ASSISTments Interactive Qualifying Project: EIR CWA

An Interactive Qualifying Project Report

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

Patrick Lee

Benjamin Babalola

Christopher Lobo

Date: December 11th, 2020

Professor Neil Heffernan

Abstract:

This project was intended to help streamline the content generation process for teachers writing common wrong answer feedbacks and instructional recommendations as part of the EIR grant awarded to ASSISTments. This was achieved by gaining an understanding of the former process of generating content through spreadsheets and then utilizing engineering techniques to build a web-based tool specifically designed to meet the needs of these teachers.

Acknowledgements

We would like to thank our project advisor, Neil Heffernan and our supervisor Ashish Gurung for providing us with this opportunity to work on this project. We would also like to thank Hilary Kreisberg and Tina Cardone, our professional teaching partners, for providing valuable feedback on how to improve the look and feel of the tool as it was in development. Thanks also go to Britt Neuhaus, Tom Bolton, Cindy Starks, Cristina Heffernan, Brittany Gonio, and Sidharath Chhatani who are all members of the EIR team at ASSISTments, and all teachers who have participated in the first and second cohort trials for the EIRCWA project. Without teachers to write content for these problems and the EIR team to handle the details of running these trials, the work this team conducted wouldn't be helping anyone.

Table of Contents

Abstract: **2**

Acknowledgements **3**

Table of Contents **4**

1 Introduction **5**
- 1.1 Definitions 5
- 1.2 Contextual Inquiries 6
- 1.3 User Stories 7

2 Project Design & Initial Implementation **7**
- 2.1 Initial Design Steps 7
- 2.2 Building the Tool 8

3 Libraries & Languages **9**
- 3.1 TinyMCE 9
- 3.2 WIRIS 9
- 3.3 MathJax 10
- 3.4 JavaScript CSS HTML 11
- 3.5 Bootstrap 12
- 3.6 ASSISTments Database 12
- 3.7 Handlebars 13
- 3.8 JSP Pages 13
- 3.9 REST API 14
- 3.10 Tomcat Server 14
- 3.11 JQuery 15

4 Improvements After Initial Design **15**

5 How the tool was received **16**

6 Recommendations for Future Changes or Studies **18**
- 6.1 Stack Page 18
- 6.2 Crowdsourcing Content 19

Appendices **20**
- Appendix A: Use Case Diagram 20
- Appendix B: Class Diagram 21
- Appendix C: Page Hierarchy 22
- Appendix D: Technology Stack 23

1 Introduction

1.1 Definitions

In order to discuss the nature of the design problem this project group aimed to solve, a brief discussion of some definitions is necessary to understand the scope of the work that was done. The EIR-CWA project, or electronic instructional recommendation/common wrong answer project, is intended to assist in the creation of common wrong answer feedbacks and instructional recommendations for commonly used problem sets in the ASSISTments system. A *common wrong answer* is an incorrect answer to an ASSISTments problem which a significant number of students gave. A single problem can have multiple common wrong answers, typically ordered by the prevalence of each common wrong answer (also referred to as a CWA). CWA feedbacks are student-facing messages intended to be shown when a student inputs a common wrong answer for a given problem, and are intended to address common misconceptions associated with the question which could lead to the student getting that particular CWA. Instructional recommendations (or IRs) are instead teacher-facing suggestions designed to aid teachers whose students are inputting common wrong answers. There is also a distinction between *problem level* IRs and *answer level* IRs. Problem IRs are recommendations that apply to all CWAs listed for a given problem,

while answer IRs are specifically tailored to address a specific CWA from an educator's perspective.

1.2 Contextual Inquiries

Before beginning the design phase of this project, two interviews with our professional partners from Lesley University were conducted with two goals in mind. First, to understand their original process for writing CWA feedback and IRs; and second to identify the intended goals our product must achieve for a minimum viable product.

The original process for writing CWA feedback and IRs was primarily done using Google Sheets. ASSISTments provided the teachers writing content with a spreadsheet containing all common wrong answer information for a given curriculum. The teacher then needed to find all CWA data from the lesson they wished to write content for and organize it. This was primarily done by cross referencing the data from the master spreadsheet with that problem set in the general ASSISTments platform. This organized data was placed in a separate table, where a sub-table was created to house the CWA feedbacks and IRs. After all content for that lesson was written, the original intention for moving that content into the ASSISTments platform was to manually convert the spreadsheets written by the teachers into a database. In summary, the manual process was incredibly involved, required manual sorting of data which could more easily be done programmatically, and the eventual conversion into a form usable within the ASSISTments platform would have been even more time consuming for developers.

