Providing the Ability to Author Scaffolds within ASSISTments

An Interactive Qualifying Project
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science

by
Smera Gora, Shannen Lin, Frank McShan, and Molly Sunray

Date:
May 3, 2022

Submitted to:
Professor Neil Heffernan
Worcester Polytechnic Institute

Table of Contents

Abstract	3
Acknowledgments	4
Introduction	5
Backend	7
Frontend	14
Next Steps	27
Recommendations	31
References	33

Abstract

This Interactive Qualifying Project (IQP) focuses on implementing Scaffolding within the newest version of ASSISTments. ASSISTments is an online learning platform where students receive immediate support and feedback on their assignments, and in return, teachers gain insight on how to best meet the needs of their students. One feature that enables this is Scaffolding, a student support that divides a problem into a set of smaller problems, which together provide a step-by-step process for solving the original problem. In this IQP, we interacted with the ASSISTments database to implement Scaffolds in the backend, send requests to the frontend, and create a view for users to create, update, and delete Scaffolds through a user interface.

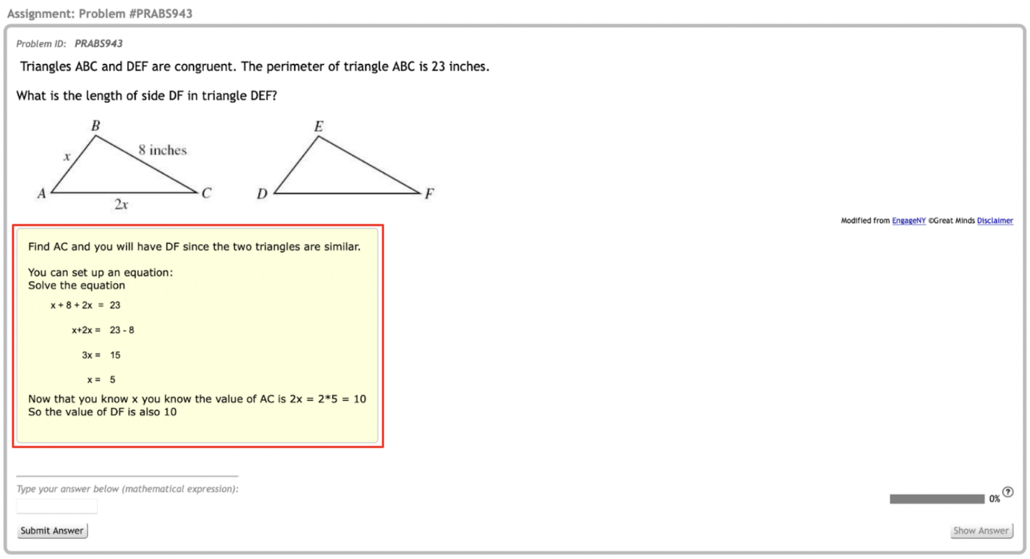
Acknowledgments

We would like to thank Professor Neil Heffernan for giving us the opportunity to work on a project within ASSISTments and for his continued guidance. We would also like to thank Aaron Haim for all of his assistance in helping us understand the code base and technologies used within ASSISTments.

Introduction

ASSISTments, an online learning tool founded by Dr. Neil Heffernan and Mrs. Cristina Heffernan, increases the rate at which students learn, ultimately making classwork more effective for both students and teachers. Students receive immediate feedback when using ASSISTments to complete assignments. At the same time, teachers receive insightful data, allowing them to tailor their class structure to the needs of their students.

When constructing assignments within ASSISTments, teachers are able to create unique problem sets consisting of a collection of related questions and answers. Additionally, teachers have the option of looking through the ASSISTments database of problems that other teachers have already created. To better integrate ASSISTments within their teaching style, teachers also have the option of integrating LMS services such as Canvas or Google Classroom, allowing assignments created through ASSISTments to appear as homework for students on these sites.



Assignment: Problem #PRABS943

Problem ID: PRABS943

Triangles ABC and DEF are congruent. The perimeter of triangle ABC is 23 inches.

What is the length of side DF in triangle DEF?

Find AC and you will have DF since the two triangles are similar.

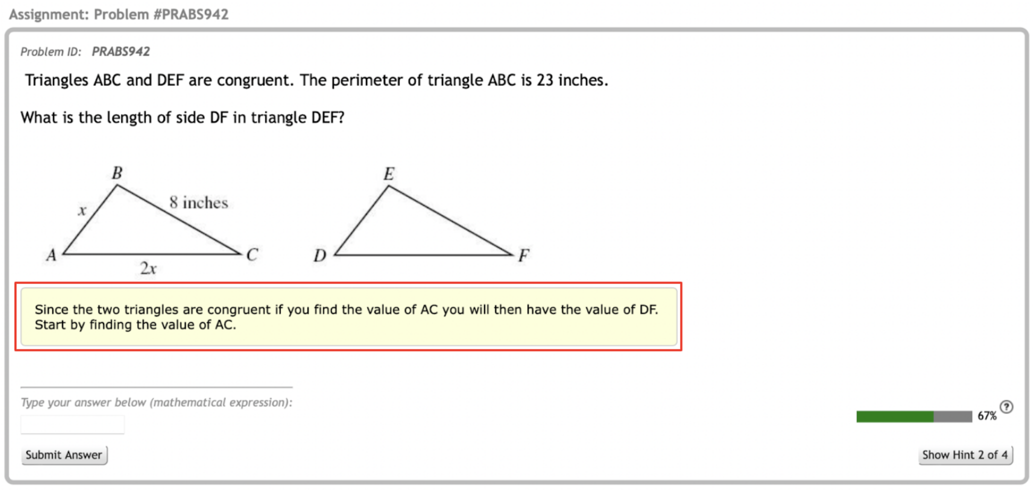
You can set up an equation:
Solve the equation

```
x + 8 + 2x  =  23

     x+2x =  23 - 8

       3x =  15

        x =  5
```

Now that you know x you know the value of AC is 2x = 2*5 = 10
So the value of DF is also 10

Type your answer below (mathematical expression):

Figure 1: An explanation within ASSISTments. To augment student learning, teachers are able to create explanations for a problem, making it easier for students to see how to arrive at the answer to the problem.



Assignment: Problem #PRABS942

Problem ID: PRABS942

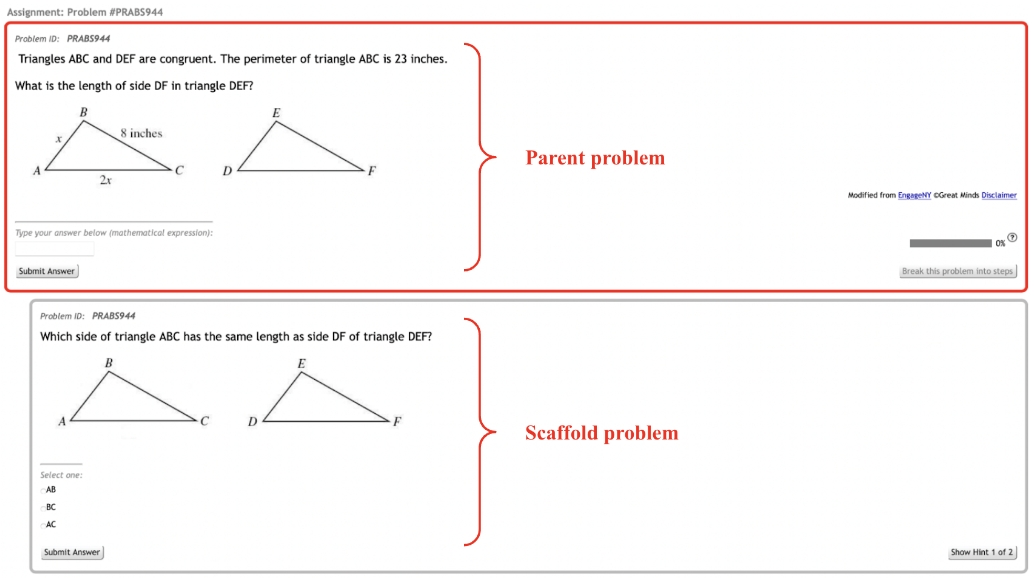
Triangles ABC and DEF are congruent. The perimeter of triangle ABC is 23 inches.

What is the length of side DF in triangle DEF?

Since the two triangles are congruent if you find the value of AC you will then have the value of DF. Start by finding the value of AC.

Type your answer below (mathematical expression):

Figure 2: A hint message within ASSISTments. Students can be provided with a series of hints to help them in solving a problem.



Assignment: Problem #PRABS944

Problem ID: PRABS944

Triangles ABC and DEF are congruent. The perimeter of triangle ABC is 23 inches.

What is the length of side DF in triangle DEF?

Type your answer below (mathematical expression):

Modified from EngageNY ©Great Minds Disclaimer

Problem ID: PRABS944

Which side of triangle ABC has the same length as side DF of triangle DEF?

Select one:

AB

BC

AC

Figure 3: An example of a Scaffolding question within ASSISTments 1.0. If having trouble with a specific problem, students are able to break it down into smaller steps, allowing them to more easily complete the problem.

With a Problem Set in ASSISTments, teachers have several opportunities to better their students' overall learning experience through tools called Student Supports which include Explanations, Hint Sets, and Scaffolding. Explanations allow teachers to explain why a certain answer is correct and why another answer may be incorrect. Hint Sets allow teachers to assist students in arriving at an answer to a problem they may be having trouble with without providing them with the actual answer. Finally, Scaffolding allows teachers to break a problem down into smaller steps, making a problem simple that may initially seem quite daunting. For example, if a question involves a series of calculations, a teacher has the option of creating a Scaffold for each calculation. As seen in Figure 3, this Scaffolding question can appear if a student answers a question incorrectly, allowing them to complete the problem step by step and thus having a higher chance of actually completing the problem. Even when students are completing a Scaffolding problem, teachers are provided with meaningful data to allow them to better tailor learning to their students.

