test cases of an assignment I want to fetch and display the existing test cases
	<b>Points:</b> 1
	<b>Implementation Status:</b> Completed
	<b>Changes:</b> None

<div><div>Mockups:</div><div><div><div>Assignment 1</div><div>Tests for Assignment 1</div><div>• test 1</div><div>Assignment</div><div>Create and Download ZIP</div></div><div><div>New Test for Assignment 1</div><div>Name</div><div>Points</div><div>Test type Coverage ✓</div><div>Include</div><div>Number</div><div>Show output <input type="checkbox"/></div><div>Skip <input type="checkbox"/></div><div>Timeout</div><div>Visibility ✓</div><div>Actual Test</div><div>Create Test</div><div>back to assignment</div></div></div></div> <tr><td data-bbox="323 856 1430 1411"><div><div>Screenshots:</div><div><div>Assignment: aa</div><div><div>Tests for aa</div><div><a href="#">Add new test</a> <a href="#">Create and Download ZIP</a></div></div><div><div>New Test for Assignment aa</div><div>Name</div><div>Points</div><div>Test type coverage</div><div>Target</div><div>Include</div><div>Number</div><div>Show output <input type="checkbox"/></div><div>Skip <input type="checkbox"/></div><div>Timeout</div><div>Visibility Visible</div><div>Actual test</div><div>Create Test</div><div><a href="#">Back to Assignment</a></div></div></div></div></td></tr>	<div><div>Screenshots:</div><div><div>Assignment: aa</div><div><div>Tests for aa</div><div><a href="#">Add new test</a> <a href="#">Create and Download ZIP</a></div></div><div><div>New Test for Assignment aa</div><div>Name</div><div>Points</div><div>Test type coverage</div><div>Target</div><div>Include</div><div>Number</div><div>Show output <input type="checkbox"/></div><div>Skip <input type="checkbox"/></div><div>Timeout</div><div>Visibility Visible</div><div>Actual test</div><div>Create Test</div><div><a href="#">Back to Assignment</a></div></div></div></div>
<div><div>Screenshots:</div><div><div>Assignment: aa</div><div><div>Tests for aa</div><div><a href="#">Add new test</a> <a href="#">Create and Download ZIP</a></div></div><div><div>New Test for Assignment aa</div><div>Name</div><div>Points</div><div>Test type coverage</div><div>Target</div><div>Include</div><div>Number</div><div>Show output <input type="checkbox"/></div><div>Skip <input type="checkbox"/></div><div>Timeout</div><div>Visibility Visible</div><div>Actual test</div><div>Create Test</div><div><a href="#">Back to Assignment</a></div></div></div></div>	

<div><div>Mockups:</div><div><div><div>Assignment 1</div><div>Tests for Assignment 1</div><div>• test 1</div><div>Assignment</div><div>Create and Download ZIP</div></div><div><div>New Test for Assignment 1</div><div>Name</div><div>Points</div><div>Test type Coverage ✓</div><div>Include</div><div>Number</div><div>Show output <input type="checkbox"/></div><div>Skip <input type="checkbox"/></div><div>Timeout</div><div>Visibility ✓</div><div>Actual Test</div><div>Create Test</div><div>back to assignment</div></div></div></div> <div><div>Screenshots:</div><div><div>Assignment: aa</div><div><div>Tests for aa</div><div><a href="#">Add new test</a> <a href="#">Create and Download ZIP</a></div></div><div><div>New Test for Assignment aa</div><div>Name</div><div>Points</div><div>Test type coverage</div><div>Target</div><div>Include</div><div>Number</div><div>Show output <input type="checkbox"/></div><div>Skip <input type="checkbox"/></div><div>Timeout</div><div>Visibility Visible</div><div>Actual test</div><div>Create Test</div><div><a href="#">Back to Assignment</a></div></div></div></div>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



	<b>Screenshots:</b> Updated in the backend
<b>Story 8</b>	<b>Story Description:</b> Feature: Update remote assignment repository As an instructor (or TA) So that I can update the assignment I want to push assignment changes to the remote GitHub repository
	<b>Points:</b> 2
	<b>Implementation Status:</b> Completed
	<b>Changes:</b> None
	<b>Mockups:</b> Updated in the backend
	<b>Screenshots:</b> Updated in the backend
<b>Story 9</b>	<b>Story Description:</b> Feature: Export assignment as <b>.zip</b> As an instructor So that I can upload an autograder assignment to Gradescope I want to download the autograder assignment as a <b>.zip</b> file.
	<b>Points:</b> 2
	<b>Implementation Status:</b> Completed
	<b>Changes:</b> None
	<b>Mockups:</b> 

## Screenshots:

Assignment: aa

Tests for aa

[Add new test](#) [Create and Download ZIP](#)

New Test for Assignment aa

Name

Points

Test type coverage

Target

Include

Number

Show output ☐

Skip ☐

Timeout

Visibility Visible

Actual test

[Create Test](#)

[Back to Assignment](#)

## Sprint 2

### Story 1

#### Story Description:

Feature: Search functionality on the assignments page

As an instructor (or TA)

So that I can search for a specific assignment to modify

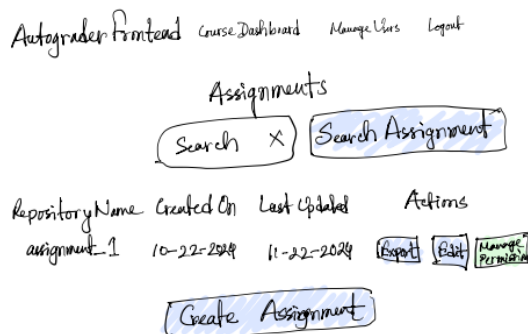
I want to access a search bar that filters the display of assignments on the page based on my search query

**Points: 2**

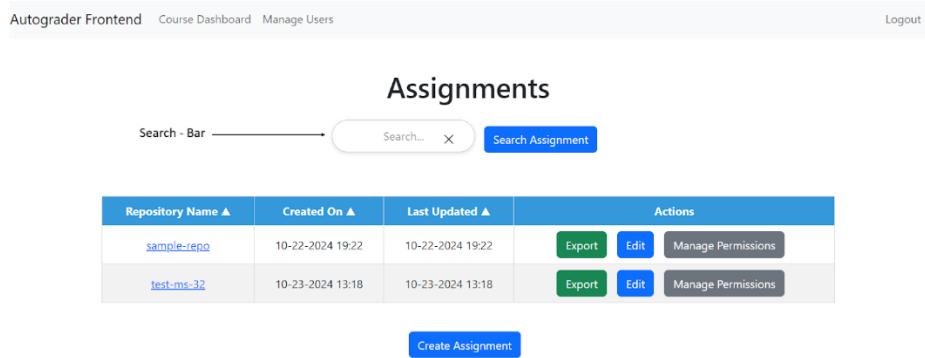
**Implementation Status:** Completed

**Changes:** None

#### Mockups:



## Screenshots:



## Story 2

### Story Description:

Feature: Update Assignments view

As a CSCE 120 GitHub organization member

So that I can more easily find assignments

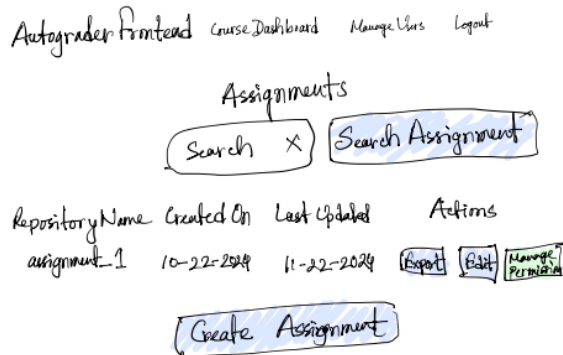
I want to see assignments displayed in a table that can be sorted by repo name, creation date, or update date

**Points: 1**

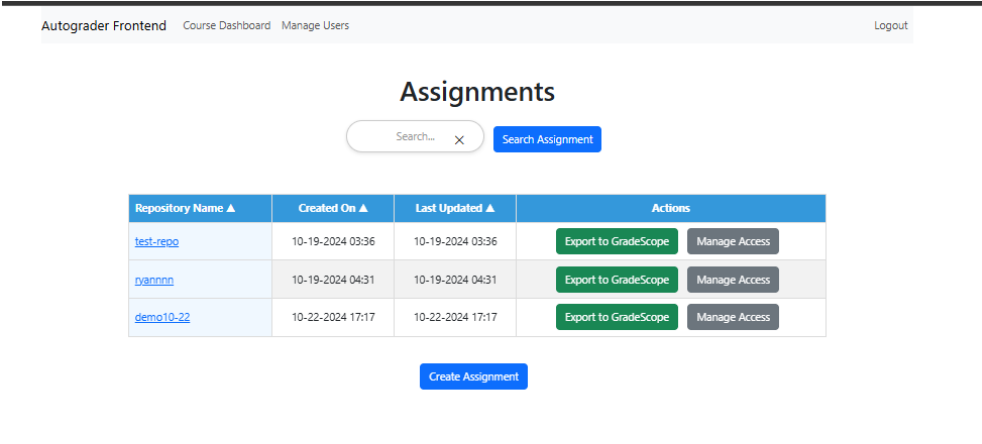
**Implementation Status: Completed**

**Changes: None**

### Mockups:



## Screenshots:



### Story 3

#### Story Description:

Feature: Logout System

As an instructor (or TA)

So that I can ensure my session is closed and my work is protected when I have completed my tasks,

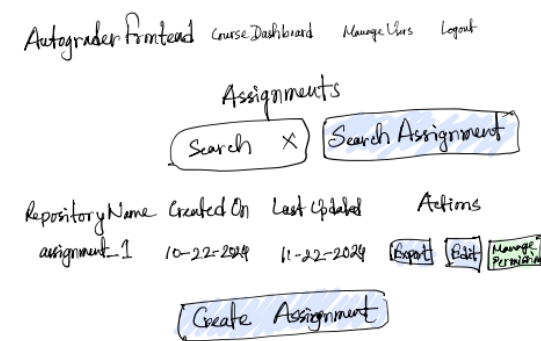
I want to securely log out of the system