1.3 User Stories

After conducting the contextual inquiries, we began to think through different possible users of the ASSISTments platform who might benefit from interfacing with the tool we were setting out to build. Consider a teacher with a number of students who are all getting a common wrong answer to one of the questions. The tool we wanted to build should allow them to send out a hint to their students based on the answer they got, and/or allow them to view an instructional recommendation for a remedial activity to help them clear up their students' misconceptions about the topic at hand.

Another likely user story that could arise during the content generation process is a teacher who writes a feedback to a common wrong answer, but who missed one possible way for a student to arrive at that answer. Other teachers familiar with that unit should be able to call that oversight to the attention of the original writer, who would then be able to go back and add more content to their original feedback.

Finally, consider an administrator using the review functions of this tool, and they come across a teacher with a particularly deep understanding of the curriculum they're writing about. That administrator should be able to reach out to that teacher user and give them review permissions, allowing more qualified people to be able to review content written by teachers before it's available to be seen by students.

2 Project Design & Initial Implementation

2.1 Initial Design Steps

After considering possible user stories, a use-case diagram (shown in Appendix A) was created to map out how the team wanted users to be able to interact with the EIR-CWA creation tool. We defined three different classes of users who would be interacting directly with the tool: teacher users, superusers, and developers. Teacher users are teachers trained by our Lesley partners to be able to write CWA feedbacks and IRs, superusers are users who are able to review and approve the writing of teacher users, and developers are the members of the IQP team, and other people involved with the development of the tool.

Based on the use case diagram and contextual inquiries, a class diagram and page hierarchy of the tool were then created (Appendices B and C, respectively). The tool would originally be divided into three web pages accessible to different users depending on their responsibilities: an index page, a write page, and a review page. The index page is accessible to all users of the tool, and would list all ASSISTments problem sets which are eligible for CWA feedback/IR creation. The create page, also accessible by all users, would allow users to create CWA feedbacks and IRs for a particular problem set. The review page would only be accessible by superusers and developers, and would allow superusers to review the work done by teacher users, flag CWA feedbacks and IRs for revision by the users that wrote them, and approve CWA feedbacks/IRs as ready to be deployed.

The server-side implementation for the tool would be controlled by two Java classes: Main and DataUtility. Main would handle navigation between the different web pages of the project, and be the primary interceptor of RestfulAPI requests for data from the client. DataUtility would be called by Main in order to retrieve data from the EIRCWA database as needed by the client.

2.2 Building the Tool

After the design decisions described in the previous section were approved, the work of building the tool began in earnest. Initially, the IQP team was tasked with designing the client-side part of the project while our ASSISTments partners provided the basic implementation of the project's server-side component. The first testable version of the tool contained an index page with five assistments problem sets and a create page allowing for the creation of user-generated content. In the following weeks, the review page was added to allow the existing superusers to inspect the content the first cohort of users had generated. The ability for the superusers to flag content for review was modified into a comments feature, allowing superusers to leave comments on user-generated content. During development of the tool, a number of languages and software libraries were used. These are explained in the following sections.

3 Libraries & Languages

3.1 TinyMCE

TinyMCE is a text editor technology used to create text boxes in the ASSISTments pages. TinyMCE was used in the project to allow users to write content and that could be saved to a database to be seen by other users or to be edited later.

In addition to a simple text box, TinyMCE has several features that allow users to customize their content within a webpage. Users have access to several fonts and can choose their font sizes and colors. The text editors have several paragraph formatting and aligning options and adjustable line heights. This accessibility makes TinyMCE a good choice for the ASSISTments webpages.

Outside of the basics that TinyMCE includes in its default editor, the team also included some plugins that could be useful to ASSISTments teachers. The ‘image media’ plugin is offered by TinyMCE themselves. Incorporating this into the toolbar of the editor allows teachers to place pictures and videos into the text box.

3.2 WIRIS

In addition to image/media plugin, ASSISTments webpages also includes an external plugin called WIRIS. WIRIS displays an interface to write math symbols within the TinyMCE text boxes. Many math problems at the 7^{th} grade level use symbols and constants that are not included in the standard character set i.e. square roots, absolute value, plus-or-minus, and more (shown below).