In this Interactive Qualifying Project, we were tasked with introducing support for Scaffolding within the newest version of ASSISTments. The first version of ASSISTments was utilized for over 15 years, but in 2020, ASSISTments 2.0 was released, bringing a total redesign to the service. Although the first version of ASSISTments included support for Scaffolding, ASSISTments 2.0 did not upon release. In this project, we were tasked with carrying over support for Scaffolding to the new system. To implement Scaffolding, we needed to implement code within both the ASSISTments backend code to save Scaffolds to the database and the ASSISTments frontend, providing users with a method of intuitively creating and designing a Scaffold.

Backend

Databases

Interacting with the backend required knowledge of two databases within ASSISTments. Scaffolds within the first and also original database, which is known as the legacy or published database, is represented by three tables: Scaffolds, Problems, and Tutor Strategies. A Problem consists of an ID, a Scaffold ID, a position, and other additional information. The Scaffold ID field on the problem references the ID of a Scaffold. The position field indicates at which position the Problem is located in the list of Problems that represent a Scaffold. The Scaffold is a type of Tutor Strategy, and contains a field referencing a Tutor Strategy in the database. Tutor Strategies, also known as Student Supports, include Explanations, Hint Sets, and Scaffolding. Within the backend, a Tutor Strategy is represented with an ID, a title, and a strategy type. The strategy type refers to either Explanations, Hint Sets, or Scaffolding.

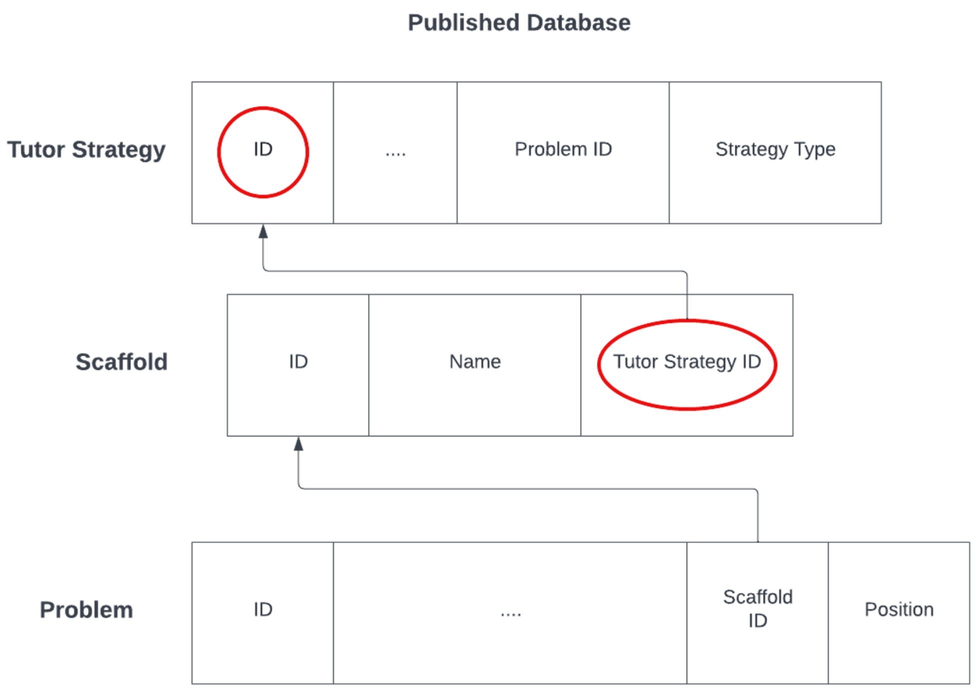


Figure 4: A diagram showcasing the various components of the published database and their respective interactions with each other.

The second and also the new database that was utilized was the draft database. The main difference with the draft database as compared to the published database is that the Scaffold in the draft database includes a problem set ID. This ID refers to the list of problems that are contained within the Scaffold. Similar to the published database, the Scaffold also contains the tutor strategy ID field which links the Scaffold to its Tutor Strategy object.

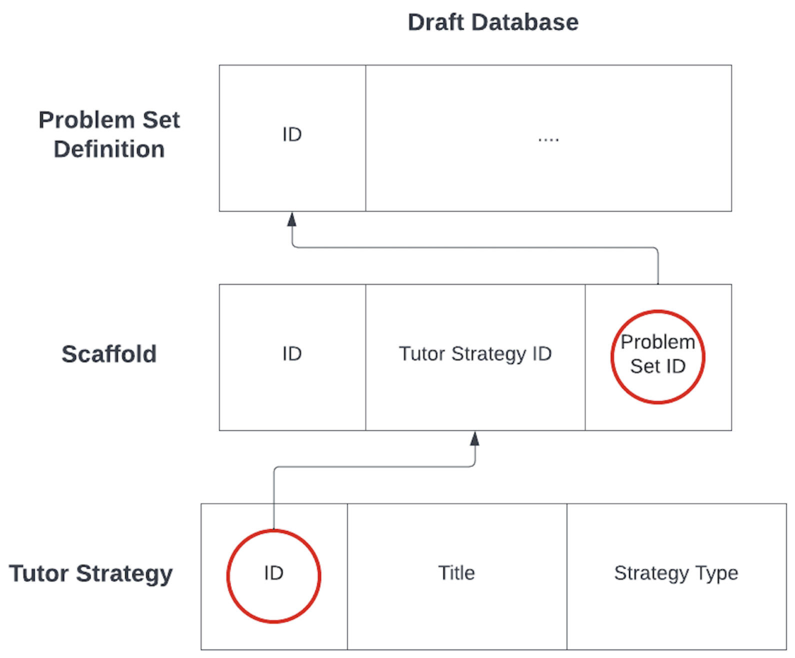


Figure 5: A diagram showcasing the various components of the draft database and their respective interactions with each other.

In this project, we worked with elements of both the frontend and backend. Starting with the backend, the main issue is that with the published database, Scaffolding problems are incorrectly sided. This means that in the published database, the problems within the Scaffold reference the Scaffold they are a part of through the Scaffold ID field, instead of having the Scaffold reference the problems that are contained within it. Within the draft version, this was modified with the addition of the problem set ID field on the Scaffold object, which references the list of problems that are in the Scaffold. In this project, we sought out to address the differences between the two databases by allowing users to add Scaffolds to either database. With the frontend, we were tasked with creating a user interface to let users create a problem set for a Scaffold and have the option to save these problems to either the draft or published databases. Users should also be able to preview their created problems and have the options to delete or edit them individually. Unfortunately, we were only able to accomplish implementing Scaffolds with the published database.

Frameworks

To better understand the existing codebase, it was necessary to familiarize ourselves with a few frameworks. One of these frameworks is Spring, “a Java platform that provides comprehensive infrastructure support for developing Java applications” (Johnson et al.) according to the Spring Framework Reference Documentation. In Java, a class provides a template for the attributes and behavior of objects of its type. For example, if we were to create an object for a Person class, the attributes of the object could be first name, last name, and date of birth, and one behavior could be calculating the age of the person based on the date of birth and the current date. An object is an instance of a class with specific attributes and behaviors (Oracle, 2022). An object of type Person could be created with first name “John,” last name “Smith,” and date of birth “January 1, 1990.” It is often the case that one class will need to rely, or depend, on instances of another class or multiple classes in order to perform some sort of function. Similarly, each of those classes might also need to depend on other classes in order to function properly, and may cause the code to become very complicated. The idea that a class depends on its many dependencies is one of the main things Spring aims to solve. The Spring container creates objects so that the developer does not have to manually construct them. These objects that are managed by the container are called beans. The managers and data access objects used are injected into their necessary locations in the Spring application.

```
{
    "problemList": [
        {
            "body":"What's 1 + 1?",
            "type": "1",
            "answers": [
                {
                    "value": "2",
                    "isCorrect": true
                }
            ]
        },
        {
            "body":"What's 4 - 3?",
            "type": "1",
            "answers": [
                {
                    "value": "4",
                    "isCorrect": false
                },
                {
                    "value": "1",
                    "isCorrect": true
                }
            ]
        }
    ]
}
```

Figure 6: A list of problems in JSON format.

Additionally, we utilized Jackson Databind, a framework used to encode and decode objects from JSON formats. JSON stands for JavaScript Object Notation, which is a text-based format for structuring data, making it easier to transmit data between the frontend and the backend. Figure 6 is an example of part of a JSON string that gets passed through a request. In this figure we can see that the JSON maps a variable, for example “body”, to a value, as in “What’s 1 + 1?”. To perform the conversion, the content of the JSON text is read into an object of the desired class. Similarly, an object can be written into a JSON format.

Controller Class and Deserialization

The Student Support Authoring Service is a service that handles authoring, editing, and deleting student supports, such as Scaffolds, in ASSISTments. A controller receives input and passes it on to managers. The managers define the application behavior and communicate user actions to the database. These user actions include any actions the user takes through the user interface. When a user saves a Scaffold on the frontend, the backend updates the database with the new content. In the Student Support Authoring Service, the main controller class is responsible for handling any interactions with supports, including the calls for getting, creating, updating, and removing supports. The controller calls on the manager to handle the logic and implementation of these requests.

To perform these commands in the backend code, the given JSON object passed in through the request is first deserialized into the desired object using Jackson Databind. The data from JSON is stored in an ObjectNode, which maps each field from the JSON object to their value. One of the values that is necessary to pass to the backend through the request is the content type, which represents one of the Student Supports, i.e. Explanations, Hint Sets, and Scaffolding. This content type is retrieved from the ObjectNode, telling us whether the request is for Scaffolding or one of the other supports. The content type of type Scaffolding tells us to create a Scaffold object and uses the fields retrieved from the node to populate the necessary fields for the Scaffold object. Depending on the type of request, the values that we retrieve from the request might differ. For example, when creating a Scaffold, the content key, or unique identifier, of the parent problem is necessary to be passed in through the request.

Create, Get, Remove, and Update

To create a Scaffold, we first create a Tutor Strategy object, which holds the ID of the problem that we are adding a Scaffold to. We also create a Scaffold object, which is populated with a generated name value and also contains the ID for the Tutor Strategy it is referencing. If the Scaffold object is successfully pushed to the database, we also persist, or add to the database, the problems in the Scaffold. Before persisting, each problem in the list is given the ID of the Scaffold so that they can be identified as part of the Scaffold. The Scaffold ID field on the Problem object is a reference to the ID of the Tutor Strategy in the published database and groups problems together that make up the same Scaffold. Persisting each problem returns the ID of the problem, and this ID is then used to link the answers to the problem when we add the answers to the database. After the problem ID is set for each answer, the list of answers is also persisted to the database. Finally, a response containing a unique identifier for the Scaffold is returned to the frontend.

