Agile Software Projects - Final Deliverable

‘MATHOTE’

Authors: Team member 1, Team member 2, Team member 3, Team member 4, Team member 5

Contents

TABLE OF FIGURES

1. BACKGROUND

1.1 Abstract

The purpose of this project is to create an innovative solution in the realm of educational technology - a Maths Assisting Web Application - *Mathote*. *Mathote* is a tool designed to bridge the gap between complex mathematical problems and intuitive understanding. It is a side helper tool designed to assist a wide range of users, from students and educators to future professionals in engineering and science. *Mathote* enhances productivity, accuracy, and understanding in mathematical work for anyone who interacts with maths in their daily life.

1.2 Introduction

The aim for this deliverable is to successfully develop a fully functional web application. All team members will be acting as full stack developers and playing a pivotal role in the development of both the front-end and back-end components of *Mathote*. Each member will bring their expertise and collaborative spirit to the project and develop respective components for front- and back-end iteratively, ensuring that the application's design and functionality are seamlessly integrated. Throughout the development of the project, this team will be referring back to the initial concept report which outlines the foundational ideas and prototype models.

1.3 Literature Review

Paper Reference	Summary	Strengths	Limitations
[1]	a case study about the influence of using mobiles phones as part of their mathematics program	caters students who were recently recovering from significant health issues	caters 3-4 students so results cannot be generalised to a broader context

[2]	Reports findings comparing and contrasting the work practices and software use of practising researchers in mathematics and engineering who share the goal of developing and defending new mathematical formulations.	class of software considered is the Computer Algebra Systems which enable users to rapidly solve, simplify, and otherwise manipulate a wide range of symbolic expressions and tools dedicated to matrix manipulations	work practices and attitudes of the participants are not representative of all expert mathematicians
[3]	investigates the extent to which student and task-related characteristics are associated with different types of note-taking and analyses how task success depends on these elements	sample of n=866 students (age: mean=13.99) are considered	only two distinct tasks could be considered. Despite parallelly constructed tasks, no consistent effect of note-taking could be demonstrated in this study
[4]	examines the note-taking habits of students in an introductory calculus course and identifies factors affecting their note-taking	identifies factors affecting note-taking in introductory calculus courses	pace of lectures, complexity of material

[5]	explores how technology is transforming mathematics education and the challenges associated with digital note-taking	focus is on three lines of research: teaching mathematics with technology, learning mathematics with technology, and assessment with technology.	issue demonstrates that old but still relevant themes require further theoretical and methodological investigation.

Over the past 15 years, research has shown that technology can significantly enhance mathematics education and improve student engagement. However, current solutions such as Computer Algebra Systems have limitations, including lack of transparency, limited support for collaboration, and transcription problems.

The need for better integration of digital tools and user-friendly interfaces is also highlighted. Based on these findings, our project aims to develop a more effective and user-friendly note-taking application that addresses these challenges and supports students in their mathematical learning journey.

1.4 Outcomes

This stage expects:

1) to give the user a full-fledged experience of mathematical note taking and aims to develop an all rounded mathematical application that integrates live mathematical calculation and note taking.
2) to provide access to the web application through a range of devices that allow for internet connectivity.
3) that the user can register themselves on to the application, login with their existing account, create, modify and delete their notes.
4) to provide the users access to the mathematical features such as a Scientific Calculator, a Graphical Calculator and a Step by Step Equations Solver for solving advanced mathematical problems and/or taking notes.

1.5 Methodology

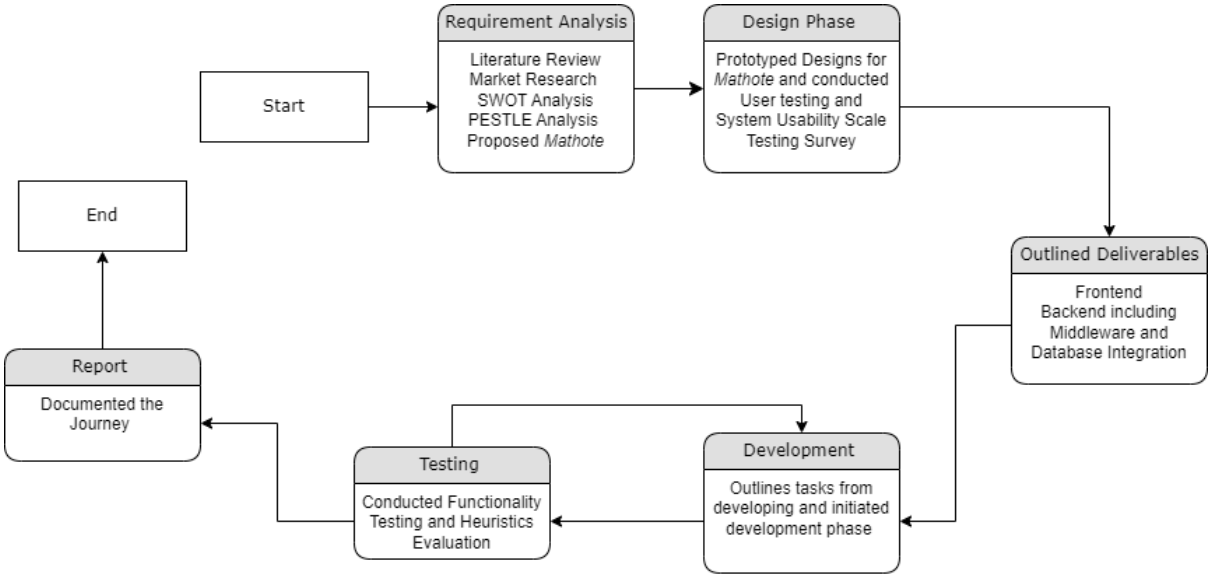


Figure 1: Methodology flowchart

As seen in Figure 1, the project commenced with a thorough analysis of what we wanted to achieve and what features were most critical. We decided on key app features like user authentication, note management, and ensuring a responsive design across various devices. We also researched and decided on the app's visual aesthetics. To gain a comprehensive understanding of user needs, we looked at comparable apps and asked prospective users for their opinions. This helped us pinpoint the essential functionalities and ideal design components, which helped us to establish precise and targeted development objectives for the project.

Following the requirements gathering phase, we entered the design phase where we prototyped a few designs and conducted a few tests - Design Testing and SUS Evaluation. The tests helped us improve our designs and strengthened our deliverables for the development - frontend and backend including middleware and database.

We made sure to test everything to confirm that all the features worked properly across different browsers - Blackbox testing. This included checking user registration, the login process, and the note management system to make sure everything ran smoothly - Whitebox testing. We also tested how the design looked on different devices to ensure it was responsive. For the purpose of testing we created survey forms and improved the app appearance and functionalities based on the feedback received. We also conducted heuristic evaluation. There were several iterations involved in each testing. Whenever we ran into bugs or issues during testing, we worked on fixing them to improve the overall experience.

2. PLANNING AND RESEARCH

2.1 Team Breakdown

Our team worked together on the front-end and back-end one after the other rather than having the team split up into two and working on different stacks. This is because all members possess different skills in the web development domain and can assist each other out when the other team member needs assistance.

2.2 Resource and Time Allocation

We divided our entire technical journey into four sprints. Sprint 1 for developing the front-end, Sprint 2 for developing the back-end, Sprint 3 for comprehensive testing, and Sprint 4 for getting ready for final submission. Each of the sprints was led by a different team member:

Sprint	Sprint Lead	Proposed Duration
Frontend	Team Member 1	1 week
Backend	Team Member 2	2 weeks
Testing	Team Member 3	2 weeks
Getting ready for submission	Team Member 4	2 weeks

Figure 2: Summarized Proposed Sprint Journey

As seen in Figure 3, Sprint 1 was allocated a duration of one week, while Sprint 2 was scheduled to be completed over a course of two weeks. Sprint 3 was planned to span two weeks to ensure thorough testing and validation of *Mathote's* functionalities. Lastly, Sprint 4 was allotted the final two weeks to finalise the report and the application, bringing everything together for submission.

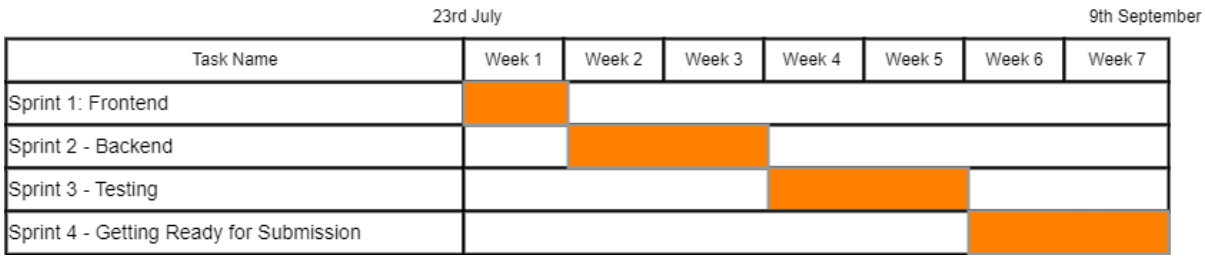


Figure 3: Proposed Gantt Chart

2.3 Sprint 1 - Front-end

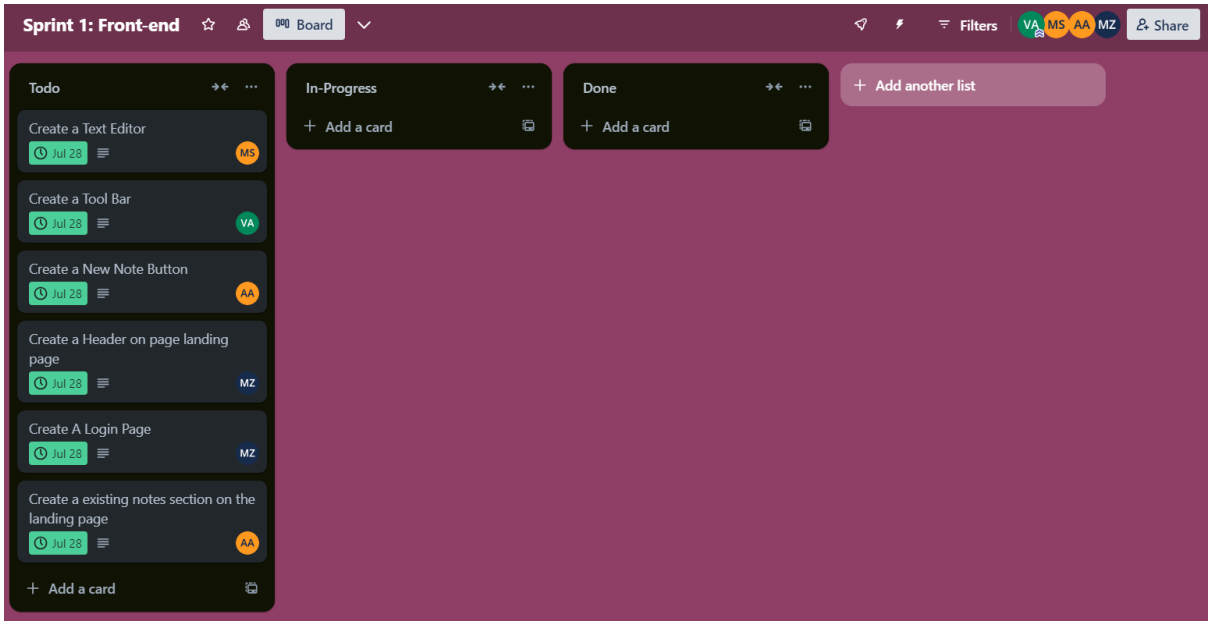


Figure 4: Trello Board for Sprint 1

The first week of initiating the technical phase was allocated to developing the front-end of *Mathote*. We conducted our first meeting and made a task list, as seen in Figure 4, allocating each of the tasks to the members and also jotting the expected time -Figure 5- it might take to complete that very task. Our aim was to develop a front-end that abides by the designs we created initially in the design phase (refer to the design phase). We deliberately chose to utilise technologies with which we were already well-acquainted, such as HTML and CSS/Bootstrap, to ensure efficiency and reliability in the development process, further allowing us to leverage our existing expertise and expedite the project timeline. While we considered modern libraries and frameworks such as React, we prioritised familiarity and stability to maintain a smooth workflow and reduce the learning curve, thus enabling the team to focus on delivering a polished product within the allocated time frame.

Tasks	Expected Time Duration	Actual Time Taken	Complete	Incomplete	Members
designing front-end	1 day	1 day	Front end Look	-	All members
login page	1 day	1 day	html and css for login page	-	Member 2
homepage: creating header	1 day	1 day	html and css for header of homepage	-	Member 2
homepage: creating a functional add notes button	1 day	1 day	add notes button that when clicked on creates a new note	-	Member 1
note writing page: creating a toolbar for each note webpage	1 day	1 day	note writing editing page with a toolbar. the toolbar consists of buttons	-	Member 4
creating a text editing area	1 day	1 day	text box where the user is able to jot down notes	-	Member 3
creating an existing notes section on the home page	1 day	-	-	Need to establish database first; existing notes section will have all have existing notes currently in the database	Member 1

Figure 5: Progression log

Designing the front-end was a collaborative effort that was executed during a meeting. All of the team members aligned on the application's design, ensuring consistency with the prototypes outlined in our proposal. As seen in Figure 6, we successfully delivered the front-end tasks within the planned timeline of one week. This efficiency was due to our familiarity with the tasks and the skills required to complete them. Nothing new had to be studied or researched related to completing the tasks as they were well within our existing knowledge base. All that was needed was basic HTML/CSS knowledge that we already developed in the prior modules. However, the existing notes section was moved forward to be integrated in Sprint 2 as it called for establishing the database connection first. It is pertinent to mention that we worked one after another, in iterations for two main reasons: simply because a few development features could not be proceeded with unless the key prerequisite elements were developed. And secondly, to avoid merging conflicts when pushing on GitHub.

All of the other tasks were completed relatively quickly, with the exception of the Toolbar, which required a relatively deeper understanding of JavaScript. Despite this, we were able to integrate the necessary functionality, keeping up with the project timeline.

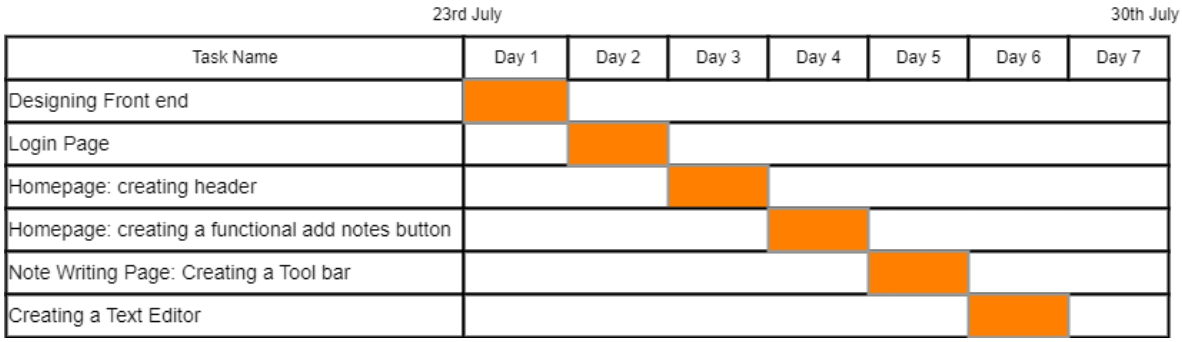


Figure 6: Gantt Chart Sprint 1 - Reality

2.4 Sprint 2 - Backend

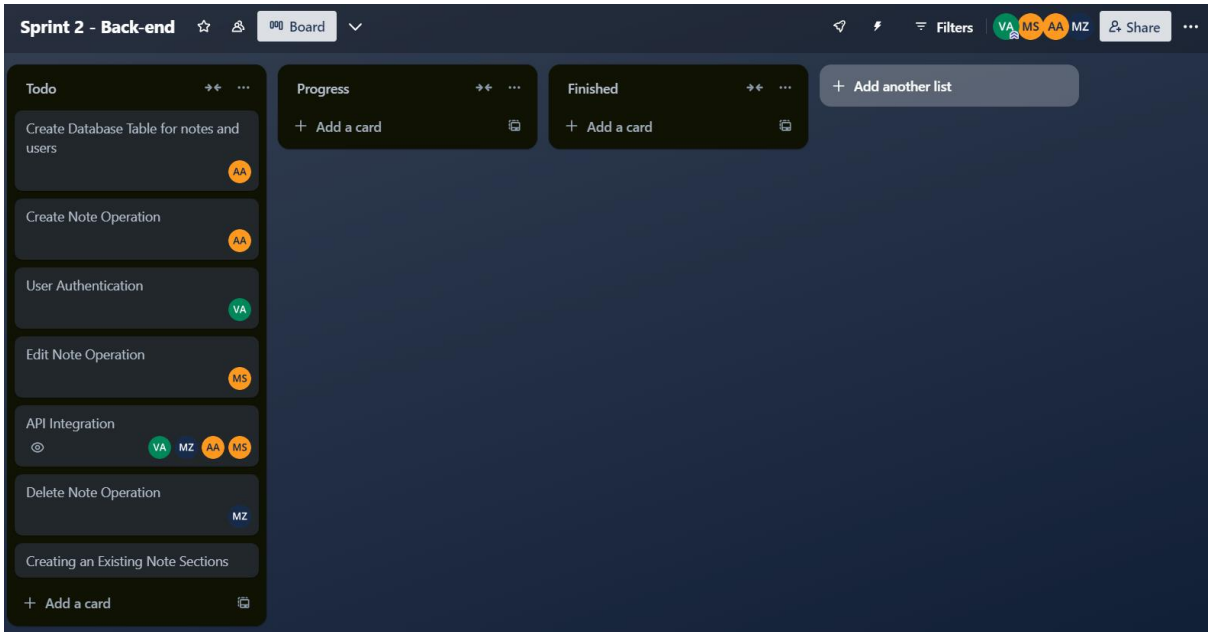


Figure 7: Trello Board for Sprint 2

The second and third week of the technical phase were allocated to developing the backend. We conducted our second meeting and made a task list -Figure 7- allocating each of the tasks to the members and also jotting the expected time it might take to complete that very task - Figure 8. We also discussed the backend technologies that we would be using for development. The discussion led us to choose Node Js, Express, EJS templating and Sqlite3 database for our backend development. We decided to use the template that was provided to us in our CM2040 module as the basis for our application development as we were using the same concepts of creating, deleting and editing a note and storing it in the database.

Tasks	Expected Time Duration	Actual Time Taken	Complete	Incomplete	Members
Discussing backend design - tools, technologies, database	1 day	1 day	Designing Backend	-	All
creating database table for users	1 day	1 day	Creating a Table	-	Member 4
creating database table for notes	1 day	1 day	Creating a Table	-	Member 1
database operation = Create Note	2 days	3 days	Creating a new note	-	Member 1
database operation = Edit Note	1 day	1 day	Editing a Note	-	Member 3
database operation = Delete Note	1 day	1 day	Deleting a Note	-	Member 2
User authentication - login	3 days	7 days	User login, session handling	-	Member 4
Graphical Calculator Integration API	1 day	1 day	Integrating a graphical calculator on the app	-	Member 4
Step By Step Equation Solver API	2 days	4 days	Integrating a step by step equation solver	-	Member 2
Scientific Calculator API	1 day	1 day	Integrating a live scientific calculator	-	All

Figure 8: Progression Log

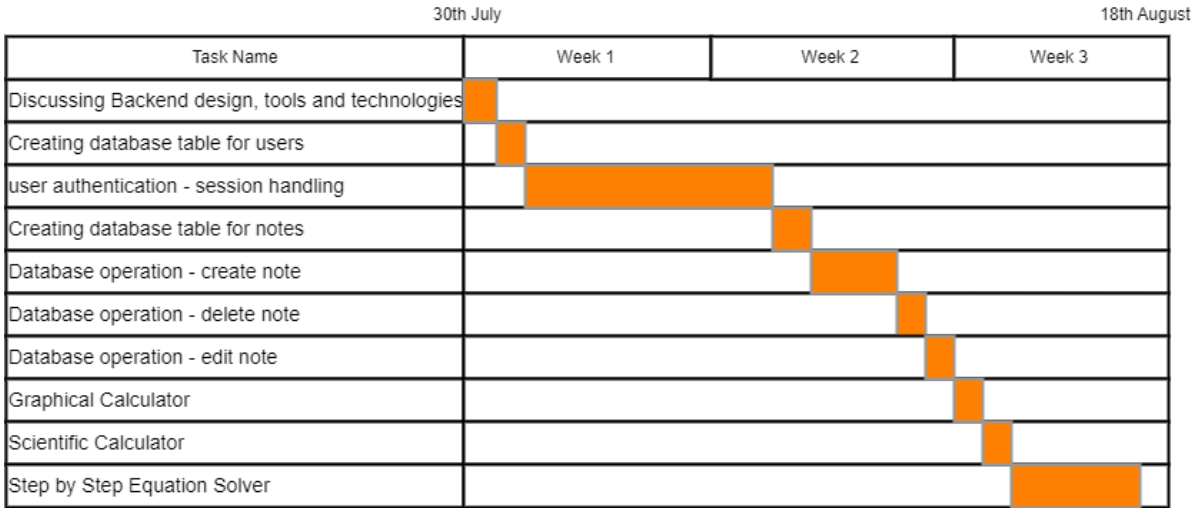


Figure 9: Gantt Chart Sprint 2 - Reality

The Gantt chart above shows a very different story to what we had planned for in the progression log. To summarise, user authentication that was planned to be completed in 3 days, took significantly longer due to complications and errors with integrating the session handling feature. Additionally, since user authentication was a completely new feature for us, it required a lot of time to perform the necessary research, trial and error and debugging. This was a little concerning as we were now, a week behind our anticipated project timeline. However, we successfully navigated it and came to a consensus to put in extra effort, and wind up testing in a week's time so that testing sprint is not compromised; we have ample time to thoroughly review our application and report.