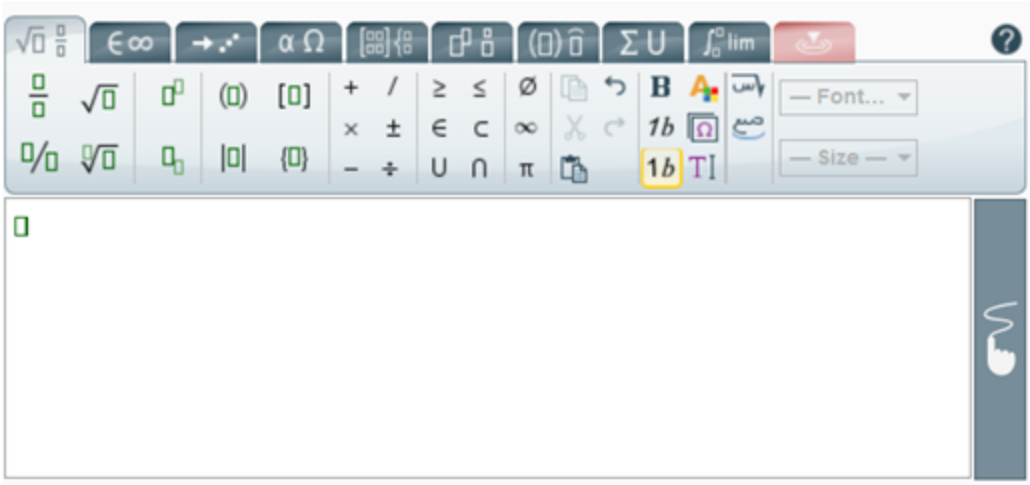


Fig. 1 The WIRIS math text editor that allows users to include math symbols in their content.

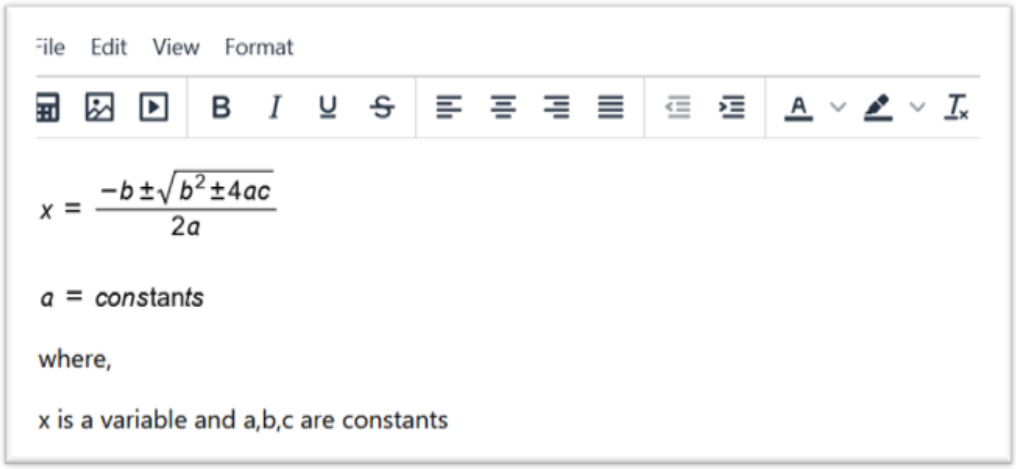


Fig. 2 Math symbols successfully added into the TinyMCE editor using WIRIS

As seen above, WIRIS allows teachers to incorporate these symbols into the editor alongside normal text. Being able to include math symbols into the editor spares teachers from having to upload images of hand-written work.

3.3 MathJax

After adding WIRIS to the TinyMCE, the team started testing and quickly noticed an issue. Although the math text was being saved successfully to the database,

displaying the text with math symbols outside of the TinyMCE editor did not work in Chrome. After further research, it was discovered that this error was due to Chrome removing MathML support from their browser. MathML is a markup language for math and science content that WIRIS uses to display non-standard symbols in HTML web pages. Since MathML is not supported by Chrome, the team decided to use MathJax to solve the problem.

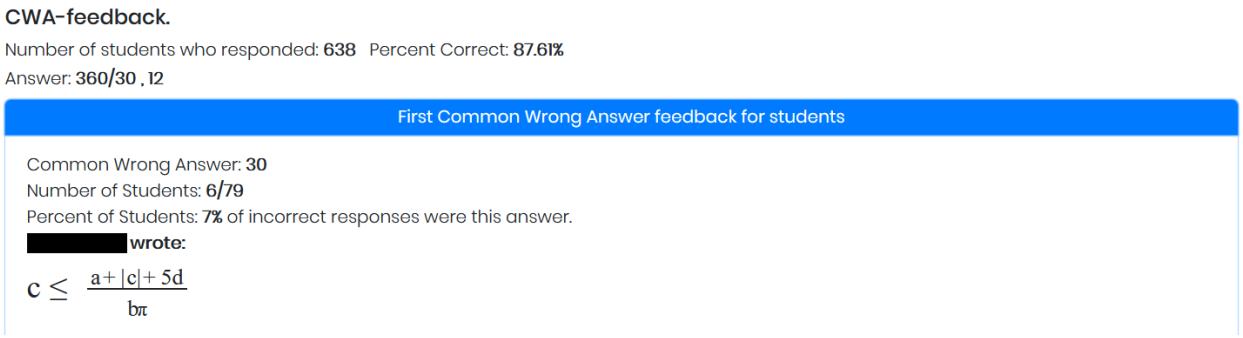


Fig 3 Math symbols are displayed properly using MathJax

MathJax is a JavaScript display engine that is compatible with all browsers. It takes in MathML content as input and displays it in HTML+CSS format. It also has compatibility with Microsoft Office and LaTeX which provides accessibility for users who may use other software for math editing. Using MathJax not only ensures users across browsers see content the same way, but also ensures that any MathML not properly formatted by untested browsers will be handled and displayed correctly.