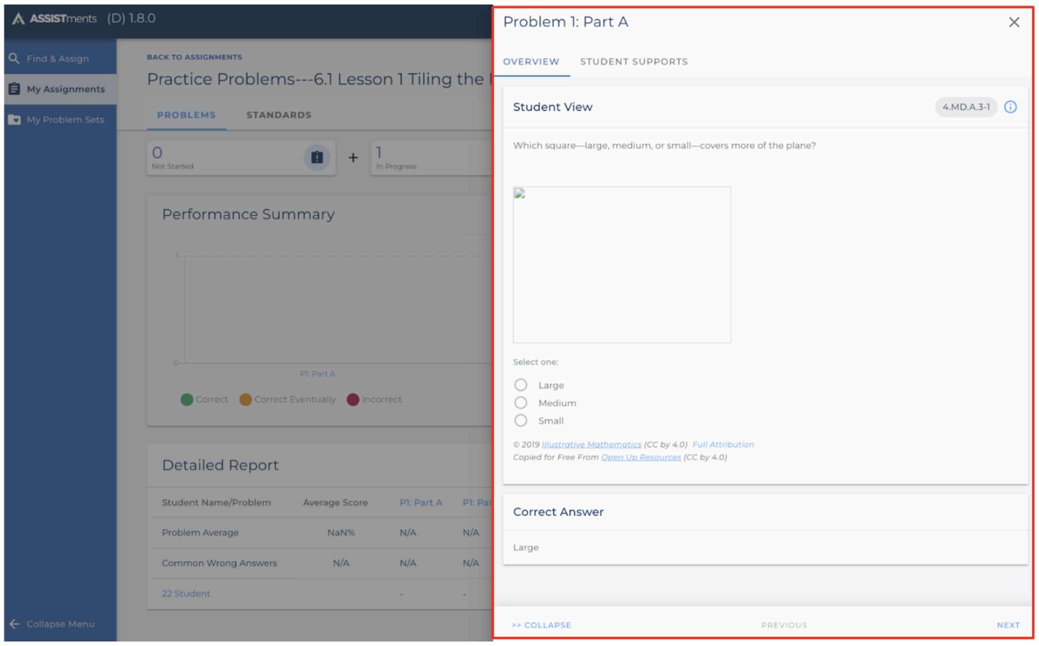
By passing in the given problem ID through the request, we can retrieve the Scaffold on that problem from the database. We first query, or search, the database for tutor strategies that match the problem ID and are of the Scaffold type. With this Scaffold, we then search the database for the list of problems where each problem's Scaffold ID matches the tutor strategy ID. The Scaffold ID field that is on a problem is a reference to a tutor strategy. We also retrieve the answers for each problem by querying the database for answers whose problem ID matches the IDs of the problems in our Scaffold. This pairing of a problem and its answers is added to a list and returned as a Scaffold.

To remove a Scaffold, the ID of the Scaffold is retrieved from its content key. Using this ID, the support can be disabled by setting its enabled field to false.

Updating a Scaffold involves both create and delete commands. The Scaffold is created in the same way as described above. Once the support is created with the necessary changes, the original Scaffold is removed from the database, again by disabling the support as mentioned above.

Frontend

ASSISTments mainly uses Vue and Typescript for the frontend. Vue.js is a frontend development framework used to build user interfaces and web applications (Vue.js, 2022). The main features of Vue include making components, which are similar to custom elements that can be reused throughout the app. It uses the standard frontend languages such as HTML, CSS, and JavaScript. Typescript is an object-oriented language that offers additional capabilities, including support for static typing and classes (TypeScript, 2022). It builds off of JavaScript, a popular language used to program functionality into web pages.



Student Name/Problem	Average Score	P1: Part A	P1: Par
Problem Average	NaN%	N/A	N/A
Common Wrong Answers	N/A	N/A	N/A
22 Student		-	-

Which square—large, medium, or small—covers more of the plane?

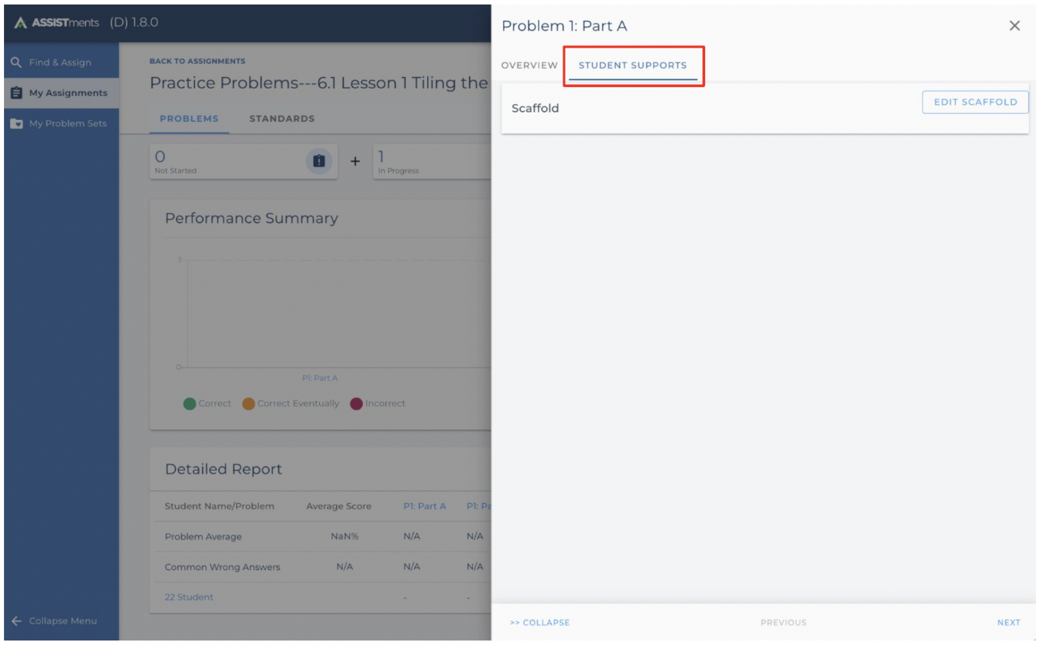
Select one:

- Large
- Medium
- Small

© 2019 Illustrative Mathematics (CC by 4.0) Full Attribution
Copied for Free From Open Up Resources (CC by 4.0)

Correct Answer: Large

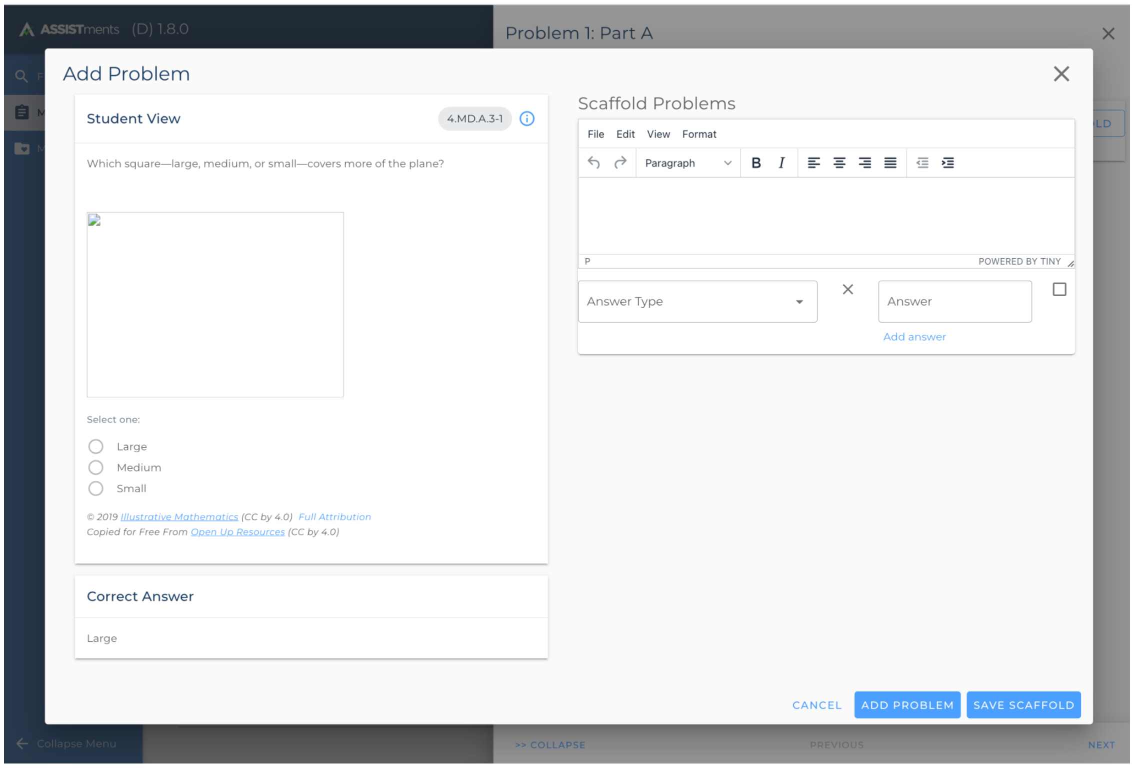
Figure 7: When a teacher views an assignment and selects one of the problems in the assignment, a sidebar will appear showing the problem.



Student Name/Problem	Average Score	P1: Part A	P1: P
Problem Average	NaN%	N/A	N/A
Common Wrong Answers	N/A	N/A	N/A
22 Student		-	-

Figure 8: We added another tab to this sidebar to allow teachers to add and edit Scaffolds.

To develop the frontend for our Scaffolding feature, we added on to the existing Teacher Experience user interface in ASSISTments. The user interface in Figure 7 shows a sidebar that opens when a teacher wishes to view a specific problem. As Scaffolding involves adding nested problems to a specific problem, we added another tab to this view shown in Figure 8 that allows teachers to add student supports, specifically Scaffold problems, to the current problem. In the Student Supports tab, users are able to select “Edit Scaffold”, which will open a dialog box, more commonly known as a pop-up window, allowing users to build and edit Scaffold problems.



