TRiTeX: A Browser-Based LaTeX Assistant for Beginners

Trista Yang

Master of Science
Advanced Technology for Financial Computing
School of Informatics
University of Edinburgh
2022

Abstract

TRiTeX is a browser-based self-help LaTeX tool that checks five main problems that happen commonly in beginners' writing. The problems include commands used inappropriately, use of obsolete commands and packages, lack of knowledge of LaTeX's features, packages loaded in the wrong order, and ugly layout. The problems are detected by pre-defined regular expression tokens with flexible matching patterns. The problematic text is then transformed into suggested text. The text that has unique identifiers, such as quantities or citation name, need to extract the identifier by a regular grammar parser. The work is evaluated by two user studies in the form of questionnaires, one on the prototype and one on the final version. We produce statistical tests to assess the improvement we made and analyse the results by the scores and scales. The final version has an aesthetic and intuitive user interface with more than 20 rules defined, each matching various patterns.

Research Ethics Approval

This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: 46726
Date when approval was obtained: 2022-05-31
The participants' information sheet and a consent form are included in an appendix, see Appendices B.1 and B.2.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Trista Yang*)

Acknowledgements

Time is a river with many eddies. Edinburgh is the last stop of my study abroad journey and I have drawn a perfect full stop here. The months have been challenging, I have only been able to finish this project because of the dedicated support I have received from so many people along the way.

First and foremost, I would like to express my strongest gratitude to my supervisor Brian Mitchell for his continuous and invaluable guidance throughout the project. He is not only a conscientious supervisor, but a knowledgeable teacher, and an earnest friend to me. I am definitely fortunate to be attracted by his project description and to have him as my supervisor.

Then I would like to thank my parents for their encouragement, patience, and love throughout my life journey. My mom is always the best listener whenever I am down or lost. I never felt alone when she was around. Though my dad is not good at expressing himself, I can still feel he is loving me in his own way.

My final thank goes to my grandmother who left us this May. She is the most important person in my childhood who brought me up during the years my parents are busy with work. I did not get to say goodbye in person because of the pandemic but I promised in our final call that I will study hard and take care of myself. Now I am here, with my acknowledgement as my farewell.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

LaTeX is a high-quality document typesetting system that is widely used by scholars especially in science, technology, engineering, and mathematics (STEM). Different from word processors which are What You See Is What You Get (WYSIWYG, pronounced whizzy-wig), LaTeX is What You See Is What You Mean (WYSIWYM, pronounced whizzy-whim) that favours logical markup. The system only accepts plain text as input and all formatting work has to be done by programming-like commands. The design separates the presentation and content to help users focus on the content but this also makes LaTeX hard for beginners.

Another difficulty for beginners is caused by the version of LaTeX. LaTeX 2ε is the main version that has been widely used since 1994 and it still dominates LaTeX3, the version improved by the experience gained from LaTeX 2ε, even though LaTeX3 started in early 1990s. This indicates that the current commands and templates should become legacy in the near future but users will still find them on the internet. Those out-dated commands may not work properly or can even crash the document production process.

This project is about designing and creating a browser-based tool, TRiTeX, to assist Edinburgh's Informatics Master's students when writing project documents in LaTeX. These students, especially those lacking high-quality formal training in LaTeX, are at the mercy of the internet's stockpile of examples that are out-dated, fragile, ugly, or just plain wrong. The boundary between LaTeX being just a means to an end (a document without "funny" fonts where it is difficult to fiddle the page count) and a genuine transferable skill (when used "properly") is, to use a LaTeX term, a rubber length. The

Informatics School's justifiable pride in teaching high-quality modern programming could be extended to LaTeX by using a self-help tool like TRiTeX.

1.2 Problem statement

TRiTeX is designed to solve five main problems (Table 1.1) that are significant in beginners' LaTeX. All these problems are addressed by drawing on a curated collection of flexible substitutions and templates. The tool will detect the problematic text and suggest corresponding correct text to users.

This document makes use of our discovery for each type of problems, for example:

1. **inappropriate use of commands**
   `\bigbreak` is used instead of the much-abused `\\` to generate appropriate white space between the blocks of the engineering requirements (Appendix A);
2. **out-dated LaTeX 2$_\varepsilon$ instead of LaTeX3**
   all the instances and variations of the LaTeX logo are wrapped in a `\hologo{}` command so that they can be used safely in headings, captions, bookmarks, and macros generally;
3. **not realising what can be achieved**
   Table 1.1 loads its descriptions from an external file and the item numbers are generated automatically;
4. **not making proper use of packages**
   the `geometry` package is loaded *after* `hyperref` as it should be; the `cleveref` package is used for cross-referencing (words like 'Table' and 'Figure' are part of the hyperlinks) and `cleveref` is loaded *after* the package `siunitx` which is used to typeset numbers and quantities correctly (`\qty{10}{\percent}` not `10\%`)

Number	Description
1	inappropriate use of commands
2	out-dated LaTeX 2$_\varepsilon$ instead of LaTeX3
3	not realising what can be achieved
4	not making proper use of packages
5	ugly design

Table 1.1: Common LaTeX problems

and create properly formatted number columns in tables (for example the first column of Table 1.1 is type `S` from `siunitx`);

5. **ugly design**

   there are no vertical lines in Table 1.1 because they disrupt the horizontal reading process especially for people with dyslexia, and the document uses the `booktabs` package to improve the appearance of tables.

1.3 Research objective

This project aims to design a tool that actively assists beginners in writing LaTeX with an easy-to-learn user interface and strong features. The research objectives can be broken down into the following research questions:

RQ1. What kinds of problems should be detected by TRiTeX?

RQ2. What technology should be used for TRiTeX's development?

- **RQ2.1** Programming language
- **RQ2.2** Integrated development environment
- **RQ2.3** Framework and packages

RQ3. What aspects should be considered to make TRiTeX's interface more usable?

RQ4. What method should be used to capture the problems? How can we generate the suggested text that retains the information from the source, such as quantities in numerical expressions and the identification in citations?

RQ5. How should we evaluate the design of TRiTeX?

RQ6. How is TRiTeX's performance comparing to other LaTeX self-help tools?

1.4 Project overview

This project contains three parts: front-end, back-end, and evaluation. Front-end refers to the graphical user interface, including the input text area, generated suggestions, and a filter that can view suggestions by tag. We design and evaluate the front-end design by Human-Computer Interaction principles. Back-end is the core part of TRiTeX: it uses a regular grammar with regular expression tokens to identify problems in users'

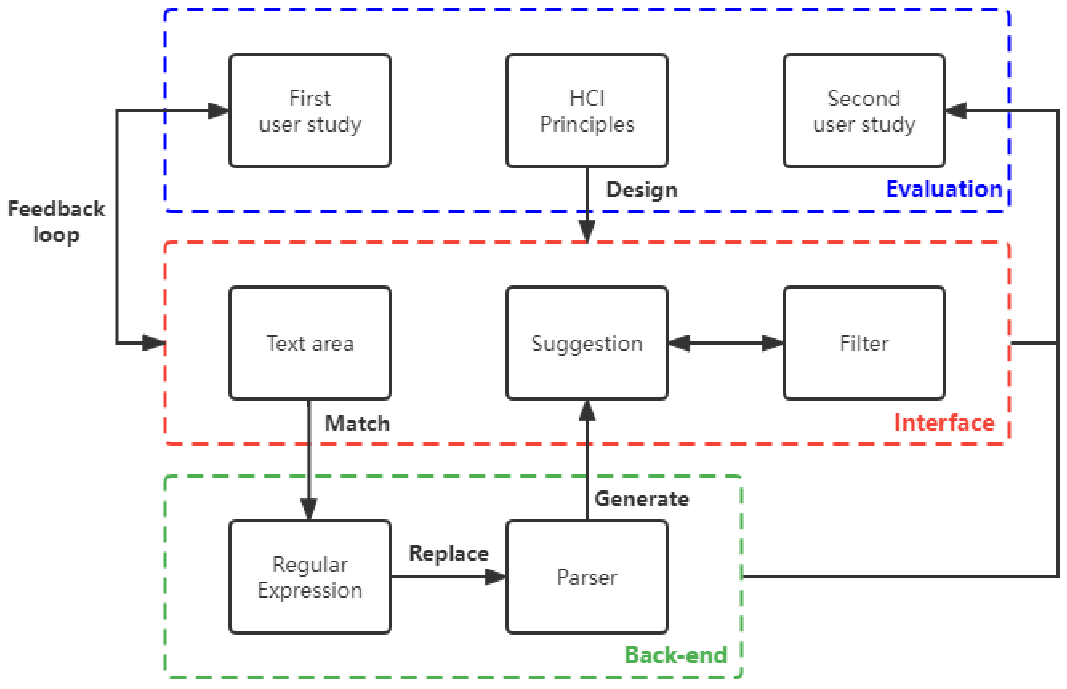


Figure 1.1: Project overview

LaTeX and the problematic text will be transformed into suggested correct LaTeX. The front-end and back-end make up the implementation work of TRiTeX.

Evaluation is also important for a human-centred tool, so we designed two user studies to receive feedback from our target users. The first study is on TRiTeX's prototype, to create a feedback loop that can improve the overall design. The second is on TRiTeX's final version, with the same questionnaire, to see the result of improvement. Figure 1.1 outlines the project.

1.5 Dissertation structure

Chapter 2 introduces LaTeX versions and packages similar to TRiTeX with discussion on their limitations and shortages. Chapter 3 describes the whole design process of the user interface, including software selection, design principles, and improvement based on the feedback loop. Chapter 4 illustrates the logic of the back-end checker with examples of each type of rule. Chapter 5 introduces the evaluation process, including the comparison and analysis of the results of two user studies. Chapter 6 concludes our current work and suggests future improvements.

Chapter 2

Background

Although LaTeX distributions have not included tools really similar to TRiTeX that can detect several types of problems and promote LaTeX3, there are some LaTeX packages that can detect obsolete or inappropriate commands which we can take as reference.

`l2tabu` is a collection of severe mistakes and obsolete commands with corresponding hints how to correct them (Trettin & Ensenbach, 2016). `l2tabu` guides users to employ LaTeX 2_ε properly. Legacy packages and commands from earlier versions are introduced in the document, each with an explanation. The authors list suggestions on obsolete commands, and indicates the outcome when using old commands.

Although `l2tabu` provides useful examples and instructions, users have to check its documentation manually to improve their LaTeX writing. This processed is automated by the `nag` package (Schwarz, 2011). `nag` detects commands, classes, and packages that are superseded and warn the users. The warning generated by `nag` is the simplified version of explanation in `l2tabu` document. Listing 1 shows the warning when the outdated command `\bf{}` is used. This approach can be helpful for users who want to improve their LaTeX 2_ε usage but are tired of reading `l2tabu`'s long documentation. Moreover, automating detection is an obvious benefit. Another worthy aspect of `nag` is it provides a deliberately "horrible" LaTeX document as a demonstration which includes outdated commands that can be detected by the package. This is a useful teaching tool.

```
Command \bf{} is an old LaTeX 2.09 command. Use \bfseries{} or
    \textbf{} instead on input line 114.
```

Listing 1: Example `nag` output

`chktex` (Thielemann, 2016) is also a tool with features relevant to TRiTeX. The package's name is an abbreviation of its usage. Whereas `nag` detects legacy commands, `chktex` detects typographic and other errors in LaTeX. For example,

`\LaTeX is a typesetter.`	LaTeXis a typesetter.
`\LaTeX\ is a typesetter.`	LaTeX is a typesetter.

The first line is a typical error detected by `chktex` as "Command terminates with space." The second line shows its correction. Note that not only is the source different, so is the output: the first command has no space between LaTeX and 'is' whereas the second does. `chktex` supports over 40 warnings, each with a corresponding detailed explanation in the documentation. The package is fully customizable and can adapt to different environments and configurations. However, `chktex`'s flexibility could mean beginners may find it hard to install, configure, and extend.

Although `chktex` and `nag` do not have known bugs and work well with other LaTeX packages, both of them have a severe limitation: they are already outdated. Some of `nag`'s recommendations to replace obsolete commands are themselves already obsolete in LaTeX3. The situation is arguably even worse for `chktex` as the package cannot understand LaTeX3's `expl3` syntax. `expl3` is LaTeX3's programming layer which aims to provide 'modern-programming-language-like syntax.' When `chktex` encounters `expl3`, it has a meltdown. The market currently lacks an up-to-date and easy-to-install LaTeX self-help tool like TRiTeX, illustrating the importance and novelty of TRiTeX.