**Points:** 1

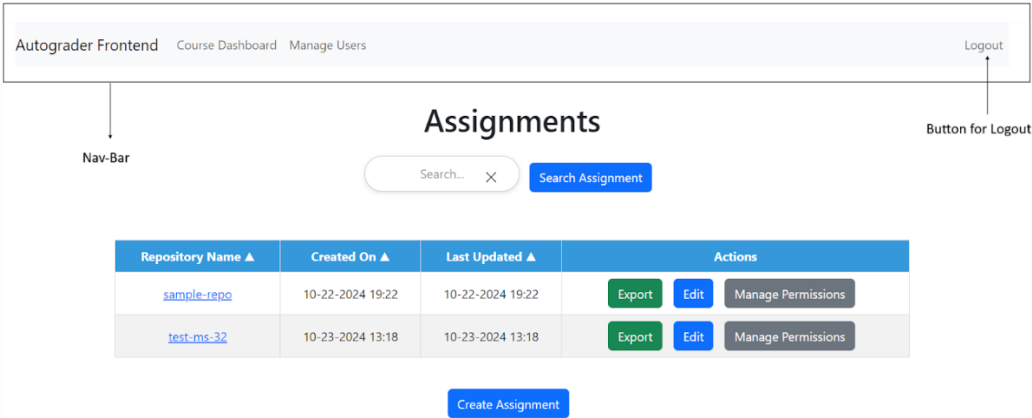
**Implementation Status:** Completed

**Changes:** None

#### Mockups:



Screenshots:



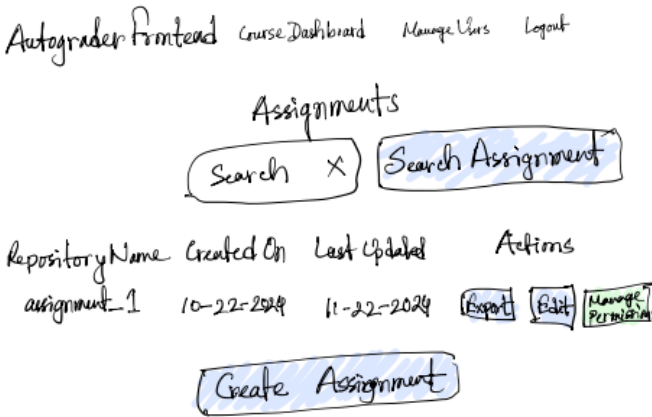
**Story 4 Story Description:**  
Feature: Navigation Bar  
As an instructor (or TA)  
So that I can easily navigate to different pages of the app (course dashboard, logout, manage users, app name),  
I want to have a navigation bar at the top of the view that lets me browse quickly to the other pages of the application

**Points:** 1

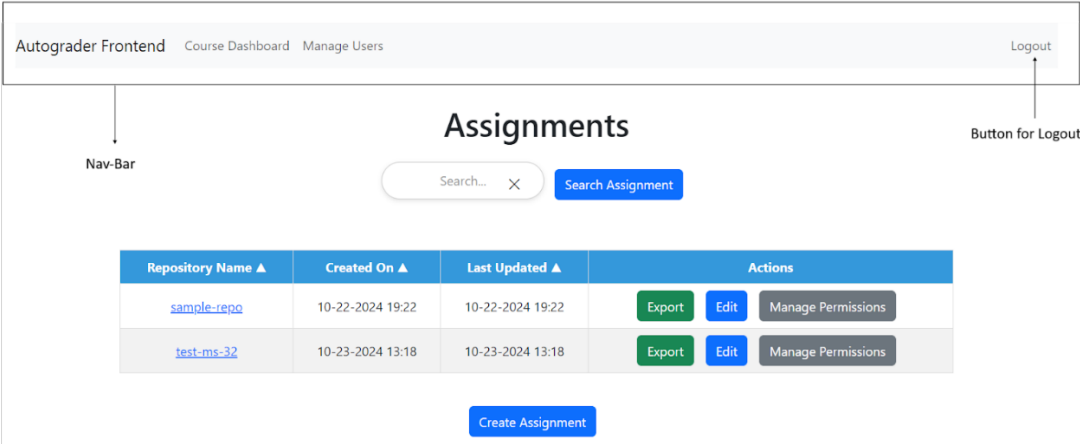
**Implementation Status:** Completed

**Changes:** None

**Mockups:**



Screenshots:



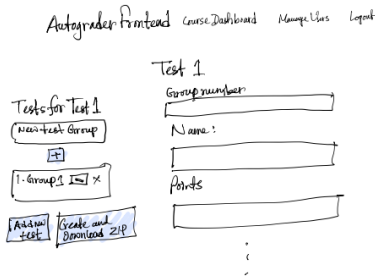
**Story 5** **Story Description:**  
Feature: Test Case Groupings Model  
As a CSCE 120 GitHub organization member,  
So that I can see which tests are related,  
I want to the app to contain groupings of test cases

**Points:** 1

**Implementation Status:** Completed

**Changes:** None

**Mockups:**



## Screenshots:

Autograder Frontend Course Dashboard Manage Users Logout

### Assignment: test-ms-32

#### Tests for test-ms-32

New Test Grouping

+

1) Miscellaneous Tests x

2) test-ms-32-grp-1 x

1) test-1 (1.0 pts.)

Add new test Create and Download ZIP

#### New Test for Assignment test-ms-32

Option to include Group Number → Group Number

Name

Points

Test type coverage

Target

Include

Test Grouping List

Presence of Tests within Group

## Story 6

### Story Description:

Feature: Test Case Groupings CRUD

As a CSCE 120 GitHub organization member,

So that I can interact with test case groupings

I want to be able to create, view, update, & delete groupings of related test cases

Points: 2

Implementation Status: Completed

Changes: None

### Mockups:

Autograder Frontend Course Dashboard Manage Users Logout

#### Test 1

##### Tests for Test 1

New Test Group

+

1. Group 1 x

Add new test Create and Download ZIP

Group number

Name:

Points

...

## Screenshots:

Autograder Frontend Course Dashboard Manage Users Logout

### Assignment: test-ms-32

#### Tests for test-ms-32

New Test Grouping

+

1) Miscellaneous Tests x

2) test-ms-32-grp-1 x

1) test-1 (1.0 pts.)

Add new test Create and Download ZIP

#### New Test for Assignment test-ms-32

Option to include Group Number → Group Number

Name

Points

Test type coverage

Target

Include

Test Grouping List

Presence of Tests within Group

## Story 7

### Story Description:

Feature: Manage user access for each assignment

As an instructor

So that I can manage a user's permissions for each assignment

I want to add a read, write, and no access privilege checkbox for each assignment under a user

**Points:** 2

**Implementation Status:** Completed

**Changes:** None

### Mockups:

ABC's Assignments

Repository	Read <input type="checkbox"/> select all	Write <input type="checkbox"/> select all
repo1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
repo2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Save Changes



## Screenshots:

Autograder Frontend Course Dashboard Manage Users Logout

### masarkar-tamu's Assignments

Repository	Read	Select All	Revoke All	Write	Select All	Revoke All
sample-repo	<input type="checkbox"/>			<input type="checkbox"/>		
test-ms-32	<input type="checkbox"/>			<input type="checkbox"/>		

Save Changes

## Story 8 Story Description:

Feature: Manage assignment-wise access for each user

As an instructor

So that I can manage all users' permissions within an assignment

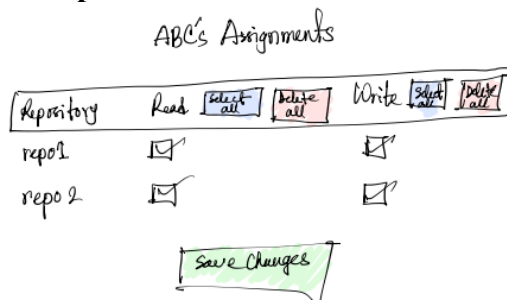
I want to add a read, write, and no access privilege checkbox for each user under an assignment.

**Points: 2**

**Implementation Status: Completed**

**Changes: None**

## Mockups:



## Screenshots:

Autograder Frontend Course Dashboard Manage Users Logout

### Access Management for test-ms-32

User	Read	Select All	Revoke All	Write	Select All	Revoke All
masarkar-tamu	<input type="checkbox"/>			<input type="checkbox"/>		

Save Changes

Story 9

Story Description:

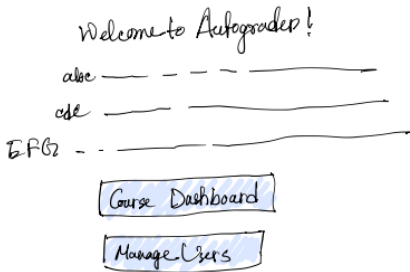
Feature: Consistent button styling and clear labels  
As a CSCE 120 GitHub organization member  
So that I understand how to navigate about the app  
I want to see buttons, rather than links, with intuitive labels