2.5 Sprint 3 - Testing

After our unexpected delay with Sprint 2, we revised the timeline for Sprint 3 (Figure 10) to span a week rather than two and were successful. In that week, we created two surveys, one for Heuristic Evaluation and one for User Testing, and circulated it. The Heuristic Evaluation survey was filled by the members of the team whereas the User Testing survey was filled by people who integrate maths in their daily life. This is further discussed in 3.2 User Testing.

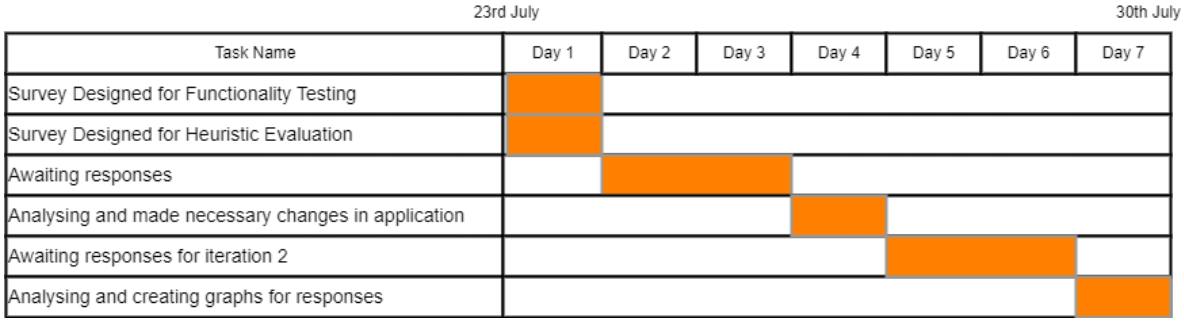


Figure 10: Gantt Chart Sprint 3 - Reality

Because we had to wind up testing in a single week - as a result of taking an extra week in Sprint 2 - things had to be crammed together and rushed through. For that we created a functionality testing survey and heuristic evaluation survey on the same day - the first day of

Sprint 3 and distributed the former to our family members and the public. As for the heuristic evaluation, the team itself filled the form without being biased. Two days were enough to receive a handful of responses. We analysed those responses and worked to improve our application based on those and redistributed the survey for iteration 2.

2.6 Sprint 4 - Getting Ready for Submission

Although the report was being written side by side with the practical work, formatting, structuring and giving it a flow still needed to be done along with research work. Moreover, evaluation for the whole journey, what worked, what did not, the limitations of our application all needed to be studied and noted in the report. These were reserved for the last Sprint of our Journey aimed at finalising our report and project for submission.

2.7 Project Management Tools

2.7.1 Trello Boards

We made use of Trello Boards to assign tasks and deadlines to the members. For better organization, we created separate Trello Boards for each of the sprints: Sprint 1 displayed tasks for frontend and Sprint 2 displayed tasks for back-end development. Sprint 3 and Sprint 4 as such did not have their trello boards. In team meetings, we discussed and finalized tasks for each sprint and added them to the respective Trello Board. Then tasks were assigned to specific team members along with corresponding deadlines. To ensure transparency and accountability throughout the project, the Trello Boards were shared with all team members, allowing them to view their assigned tasks and update their progress in real time.

2.7.2 Git/GitHub

We made use of Git/GitHub to accommodate version controlling and collaboration. We had a member create a repository on their GitHub and then add the rest of members as contributors to the repository. Each of the other members subsequently created copies of the repository on their machines locally and performed their tasks. As soon as a member was done with their task, they would push the changes on to GitHub and inform the rest of the members of the update. The other members would then update the repository on their machines locally.

2.7.3 Google Meet

Google Meet was used to conduct meetings with all the members. It was where the entirety of meetings from requirements to the testing phase were conducted. Before a meeting, one of the members would create a meeting link and share it with all group members on Slack so they could join it. Important matters, research updates, designing the application were among the various things that were discussed in Google Meet. Meetings lasted at most 1 hour with occasional 30 mins extension.

2.7.4 Slack

Slack was our main tool for communication and updates. We used it to share resource links, schedule virtual meetings, and set agendas. It helped us discuss project details, assign roles, and work together effectively. By keeping all our discussions and updates in one place, Slack made sure everyone was informed and was on the same page.

3. DESIGN AND DEVELOPMENT

3.1 Design Testing

We had previously conducted a System Usability Score Survey in our proposal which aided us in finalising the design approach that we wanted to move forth with for our application. After several iterations and the feedback gathered during the survey, the design we ultimately implemented achieved a SUS score of 65.25. The following are casual and formal wireframes of that very design:

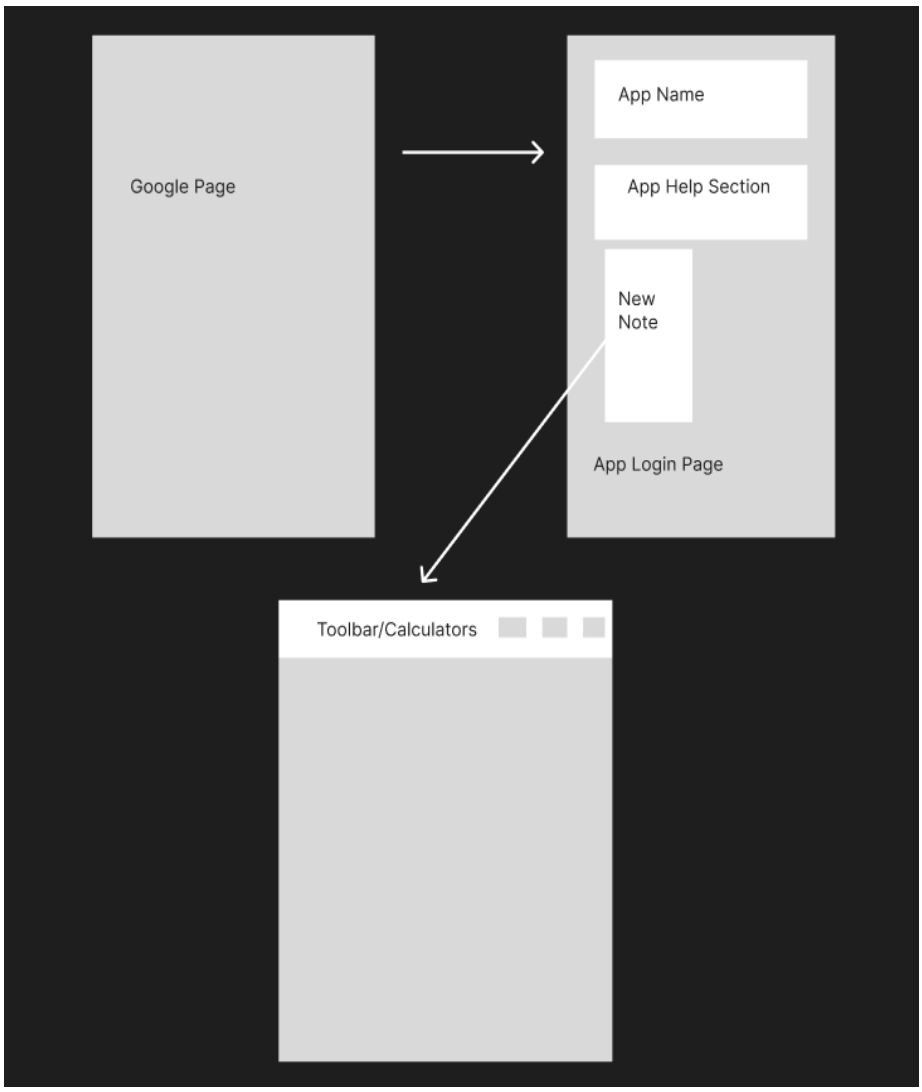


Figure 11: Casual Wireframe Design

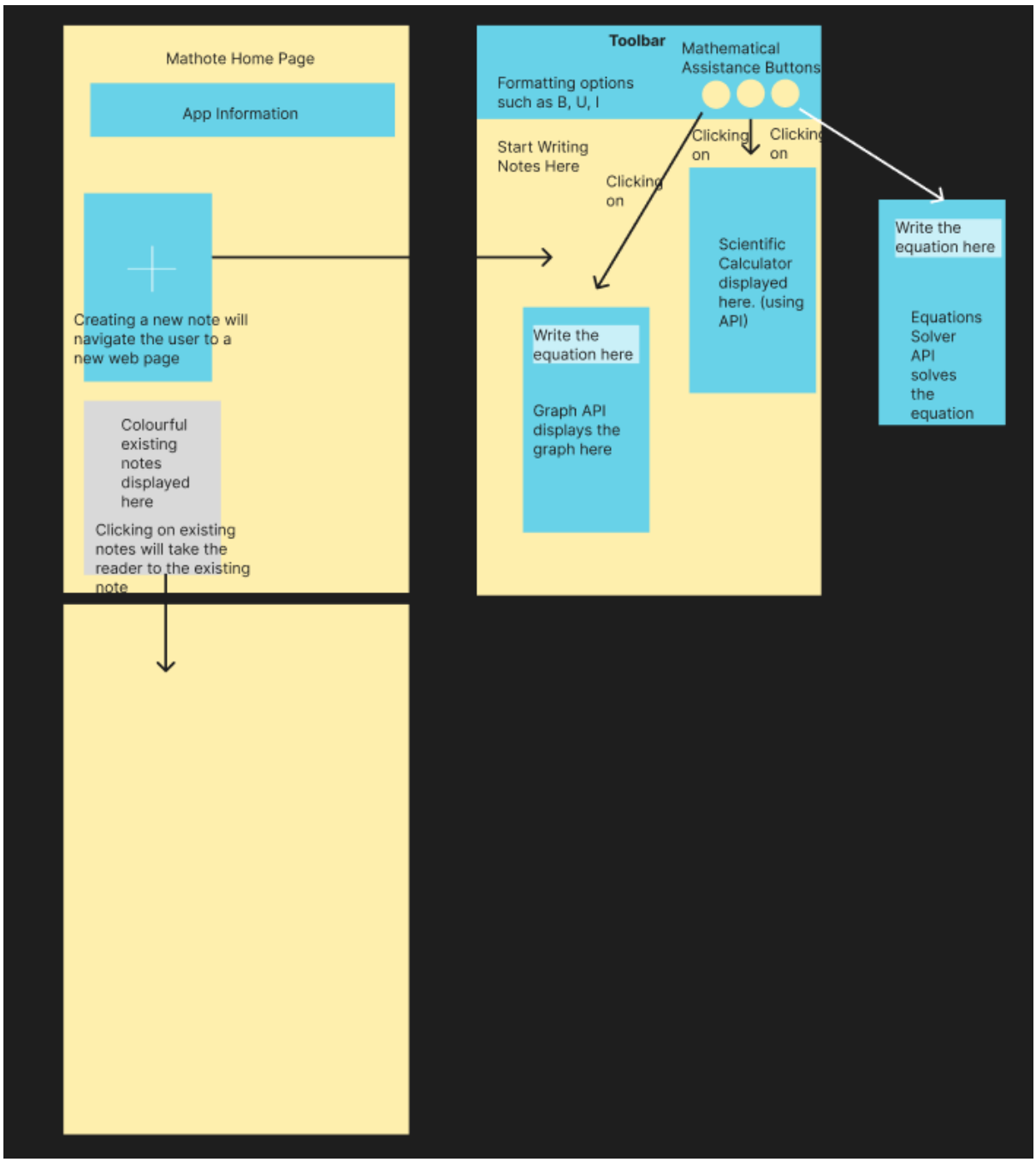


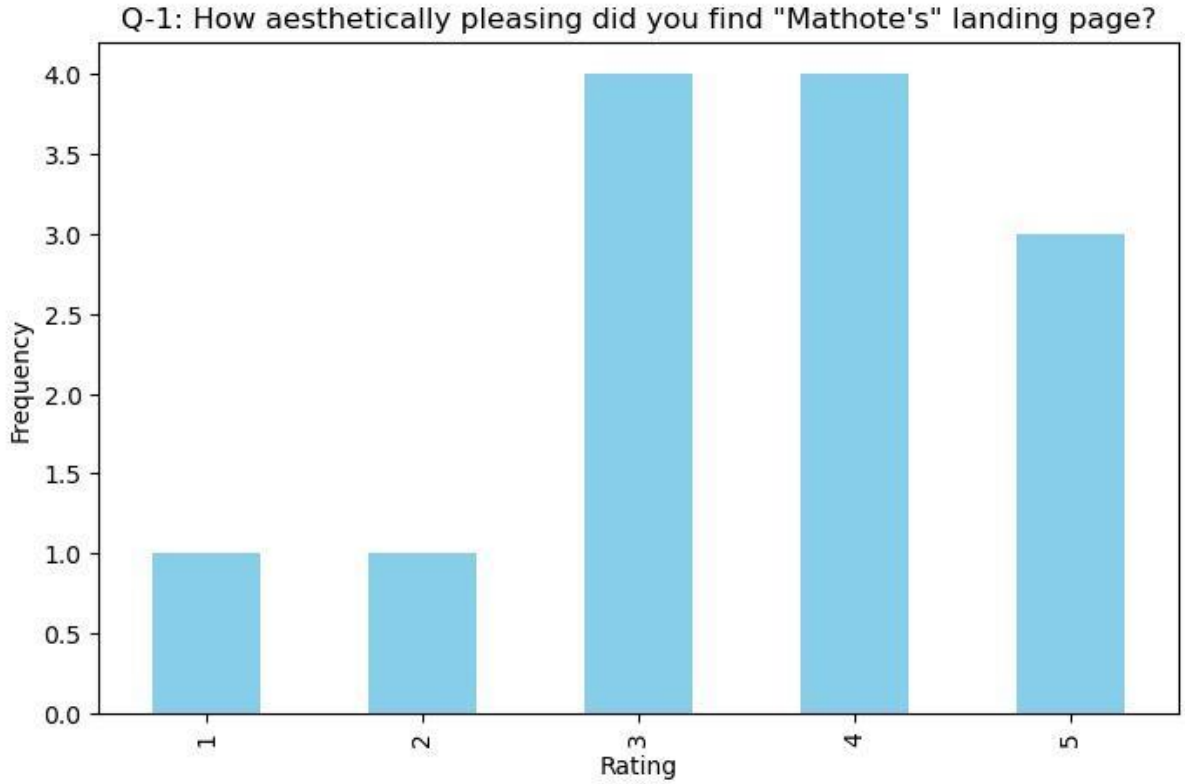
Figure 12: Proposed Design

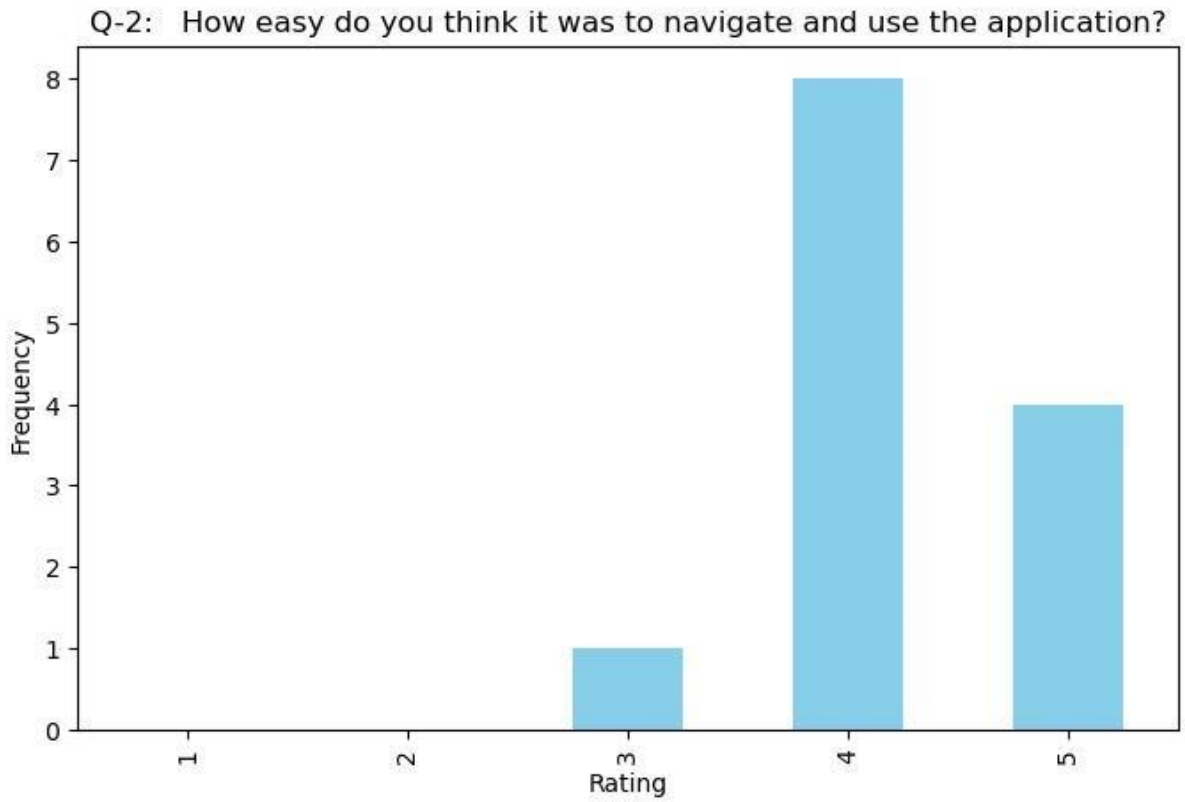
3.2 User Testing

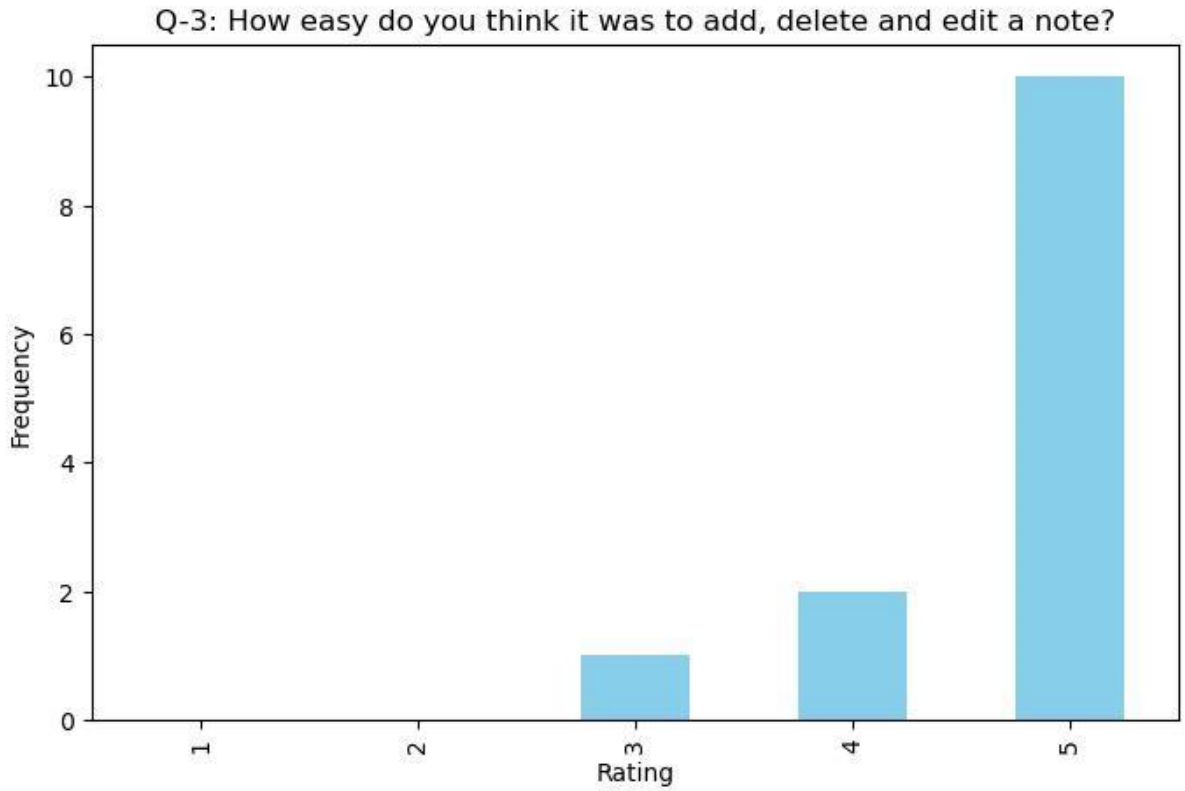
Recognizing that testing based on the design had already been conducted, we figured it would be plausible if we evaluated the ease of site navigation. Besides, we performed user testing to assess the implementation and functionality of the features—Creating, Deleting and Editing a Note, Scientific Calculator, Graphical Calculator and Equation Solver—that we incorporated into our program based on the design proposed above. This user test consisted of 11 objective questions (Appendix A).

