Remind Math Project

Design Document

Last Updated 12/9/2016

Eric Stanton

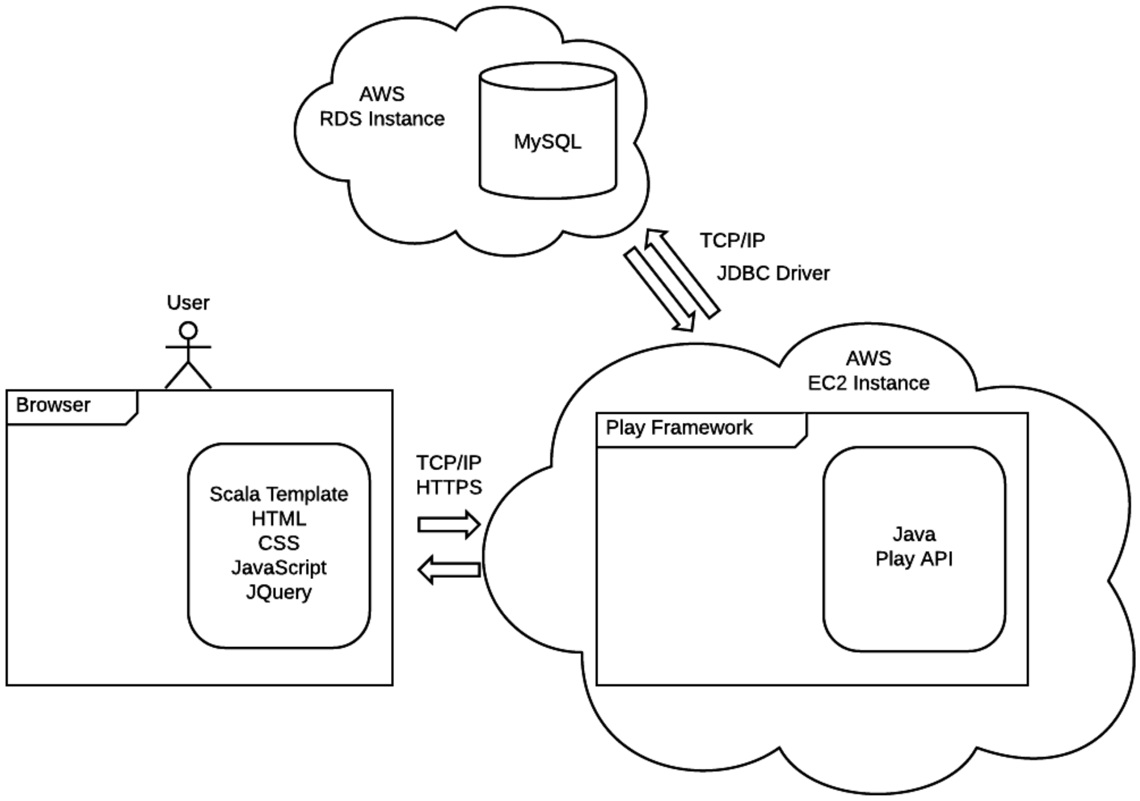
Summary

The Remind Math Project is a test generation tool to help user's study for the math portion of the GED test. The tool can also be used by teachers and tutors to generate tests for their students.

From the user's standpoint, the remind math project can be broken down into 2 main objectives: Test Generation and Test Grading.

Web Architecture

In simple terms, the Remind Math Project is a web application. There is a front end user interface accessed by the user in a web browser like Internet Explorer, Firefox, or Chrome and a back-end Server accessed through the browser using the front-end application. The front end is built with Scala Templates, jQuery, JavaScript, HTML, and CSS and the backend is built with Java and the Play Framework. For persistence, there is also a MySQL database.