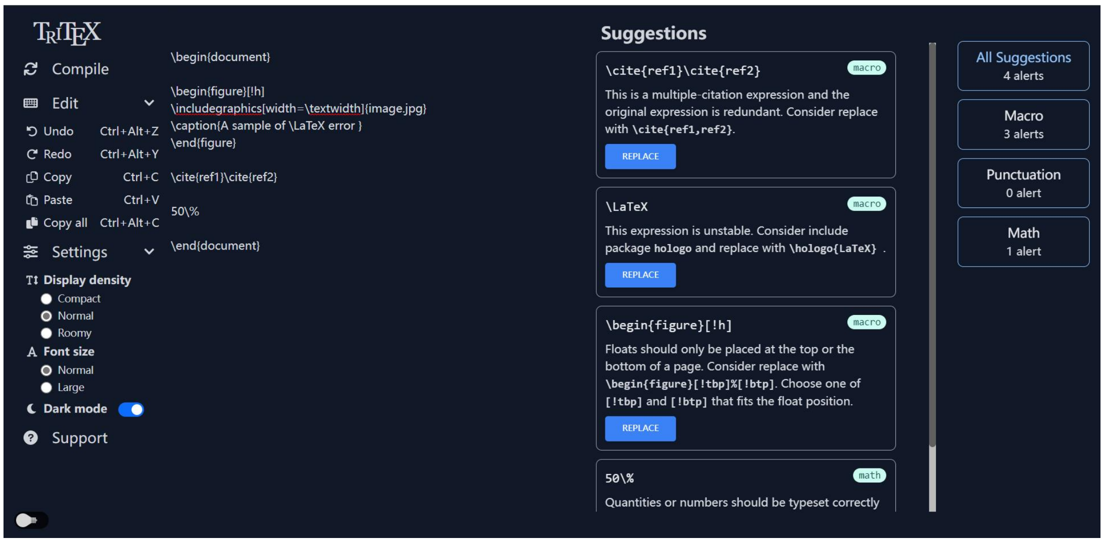
Chapter 3

User interface

The graphical user interface (GUI) and resulting user experience (UX) are critical to the project. Generally speaking, the GUI should be easy and intuitive. We are introducing the whole GUI design process in this section from differing aspects: software selection, design principles, design process, and improvements gained from the feedback loop. Figure 3.1 shows the final interface of TRiTeX in dark mode.

3.1 Software selection

TRiTeX is a browser-based tool which means Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript are the core technologies used in the development. HTML describes the structure of the web page, indicating the types of



```
\begin{document}

\begin{figure}[!h]
\includegraphics[width=\textwidth]{image.jpg}
\caption{A sample of \LaTeX error }
\end{figure}

\cite{ref1}\cite{ref2}

50\%

\end{document}
```

Display density: Compact [✓ Normal] Roomy

Font size: [✓ Normal] Large

Dark mode [✓]

`\cite{ref1}\cite{ref2}` macro
This is a multiple-citation expression and the original expression is redundant. Consider replace with `\cite{ref1,ref2}`.

`\LaTeX` macro
This expression is unstable. Consider include package `hologo` and replace with `\hologo{LaTeX}` .

`\begin{figure}[!h]` macro
Floats should only be placed at the top or the bottom of a page. Consider replace with `\begin{figure}[!tbp]%[!btp]`. Choose one of `[!tbp]` and `[!btp]` that fits the float position.

`50\%` math
Quantities or numbers should be typeset correctly

Figure 3.1: TRiTeX graphical user interface design (Dark mode)

elements such as "button," "paragraph," and "table." CSS styles the HTML including colours, fonts, and shapes. CSS is separates content from presentation and makes the maintenance of the site easier for developers. The same content can be switched to different styles for different purposes by using a CSS stylesheet. JavaScript makes the elements created by the HTML and styled by the CSS interactive.

As the author had no prior experience in web design and programming, we have compared frameworks and editors. We have compared the widely used generic IDE Visual Studio Code (Microsoft, 2016) with WebStorm IDE ("WebStorm: The Smartest JavaScript IDE by JetBrains", n.d.) which is specifically designed for JavaScript-led website development. VS Code is open-source and lightweight but needs time for configuration and extension installation. WebStorm also has extensions but is pre-optimised for JS, HTML, and CSS, including robust built-in tools such as code inspection, refactoring, and quick fixes. We choose WebStorm since its features help JavaScript programmers, especially beginners, avoid legacy, buggy, or redundant code.

Pre-built CSS frameworks like Bootstrap and Foundation are commonly used in responsive UI implementation. Bootstrap (Otto, 2000) was initially applied to TRiTeX for its easy-to-modify and easy-to-learn feature but we found it conflicts with other CSS. Investigations into alternatives made us switch to Tailwind CSS ("Tailwind CSS - Rapidly build modern websites without ever leaving your HTML." 2017) which is also easy to use but had no CSS conflicts and is lighter weight and more modern. Tailwind CSS is a utility-first framework that seems to be gaining popularity. The main difference from Bootstrap is that Tailwind does not provide predefined component classes or themes but builds from scratch. This sounds more complicated but stops the design being constrained by the predefined components while still fully satisfying our requirements. TRiTeX does not use a "typical" website layout. Tailwind supports this more readily than Bootstrap.

Compatibility is also an important factor, referring both to compatibility being cross-browser and cross-platform. We must ensure TRiTeX works consistently on different browsers and operating systems. Therefore we used ECMAScript 2015, also known as JavaScript ES6. This uses the ECMA-262 specification to guarantee cross-browser compatibility in as much as such compatibility can be guaranteed. Besides browser differences, operating systems influence compatibility: the same browser may show the site with slight differences across different platforms. TRiTeX was developed on a Windows system but primarily tested on a Ubuntu virtual machine (VM). We want to ensure TRiTeX is compatible with each major browser on both platforms, therefore, we test the compatibility of TRiTeX in the most popular web browsers on both platforms. Details and results are in Chapter 5.

3.2 Design principles

Utility, instead of visual design, is an important factor determining the success or failure of a browser-based tool. Following professional design principles can guide web designers to produce a usable and utilized site. There are many different design principles from different industries and we need to prioritize those which benefit us most. We should keep the main goals and functionality of TRiTeX in mind and choose and incorporate the principles according to our needs. We focus on creating an effective and aesthetic web design in this section. Implementation are discussed in Section 3.3.

3.2.1 Design guideline

We have distilled the guideline for TRiTeX's design according to the most popular design principles for UX and GUI suggested by professionals from four sources (Babich, 2019; Memon, 2021; Nielsen, 1994; Yablonski, 2020). Of these, Jakob Nielson's "10 Heuristics for User Interface Design" (1994) is the most general and useful one where we strictly comply with every heuristic in it.

1. **Always speak the user's language**

   We should use words and concepts users are familiar with as the users always prefer a UI that works the same as their common knowledge. A simple example is always saying "copy" instead of "duplicate." This is important as we want users to focus on TRiTeX's content rather than deciphering or learning its interface.

2. **Ensure user freedom**

   Always allow users to go back to the previous state of the site, so they can backtrack their actions easily. TRiTeX's Undo and Redo lets users restore their text in case they accidentally accepted a suggestion without first reading through it. This also empowers users to experiment safely with changes.

3. **Make design minimalist and aesthetic**

   This principle is extremely important for TRiTeX as it is a browser-based tool rather than a website with innovation. Thus the interface should only contain essential information. Irrelevant elements distract users.

4. **Make the system suitable for both novices and experts**

   We definitely want to make beginners comfortable when using TRiTeX but not at the expense of experienced users. Keeping the options on navbars but setting keyboard shortcuts on frequent actions is one way to achieve it.

5. **Provide efficient help documentation**

   Providing useful documentation is necessary and beneficial during tool development. Explain each component of the tool in concise language, clarify possible problems and make the documentation easy to search.

6. **Give users options but minimize the number of choices**

   Settings for the system should be provided to users based on their personal preferences and needs. While this does not mean more choices can improve user experience, instead, too many choices may exhaust users. We do not want "choice overload" affecting users' efficiency on the main part of the site. Leave only necessary choices to users.

7. **Avoid abusing modal dialogues**

   A modal dialogue is a window that forces users to interact with it before coming back to their current workflow. It provides a focused and contextual interaction but prevents users from accessing the rest of the interface. Modal dialogue also requires immediate attention and interrupts the users from their current work which will make it annoying when it is used in unnecessary situations. We should only use modal dialogue for important warnings and avoid presenting it with nonessential information.

3.2.2 Accessibility

Making the design inclusive is important as we want TRiTeX to be used widely, hence we take accessibility seriously. We only consider visual and cognitive disabilities as TRiTeX does not interact with users by auditory or mobility elements. The World Wide Web Consortium (W3C) has published an international standard for web content design called Web Content Accessibility Guidelines (WCAG). TRiTeX's design follows the globally recognized WCAG 2.1 AA standard.

The Coblis colour blindness simulator (2016) and WebAIM contrast checker (2019) provide lots of help in dealing with designing for visual accessibility. Initially, we considered only using colours to represent different types of errors. However that thought proved problematic for colour blindness users to distinguish error types. Colour

Blind Awareness (2022) reports an estimated 300 million people with colour blindness worldwide including 1 in 12 men. Therefore, we use additional visual cues and labels on error types to assist users with visual disabilities.

Low-vision disability is common among all ages, so all the text in TRiTeX has an appropriate contrast ratio to the background in order to guarantee readability for affected users. Allowing for accessibility also makes it readily usable for everyone. The success criteria of the colour contrast ratio in WCAG 2.1 AA is at least 4.5:1 for normal text. All components of TRiTeX meet this standard.

Besides visual impairment, users can have cognitive disabilities. Dyslexia, also known as reading disorder, is a condition causing problems with spelling, reading, and writing. We have invested substantial effort to make TRiTeX a friendly workspace for users with dyslexia. We divided the suggestions into cards to avoid large blocks of text, and intuitive icons are provided consistently to support reading since they help form an easily identified, minimal reading association between a particular problem and a suggested solution. We discarded italics to represent key shortcuts and avoided writing in block capitals. The combination of pure white (#FFFFFF) and pure black (#000000) was avoided as users with dyslexia can be sensitive to high contrast ratio. Since dyslexia affects different people differently, we also provided personal preference settings on display density and font size. Specifically 150 % inter-word spacing, which is the preferred line height by WCAG 2.1 is the default setting and a roomier option of 200 % is provided. The default font size is 18 px and a larger option of 20 px can be easily configured in settings.

3.3 Design process and improvement

We divided the design process into four components: navigation bar ("navbar"), input text area, suggestions, and filter. We initially gave each component basic functionality and then improved them with extra features to make the tool more helpful for beginners. Finally, a user study with LaTeX beginners as participants is taken to receive feedback and make further improvements. This study is important to create a feedback loop by listening to our target users' voices. This section introduces the design process of TRiTeX and the improvements we made.

3.3.1 Navigation bar

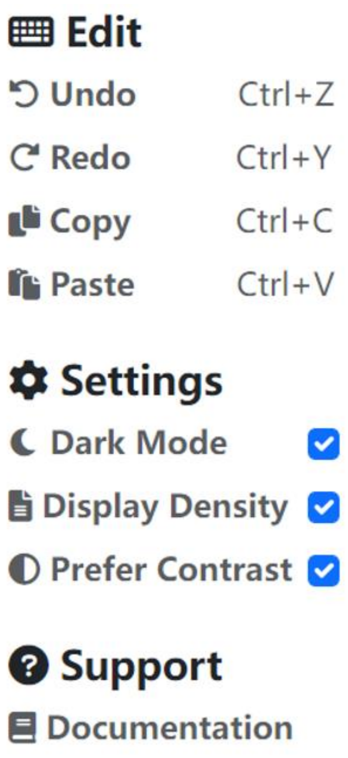
The navigation bar, usually abbreviated as navbar, holds an important role in web design to aid users in accessing information. We put the navbar on the left-hand side of the interface, providing the main features of TRiTeX. Our initial design divides the navbar into three sections, Edit, Settings and Support.

Edit provides features including Undo, Redo, Copy, Copy all, and Paste, each with a keyboard shortcut. Personal preference options are available in Settings, including dark mode, display density, and preferred contrast. Support provides the documentation, specifying the rules and features of TRiTeX.