3.2.1 First Iteration

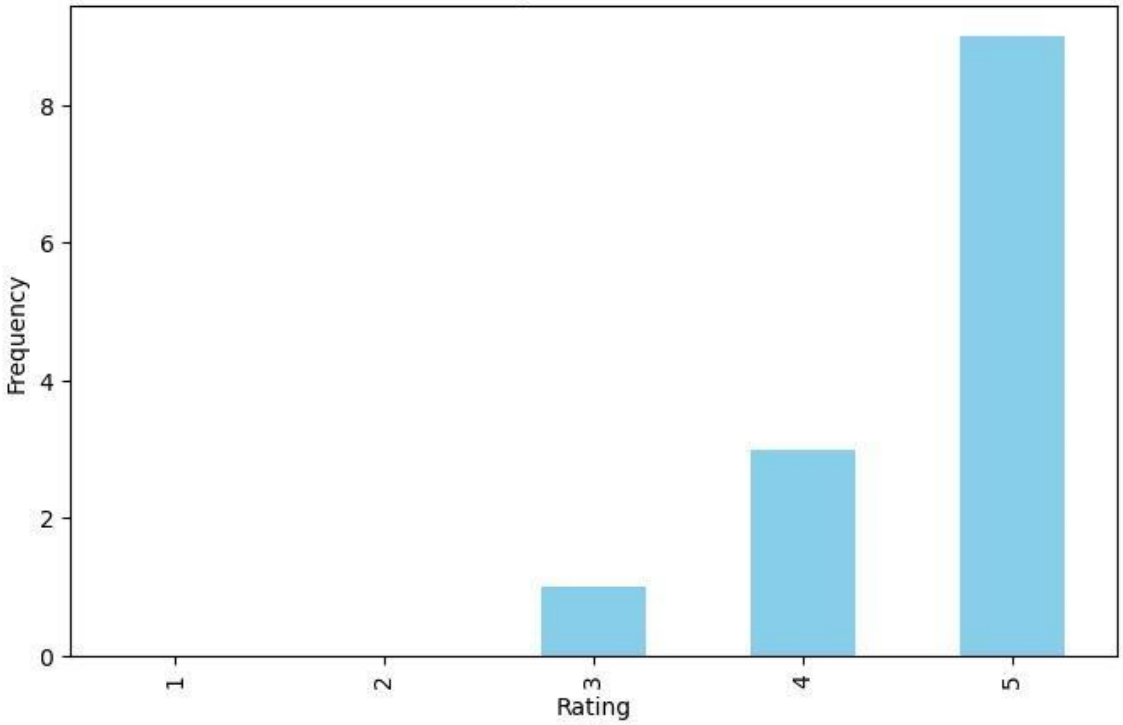
We generated a survey questionnaire comprising 11 questions, each focusing on assessing the ease of use of the site. The questions were designed to evaluate whether users found the site easy to navigate and whether or not they could effectively use the site's features. Following are the survey results for each question:



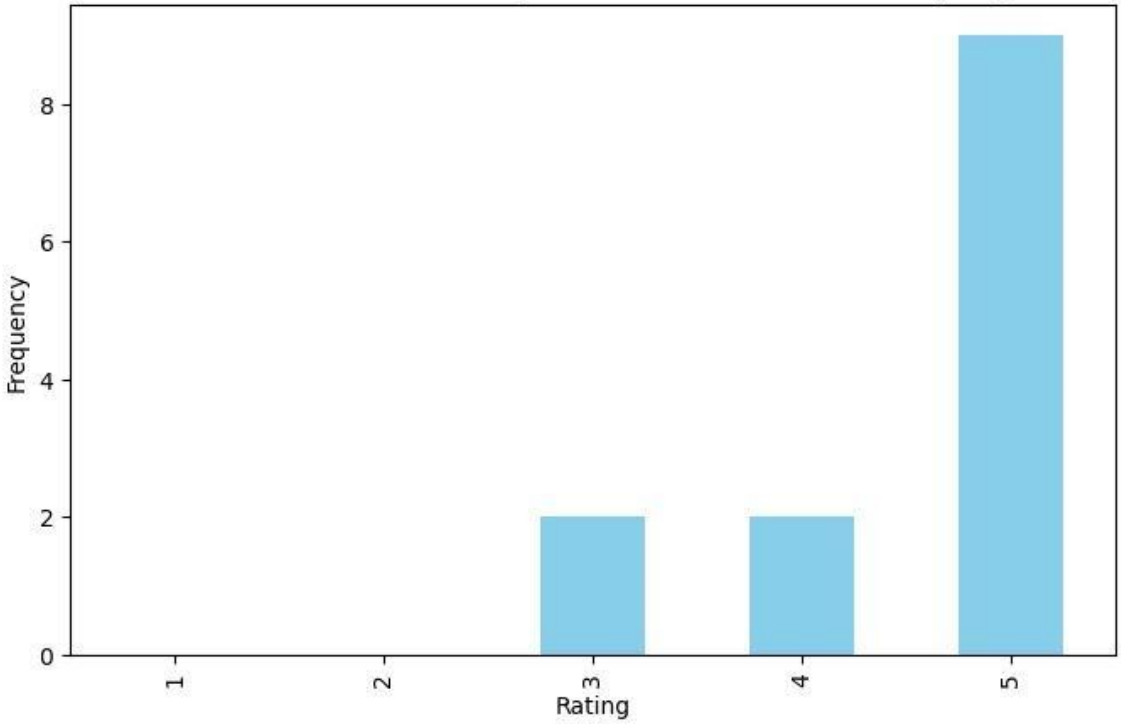

Q-2: How easy do you think it was to navigate and use the application?
Frequency
0
1
2
3
4
5
6
7
8
1
2
3
4
5
Rating

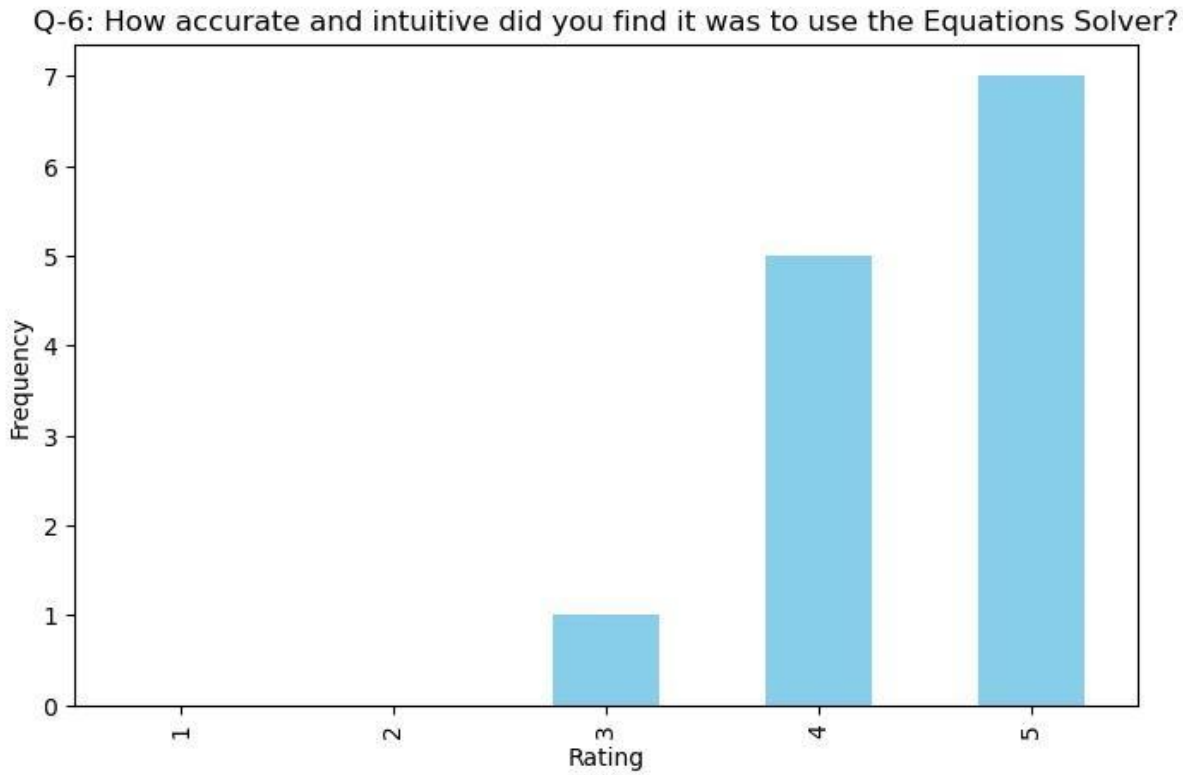

Q-3: How easy do you think it was to add, delete and edit a note?
Frequency
0
2
4
6
8
10
1
2
3
4
5
Rating

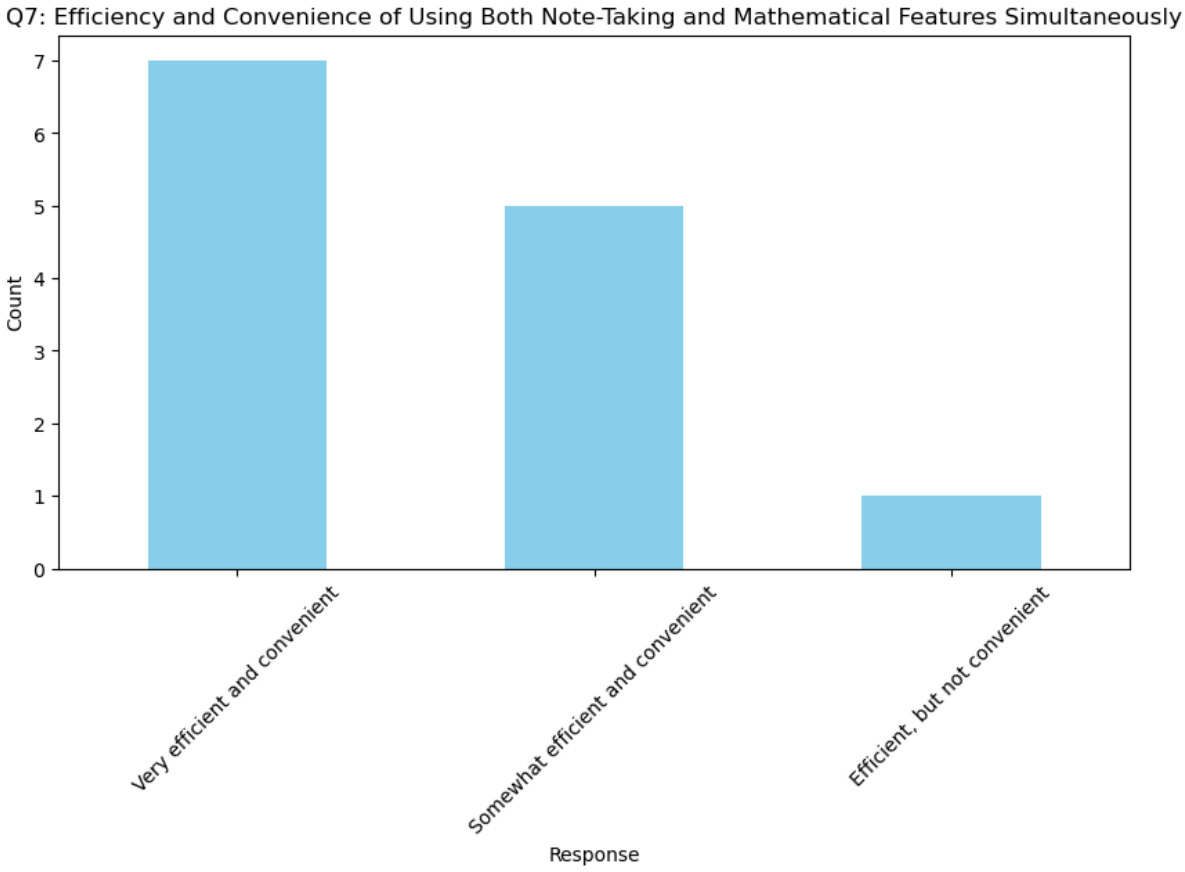
Q-4: How accurate and intuitive did you find it was to use the scientific calculator?

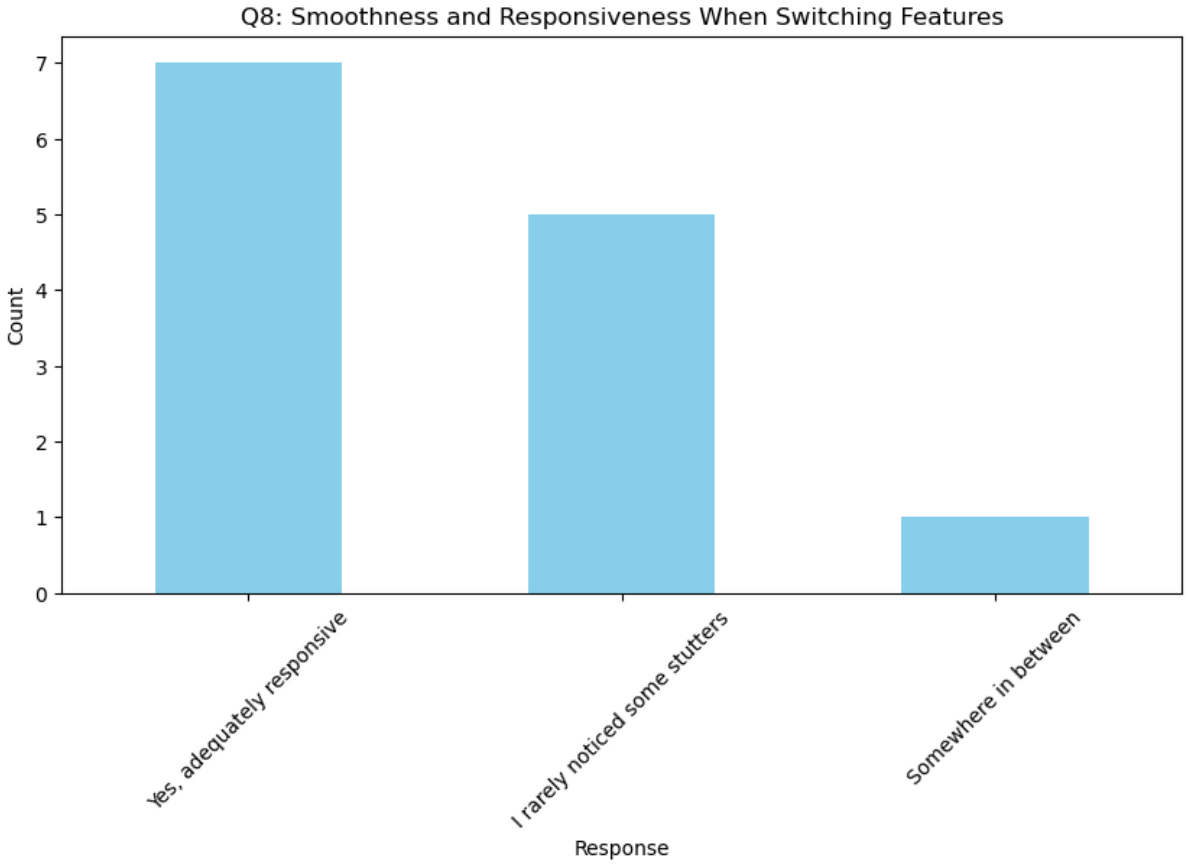


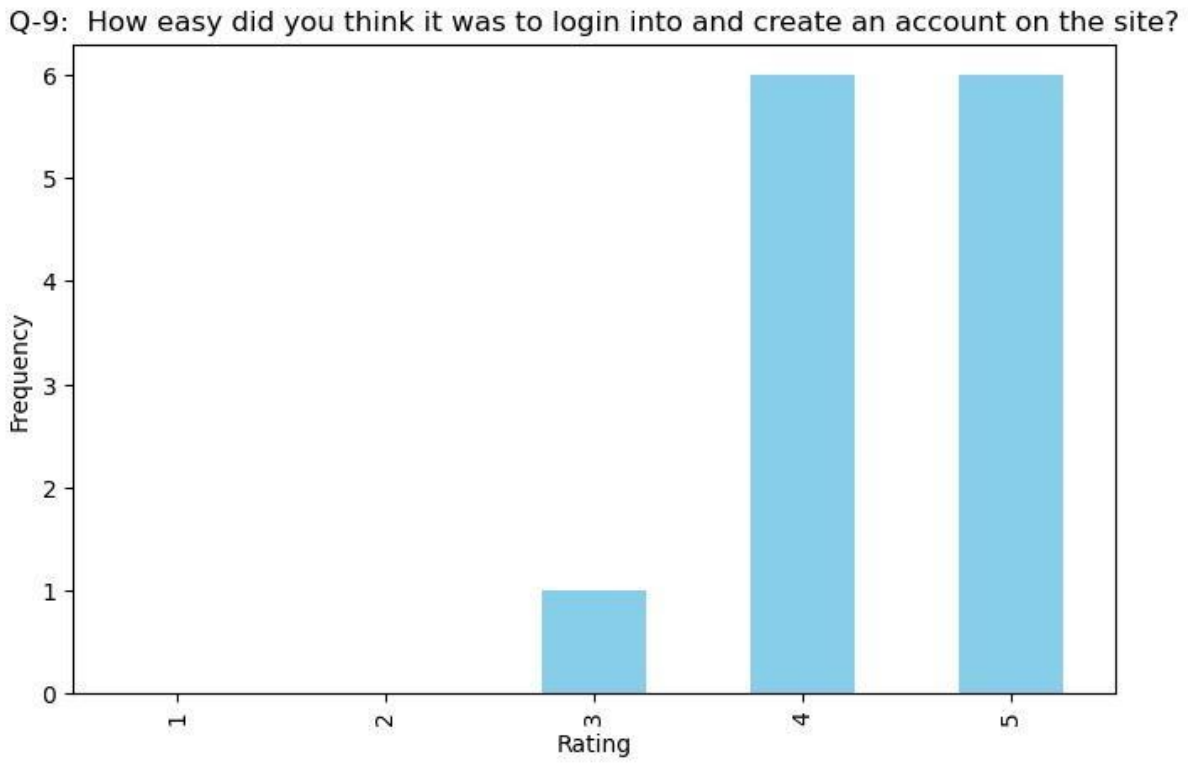
Q-5: How accurate and intuitive did you find it was to use the Graphing calculator?

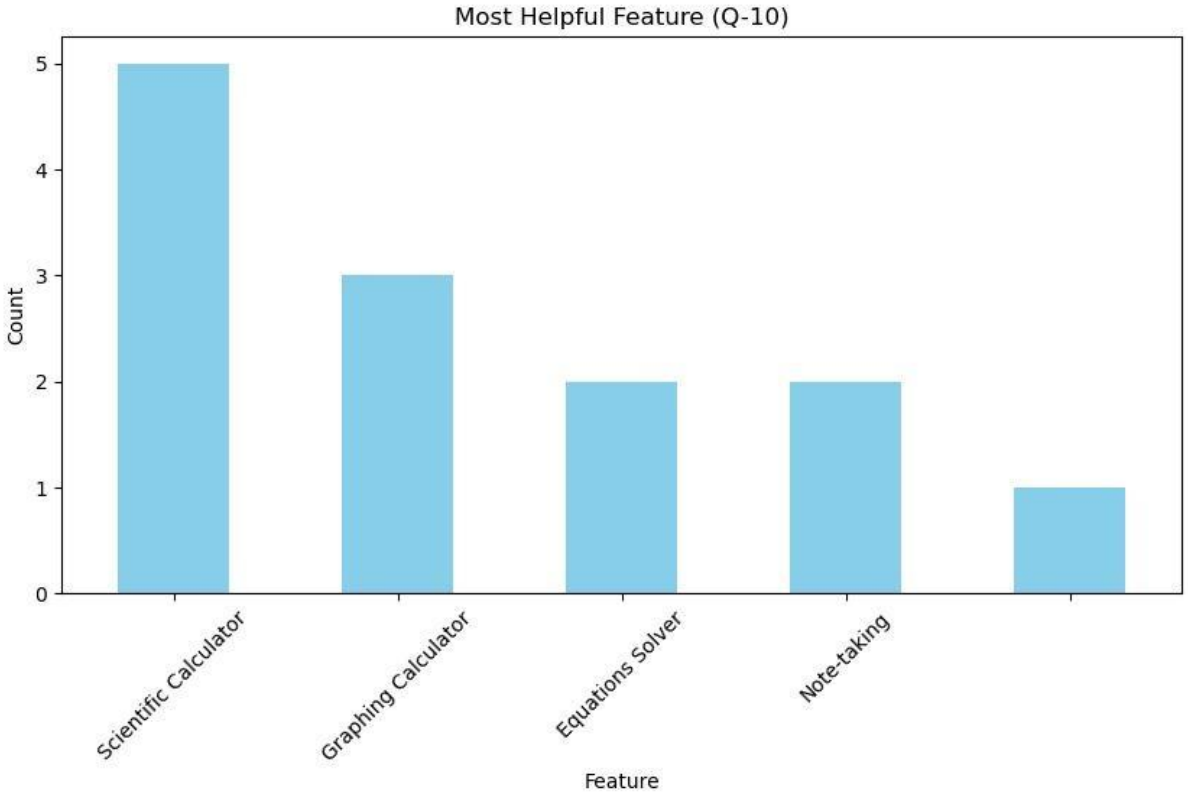



Q-6: How accurate and intuitive did you find it was to use the Equations Solver?
Frequency
0
1
2
3
4
5
6
7
1
2
3
4
5
Rating


Q7: Efficiency and Convenience of Using Both Note-Taking and Mathematical Features Simultaneously
Count
0
1
2
3
4
5
6
7
Very efficient and convenient
Somewhat efficient and convenient
Efficient, but not convenient
Response


Q8: Smoothness and Responsiveness When Switching Features
Count
0
1
2
3
4
5
6
7
Yes, adequately responsive
I rarely noticed some stutters
Somewhere in between
Response


Q-9: How easy did you think it was to login into and create an account on the site?
Frequency
0
1
2
3
4
5
6
1
2
3
4
5
Rating


Most Helpful Feature (Q-10)
Count
0
1
2
3
4
5
Scientific Calculator
Graphing Calculator
Equations Solver
Note-taking
Feature