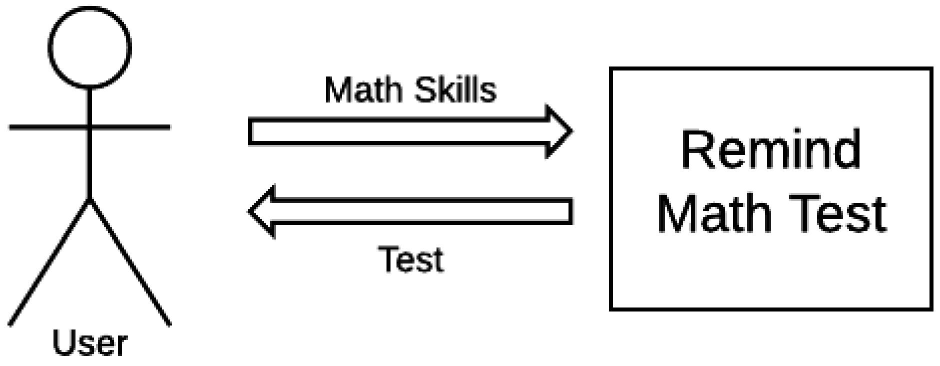


The browser exists on the user's local computer, tablet, or phone and communicates with the backend application (via the front-end application) using HTTPS. Although there is no truly sensitive data in the Remind Math Project, HTTPS along with an SSL certificate is required to protect the user's login credentials and create a secure connection between the browser and application.

The back-end server is an AWS EC2 instance and the application is run in a web container that is launched with the application by running the distributable executable (created by the dist task in the play framework). The backend application communicates with the MySQL database by way of TCP/IP using a JDBC driver. There is no direct interaction between the user or browser with the database. All interactions must go through the back-end application.

Test Generation

Test generation receives math skills based on user selections. These skills can be entered manually or automatically assessed based on previous test experiences. The returned test will include all test questions including any diagrams and labels, text problem to solve, and possible answers. Only the math skills requested by the user will be included in the math test.



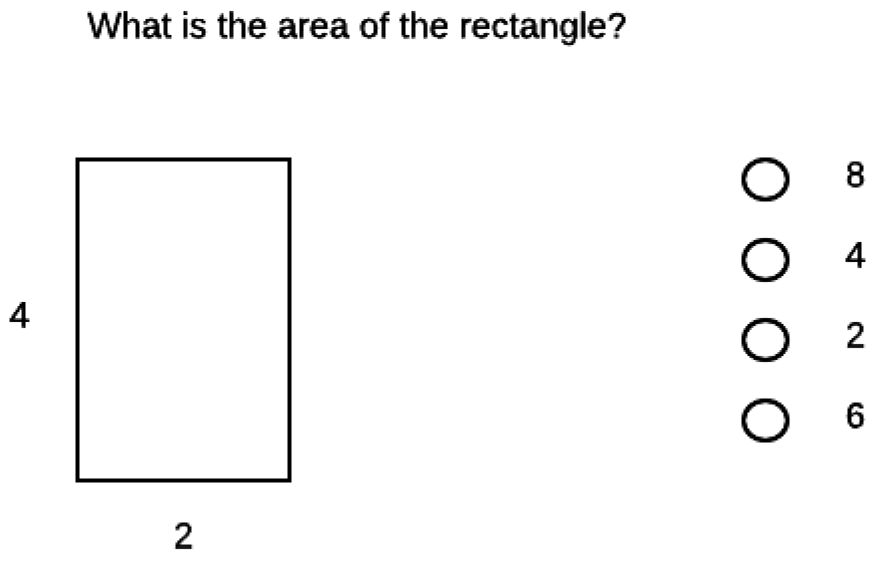
Tests must be generated on the back-end server and passed to the front-end. Tests generated on the front-end are subject to user inspection and tampering such that a user could give themselves easier questions or discover the answers during or after generation.

Tests must be persisted in the database before returning to the user so the submitted questions can be validated against the generated questions.