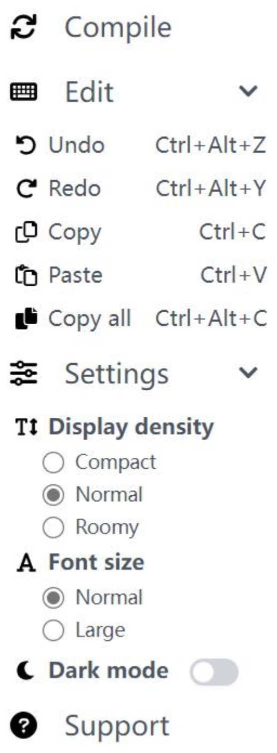
We found problems with the initial design according to our web testing process and feedback from the user study. At first, we set the shortcut of Undo as Ctrl + Z and Redo as Ctrl + Y to comply with common conventions. Undo is supposed to revert the text replacement and shows the suggestion again. However as TRiTeX is browser-based, that shortcut conflicts with the browser's built-in function. Pressing Ctrl + Z will probably execute the browser's undo function for the text area and thus undo the last change in the user's source but not show the suggestion again. Therefore, we changed the shortcuts of Undo to Ctrl + Alt + Z and Redo to Ctrl + Alt + Y respectively, preventing the conflict while keeping it easy for users to remember. Overriding browsers' in-built short-cuts have inherent difficulties and inconsistencies as any Overleaf user can testify having pressed Ctrl + R for find-and-replace only to have the browser reload the page thus losing all the edits stored in the browser's text area history.

We found more functions can be added to facilitate usage. TRiTeX is a tool, not an editor, so we expect users to write elsewhere. They paste their work into TRiTeX to check the errors and copy the amended text back to their LaTeX editor. In this case, some may use the browser's built-in function Select all, to copy the text out. We simply added a Copy all function as a component of Edit with Ctrl + Alt + C as the shortcut.

Regarding the settings, we have received feedback from the user study mentioning that the check boxes for display density and preferred contrast do not make sense. To make these clearer, we display the options of display density by radio buttons with labels: compact, normal, and roomy. In addition, we added a new preference setting for font size. It only provides two options, normal and large, to avoid overwhelming users especially as browsers include built-in page zoom for further refinement. We also removed the preferred contrast as it was initially designed to make the site more



(a) Initial design of navbar



(b) Final design of navbar

Figure 3.2: Comparison between the initial and final designs of navbar

accessible for people with visual impairment, but now both dark mode and light mode have a colour contrast within a suitable range. We used CSS to style the dark mode checkbox so that it is a toggle resembling a light bulb that can be switched on and off.

Lastly, we made changes to the overall navbar design. With much information all shown together, users may not find the feature they need quickly. We thus made Edit and Settings collapsible to give users freedom. Also, as Support only contains documentation, it is redundant to keep documentation as a sub-element. We have removed the sub-element and users can easily access help and documentation without expanding the navbar. We also added a Compile button on the top of the navbar to generate suggestions for users' input text. A comparison between the initial and the final design of navbar is shown in Figure 3.2

3.3.2 Text area

The text area is used to input and display users' LaTeX document, where we set its width as 30 % of full screen to make it suitable for users to read. We also set a placeholder text for it to give users a hint on the expected input for the field. The placeholder text is initially set as “Paste your document here to start!” but we found it causing confusion according to the user study. The placeholder does not clarify what “document” refers to as a LaTeX project can be made up of many files. Also, TRiTeX generates suggestions when Compile is clicked instead of when text is pasted, which should be explained in the placeholder. Considering the feedback, we changed the placeholder text to “Paste (CTRL+V) the whole .tex document here and click ‘Compile’ to start!” and also added instructions to the Suggestions area, see Section 3.3.3.

3.3.3 Suggestions and filter

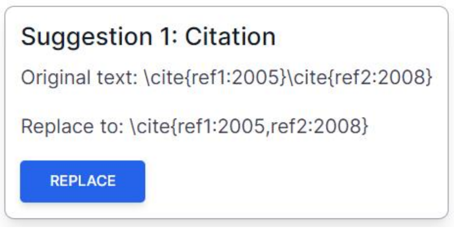
Suggestions and Filter work together, where each suggestion is shown on a card with its relevant information and the filter can select to view a specific type of suggestion or all suggestions. The suggestion card is designed to contain three pieces of information: type of error, original text, and suggested replacement text. Each card contains a “Replace” button that will replace the original text with the suggested text when clicked. The filter contains four options, all suggestions, macro, punctuation, and math. The selected option in the filter will be shown with a blue border and all other options with a grey border, where the contrast ratio is high enough for colour blindnesses to distinguish.

We found the design of the suggestion card can be improved by displaying more information on it. We added an explanation for each suggestion on why the original text is not correct and show the tag of each suggestion on the upper-right corner of the card. We also set the font of the original text and replacement text to a monospaced font, to let the user easily catch the key information on the card. Figures 3.3a and 3.3b takes a multiple-cite error as an example to see the difference between the initial design and the improved design of suggestion cards.

The initial design shows another weakness, the user will not receive any feedback if there is no error detected by TRiTeX. To avoid confusion, we designed two cards to explain the current status, one shown when the text area is empty and one shown when no issue was found for the selected type. Figures 3.3c and 3.3d shows the design for the cards, where we use icons and colours to show key information on the card.

3.3.4 Overview and documentation

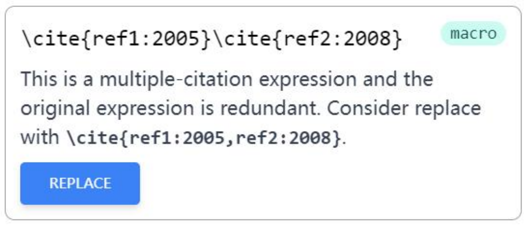
TRiTeX consists of two web pages, the main page and the documentation. Figure 3.1 shows the main GUI of TRiTeX, which strictly follows the design principles discussed in Section 3.2. The interface is intuitive with a simple but aesthetic visual design.


Suggestion 1: Citation

Original text: \cite{ref1:2005}\cite{ref2:2008}

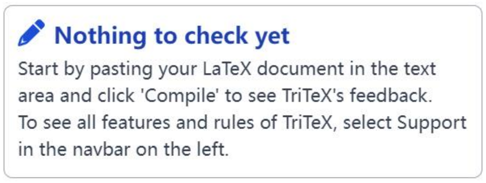
Replace to: \cite{ref1:2005,ref2:2008}

(a) Initial design of suggestion card


`\cite{ref1:2005}\cite{ref2:2008}`

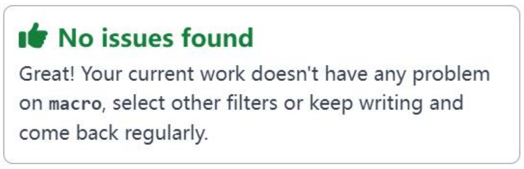
This is a multiple-citation expression and the original expression is redundant. Consider replace with **`\cite{ref1:2005,ref2:2008}`**.

(b) Final design of suggestion card


Nothing to check yet

Start by pasting your LaTeX document in the text area and click 'Compile' to see TriTeX's feedback. To see all features and rules of TriTeX, select Support in the navbar on the left.

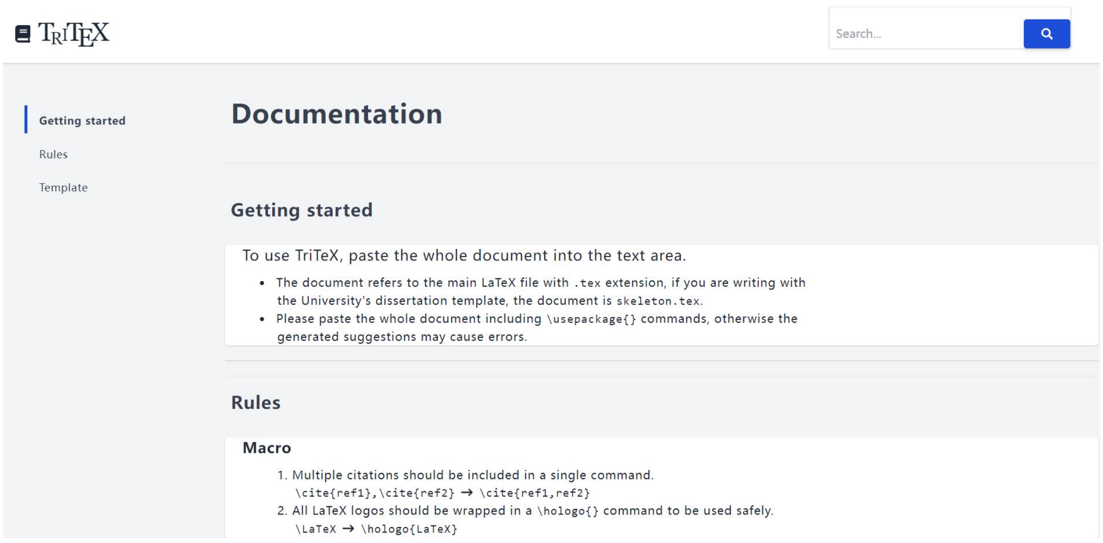
(c) Card for empty text area


No issues found

Great! Your current work doesn't have any problem on `macro`, select other filters or keep writing and come back regularly.

(d) Card for no issue text

Figure 3.3: Suggestion card designs



Documentation

Getting started

To use TriTeX, paste the whole document into the text area.

- The document refers to the main LaTeX file with `.tex` extension, if you are writing with the University's dissertation template, the document is `skeleton.tex`.
- Please paste the whole document including `\usepackage{}` commands, otherwise the generated suggestions may cause errors.

Rules

Macro

1. Multiple citations should be included in a single command.
   `\cite{ref1},\cite{ref2}` → `\cite{ref1,ref2}`
2. All LaTeX logos should be wrapped in a `\hologo{}` command to be used safely.
   `\LaTeX` → `\hologo{LaTeX}`

Figure 3.4: TRiTeX documentation

The documentation page is designed in the same colour scheme as the main page, with a search box and a scrollspy indicating the currently active content in the viewport. The documentation contains information about TRiTeX's input text, rules, and a template LaTeX document with all types of errors, as shown in Figure 3.4.

Chapter 4

Back-end checker

We use Node.js (Dahl, 2007), a runtime environment that uses JavaScript on the server to handle the back-end work of TRiTeX. Node.js contains a set of built-in modules that can be used without further installation and a package manager that easily install JavaScript packages with a single line of command. This chapter introduces the back-end implementation process of TRiTeX, including the Undo and Redo functions and the syntax of the checker.

4.1 Back-end process

To understand how we design the functions, we start by introducing how TRiTeX works. Users paste their LaTeX document in the text area and click the "Compile" button to start back-end checking. The back-end receives the raw value of the text area. As JavaScript uses a backslash(\) as an escape character, it will misunderstood LaTeX commands. We use JavaScript's `.replace()` function to escape all the backslashes in the LaTeX documents. For example, `\begin{document}` becomes `\\begin{document}`.

The back-end then detects problematic text in the escaped text by regular expression tokens. A regular expression (regex) is a string that describes a search pattern in a text. It is usually used for "find" or "find and replace" operations on strings. For example, `[bf]at` can match `bat` and `fat`, but not match `cat` or `mat`. Regexs help us create a flexible pattern to match the error in the input text.

When a problem is detected by the tokens, the back-end calls the corresponding function with the problematic text sent as a parameter. Each rule uses a different function to transform the original text into the suggested text, which is discussed in Section 4.3. We have created five empty arrays to store information for generating suggestion cards, including the problematic text, warning sentence, suggested text, type of problem, and package if available. The functions push the corresponding information for each error to the arrays. When all problematic text has run its function, the back-end generates a suggestion card for each problem.

Each suggestion card has a unique ID and a "Replace" button to replace the original text with the suggested text in the text area. The replace function finds and replaces the original text and the card is deleted. The filter works together with the suggestion cards, where it gets the index of errors for each tag from the array that stores the type of problems. The filter is used to display suggestion cards by selected type.

4.2 Undo and redo

We create undo and redo functions for TRiTeX, to allow the user freedom to complete their work. Users may accidentally resolve a suggestion they do not agree with or found the initial decision is correct. An undo and redo function can make users more confident with using TRiTeX. The browsers have built-in undo functions on the actions in the text area but they cannot deal with custom actions (replace the original text with suggestions) in TRiTeX.