3.4 JavaScript CSS HTML

The primary languages used to create the web pages were Javascript, CSS, and HTML. HTML is used to create elements and basic structure of pages. This includes headers, text boxes, buttons, and URLs. CSS is used to apply style rules to the HTML

elements to make them visually appealing. CSS is written separately from the HTML so that a set of style rules can be made and applied to all sites to ensure uniformity. CSS affects spacing, text and background colors, and the size of text and elements on the screen. The functionality of elements on the webpages is written in JavaScript. JavaScript is important for element functions locally in the user browser, and for relaying information to and from the database using jQuery.

3.5 Bootstrap

Bootstrap is an open-source CSS framework used to create dropdown menus in the page. Dropdowns are lists of links that are hidden until the user opens them by clicking on the dropdown button. This feature prevents clutter on the webpage by hiding the links until they are needed and does not cause inconvenience to the user.

Fig 4 A dropdown menu made using bootstrap

3.6 ASSISTments Database

The database is an integral part of the ASSISTments design. It stores all the data that is created by users. This includes content written by teachers, statistics on how students performed, problem-set ids, and much more. The database consists of several

tables for specific information, each consisting of columns and rows. Each row represents an object, and each column represents an attribute that each object has.

3.7 Handlebars

```
<p class = 'ps-pr-id-header'>Problem List:</p>
<div class='list-group problem-select-container'>
    {{#each data}}
        <a href='#dropdown-list{{@index}}' id='problem-select-button{{@index}}' cl
        <div id='dropdown-list{{@index}}' class='list-group problem-parts-containe
        {{#each parts}}
            <!--index: [{{@../index}}, {{@index}}]-->
            <a href='#' id='problem-select-dropdown{{@../index}}-{{@index}}' class
        {{/each}}
        </div>
    {{/each}}
```

Fig. 5 an example template in Handlebars

Javascript Handlebars is a templating system. What that means is if you were to call it during your jsp page, jsp being essentially a version of html that works with handlebars, it would replace your handlebars call with the template that you set in your handlebars file of that name. We used this for things that involved lists, and it proved to be very useful specifically when making lists of all the names of problem sets in our create, review, and stack pages, and it was also especially useful when going over the details of problem sets, where it had all the information like comments, and problem information, as there were many different problem parts that were being listed that reused portions of code, which is perfect for handlebars.

3.8 JSP Pages

JSP, standing for Java Server Pages, is a web page building language. In the aspects of code, it looks and feels very similar to HTML, which is another very popular

web page building language. The main difference between JSP and HTML, however, is that JSP runs on a particular server, while HTML runs on the client. Another difference between JSP and HTML is that we are able to run our Handlebars code calls in our JSP pages. What this means is that we are able to have the templating system working if we were to use JSP rather than if we were to use HTML, and this allows us to get lists working where we can reuse a section of code multiple times without rewriting the same thing over and over again.

3.9 REST API

REST API are specific kinds of APIs, and to define them, it is a good idea to define APIs as well. API stands for application programming interface. APIs are what enable different programs of different types to communicate with each other. Through an API, your HTML and JavaScript can communicate to the Java in the back end and have them all work in harmony. The term REST is a format of an API. REST stands for representational state transfer. REST APIs are just a specific format of APIs, generally linking pieces of data (responses) to certain URLs (requests).

3.10 Tomcat Server

Tomcat servers are specific application servers from the Apache Software Foundation. They execute Java servlets, which are small programs that run on servers, and they can also render JSP pages. Our project runs on an Apache Tomcat server, and through that we can load our JSP pages with our Handlebars templates.

3.11 JQuery

JQuery is a JavaScript library that simplifies a lot of the bulk that comes with JavaScript. One of the main things we used from JQuery was AJAX. AJAX is a function from JQuery that simplifies many lines of JavaScript code into one function that we can use. Simplified, AJAX connects our front end to our back end, allowing our web page to talk to the Java code we have running in the background. Normally in JavaScript, this would take many lines of code, but with the help of AJAX through JQuery, this is a lot shorter than it otherwise would be. We used AJAX specifically in a lot of places. For example, in every page, when it gets loaded, we have to retrieve things like the comments and tags that were already attributed to that problem set

4 Improvements After Initial Design

After releasing the initial page, the team received feedback from the team and the users and decided to implement some new features.