Test Taking

After a test is generated, the user will take the test in the browser. Each answer will be selected by the user in the browser in the form of an html radio button. A radio button is ideal because of its single selection nature (compared to a checkbox with multi-selection nature) and full selection visibility (compared to a select/drop box with single selection visibility). This has the added benefit of looking familiar to paper test takers as it appears like a bubble selection as seen on old-fashioned Scantron tests. Remember, not all users are young and tech-savvy, so it is important to use an obvious, intuitive, and easy to use interface.

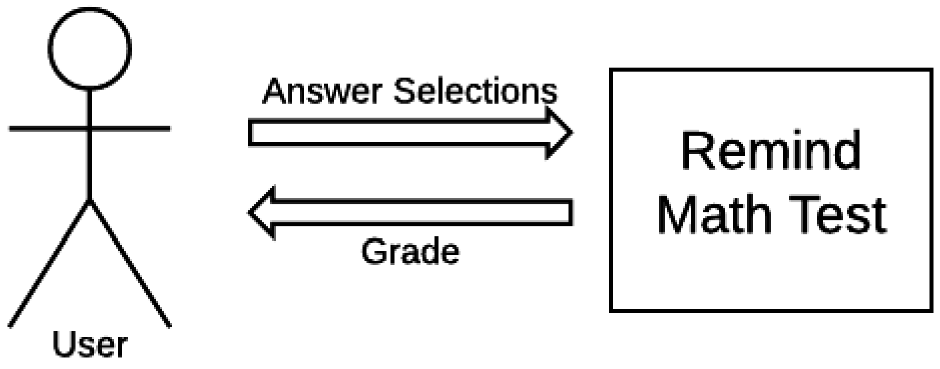
Example of a question:



Tests will be submitted over HTTPS using an html form submission.

Test Grading

After the user has answered all of the test questions, they will be provided a grade. In addition, the software will save and track the user's progress overall as well as their progress for each specific math skill. In addition to a grade, the user will get their original test back with explanations of the correct answer and how to solve the problem.

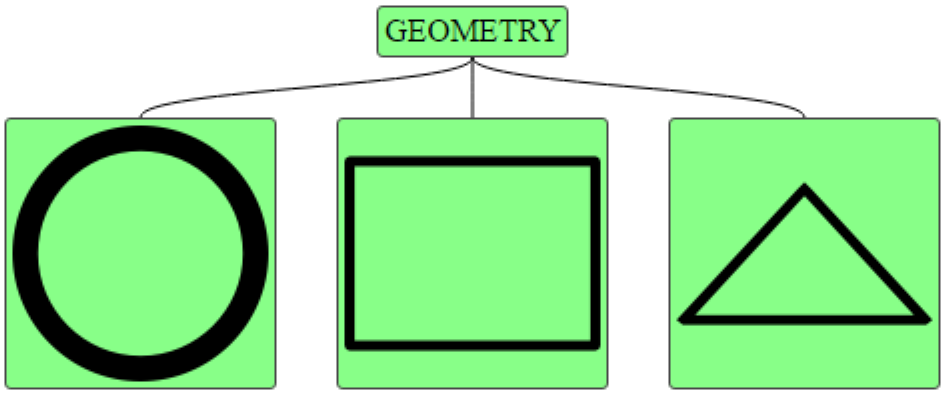


Tests must be graded on the back-end server. Grading a test on the front-end would require passing the answers to the user's browser. A savvy user could find the answers by inspecting the html or javascript. Remember, this is a math test, not an html inspection test.