Points: 1

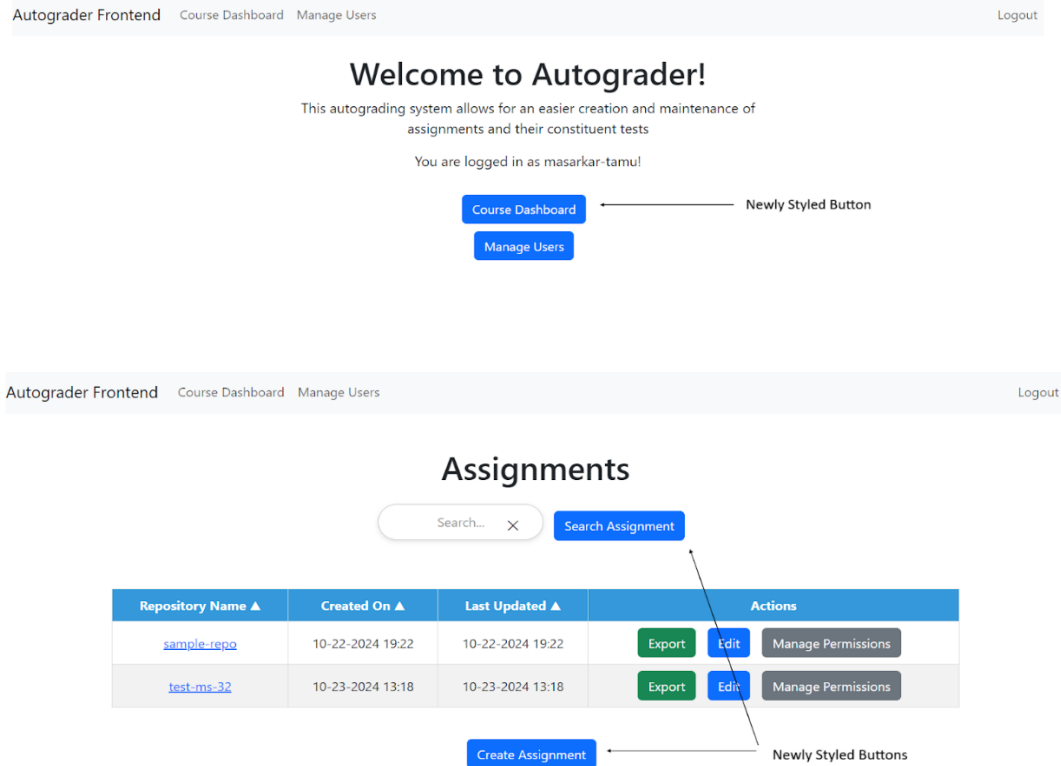
Implementation Status: Completed

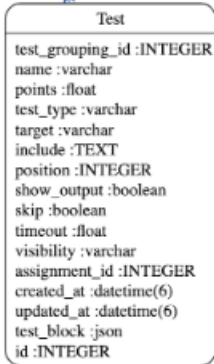
Changes: None

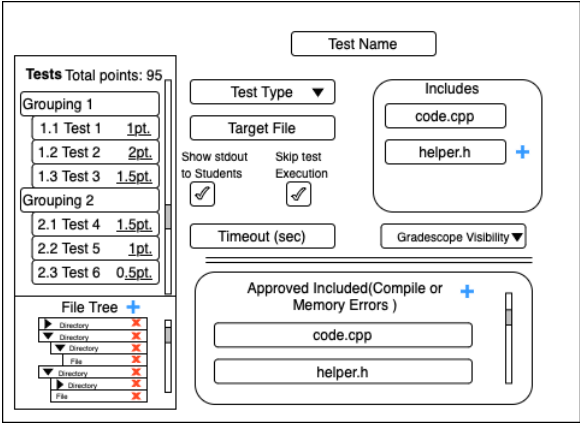
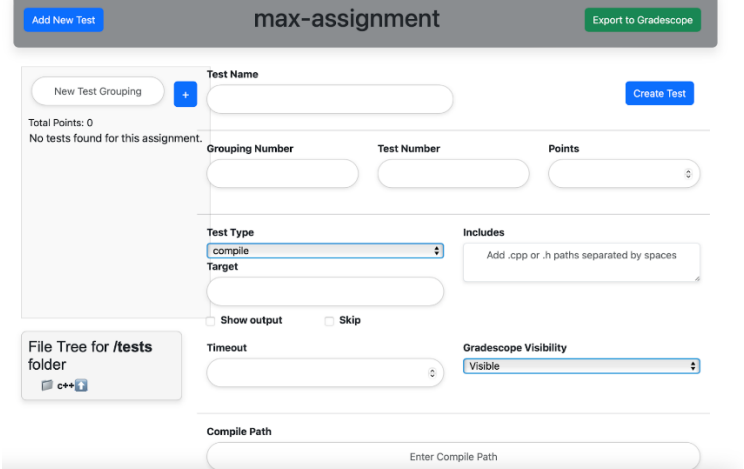
Mockups: Links were converted



Screenshots:



<b>Story 10</b>	<b>Story Description:</b> Feature: Fix production environment As a CSCE 120 GitHub organization member So that I can use the autograder website I want to properly functional production environment
	<b>Points:</b> 2
	<b>Implementation Status:</b> Completed
	<b>Changes:</b> None
	<b>Mockups:</b> Updated in the backend
	<b>Screenshots:</b> Updated in the backend
<b>Sprint 3</b>	
<b>Story 1</b>	<b>Story Description:</b> Feature: Update Test schema As a CSCE 120 GitHub organization member So that I can more easily create tests of different types I want the test block to be represented in the backend as a JSONB type
	<b>Points:</b> 2
	<b>Implementation Status:</b> Completed
	<b>Changes:</b> None
	<b>Mockups:</b> 
	<b>Screenshots:</b> Updated in the Database

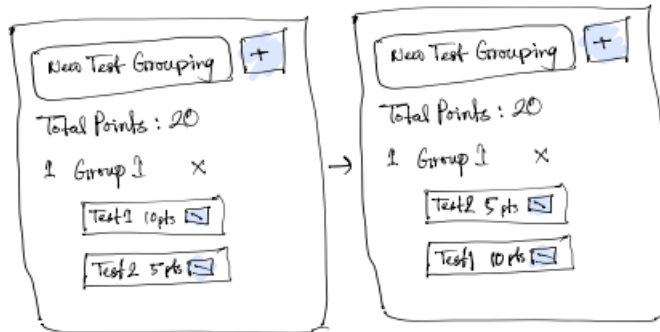
<div>Story 2</div>	<div><div><div>Story Description:</div><div>Feature: Test Case Editor UI Updates</div><div>As a CSCE 120 GitHub organization member</div><div>So that I can more easily create tests of different types</div><div>I want the test block to be able to see different fields for different test types</div></div><div>Points: 1</div><div>Implementation Status: Completed</div><div>Changes: None</div><div><div>Mockups:</div><div></div></div><div><div>Screenshots:</div><div></div></div></div>
<div>Story 3</div>	<div><div>Story Description:</div><div>Feature: Move Tests b/t Groupings</div><div>As a CSCE 120 GitHub organization member,</div><div>So that I can change which grouping a test is in,</div><div>I want to be able to drag and drop tests from one grouping to another</div><div>(Concatenate grouping &amp; test number to produce test case numbers in code.tests)</div></div>

**Points: 1**

**Implementation Status: Completed**

**Changes: None**

**Mockups:**


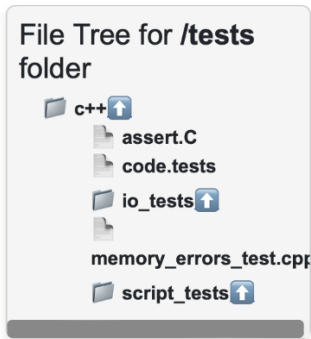


**Screenshots:**

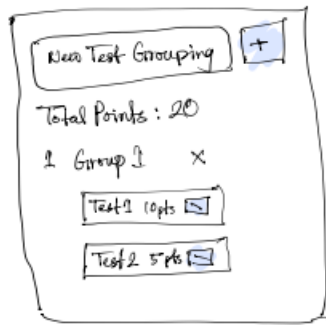
This screenshot shows the 'New Test Grouping' interface. On the left, a sidebar contains a 'New Test Grouping' button with a plus icon, 'Total Points: 20', and a list of tests under '1) Miscellaneous x'. The tests are '1) test1 (10.0 pts.)' and '2) test2 (10.0 pts.)', each with a blue arrow icon. A red box highlights these two test items. The main area on the right has a 'Test Name' field, a 'Create Test' button, and input fields for 'Grouping Number', 'Test Number', and 'Points'. Below these are 'Test Type' (a dropdown menu), 'Includes' (a text area for file paths), and 'target' (a text field). At the bottom, there are checkboxes for 'Show output' and 'Skip'.

**After drag and drop**

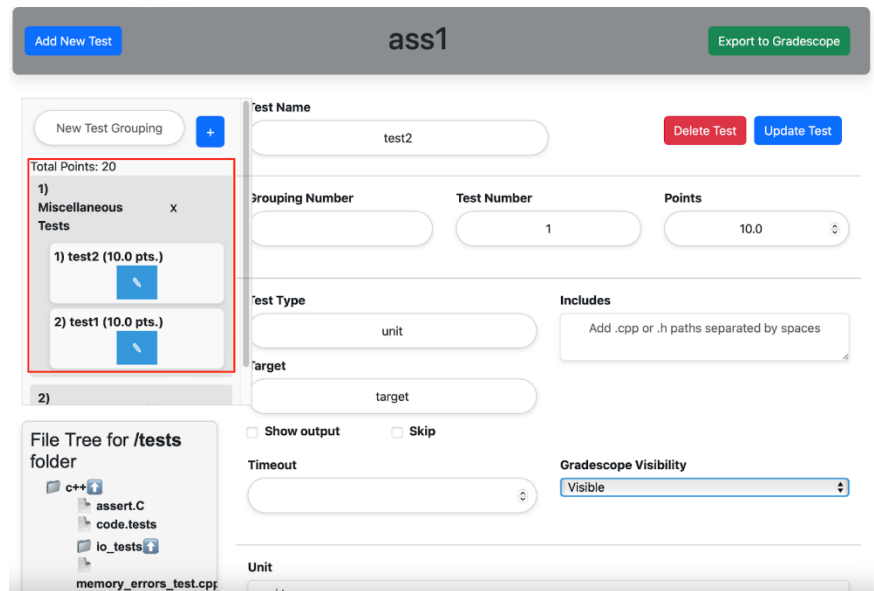
This screenshot shows the interface after a drag-and-drop action. A dark grey header bar at the top contains an 'Add New Test' button, the text 'ass1', and an 'Export to Gradescope' button. The 'New Test Grouping' sidebar on the left is identical to the previous screenshot, with a red box highlighting the test items. The main configuration area now shows 'test2' in the 'Test Name' field. The 'Test Number' field contains the value '1', and the 'Points' field contains '10.0'. The 'Test Type' dropdown is set to 'unit', and the 'target' field contains 'target'. The 'Includes' field remains empty.

<p><b>Story 4</b></p>	<p><b>Story Description:</b>  Feature: File upload  As a CSCE 120 GitHub organization member  So that I can use specific files in tests  I want to be able to upload a file to the remote repository</p> <p><b>Points:</b> 3</p> <p><b>Implementation Status:</b> Completed</p> <p><b>Changes:</b> None</p> <p><b>Mockups:</b></p>  <p><b>Screenshots:</b></p> 
<p><b>Story 5</b></p>	<p><b>Story Description:</b>  Feature: Points edit  As a CSCE 120 GitHub instructor  So that I can edit each test card with points and display in the test grouping bars  I want to calculate the overall points to meet the 100</p> <p><b>Points:</b> 2</p> <p><b>Implementation Status:</b> Completed</p> <p><b>Changes:</b> None</p>

## Mockups:



## Screenshots:



**Story 6 Story Description:**  
Feature: Assignments table UI update  
As a CSCE 120 organization member  
So that I can more intuitively interact with the Assignments table  
I want to see more accessible styling and more clear labeling

**Points:** 1

**Implementation Status:** Completed

**Changes:** None

Mockups:

Sorting Directions

Assignments

Assignment Name▲	Created On▲	Last Updated▲	Actions
Assingment_1	10-15-2024 23:17	10-15-2024 23:17	<div>Export to Gradescope</div> <div>Manage Access</div>

Hyper Link to take you to view page of the tests

Left Aligned

Screenshots:

Autograder Frontend

Course Dashboard

Manage Users

Logout

Assignments

Search...