Which square—large, medium, or small—covers more of the plane?

Select one:

- Large
- Medium
- Small

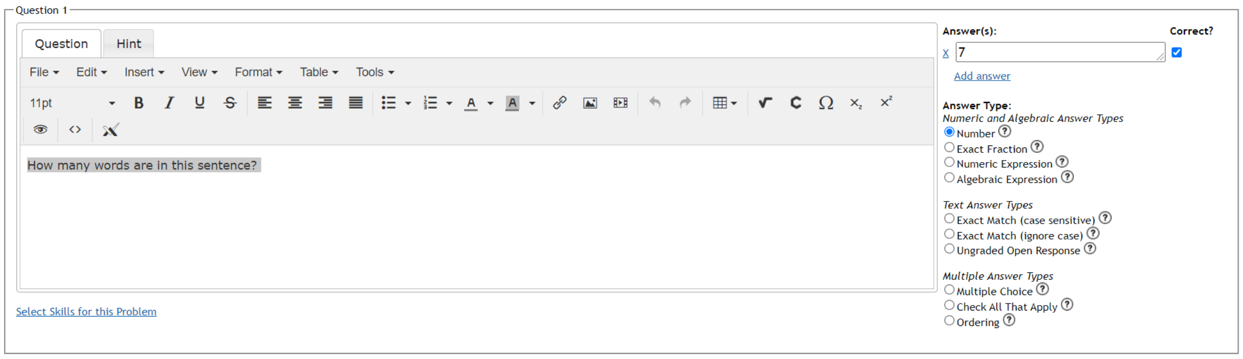
© 2019 Illustrative Mathematics (CC by 4.0) Full Attribution
Copied for Free From Open Up Resources (CC by 4.0)

Correct Answer

Large

Figure 9: The Scaffold dialog box, in which users can add and edit problems in the Scaffold.

The pop-up window in Figure 9 shows all the code related to building Scaffold problems. It is divided into two halves. On the left-hand side of the box, the teacher can view the current problem and the answers. As mentioned before, one of the main features of Vue is components. If we look at Figure 7 and Figure 9, we can see that they are both using the same problem view. Once a component is created, you can reuse it as a template in other parts of the code. The right-hand side of the dialog box is where the teacher can add Scaffold problems using our problem editor.



How many words are in this sentence?

Answer(s): 7

Correct? ✓

Answer Type:

Numeric and Algebraic Answer Types
- ● Number
- ○ Exact Fraction
- ○ Numeric Expression
- ○ Algebraic Expression

Text Answer Types
- ○ Exact Match (case sensitive)
- ○ Exact Match (ignore case)
- ○ Ungraded Open Response

Multiple Answer Types
- ○ Multiple Choice
- ○ Check All That Apply
- ○ Ordering

Figure 10: The problem builder in version 1 of ASSISTments.

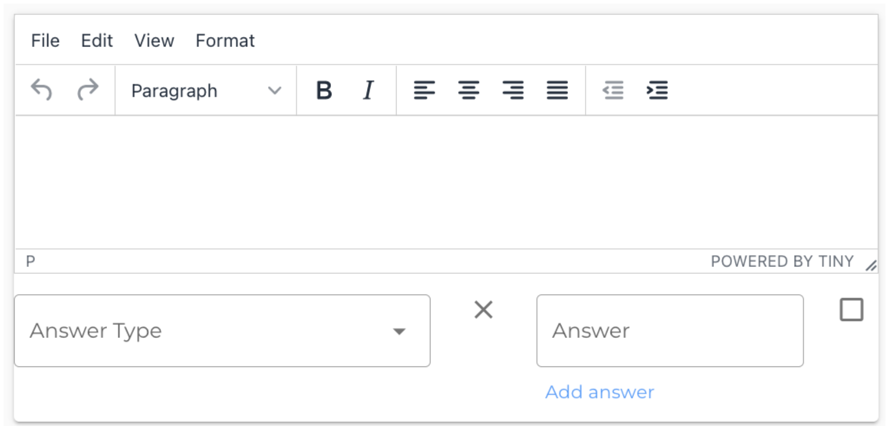


Figure 11: Our problem editor.

The problem editor was another component view we created that allows the user to build Scaffold problems. For the content of our problem editor, we referenced the problem builder used in version 1 of ASSISTments as shown in Figure 10. One thing we needed to add was a text editor for the user to write the problem. We decided to use TinyMCE, an open-source rich text editor, to take in user input. The problem editor we created is in Figure 11.

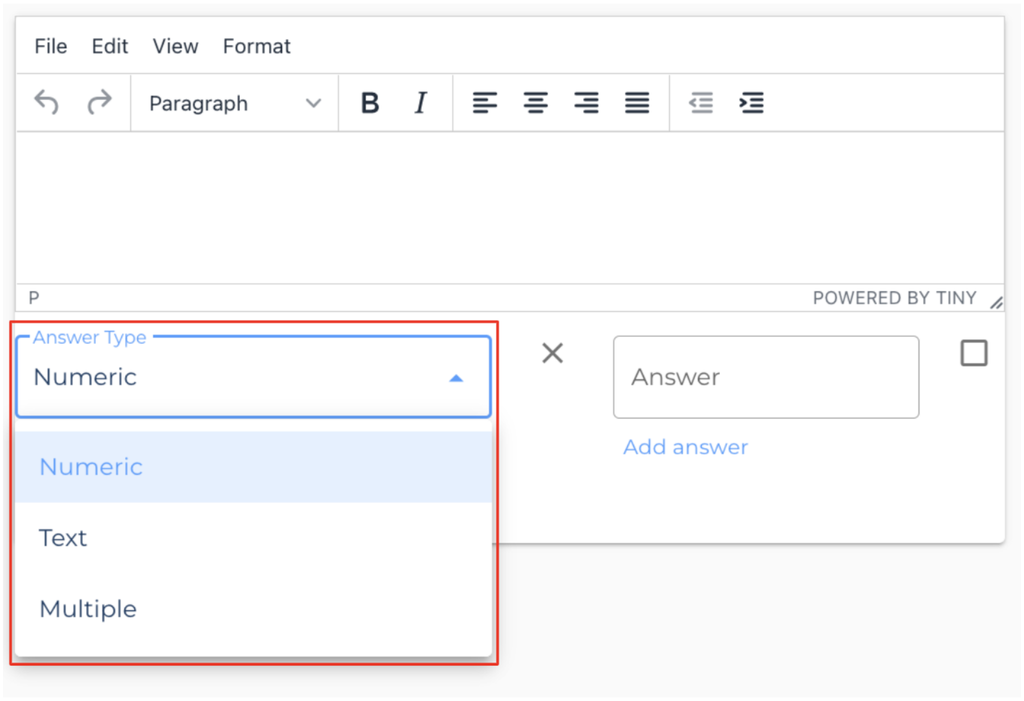


Figure 12: From the answer dropdown, the user can select among the 3 types of answers: Numeric, Text, and Multiple.

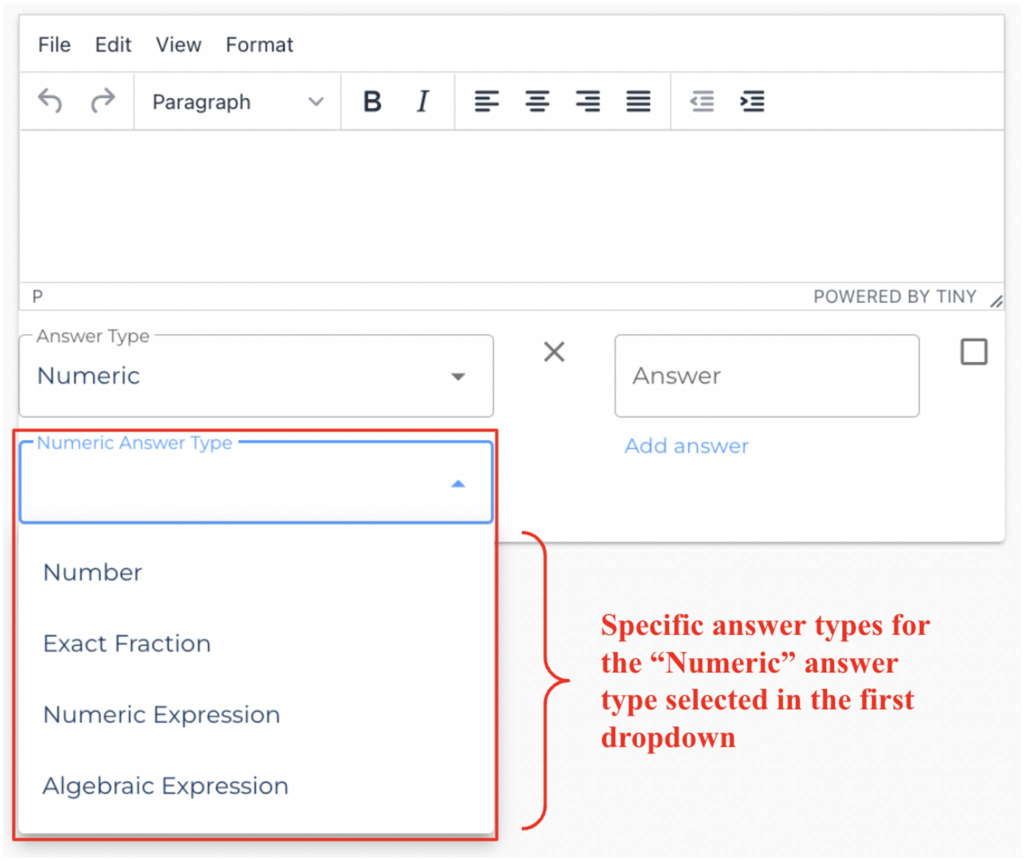


Figure 13: After selecting one of the overarching answer types, another dropdown appears allowing the user to select a more specific answer type.