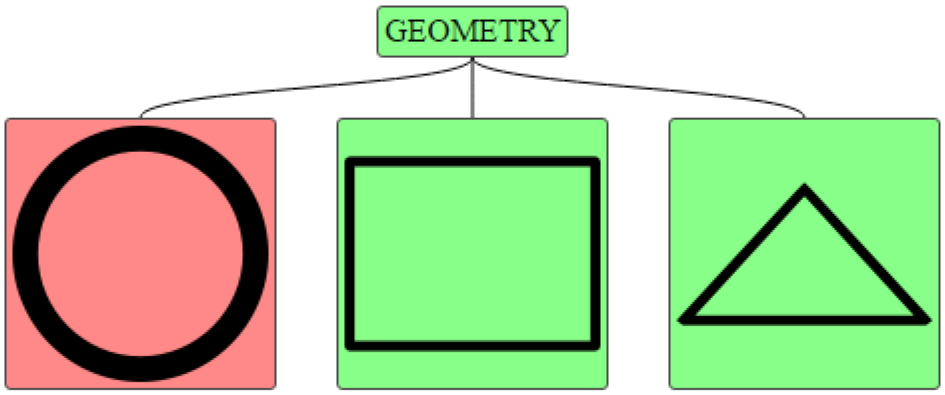
Math Skills

Math skills are grouped and can be selected by the user either as a full group or in part. Skill groups are comprised of related skills, usually as a lesson, series of lessons, topic, or category you would expect to learn in formal education. Each individual skill represents a user's ability to handle a certain type of problem.

Here is an example of a skill grouped entirely selected



Here is an example of a partial selection



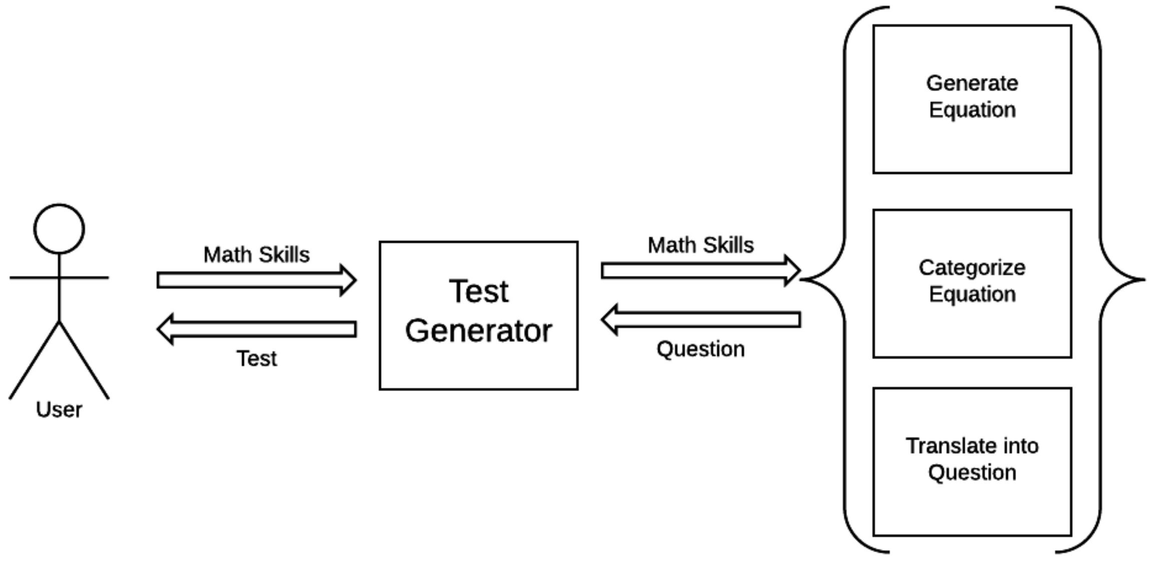
Skills groups must be learned in order, but skills do not. For example, you must master arithmetic before moving on to geometry. However, once you are advancing into geometry, you may choose to be tested about triangles before rectangles, even though it is recommended to learn about rectangles first.

Skills groups and items are generated using the treant-js library. Skill selections capabilities were developed using jQuery and integrated with the treant-js objects manually.

Skills selections will be passed from the UI to the back-end by way of an HTML form submission.

Math Test Modules

Based on the skills selected by the user, a math test will be created by the system. Question generation can be broken into 4 main components and visualized below.



Test Generator

The first component is the test generator and is the first functional module in the service layer of the back-end web application. The test generator is responsible for using the other 3 modules to create a test with reasonable skill diversity based on the user's request. The test generator is responsible for throwing away questions that hinder the creation of an appropriate test. A test is considered appropriate when it contains at least 1 question from each skill requested by the user and in the case of few skill selections, a reasonable proportion of each skill.