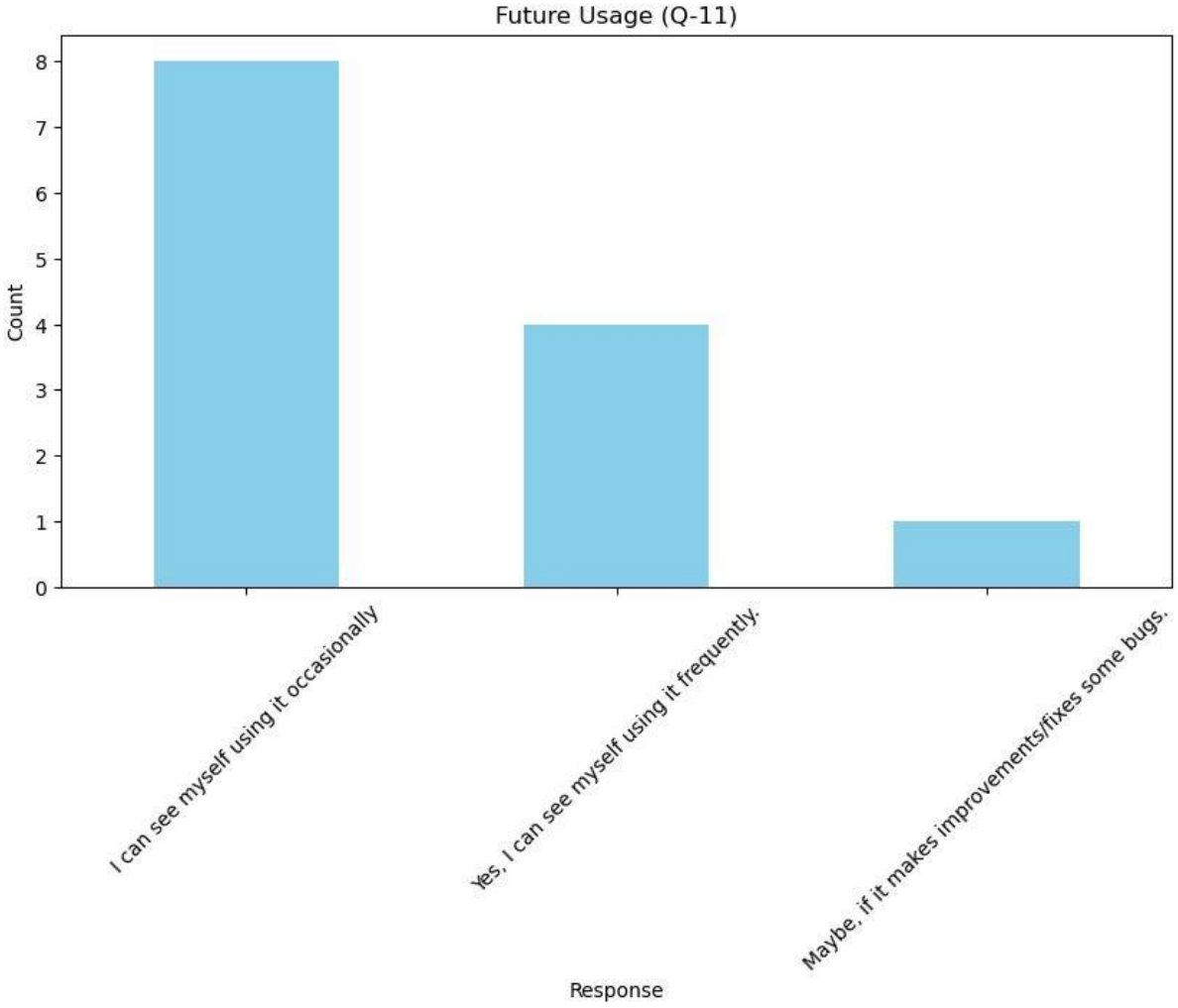

Future Usage (Q-11)
Count
0
1
2
3
4
5
6
7
8
I can see myself using it occasionally
Yes, I can see myself using it frequently.
Maybe, if it makes improvements/fixes some bugs.
Response

Figure 13: 11 Survey Questions Results - Iteration 1

Although the survey revealed a variance in user responses regarding the landing page being aesthetically pleasing, with scores ranging from 1 to 5, most users (60%) agreed that the app was adequately responsive when switching features. Users appreciated the ease of navigation and note management as well. The scientific, graphing, and equations solver calculators were particularly well-received for their accuracy and intuitiveness, with 75% of users rating them highly. However, there were some areas identified for improvement, such as the efficiency and convenience of using both note-taking and mathematical features simultaneously, and occasional performance stutters. Taking the identified areas into account, we made the following pertinent enhancements to the application:

1. We added tags to mouse hover over buttons

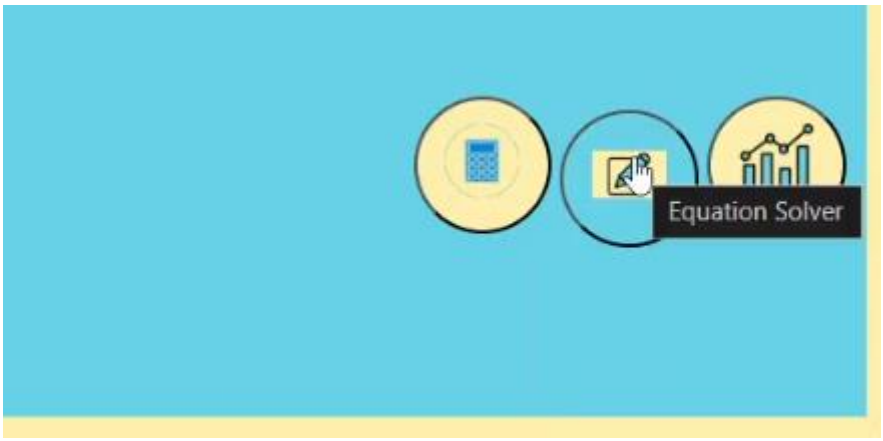


Figure 14: Tags when mouse hover over buttons

To add tags as seen in Figure 14, we added `title` attributes to the buttons in the code to provide tooltips that appear when the user hovers over them. For example, the button for making text bold has the attribute `title="Make text Bold (Ctrl + B)"`, which displays the tooltip "Make text Bold (Ctrl + B)" when hovering over. Similarly, the buttons for italic and underline have `title` attributes with appropriate descriptions. Additionally, the buttons for the Scientific Calculator, Equation Solver, and Graphical Calculator also have `title` attributes to describe their functions. These `title` attributes enhance the user experience by providing helpful information about each button's functionality.

2. We arranged note cards neatly

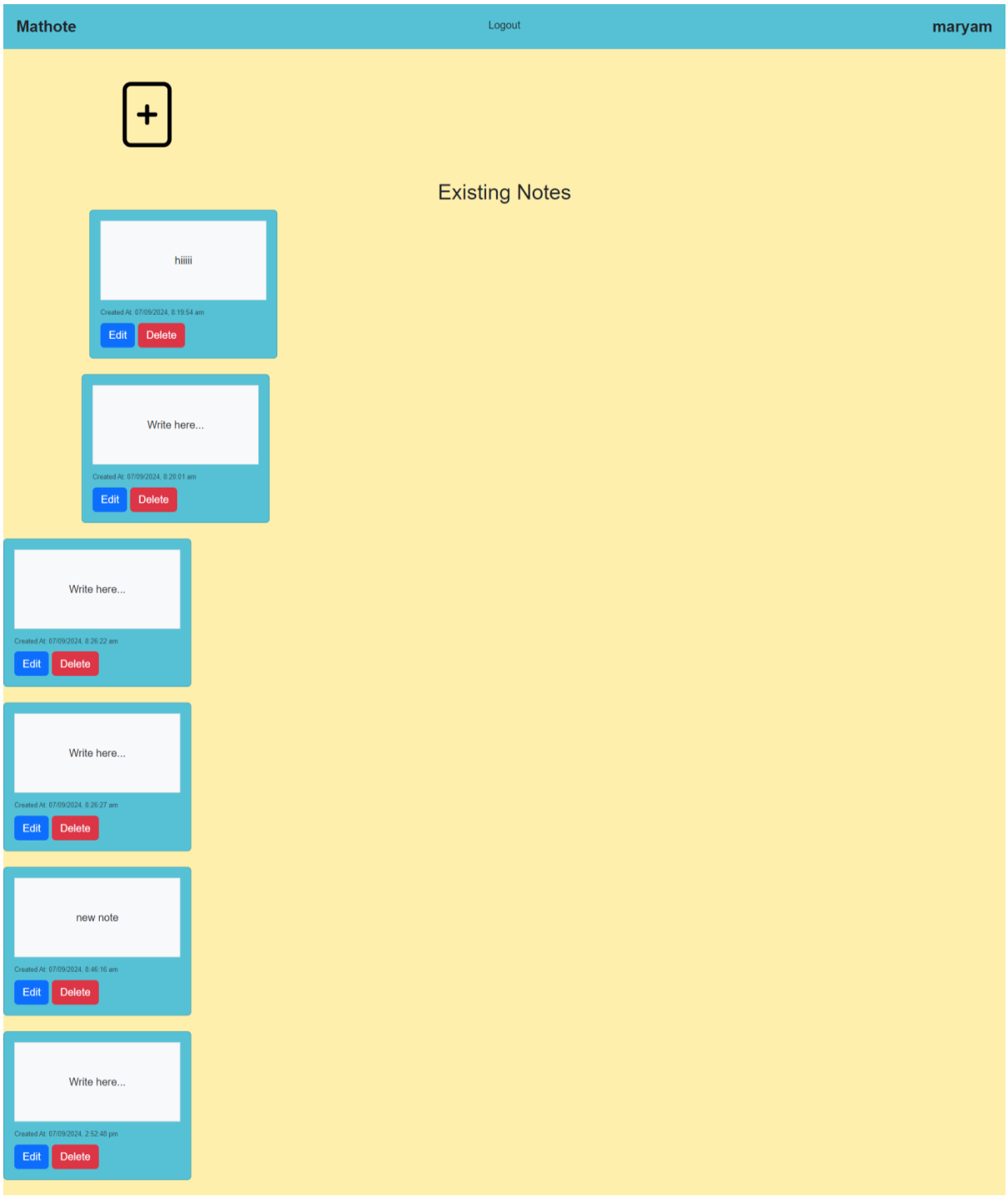


Figure 15: (Before) Note cards arrangement

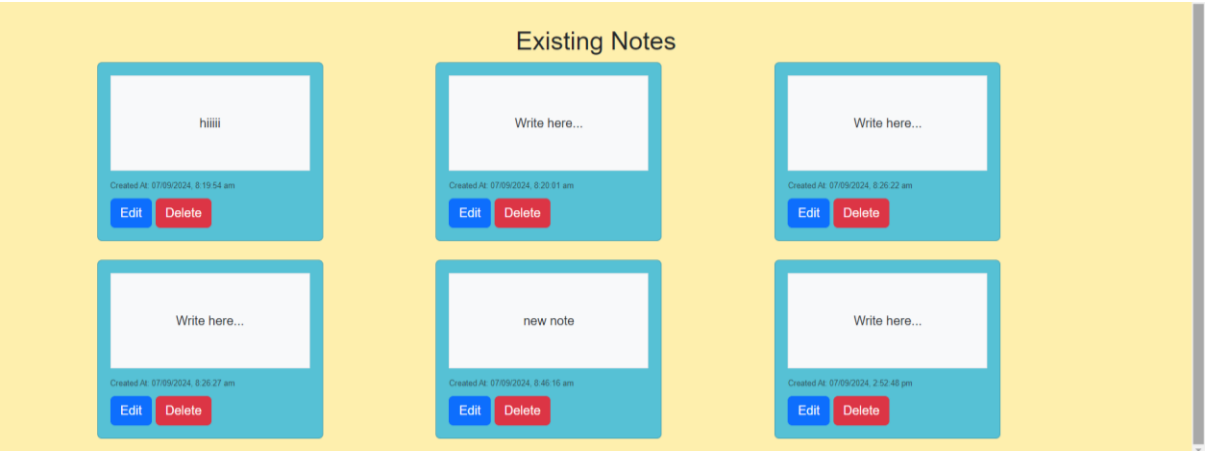


Figure 16: (After) Note cards arrangement

To solve the issue of the cards not being arranged in a grid (Figure 15), we utilised Bootstrap's grid system. Specifically, we wrapped the cards in a `row` div and assigned each card a `col-md-4` class, which ensured that the cards were displayed in a three-column layout on medium and larger screens. This approach leveraged Bootstrap's responsive design capabilities to automatically adjust the layout based on the screen size. Additionally, we added the `mb-4` class to each card to provide consistent spacing between them. These changes allowed the cards to be neatly organised in a grid while preserving their original styling (Figure 16). Below snippet shows the modified code for the cards:

```
<h2 class="existing">Existing Notes</h2>
<% let counter=1; %>
<div class="container">
  <div class="row">
    <% content.forEach(function(note) { %>
      <div class="col-md-4">
        <div class="card mb-4" style="width: 18rem; background-color: #56c1d5;">
          <div class="card-body">
            <p class="card-text border py-5 text-center bg-light" id="note-<%= counter %>"
              style="background-color: #feefad;">
              <%= note.body %>
            </p>
            <p class="card-subtitle mb-2 text-body-secondary" style="font-size: x-small;">
              Created At: <%= new Date(note.timestamp).toLocaleString() %>
            </p>
            <a href="/notes/edit/<%= note.id %>" class="btn btn-primary">Edit</a>
            <a href="#" class="btn btn-danger" onclick="deleteNote('<%= note.id %>')">Delete</a>
          </div>
        </div>
      </div>
      <% counter++; %>
    <% }); %>
  </div>
</div>
</div>
```

Figure 17: Note card Bootstrap code after modifications

3. Calculators should toggle visibility when their icons are clicked

To fix the issue of calculators not disappearing when their icons were clicked again, we edited the `openSciCalculator` and `openGraphicalCalculator` JavaScript functions as seen below. These functions check the current display state of the calculator divs. If hidden (`display: none`), the function sets the display to `block` and initialises the calculator if needed. If visible (`display: block`), the function sets the display to `none`, hiding the calculator. This ensures calculators toggle visibility with each icon click, preventing multiple instances and saving space:

```
//--------------------Scientific Calculator--------------------------------

// Function to open the scientific calculator
function openSciCalculator() {
    const calculatorDiv = document.getElementById("sci-calculator");
    if (calculatorDiv.style.display === "none" || calculatorDiv.style.display === "") {
        calculatorDiv.style.display = "block"; // Show the calculator
        if (!calculatorDiv.dataset.initialized) {
            const calculator = Desmos.ScientificCalculator(calculatorDiv); // Initialize the calculator
            calculatorDiv.dataset.initialized = true;
        }
    } else {
        calculatorDiv.style.display = "none"; // Hide the calculator
    }
}
```

```
//---------------------Graphical Calculator---------------------------------

// Function to open the graphical calculator
function openGraphicalCalculator() {
    const elt = document.getElementById('graphical-calculator');
    if (elt.style.display === "none" || elt.style.display === "") {
        elt.style.display = "block"; // Show the calculator
        if (!elt.dataset.initialized) {
            const calculator = Desmos.GraphingCalculator(elt); // Initialize the calculator
            elt.dataset.initialized = true;
        }
    } else {
        elt.style.display = "none"; // Hide the calculator
    }
}
```

Figure 18: Calculators' JS code after modifications

Additionally, we used CSS Flexbox to ensure that both calculators appear side by side when visible and above the text editor, maintaining a clean and organised layout.

```
<div class="calculators">
    <div id="sci-calculator" class="calculator-container" style="width: 430px; height: 400px;"></div>
    <div id="graphical-calculator" class="calculator-container" style="width: 600px; height: 400px;"></div>
  </div>
```

Figure 19: Calculators' EJS code after modifications

```css
/* Style the calculators section */
.calculators {
  display: flex;
  flex-wrap: wrap;
  gap: 10px;
  margin-bottom: 20px;
}
```

Figure 20: Calculators' CSS code implemented

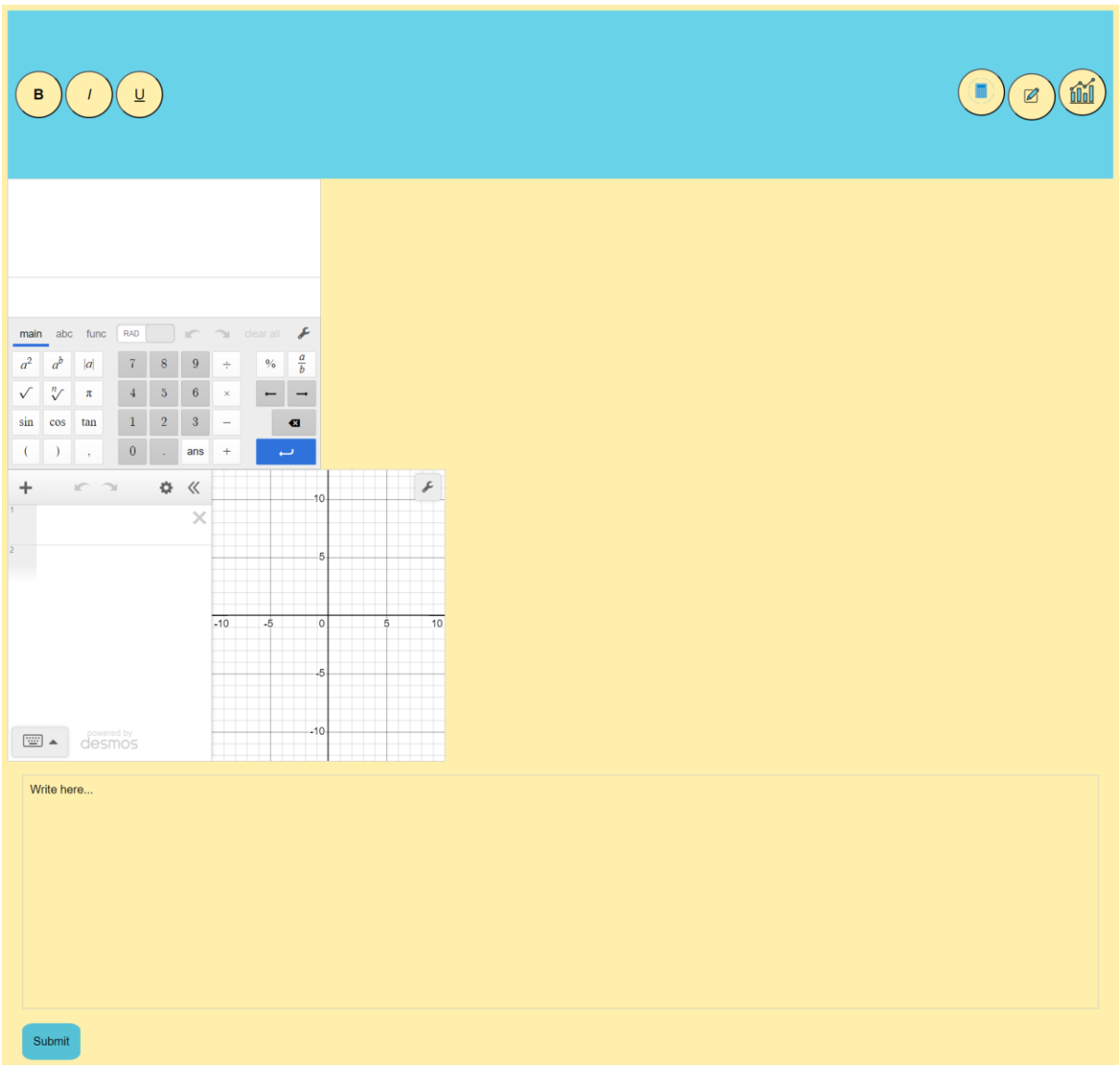


Figure 21: (Before) Calculators displayed below each other

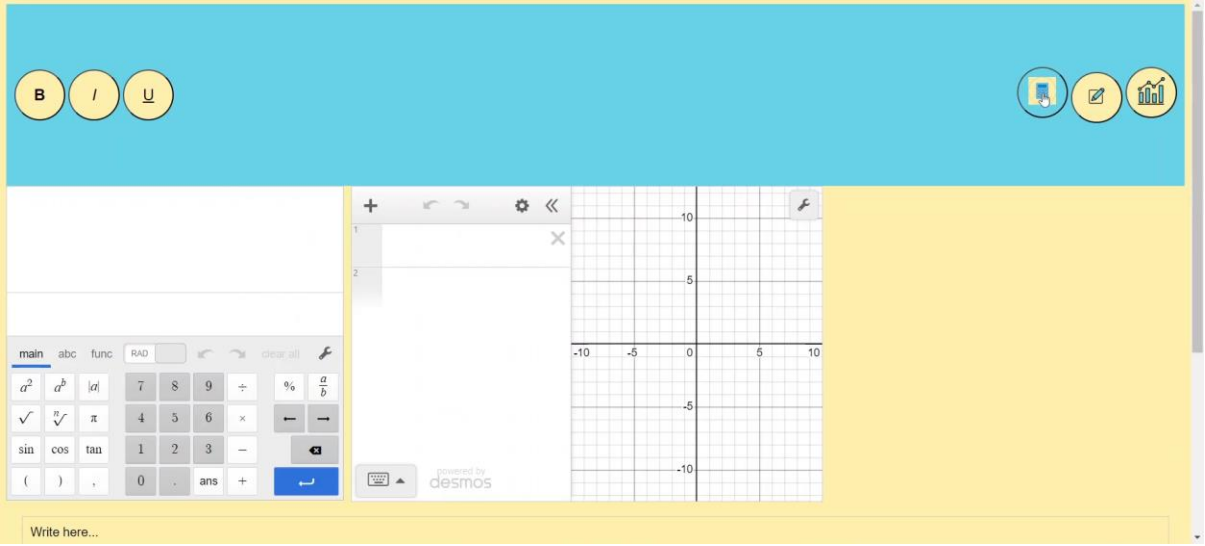


Figure 22: (After) Calculators displayed next to each other

As seen above in the before and after snapshots, calculators appear side by side rather than appearing below each other, reducing white space.