When teachers were reviewing other teachers' CWA's on the Review page, they wanted to be able to leave suggestions on changes they would make or concepts they felt were missing. In order to do this, the team created a comments section under each CWA on the review page. This comment section consists of a list of comments left by teachers and a TinyMCE textbox in which teachers could leave a comment or the writer of the CWA.

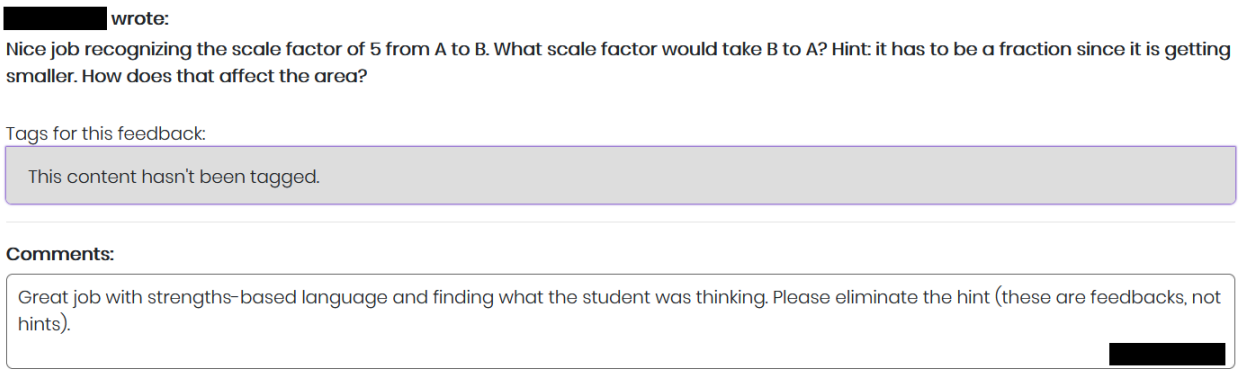


Fig 6 The comment section where a user has left a comment on another user's work.

The new comments section was well received and as the process of teachers reviewing teachers was important, the partners requested a new feature. Originally, the product was released with a Create page where teachers could create CWA feedbacks, and a Review page where teachers could review individual teachers' CWA feedbacks. The Stack page was created so that teachers who underwent the same training, could review each other's work in a more efficient manner. When looking at the CWA feedbacks on the Review page, users can only see the feedback written by one teacher at a time. Users had to navigate back one page, select a new teacher, and then find and reselect the same problem-set in order to review another teacher's feedback. The Stack page eliminates these extra steps by displaying the feedback written by everyone in the user's cohort consisting of the teachers with whom they did initial ASSISTments training.

5 How the tool was received

Initially, we tested our work on the first IM cohort with 5 problem sets. This was our first group, and there were some general issues that we had to fix. Small bugs and

things were definitely apparent, but through receiving feedback from the users we were able to squash some of the quickly noticeable ones. Things that made it hard for the users to actually run and operate the tool were the first to get corrected, and then we added things that were specifically attuned into things that the first cohort requested.

Later, we tested our work with a second IM cohort, and eventually with the ENY group. In general, they had positive things to say. They were finding it interesting how the system worked, and they enjoyed the puzzle of figuring out how students came to the answers that they did to help them when commenting on their work, and enjoyed how they were able to review other teachers' work to see how other people responded to students who answered the same thing the same way.

In general, test users of our tool enjoyed the use of it, and when they found things like bugs, we went through the process of fixing the bugs. Users also found getting used to what the program does to be interesting; for example, things like going through the math to find where the students came with the answer they did posed an interesting challenge to the different cohorts. This became especially apparent when getting into the less common wrong answers. However, seeing how other teachers responded to the students who got those answers greatly helped out the other teachers who were not sure how to respond.

6 Recommendations for Future Changes or Studies

6.1 Stack Page

The way the user trials are structured changed between the first and second cohorts. Because of this change, the stack page implemented as part of the response to our users' feedback on the tool will no longer be able to serve its intended purpose. The stack page was useful to superusers in the first trial because all teacher users were creating content for the same five problem sets provided to them. However, the second cohort has been structured differently: all users in the second cohort have received different lessons in order to generate content for a wider variety of problem sets across multiple curricula. Thus, the stack page's original purpose, displaying all user-generated content for a given problem set, has become obsolete.

The stack page still houses some functions which are vital to the tool. Currently, the stack page is the only part of the tool which allows for superusers to provide tags for user-generated content. The stack page was also the original intended home for the content approval function, which would eventually mark user-generated content as ready to be moved into the main ASSISTments platform. For these reasons, the stack page has remained largely intact, in spite of its obsolescence. There are two possible paths forward in the future of the stack page which other projects will have to address: either the critical functions of the stack page must be moved elsewhere (likely to the review page), or the original functionality of the stack page (to display all users' content for a problem set) must be removed in favor of the current functionality of the stack page.