We use a JavaScript stack to achieve undo-redo operations. A stack is a linear data structure, we create two arrays, one for undo and one for redo to store stacks. The stack follows the Last In First Out (LIFO) principle, that is the item inserted at last will be the first one to come out. The insert operation is called push and the delete operation is called pop, both take place from the end of the array. As mentioned in Section 4.1, TRiTeX stores the information of suggestion cards by arrays. Therefore, we set up five arrays each for the undo and redo stacks to store the information of actions. When the user replaces a problematic text with a suggestion, we push the information of the card to the undo stack. When the user undoes the action, we replace the last element in the suggested text array with the last element in the original text array and generate the suggestion card again. We then pop the undo stack and push the information to the redo stack. Redo works in a similar process, where we replace the original text with the suggestion and delete the card.

4.3 Syntax and rules

The core part of the back-end of TRiTeX is how we transform problematic LaTeX into "correct" LaTeX. Different types of problems require different solutions.

4.3.1 Patterns without unique identifiers

The easiest type of problems are those matches a pre-defined pattern and can be dealt with pure regular expression. For example, `\begin{figure}[h]` is a wrong use of the placement specifier parameter that happens commonly. `[h]` is the specifier that puts the float approximately in the same place as in the source text, but tables and figures should all go to the top or bottom of a page. Hence, we need other specifiers to replace `h`. `H` places the float exactly the same place as in the source tet, `t` refers to the top, `b` refers to the bottom, and `p` allows the float to take up a whole page itself. Another specifier `!` is special, there are many internal parameters in LaTeX that constrain the float's position and `!` can override them.

Therefore we use `[!tbp]` or `[!btp]` as the specifiers to position the float, depending on the user's preference to have it at the top or bottom of a page. There are many similar inappropriate patterns besides `\begin{figure}[h]`, for example, `\begin{table}[!h]`, `\begin{figure}[htbp]`, and `\begin{table}[tbpH]`. We have to create a flexible matching pattern that can detect all these problems.

The regular expression to find this type of problem is:

```
\\begin{(table|figure)}\s?\[[tbphH!]{0,5}[hH][tbphH!]{0,5}\]
```

We also generate a railroad diagram for the regex by Avallone (Last updated 2021), which helps us better understand the syntax of this expression, see Figure 4.1.

The syntax diagram should be read from the left to the right. The matched string starts with `\begin{`, followed by the float type, which can be `table` or `figure`, then

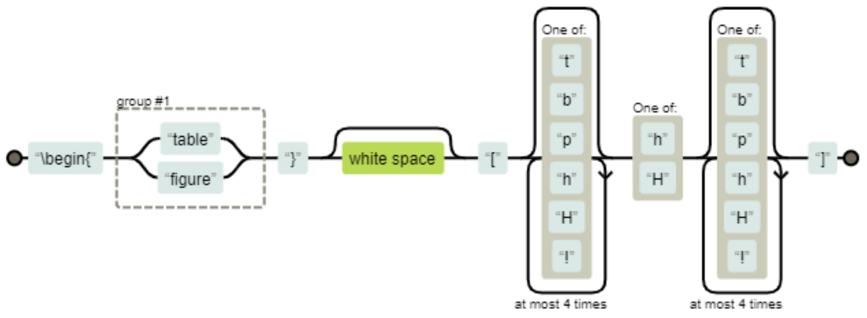
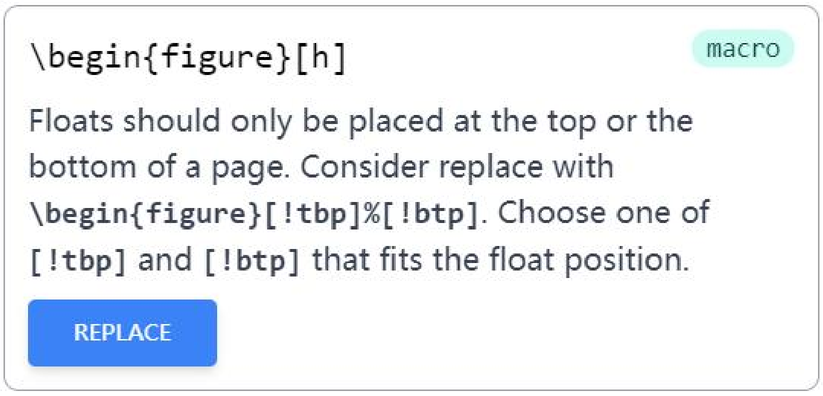


Figure 4.1: Float specifier regex syntax diagram



\begin{figure}[h]

Floats should only be placed at the top or the bottom of a page. Consider replace with **\begin{figure}[!tbp]%[!btp]**. Choose one of **[!tbp]** and **[!btp]** that fits the float position.

Figure 4.2: Card design for float position

ends the brace by `}`. A whitespace can be allowed after the braces depending on user habits. Inside the square bracket, it matches all types of combinations of the place specifiers, with `h` or `H` necessary and other specifiers are allowed for at most four times before and after the `h/H`. This can detect all combinations of placement specifiers that are used inappropriately.

When the pattern is detected, the back-end checks whether the float type is a figure or a table by a simple if-else function. The suggestion for each float type is fixed: `\begin{figure}[!tbp]%[!btp]` and `\begin{table}[!tbp]%[!btp]`. LaTeX uses % to start in-line comment. As we cannot automate the choice of top or bottom and other in-text changes may cause repositioning, we provide one correct version with its alternative provided as a comment. To clarify this to users, we added an explanation of the design in the suggestion card, as shown in Figure 4.2.

We have met a problem when testing this rule, that the original text in the text area has not been replaced when we click the replace button. After checking the back-end code, we found the problem is caused by JavaScript's `replace()` function. The function takes two input parameters, where the second replaces the first in the string. The first parameter is a regular expression and the square brackets will be read as part of the regex syntax. To avoid the problem, we added a backslash before each square bracket of the original text to escape the bracket.

4.3.2 Patterns with identifiers

A more complicated pattern contains identifiers that we need to keep from the source. For example, `\cite{ref1}\cite{ref2}` is a multiple citation which should be corrected to `\cite{ref1,ref2}`. We need to retain the information, that is the identification of

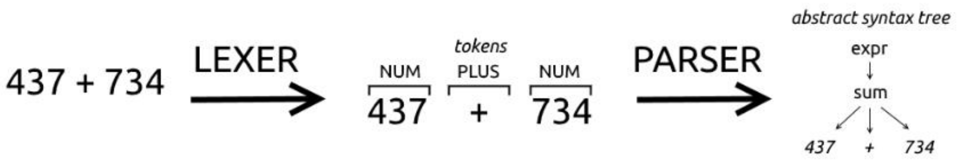


Figure 4.3: A simple arithmetic example on parsing process (Stalla, 2022)

the citation in this case, from the source. We use a parser for each pattern to achieve this purpose. A parser is always used as a component of a compiler or a translator, it contains two components that make up the parsing process. The first stage is a lexer, also known as a scanner, that breaks the string into tokens. Each token is a group of characters and the compiler understands the string by tokens as the unit. The second stage is the parser, which uses a syntax tree that combines pre-defined grammar with the tokens from the first stage. Figure 4.3 is a simple arithmetic example of the process. We can see the lexer groups the character ‘4’, ‘3’, ‘7’ into a number token ‘437’. It then uses a parse tree to represent the calculation process of the original string.

We use PEG.js (Futago-za Ryuu, 2017), a parser generator for JavaScript, to extract the identifiers from the problematic text. PEG.js generates the parser by a grammar specifying the expected input and the function’s output. The parser can be generated online and downloaded as a JavaScript file. Another method to generate the parser is to call the `peg.generate` function and pass the grammar as a parameter. We used the first one during the implementation. Although it requires a slightly larger space to store the files, it can speed up the back-end process and improve the user experience.

Another aspect we considered in designing the parser is whether the input passed to the parser should be pre-processed. Different from many other parser generators, PEG.js combines the lexer and parser all in one. This makes developing and understanding the grammar easier but do has other drawbacks. The use of a lexer can easily filter out the noise in the input, such as whitespaces and comments, forwarding only meaningful tokens to the parser. We need to pay extra effort on describing the grammar to ignore the noise if we use the original source without pre-processing. Also, parsing the whole text input has been tested to be time-consuming. Therefore, we use regular expression tokens to match the error from the original input at first and send only the problematic text to the parser.

We use an example here to better understand how the parser works. We want to solve the multiple citation problem in the input text and start with defining the regular expression token: `\\cite{[^\\\{\}]{0,}}[,~\s]{0,}){2,}`

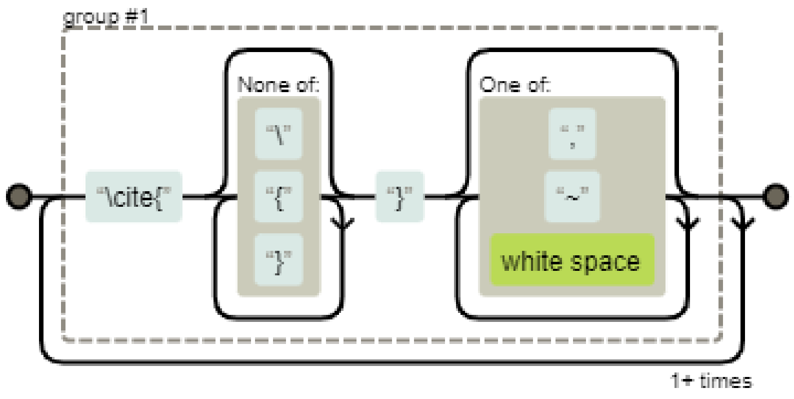
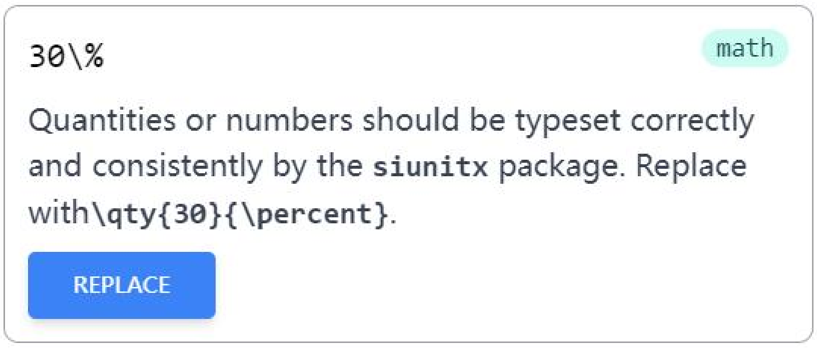


Figure 4.4: Multiple citation regular expression syntax diagram

The regex matches the `\cite{identifier}` pattern that appears more than one time continuously. We allow a flexible pattern of punctuation after each citation, for example, `\cite{A},\cite{B}` and `\cite{C}~\cite{D}` can both be detected. With the problematic text received by the parser, it parses the text and outputs the identifier. For this example, the grammar is shown below.

```
cite=token:(citetoken)*
citetoken="\\\\cite{"
          whitespace*
             id:[^\\\{\}]+
          whitespace*
             "}"    {
             return id.join('');
             }
whitespace  = [\t\n\r]
```

The grammar extracts the identifier by defining the string between `\\cite{` and `}` as an ID token. The parser returns all IDs as a string by JavaScript's `join()` function. The function joins the selected elements into a string, with a comma as the default separator. For example, parsing the input `\\cite{ref1},\\cite{ref2}` generates the output identifier `ref1,ref2`. We use double backslashes here to escape the backslash in the original command. The multiple citation function then generates the suggested text by the identifier, which is `\\cite{ref1,ref2}` in this example and pushes the information to the corresponding arrays. Figure 3.3b shows a suggestion card for a multiple citation problem.


30\%

Quantities or numbers should be typeset correctly and consistently by the **siunitx** package. Replace with**\qty{30}{\percent}**.

Figure 4.5: Card design for typesetting quantities

4.3.3 Patterns with package applied

Some wrongly used LaTeX commands are caused by users' lack of knowledge of extra packages. TRiTeX also provides suggestions that require extra packages. The patterns are matched by regex and some are parsed to extract identifiers. The only difference is the value they push to the array for packages. The replace function gets the corresponding element in the package array, if the value is 'no', it ignores the package relevant operations. Else the function detects if the required package already exists, and if not, we add `\usepackage{package_name}` by matching other packages and `\begin{document}`. This is why TRiTeX requires the user to paste the whole document instead of snippets.