Search Assignment

Assignment Name▲	Created On▲	Last Updated▲	Actions
max-assignment	10-28-2024 16:54	10-28-2024 16:54	<div>Export to Gradescope</div> <div>Manage Access</div>
assignment-test-19	11-02-2024 00:41	11-02-2024 00:41	<div>Export to Gradescope</div> <div>Manage Access</div>

Create Assignment

Story 7

Story Description:

Feature: Test Block Partial

As a CSCE 120 organization member

So that I can see different test block partials based on the test type

I want to see different partials when I select the different test type

Points: 1.5

Implementation Status: Completed

Changes: None



## Mockups:

Test type  
Compile

Includes  
Add .cpp or .h files

Target

Show Output Skip

Time out  
Gradescope Visibility  
Visible

Compile Path

Add Compile Path

## Screenshots:

When the test type is compile, render compile partial

Test Type  
compile

Includes  
Add .cpp or .h paths separated by spaces

Target

Show output Skip

Timeout  
Gradescope Visibility  
Visible

Compile Path  
Enter Compile Path

Add Compile Path

When the test type is unit, render unit partial

Test Type  
unit

Includes  
Add .cpp or .h paths separated by spaces

Target

Show output Skip

Timeout  
Gradescope Visibility  
Visible

Unit  
Enter Unit

## Story 8 Story Description:

Feature: File Browser Partial

As a CSCE 120 GitHub organization member

So that I can see the file tree of the remote repo

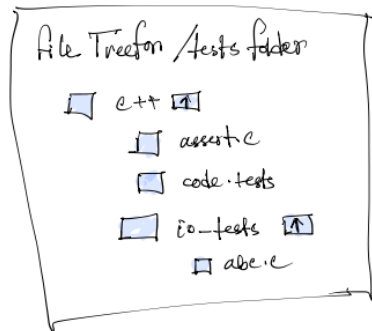
I want to see a directory dropdown

**Points:** 2

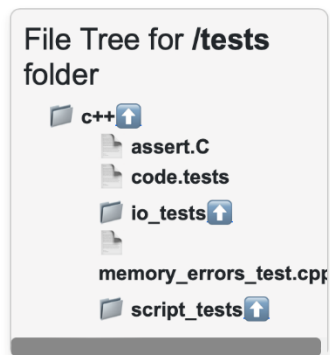
**Implementation Status:** Completed

**Changes:** None

**Mockups:**



**Screenshots:**



## Story 9 Story Description:

Feature: Add Cucumber config to enable JS scenarios testing

As a CSCE 120 GitHub organization member

So that I can have the correct functionality of the autograder

The JavaScript codes need to be tested properly

**Points:** 1

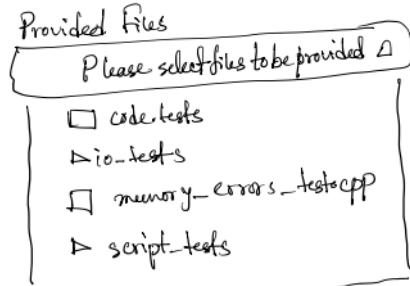
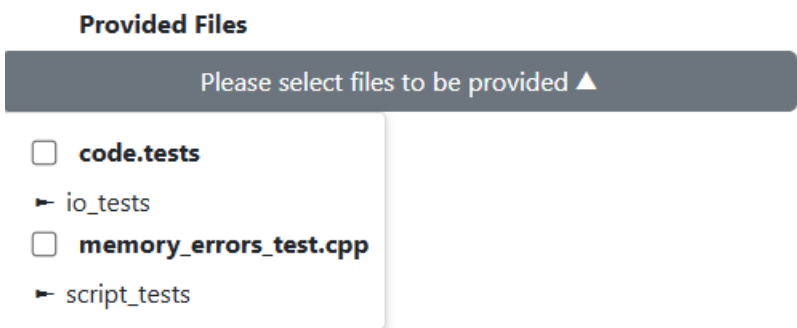
**Implementation Status:** Completed

**Changes:** None

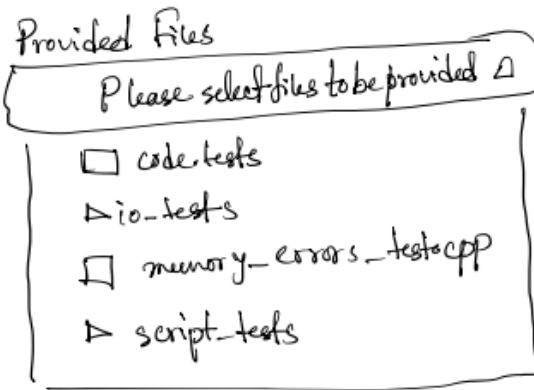
**Mockups:** Done in the backend

**Screenshots:** Done in the backend

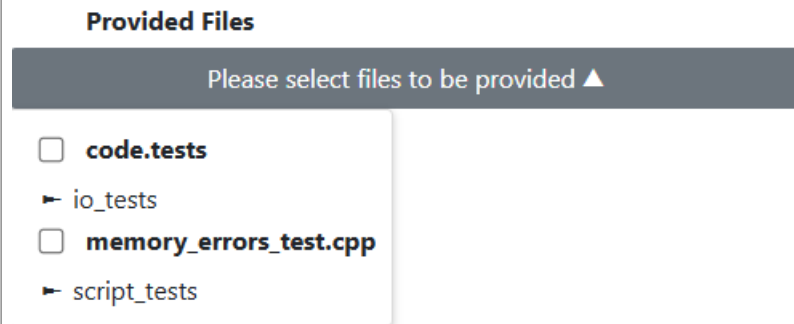
**Sprint 4**

<b>Story 1</b>	<p><b>Story Description:</b>  Feature: File Selection  As a CSCE 120 organization member  So that I can select the files for a test  I want to see a dropdown of nested files with a checkbox</p> <p><b>Points:</b> 2</p> <p><b>Implementation Status:</b> Completed</p> <p><b>Changes:</b> None</p> <p><b>Mockups:</b></p>  <p><b>Screenshots:</b></p> 
<b>Story 2</b>	<p><b>Story Description:</b>  Feature: Select files for Includes attribute  As a CSCE 120 GitHub organization member  So that I can select files for Includes attribute when dealing with test cases  I want to select files from a file-tree dropdown for the Includes attribute</p> <p><b>Points:</b> 2</p> <p><b>Implementation Status:</b> Completed</p> <p><b>Changes:</b> None</p>

### Mockups:



### Screenshots:



### Story 3

#### Story Description:

Feature: Test Editor UI Updates

As a CSCE 120 organization member

So that I can more intuitively interact with the test editor

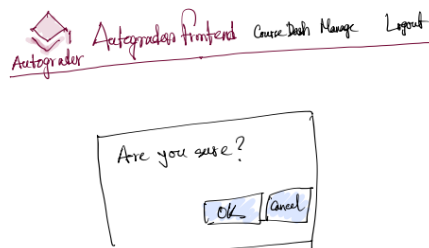
I want to see more accessible styling, more clear labeling, and confirmation for permanent actions

Points: 1

Implementation Status: Completed

Changes: None

### Mockups:



## Screenshots:

**Autograder Frontend** Course Dashboard Manage Users Logout

**test\_assignment** Export to Gradscope

Add New Test

New Test Grouping

Total Points: 0.00

1) Miscellaneous x Tests

File Tree for /tests folder

c++

**Test Name \***

**Grouping Number**

**Test Number**

**Points \***

**Test Type \*** Please select a test type

**Target \*** Select a target file

**Provided Files** Please select files to be provided

**Timeout** 0 seconds Seconds

**Gradscope Visibility** Visible

☐ Show Output ☐ Skip

**mainak-assignment** Export to Gr

**Name \*** test-a1 Delete Test

**Story 4 Story Description:**  
Feature: Update Assignment schema to include approved\_files attribute  
As a CSCE 120 GitHub organization member  
So that I can input approved files when creating an assignment  
I want the database schema to contain an attribute for this

**Points:** 1

**Implementation Status:** Completed

**Changes:** None

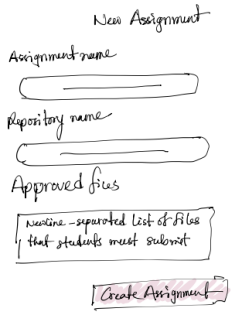
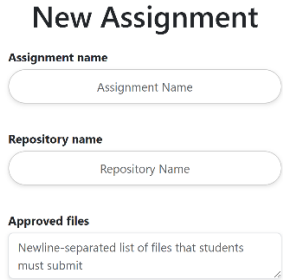
**Mockups:** Updated the database

**Screenshots:** Updated the database

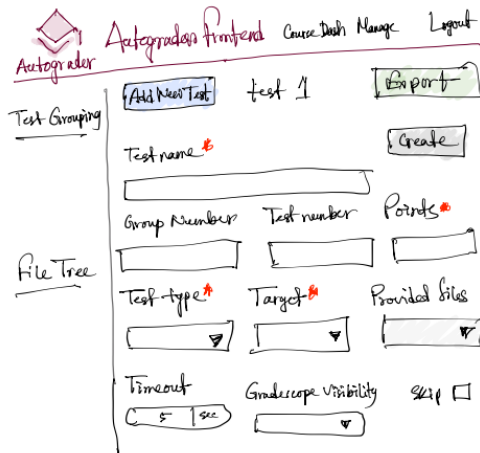
**Story 5 Story Description:**  
Feature: Approved Files UI  
As a CSCE 120 GitHub organization member  
So that I can input approved files when creating an assignment and invoke them in tests  
I want the necessary user interfaces to do so