6.2 Crowdsourcing Content

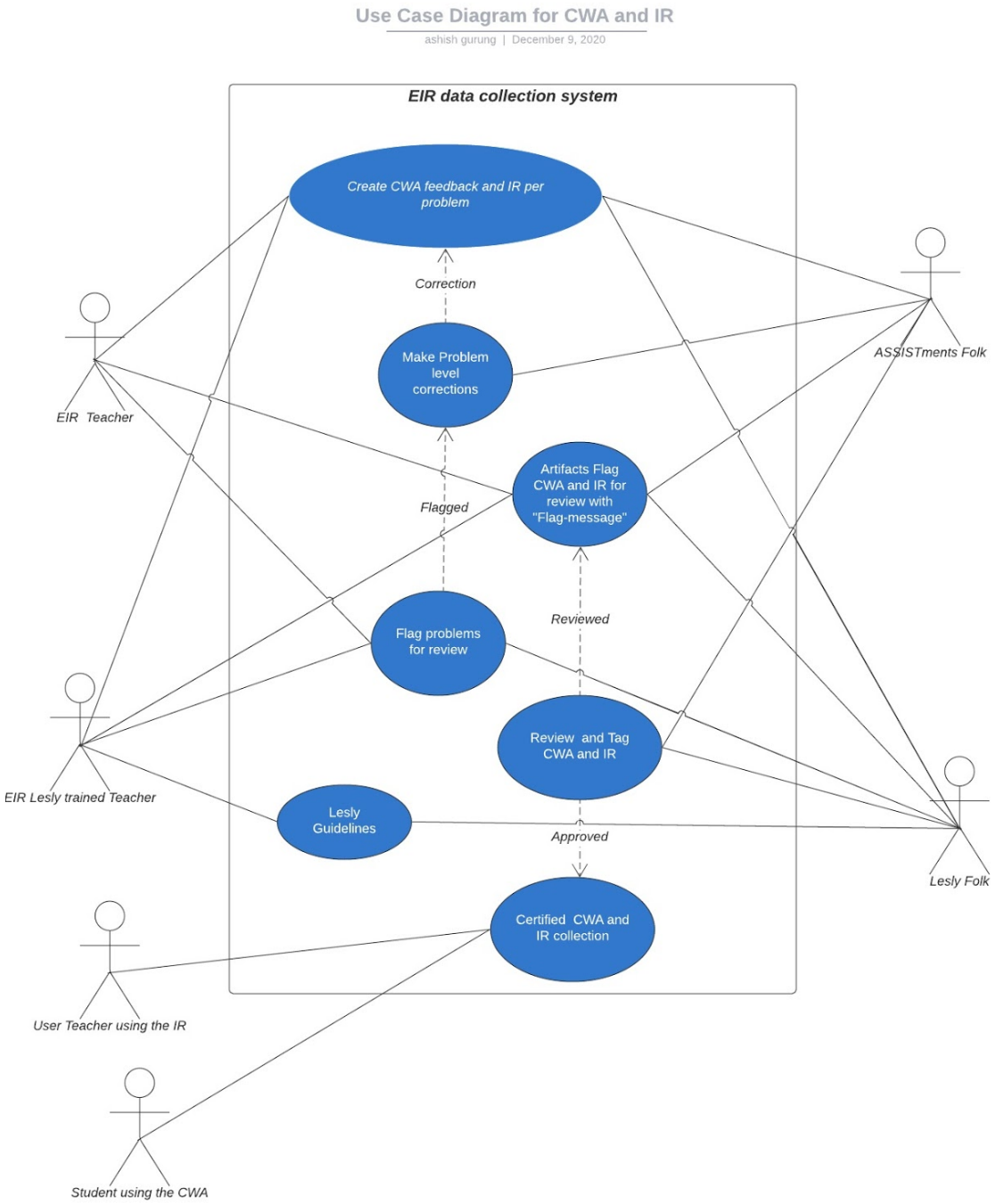
The other primary change that would likely increase the utility of the tool would be the introduction of crowdsourcing user-generated content. While the second trial of the tool will generate much more content than the first and the ability to curate the content our users generate is extremely valuable, expanding the pool of users to allow any ASSISTments user to create CWA feedbacks and IRs would greatly increase the volume of available content for the general ASSISTments platform.