Some packages require a specific loading order. The `hyperref` package should usually be the last but there are exceptions, for instance, `geometry` and `cleverref`. The `cleverref` should be loaded after `siunitx` which is used to provide a clear and consistent combination of mathematical value and unit. Without the package, users usually use `30\%` to represent 30 percent, or use `$90^\{circ}$` to represent 90 degree angle. Both expressions work but they do not typeset the quantities or numbers correctly. Here we need to load the `siunitx` package. We match `\usepackage{cleverref}` and load `siunitx` before it if exists. If not, `siunitx` is loaded before `hyperref` or on the line before `\begin{document}`. Figure 4.5 shows the suggestion card for typesetting a quantity inappropriately.

There are more than 20 syntax that each matches a flexible pattern of errors in the back-end of TRiTeX, detecting different types of problems.

Chapter 5

Evaluation

5.1 User study

A user study is a useful approach for understanding users and improving our design. Designers always assume users share the same understanding with them without realising it, this is an availability bias that needs to be eliminated by interacting with users. We have designed two user studies to evaluate TRiTeX's design and participants in both studies are university students in STEM. The first user study is designed to receive feedback from them, so we can figure out some problems to be improved. The second study produces a data-supported assessment of our work by our target users. The study is in the form of a questionnaire.

5.1.1 Questionnaire design

As the two user studies have different objectives, our original thought was to set two different questionnaires that focus on different aspects. However, this cannot evaluate whether the improvement work is successful as the questions are different. Therefore we designed a questionnaire containing both open-ended and close-ended questions that were helpful for analysing results and receiving feedback. The questionnaire is designed using these four principles:

1. **Keep the questionnaire brief with only important questions**
   This benefits both users and designers. Data collection and analysis for a long questionnaire with slightly varied questions can be time-consuming and pointless. Users may also lose interest with a huge amount of questions and answer the questions without focus.

2. **Use simple and straightforward terms instead of unnecessary jargon**
   Avoid confusions that may influence users' responses.
3. **Order the questions logically**
   Generally, the questionnaire should start with questions about user information to distinguish levels of LaTeX experience within the user base. Questions and ratings about the design in the middle and end up with open-ended questions that illicit users' opinions and experiences.
4. **Ask one question at once**
   Asking multiple questions at once can make users stressed about answering them. Also, if questions about scaling include more than one component, users will be confused if they have different answers on each component. For example if a question asks "Is the interface intuitive and aesthetic?" then a user who finds the interface aesthetic but not intuitive does not have a corresponding choice that captures both aspects of their opinion.

System Usability Scale (SUS) is also a worthwhile measure for product evaluation. SUS is a "quick and dirty" usability scale created by Brooke et al. (1996). It contains 10 questions each with five options, from Strongly disagree via Neutral to Strongly agree. The questions focus on the usability of a product and the responses are converted to a score that produces a percentile ranking. As TRiTeX should be measured from different aspects, we only take SUS as a reference when designing the questionnaire.

According to the principles and references discussed above, we designed a questionnaire that contains 10 questions (see Appendix B.3). Q1 asks the user's LaTeX level which helps us divide the users into groups. Q2–Q4 branch from Q1, showing only when the user has used LaTeX before. Those questions focus on the method they learnt LaTeX and their experience with LaTeX self-help tools. Q5–Q7 are Likert scales, focusing on the interface, features, and user experience respectively. A Likert scale is a close-ended and forced-choice scale with choices from one end to another. In Q5–Q7, six choices are provided for each question: Strongly disagree, Disagree, Neutral, Agree, Strongly agree, and Don't know / Prefer not to say. Q8 is a net promoter score (NPS) that gauges the user's likeliness of using TRiTeX in the future on a scale from 0 to 10. Q9–Q10 are open-ended questions that help identify weak spots: one asking suggestions for improvement and one asking about future features to add to TRiTeX.

5.1.2 User study analysis

The questionnaire can be divided into four sections: user base, Likert scales, Net Promoter Score, and open-ended questions. We analyse the results for the first three by different methods in this section. Feedback from open-ended questions is discussed in Section 3.3 and Section 6.2.

5.1.2.1 User base

We have collected 30 responses in total with 15 participants for each study. The mean completion time of the questionnaire is less than 5 minutes, which indicates that the length is acceptable and can keep users focused on the questions. All the participants are university students, including both undergraduates and postgraduates. For the LaTeX level of participants, we found none of the students suggests he/she is confident in his LaTeX writing. Only 3 participants never used LaTeX before and all other participants have experience but are not good at it, as shown in Table 5.1.

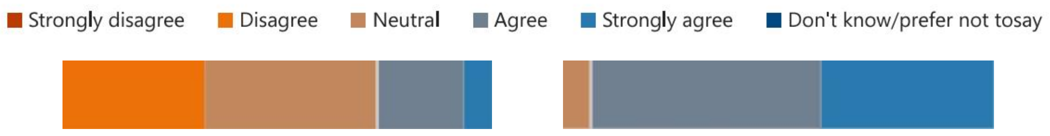
In terms of the method they learnt LaTeX, we surprisingly found none of the participants has taken relevant courses at University or School. Most participants search for guides during the writing, some read online tutorials and a few do both. This result indicates that Universities' lack of formal training on LaTeX also proves the importance of TRiTeX as the sources on the internet are unreliable and can be obsolete. Another unexpected result is that none of the participants has used LaTeX self-help tools before so we failed to collect responses on Q4.

5.1.2.2 Likert scales

The Likert scales contain three topics, interface, feature, and user experience. We focus on the indicators that received negative responses and discuss our improvement on them. 73 % responses to the argument "I know how to use TRiTeX without reading

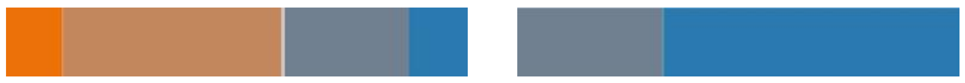
LaTeX level	**Beginner**	**Novice**	**Intermediate**	**Expert**
First study	2	8	5	0
Second study	1	8	6	0
Total	**3**	**16**	**11**	**0**

Table 5.1: LaTeX level of participants



(a) First study (b) Second study

Figure 5.1: Response distribution on “I know how to use TRiTeX without reading documentation.”



(a) First study (b) Second study

Figure 5.2: Response distribution on “The information on suggestion cards is clear and helpful.These figures have duplicate labels with Figure 5.1

.

documentation.” are Disagree and Neutral. This shows the instruction on the interface is unclear to users, hence we amended the placeholder text and added an instruction to the default suggestion card. Details of the improvement are discussed in Sections 3.3.2 and 3.3.3. The final version has received mostly positive feedback and only one response is Neutral, as shown in Figure 5.1.

The argument “The information on suggestion cards is clear and helpful.” also receives more than half neutral or below responses. We believe it is because the original design only presents the original text and the suggested replacement text, without any explanation that helps users to understand the suggestion. To change the situation, we have redesigned the layout of the suggestion cards and added an explanation of the problem to the card. Details are discussed in Section 3.3.3 and improvement of the card design is shown in Figure 3.3. The improved version receives satisfying feedback on the argument, with most participants choosing Strongly Agree. Figure 5.2 shows the response distribution for this argument.

We also compare the results of the studies by converting the options into scores, with Strongly disagree - 1 point, Disagree - 2 points, Neutral - 3 points, Agree - 4 points, and Strongly agree - 5 points. As none of the responses chose Don’t know/Prefer not to say, we will ignore this option in the analysis. The total score is 35 points for the interface, 15 points for the features, and 20 points for the user experience. The overall score for the Likert scale is 70 points.

Category	**Min**	**Max**	**Avg**	**Med**
Interface	65.7	91.4	83.2	82.9
Feature	53.3	86.7	74.6	73.3
UX	70.0	95.0	83.7	85.0
Overall	**70.0**	**85.7**	**81.5**	**81.4**

(a) First user study

Category	**Min**	**Max**	**Avg**	**Med**
Interface	85.7	100.0	95.6	94.3
Feature	80.0	100.0	90.2	86.7
UX	85.0	100.0	95.0	95.0
Overall	**85.7**	**98.6**	**94.3**	**94.3**

(b) Second user study

Table 5.2: Likert scale statistics (%)

	μ_1	s_1	μ_2	s_2	n	**t-score**	**p-value**
Interface	29.13	4.05	33.47	0.92	15	−4.047	0.0011
Feature	11.23	1.21	13.53	0.42	15	−6.955	<0.0001
UX	16.73	1.42	19.00	0.63	15	−5.659	<0.0001
Overall	**57.06**	**6.03**	**66.00**	**2.12**	**15**	**−5.417**	**<0.0001**

Table 5.3: Welch's test result

Table 5.2 shows the statistics of both studies, including minimum, maximum, average, and median on each topic and the overall score. As each topic has a different total score, we convert each result to a percentage. For example, the overall average score is 57.06 out of 70 for the first study, the value is $\frac{57.06}{70} = 81.5\ \%$ in the table.

We can see the results have improved substantially from the first study to the second. The study on the final version has received exclusively positive feedback, with each topic receiving maximum score responses. Indeed the maximum overall score for the first study is equal to the minimum overall score for the second study.

To strengthen our interpretation of the results we apply a Welch's t-test to each topic's result and the overall score. Welch's t-test is a statistical method used to determine if two individual populations have equal means. Different from the commonly used Student's t-test, Welch's t-test does not assume the populations share the same variance. In order to perform statistical significance tests, we need the following hypotheses:

Hypothesis H_0 (null hypothesis)**:** the means between the scores are equal (there is no difference)

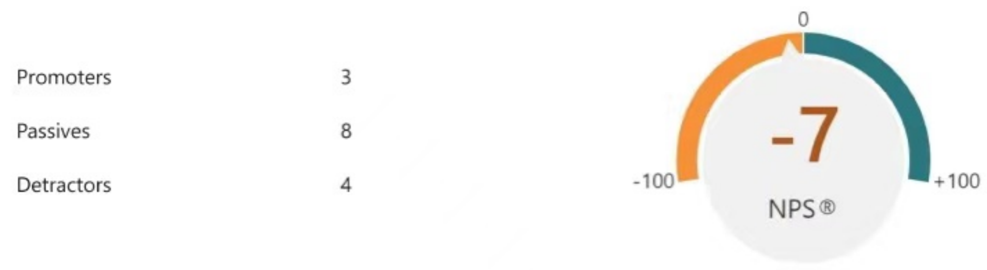
Hypothesis H_1 (alternative hypothesis)**:** the means between the scores are not equal (there is a difference)

The Welch's t-test takes the mean, standard deviation, and sample size for each group to calculate the t-score. The t-score can be converted to the p-value, which is the probability of observing the null hypothesis. Table 5.3 shows the result of Welch's t-test, where μ, s, n refers to the mean, standard deviation, and sample size respectively and the subscript indicates which user study.

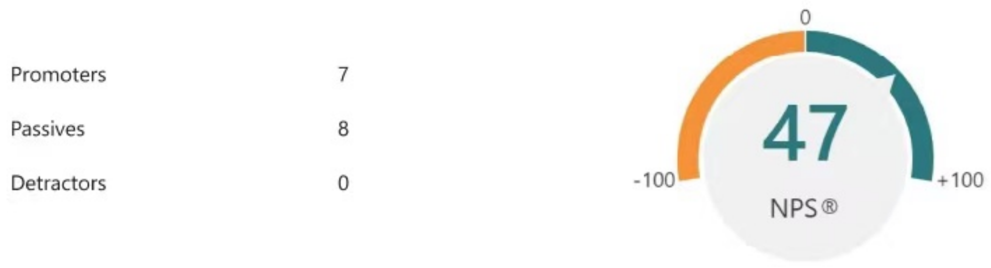
The p-values for all topics and the overall score indicate that the null hypothesis is rejected at the significance level $\alpha = 0.01$. There is a difference between the mean of the scores, meaning our improvement is probably meaningful for 99 % of users.

5.1.2.3 Net Promoter Score

Q8 asks how likely the user will keep using TRiTeX in the future, on a scale of 0 to 10. This is a Net Promoter Score (NPS) which is a measure of the satisfaction and enthusiasm of users for a product (Reichheld, 2003). Users who respond 0–6 are detractors unhappy with TRiTeX, 7 or 8 are passives who are satisfied with it, and 9 or 10 are promoters enthusiastic about it. Note that we only treat the responses by category, not by specific numbers. For example, 0 and 6 each count as an equal detractor in our analysis. Although the results of NPS are often overstated in business circles (Keiningham et al., 2007), its use here to indicate general satisfaction is acceptable.



