Problem Statement and Goals
Flow

Team 9, min-cut
Ethan Patterson
Hussain Muhammed
Jeffrey Doan
Kevin Zhu
Chengze Zhao

Table 1: Revision History

Date	Developer(s)	Change
Date1	Name(s)	Description of changes
Date2	Name(s)	Description of changes
...	...	...

1 Problem Statement

1.1 Problem

The speed of thought should be the limiting factor in efficiency when typing or taking notes. Editors like Vim and Emacs follow this philosophy for text through keyboard driven shortcuts and commands. However, users can be limited in applying a similar approach to creating simple diagrams and drawings alongside text. This would be particularly useful when creating documents that need to be completed in a timely manner, such as taking notes in a lecture. For example, disciplines like engineering require taking notes with a combination of text and diagrams, whether it be simple tables, state machines, or circuits.

Familiar solutions exist for taking notes involving both text and diagrams, but each have some limitations:

- **Taking notes with pen and paper/tablet.** While this gives the freedom of drawing diagrams the way you want, you are forced to print text rather than type it. There is also no option for copy/pasting repetitive parts of a diagram. Writing out longer thoughts or sentences can take

a while, and potentially lead to falling behind. Correcting mistakes or repositioning diagrams/text can also be timely.

- **Taking notes with your computer and keyboard.** While this can allow for faster speeds with typing, options for inserting diagrams are limited and/or slow. Existing diagram description domain-specific languages exist such as (PlantUML, Mermaid.js). However, they focus specifically and only on diagram creation, which requires definition, rendering, and then insertion into your note document. Beyond this, manipulating text rather than the visual objects themselves can lead to inefficiencies when forgetting syntax and lack immediate feedback from changes. Other options embedded in OneNote or Draw.io follow a mouse-based workflow, which can be cumbersome for various reasons (e.g. navigating to the menu for a given shape you would like to select).

- **Hybrid approach.** It is possible to use both of the above approaches together, switching between drawing and typing. This carries its own cognitive load for swapping context between two media. Swapping between a computer and pencil and paper means the notes will be stored in different places. Swapping between a computer and drawing tablet also means purchasing another device.

The overarching problem is that no unified digital platform exists for efficiently taking notes involving text and diagrams using a keyboard-driven approach.

1.2 Inputs and Outputs

1.2.1 Inputs

- Combinations of keystrokes that:
  - Spell out defined commands/shortcuts
  - Spell out note text for display
- User defined keybinds

1.2.2 Outputs

- Displayed text on the screen.
- Displayed diagrams/geometry on the screen.
- Content (i.e. text and diagrams) on screen that can be saved in a file and re-opened to display the same content

1.3 Stakeholders

Broadly, the stakeholders for this project will all fall into a group of more technically savvy individuals who are likely comfortable using keyboard shortcuts in other familiar software programs. The target users should be able to memorize a combination of shortcuts and commands to align with the keyboard based usage.

- People who use keyboard based workflows and prefer not to use their mouse to take notes that involve simple diagrams.
- People who do not have access to a drawing tablet, but would like to take digital notes involving diagrams in an efficient way.
- People who often take notes on a computer and see value in learning a new system to improve their note taking efficiency.

1.4 Environment

1.4.1 Hardware Environment

The hardware environment will be a personal computer running a modern operating system (i.e. Windows, MacOS, or Linux). The computer will have a keyboard and mouse for user input.

1.4.2 Software Environment

The software environment will be a desktop application.

2 Goals

- Allow users to create, edit, and save documents involving text and diagrams using only their keyboard.
- Allow users to generate simple custom geometry involving lines and predefined shapes (e.g. circle, rectangle, triangle).
- Allow users to directly interact with displayed geometry (i.e. not the underlying text representation of the geometry and no required text to visual render step).
- Allow users to save custom geometry as reusable commands/shortcuts for future use.
- Provides the user the potential to take notes involving diagrams more efficiently (time-wise) than other individual/combined note taking applications.

3 Stretch Goals

- Provides the user the potential to take notes involving diagrams more efficiently than by hand on paper/drawing tablet.
- Ability to export documents as images and/or PDFs.
- Support for collaborative note taking.
- Support for LaTeX math rendering.
- Support for Vim style keybindings when editing text.

4 Extras

- Usability testing
- User manual

Appendix — Reflection

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

    - Chengze - I came up with many ideas for on stakeholders and goals based on our initial problem statement. And I was able to clearly articulate these ideas in the document.
    - Ethan - Through filling out the problem statement and goals sections, I gained a more concrete idea of the requirements for our project. I think it helped flesh out the converging goals of improved efficiency and support for keyboard driven workflows. This was particularly helpful in the stakeholders section, which is a crucial section for creating goals and usability tests going forward. Originally we had broad ideas for stakeholders like "Students" and "Professionals", but from the brainstorming done in other sections we were able to narrow them down to more precise descriptions that we will design the product for.
    - Hussain - While writing the Proof of Concept Demonstration Plan, I gained a key understanding of the risks involved in the project. This not only helped me understand the challenges we may face, but also allowed me to bring up the risks to my team members and the TA. This further lead to in depth discussions on how we can mitigate these risks. Confirming an effective proof of concept demonstration plan was a key milestone for our team, giving us confidence in the feasibility of our project.
    - Jeff - Our first meeting discussing how we wanted to assign roles and tasks for this deliverable went very well. We were able to easily distribute tasks to members without any disagreements.
    - Kevin - Our group was able to easily assign everyone their work and get the work done when needed. Additionaly, the team communicated effectively meaning that everyone was participating.

2. What pain points did you experience during this deliverable, and how did you resolve them?

- Chengze - I found it difficult to limit the number of goals of the project to be within the appropriate number of main goals suggested. I resolved this by discussing with my TA and my teammates, we agreed to focus on the core functionalities of the project and leave some features as stretch goals.
- Ethan - It was difficult to come up with "selling point" goals and describe them in a concise way. We felt that a lot of our goals sound similar at first glance, but they subtly describe different goals for what the system would provide/look like. In general it was also a challenge to remind myself to focus on "what" rather than "how" for our problem statement and goals.
- Hussain - One pain point I experienced was writing the expected technologies section. I was initially confused as to how specific I should be when listing the technologies. A simple confirmation from the TA helped me understand that I should list the specific technologies I expect to use, but also note that these may change as the project progresses.
- Jeff - During this deliverable, I found it difficult to navigate and install all the tools required for this capstone. Additionally, I had issues running the make file and couldn't troubleshoot it. In order to resolve this issue, I asked my team members who had already finished configuring their environment for assistance.
- Kevin - One pain point of this deliverable was understanding the GitHub merge checks. As I had never used this type of code managment before, we had issues merging when needed.

3. How did you and your team adjust the scope of your goals to ensure they are suitable for a Capstone project (not overly ambitious but also of appropriate complexity for a senior design project)?

    - We discussed the goals and stretch goals as a team and prioritized the core functionalities of the project. Instead of focusing on the features/goals that satisfy all the stakeholders, we decided to go with the functionalities that we desire to have in the project. And for those goals that are challenging to implement, we put them as stretch goals so that we can focus on the main goals first, and then work on the stretch goals if we have time.