4. Utilise the excessive white above note editor

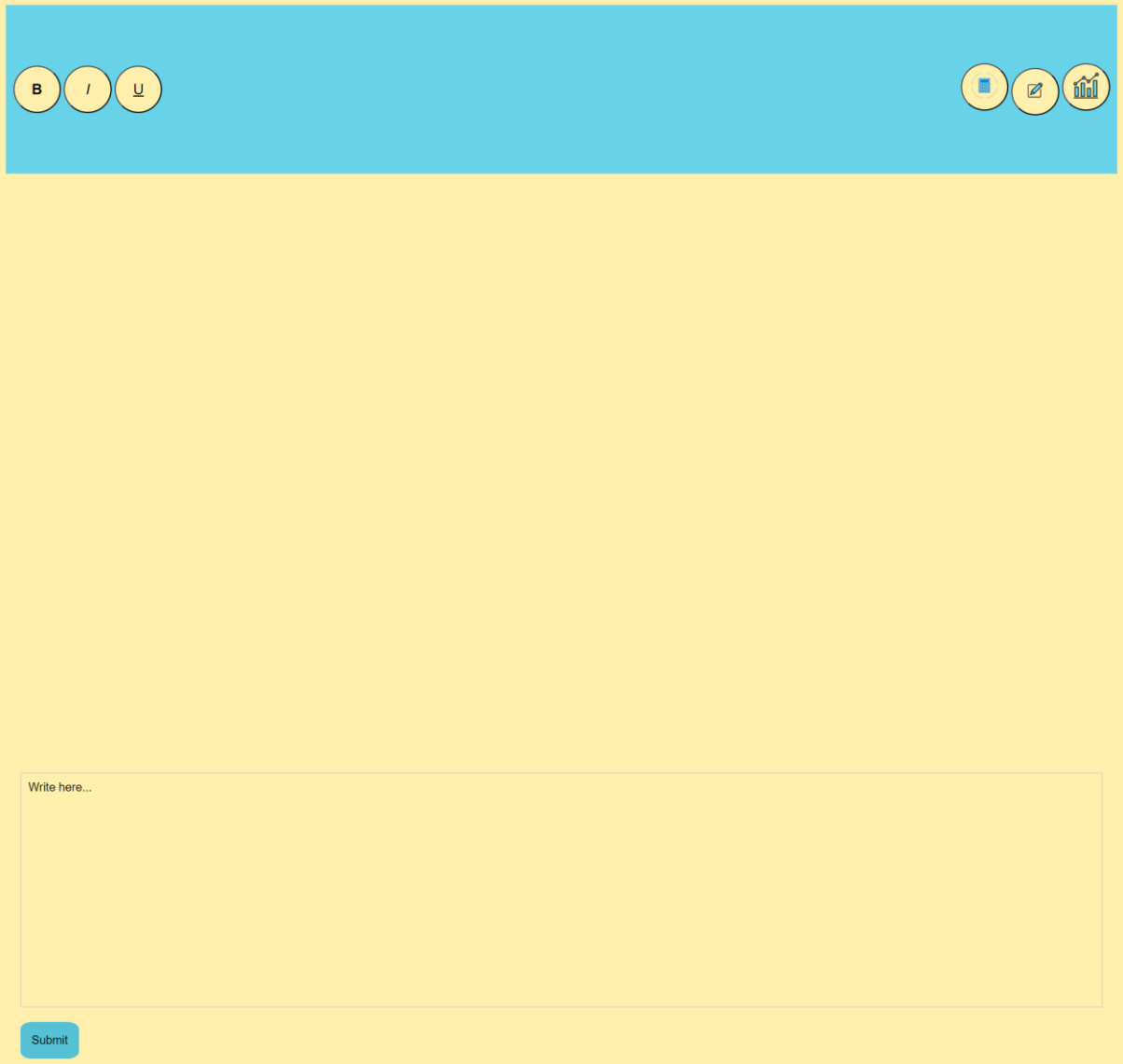


Figure 23: (Before) Excessive white space between toolbar and note editor

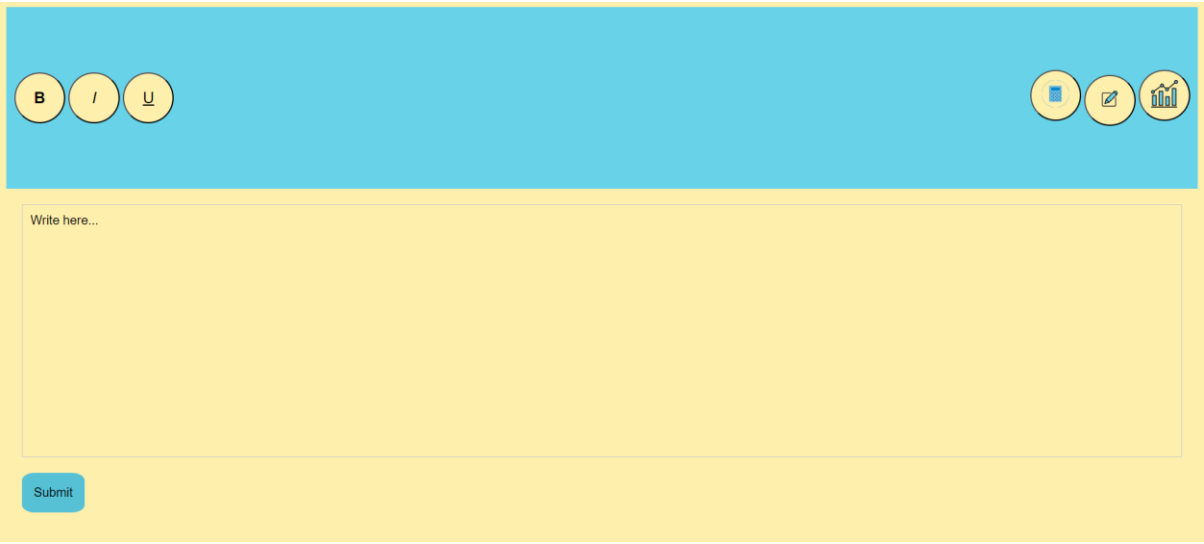


Figure 24: (After) Utilised white space between toolbar and note editor

Firstly, we reduced extra white space by placing calculators next to each other rather than below each other (discussed in point 3). However, as seen in Figure 23, there was still a lot of white space. It took us some time to figure out that this extra white space was caused by the 'container of calculators,' even though the calculators are hidden and can be hidden again if their icons are clicked:

```css
/* Hide the calculator container by default */
.calculator-container {
  display: none;
}
```

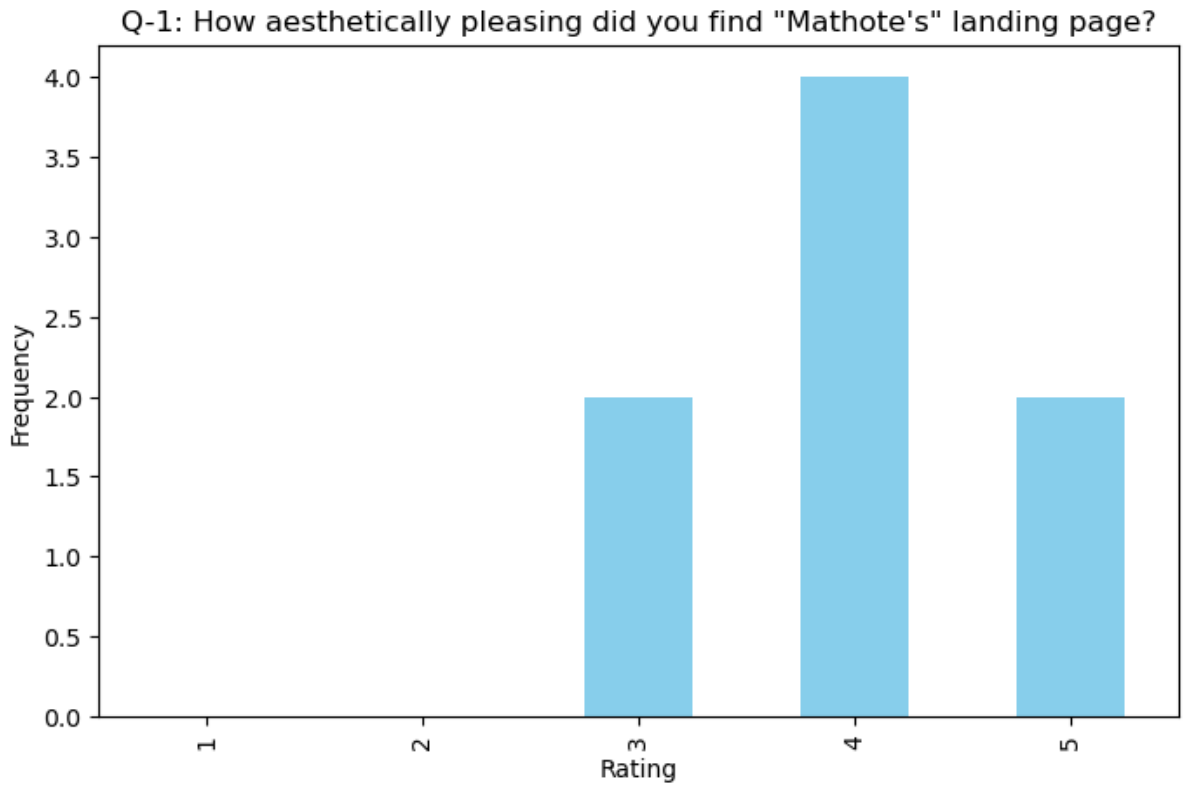
Figure 25: Calculators another CSS code implemented

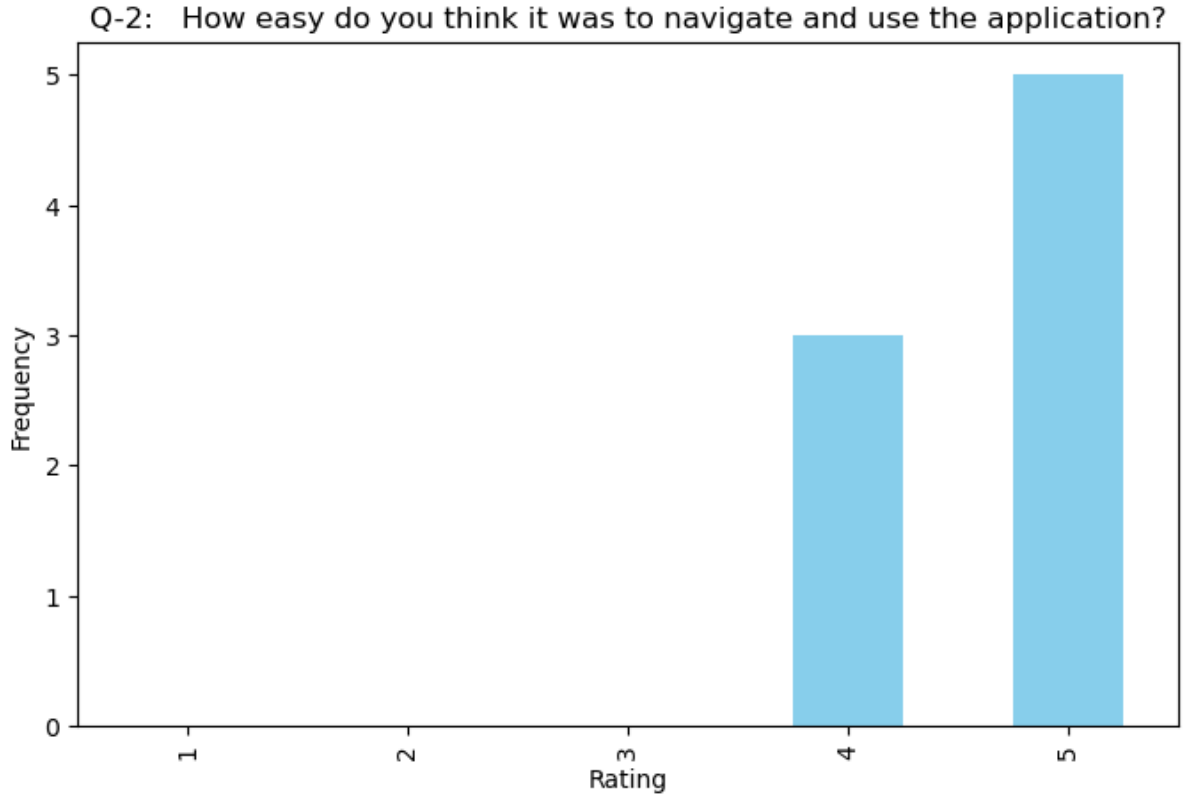
Setting the calculator containers to `display: none` in CSS ensures they don't take up any space on the page when hidden. This prevents unnecessary white space and layout issues, even though they are initially hidden by JavaScript. Without this CSS rule, hidden elements can still affect the layout and spacing of other elements.

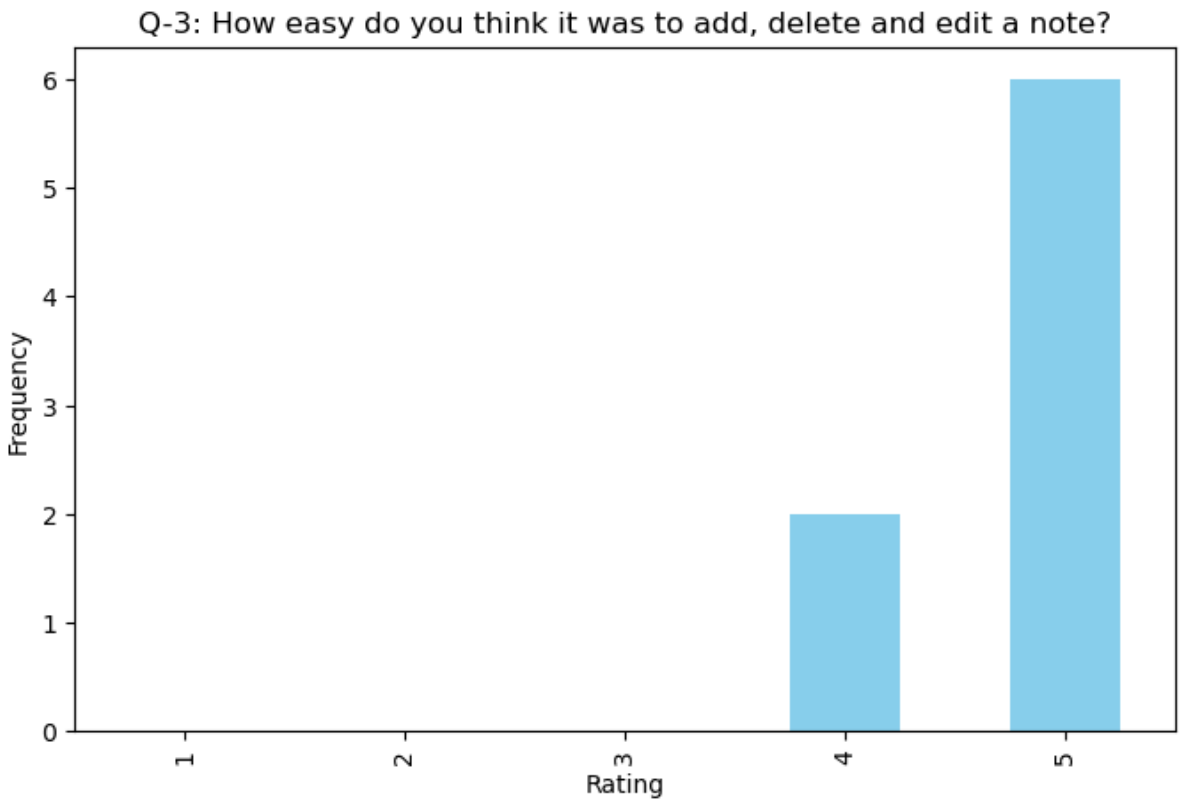
After incorporating these changes, we rolled out the survey again to gauge user responses for second iteration.

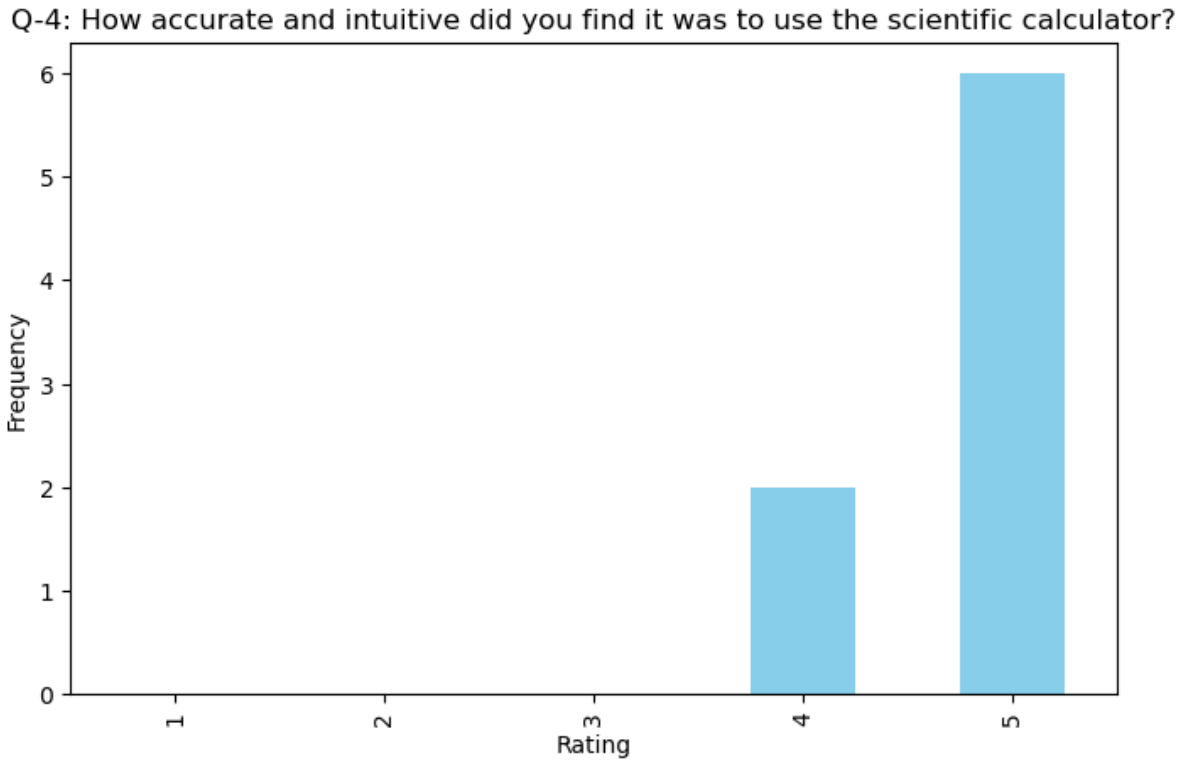
3.2.2 Second Iteration

Following are the results for each question of the redistributed survey:

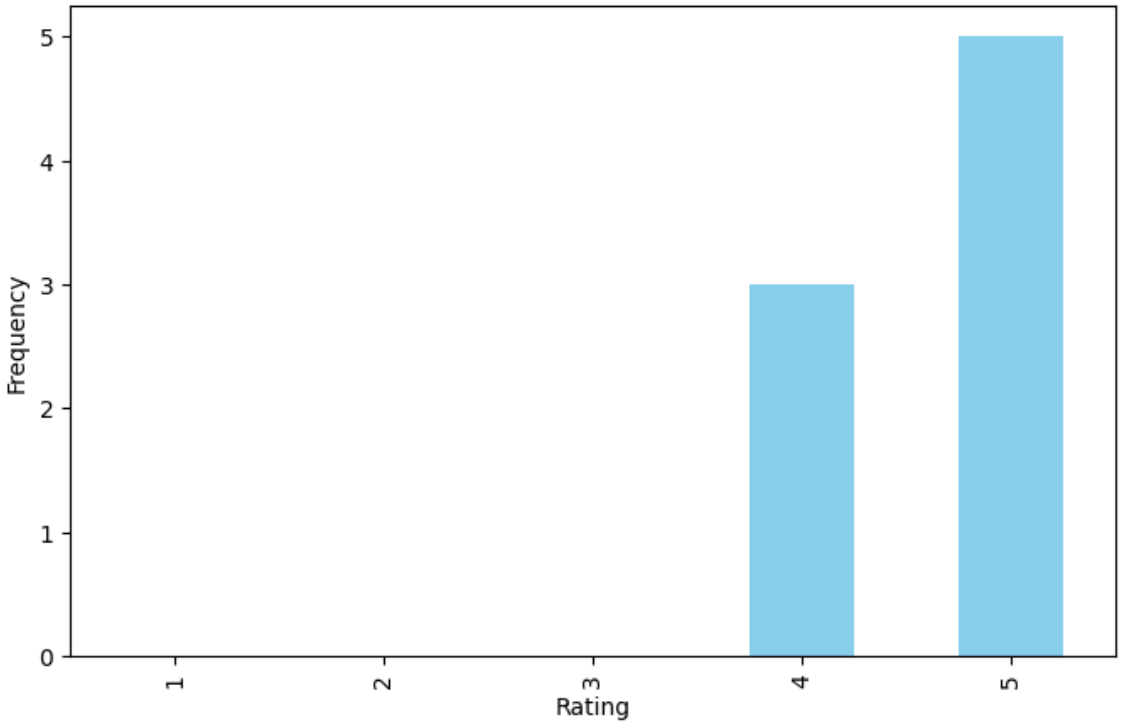




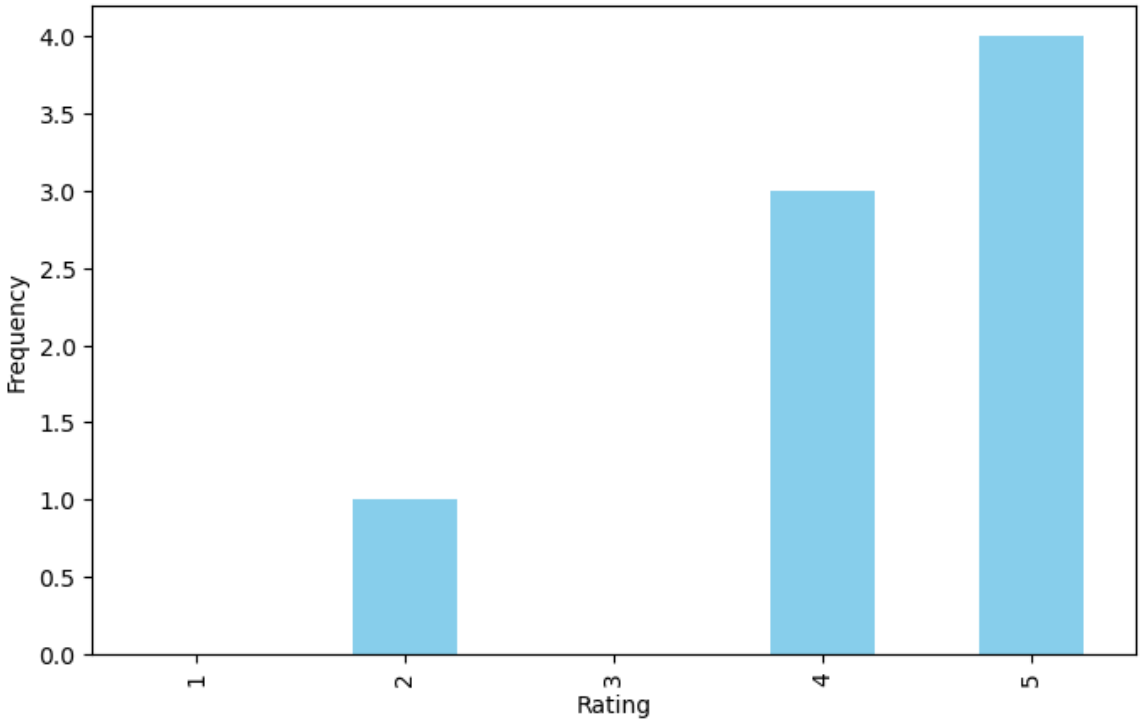

Q-3: How easy do you think it was to add, delete and edit a note?
Frequency
0
1
2
3
4
5
6
1
2
3
4
5
Rating


Q-4: How accurate and intuitive did you find it was to use the scientific calculator?
Frequency
0
1
2
3
4
5
6
1
2
3
4
5
Rating

Q-5: How accurate and intuitive did you find it was to use the Graphing calculator?

