Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Title

Author:
Your name

Supervisor:
Name of supervisor

Submitted in partial fulfillment of the requirements for the MSc degree in Type of Degree of Imperial College London

August 17, 2022

Abstract

Your abstract.

Acknowledgments

Comment this out if not needed.

Contents

Chapter 1

Introduction

The application of remote learning in higher education has increased massively since the explosion of COVID-19 [1]. Many institutions including Imperial College London moved their courses online during lockdowns. These courses are taught based on online systems. However, some online systems are not capable of conducting the digitalisation of teaching tasks such as lecture recording/playing and assignment marking etc. For example, some departments of Imperial College London were using Möbius for online assignment and assessment, but it was complained by students and staffs that the system was hard to use. This thus leads to a demand of updating or developing new systems so that students and teachers can get better experience.

Therefore, the college has decided to develop a new online learning system named Lambda Feedback. This system is a web application which allows students to do assignments posted by teachers and get feedback immediately. Comparing to its predecessor Möbius, Lambda Feedback has modern user interface, friendly interaction and much more features. One of the features is the expression input. In many assignment questions, solutions are math expressions. In Lambda Feedback, users can input math expressions in a text field, and the system is able to understand the expression and check if it is correct. Figure 1.1 is a screenshot of a demo in Lambda Feedback. The webpage displays a navigation sidebar, the question, and the answer area, where users can input the expression.

However, the current system only supports keyboard input for math expressions. Users need to type the whole expression including symbols and operators, which can be very inconvenient and complex especially for complicated equations. Consequently, an extension with a new method of input is needed for the system.

Handwriting input is considered appropriate for the system because of its convenience and simplicity. Although there are other input methods such as equation editor and image upload, they either require users to learn how to use, or need some extra steps. This means they are not suitable as the major approach of expression input, but can be a good supplement to handwriting input in some special situations.

This project thus aims to develop a component with handwriting input and equa-

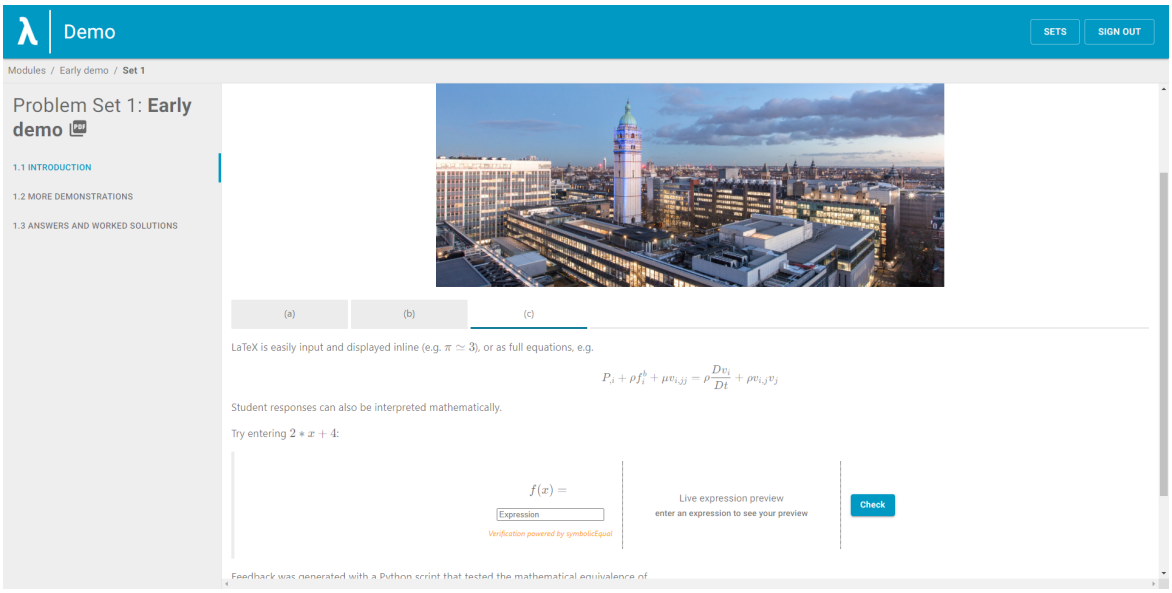

Figure 1.1: Screenshot of a Demo in Lambada Feedback System

tion recognition, and integrate it to the Lambda Feedback system. The component shall enable the users to write on it either via mouse/touchpad on computers or fingers/pens on mobile devices. After users finish input, it should be able to recognize the equation and display the result alongside the writing area. Integration with the Lambda Feedback system should be carried out after the component is tested available.

This report records the development of the project. Chapter 2 investigates several approaches to realise the software component and analyses their advantages and drawbacks. Chapter 3 states the progress of development and explains the implementation of the software component. In Chapter 4, progressive results are demonstrated and the final achievement is shown. An evaluation of the results are included in this chapter as well. Finally, in chapter 5, a conclusion of the project is drawn, and possible future improvements are listed.

Chapter 2

Background

This chapter records the background research, which includes general reviews on articles, tutorials and existing projects that have helped the development of this project, as well as potential choices that could be taken and why they were discarded, along with the reasons of selecting current tools. An overview of the project is made below before diving into details.