In implementing crowdsourcing, many changes to the original structure of the tool will need to be made in order to accommodate the new context of the tool's usage. These changes must be accompanied by a way to sort and rank all available comments on a user's content, as well as a similar measure for the content's implementation on the ASSISTments platform. Research into popular ranking algorithms on forum sites such as Stack Overflow and Reddit would provide future teams with a baseline in designing their own ranking algorithms.

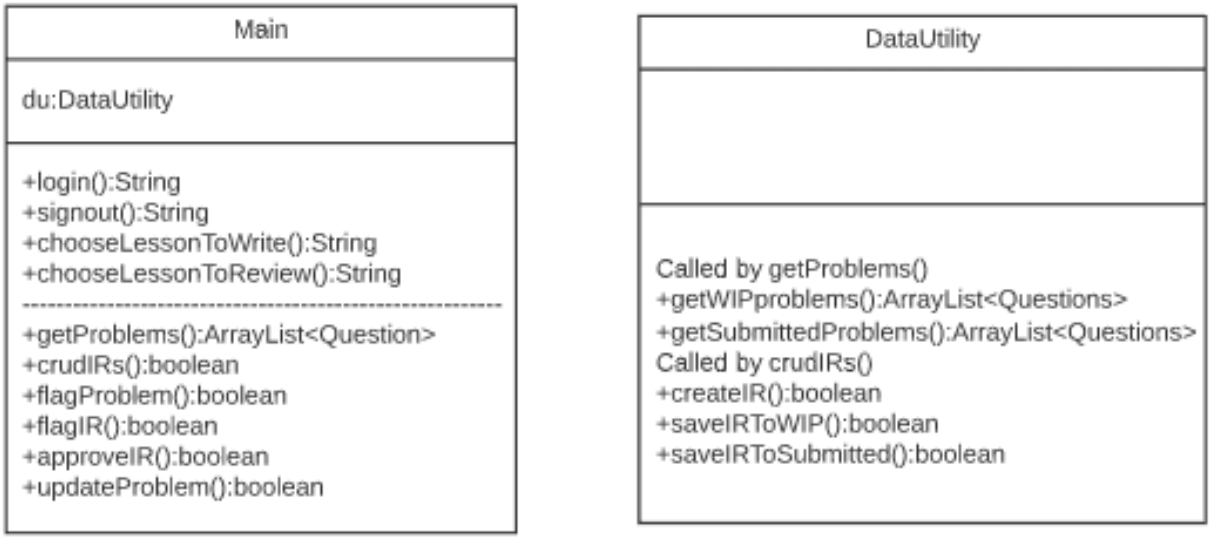
Additionally, there are significant trade-offs between crowdsourced content and content curated by our superusers. Crowdsourcing would allow for a fast proliferation of user-generated CWA feedback and/or IR content, but there is a potential loss of quality which must be investigated. A study on the effectiveness of curated content versus crowdsourced content and a control group which received no additional hints from the EIRCWA pool of feedbacks would provide valuable insight on the effectiveness of the project, and would be crucial in deciding which content generation method will best serve the students who are using the ASSISTments platform.