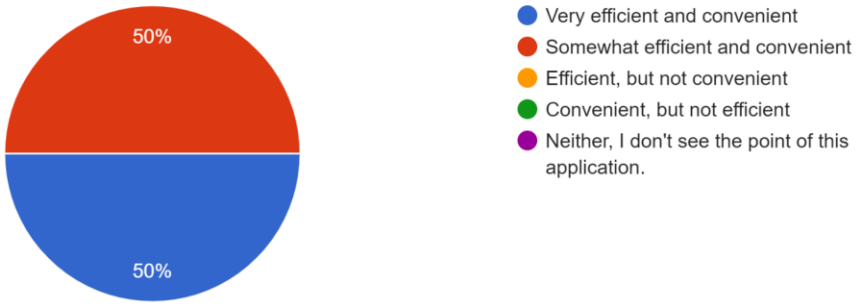


Q-6: How accurate and intuitive did you find it was to use the Equations Solver?



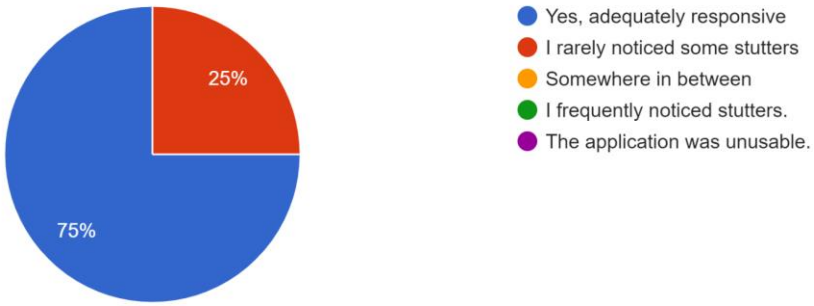
Q-7: In your opinion, do you think it was efficient and convenient to use both the note-taking and mathematical features of the application simultaneously?

8 responses

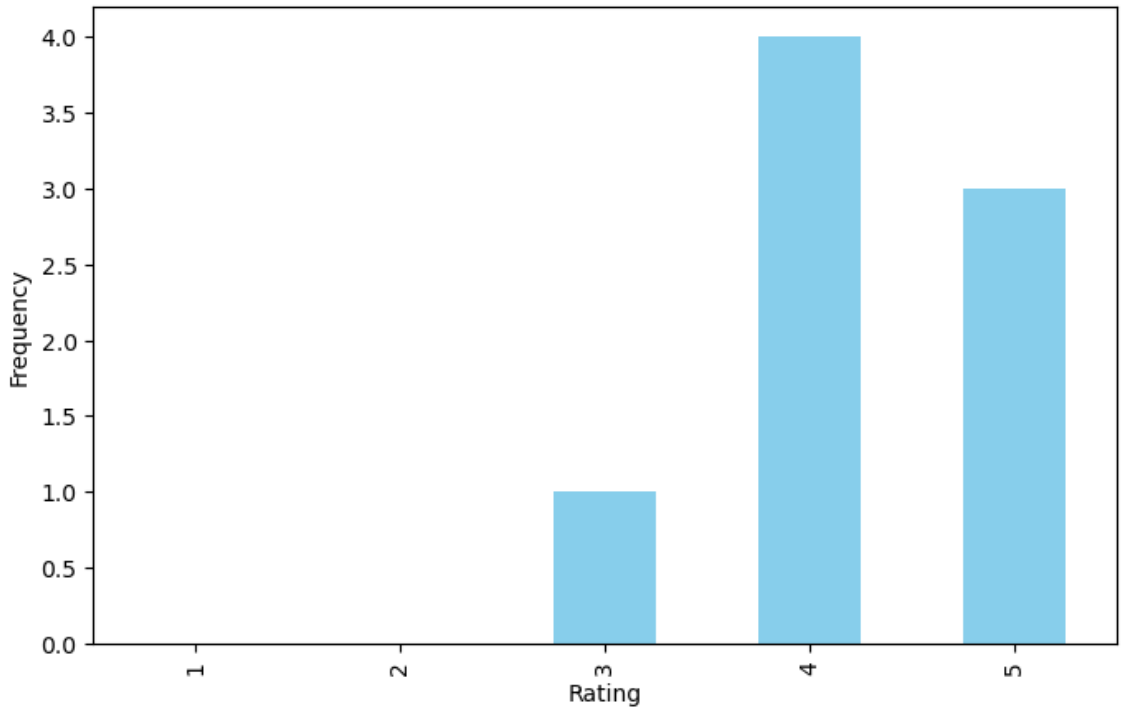


Q-8: Was the app smooth and responsive when switching from one page or feature to another?

8 responses

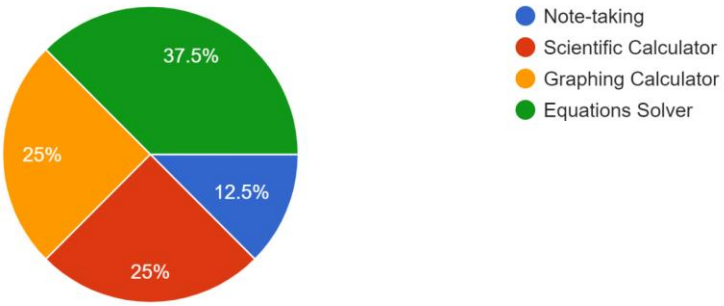


Q-9: How easy did you think it was to login into and create an account on the site?



Q-10: Which feature of "Mathote" did you find the most helpful?

8 responses



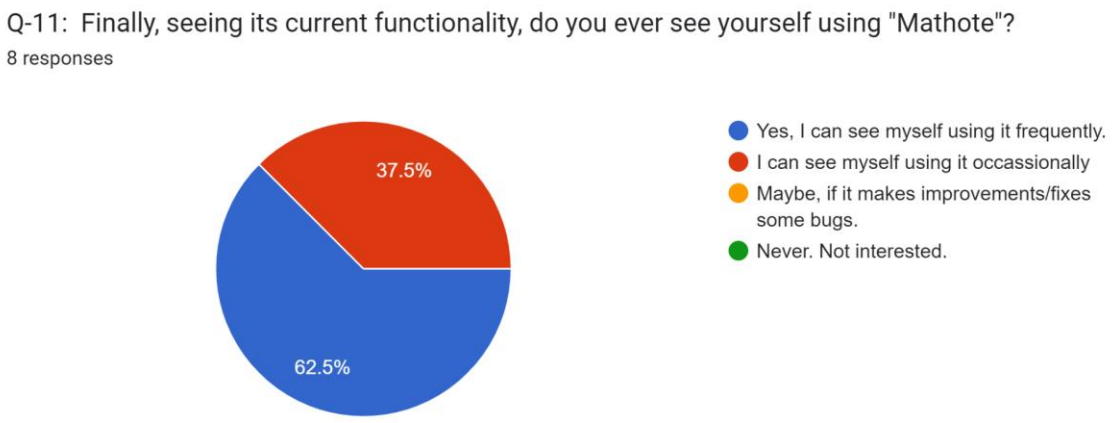


Figure 26: 11 Survey Questions Results - Iteration 2

The second iteration of the *Mathote* survey provided valuable insights into user experiences and perceptions. The survey revealed that most users found the landing page to be aesthetically pleasing, with most scores ranging from 4 to 5. The ease of navigation was rated highly as well, indicating strong usability. Users found it easy to manage notes, with high ratings (4 to 5) for adding, deleting, and editing notes. The scientific, graphing, and equations solver calculators were also rated highly for accuracy and intuitiveness.

When it came to the efficiency and convenience of using both note-taking and mathematical features simultaneously, responses varied. Many users found it "Very efficient and convenient," while others rated it as "Somewhat efficient and convenient" or "Efficient, but not convenient." This indicates that there is room for improvement in making the experience more seamless. Besides, most users found the app to be adequately responsive, though with a few noted occasional stutters, suggesting overall good performance with minor issues to address.

The ease of logging in and creating an account was rated highly as well, with most users giving it a 4 or 5. The most helpful features mentioned were the Scientific Calculator, Equations Solver, Graphing Calculator, and Note-taking, highlighting the value users place on these specific tools. Finally, the majority of users responded positively when asked if they see themselves using the app in the future, indicating they would use it frequently or occasionally. Some users mentioned they might use it if improvements are made, suggesting that ongoing enhancements could further increase user satisfaction and retention.

Overall, the survey indicates that while the app is generally well-received, there are areas for improvement, particularly in the visual appeal, seamless integration of features, and minor performance issues.

On the basis of the Iteration 2 results, we made changes to the functionality of equation solver. Results to the equation solving API are generated at the bottom of the screen and sometimes are not visible to the user unless they scroll down. We implemented an automatic scrolling feature in the code i.e. as follows:

```
// Scroll to the result div
resultDiv.scrollIntoView({ behavior: "smooth" });
```

Figure 27: Equation Solver JS code after modifications

In function displayResults(results) in script.js. Now, when an equation is solved, the page automatically scrolls to the newly created result div, ensuring that users can immediately see the solution without any confusion and without having to manually scroll down. This enhancement significantly improves the usability and overall experience for users.

3.3 Heuristic Evaluation

We also conducted Heuristic Evaluation to gauge where *Mathote* stands according to expert guidelines. We designed a survey based on Jakob Nielson's 10 evaluation guidelines. The questionnaire handled objective type questions on a 5-point scale with 5 being "Extremely Agree" and 1 being "Extremely Disagree". This approach enabled us to systematically assess the usability of our application in line with established evaluation guidelines [6] (Appendix B).

1 - 2	
3 - 4	
5	

Figure 28: Key for 5-point scale

Figure 29: Usability Heuristics

In order to evaluate the response on the Heuristic Evaluation [7], for each of the questions we calculated the average score and then extracted the integer part of that average:

Heuristic	Scale For Evaluation
Visibility of system status	3
Match between system and real world	3
User Control and Freedom	3
Consistency and Standards	4
Error Prevention	2
Recognition rather than recall	2
Flexibilty and Efficiency Use	4
Aesthetic and Minimalist	4
Help user recogize, diagnose and recover from errors	4
Help and Documentation	1

Figure 30: Heuristic Evaluation Results

The results in general demonstrate the limitations of our application. A score of 4 on the heuristics mentioned above suggest that the system performs well in maintaining consistency, allowing users flexibility, having a clean design, and offering effective error recovery support. Whereas a score of 3 suggests that there is room for improvement in some areas, though the system still functions relatively well. A score of 2 in "error prevention" and "recognition rather than recall" suggests that the system struggles with error prevention and making information readily available, instead of relying on users to remember details. Lastly a score of 1 in "Help and Documentation" suggests the lack of documentational support available for *Mathote.*

Some improvements that we propose for future works after analysing the results of the evaluation include keeping the users informed about the status change by providing adequate feedback to users in *Mathote*, making the system match the real world by simplifying the UI even more than the current UI design. Providing the user more control and freedom with the application, preventing errors from occurring, developing the system such that recognition is promoted rather than recalling and documenting a guide for *Mathote* are additional improvements needed.

3.4 Colours

Since *Mathote* is a note-taking and mathematics application, for our interface we thought of opting for colours that boost productivity. Among the colours that are known for boosting productivity - red, blue, yellow, green - we opted for the combination of blue and yellow colours as our base for the application from ColorHunt.

Figure 31: Colour Palette used in our application design: 2nd and 4th (from top)

From the colour palette above we choose the bottom-most yellow colour and the second blue colour from the top.

Blue for the reason that it "calms the mind and aids concentration" thus enhancing wakefulness. Blue also encourages better mind flow and performance by lowering blood pressure and slowing the heartbeat hence promoting a relaxing atmosphere to study in. Stronger shades support brain's thought processes, while lighter shades help improve concentration [8].

Yellow on the other hand is chosen for the reason that it is energising and "radiates positivity". Yellow gets you right in the mood for producing great work [9].

3.5 Components of the Software

The application features a front-end that allows users to create, delete, and edit notes, as well as access tools such as a Scientific Calculator, Graphical Calculator, and a Step-by-Step Equation Solver. The backend, on the other hand, contains functionality for storing notes in the database, altering them, and deleting them from the database. Additionally, the backend handles API integrations for the Step-by-Step Equation Solver, Scientific Calculator, and Graphical Calculator, ensuring seamless functionality across all features.

3.6 Presentation Layer

The web application itself constitutes the presentation layer of the OSI model. We made our presentation layer using simple HTML, CSS and JavaScript with a slight use of Bootstrap CSS library.

The Views directory (Figure 32) inside the Project directory handles the presentation layer of the application. It consists of a register page where a user can register to the site, a login page

whereby a user can login into their personal dashboard, a homepage for creating a note, an edit note page for editing a note. The application features an index (landing) page, Figure 33, that serves as the default entry point, and renders the site landing page consisting of a login and a register button. The user can choose to either log in if they are already registered or register if they are new to the site. After the user has registered to the site, they then have to login as well to make use of the application. Only registering does not grant a user the access to its features. Once authenticated, users are redirected to the homepage (Figure 34), which acts as a user's personal dashboard and is the primary interface for managing their notes.

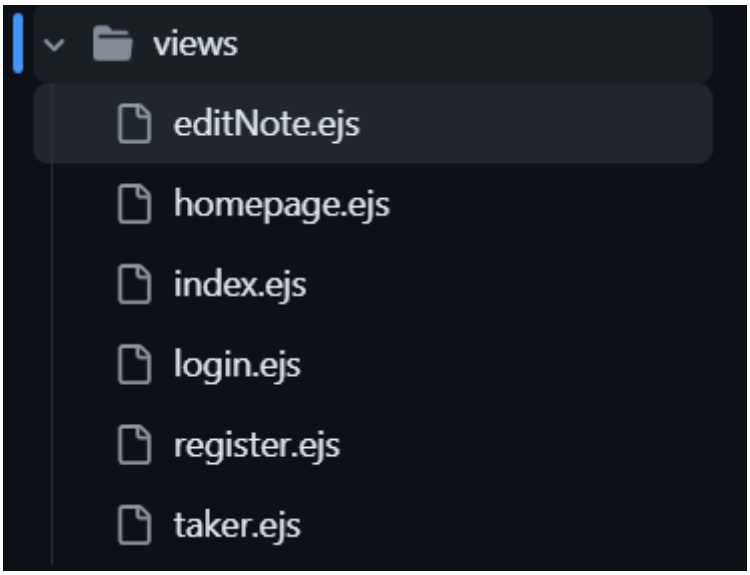


Figure 32: Views Folder from the Project Directory

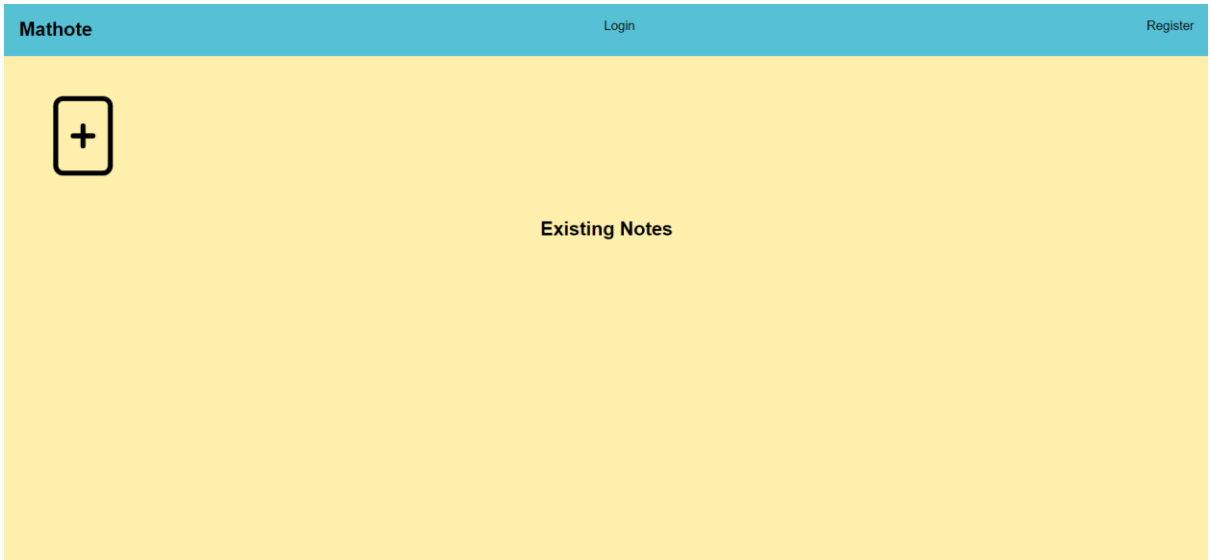


Figure 33: The default landing page of Mathote

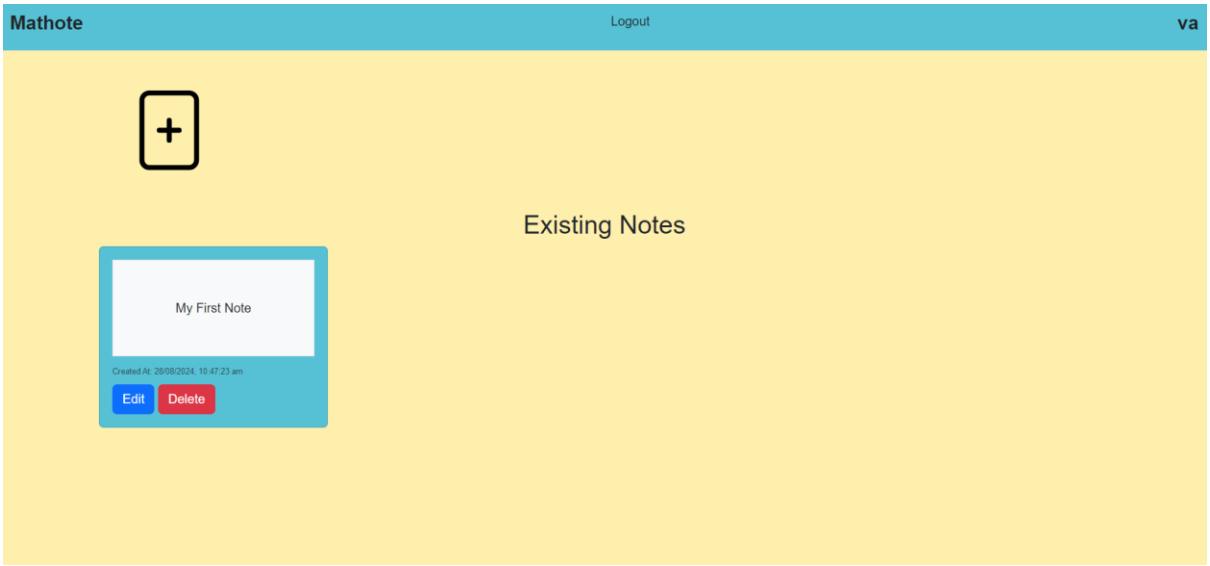


Figure 34: User's personal dashboard once their user session is active

3.7 Structure and Implementation

Our main entry to the application is via the index.js file which navigates the user to the index page of the site. The index.js imports all the modules that are needed for the application to run. It also initialises the routes that the application is made on.

While the index.js navigates the index page of the site, the rest of the routes are rendered via the routes folder containing users and the notes routes. The user routes handle the endpoints related to registering and logging in, seeing as they display the login and register pages. They also process the body of the forms via POST method to store user information such as username, email and password onto the database.

The notes route on the other hand handles the CRUD operations of the database with regards to creating, deleting and editing a note. This route also displays the pages for editing and creating a note via the GET method.

Whereas as the backend is managed using the process mentioned above, the frontend is managed using the views folder which contains all the frontend template files for rendering the home, register, login, creating a home page and editing a note page. These pages are styled using the style.css file in the public folder.

The public folder also contains JavaScript code for handling the Scientific calculator, Graphical Calculator and Step by Step equation solver.

