Internship Mid-Report:

Enhancing Markdown with a Widget System

Abdelmojib Chouhbi

Al Akhawayn University in Ifrane

Field Supervisor: Mr. Abderrahmane Bouamri

Academic Supervisor: Dr. Nasser Assem

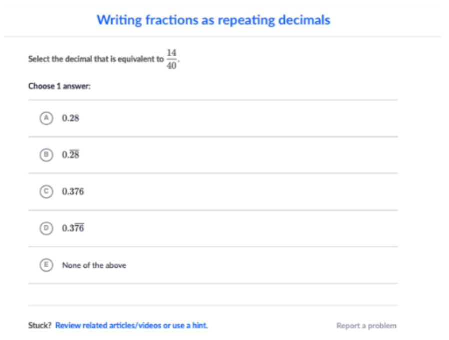
Contents

Introduction

The purpose of this internship is to develop a new question editor for the internal uses at Mathscan, and the software to support it.

Company Overview

Mathscan is an online learning platform designed to identify and address learning gaps in mathematics. Mathscan's online exercises allow learners to pinpoint and work on their specific weaknesses. Mathscan operates as a Software as a Service and focuses on serving private schools in Morocco and internationally.



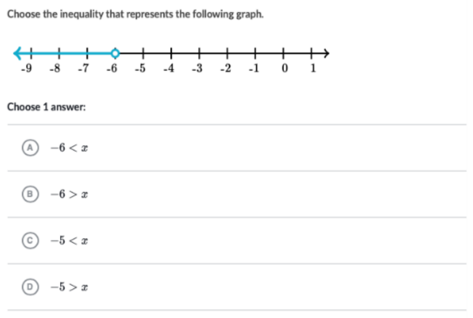
Writing fractions as repeating decimals

Select the decimal that is equivalent to $\frac{14}{40}$.

Choose 1 answer:

- Ⓐ 0.28
- Ⓑ $0.\overline{28}$
- Ⓒ 0.376
- Ⓓ $0.3\overline{76}$
- Ⓔ None of the above

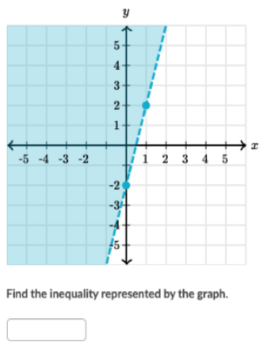
Stuck? Review related articles/videos or use a hint. Report a problem



Choose the inequality that represents the following graph.

Choose 1 answer:

- Ⓐ $-6 < x$
- Ⓑ $-6 > x$
- Ⓒ $-5 < x$
- Ⓓ $-5 > x$



Find the inequality represented by the graph.

The exercises offered are varied in interactivity. They go from simple multiple-choice questions to game like experiences such as dragging and dropping elements, stretching columns in a plot or even drawing points and lines in graph.

Current System

To create these questions, Mathscan utilizes Perseus. Khan Academy's exercise question editor and renderer. Perseus makes it simple to create interactive problems.

As it is under MIT license, the source code is available and can be scrutinized. From reading the code, the following 3 points are deduced:

- Perseus is essentially a Markdown Editor and Renderer.
- Perseus injects custom components into the markdown to provide interactivity. This report will refer to these components as "widgets" from now on.
- Perseus uses Reacts. Further research indicates that the development of Perseus has closely paralleled that of React. In fact, Perseus stands as one of the earliest use cases of React.

Problematic Analysis

The current system has some drawbacks:

- Lack of Answer integrity: Perseus checks the answers locally. Thus, the correct answers are passed to the client. Which means the learner can intercept the correct answer.
- Lack of Usability: Some widgets are not intuitive for the learners to use.
- Lack of Freedom: Some exercises are not possible to design with the available widgets.
- Lack of Control: Mathscan cannot easily modify or create new widgets.

Solution Approach

Mathscan decided to develop an in-house solution for their exercise questions. The solution will be divided into three main sets:

- Question Editor
- Question Renderer
- Widgets

To provide a proof of concept for the new solution, the internship will not only involve the development of the editor but also the renderer and a series of widgets.

Current Progress

Overview

After discussion with Hamza the question writer at (the primary user of Perseus's editor).Further experimentation with Perseus and research into design editing tools, particularly two conference talks, have guided the development:

- Creating Interactive Learning Interfaces at Khan Academy - @Scale 2014 – Web
- "Diagrammar: Simply Make Interactive Diagrams" by Pontus Granström (Strange Loop 2022

Key Specifications

From this research, the following specifications have been established:

- The Editor shall be a WYSIWYG (What You See Is What You Get) text Editor.
- The Editor shall extend the traditional WYSIWYG features (headers, bold, italic) with the ability to add widgets similarly to how any markdown element can be added.
- The widgets parameters shall be editable through a custom UI and not directly through text.

- The markdown text shall only contain references to the widgets, while actual widget data will be stored in metadata. Thus, preventing large and unreadable markdown files.

Thus, our requirements include:

- An extendable text editor for integrating custom features.
- An extendable markdown renderer for rendering numerous custom widgets.

Technology Selection

Text Editor

After researching web-based text editors, CodeMirror was selected for its:

- Excellent documentation
- High extendibility
- Built-in lifecycle for managing text changes (important for detecting widget references)
- Widespread industry use, providing additional resources.

Markdown Renderer

When it comes to the render the chosen solution is the react package react-markdown because it:

- Integrates easily with other components
- Is highly extendable (ideal for adding LaTeX support)
- Parses markdown as an Abstract Syntax Tree (useful for handling custom elements like widgets).

Challenges and Learning

Dom manipulation with React:

Direct DOM manipulation can disrupt React's lifecycle. We encountered the following challenges:

Integrating CodeMirror

CodeMirror is not a react library and conducts direct DOM manipulations. In order to interact with react we set up a React component that returns an empty <div> element. We initialize CodeMirror within a useEffect hook, passing the empty <div> as our ref using the useRef hook.

This works because from React's perspective, the <div> contains nothing, so it does not monitor changes within it, allowing CodeMirror to function without interference.

Rendering Widgets with SVG

For certain widgets, SVG were used for rendering, which involved DOM manipulations that complicated integration with React. This was addressed using specialized libraries like mafs.

CI/CD Setup

At the beginning of the project, a Storybook app was established to showcase the developed components, allowing project stakeholders to test them.

Later, testing older questions with the new renderer required setting up a Laravel project, which necessitated copying components into the Laravel app for each modification.

This manual process meant we had to repeatedly copy and paste components into the Laravel app for every change.

To streamline this, the workflow was automated by extracting the components into their own Git repository and using them as git submodules in both projects. This approach ensured that changes made in one location would be accessible to all projects through git commits.