Another aspect of the problem editor is selecting the answer type for the problem through a drop-down menu. Figure 12 displays the three overarching answer types within ASSISTments: Numeric, in which the student must input a number; Text, in which the student must input a string; and Multiple, in which the student must choose the correct answer(s) among a selection of options. Each overarching answer type also has a more specific answer type to select. For example, selecting Numeric from the dropdown will make another dropdown appear from which you can choose whether or not the answer should be a Number, Exact Fraction, Numeric Expression, or Algebraic Expression. This is shown in Figure 13.

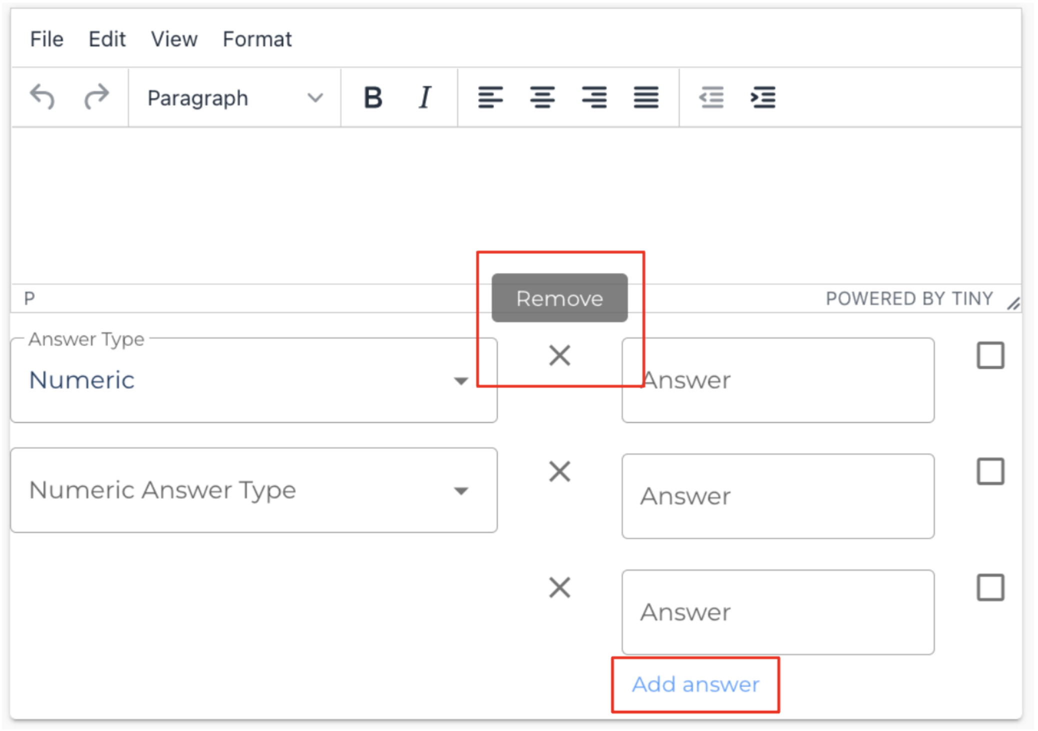


Figure 14: Users can add more answers by selecting the “Add answer” text. Hovering over the “X” mark will show a “Remove” text.

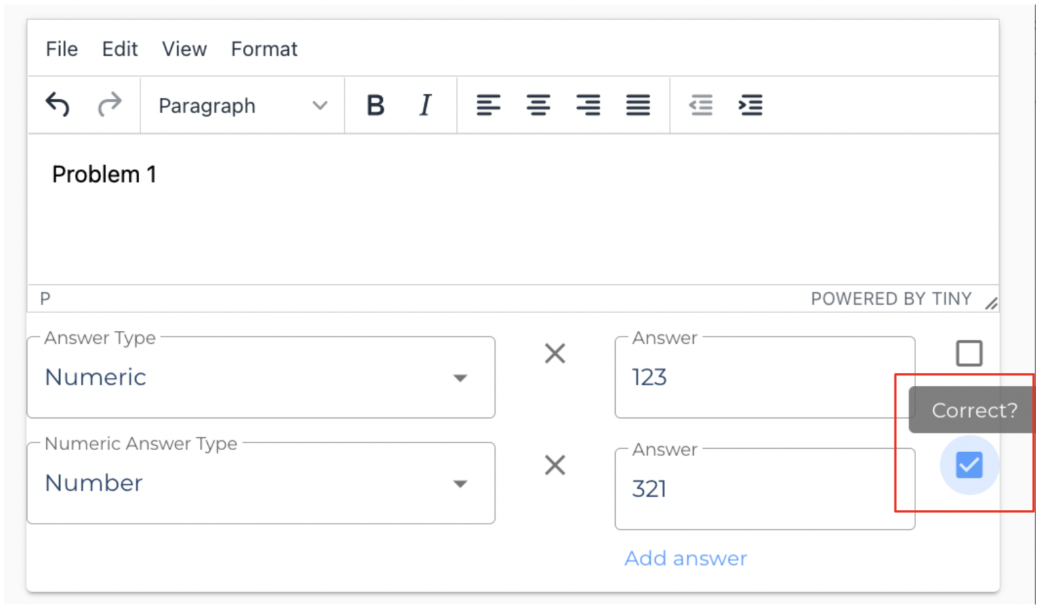


Figure 15: Users can input the value for answers and select whether or not they are correct. Hovering over the checkbox will show a "Correct?" text.

After specifying the answer type, users are also able to write out answers for the problem. We provide a text field to allow users to enter in a value for an answer. Users are able to select, using a checkbox, whether or not the value is correct. They can add more answers by selecting the highlighted text which says "Add answer", or remove an answer by selecting the "X" mark as shown in Figure 14. We also made it so that text would show when users hover over certain parts of the editor. Figure 15 shows that hovering over the checkbox will display a "Correct?" text while hovering over the "X" will display a "Remove" text as shown in Figure 14.

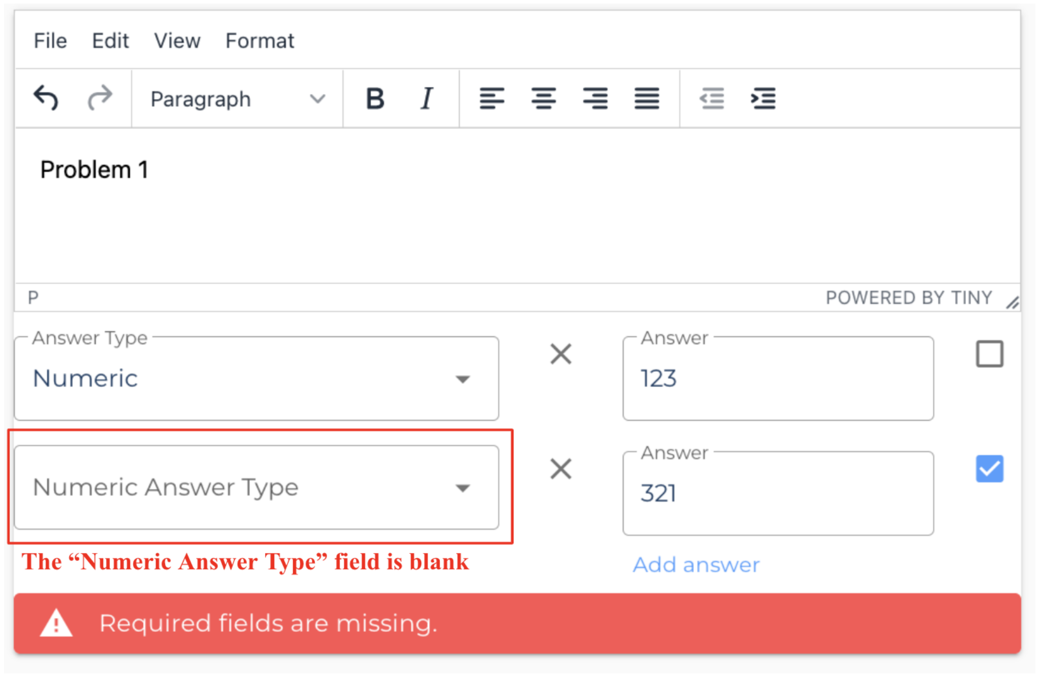


Figure 16: When a user does not provide input for one of the fields in the problem editor, an alert appears stating that required fields are missing.

When creating a problem, users are required to input problem text, an answer type, and an answer. If any of these fields are missing and the user attempts to save the problem, it will not be saved and users will be provided with an alert that one or more of the required fields for creating a problem are currently missing. This can be seen in Figure 16. The alert will disappear once all of the required fields have been filled in and the problem is successfully saved.