**Points:** 1

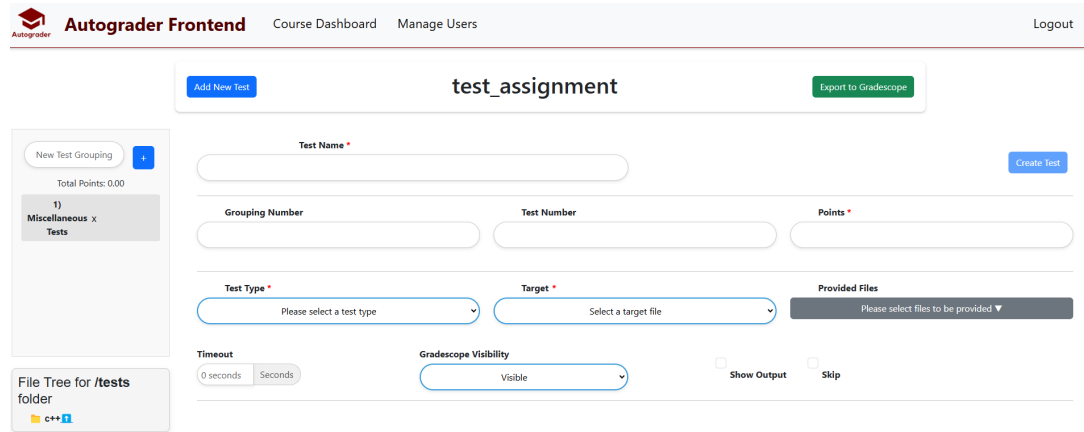
**Implementation Status:** Completed

	<p><b>Changes:</b> None</p>
	<p><b>Mockups:</b></p>  <p><b>Screenshots:</b></p> 
<p><b>Story 6</b></p>	<p><b>Story Description:</b></p> <p>Feature: "Create Test" should be grayed out until minimum necessary information is entered</p> <p>As a CSCE 120 GitHub organization member</p> <p>So that I do not see the create button active while creating a test</p> <p>I want the ‘Create Test’ button to be grayed until the required fields are filled and the required fields should have an asterisk next to them.</p> <p><b>Points:</b> 2</p> <p><b>Implementation Status:</b> Completed</p> <p><b>Changes:</b> None</p>

## Mockups:



## Screenshots:



## Story 7

### Story Description:

Feature: Live update for points

As a CSCE 120 GitHub organization member

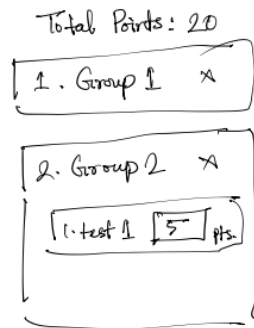
So that I can update the points for a test case live without having to explicitly save it  
I want a single points editor text field

**Points:** 3

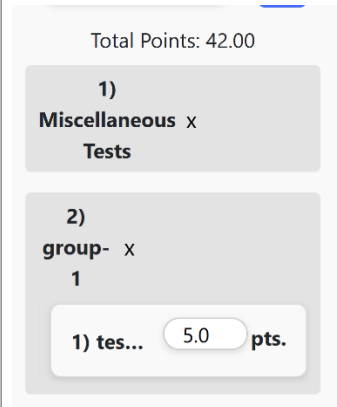
**Implementation Status:** Completed

**Changes:** None

### Mockups:



### Screenshots:



**Story 8 Story Description:**  
Feature: Login Page Changes  
As a CSCE 120 GitHub organization member  
So that I can have ease of use  
I want to be automatically be taken past the login page if I am already logged in

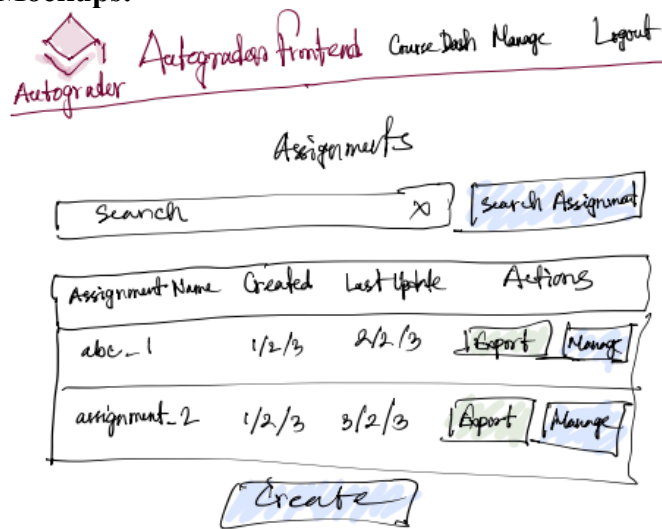
**Points: 1**

**Implementation Status:** Completed

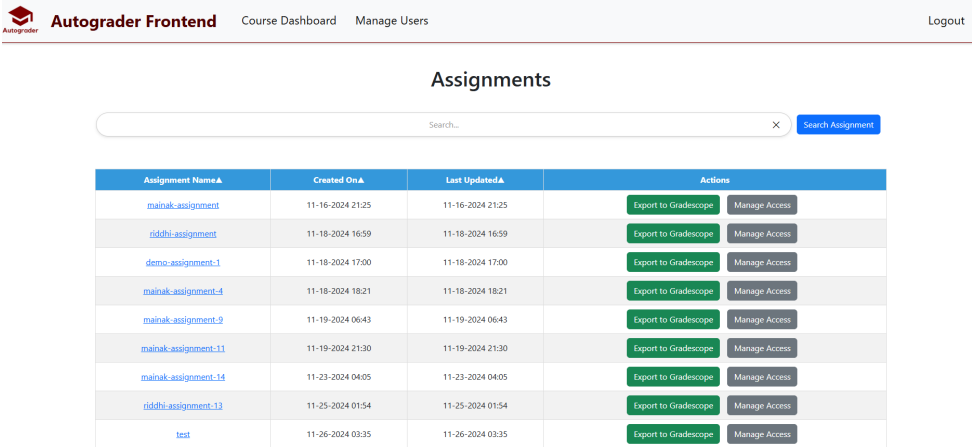
**Changes:** None



Mockups:



Screenshots:



- Story 9

**Story Description:**

Feature: Set files\_to\_submit in run\_autograder script

As a CSCE 120 GitHub organization member

So that I can control which files students can submit

I want the run\_autograder script to be updated with these upon assignment creation
- Points:** 2
- Implementation Status:** Completed
- Changes:** None
- Mockups:** Updated in the backend
- Screenshots:** Updated in the backend

**Story  
10**

**Story Description:**

Feature: General UI changes

As a CSCE 120 GitHub organization member

So that I can navigate easily, complete tasks efficiently

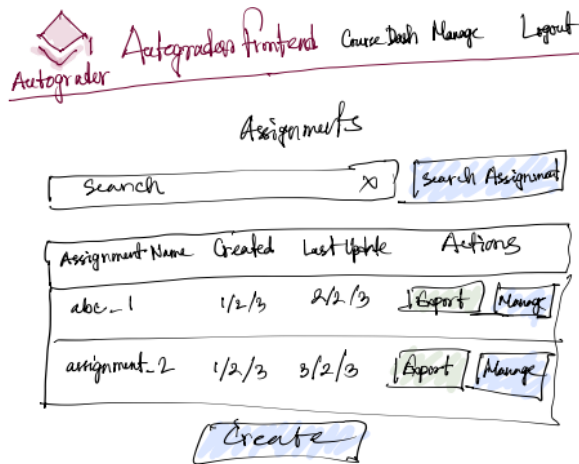
I want the user interface to be intuitive, aesthetically pleasing

**Points:** 1

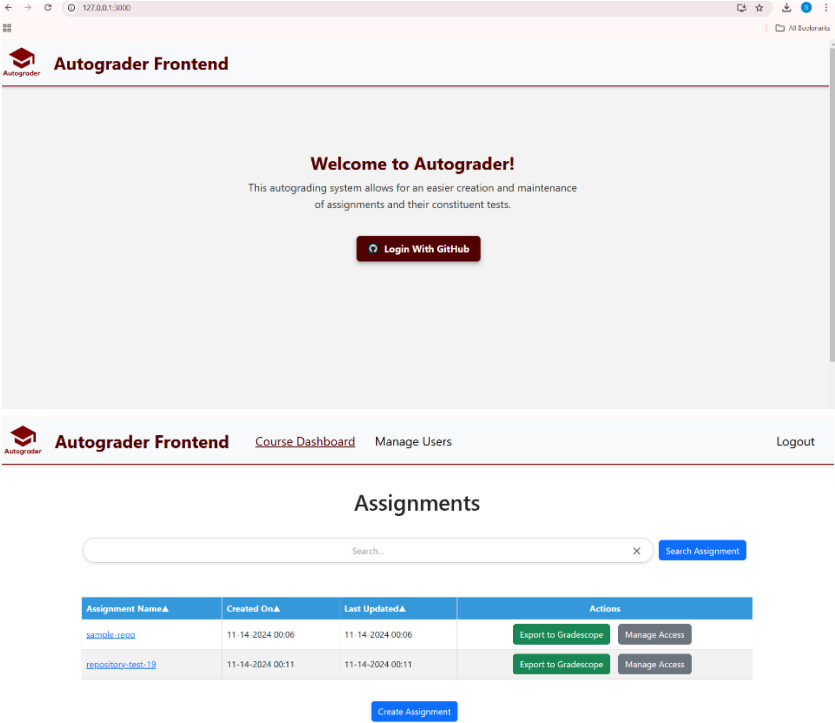
**Implementation Status:** Completed

**Changes:** None

**Mockups:**



Screenshots:



Story 11

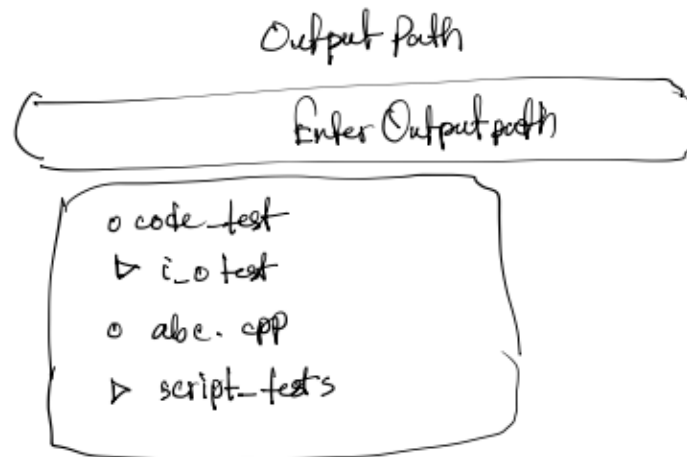
**Story Description:**  
Feature: Radio button for single-choice file selection  
As a CSE 120 organization member  
So that I can select the particular file I want to upload for single-choice fields  
I want the file-selection-tree dropdown to have a radio button beside each file name for selection

