Flow

Requirements Standard Plan

Hussain Muhammed, Jeffrey Doan, Kevin Zhu, Chengze Zhao, Ethan Patterson

Version 0, 2025-10-11

Table of Contents

Academic Integrity Disclaimer

We would like to acknowledge that as a dedicated students of McMaster University, we have thoroughly read and comprehended the Academic Integrity Policy published by the university. We are committed to upholding the principles of academic honesty and integrity in all aspects of our educational journey. We understand the importance of acknowledging the work and ideas of others, and we pledge to ensure that all our academic endeavors are conducted with the utmost originality and compliance with the university's policy.

We affirm that the content presented in this document is entirely our own, and any external sources used have been appropriately cited and referenced.

Hussain Muhammed

As I submit my work, I, **Hussain Muhammed**, take full responsibility for the integrity of my work and promise to avoid any form of plagiarism, cheating, or dishonest behavior. This acknowledgment serves as a testament to my dedication to academic excellence and the fostering of a trustworthy academic community at McMaster University.

Jeffrey Doan

As I submit my work, I, **Jeffrey Doan**, take full responsibility for the integrity of my work and promise to avoid any form of plagiarism, cheating, or dishonest behavior. This acknowledgment serves as a testament to my dedication to academic excellence and the fostering of a trustworthy academic community at McMaster University.

Kevin Zhu

As I submit my work, I, **Kevin Zhu**, take full responsibility for the integrity of my work and promise to avoid any form of plagiarism, cheating, or dishonest behavior. This acknowledgment serves as a testament to my dedication to academic excellence and the fostering of a trustworthy academic community at McMaster University.

Chengze Zhao

As I submit my work, I, **Chengze Zhao**, take full responsibility for the integrity of my work and promise to avoid any form of plagiarism, cheating, or dishonest behavior. This acknowledgment serves as a testament to my dedication to academic excellence and the fostering of a trustworthy academic community at McMaster University.

Ethan Patterson

As I submit my work, I, **Ethan Patterson**, take full responsibility for the integrity of my work and promise to avoid any form of plagiarism, cheating, or dishonest behavior. This acknowledgment serves as a testament to my dedication to academic excellence and the fostering of a trustworthy academic community at McMaster University.

Control Information

Version	Delivery		Info	
	Deadline	*Delivered*	Developers	Change
V1	Oct 10, 2025	Oct 10, 2025	Hussain, Oliver, Jeff, Kevin, Ethan	Initial release
V2				
V3				

(G) Goals

(G.1) Context and Overall Objectives

Flow is designed to address gaps in existing note-taking systems, which lack efficient methods for integrating text and diagrams through a unified keyboard-driven workflow. Current solutions impose trade-offs in speed, flexibility, and workflow continuity, as explored in G.2.

The goal of Flow is to create an intuitive system that allows for keyboard-driven creation and editing of notes involving text and diagrams. Although there is an expectation that there will be a learning curve to become proficient with the system, the intended benefit is that the user will be able to create and edit notes more quickly than with existing systems.

(G.2) Current situation

Flow arises from the fact that students, especially in technical disciplines such as engineering, often need to create notes containing both structured text and quick sketches of diagrams like state machines, circuits, or tables. Often times these notes also need to be created in a timely manner, as lectures can move quickly.

There are many existing options for taking notes, each with their own trade-offs:

- Pen and paper
  - Not easily editable or shareable
  - Handwriting may be slower than typing
- Tablets with stylus support (e.g. iPad with Apple Pencil)
  - Can be expensive
  - Handwriting may be slower than typing
- Traditional text editors (e.g. Microsoft Word, Google Docs)
  - Limited support for creating diagrams within the app
- Mouse-driven editors (e.g. Draw.io, OneNote)
  - Can be slow to navigate menus for specific geometry
  - Limited keyboard shortcut support for creating and editing geometry
  - May require exporting and importing images into a separate text editor for note-taking (e.g. Draw.io)
- Text-based diagram languages (e.g. Mermaid, PlantUML)
  - Separate definition from rendering, forcing users to manage an inefficient write-compile-insert cycle
  - Requires integration with another text editor for note-taking text
- Document markup languages (e.g. LaTeX, Typst)
  - Limited to a more traditional page format rather than infinite workspace like OneNote for example
  - Editing underlying text rather than geometry itself

On top of inefficiencies to workflow, an important note is that none of these solutions are fully keyboard-driven. This is a key aspect of the Vim ideology that Flow aims to embody by providing a unified system for creation and editing of notes involving both text and diagrams exclusively through the keyboard.

(G.3) Expected Benefits

🔄 Nothing available at this point.

(G.4) Functionality overview

🔄 Nothing available at this point.

(G.5) High-level usage scenarios

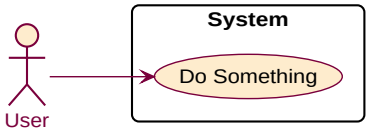

Figure 1. High Level use cases diagram

🔄 Nothing available at this point.

(G.6) Limitations and Exclusions

🔄 Nothing available at this point.

(G.7) Stakeholders and requirements sources

🔄 Nothing available at this point.