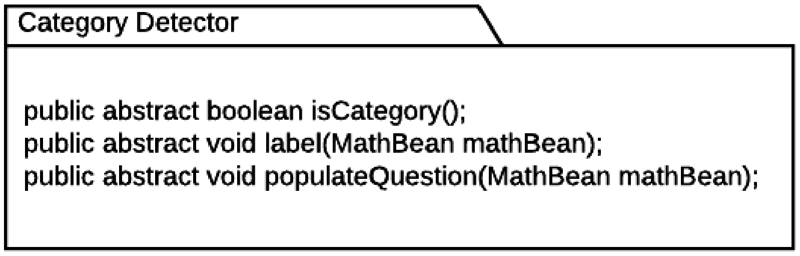
Equation Generator

The equation generator is the starting point for each question. This module creates at random (with simple rule based restrictions) an equation to be used in the math question. Note that an equation and a question are not the same thing. The equation represents the last step the user would reasonably be expected to solve in order to find the correct answer to the question, though this is not required as solutions can be achieved in many different ways. The equation generator will use all available number types, available operations, and comparators (=, <, >, etc).

Equation Categorizer

The equation categorizer takes in an equation from the equation generator and uses its category detection strategies to determine which types of questions can be created from the equation. A category is enabled or disabled based on the the user's selected skills. There can be one or many categories per skill. Categories do not overlap and may be duplicated to accommodate this trait.

All category detectors must implement isCategory(). Each implementation is expected to be straightforward and simple, although there are expected to be over 100 defined detection strategies. Category detectors must implement 2 additional translator methods called label and populateQuestion, to be used by the question translator.

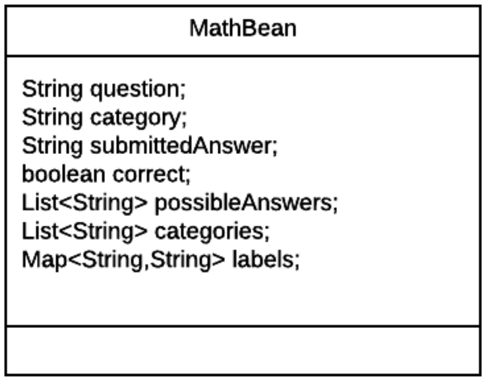


```
public abstract boolean isCategory();
public abstract void label(MathBean mathBean);
public abstract void populateQuestion(MathBean mathBean);
```

Question Translator

A question translator will take in a math bean and populate it with a math question based on the category chosen by the equation categorizer and the equation created by the equation generator. The translation step includes

Math Bean



```
String question;
String category;
String submittedAnswer;
boolean correct;
List<String> possibleAnswers;
List<String> categories;
Map<String,String> labels;
```

A math bean is a transient object containing everything needed to return to the front-end to display a question as well as everything needed to pass between the different modules in order to create the question. A Math Bean will be persisted in the database for test validation and grading after a user completes a test. Included in a math bean is the question, a list of possible answers, the list of possible category types (stored in case of recategorization), the chosen category, and labels required to create a diagram on the front end. This object will be passed between many modules while generating questions, persisted to the database for validation and grading, and passed to the front-end for question display. Sensitive fields will be stripped from the math bean before returning to the front-end.

Display

The type of test the user wants to generate will be selected through a browser in the UI. Tests can be cumulative or targeted based on skills selections.

To generate a test, the UI will request math beans from the backend via the scala template, seen as questions from the UI perspective, for display to the user. The Math Bean will be used to create the UI for the question, possible answers, and diagrams. Diagrams will be drawn on the front end using an HTML canvas and, in some cases, a javascript library. Charts and graphs will be created using HighCharts.

Any associated labels will be populated and text questions will be used in any form returned from the back-end. Any categories requiring a diagram will be drawn on the front-end creating scale and diagram labels based on the labels in the math bean. Questions will not always be entirely to scale in order to facilitate clear labeling of the questions.