**Points:** 1

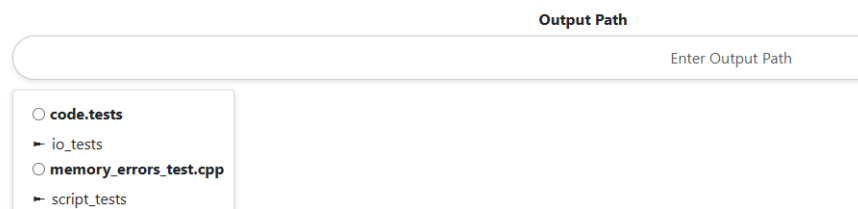
**Implementation Status:** Completed

**Changes:** None

### Mockups:



### Screenshots:



### Story 12

#### Story Description:

Feature: Move Tests b/t Groupings

As a CSCE 120 GitHub organization member,

So that I can change which grouping a test is in,

I want to be able to drag and drop tests from one grouping to another

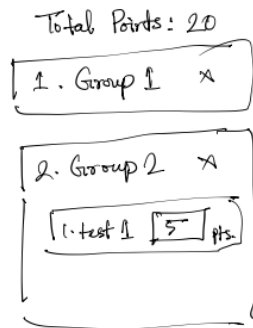
(Concatenate grouping & test number to produce test case numbers in code.tests)

**Points:** 1

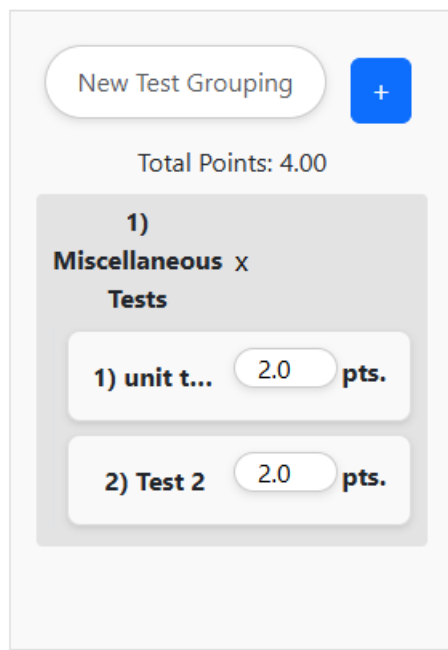
**Implementation Status:** Completed

**Changes:** None

### Mockups:



### Screenshots:



**Stories for future development:** Because of time constraints and being lower priority, the following user stories are left for future development.

<b>Story 1</b>	<b>Story Description:</b> Feature: Adding new deploy keys As an instructor So that I can ensure secure access to the assignment repositories, I want to have the ability to add new deploy keys for assignment repositories in the CSCE 120 organization.
----------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Story 2</b>	<b>Story Description:</b> Feature: Add GitHub user to organization As an administrator So that I can give new instructors and TAs access to the app I want to be able to add new GitHub users to the organization through the app
<b>Story 3</b>	<b>Story Description:</b> Feature: Remove GitHub user from organization As an administrator So that I can revoke instructors' and TAs' access to the app I want to be able to remove GitHub users from the organization through the app
<b>Story 4</b>	<b>Story Description:</b> Feature: Pull remote changes As a CSCE 120 GitHub organization member So that I can see modifications made (from outside of the app) to the remote repository I want the app to pull changes from the remote repository
<b>Story 5</b>	<b>Story Description:</b> Feature: Parse tests coming in from remote As a CSCE 120 GitHub organization member So that the app's database is updated with outside changes I want the app to parse outside code.tests file changes
<b>Story 6</b>	<b>Story Description:</b> Feature: Test refactoring As a developer So that I can more effectively test my code I want to have more flexible and usable Cucumber step definitions to mock the behavior of third-party services that the app communicates with
<b>Story 7</b>	<b>Story Description:</b> Feature: Add administrator role As the CSCE 120 course coordinator So that I can have a higher level of permission in the app I want to have an "administrator" role
<b>Story 8</b>	<b>Story Description:</b> Feature: User Can Edit approved_files As a CSCE 120 GitHub organization member So that I can manage the approved_files I want to be able to edit the approved_files allowed in the assignment

<b>Story 9</b>	<b>Story Description:</b> Feature: Approved Include Textbox As a CSCE 120 GitHub organization member So that I can only put one entry per a line I want to be limited and expected to put one entry per a line
----------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Team Roles

During each iteration of the project, we changed the role of the project owner and the scrum master and ensured that each member was assigned to at least one of these roles throughout the project cycle. The following table summarizes the team roles for each sprint:

<b>Sprint No.</b>	<b>Product Owner</b>	<b>Scrum Master</b>
Sprint 1	Max Smith	Md Hasan Al Banna
Sprint 2	Riddhi Ghat	Mainak Sarkar
Sprint 3	Saksham Mehta	Qinyao Hou
Sprint 4	Mainak Sarkar	Ryan Gonzalez

### Scrum Iterations

On each sprint, stories were added based on the customer requirements. The following table summarizes the accomplished tasks and the completed points:

<b>Sprint No.</b>	<b>Task</b>	<b>Points</b>	<b>Total</b>
Sprint 1	Create a Login System for the App	4	19
	Create an Access Control Management System	3	
	Create a New Assignment	3	
	Add a Test to an Assignment	1	
	Review Tests in an Assignment	1	
	Update test for an assignment	1	
	Regenerate .tests file on test creation/update/deletion	2	
	Update remote assignment repository	2	
	Export assignment as .zip	2	

Sprint 2	Search functionality on the assignments page	2	15
	Update Assignments view	1	
	Logout System	1	
	Navigation Bar	1	
	Test Case Groupings Model	1	
	Test Case Groupings CRUD	2	
	Manage user access for each assignment	2	
	Manage assignment-wise access for each user	2	
	Consistent button styling and clear labels	1	
	Fix production environment	2	
Sprint 3	Update Test Schema	2	14.5
	Test Case Editor UI Updates	1	
	Move tests Between Groupings	1	
	File Upload	3	
	Test Grouping UI Updates	2	
	Assignments Table UI Update	1	
	Implement Test Block Partial	1.5	
	File Browser Partial	2	
	Add Cucumber config to enable JS scenarios testing	1	
Sprint 4	File Selection Partial	2	18
	Includes field should be replaced with checkbox file selection	2	
	Test Editor UI Updates	1	
	Update Assignment schema to include approved_files attribute	1	
	Add field to set approved_files during assignment creation	1	
	"Create Test" should be grayed out until minimum necessary information is entered	2	



	Remove the existing edit points update form	3	
	Login Changes to take you dashboard rather than prompting you to login again	1	
	Set files_to_submit in run_autograder script	2	
	General UI changes	1	
	Radio button for single-choice file selection	1	
	Implement drag and drop between groups	1	
<b>Total</b>			66.5

### Member Contribution

The following table shows how many stories and points are completed by each team member on each sprint:

Members	Total Individual Points  Sprint 1 points are divided by 2 to stay consistent with how we calculated individual contribution in later sprints (Sprint 1 + Sprint 2 + Sprint 3 + Sprint 4)	Stories Contributed	Total Points Out of the Team Total (66.5)	Percentage of Development
Qinyao Hou	$2.5 + 1 + 1.5 + 3 = 8$	$4+1+3+3=11$	8	12.03%
Saksham Mehta	$2.5 + 2 + 2 + 2 = 8.5$	$2+3+2+3=10$	8.5	12.78%
Max Smith	$2.5 + 4 + 4.5 + 3 = 14$	$2+3+4+3=12$	14	21.05%
Md Hasan Al Banna	$2.5 + 1 + 2 + 2 = 7.5$	$4+2+2+3=11$	7.5	11.28%
Mainak Sarkar	$3.5 + 3 + 1 + 3 = 10.5$	$2+3+1+3=9$	10.5	15.79%
Ryan Gonzalez	$2 + 1.5 + 1.5 + 2 = 7$	$2+2+2+2=8$	7	10.53%
Riddhi Ghate	$3.5 + 2.5 + 2 + 3 = 11$	$2+3+1+3=9$	11	16.54%

## Customer Meetings

Here we provide list of customer meeting dates and description of what happened at the meetings.

Customer Meeting Dates	Descriptions
Fri Sep 20, 2024 10:30am – 11:30am (CDT)	<p>The meeting was mainly about the basic functionalities and workflows of the Autograder Frontend Software Application. The original expectation and the main customer's need is as follows:</p> <p><b>Project Specifications:</b></p> <ul style="list-style-type: none"><li>• Automation of new assignment repository</li><li>• Automation of key generation</li><li>• Must have C++ support, nice to have Java, Python</li><li>• Updated test case numbering when tests inserted into middle of document</li><li>• Test reordering support<ul style="list-style-type: none"><li>◦ Selection of test numbering scheme</li></ul></li><li>• Interfaces for creating different types of tests<ul style="list-style-type: none"><li>◦ Unit Test type is highest priority<ul style="list-style-type: none"><li>▪ Select b/t ASSERT/EXPECT</li></ul></li></ul></li><li>• Ability to upload/display several possible solutions and run test against solutions</li><li>• Exclusive access to a file between users</li><li>• Running C++/Bash in browser</li><li>• (Ability to manage user permissions from the app (for TAs and other authorized users)?)</li></ul>