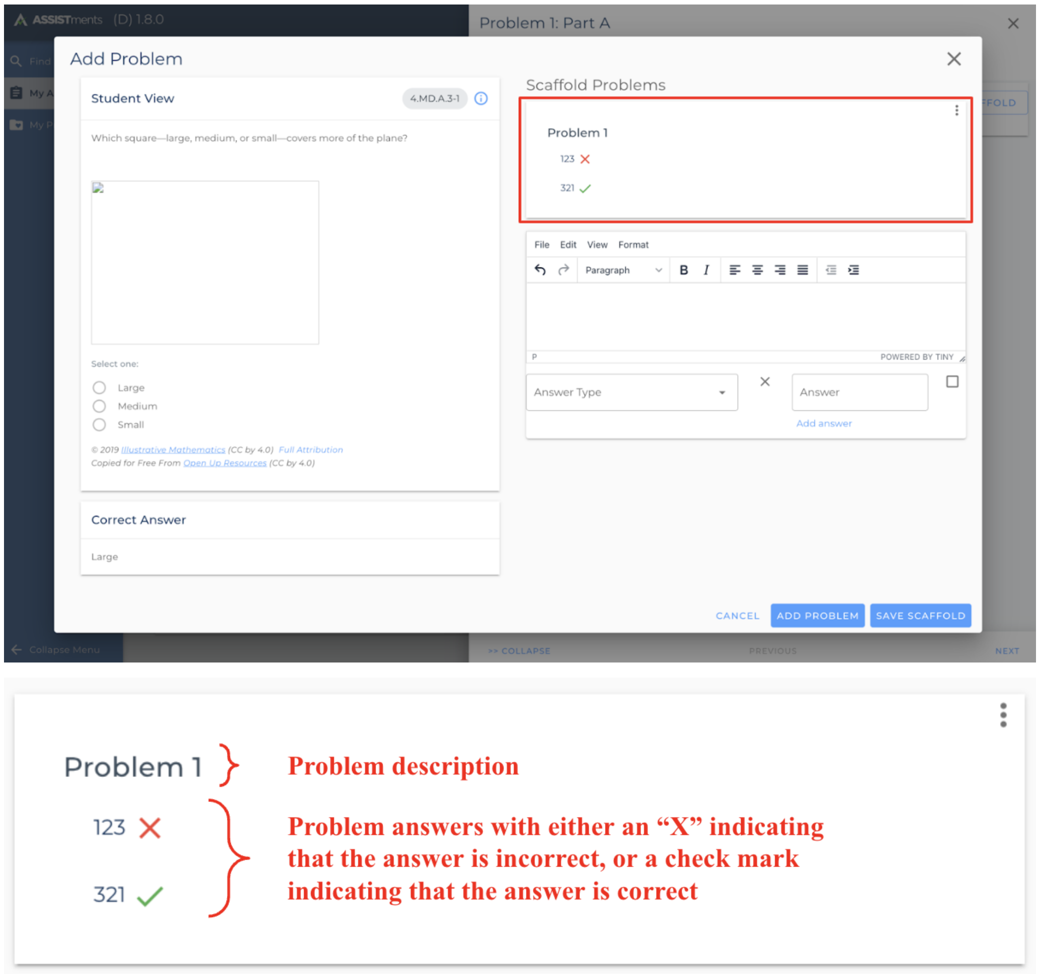

ASSISTments (D) 1.8.0
Problem 1: Part A
Add Problem
Student View
4.MD.A.3-1
Which square—large, medium, or small—covers more of the plane?
Select one:
Large
Medium
Small
© 2019 Illustrative Mathematics (CC by 4.0) Full Attribution
Copied for Free From Open Up Resources (CC by 4.0)
Correct Answer
Large
Scaffold Problems
Problem 1
123
321
File Edit View Format
Paragraph
POWERED BY TINY
Answer Type
Answer
Add answer
CANCEL
ADD PROBLEM
SAVE SCAFFOLD
Problem 1
123
321
Problem description
Problem answers with either an "X" indicating that the answer is incorrect, or a check mark indicating that the answer is correct

Figure 17: After adding the problem, the problem is displayed as a card containing the problem description, answers, and which answers are correct. Users have the ability to edit and delete problems through the button consisting of three vertical dots in the upper right corner of the card.

Clicking on the "Add Problem" button at the bottom right of the dialog box will add the problem to the Scaffold. This will create a card which will show the problem you just created as shown in Figure 17. This will display the problem description, the answers, and whether or not each answer is correct. In order to actually save the Scaffold however, the user must click on the "Save Scaffold" button at the bottom right next to the "Add Problem" button. This will retrieve all the values from the problem editor and store all the values in a JavaScript object, which can be easily converted to JSON format and sent to the backend through a request. This will add it to the database such that when the page is refreshed, the page can retrieve the Scaffold problems from the database and display it to the user.

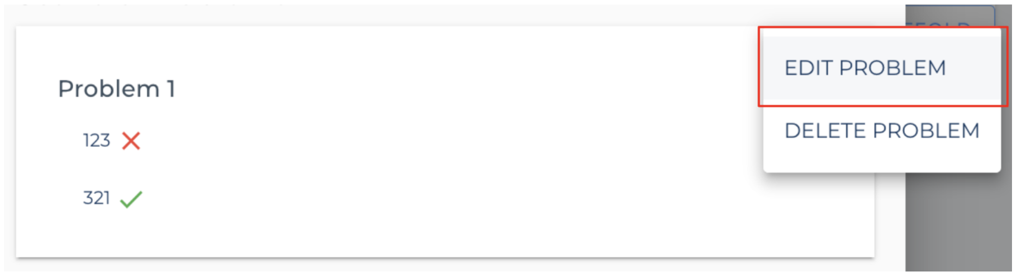
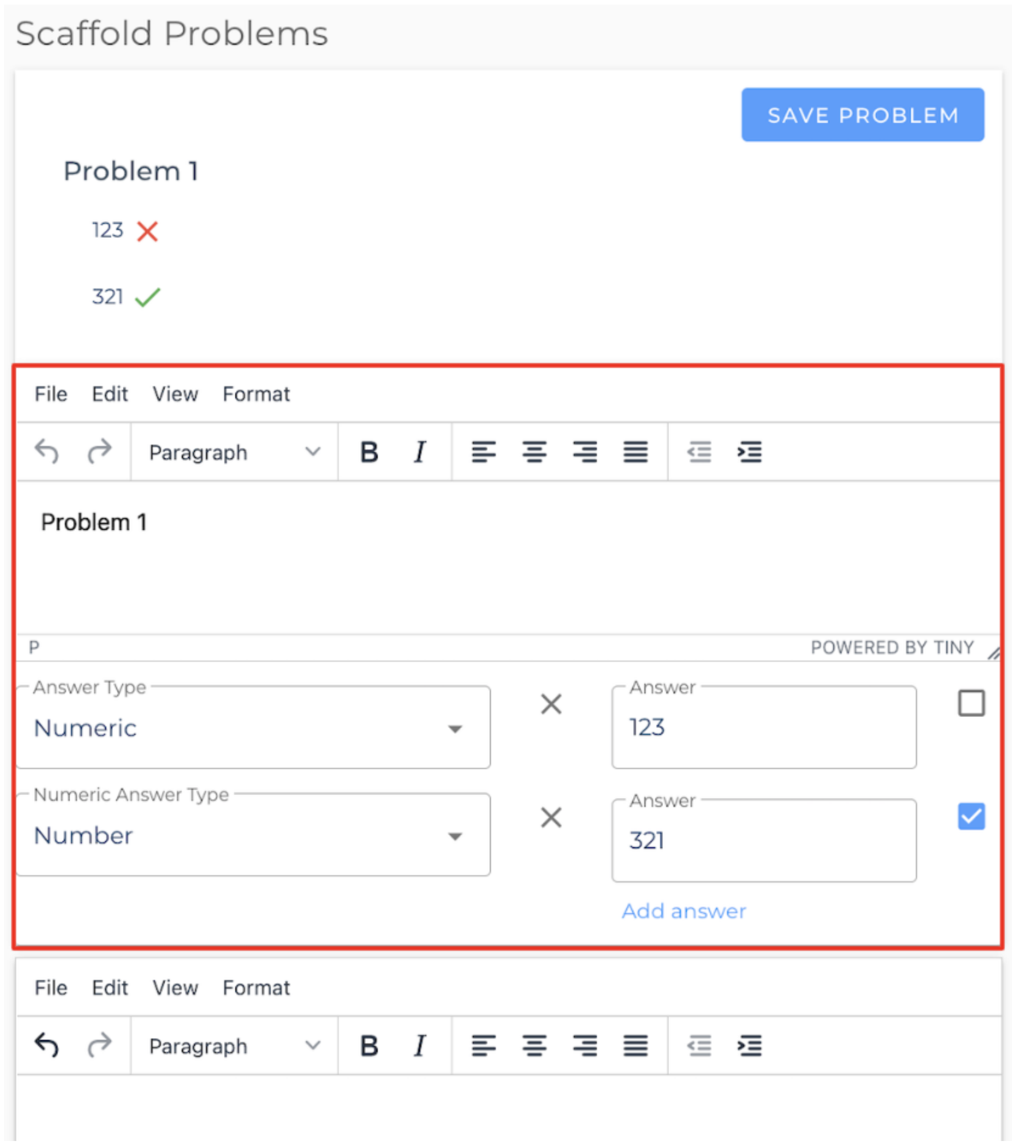


Figure 18: The “Edit Problem” option.