(a) NPS score of the first study



(b) NPS score of the second study

Figure 5.3: NPS score comparison

The NPS score is calculated by %promoters-%detractors. It can range from −100 (all detractors) to 100 (all promoters): the closer to 100, the more favourable the result. An NPS score greater than 0 indicates the product has more promoters than detractors. As the NPS benchmark can be completely different across industries, there is no standard for a “good” NPS score, so we evaluate by comparing the results.

Figure 5.3 shows the NPS score of TRiTeX in each study. We can see both studies have 8 passives. There are 3 promoters and 4 detractors in the first study, forming an NPS score of −7. The second study has surprisingly received a score of 47 with no detractors and all 7 non-passive responses are promoters. This result suggests that the user satisfaction of TRiTeX has increased noticeably from our prototype to the final version. Even if we analyse the final version alone, we still find the NPS result is highly positive. All users are satisfied with TRiTeX and nearly half of them are enthusiastic about it. They are likely to use TRiTeX in their future LaTeX writing.

Chapter 6

Conclusions

6.1 Summary

In this project, we have designed a browser-based LaTeX self-help tool to assist beginners in writing proper LaTeX. We designed a prototype first for TRiTeX and created a feedback loop through user studies to improve the design. We can conclude our work by answering the research questions we discussed in Section 1.3.

RQ1: TRiTeX solves five types of problems: commands used inappropriately, use of obsolete commands and packages, lack of knowledge on LaTeX's features, package loaded in wrong order, and ugly layout. Each type contains pre-defined templates to match and replace the problems.

RQ2: TRiTeX uses HTML and CSS to create the layout of the web page and the back-end is written in JavaScript with node modules applied. We use JetBrains' WebStorm IDE in respect of its smart and powerful features. Tailwind CSS is a fast and customizable framework and we use it to utilize the front-end design.

RQ3: The interface provides user settings on font size, line height, and dark or light mode. These ensure users' personal preference and also make TRiTeX accessible for more people. The functions can be executed by both button and keyboard shortcuts to ensure user experience of both novices and experts.

RQ4: We capture the problems by pre-defined regular expression patterns. The patterns are flexible and the back-end transforms them into correct text. For patterns with identifiers, we use a parser generated by PEG.js to extract the value of the identifier.

RQ5: We took two user studies in the form of questionnaire to evaluate our design, one for the prototype and one for the final version, and analyse the result by different methods to see the effect of improvement and the overall feedback.

RQ6: Comparing to other LaTeX self-help tools we discussed in Chapter 2, we can see TRiTeX has its strength and drawbacks as well. TRiTeX is easy to install and configure and it provides up-to-date suggestions, which is friendly to novices. The installation guide on `chktex` is hard for users who still need a self-help tool and `nag` provides only out-dated LaTeX 2ε suggestion. However, both tools have more configuration options, which means the environment is more complex and customizable.

6.2 Future work

We have received some advice on new features from the open-ended questions in the user study but have not got time to implement them. A replace-all function should be added to the suggestion area. Users may be tired of clicking on the replace buttons one by one if they agree with all the suggestions. This is not hard to achieve in technology and I would like to take this task as the next step of TRiTeX. Other suggested features are not that prior, including more colour schemes and more text fonts.

There are two known limitations of the current version of TRiTeXto be improved. First, the back-end can not understand which part of the document is an in-line comment. We have paid much effort to make the checker ignore the comments. However, we cannot delete all comments in the text area as there can be notes useful for users, the replace function still detects text in in-line comments and replace them. While this problem is not "fatal" and it only influences patterns with extra packages, we make an explanation of it in the documentation.

Second, we have tried to replace all types of labels entered manually, for example, "Section 4" and "Table 1.1" should be replaced with Chapter 4 and Table 1.1. These references should use the `cleverref` package and `\cref{}` command to generate labels automatically. However, as the environment for floats (tables and figures) is complicated and many users keep some floats labelled and some not, it is extremely hard to write a replacement syntax without a risk.

6.3 Reflections

Through this project, I have learned a lot of knowledge about web design which is a completely new field for me. I wasted lots of time on the front-end design at first learning and testing new items. Fortunately, I have finally designed an interface I am satisfied with. The UI is both aesthetic and minimized. While there have been shortages in the design. As I did not understand the relationship between CSS and HTML, many markups are messy with some styles defined in the CSS and some defined in-line. There are also redundant codes that can be defined as a single class in the CSS. Those experiences and lessons can help me with future web design projects a lot.

JavaScript is also a new programming language for me, which is challenging for me. Its syntax and expressions are completely different from other programming languages I have used before. The definitions and concepts, for example, DOM, is abstract and have been hard to understand for me. Therefore I took the time to learn the language. The escape characters have crushed the project once as I did not find a way to keep the backslashes in LaTeX commands from the original input. After trying different functions to keep the raw value, I finally found the solution is to escape the backslashes in the input text before doing any operation on it.

Although there are many difficulties encountered, the software we selected is proved to be reliable. Tailwind CSS is a light-weighted and customizable framework which can be my first choice in web development CSS tools. WebStorm has many strong built-in features and useful plugins that help me a lot in programming and designing. I have learnt a lot from the development of TRiTeX in both front-end and back-end implementation. Regular expression and syntax trees are also strong language processing tools which can be useful in future work.

References

Avallone, Jeff. (Last updated 2021). regexper-static.

Awareness, Colour Blind. (2022). Colour Blindness. *Colour Blind Awareness*. https://www.colourblindawareness.org/colour-blindness/

Babich, Nick. (2019). The 12 Do's and Don'ts of Web Design | Adobe XD Ideas. *Ideas*. https://xd.adobe.com/ideas/principles/web-design/12-dos-donts-web-design-2/

Brooke, John et al. (1996). SUS - A quick and dirty usability scale. *Usability evaluation in industry*, *189*(194), 4–7.

Colblindor. (2016). Coblis — Color Blindness Simulator – Colblindor. *Color-blindness.com*. https://www.color-blindness.com/coblis-color-blindness-simulator/

Dahl, Ryan. (2007). Node.js. https://nodejs.org/en

Futago-za Ryuu, David Majda. (2017). PEG.js: Parser Generator for JavaScript.

Keiningham, Timothy L, Cooil, Bruce, Andreassen, Tor Wallin, & Aksoy, Lerzan. (2007). A Longitudinal Examination of Net Promoter and Firm Revenue Growth. *Journal of Marketing*, *71*(3), 39–51. https://doi.org/10.1509/jmkg.71.3.39

Memon, Masooma. (2021). The 21 Main UX Laws Every Designer Must Follow + Examples. *Maze*. https://maze.co/collections/ux-ui-design/ux-laws/

Microsoft. (2016). Visual Studio Code. *Visualstudio.com*. https://code.visualstudio.com/

Nielsen, Jakob. (1994). 10 Heuristics for User Interface Design. *Nielsen Norman Group*. https://www.nngroup.com/articles/ten-usability-heuristics/

Otto, Mark. (2000). Bootstrap. *Getbootstrap.com*. https://getbootstrap.com/

Reichheld, Frederick F. (2003). The one number you need to grow [PMID: 14712543]. *Harvard Business Review*, *81*(12), 46–54.

Schwarz, Ulrich M. (2011). CTAN: Package nag - Detecting and warning about obsolete LATEX commands. *www.ctan.org*. Retrieved August 13, 2022, from https://www.ctan.org/pkg/nag

Stalla, Alessio. (2022). A Peggy.js Tutorial. https://tomassetti.me/a-peggy-js-tutorial/

Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. (2017). *tailwindcss.com*. https://tailwindcss.com/

Thielemann, Jens T. Berger. (2016). CTAN: Package chktex - Check for errors in LATEX documents. *www.ctan.org*. https://www.ctan.org/pkg/chktex

Trettin, Mark, & Ensenbach, Marc. (2016). CTAN: Package l2tabu - Obsolete packages and commands. *www.ctan.org*. Retrieved August 13, 2022, from https://www.ctan.org/pkg/l2tabu

WebAIM. (2019). WebAIM: Contrast Checker. *Webaim.org*. https://webaim.org/resources/contrastchecker/

WebStorm: The Smartest JavaScript IDE by JetBrains. (n.d.). *JetBrains*. https://www.jetbrains.com/webstorm/

Yablonski, Join. (2020). *Laws Of UX*. O'Reilly. ISBN: 9781492055310. Retrieved July 31, 2022, from https://lawsofux.com/en/

Appendix A

Requirements engineering

TRiTeX **MUST**:

1. be easy and intuitive to use
2. be easy to install
3. be easy to configure
4. have useful configurability
5. be easy to extend functionality
6. provide some genuine benefit
7. correct at least some typical rookie errors
8. run reliably with a consistent GUI on at least two commonly used main-stream browsers

TRiTeX **MUST NOT**:

1. mislead users into a false sense of security or smugness about their LaTeX
2. provide suggestions that are out-dated or wrong, or which directly lead to ugly layouts

TRiTeX **SHOULD**:

1. support users with simple accessibility requirements
2. meet or exceed accessibility requirements or guidelines
3. adhere to good HCI design principles
4. follow good software engineering and programming practices
5. be easy to maintain
6. allow personal preferences for user control
7. use suggestions gleaned from reliable sources
8. prefer LaTeX3 code over LaTeX 2_ε
9. be rapid to use for documents requiring many suggestions

TRiTeX **SHOULD NOT**:

1. unduly confuse typical Edinburgh Informatics students
2. exclude users with simple accessibility needs
3. be limited to one browser or operating system

Appendix B

User Study

B.1 Participants' information sheet

Participant Information Sheet

Project title:	TriTeX – A Browser-Based LaTeX Assistant for Beginners
Principal investigator:	Brian Mitchell
Researcher collecting data:	Trista (Yiying) Yang

This study was certified according to the Informatics Research Ethics Process, RT number 46726. Please take time to read the following information carefully. You should keep this page for your records.

Who are the researchers?

Brian Mitchell, Trista Yang.

What is the purpose of the study?

Our aim is to create a tool that actively helps beginners, so we want to hear our target users' feedback. The study takes users' comments as feedback to improve the editor and also the study result is used to evaluate our work.

Why have I been asked to take part?

You are a beginner to LaTeX who is the target user of TriTeX.

Do I have to take part?

No – participation in this study is entirely up to you. You can withdraw from the study at any time, up until we have finished data analysis and conclusion without giving a reason. After this point, personal data will be deleted and anonymised data will be combined such that it is impossible to remove individual information from the analysis. Your rights will not be affected. If you wish to withdraw, contact the PI. We will keep copies of your original consent, and of your withdrawal request.

What will happen if I decide to take part?

During the study, you will answer a questionnaire with both close-ended and open-ended questions regarding the interface and feature design of TriTeX.

Are there any risks associated with taking part?

There are no significant risks associated with participation.

Are there any benefits associated with taking part?

No.

What will happen to the results of this study?

The results of this study may be summarised in published articles, reports and presentations. Quotes or key findings will be anonymized: We will remove any information that could, in our assessment, allow anyone to identify you. With your consent, information can also be used for future research. Your data may be archived for a maximum of 4 years. All potentially identifiable data will be deleted within this timeframe if it has not already been deleted as part of anonymization.

Data protection and confidentiality.

Your data will be processed in accordance with Data Protection Law. All information collected about you will be kept strictly confidential. Your data will be referred to by a unique participant number rather than by name. Your data will only be viewed by the researcher Trista Yang and principla investigator Brian Mitchell.

All electronic data will be stored on a password-protected encrypted computer, on the School of Informatics' secure file servers, or on the University's secure encrypted cloud storage services (DataShare, ownCloud, or Sharepoint) and all paper records will be stored in a locked filing cabinet in the PI's office. Your consent information will be kept separately from your responses in order to minimise risk.

What are my data protection rights?

The University of Edinburgh is a Data Controller for the information you provide. You have the right to access information held about you. Your right of access can be exercised in accordance Data Protection Law. You also have other rights including rights of correction, erasure and objection. For more details, including the right to lodge a complaint with the Information Commissioner's Office, please visit www.ico.org.uk. Questions, comments and requests about your personal data can also be sent to the University Data Protection Officer at dpo@ed.ac.uk.

Who can I contact?

If you have any further questions about the study, please contact the lead researcher, Trista Yang, s1810787@ed.ac.uk.

If you wish to make a complaint about the study, please contact inf-ethics@inf.ed.ac.uk. When you contact us, please provide the study title and detail the nature of your complaint.

Updated information.