<p>Fri Sep 27, 2024 11am - 12pm (CDT)</p>	<p>Based on the first meeting, the group members further discussed about the design and workflow of this application. During this we raised some specific questions that are important for the implementation that we put out these in this customer meeting. The questions and some points in the meeting are as follows:</p> <ul style="list-style-type: none"> <li>• Recurring meeting of Fridays @ 11:00am <ul style="list-style-type: none"> <li>◦ Weekly hybrid meetings in conference room on floor 3 of Peterson</li> </ul> </li> <li>• Mockups/stories <ul style="list-style-type: none"> <li>◦ Discuss what we plan to do for Sprint 1</li> <li>◦ Work out any issues</li> </ul> </li> <li>• Questions <ul style="list-style-type: none"> <li>◦ Hosting issue (must have backend as part of class requirements)</li> <li>◦ What is the difference between the Instructor/TA roles?</li> <li>◦ Can someone modify the assignments without using our app? <ul style="list-style-type: none"> <li>▪ If yes, this will be handled in a future sprint</li> </ul> </li> <li>◦ On what basis, should we add a new user? <ul style="list-style-type: none"> <li>▪ Currently, we are planning to draw from the GitHub organization roles (owner, manager) and use those to inform app permissioning</li> <li>▪ If a user is not in the GitHub organization they'd be directed to a page to request access</li> </ul> </li> <li>◦ Desired level of WCAG conformance (A, AA, or AAA)</li> </ul> </li> </ul>
-------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p>Mon Oct 7, 2024 2:30pm – 3:30pm (CDT)</p>	<p>We basically demoed the foundational aspects of the autograder application. Broadly speaking, our demo can be proliferated as follows:</p> <ul style="list-style-type: none"> <li>• Creating the Login System for our App.</li> <li>• Creating the Access Control Management System that allows the admin(s) of the organization to add new users and manage their level of access.</li> <li>• Creating the home page of the App, that includes the link to the form for creating new assignments.</li> <li>• Developing the CRUD operations for tests within individual assignments, and updating the “code.tests” specification file accordingly, depending on the operation performed.</li> <li>• Enabling the functionality that allows the assignment to be exported for upload to the autograder in Gradescope.</li> </ul> <p>Based on our demo, in our client meeting there are also feedbacks from our client as follows:</p> <ul style="list-style-type: none"> <li>• Misc. feedback <ul style="list-style-type: none"> <li>◦ Client wants addition of a navigation bar</li> <li>◦ Client wants more descriptive button names</li> <li>◦ Client wants more consistent styling</li> </ul> </li> <li>• Course overview page feedback <ul style="list-style-type: none"> <li>◦ Client wants search bar/sorted tabular entries for course assignments</li> <li>◦ Client wants removal of redundant data (Assignment Name)</li> <li>◦ Client wants ability to regenerate assignment deploy keys (and possibly clear existing)</li> </ul> </li> <li>• Test creation feedback <ul style="list-style-type: none"> <li>◦ Client wants ability to create groupings of tests</li> <li>◦ Client wants more organized view/formatting for creating/modifying tests</li> <li>◦ Client wants insertion of whitespace in between code.tests entries</li> </ul> </li> <li>• Access control feedback <ul style="list-style-type: none"> <li>◦ Client wants “matrix” of checkboxes to select specific permissions</li> <li>◦ Client wants both an assignment view (in which all users’ access can be modified for a specific assignment), as well as the existing user view (which allows an administrator to modify all of a user’s permissions for all assignments)</li> </ul> </li> </ul>
<p>Mon Oct 14, 2024 12pm – 1pm (CDT)</p>	<p>We briefly shared our sprint plan to the client and discussed about some of the stories for the next sprint. The customer stressed the expectation to have the dynamic test block to create and modify a test case in the front page easily.</p>

<p>Tue Oct 22, 2024 12pm – 1pm (CDT)</p>	<p>We demoed our enhancements based on Sprint 1, mainly focus on usability and create a more intuitive user interface on top of the core functionality established in Sprint 1. This is driven by the client’s desire for an intuitive and accessible application. The stories we demoed are as follows:</p> <ul style="list-style-type: none"> <li>• The search bar functionality to search for assignments by repo name</li> <li>• A tabular display for assignments with multiple attributes</li> <li>• Sorting functionality of the tabular display</li> <li>• Log out functionality of the Application</li> <li>• In the test editor page we add “Test Case Grouping” model to implement the functionality of operations based on test groups</li> <li>• Management of access to the assignment</li> <li>• UI changes from link to button in the assignment page</li> </ul> <p>Based on our demo, in our client meeting there are also feedbacks from our client as follows:</p> <ul style="list-style-type: none"> <li>• Misc. feedback <ul style="list-style-type: none"> <li>◦ Client wants the user to be rerouted directly to the course overview page, if they are logged in and they try to access the login page of the app.</li> </ul> </li> <li>• Course overview page feedback <ul style="list-style-type: none"> <li>◦ Client wants the assignment repository name to be linked to the Assignment Edit page.</li> <li>◦ Client wants to remove the “Edit” option from the Actions column.</li> <li>◦ Client wants the name of two sets of buttons to be changed, namely: “Export” button and “Manage Permissions” button.</li> <li>◦ Client wants to make some adjustments to the assignments table’s styling: (a) he wants the text to be left justified, and (b) he wants the sorted column to be highlighted.</li> </ul> </li> <li>• Individual Assignment Edit Page <ul style="list-style-type: none"> <li>◦ Client wants points to be edited outside of individual tests</li> <li>◦ Client prefers not to view points associated with individual tests alongside their corresponding test names.</li> <li>◦ Client wants the assignment edit page to be more organized, with consistent naming conventions.</li> </ul> </li> <li>• Test Grouping creation feedback <ul style="list-style-type: none"> <li>◦ Client wants to remove the ability to update grouping names</li> </ul> </li> <li>• Test Creation feedback <ul style="list-style-type: none"> <li>◦ Client wants to incorporate numbering to tests.</li> <li>◦ Client wants test-type to be static after initial creation (remove ability to edit test type in “Edit Test” feature)</li> <li>◦ Clients visual confirmation when a test is deleted.</li> </ul> </li> <li>• Dynamic Test Editor</li> </ul>
----------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Mon Oct 28, 2024 12pm – 1pm (CDT)	We demoed the successful deployment that we failed in the last meeting. And we shared our sprint plan for the following sprint with the client.
Mon Nov 4, 2024 12pm – 1pm (CST)	<p>Our modifications are based on the usability and functionality of the test case editor, as well as implementing UI changes requested by the client. Our stories demoed are as follows:</p> <ul style="list-style-type: none"> <li>• The dynamic test block partials rendering based on the test type selected</li> <li>• Update the test case editor page UI based on our wireframe designed by the client's requirement.</li> <li>• Drag and drop functionality between test cases within test groupings</li> <li>• Test grouping board UI modification including adding points editor for each test case, total points above and confirmation before deleting a test case</li> <li>• File upload feature that can upload files from the user's machine</li> <li>• Display file directory in dropdown format</li> </ul> <p>The client requested the following additions:</p> <ul style="list-style-type: none"> <li>• Radio button for single-choice file selection</li> <li>• Live update for points, rather than having to click several buttons</li> <li>• Includes field should be replaced with checkbox file selection</li> <li>• "Create Test" should be grayed out until minimum necessary information is entered</li> <li>• Asterisks should be next to all required fields</li> <li>• Add units to Timeout field</li> <li>• Assignment creation needs to make necessary updates to run_autograder script</li> <li>• Test case deletion should require confirmation</li> <li>• Approved includes can just be a textbox that expects one entry per line</li> </ul>
Mon Nov 11, 2024 12pm – 1pm (CST)	<p>We fix the bugs in file selection story Cucumber tests and display the file selection implementation to client and also we demo our sprint plan to the client. Then we discuss some issues about the specific requirements of client as follows:</p> <ul style="list-style-type: none"> <li>• Right justified points scroller</li> <li>• Display the total of points dynamically</li> <li>• Sum of points for each group (non-editable) (good to have, but not needed)</li> <li>• Change the label of Includes field → Provided Files</li> <li>• Drag and drop between the groups along with tests</li> <li>• Meeting on Monday/Tuesday post submission</li> </ul>

Mon Nov 18, 2024 12pm – 1pm (CST)	<p>We displayed our modifications to adhere to common UI conventions at the request of the client. We also added in missing functionality related to assignment creation that was brought to our attention by the client. The detailed information is as follows:</p> <ul style="list-style-type: none"> <li>• We refactor the include attribute to a dropdown UI rather than a text field</li> <li>• The logic to prevent user from hitting “Create Test” button while needed attributes are not completed.</li> <li>• Radio button for single-choice file selection</li> <li>• Dynamically updated test points editor</li> <li>• Addition of “approved_files” attribute when creating assignment</li> <li>• Set “files_to_submit” in “run_autograder” script</li> <li>• Drag and drop functionality of test groups</li> <li>• General UI changes</li> <li>• Fix the bugs of File selection partial</li> </ul> <p>The client requested the following additions:</p> <ul style="list-style-type: none"> <li>• Fix ssh key issue (see issue in GitHub)</li> <li>• Remove unused test editor fields</li> <li>• Ensure test card points are displayed with two decimal places</li> <li>• Includes not updating in code.tests</li> <li>• Make points not required (default to 0)</li> <li>• Approved files should be editable</li> <li>• Test case drag-and-drop not moving</li> <li>• Groupings should be movable</li> <li>• Redundant seconds in test editor</li> <li>• Fractional seconds for Timeout</li> <li>• Move Delete Test button to bottom</li> <li>• Rename to something like "Autograder Builder"</li> </ul>
-----------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## BDD/TDD Process & Benefits/Problems

Here we will describe how we utilize BDD/TDD methodologies to guide development.

### BDD/TDD Process:

**1. Test Creation:** Before implementing any functionalities, we began by writing RSpec tests to validate individual methods and models. For user-facing features, we created Cucumber scenarios to define expected behaviors based on user stories.

**2. Development Cycle:** We followed a Red-Green-Refactor cycle: writing failing tests (Red), implementing the minimum code to pass the tests (Green), and refactoring the code for clarity and efficiency.

**3. Integration of TDD and BDD:** while RSpec ensured the correctness of isolated components, Cucumber ensured that those components worked cohesively to meet user requirements. For example, when implementing the drag-and-drop feature for test reordering, RSpec tests validated the backend sorting logic, while Cucumber scenarios ensured the feature worked smoothly from the user's perspective.

**4. Automation of Tests:** By integrating RSpec and Cucumber into the CI workflow, tests run automatically, ensuring all new features or fixes pass before merging. Also we use SimpleCov gem to maintain above 90% coverage of Cucumber and RSpec tests.

#### **Benefits:**

- 1. Code Quality:** Writing tests first helped us identify edge cases early and ensured thorough validation of the code.
- 2. Clear Requirements:** Writing tests first provided clarity on user expectations and served as reminder while developing.

#### **Problems:**

- 1. Time Consumption:** Writing tests upfront required significant time and effort, especially during initial sprints.
- 2. Evolving overhead:** Occasionally, we encountered unexpected Cucumber tests issues or unit test issues. For example the browser or JavaScript execution timing issues. Thus we need to evolving user stories and rewriting or modifying tests frequently, which added overhead.