Overview

The component to develop is a TypeScript (strong typed version of JavaScript) web application which has two major functionalities:

1. Have a canvas where users can write equation
2. Ability to recognise the equation written on canvas and display the result

Moreover, the component should be integrated into Lambda Feedback, so it is necessary to introduce the system because they will use same frameworks and language.

Lambda Feedback is a web application which has a frontend and a backend. The frontend is built upon *NextJS* (a framework built on *React*), and the backend is built upon *NestJS*. Both are written in Typescript (These frameworks will be introduced in Chapter 2.3). The major part of the project will be installed in a component of the frontend named *ResponseArea*, which is responsible for receiving input and giving result to users.

It is difficult to develop the component directly upon Lambda Feedback, as it already had much code when this project launched. Therefore, it was decided to build a web application independent of Lambda Feedback to save time on reading code and testing functionality.

Although the final independent web application must be written under the same framework with Lambda Feedback, i.e. *React*, there is another way, *Vanilla JavaScript* (also referred to as plain JS), to start with. Table 2.1 compares the two methods:

Table 2.1: Comparison of Two Methods for Independent Web Application

Method	Advantages	Disadvantages
Vanilla JavaScript	Easy to start & Require no extra configuration	Does not fit Lambda Feedback
React	Uses same framework as Lambda Feedback - does not need extra steps for integration	Need to study React before start

After careful consideration, it was decided to start by using *Vanilla JavaScript* to get familiar with functionalities, and then converting the application to *React*. This makes the procedure more fluent and only need to focus on one emphasis for every progress.

Overall, the procedure of the project is:

1. Develop a web application with *Vanilla JavaScript* which enables users to draw equations, and recognise the equation
2. Move the web application under *React* framework
3. Integrate the application as a component to Lambda Feedback

Despite using different frameworks during the procedure, the approaches to realise functionalities are the same. Hence, the background research mainly focuses on three parts:

- A canvas where users can draw stuffs
- Functionality of mathematical equation recognition
- *React* framework and integration with the Lambda Feedback

Therefore, the remaining part of this chapter is divided into three sections, each introducing one part.

2.1 Canvas

A handwriting canvas is needed for the component, which should do the following:

- Users can draw and edit stuffs
- The canvas can output the content, either by image or stroke data

Here, stroke data means positions that represent points on the strokes. For example, stroke `{1, 1}, {2, 2}, {3, 3}` means a line starting at point (1, 1), passing point (2, 2) and ends at (3, 3). All paths drawn on the canvas can be expressed as stroke data.

Two HTML elements - `canvas` and `SVG` - were investigated for this part.

The `canvas` element is an HTML container used to draw graphics on web page. Graphics are drawn by APIs, which has several methods for drawing, including paths, boxes, circles, text, and adding images [2].

The component can utilize its paths method to display the user's strokes: once the user drags the pointer, the component fetches its position in the canvas, and draws a path to the next position the pointer moves to.

The `canvas` element also provides APIs to output content as images, while stroke data is generated when recording positions of the pointer.

Drawing on `canvas` is easy because all graphics is drawn by APIs, so the component only needs to 'tell' `canvas` how to draw. However, there are two drawbacks of this element:

1. To use `canvas` APIs, the HTML element should be referred. This in *React* will be using `React Refs`. Unlike in *Vanilla JS*, API callings in *React* should be carefully considered with *React Component Lifecycle* [3]. This will affect the performance of the component and brings extra work.
2. The `canvas` element is represented as a Bitmap in webpage. This means when transforming the size of the canvas, graphics might look blurry [4]. This will be further discussed in Chapter 3.

In contrast with `canvas`, the `SVG` element does not have listed problems. The term SVG (Scalable Vector Graphics) means an XML based vector image format for defining 2D graphics. Therefore, as an element in HTML, `SVG` simply represents a container for SVG graphics.

Because it is 'vector graphic', regardless of the ratio of rendered size and programmed size, users can always get a clear graph of what they draw. This thus solve the blurry problem of `canvas`.

`SVG` has several sub-element including `Path`. Data of a `Path` is a string containing commands and points, and is attached to its property 'd'.

To show strokes that a user draws, instead of containing them in one `canvas` element, multiple `Path`s are used to display the user's drawing, each for one stroke. It is more flexible for the programme to monitor and control them.

This feature fits *React Component Lifecycle* perfectly as any change of path data causes *React* to 'react' to it, and render the path with updated data.

Overall, `SVG` is more suitable than `canvas` for this project due to its vector property and compatibility with *React*.

2.2 Equation Recognition

2.3 Integration with Lambda Feedback

Chapter 3

Implementation

Chapter 4

Experimental Results

Chapter 5

Conclusion

Bibliography

[1] Ali W. Online and remote learning in higher education institutes: A necessity in light of COVID-19 pandemic. Higher education studies. 2020;10(3):16-25. pages 1

[2] Canvas API - web apis: MDN;. Available from: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API. pages 5

[3] React Component Lifecycle;. Available from: https://reactjs.org/docs/react-component.html#the-component-lifecycle. pages 5

[4] Spiess P. How to build a freehand drawing using react;. Available from: https://pspdfkit.com/blog/2017/how-to-build-free-hand-drawing-using-react/. pages 5