3.8 Browser Support

The code in our app includes a mix of EJS, CSS, and JavaScript, which are widely supported across modern web browsers. The EJS and CSS elements used, such as `<div>`, `<button>`, and CSS properties like `font-family` and `background-color`, are universally supported by all

major browsers, including Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, and Opera [10]. The JavaScript functions, including those for formatting text, opening calculators, and handling fetch requests, are also supported across these browsers [11]. Additionally, libraries like Desmos for scientific and graphical calculators are compatible with modern browsers.

However, it is important to ensure that JavaScript is enabled in the user's browser settings for the code to function correctly. In Firefox, we encountered an issue where alert messages would appear and then immediately disappear without giving the user the time to interact with them. This happened whenever the page reloaded or navigated to a different page (e.g., using window.location.href or window.location.reload()). The alert message would briefly show up but vanish almost instantly due to the page reload or navigation, preventing the user from seeing or dismissing the alert. To address this, we used setTimeout() to introduce a brief delay after the alert is displayed, ensuring the alert message remains visible until the user clicks "OK." Only then does the page reload or redirection occur, allowing the alert box to function correctly without being cut off prematurely.

3.9 Register and Login

We created both the register and login pages in our application to manage user authentication. The register page (Figure 38 and 39) allows users to sign up by providing an email address, a username, and a password. The password is encrypted before being securely stored in the database. Once registered, users are redirected back to the landing page, from where they can access the login page. To initiate a session, the user must log in using the username and password they registered with (Figure 36 and 37). This process ensures secure user authentication while maintaining a seamless flow between registration and login.

In order to implement the login and register functionality, we created a separate route in the routes folder called the "users" route that integrates functionality to GET the register and login pages and POST the body of the forms inside those login/register pages to the database behind:

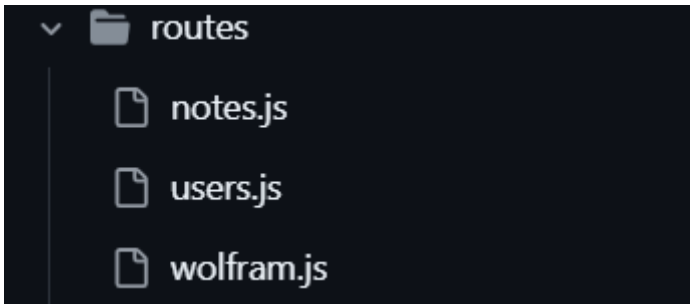


Figure 35: Routes created in Mathote

```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Login</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5
    <link rel="stylesheet" href="/public/style.css">
    <style>
        body {
            background-color: #feefad;
        }
    </style>
  </head>
  <body>
    <div class="container-sm">
        <header class="login-header">
            <h1>Mathote</h1>
        </header>
        <form action="/users/login" method="post">
            <div class="mb-3">
                <label for="username" class="form-label">Username</label>
                <input type="text" class="form-control" id="username" name="username">
            </div>
            <div class="mb-3">
              <label for="inputPassword5" class="form-label">Password</label>
              <input type="password" id="password" name="password" class="form-control" aria-describedby="passwordHelpBlock">
            </div>
            <button type="submit" class="btn btn-primary">Log In</button>
        </form>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9f
  </body>
</html>
```

Figure 36: Login page HTML

```
// Route to render the login page
router.get("/login", (req, res) => {
  res.render("login");
});

// Route to handle user login
router.post("/login", (req, res) => {
  const { username, password } = req.body;

  // Retrieve the user from the users table
  db.get(`SELECT * FROM users WHERE username = ?`, [username], (err, user) => {
    if (err || !user) {
      return res.send("User not found");
    }

    // Compare the provided password with the stored hashed password
    const passwordIsValid = bcrypt.compareSync(password, user.pass_word);

    if (!passwordIsValid) {
      return res.send("Invalid password");
    }

    // Create a session for the user
    req.session.userId = user.user_id;
    req.session.username = user.username;

    console.log(req.session.userId, req.session.username);

    res.redirect("/homepage");
  });
});
```

Figure 37: Login route

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Register</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5
    <link rel="stylesheet" href="/public/style.css">
    <style>
        body {
            background-color: #feefad;
        }
    </style>
  </head>
  <body>
    <div class="container-sm">
        <header class="login-header">
            <h1>Mathote</h1>
        </header>
        <form action="/users/register" method="post">
            <div class="mb-3">
                <label for="email" class="form-label">Email address</label>
                <input type="email" class="form-control" id="email" name="email" placeholder="name@example.com">
            </div>
            <div class="mb-3">
                <label for="username" class="form-label">Username</label>
                <input type="text" class="form-control" id="username" name="username">
            </div>
            <div class="mb-3">
            <div class="mb-3">
                <label for="inputPassword5" class="form-label">Password</label>
                <input type="password" id="password" name="password" class="form-control" aria-describedby="passwordHelpBlock">
            </div>
            <button type="submit" class="btn btn-primary">Register</button>
        </form>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9f
  </body>
</html>
```

Figure 38: Register page HTML

```
6   // Route to render the registration page
7   router.get("/register", (req, res) => {
8     res.render("register");
9   });
10
11  // Route to handle user registration
12  router.post("/register", (req, res) => {
13    const { email, username, password } = req.body;
14
15    // Check if all fields are provided
16    if (!username || !password || !email) {
17      return res.send("All fields are required");
18    }
19
20    // Hash the password
21    const hashedPassword = bcrypt.hashSync(password, 8);
22
23    // Insert the new user into the users table
24    db.run(
25      `INSERT INTO users (username, pass_word) VALUES (?, ?)`,
26      [username, hashedPassword],
27      function (err) {
28        if (err) {
29          return res.send("Error registering user");
30        }
31
32        const userId = this.lastID;
33
34        // Insert the email into the emails table
35        db.run(
36          `INSERT INTO emails (user_id, email_address) VALUES (?, ?)`,
37          [userId, email],
38          function (err) {
39            if (err) {
40              return res.send("Error registering email");
41            }
42
43            res.redirect("/homepage");
44          }
45        );
46      }
47    );
48  });
```

Figure 39: Register route

3.10 Modules

Package Name	**Use Case**
express	Framework for developing node js applications
body-parser	used to access body of form data
sqlite3	to assist in the use of the sqlite3 database that we have used.
ejs	to dynamically display note data to the user
dotenv	to read env configuration file
bcrypt-nodejs	to encrypt passwords
session	for session handling

3.11 Endpoints

Endpoint	**Method**	**Description**
/	GET	renders the index page
/homepage	GET	renders home page after user has logged in
/users/login	GET	renders the login page
/users/register	GET	renders the register page
/users/login	POST	selects the relevant notes from the user information that was entered
/users/register	POST	process the form and enters user information into the database
/notes	GET	renders the create a note page
/notes/submitNote	POST	inserts a note entry into the database
/notes/edit/:id	GET	renders the note that the user

		wants to edit
notes/edit/:id	PUT	updates the note content in the database with the edited information
notes/delete/:id	DELETE	deletes the relevant note from the database

3.12 Database

We chose Sqlite3 for our web application since it is lightweight, easy to use, simple and provides file-based structure that fits our project needs. We did our midterm project using Sqlite3 and became familiar with its functionality. This familiarity allowed us to work more efficiently and build on our previous experience.

Unlike MySQL, MongoDB and Oracle which are designed for large-scale, high-currency projects, SQLite3 is ideal for smaller applications, providing a self-contained database without requiring a separate server or complex setup. Its minimal resource usage, zero configuration setup and portability allowed us to accelerate our app’s development and focus on core application features.

SQLite3's seamless integration with the tools that we were already using, particularly with Node.js and Express.js, allowed us to maintain a streamlined development process.
By choosing SQLite3, we miss out on some advanced features that MySQL, MongoDB, and Oracle offer, like stored procedures, advanced security, and complex query optimization. While we do not need these features for our current project, we might need them later as our application grows. Hence, when it happens, we can always switch to a more powerful database, but for now, SQLite3 is a solid choice for our current needs.

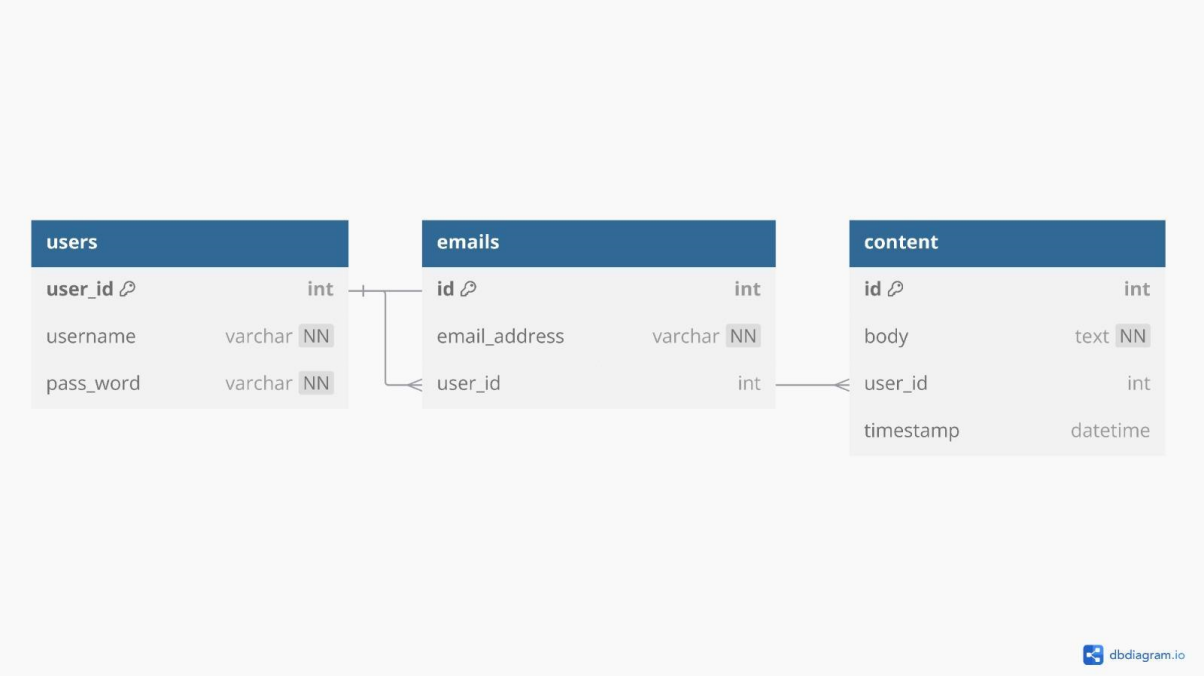


Figure 40: Entity Relationship Diagram

Our database consists of three tables namely users, emails and content (notes). The users table consists of a primary key user_id, username and pass_word. The emails table consists of primary key id, email_address and a foreign key user_id. The content table on the other hand consists of primary key id, body foreign key, user_id and timestamp. The types of relationship include a one-to-many relationship between the users and the emails. And a one-to-many relationship between the emails and content. Hence, this indirectly results in a one-to-many relationship between the users and the content as one user can create many notes/content.

```sql
-- This makes sure that foreign_key constraints are observed and that errors will be thrown for violations
PRAGMA foreign_keys=ON;

BEGIN TRANSACTION;

-- Create your tables with SQL commands here (watch out for slight syntactical differences with SQLite vs MySQL)

-- Create the users table if it doesn't already exist
CREATE TABLE IF NOT EXISTS users (
    user_id INTEGER PRIMARY KEY AUTOINCREMENT, -- Unique identifier for each user
    username TEXT NOT NULL UNIQUE, -- Username must be unique and not null
    pass_word TEXT NOT NULL -- Password must not be null
);

-- Create the emails table if it doesn't already exist
CREATE TABLE IF NOT EXISTS emails (
    id INTEGER PRIMARY KEY AUTOINCREMENT, -- Unique identifier for each email
    email_address TEXT NOT NULL UNIQUE, -- Email address must be unique and not null
    user_id  INTEGER, -- The user that the email account belongs to
    FOREIGN KEY (user_id) REFERENCES users(user_id) -- Foreign key constraint linking to the users table
);

-- Create the content table if it doesn't already exist
CREATE TABLE IF NOT EXISTS content (
    id INTEGER PRIMARY KEY AUTOINCREMENT, -- Unique identifier for each content entry
    body TEXT NOT NULL, -- Content body must not be null
    user_id  INTEGER, -- The user that the content belongs to
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP, -- Timestamp of when the content was created, defaults to current time
    FOREIGN KEY (user_id) REFERENCES users(user_id) -- Foreign key constraint linking to the users table
);

COMMIT;
```

Figure 41: Database Table in Code

3.13 Bootstrap

Other than using our own self written CSS, we used a built-in library, Bootstrap, in our application developing process. We used it for creating a “create a new note button”

Figure 42: Create note button

```
<div class="newNote">
  <a onclick="window.location.href='/notes' ">
    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" +52 class="bi bi-file-plus"
      viewBox="0 0 16 16">
      <path d="M8.5 6a.5.5 0 0 0-1 0v1.5H6a.5.5 0 0 0 0 1h1.5V10a.5.5 0 0 0 1 0V8.5H10a.5.5 0 0 0 0-1H8.5z" />
      <path
        d="M2 2a2 2 0 0 1 2-2h8a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H4a2 2 0 0 1-2-2zm10-1H4a1 1 0 0 0-1 1v12a1 1 0 0 0 1 1h8a1 1 0 0 0 1-1V2
    </svg>
  </a>
</div>
```

Figure 43: Create note button bootstrap code

We also used Bootstrap to create a note card used to display note details such as when it was created and the content of the note. The card also contains a delete and edit button to edit or delete that very note/card.

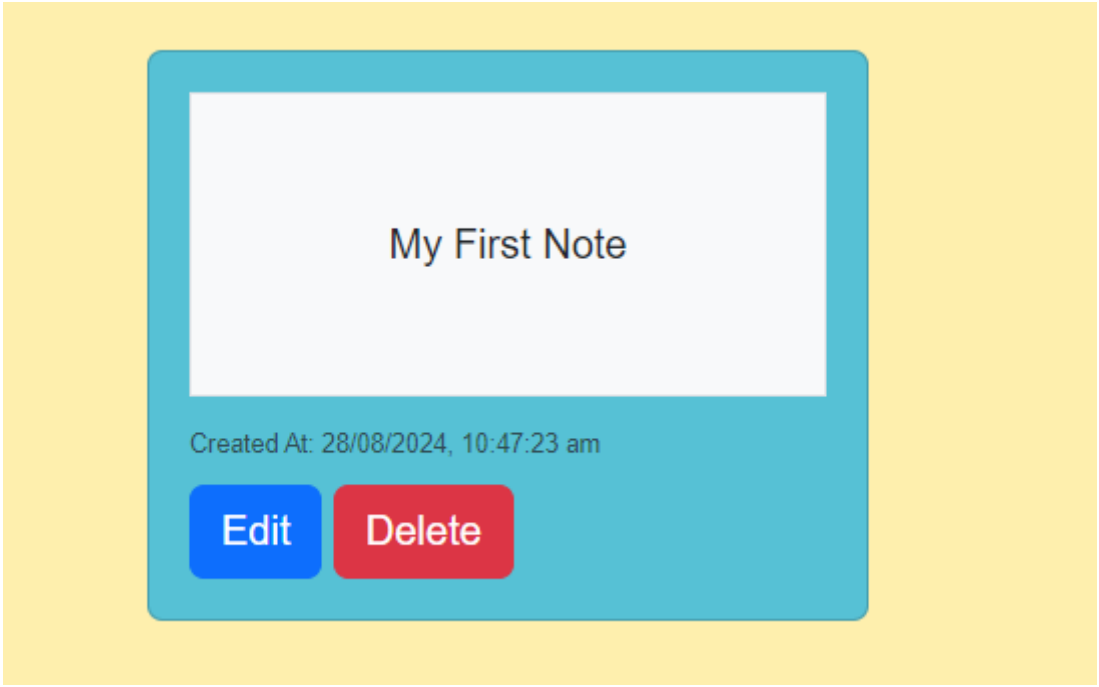


Figure 44: Note card look