If the research project changes in any way, an updated Participant Information Sheet will be made available on http://web.inf.ed.ac.uk/infweb/research/study-updates.

Alternative formats.

To request this document in an alternative format, such as large print or on coloured paper, please contact Trista Yang, s1810787@ed.ac.uk.

General information.

For general information about how we use your data, go to: edin.ac/privacy-research

THE UNIVERSITY *of* EDINBURGH
informatics

B.2 Participants' consent form

Participant number:______________________

Participant Consent Form

Project title:	TriTeX – A Browser-Based LaTeX Assistant for Beginners
Principal investigator (PI):	Brian Mitchell
Researcher:	Trista (Yiying) Yang
PI contact details:	brian.x.mitchell@ed.ac.uk

By participating in the study you agree that:

- I have read and understood the Participant Information Sheet for the above study, that I have had the opportunity to ask questions, and that any questions I had were answered to my satisfaction.
- My participation is voluntary, and that I can withdraw at any time without giving a reason. Withdrawing will not affect any of my rights.
- I consent to my anonymised data being used in academic publications and presentations.
- I understand that my anonymised data will be stored for the duration outlined in the Participant Information Sheet.

Please tick yes or no for each of these statements.

1.	I allow my data to be used in future ethically approved research.	**Yes**	**No**
2.	I agree to take part in this study.	**Yes**	**No**

Name of person giving consent	Date dd/mm/yy	Signature
Name of person taking consent	Date dd/mm/yy	Signature

THE UNIVERSITY *of* EDINBURGH
informatics

B.3 Questionnaire

TriTeX: A Browser-Based LaTeX Assistant for Beginners

This study was supervised by Dr Brian Mitchell, certified according to the Informatics Research Ethics Process from the University of Edinburgh, RT number 46726. There are no significant risks associated with participation. Participation in this study is entirely up to you. You can withdraw from the study at any time without giving a reason. After this point, personal data will be deleted, and anonymised data will be combined such that it is impossible to remove individual information from the analysis. Your rights will not be affected. If you wish to withdraw, contact the PI. We will keep copies of your original consent and of your withdrawal request. Please note any data anonymised or aggregated with other data and results of analysis prior to any request for withdrawal will be maintained as the data anonymisation and processing procedure cannot be reversed. Your data will be processed in accordance with Data Protection Law. All data collected about you will be kept strictly confidential. Your data will be referred to by a unique participant number rather than by name. Your data will only be viewed by the research team [Brian Mitchell: brian.x.mitchell@ed.ac.uk, Trista Yang: s1810787@ed.ac.uk].

All electronic data will be stored on a password-protected encrypted computer, on the School of Informatics' secure file servers, or on the University's secure encrypted cloud storage services (DataShare, ownCloud, or Sharepoint), and all paper records will be stored in a locked filing cabinet in the PI's office. Your consent information will be kept separately from your responses in order to minimise risk. Questionnaire will be guided by a set of questions and was set up to take less than 5 mins. The results of this study may be summarised in published articles, reports and presentations. Quotes or key findings will be anonymised: We will remove any information that could, in our assessment, allow anyone to identify you. With your consent, information can also be used for future research. Your data may be archived for a minimum of 5 years to allow documentation as part of a longitudinal study and feed lessons learnt in each run into subsequent sessions.

The University of Edinburgh is a Data Controller for the information you provide. You have the right to access information held about you. Your right of access can be exercised in accordance with Data Protection Law. You also have other rights, including rights of correction, erasure and objection. For more details, including the right to lodge a complaint with the Information Commissioner's Office, please visit www.ico.org.uk. Questions, comments and requests about your personal data can also be sent to the University Data Protection Officer atdpo@ed.ac.uk.

* Required

1. What is your LaTeX's level? *

   - ○ Completely new to it
   - ○ Beginner/Used but not good at
   - ○ Intermediate/Have relatively sound knowledge of basics
   - ○ Expert/Confident with LaTeX writing
   - ○ Prefer not to say

2. How did you learn LaTeX? *

   - ☐ Educated in school/university
   - ☐ Learn from online tutorials
   - ☐ Search for guide during writing
   - ☐ Other

3. Have you used LaTeX self-help tools or packages before? *

   - ○ Yes
   - ○ No

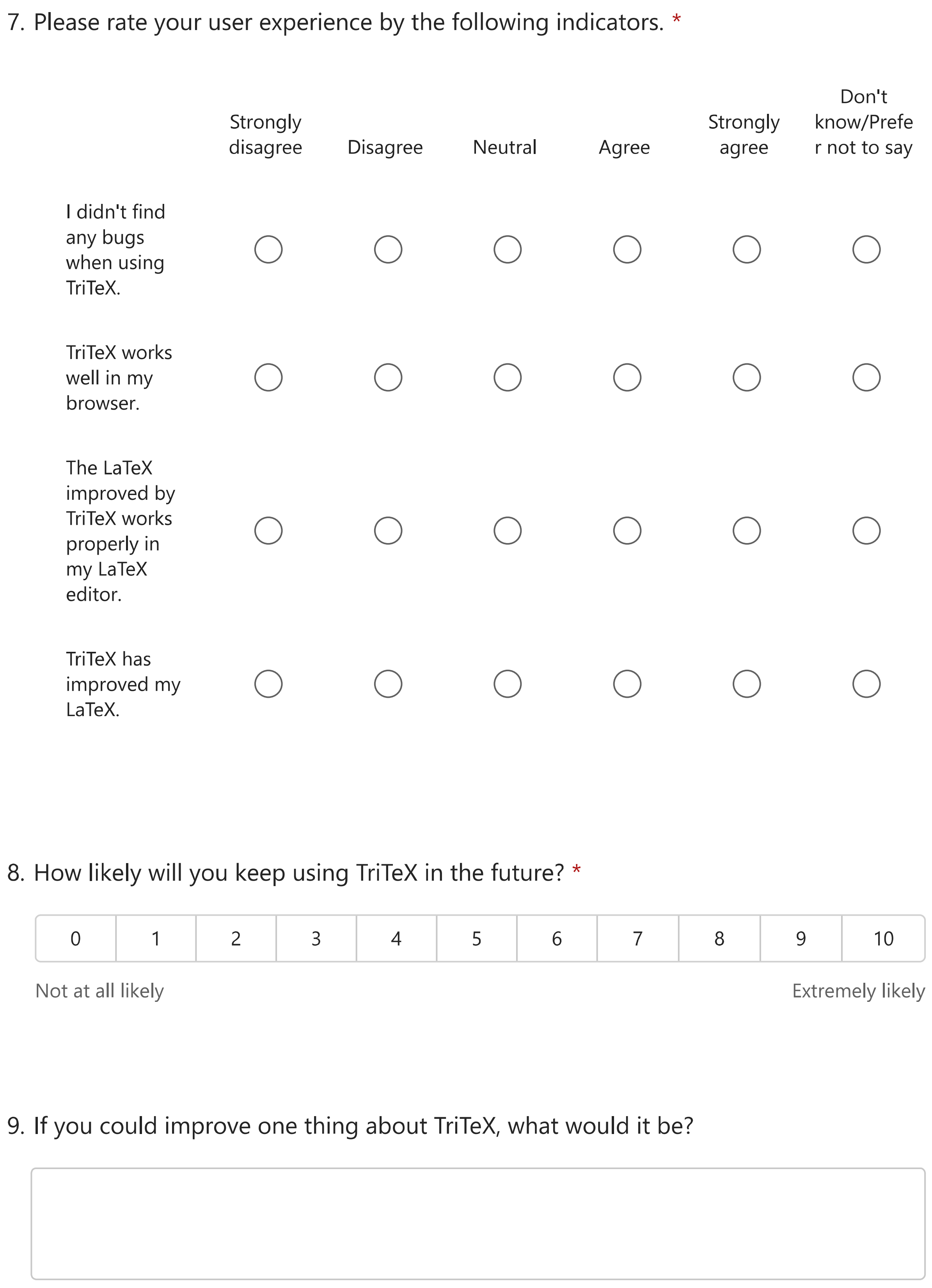
4. Which one did you use? Was there any feature impressive to you?

5. Please rate the interface of TriTeX by the following indicators. *

	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree	Don't know/prefe r not tosay
The use of words in the interface is easy to understand.	○	○	○	○	○	○
The use of icons in the interface is easy to understand.	○	○	○	○	○	○
I know how to use TriTeX without reading documentati on.	○	○	○	○	○	○
The interface design is intuitive.	○	○	○	○	○	○
The interface design is aesthetic.	○	○	○	○	○	○
The font colour is accessible to me.	○	○	○	○	○	○
The font size is accessible to me.	○	○	○	○	○	○

6. Please rate the features of TriTeX by the following indicators. *

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	Don't know/Prefer not to say
The features in the navigation bar are useful to me.	○	○	○	○	○	○
The information on suggestion cards is clear and helpful.	○	○	○	○	○	○
The settings options satisfy my needs.	○	○	○	○	○	○



7. Please rate your user experience by the following indicators. *

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	Don't know/Prefer not to say
I didn't find any bugs when using TriTeX.	○	○	○	○	○	○
TriTeX works well in my browser.	○	○	○	○	○	○
The LaTeX improved by TriTeX works properly in my LaTeX editor.	○	○	○	○	○	○
TriTeX has improved my LaTeX.	○	○	○	○	○	○

8. How likely will you keep using TriTeX in the future? *

0	1	2	3	4	5	6	7	8	9	10

Not at all likely — Extremely likely

9. If you could improve one thing about TriTeX, what would it be?

10. Is there any feature you think TriTeX should add?

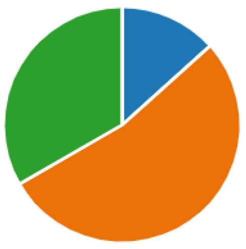
B.4 Results of first user study

TriTeX: A Browser-Based LaTeX Assistant for Beginners (First User Study)

15	04:33	Active
Responses	Average time to complete	Status

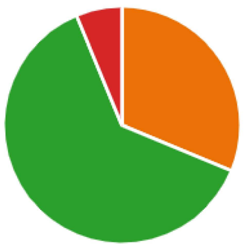
1. What is your LaTeX's level?

●	Completely new to it	2
●	Beginner/Used but not good at	8
●	Intermediate/Have relatively so...	5
●	Expert/Confident with LaTeX wri...	0
●	Prefer not to say	0



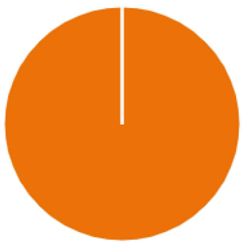
2. How did you learn LaTeX?

●	Educated in school/university	0
●	Learn from online tutorials	5
●	Search for guide during writing	10
●	Other	1



3. Have you used LaTeX self-help tools or packages before?

●	Yes	0
●	No	13



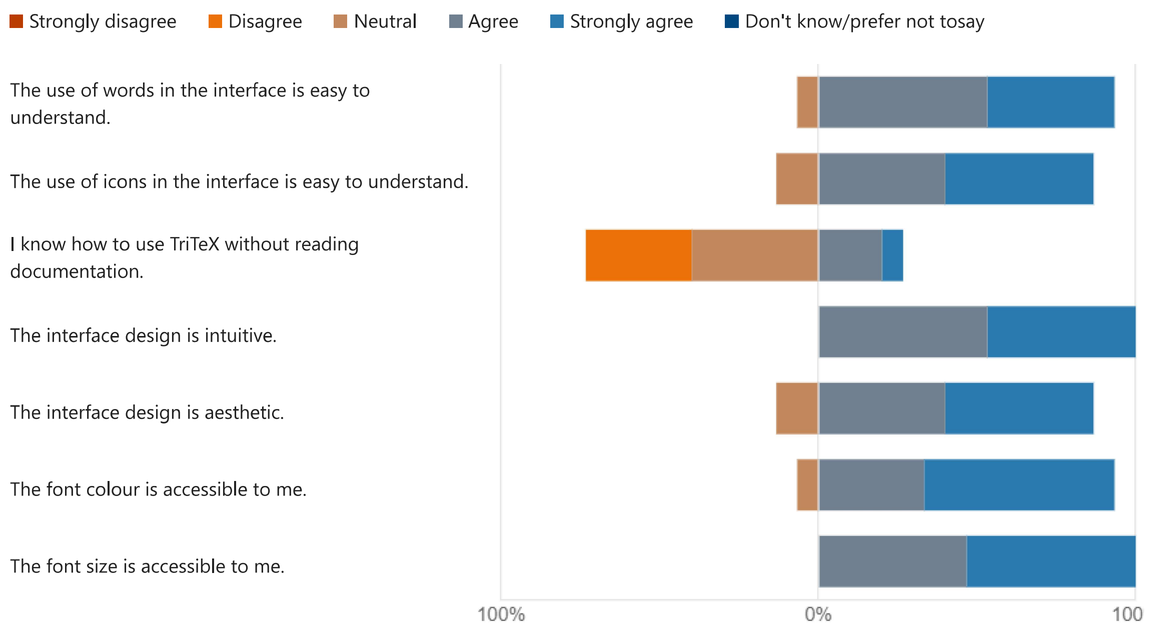
4. Which one did you use? Was there any feature impressive to you?