The diagrams in the UI are the most complicated section and each type will be generated by way of a javascript function. The javascript function will draw exactly 1 kind of diagram, but it will draw it dynamically to adjust for scale and, in the case of charts, type (pie, line, bar, etc). The diagram will be labeled by the function parameters, which come from the MathBean labels. There will be approximately 50 javascript functions to draw diagrams.

There will also be a UI for the user to check their progress on the tests and their current skill levels for each skill category. HighCharts will be used to generate the charts related to user progress and skill level.

32 Week Phase I R&D Plan

The Remind Math Project will adopt an agile methodology consisting of 2 week sprints and release trains consisting of 4 sprints. This will be a total of 4 release trains. The first 2 release trains represent the Fall semester and the second 2 release trains represent the Spring semester. Testing for each sprint will be included within the sprint. Another way to put this is that each user story in a sprint will not be considered complete or accepted until unit and integration testing is complete.

Release Train 1

Release Train 1 incurs a heavy research and setup cost necessary for smooth development. This does not mean there will be no deliverables. Tangible milestones will include equation generation and categorization procedures.

Sprint 1.1: The first sprint will be spent researching current implementations of products filling similar roles. These implementations will be analyzed for both their good qualities (for inspiration) and bad qualities (to improve on). Commercial approaches and cost models will also be explored during this research into competitive products. At the end of sprint 1, a good list of requirements and expectations should be set on the project in terms of goals and workflows.

Sprint 1.2: This sprint will be spent researching development and infrastructure frameworks to support the creation of the Remind Math Project. At the end of this time, a project should be set up and deployed with no functionality, but basic support so that you are ready to start writing and deploying code to create the Remind Math Project. All major initial and systemic hurdles related to frameworks and hosting should be solved by the end of this time box.

Sprint 1.3: The most straightforward and first step of generating a question is the random creation of a math equation. This is a good opportunity to set up the basic code design including a flexible contract between the front and back end aspects of the system, a UI capable of displaying the response based on that contract, and the back-end system capable of populating the contract with values for the UI to present to the user. The deliverable at the end of Sprint 3 is a web page displaying randomly generated equations.

Sprint 1.4: The next step is skill categorization of the equations. An extensible implementation is needed to cover the many expected skill categories including some methods that will aid skill detection modules. Each detection module will represent 1 skill, though there may be multiple detection modules per skill. The most important deliverable of this sprint is the flow from equation generation to equation categorization such that new skill categories can be added quickly and easily. That being said, there is also an expectation of having 10 skill categories and detectors completed by the end of this sprint. The deliverable for Sprint 4 is an updated UI to display both the randomly generated equation and the categories that the equation belongs to.

Release Train 2

The completion of release train 2 results in a full process workflow to create full questions for the Remind Math Project. This includes skill selection, complete question generation, display, and answer choices.

Sprint 2.1: The first sprint in release train 2 will be to create a way to generate formal questions based on the equations and categories generated in release train 1. The deliverable for this sprint will be the display of arithmetic problems in 2 formats (horizontal and vertical). These equations should be visible in the UI. Other equation types should still be categorized, but no formal display will be available at this stage.

Sprint 2.2: This sprint will include the first drawing tasks for the project. Two primary features will be necessary to draw a question in the Remind Math Project. First, there needs to be a system of labeling. Drawings with no labels would not produce math questions that a user could solve. Second, the category and labels need to be used by the front-end to draw the shapes necessary to display a question with a diagram. The scope of this sprint is to create the system of labeling and diagram drawing in a duplicatable and, where possible, extensible way. The deliverable of this sprint is the creation of diagrams representing circles, squares, rectangles, and triangles. These diagrams should be included in the UI and output in a browser alongside their equation and category with a text based question for the user to solve based on the diagram.