```
<h2 class="existing">Existing Notes</h2>
<% let counter=1; %>
<div class="container">
  <div class="row">
    <% content.forEach(function(note) { %>
      <div class="col-md-4">
        <div class="card mb-4" style="width: 18rem; background-color: #56c1d5;">
          <div class="card-body">
            <p class="card-text border py-5 text-center bg-light" id="note-<%= counter %>"
              style="background-color: #feefad;">
              <%= note.body %>
            </p>
            <p class="card-subtitle mb-2 text-body-secondary" style="font-size: x-small;">
              Created At: <%= new Date(note.timestamp).toLocaleString() %>
            </p>
            <a href="/notes/edit/<%= note.id %>" class="btn btn-primary">Edit</a>
            <a href="#" class="btn btn-danger" onclick="deleteNote('<%= note.id %>')">Delete</a>
          </div>
        </div>
      </div>
      <% counter++; %>
    <% }); %>
  </div>
</div>
</div>
```

Figure 45: Note card bootstrap code

3.14 NodeJS

NodeJS is really good at handling many connections and real-time updates. Since we have worked with Node.js before, we were already comfortable with it, which made development smoother. Using JavaScript for both the front-end and back-end helped streamline our process, and the wide range of tools and libraries available through npm made it easy to get everything we needed.

While on the other hand, PHP is simple to set up, it is not as good at handling real-time features and high traffic. Django offers a lot of built-in tools and strong security, but it does not scale as well for heavy traffic. Ruby on Rails is excellent for quick development but can slow down with heavy use.

Node.js's event-driven design lets us handle multiple tasks at once without slowing things down, which is perfect for our needs. Its flexibility and performance, combined with our previous experience, made Node.js the best choice for building a responsive and efficient web app.

3.15 HTML/CSS

In our web app, we utilised HTML, CSS, and EJS because they were easy to learn and implement. HTML gave us the basic structure for our pages, CSS made them look nice and work well on different devices, and EJS helped us add dynamic content without too much hassle. It was a straightforward setup that fit our needs perfectly.

While React is excellent for building super interactive and dynamic user interfaces, it has a steeper learning curve. For our project, which was more about server-side rendering and simpler content, React felt like an overkill. So, we chose to keep things simple with HTML, CSS, and EJS, which helped us stay focused and avoid the extra complexity that React would have brought.

3.16 API Integration

Our unique feature in our note-taking app is the integration of Math tools. It is unique because most note-taking applications have little to no Maths tools. While many note-taking apps allow users to insert symbols and equations, they don't solve anything. We have implemented a scientific calculator, graphical calculator, and equation solver using APIs.

Initially, we planned to integrate four different types of APIs: Desmos API, Math.js API, Wolfram Alpha Full Results API, and the Symbolab - AI Math Calculator API. However, we ended up using only two.

3.16.1 Desmos API

For the **Graphic Calculator**, we used the Desmos API as planned. Desmos is well-known for its powerful graphing capabilities, making it an excellent choice for visualising mathematical functions and equations. Desmos was very easy to integrate into our web application requiring significantly less time.

For the **Scientific Calculator**, we initially planned to use the Math.js API. Math.js is a comprehensive library that offers a wide range of mathematical functions and capabilities. However, we ended up integrating an online web calculator using iframe tag instead.

3.16.2 Desmos API and Wolfram Alpha Full Results API

For the **Equation Solver**, we chose the Wolfram Alpha Full Results API. Wolfram Alpha, renowned for its computational intelligence and ability to solve complex equations, was a perfect fit for this feature.

Although we had initially planned to use the Symbolab - AI Math Calculator API for all maths features, including the Graphic Calculator, Scientific Calculator, and Equation Solver, we chose to rely on Desmos and Wolfram for these maths features.

We chose Desmos and Wolfram over Symbolab because they both excel in their specialised areas. While Symbolab provides excellent step-by-step solutions for mathematical problems, it does not offer the same level of graphical capabilities as Desmos, which is renowned for its powerful graphing calculator. Additionally, Wolfram Alpha provides comprehensive visualisations and graphical results, making it a more versatile tool for both graphical and scientific calculations. This graphical capability is crucial for a more interactive and visual learning experience.

3.16.3 Implementation

Desmos: In the code, the Scientific Calculator and Graphical Calculator from Desmos are implemented using embedded iframes in the HTML files. The iframes source is set to the respective Desmos calculator URLs. These iframes load the respective Desmos calculators directly from their website Desmos API v1.9 documentation . The calculators are fully functional within the iframes, allowing users to interact with them as they would on the Desmos website.

Wolfram Alpha: The Wolfram Alpha equation solver is implemented using the Wolfram Alpha API, which requires a valid app ID to function. The app ID is stored in the WOLFRAM_APP_ID environment variable.

In order to get the App ID, steps from https://github.com/Genbox/WolframAlpha were taken:

First you need to get a Wolfram|Alpha AppId from their website.

1. Go to https://developer.wolframalpha.com/portal/signup.html and create an account if you don't already have one.
2. Go to https://developer.wolframalpha.com/portal/myapps/index.html and click "Get an AppID"
3. Just follow their wizard and then you will have an AppID in the format: XXXXXX-XXXXXXXXXX

4. EVALUATION

4.1 Application Evaluation

The key features that were initially highlighted in our proposal, we were successfully able to incorporate those into *Mathote*. One of the most notable achievements is the integration of the Maths features: the Scientific Calculator, Graphical Calculator, and Equation Solver. These tools are fully functional and provide significant value to users who need to perform complex calculations and solve equations. The scientific calculator allows users to carry out advanced mathematical operations, while the graphical calculator helps in visualizing functions and plotting graphs. The equation solver is particularly useful for students and professionals who need to solve algebraic equations quickly and accurately. Together, these tools make Mathote a powerful application for anyone involved in scientific, engineering, or mathematical work.

Another important achievement is the inclusion of Basic Text Formatting options. Users can format their notes with bold, italic, and underline options, which helps in emphasizing important points and organizing information more effectively. Although these formatting

options are basic, they still provide a level of customization that enhances the note-taking experience. Users can highlight key concepts, differentiate between different types of information, and create a more readable and structured set of notes. This feature, while simple, adds a layer of functionality that is essential for effective note-taking.

Although *Mathote* caters to the initially highlighted features, we were unable to make it a full-fledged note taking application as the possibilities are endless. Several features can be further implemented in it however, given the time constraint, we realised that it would be beyond the scope and would not be technically feasible. Except for Bold, Italic and Underline, *Mathote* does not account for rich text formatting such as font colour, font size, font highlighting, font type, etc. It lacks what helps structure and format notes and has a minimal toolbar with the Scientific Calculator, Graphical calculator and Equation Solver Buttons that work. Besides, with note taking application having accessibility features such as audio-to-text conversion feature, screen reader software, varying font size options etc. [12], *Mathote* currently does not cater to these features, limiting its user base. The current version of the app also does not allow users to upload and store documents like PDFs, Word files, or spreadsheets. This can be a real inconvenience for users who need to reference or annotate documents, forcing them to juggle between different apps to manage their documents, disrupting their workflow and making things less efficient. By centralizing everything, a document upload function would make note-taking easier and more organized for users.

While most of the note taking apps have integrated cameras, *Mathote* currently does not have a built-in feature yet to take and embed photos directly into notes. This can be a hassle for users who need to quickly capture visual information, like diagrams, whiteboards, or documents during meetings or lectures. Instead of taking a photo and instantly adding it to their notes, users would have to rely on other apps to take pictures and then manually upload them (a feature that is already absent in our app). While our app is already quite useful, adding an image capture feature would be a great improvement, making it easier for users to seamlessly include visual elements in their notes and enhancing the overall functionality.

Mathote does not permit multiple users for a single note. It creates a private user session. In a single session, multiple users cannot collaborate together and create notes. A single user id creates, edits, and deletes each note inside a single session. Furthermore, unlike traditional note taking applications, *Mathote* does not allow users to save notes on their desktop; the only place the note is saved is the database that we created in the data tier, thus highlighting another potential area for improvement. The absence of the voice-to-text transformation feature, which is particularly beneficial for users with physical disabilities or those who find typing cumbersome, is another pertinent feature that can be integrated in the future. According to a study on voice note-taking applications, integrating this feature can significantly enhance the usability and accessibility of note-taking applications [13]. One major benefit of this software is reducing pencil lead usage. With the speech-to-text function, students can listen to lectures while the software transcribes notes for them. This saves time and effort, allowing students to focus more on the lecture and retain more information [13].

4.2 Technical Limitations

4.2.1 Lack of security measures implementation

Although we have taken steps to secure our application by encrypting passwords before storing them in the database, our security measures still have some shortcomings. Other crucial aspects like secure data transmission (such as HTTPS), thorough input validation, or robust protection against common vulnerabilities like SQL injection and cross-site scripting (XSS) are not fully implemented. These are essential for safeguarding user data and ensuring overall application security. While our encryption efforts do add a layer of protection, the absence of these additional security practices leaves our application potentially exposed to various risks and attacks that could compromise user information and application integrity.

4.2.2 Downsides of using the languages, frameworks and APIs

JavaScript is a versatile and widely-used language, but it has its drawbacks. One major issue is its client-side security vulnerabilities, as the code is executed on the user's browser, making it susceptible to attacks. To improve this, if we were to do the project over again with no time constraints, we would implement Content Security Policy (CSP) to prevent cross-site scripting (XSS) attacks, regularly update libraries and frameworks to patch security vulnerabilities, and follow secure coding practices and input validation. Additionally, JavaScript can be inconsistent across different browsers, which may lead to unexpected behaviour and bugs [14]. To address this, we can use polyfills and transpilers like Babel to ensure compatibility across different browsers, and test your application on multiple browsers to identify and fix inconsistencies.

SQL is a powerful language for managing relational databases, but it can be complex to learn and use effectively. If we could do the project over again, we could simplify database interactions with ORM tools like Sequelize. The rigid schema can be managed with schema migration tools, and for flexible data, consider NoSQL databases. The rigid schema structure can make it difficult to adapt to changing requirements, and scaling SQL databases can be challenging as the size of the data grows [15]. Additionally, SQL databases may not be the best choice for handling unstructured or semi-structured data [16]. To handle large data, we could use sharding and indexing.

Express.js is a minimalist web framework for Node.js, but its unopinionated nature can lead to inconsistent codebases and a lack of clear structure [17]. This can make it difficult to maintain and scale the application, especially as the project grows. Additionally, Express.js may not provide all the built-in features that other more opinionated frameworks offer, requiring developers to rely on third-party middleware. Thus, if we were to redo the project, for more features, we would use third-party middleware or consider frameworks like NestJS for larger projects.

EJS (Embedded JavaScript) is a templating engine that allows you to generate HTML with JavaScript, but it has some downsides. One major issue is the mixing of logic with presentation, which can make the code harder to read and maintain. Additionally, EJS may not have all the advanced features that other templating engines offer, which could limit its usefulness in more complex projects [18]. EJS mixes logic with presentation, making it hard to maintain. If we were to rework, we would separate logic using helper functions and partials, or use frameworks like React for better separation. For advanced features, we could consider other templating engines like Handlebars.

SQLite is a lightweight and easy-to-use database, but it has limitations in terms of scalability and concurrency. It may not be suitable for handling large-scale applications or multiple concurrent write operations. Additionally, SQLite's performance can degrade as the database size increases, making it less suitable for projects with large amounts of data [19]. If we could start over, we would switch to databases like PostgreSQL for larger projects. We could optimize queries and use indexing to maintain performance.

Wolfram Alpha API is a powerful tool for accessing computational knowledge, but it can be expensive to use, especially for projects with high API call requirements. The cost can be a significant barrier for university projects with limited budgets [20]. Additionally, the API may have limitations in terms of the number of free calls available, which could restrict its usage [21]. To avoid potential issues like these, we could optimize API usage to reduce calls and explore alternative APIs if we could start our project again. And use the free tier for development and plan for costs in production.

Desmos API is great for embedding interactive math into web applications, but it can have performance and memory implications if multiple instances are created. Additionally, the API may not provide all the features needed for more complex mathematical visualizations, which could limit its usefulness in certain projects [22]. if we could start over, we could optimize instance creation and use profiling tools to improve performance. For more features, combine Desmos with other libraries.

4.3 Team Experience

This project was our team’s first experience with agile development, and it brought its own set of challenges. We were all relatively new to agile methodologies, which meant that getting up to speed on things like sprint planning, task management, and iterative development took some time. We had to do a lot of research and relied heavily on example reports to understand and apply these practices effectively. Our unfamiliarity with agile processes occasionally caused confusion and slowed down our progress. The learning curve was steep, and there were moments when our lack of experience in agile development impacted our efficiency and teamwork but nonetheless, we made significant strides.

4.3.1 Back-End Implementation

Main Learnings:
Amidst the duration of the project, the team collectively learnt a number of things. We learnt how to properly implement APIs such as Desmos, iframe tag, Wolfram etc. into our projects to create the various different mathematical features of the application, furthermore, we gained more in-depth understanding of the way the many different note-taking applications currently existing on the market are designed and the multitude of back-end processes that are going on at any given time to keep the application running.

In addition, we learnt how to weigh the pros and cons of different tools and APIs to better understand which better suited our wants and needs. For example, when we abandoned Symbolab and Math.js in favour of other more suitable tools. This is something that will be imperative to all of us in our future as software developers when we are met with a decision to choose between different items in our toolbox as we construct a functioning web application.

Results and Downfalls:
As a result of this, we were able to create an intuitive application that has in-built scientific and graphing calculators as well as a proper equation solver, that is bound to prove helpful for any existing or aspiring mathematics student. However, there were also some drawbacks and obstacles that came our way during the back-end implementation of the software. The main one being the fact that our application was more complex than we realised. The scope of this project's implementation was far beyond any work or project we had done prior, and that caused some hindrances. There were unforeseen delays, and times of confusion, but thankfully, due to our careful pre-emptive planning and the support of the midterm coursework as a blueprint, we were able to overcome these challenges satisfyingly.

4.3.2 Front-End Implementation:

Main Learnings:
Although we prioritised familiarity when it came to the front-end implementation of the software, we were still able to learn a wide array of new things. For example, we learnt how to use Node.js, SQLite3, Express.js, HTML, CSS and Bootstrap simultaneously to produce a harmonious result. Despite our prior experience with some of these libraries and languages, it was still a new and challenging approach to use them all efficiently in conjunction. Since the scope of this project was larger than ever before, it polished our existing skills and made us more confident in our abilities.

Furthermore, as a key learning component of the agile software projects module was to create an intuitive, user friendly and structured design, we really made our best efforts to keep this in mind at all times unlike ever before. Though we had previously worked with these tools, it was still a new experience and learning outcome to use them in such a way such that the end result is pleasant for the user.

Result:

The results prove this idea as our application is very seamless, systematic and beginner friendly without compromising on functionality. We also added some additional quality-of-life features such as the note card function using Bootstrap, which provides a brief summary of the existing note pages along with some further details.

Using SQLite3 we were also able to implement an adequate database that stored usernames, passwords and notes, this is what made the project usable for day to day users by associating notes to user profiles.

HTML, CSS and JavaScript libraries made our web application look professional and appealing to the eye. Although we were fortunate to have some experience in these areas, implementing them in a full-fledged web application was still an intriguing challenge.

4.3.3 Testing Phase

Main Learnings:
Testing phase was a bit difficult for us as we hit a roadblock deciding on the questions for the survey. We wanted to get as much information from the user as possible without asking them for details and making them bored.

To that effect, we first outlined the key features of our application in terms of design and functionality. This included the landing page, mathematical features, note-taking features etc. We then generated a survey that tackled those exact components. After that, we simply created straightforward questions that would be easy to answer for any maths student about each and every individual aspect of the application. We did this by dividing the survey into three parts, the first part of the survey asked about the design and navigation of the application, the next part focused on the functionality, specifically the note-taking, scientific calculator, graphing calculator and equations solver and the final part asked about the ease of use as well as the users' overall thoughts about the utility of *Mathote*.

This really helped us gauge the overall opinion that others had on our application thus far, and whether or not they thought our application had any use in real-life scenarios. The responses we received were both supportive, yet critical in some crucial aspects, hence, helping us understand that we were making something worthwhile and which aspect exactly needed more work and eventually, this led to us creating a more well-rounded, versatile application.

The Heuristic Evaluation survey was filled by the team members and we made sure to remain unbiased when answering the questions. The team after reading the questionnaire and answering the questions realised the limitations of *Mathote* and inferred from them the solutions that could be implemented in future work.

We learned a great deal about the importance of user review and feedback and discovered that at times, the developer may ignore certain things that impact user experience because as

developers, we sometimes fail to see the application from the user's perspective. So, user testing is one way we can see our application from the eyes of our users.

Results:
We received major feedback from the user testing survey and worked accordingly to make improvements to *Mathote.* We had finished conducting iteration 2 based on the changes made as a result of iteration 1 and received positive feedback from the users.

The results as received from the Heuristic Evaluation survey still need to be worked on and our application improved based on those responses. In the future, we aim to improve *Mathote* keeping in mind the results.

4.3.4 API Integration and Development Tools

When working on *Mathote*, integrating APIs turned out to be more complicated than expected. Our team's experience was primarily with HTML and CSS, and we had limited exposure to backend technologies and API integration. This lack of familiarity meant that we had to spend extra time learning about how APIs work, troubleshooting integration issues, and figuring out the best practices for connecting our app to external services. While we eventually managed to get everything functioning correctly, the process was not as smooth as it could have been with more experience. The extra effort required for API integration highlighted the gap in our skills and showed us where we need to improve for future projects.

4.4 Future Prospects

Despite creating what was proposed in the proposal report, we acknowledge that certain necessary functionalities, which are essential for the full realisation of our vision, have been deferred for future implementation. Our next aim is to improve and refine the application and focus on addressing the limitations mentioned above. There is immense potential to scale this application and make it a fully functional mathematics integrating note taking application - *Mathote* - that lives up to its name.

We are committed to incorporating these additional features in future updates, ensuring that our application evolves in line with both our original vision and the valuable feedback we received from our users.

In the future, our note-taking app could evolve to include a **collaboration feature**, enabling users to work together in real-time, enhancing teamwork and productivity. We could integrate **LaTeX**, allowing users to create and format complex mathematical expressions seamlessly. By adding **accessibility features**, we could ensure that the app is usable by everyone, including those with disabilities. An **enhanced and rich text editing space** along with **advanced formatting tools** could provide users with greater flexibility and creativity in their note-taking. Incorporating **AI-powered speech-to-text** functionality could make it easier for users to convert spoken words into written notes, saving time and effort. Finally, offering the option to **download saved notes** could provide users with the convenience of accessing their notes

offline and sharing them across different platforms. These future enhancements could significantly elevate the app's functionality and user experience.

5. SUMMARY

This journey commenced with our hands wrapped around our heads trying to figure out what it is that we plan on making. Having gone through various researches, market analysis, reviewing literature, we found it difficult to ideate what we wanted to make and what is currently a limitation in the market. Whether it was the fear of not having enough skills to make something along the lines of the proposed application or the fear of the idea not being good enough, it was something that we really struggled with.

Despite the setbacks - feeling like our skills are not on par with what we ideated hence the delays in the developmental process - and a strenuous journey, we conclude it with a successful *Mathote* prototype, laying the groundworks for future work and scaling it. It is fairly safe to say that we were successful in undertaking our largest project yet with the most complex implementation and doing it in a way that did not leave any of us displeased. In particular, our decision for everyone to work as full-stack developers and not split up the team in back-end and front-end was a good decision since we were able to finish each section fairly. Otherwise, as highlighted above, since the backend aspect was more challenging, it is very likely that we could have ended up in a situation where the backend team was overworked leading to overall inefficiency. This could have led to a difficulty in meeting the project deadline.

Throughout this project, all team members learnt a great deal of things from new skills and concepts to the polishing of existing ones. Having successfully laid the groundwork and blueprint in our midterm report, we succeeded in achieving all objectives through dedication and consistency and *Mathote* is the evidence of that.

6. REFERENCES

[1] N. C. Helen Willacy, “Making Mathematics Learning More Engaging for Students in Health Schools through the Use of Apps,” 19 April 2017. [Online]. Available: https://www.mdpi.com/2227-7102/7/2/48. [Accessed 20 June 2024].

[2] M. T. &. E. L. Andrea Bunt, “Challenges and Opportunities for Mathematics Software in Expert Problem Solving,” 11 March 2013. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/07370024.2012.697020. [Accessed 20 June 2024].

[3] D. L. T. E. Lisa-Marie Wienecke, “Taking notes as a strategy for solving reality-based tasks in mathematics,” N/A, Lüneburg, 2023.

[4] M. A. Freitag, “Note-Taking Practices of Students in College Mathematics,” 31 August 2019. [Online]. Available: https://link.springer.com/article/10.1007/s40753-019-00099-0 . [Accessed 21 June 2024].

[5] J. T. &. M. T. Hans-Georg Weigand, “Mathematics teaching, learning, and assessment in the digital age,” 10 July 2024. [Online]. Available: https://link.springer.com/article/10.1007/s11858-024-01612-9 . [Accessed 25 July 2024].

[6] J. Nielsen, “10 Usability Heuristics for User Interface Design,” 30 Jan 2024. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed 24 July 2024].

[7] A. Fard, “Newbie Heuristic Evaluation Mistakes To Avoid,” 18 Jan 2022. [Online]. Available: https://uxbooth.com/articles/newbie-heuristic-evaluation-mistakes-to-avoid/. [Accessed 24 July 2024].

[8] B. Whittle, “3 Colours that Improve Concentration & Productivity,” 5 December 2014. [Online]. Available: https://www.barker-whittle.com.au/blog/3-colours-improve-%20concentration-productivity. [Accessed 30 June 2024].

[9] H. Genever, “4 Colors That Give You an Unexpected Productivity Boost,” Redbooth, N/A N/A N/A. [Online]. Available: https://redbooth.com/hub/colors-unexpected-productivity-boost/. [Accessed 30 June 2024].

[10] N/A, “HTML Reference - Browser Support,” N/A N/A N/A. [Online]. Available: https://www.w3schools.com/tags/ref_html_browsersupport.asp. [Accessed 2 September 2024].

[11] N/A, “JavaScript,” Developer Mozilla, N/A N/A N/A. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Accessed 3 September 2024].

[12] N/A, “Accessibility features for Notes®,” HCLSoftware, N/A N/A N/A. [Online]. Available: https://help.hcl-software.com/notes/14.0.0/client/acc_essibility_c.html. [Accessed 5 September 2024].

[13] J. W. Casson Qin, “AN INTELLIGENT MOBILE APPLICATION TO ASSIST IN TAKING MATHEMATICAL NOTES USING SPEECH RECOGNITION AND NATURAL LANGUAGE PROCESSING,” [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://csitcp.org/paper/13/139csit19.pdf. [Accessed 3 September 2024].

[14] freeCodeCamp, “The Advantages and Disadvantages of JavaScript,” freeCodeCamp, 5 December 2019. [Online]. Available: https://www.freecodecamp.org/news/the-advantages-and-disadvantages-of-javascript/. [Accessed 5 September 2024].

[15] simranjenny84, “Advantages and Disadvantages of SQL,” GeeksforGeeks, 2 May 2023. [Online]. Available: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-sql/. [Accessed 5 September 2024].

[16] N/A, “Advantages and disadvantages of using SQL in data analysis practices,” dataflake, N/A N/A N/A. [Online]. Available: https://www.dataflake.co/data-analytics/advantages-and-disadvantages-of-using-sql-in-data-analysis-practices. [Accessed 5 September 2024].

[17] Z. Jiang, “Express.js: The dangers of unopinionated frameworks and best practices for building web applications.,” DEV Community, 12 June 2023. [Online]. Available: https://dev.to/weipengjiang/expressjs-the-dangers-of-unopinionated-frameworks-and-best-practices-for-building-web-applications-3g93. [Accessed 5 September 2024].

[18] apj752003, “What are the pros & cons of EJS for Node.js Templating ?,” GeeksforGeeks, 15 April 2024. [Online]. Available: https://www.geeksforgeeks.org/what-are-the-pros-cons-of-ejs-for-node-js-templating/. [Accessed 5 September 2024].

[19] S. Ravoof, “Understanding Database Technology: SQLite vs MySQL,” Kinsta, 2 November 2022. [Online]. Available: https://kinsta.com/blog/sqlite-vs-mysql/. [Accessed 5 September 2024].

[20] r/math, “Are there any alternatives to the Wolfram API that are either cheaper or has way more free calls?,” Reddit, N/A N/A 2017. [Online]. Available: https://www.reddit.com/r/math/comments/6mfbba/are_there_any_alternatives_to_the_wolfram_api/?rdt=34389. [Accessed 5 September 2024].

[21] r/programming, “Wolfram Alpha API Now Free and Open to All,” Reddit, N/A N/A 2010. [Online]. Available: https://www.reddit.com/r/programming/comments/f6z56/wolfram_alpha_api_now_free_and_open_to_all/. [Accessed 5 September 2024].

[22] N/A, “Quick Start,” N/A N/A N/A. [Online]. Available: https://www.desmos.com/api/v1.8/docs/geometry.html. [Accessed 5 September 2024].

7. APPENDICES

7.1 Appendix A: User Testing Survey

Q-1: How aesthetically pleasing did you find "Mathote's" landing page?

- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5

Q-2: How easy do you think it was to navigate and use the application?

- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5

Q-3: How easy do you think it was to add, delete and edit a note?

- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5

Q-4: How accurate and intuitive did you find it was to use the scientific calculator?

- 1
- 2
- 3
- 4
- 5

Q-5: How accurate and intuitive did you find it was to use the Graphing calculator?

- 1
- 2
- 3
- 4
- 5

Q-6: How accurate and intuitive did you find it was to use the Equations Solver?

- 1
- 2
- 3
- 4
- 5

Q-7: In your opinion, do you think it was efficient and convenient to use both the note-taking and mathematical features of the application simultaneously?

- ○ Very efficient and convenient
- ○ Somewhat efficient and convenient
- ○ Efficient, but not convenient
- ○ Convenient, but not efficient
- ○ Neither, I don't see the point of this application.

Q-8: Was the app smooth and responsive when switching from one page or feature to another?

- ○ Yes, adequately responsive
- ○ I rarely noticed some stutters
- ○ Somewhere in between
- ○ I frequently noticed stutters.
- ○ The application was unusable.

Q-9: How easy did you think it was to login into and create an account on the site?

- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5

Q-10: Which feature of "Mathote" did you find the most helpful?

- ○ Note-taking
- ○ Scientific Calculator
- ○ Graphing Calculator
- ○ Equations Solver
- ○ Other...

Q-11: Finally, seeing its current functionality, do you ever see yourself using "Mathote"?

- ○ Yes, I can see myself using it frequently.
- ○ I can see myself using it occassionally
- ○ Maybe, if it makes improvements/fixes some bugs.
- ○ Never. Not interested.

7.2 Appendix B: Heuristic Evaluation

1. The application **keeps users informed about its status** *appropriately* and *promptly*.

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5

2. The application **shows information in ways users understand** from how the *real world* operates, and in *the users' language*.

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5

3. This application **offers user control** and let him/her undo errors *easily*.

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5

4. This application is **consistent** so users aren't confused over what different words, icons, etc. mean

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5

---

5. This application **prevents errors** – a system should either *avoid conditions where errors arise* or *warn users before they take risky actions* (e.g., "Are you sure you want to do this?" messages)

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5

6. This application has **visible information, instructions, etc. to let users recognize options, actions, etc.** instead of forcing them to rely on memory.

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5

7. This application is Flexible so experienced users find faster ways to attain goals.

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5

8. This application has **no clutter**, containing only relevant information for current tasks

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5

9. This application **provides plain-language help** regarding errors and solutions.

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5

10. The application **lists concise steps in lean, searchable documentation** for overcoming problems

- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5