### **Configuration Management Approach**

Our project relied on effective configuration management to ensure smooth collaboration, consistent environments, and reliable software delivery. This included managing version control, development environments, branching strategies, and releases.

#### **Configuration Management:**

- 1. Version Control:** We used Git and Github for version control, ensuring that every team member worked on the latest version of the code base. Our branching strategy included: A main branch for most updated code base and feature branches for implementing individual tasks or user stories. Code changes in story branches were reviewed and merged through pull requests, with automated CI workflows testing each modifications.
- 2. Environment Management:** We maintained consistent development environments using shared **config/local\_env.yml** file for environment variables. Also by adding CLIENT\_ID and CLIENT\_SECRET variables into the yml file to activate specific client access while developing.



**3. CI Integration:** Continuous Integration workflows in GitHub Actions ran RSpec and Cucumber tests automatically, ensuring all code changes met quality standards before merging. It also runs rubocop to ensure no code offenses.

**4. Dependency Management:** All the Dependencies were managed using Bundler to ensure the same versions of libraries and gems across all team members and deployment environments. Member can simply run bundle install to meet with all the dependencies.

**Branches:** Throughout the project, we created story-based branches, most of them are tied to a specific task or user story. The naming format of story branches is: feature/story\_number-story\_name (For example: feature/62-dynamic-test-block). The story branches act as the main branch of each story and it can have several sub-branches based on specific tasks of each story. We use this naming strategy to maintain all the updates and merges are in order.

**Releases:** We followed a sprint-based release strategy. A release was made at the end of each sprint, with new features deployed to the staging environment for testing.

## Heroku Deployment Issues

During the production release process to Heroku, we encountered an issue related to Heroku's ephemeral storage. Heroku provides a temporary filesystem, which is cleared periodically during dyno restarts or scaling operations. This caused disruptions for our app, which relies on local file storage for critical operations.

For instance, after an assignment was created, if a user attempted to create a test or export a zip file, the application would throw an error because the repository required for these operations was no longer present locally. This occurred because Heroku's ephemeral storage removed the repository files during a dyno restart.

To address this issue, we implemented a solution where the app checks if the repository exists locally before performing the requested operation. If the repository is missing, the app first clones the repository from the remote source and then proceeds with the user's request. This workaround ensured that the app could function reliably even with Heroku's ephemeral storage limitations.

While this issue added complexity to our deployment process, it highlighted the importance of designing our application to adapt to Heroku's unique architecture. This experience also underscored the value of thorough testing in identifying and resolving deployment-specific issues.

## Tools/Gems

Our project relied on a variety of tools and gems to streamline development, enhance collaboration, and ensure code quality. These tools addressed different aspects of the development lifecycle, including version control, testing, debugging, and performance monitoring.

Gem/Tool Name	Description	Benefits	Problems/Challenges
dotenv-rails	Manages environment variables securely using <code>.env</code> files.	Ensures sensitive data (e.g., API keys) is not hardcoded into the application.	Requires team members to maintain consistent <code>.env</code> files locally.
web-console	Provides a Rails console in the browser for debugging during development.	Speeds up debugging by allowing real-time code evaluation.	Only works in development mode, so it cannot assist in production debugging.
RSpec	A testing framework for Ruby, focused on unit and integration tests.	Ensures code correctness and facilitates TDD practices.	Writing detailed tests can be time-consuming.
Cucumber	Enables Behavior-Driven Development (BDD) through human-readable feature files.	Improves collaboration by focusing on user behavior.	Managing changes in feature files as requirements evolve can become challenging.
Capybara	Simulates real user interactions for feature testing.	Enables automated testing of user flows, reducing manual effort.	Tests can become flaky due to timing issues with JavaScript execution.
SimpleCov	Provides test coverage reports, highlighting untested code.	Helps maintain high test coverage and identify areas needing tests.	Coverage reports may require additional configuration to exclude irrelevant files.
WebMock	Allows mocking of HTTP requests and responses in tests.	Ensures tests are isolated from external APIs and network issues.	Requires careful setup to mock all external requests accurately.
Brakeman	A static analysis tool to detect security vulnerabilities in Rails applications.	Identifies potential security issues early in development.	Occasionally reports false positives, requiring manual review.

turbo-rails	Provides fast, single-page-like interactivity for Rails apps without heavy JavaScript frameworks.	Simplifies creating modern, interactive UIs with minimal JavaScript.	Some interactions require custom solutions, which can add complexity.
stimulus-rails	Adds a lightweight JavaScript framework to handle interactivity.	Makes JavaScript modular and reusable, simplifying frontend development.	Requires some learning curve for developers new to the framework.
octokit	A Ruby client for the GitHub API.	Simplifies interaction with GitHub, such as automating repository tasks.	Requires careful handling of API authentication and rate limits.
pg	Integrates PostgreSQL as the database for production environments.	Provides a robust and scalable database solution for production.	Setup can be more complex compared to SQLite in development.
aws-sdk-s3	A Ruby SDK to interact with Amazon S3 for file storage.	Allows secure and scalable file storage for production environments.	Can incur additional costs if not properly monitored.
puma	A web server designed for high concurrency and performance.	Handles multiple requests efficiently, ensuring faster response times.	Requires fine-tuning for optimal performance in high-load scenarios.
acts_as_list	Manages lists in Active Record models with position tracking.	Simplifies list management for sortable models.	Requires careful handling of reordering to avoid performance issues with large datasets.
omniauth	A flexible authentication library for integrating third-party login services.	Eases integration with OAuth providers like GitHub.	Requires proper configuration for security and compliance with provider-specific requirements.
omniauth-github	Provides GitHub-specific OmniAuth strategies for authentication.	Allows users to authenticate via GitHub easily.	Dependent on GitHub API rate limits and availability.
omniauth-rails_csrf_protection	Adds CSRF protection for OmniAuth requests in Rails applications.	Enhances security by preventing CSRF attacks during authentication.	Requires careful testing to ensure it doesn't interfere with login flows.

faraday-retry	Adds retry functionality to Faraday HTTP client requests.	Improves resilience by handling transient network failures automatically.	Overuse of retries can lead to unnecessary delays or masking of real issues.
faraday-multipart	Enables multipart form data handling in Faraday HTTP requests.	Simplifies file uploads via HTTP requests.	Requires careful configuration for file size limits and content types.
git	Provides access to Git commands via Ruby.	Allows programmatic interaction with Git repositories.	May require manual installation of Git on the system to work properly.
rubyzip	Handles ZIP file creation and extraction in Ruby applications.	Simplifies working with compressed files in the application.	Large ZIP files may require additional memory or processing time.
selenium-webdriver	Provides browser automation for system and integration tests.	Allows comprehensive testing of user interactions in real browsers.	Can cause flaky tests due to browser or environment-specific issues.
database_cleaner	Manages test database cleanup between test runs.	Ensures a consistent state for tests by cleaning the database.	Requires careful setup to avoid accidentally cleaning production data.
rubocop	A Ruby code linter and formatter.	Helps enforce coding standards and style guidelines.	Strict rules can sometimes lead to lengthy code revisions.
rubocop-rails	Adds Rails-specific rules to RuboCop.	Ensures Rails-specific best practices are followed.	Rules may conflict with project-specific conventions.
rubocop-performance	Adds performance-focused linting rules to RuboCop.	Identifies potential performance bottlenecks in the code.	May require developer experience to validate the recommendations.
rubocop-rspec	Adds RSpec-specific linting rules to RuboCop.	Ensures consistent and best practices for RSpec tests.	Can be overly strict for some test styles.
rubycritic	Generates code quality reports with maintainability scores.	Provides an overview of code health and areas for improvement.	Reports may require additional context to interpret effectively.

GitHub	A version control and collaboration platform for managing code repositories	Simplifies collaboration with features like pull requests, issues, and CI workflows via GitHub Actions	Requires clear team communication to avoid merge conflicts or overlapping work.
Code Climate	A tool for static analysis and maintainability tracking of code bases	Provides metrics for code quality, test coverage, and maintainability, enabling better code health decisions.	Reports may require manual review to filter out false positives or adjust configurations to match the project.

## Repository Contents and Deployment Process

**Repository Contents:** The repository is organized to streamline development and deployment. Key directories and files include:

1. **/app:** Contains the core application logic, including models, controllers, and views.
2. **/config:** Holds application and deployment configurations, including database.yml and environment files.
3. **/db:** Contains database migrations and schema definitions.
4. **/Gemfile:** Lists all Ruby dependencies for development and production environments.
5. **.github/:** Configures GitHub Actions for CI/CD, automating testing and deployments.
6. **Procfile:** Defines how the application runs in Heroku.

**Deployment Process:** The deployment process involves preparing the repository, configuring the environment, and deploying the application to Heroku. Below are the detailed steps:

### 1. Prerequisites

- A Heroku account with access to the Heroku CLI.
- A GitHub organization with OAuth credentials for authentication.
- GitHub repositories for the application and core functionality, forked into the organization.

### 2. Steps for Deployment

#### - Set up Heroku:

1. Create a Heroku application using “heroku create <app-name>”.
2. Configure the stack to heroku-22 using “heroku stack:set heroku-22” to ensure Git is available.
3. Attach a PostgreSQL database using “heroku addons:create heroku-postgresql:essential-0”.

**- Prepare the Repository:**

1. Add the “pg” gem in the Gemfile under the “:production” group.
2. Create a Procfile if not already present, with the content: “web: bundle exec puma -C config/puma.rb”.

**- Push to Heroku:**

1. Add Heroku as a remote repository using “heroku git:remote -a <app-name>”.
2. Push the main branch to Heroku using “git push heroku main”.

**- Run Post-Deployment Tasks:**

1. Apply database migrations: “heroku run rails db:migrate”.
2. Seed the database if needed: “heroku run rails db:seed”.
3. Restart dynos using “heroku restart”.

**3. Post-Deployment Tasks**

**- Connect GitHub OAuth for user authentication:**

1. Register a new OAuth app in GitHub with the callback URL matching the Heroku application URL.
2. Add the GitHub OAuth credentials (GITHUB\_CLIENT\_ID and GITHUB\_CLIENT\_SECRET) as Heroku config variables.

**- Verify the application:**

1. Open the application in the browser using “heroku open”.
2. Check logs for any issues using “heroku logs —tail”.

**Links**

[Slack Invite](#)

[GitHub Repo](#)

[GitHub Projects](#)

[Heroku Deployment](#)

[Presentation & Demo Video](#)