Sprint 2.3: In the third sprint of the release train, we will generate the selectable answers related to the question. This will create 1 correct answer and 3 incorrect answers. The answers will be randomized in terms of order. The correct answer will be solved based on the equation generated in release train 1. By adjusting the equation, we will generate plausible incorrect answers as well. The deliverable of this sprint is an updated UI that includes the newly generated answer choices.

Sprint 2.4: The final sprint of this release train will be the creation of skills selections in the UI. By this sprint, skills should exist in the form of skill categories and detectors in the backend, but there is not yet a way for the user to select the skills they wish to be tested on. The deliverable of this sprint is a UI that the user can select the currently available skills and generate questions related to those selections.

Release Train 3

This release train is about expanding on the groundwork completed in the previous 2 sprints. The milestone achievement from release train 2 is an end-to-end solution to create questions, but it is limited in terms of categorization, translation, and display.

Sprint 3.1: Expanding on the categorization completed in sprint 1.4, a full catalog of math skills related to the GED must be collected and categorized. The expectation is that there will be approximately 100 skills in total. The groundwork is already available from previous work. All that remains is grouping the skills and implementing each one. The deliverable for this sprint is an updated skills selection UI so the user can select all of the new skills and an updated question display so you can see the correct categorization of the equations for the new skill categories added in this sprint.

Sprint 3.2: This sprint will be dedicated to the expansion of translation methods. Like categorization in sprint 3.1, the groundwork already exists for this. Unlike categorization, however, translation is a higher level of effort per skill. As such, this sprint will be dedicated to the translation of non-diagram based questions like algebra and logic as well as 3D geometry based questions like cubes and spheres. The deliverable of this sprint is an updated UI that includes the display of diagrams and labels for 3D geometry based questions and questions that do not include diagrams.

Sprint 3.3: Continuing the progress in sprint 3.2, this sprint will focus on chart based translations, particularly those related to statistics and graphical analysis. The deliverable of this sprint is an updated UI that includes chart and graph based questions

Sprint 3.4: The last sprint in release train 3 is for developing the grading mechanics. The UI will need to be updated to allow answer selection and test submission. The backend will need to create a method to grade the test and provide a test score. The deliverable of this sprint is an updated UI where the user is able to select answers, submit the test, and receive a grade.

Release Train 4

The theme of release train 4 is user progress. By the end of this release train, the Remind Math Project should include a history of the user's tests, reporting on their progress, and measurements of their skill levels overall and by skill category.

Sprint 4.1: The first sprint in release train 4 is to create a mechanism of persistence. Specifically, the graded tests will need to be stored in the database and associated to the user that submitted the test. Each question and answer stored will need to include the skill category, the question level of difficulty, and if the user answered correctly.

Sprint 4.2: This sprint will be dedicated to creating the user's skill level per category. Based on the records created in the database resulting from the work in sprint 4.1, we can analyze skill based success rates of the user's answers to generate the skill levels for each category. The deliverable for this sprint is a UI where the user can see all of their current skill capabilities with a score associated to their current level.

Sprint 4.3: Sprint 4.3 will be dedicated to the development of the exam readiness metric. Exam readiness will be calculated based on the user's skill levels for each category. Categories the user has no experience with will result in a score of zero. The deliverable for this sprint is the UI from sprint 4.2 updated to include an overall "Exam Readiness" rating at the top of their skill ratings. The UI should also be updated to allow targeting of the skills the user is struggling with, at their discretion, to improve their overall readiness score.

Sprint 4.4: The final sprint of release train 4 and all of Phase 1 is to create the final test that includes all skills in the math portion of the GED exam and stores them to include a full history of the user's test progress over time. Key features of this sprint are a mechanism to create skill category distributions such that the test reflects a ratio similar to the real math section of the GED, an expansion of that concept to all tests, except as a normal distribution, and the persistence in the database of the test results, marked for separate analysis from other tests. The deliverable of this final sprint is an updated UI where the user can take a GED math prep test including all skills and review their progress over time for these tests.

Contact Information

For questions about this documentation, please contact Eric Stanton by email at stanton.eric@gmail.com.