Problem 1

Answer Type: Numeric

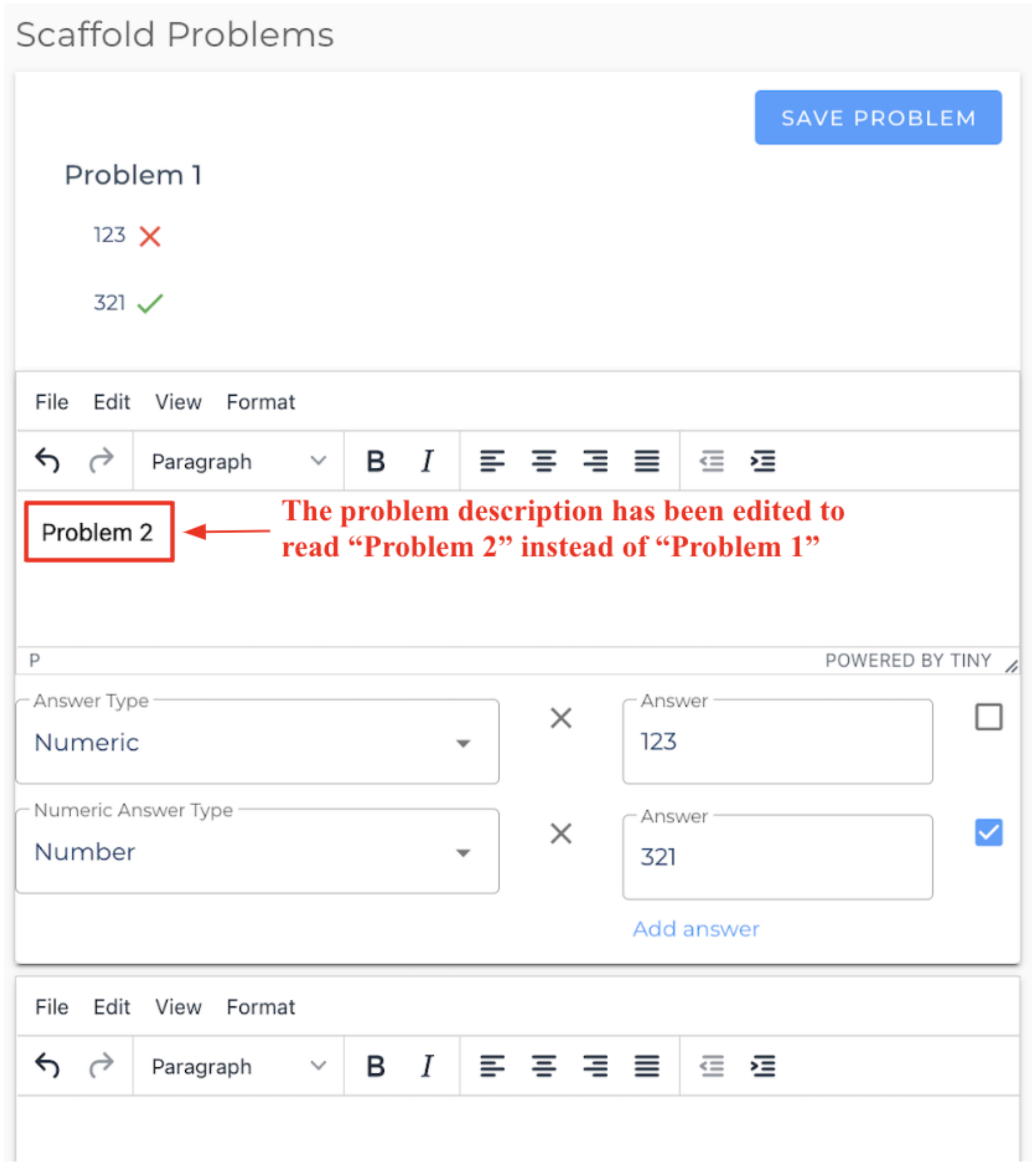
Numeric Answer Type: Number

Answer: 123 ☐

Answer: 321 ☑

Figure 19: The problem editor with the populated fields.

To edit the problem and enter the problem editor again, the user can click on the “Edit Problem” button, shown in Figure 18, that appears after clicking on the three vertical dots in the upper right corner of the problem card (see Figure 17). This opens up the editor with the text boxes and dropdowns filled in with the current values for the problem below the card displaying the problem. This can be seen in Figure 19. There is also a “Save Problem” button that appears in the upper right corner of the problem card. To modify this problem, the user can make the necessary changes in the editor and save the problem by clicking the “Save Problem” button. Doing so closes the editor and shows the changes in the problem card. Figure 20 shows the problem being modified and Figure 21 shows the saved problem with the modified content. Editing an already existing Scaffold will set a flag that lets the program know that the Scaffold needs to be updated in the backend. Upon clicking the same “Save Scaffold” button in the bottom right corner, it will send a request to the backend to update the Scaffold instead of to create a new Scaffold.



Problem 1

123 ✗

321 ✓

Problem 2

The problem description has been edited to read "Problem 2" instead of "Problem 1"

Answer Type: Numeric

Numeric Answer Type: Number

Answer: 123 ☐

Answer: 321 ☑

Figure 20: The problem editor with the populated fields and the problem description edited.

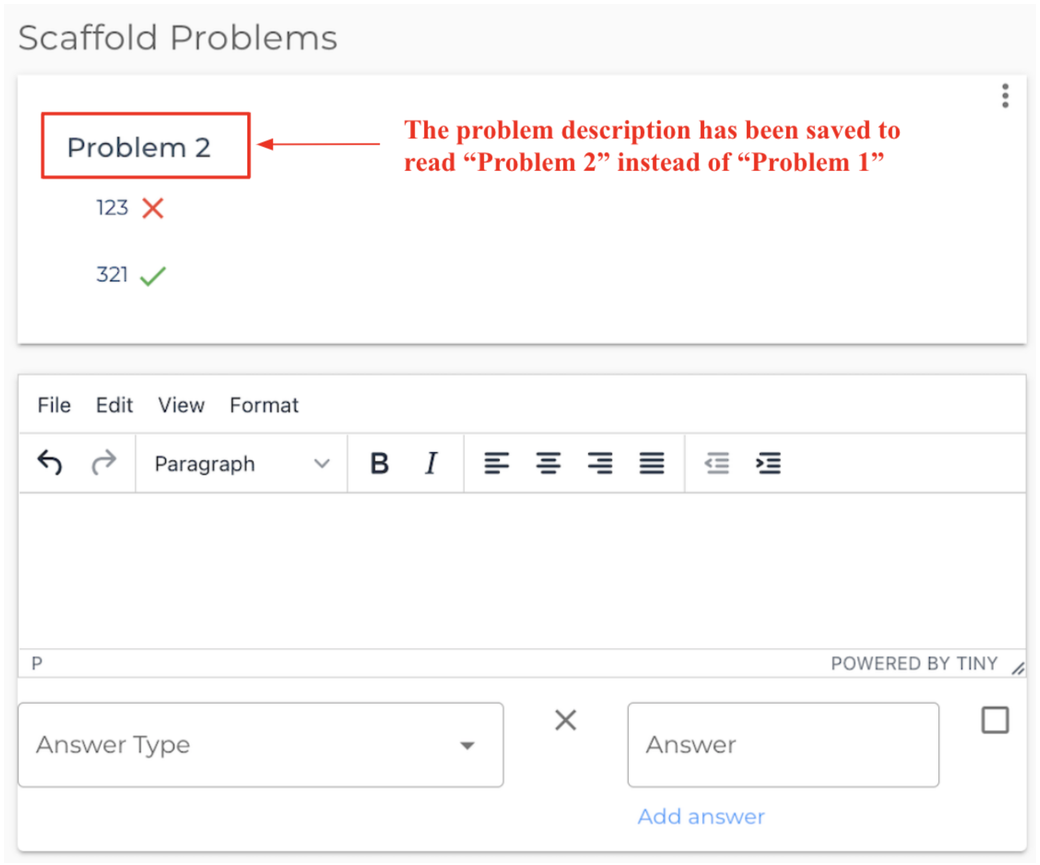


Figure 21: The problem card with the new problem description saved as “Problem 2” instead of “Problem 1.” The problem editor has been cleared.

Users can delete the problem by again locating the button consisting of three vertical dots in the upper right corner of the problem card to view the “Delete Problem” option as shown in Figure 22. Clicking this button causes the problem card to disappear. Clicking the “Save Scaffold” button after deleting a problem will cause the program to send a request to the backend to update the Scaffold. If deleting a problem makes it so that there are no more problems in the Scaffold, clicking the “Save Scaffold” button will send a delete request to the backend to remove the Scaffold, and the user will need to create a new Scaffold the next time they try to add problems to the Scaffold.

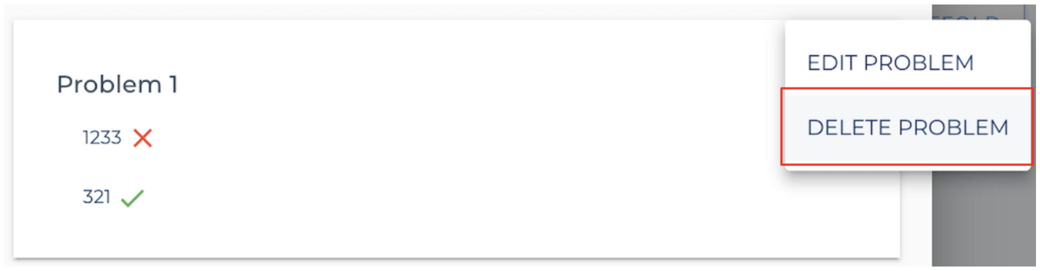
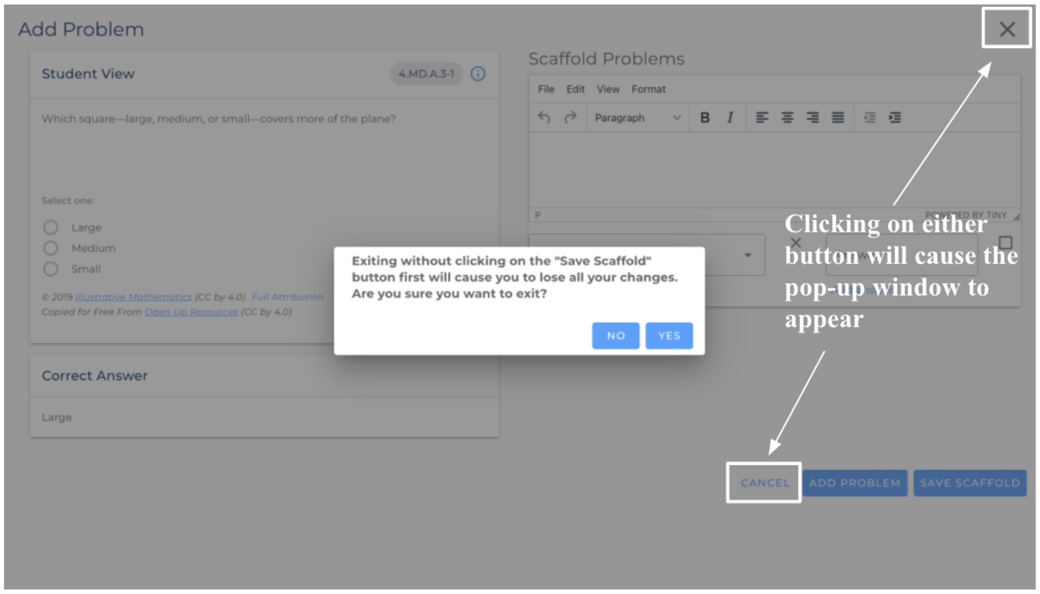


Figure 22: The "Delete Problem" option.



Exiting without clicking on the "Save Scaffold" button first will cause you to lose all your changes. Are you sure you want to exit?

Figure 23: The pop-up window that appears when the user attempts to exit without saving the Scaffold first.

If the user clicks the "X" in the upper right corner or the "Cancel" button, a pop-up window will appear warning the user that any changes they made to the Scaffold will not be saved. This is shown in Figure 23. If the user wishes to discard their changes, they can confirm they want to exit by clicking "Yes." If the user intended to save their changes to the Scaffold, they should click "No" to continue editing the page and click "Save Scaffold" at the bottom right to save the Scaffold.