Appendices

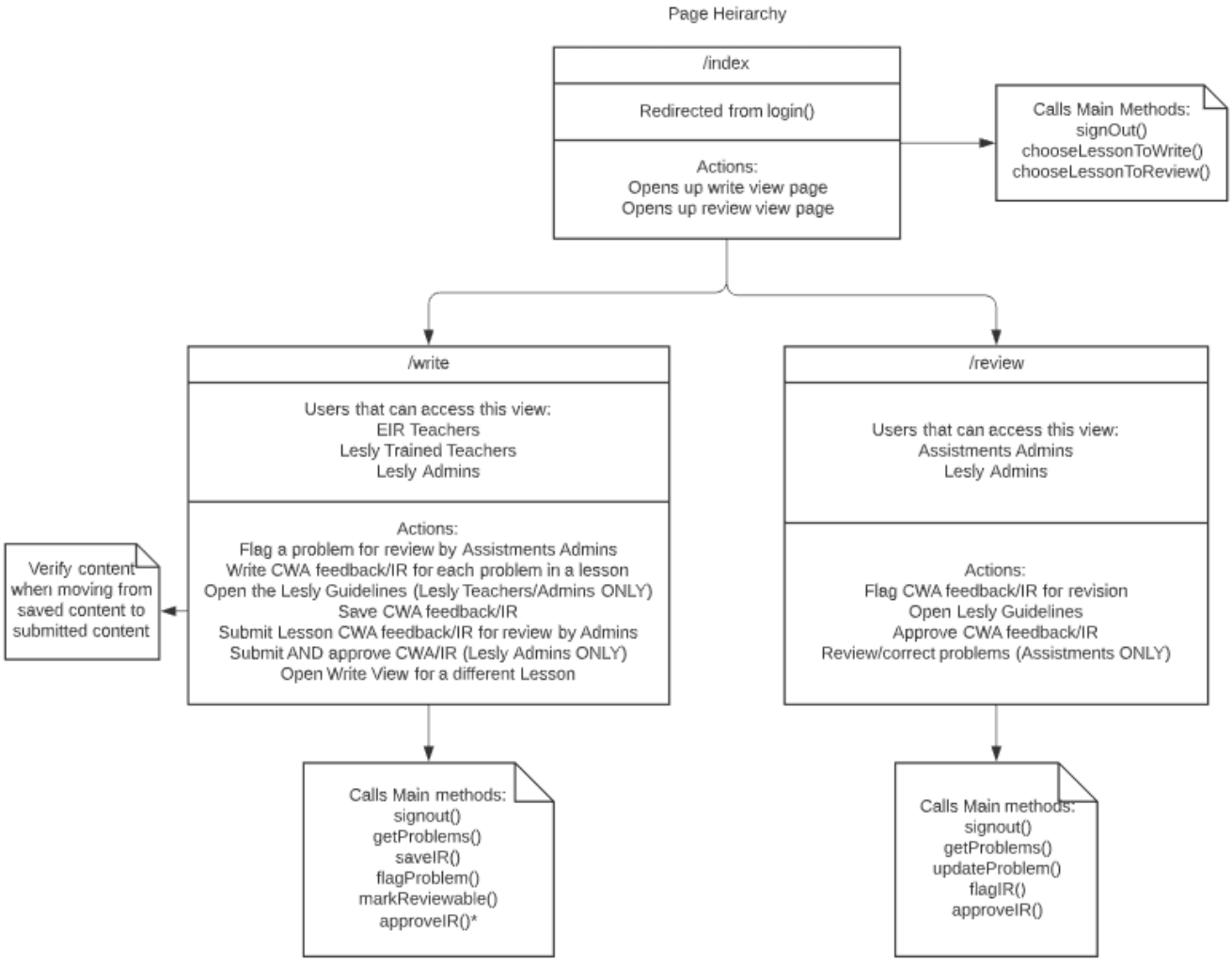
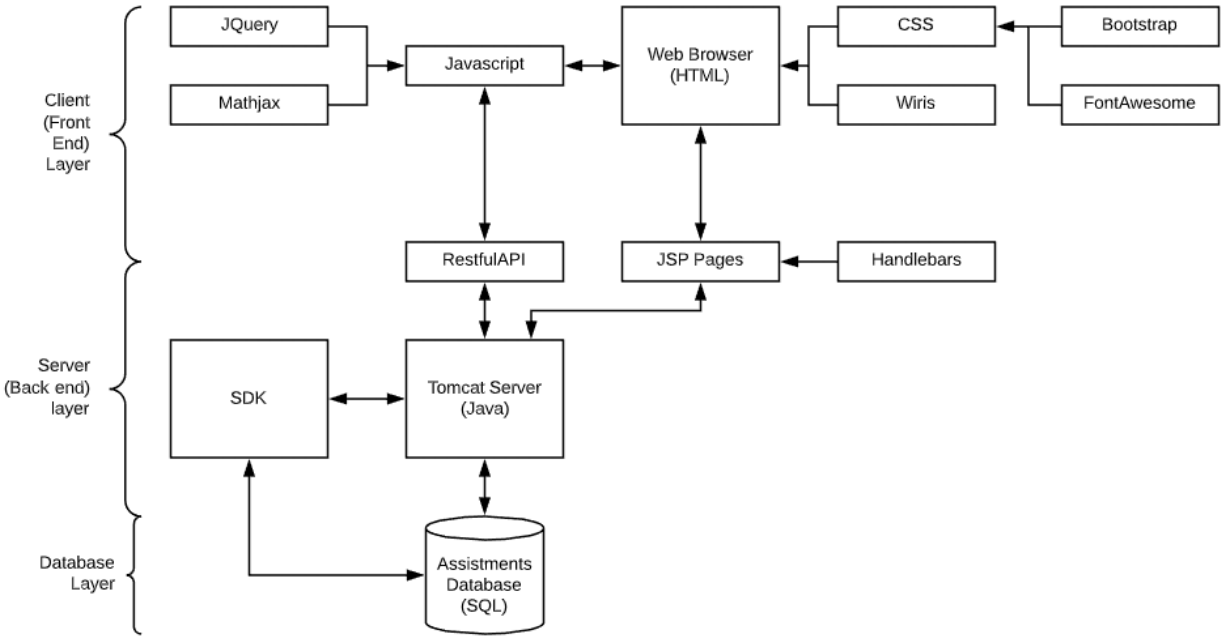
Appendix A: Use Case Diagram



Appendix B: Class Diagram



Appendix C: Page Hierarchy



Appendix D: Technology Stack