(E) Environment

(E.1) Glossary

- **Vim**: Vim is a highly configurable, keyboard-driven text editor built to make creating and changing any kind of text very efficient. [VI25]
- **UI/UX**: User Interface/User Experience.
- **PoC**: Proof of concept. Used to show feasibility of the project.
- **UML**: Unified Modeling Language. Modeling language used for diagrams to standardize modeling and minimize confusion from differences in models.
- **V&V**: Verification and validation. Testing that will be done to check if the project fulfills the requiremnts.
- **SRS** Software Requirements Specification. Document that contains requirements that will be used for the develoment of the project.

(E.2) Components

- **File system**: The system needs a to installed on a system with a file system to store files needed for the application. It will also store files holding notes, letting the user access them later.
- **Keyboard/Mouse/Trackpad/Touchscreen**: The system will use keyboard and mouse for input. These are the standard and primary input for computers. Other methods such as trackpad and touchscreen will be considered similar to mouse input and not prioritized.
- **Display**: As a computer program, the system will display its outputs with a corresponding screen. It should be able to use this display to show the user anything it need, ex current note, errors, tools, etc.

(E.3) Constraints

- **Must be compatible with at least Windows/MacOS/Linux.**: Windows/Mac/Linux are the main operating systems used by computers and correspondingly are the systems that the program must be able to run on. Priority will be given to the operating system that is the most popular, Windows then MacOS then Linux.
- **Must not need unauthorized modifications to data and other systems on device.**: Data and systems may require authorization to view or modify. To prevent errors or distrust in the program it is best to not modify files for other programs.
- **Must not break any laws (Copyrights, Digital security, etc).**: As software, this program needs to follow all laws and regulations relating to laws. These will typically be Copyright/Licensing laws and Digital Security laws. This means that actions such as collecting personal information or using other code must be checked for following the regulation for where we intend to distribute this software.

(E.4) Assumptions

- **User is understanding of how to download/install basic windows programs.**: As we cannot be there to install this program on every computer it will run on, we assume that the user is able to install it by themselves provided they are provided an installer program file.
- **User understands basic computer I/O (Keyboard, mouse, screen).**: To properly intact with the program users would be expected to know how to use their keyboard and mouse to navigate the program. This would include keyboard shortcuts.
- **User not needlessly interact with Files in the system.**: The Program will store taken note files on the computer. It is assumed that the user will not do changes to files that may cause issues when reading the files.
- **Device being run on has proper libraries or user can install them when provided with redists.**: If our program uses other libraries, they need to install for it to work. If the user can install the program than it is assumed that the user can also install required libraires. These libraries need to be provided either as a list or as redistributable files.
- **The device running the program has enough space for the program to function.**: The program will require drive space to store the program and any notes created by the user. If there is not enough space the user should be notified, and the program should continue to hold the current note.

(E.5) Effects

- **Requires storage space from computer (Lower computer storage space).**: The program will require space on the computer drive to both store the program and any notes that are taken. If the computer does not have the required space then the program may not work properly.

(E.6) Invariants

- **The system must preserve notes unless the user requests they be deleted.**: As notes are the primary goal of the system any notes created should stay open unless they are saved or the user decides that they can be deleted.

(S) System

(S.1) Components

Nothing available at this point.

(S.2) Functionality

Nothing available at this point.

(S.3) Interfaces

Nothing available at this point.

(S.4) Detailed usage scenarios

Nothing available at this point.

(S.5) Prioritization

Nothing available at this point.

(S.6) Verification and acceptance criteria

Nothing available at this point.

(P) Project

(P.1) Roles and personnel

🔄 Nothing available at this point.

(P.2) Imposed technical choices

🔄 Nothing available at this point.

(P.3) Schedule and milestones

🔄 Nothing available at this point.

(P.4) Tasks and deliverables

🔄 Nothing available at this point.

(P.5) Required technology elements

🔄 Nothing available at this point.

(P.6) Risk and mitigation analysis

🔄 Nothing available at this point.

(P.7) Requirements process and report

Requirements Process The requirements elicitation process was conducted in a simplified and collaborative manner, reflecting the project's scope and resources. The team collectively engaged in discussions to identify and document the essential features, goals, and requirements for the project.

The process involved the following steps: * **Brainstorming Sessions**: Initial brainstorming sessions to gather the main ideas and requirements. Each member contributed their perspectives on what the project should achieve. Factors considered included key stakeholders, user needs, technical feasibility, and project risks. * **Informal Discussions**: Ongoing informal discussions to refine and clarify the requirements. This included addressing any ambiguities and ensuring a shared understanding among team members. Finer details, including specific requirements and project scope were discussed and agreed upon. * **Documentation**: The requirements were documented in a collaborative manner, with team members contributing to the creation of this Software Requirements Specification (SRS) document. The document was reviewed and updated as needed to reflect any changes or new insights gained during the process. * **Feedback Integration**: Although formal stakeholder feedback was limited, the team acted as both developers and stakeholders, ensuring that the requirements aligned with the project's goals and

constraints. Further feedback from peers and mentors will be sought in future iterations.

For future iterations, a more structured elicitation process may be adopted, including user research, prototyping, and systematic requirements validation to ensure comprehensive coverage of user needs and project goals.

References

- [1] Vim - the ubiquitous text editor. 2025. https://www.vim.org/