Next Steps

Our project can be expanded on by implementing additional features on Scaffolds. For example, users could be given the option to add hint messages and explanations on Scaffold problems. If a student is struggling to answer a problem, they could click a button to ask for a hint. This would display a hint message for the current problem to lead the student in the correct direction to solve the problem. In addition, if a student is unable to answer the problem correctly after a number of tries, they could view an explanation for how to solve the problem.

Another feature that can be implemented within our project is mistake messages. Common Wrong Answer Feedback is currently implemented within ASSISTments but not for Scaffolding. When students incorrectly answer a problem, a mistake message can appear to both inform them of their mistake and how to move forward in solving the problem. Mistake messages can be quite helpful if there is a common mistake students may make in solving a specific problem, as teachers will have the ability to offer feedback to students in a more generalized manner.

Additionally, an option for nested Scaffolds could be added. For example, if a student gets a Scaffold question wrong, there could be a Scaffold on that question that will help the student reach the right answer. This may be useful if a Scaffold problem is still too complicated for a student to get through and would benefit from breaking down the problem further.

For future projects, we would also like to suggest some changes to the user interface. Currently, the TinyMCE text editor that holds the problem description cannot handle images or other media. Being able to add pictures would be useful as pictures are often important to visualize math problems. Videos could also provide support by explaining concepts to students. Secondly, our UI allows teachers to make Scaffolds without any correct answers. However, it would be necessary to have a correct answer for each problem for Scaffolding problems. For this reason, we suggest implementing code to make sure users put in at least one correct answer for the Scaffolding problem.

We also have a few suggestions in terms of the appearance of our components. Firstly, in the Scaffold card located under student supports, we suggest adding some margins around the "Edit Scaffold" button in Figure 24 and having it in line with the Scaffold text so that it looks more professional.

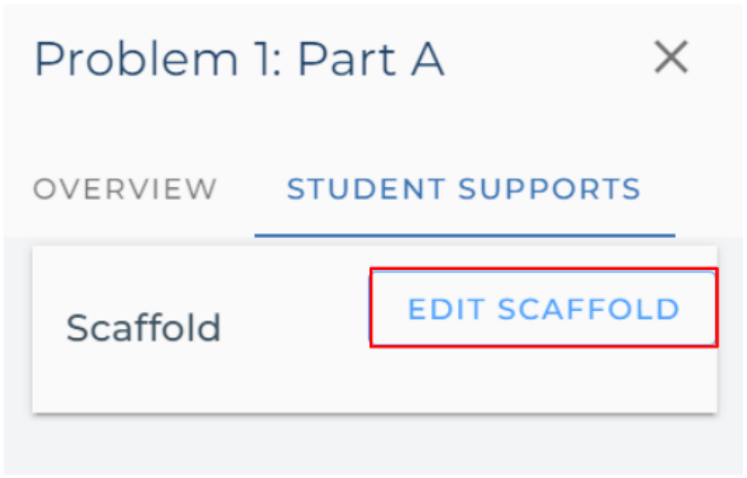


Figure 24: The "Edit Scaffold" button.

Another suggestion we have is to add margins around the "Answer Type" dropdowns shown in Figure 25, just to reduce the cluttered look. Secondly, the "X" button in the same figure corresponds to the "Answer" text box, but it is a little hard to tell even though we have text that shows when the user hovers over the "X" button. For that reason, we suggest bringing the "X" closer and on the same horizontal line to the center of the "Answer" text box. Another suggestion we have is to remove the "X" from the first "Answer" text box, and only have it for any other answers that are added, since a Scaffold should have at least one answer on it.

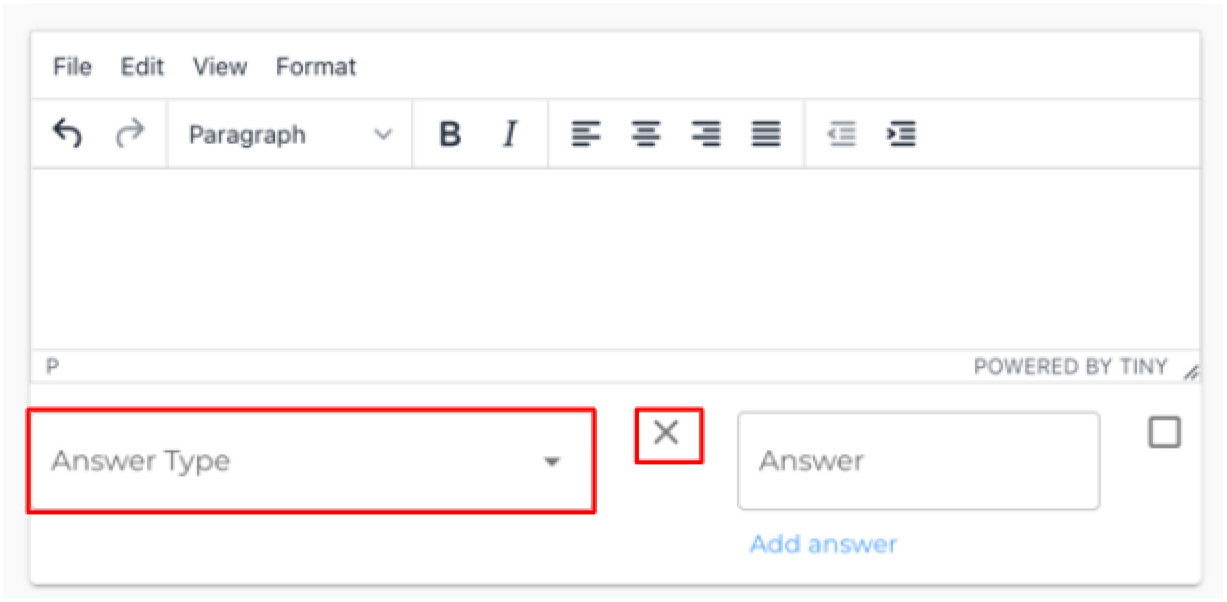


Figure 25: The problem dialog.

Next, we would suggest moving the Problem preview to start in line with the center of the “Save Problem” button since these components are slightly out of line with each other creating an unbalanced look as seen in Figure 26. Referring to the same figure, it would also look better if the TinyMCE font was consistent with the font of the “Answer” text.

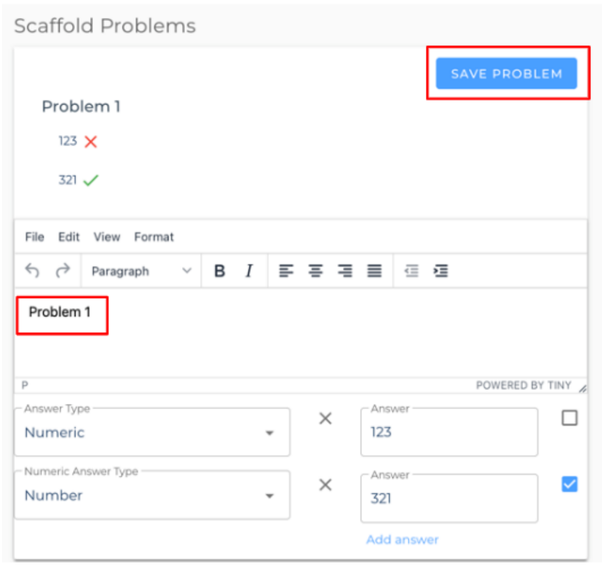


Figure 26: Problem dialog once a problem has been saved.

We also realized that the placements of our buttons at the bottom right of the Scaffold dialog box may be a bit misleading. Having the buttons to cancel out of the Scaffold dialog, add a problem to a Scaffold, and save a Scaffold so close to each other might confuse the user into accidentally pressing the wrong button. It is also unintuitive for the user to have to click the "Add Problem" button first to add the problem to the list before clicking "Save Scaffold" as many users may mistakenly press the "Save Scaffold" button without pressing the "Add Problem" button, which would lose all their changes. Making changes to the interface to address this problem is essential.

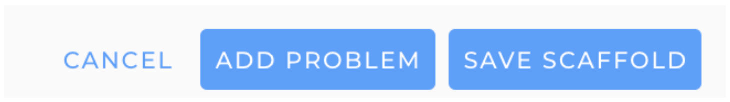


Figure 27: The buttons to cancel out of the Scaffold dialog, add a problem to a Scaffold, and save a Scaffold.

Our team has implemented Scaffolding on the published database which is currently being used on ASSISTments, but ASSISTments will be moving over to the draft database. The draft database is much less complicated and more efficient for ASSISTments to use, and therefore implementing Scaffolding functionality in the draft database would be the next step in this project.

Recommendations

For students joining related projects in the future, we recommend that they be proactive about onboarding fully before digging into the technicalities of the project. Secondly, we would encourage them to look around the ASSISTments platform before beginning the project, not only the current ASSISTments but also ASSISTments 1.0. Looking at ASSISTments 1.0 helped our team understand the data needed to create a problem as well as what was necessary for the frontend design. Next, since our team faced many technical difficulties with setting up the project, we would recommend getting through the setup as quickly as possible. ASSISTments is a fast moving environment, as the code and design is constantly changing, so we would encourage the team to check in with their advisor once a week to keep their code up to date with the most recent changes in ASSISTments. Lastly, some tools and frameworks that students should read up on are Maven, the Spring Framework, Jackson Databind, and Vue, and a background in Java and TypeScript would be very useful.

References

ASSISTments. (2022). *Free Education Tool for Teachers.* ASSISTments. https://new.assistments.org

Johnson, R., Hoeller, J., Donald, K., Sampaleanu, C., Harrop, R., Risberg, T., Arendsen, A., Davison, D., Kopylenko, D., Pollack, M., Templier, T., Vervaet, E., Tung, P., Hale, B., Colyer, A., Lewis, J., Leau, C., Fisher, M., Brannen, S., … Webb, P. *Spring Framework Reference Documentation*. Spring. https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/beans.html

Oracle. (2022). *Lesson: Classes and Objects*. Java Documentation. https://docs.oracle.com/javase/tutorial/java/javaOO/

TypeScript. (2022). *JavaScript with Syntax for Types*. TypeScript. https://www.typescriptlang.org/

Vue.js. (2022). *Introduction*. Vue.js. https://vuejs.org/guide/introduction.html