0

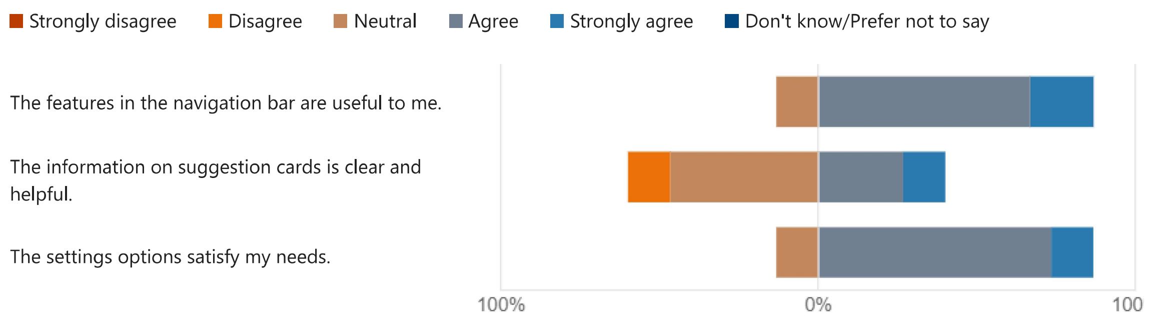
Responses

Latest Responses

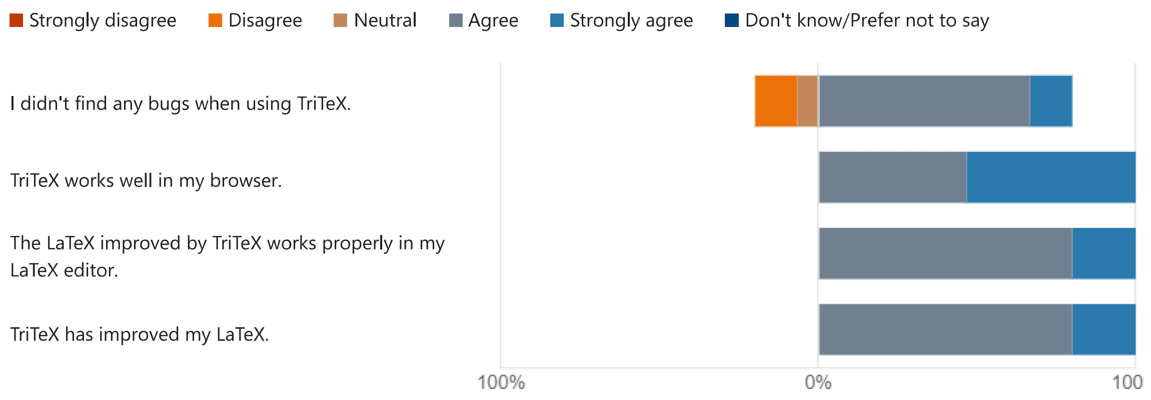
5. Please rate the interface of TriTeX by the following indicators.



6. Please rate the features of TriTeX by the following indicators.

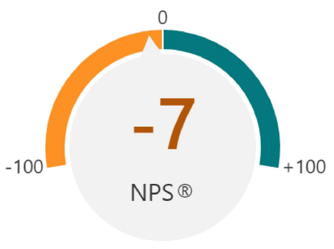


7. Please rate your user experience by the following indicators.



8. How likely will you keep using TriTeX in the future?

Promoters	3
Passives	8
Detractors	4



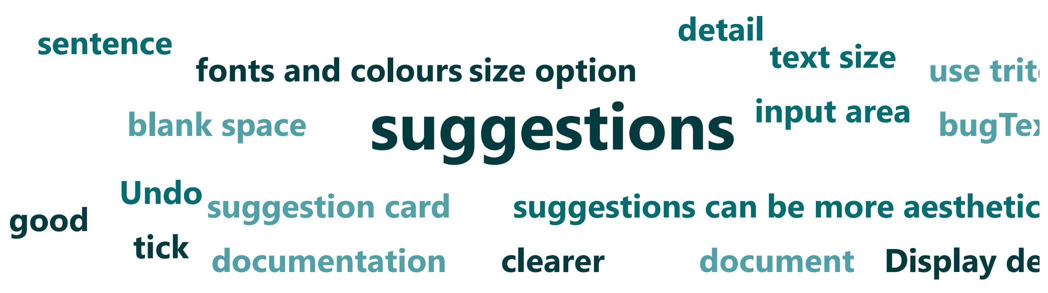
9. If you could improve one thing about TriTeX, what would it be?

9

Responses

Latest Responses

2 respondents (**22**%) answered **suggestions** for this question.



10. Is there any feature you think TriTeX should add?

4

Responses

Latest Responses

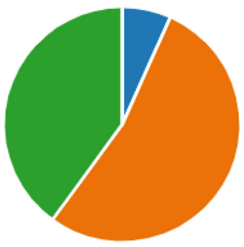
B.5 Results of the second user study

TriTeX: A Browser-Based LaTeX Assistant for Beginners (Second User Study)

15	04:27	Active
Responses	Average time to complete	Status

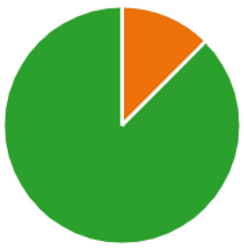
1. What is your LaTeX's level?

●	Completely new to it	1
●	Beginner/Used but not good at	8
●	Intermediate/Have relatively so...	6
●	Expert/Confident with LaTeX wri...	0
●	Prefer not to say	0



2. How did you learn LaTeX?

●	Educated in school/university	0
●	Learn from online tutorials	2
●	Search for guide during writing	14
●	Other	0



3. Have you used LaTeX self-help tools or packages before?

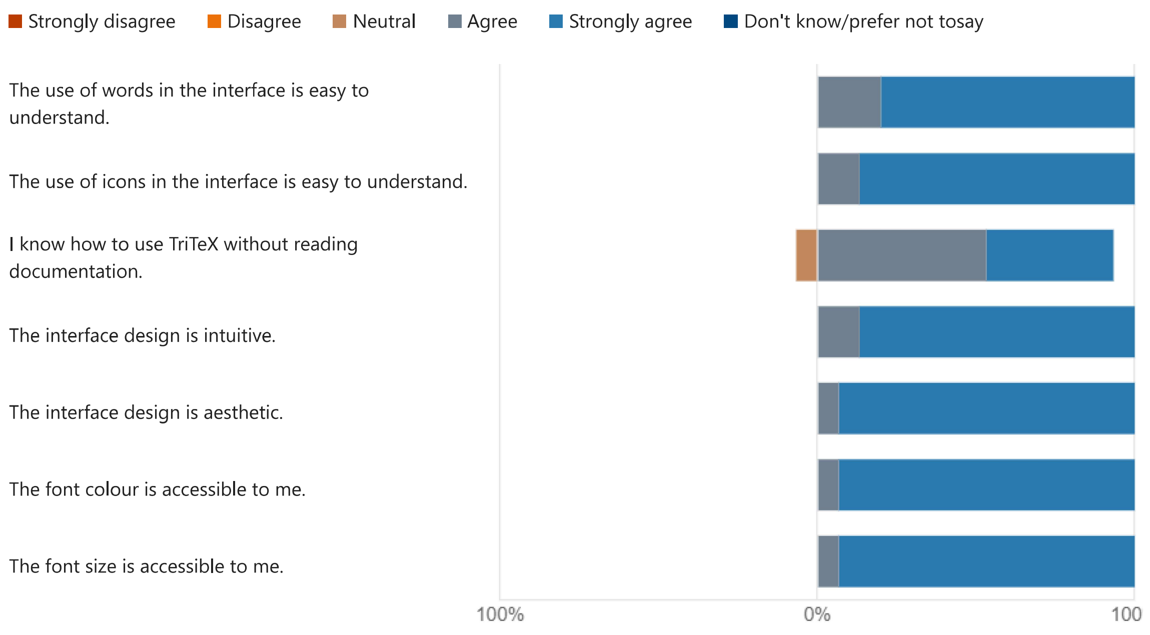
●	Yes	0
●	No	14

4. Which one did you use? Was there any feature impressive to you?

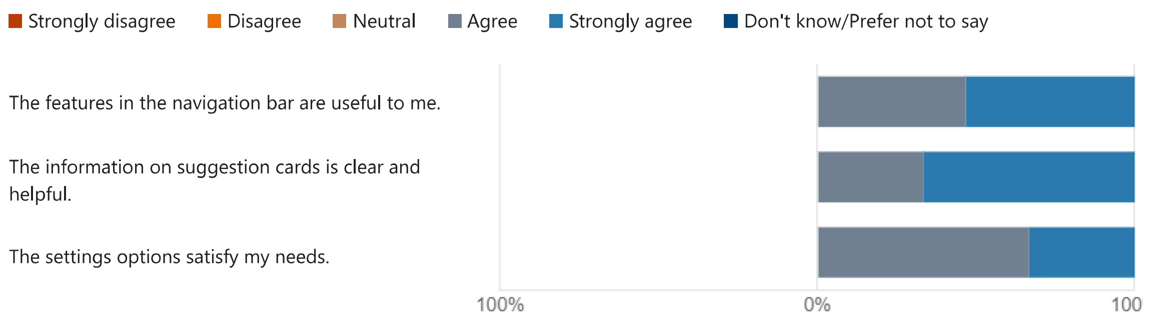
0
Responses

Latest Responses

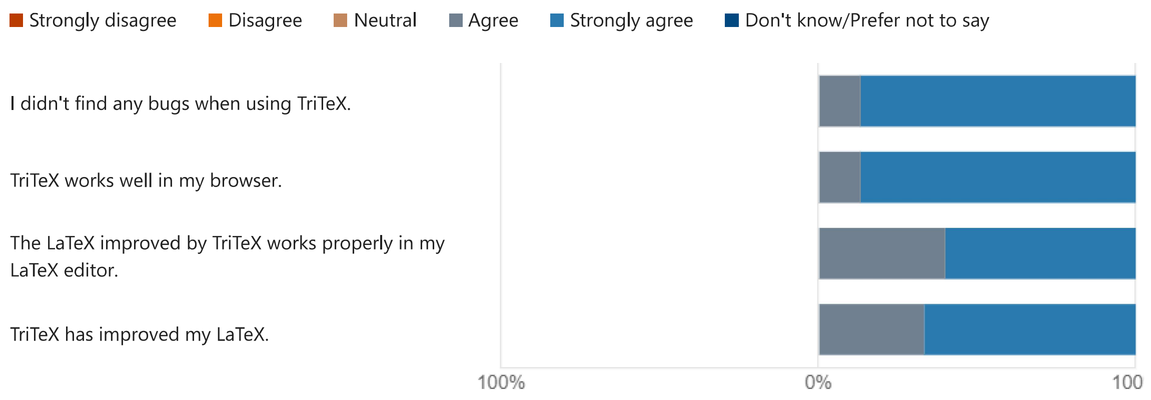
5. Please rate the interface of TriTeX by the following indicators.



6. Please rate the features of TriTeX by the following indicators.

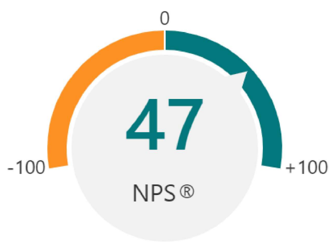


7. Please rate your user experience by the following indicators.



8. How likely will you keep using TriTeX in the future?

Promoters	7
Passives	8
Detractors	0



9. If you could improve one thing about TriTeX, what would it be?

3
Responses

Latest Responses
"The text input should be used with colors like in a latex editor."
"Highlight the original text in textbox."

10. Is there any feature you think TriTeX should add?

4
Responses

Latest Responses
"Add more settings to the system."