System Requirements Specification

Paper Trail

Dr. Zihan Wu

Ben Yandell, Robert Kulow, Anthony Veilleux, Arius Ahmad, Vasu Patel

10/15/2025

V0.1

Table of Contents

1. Introduction

This is a capstone project commissioned by Dr. Zihan Wu. This project will fulfill the capstone requirement for a Computer Science Bachelor's degree for Ben Yandell, Robert Kulow, Anthony Veilleux, Arius Ahmad, and Vasu Patel. The project is intended to give students a low-stakes opportunity to develop their emotional intelligence and software development skills. This project will aim to use JavaScript and Google Scripts to create a seamless and intuitive note assistance software natively for Google Docs and Google Drive that will help researchers organize their notes and write up documents.

1.1 Purpose of This Document

The purpose of this document is to outline the system requirements, so the development staff will be able to differentiate what they need to do, and when they are done. Specifically, this document will cover the functional and non-functional requirements of the software, what kind of user interface the software will utilize, how deliverables will be structured and presented, and any open issues that may present themselves during the development of the software. The software will be designed so that future users can easily utilize the software to assist their own research projects.

1.2. References

No references have been created so far, but this document will be updated with them as the project progresses.

1.3. Purpose of the Product

Previously, the client noticed that Google Docs doesn't natively support project management functions. When they tried to use third-party applications like Notion and Asana, they found them to be very convoluted and unintuitive to use. Prompting them to commission a native, intuitive plugin that works directly within Google Docs while writing. They proposed a hashtag system where any paragraph found below the hashtag is considered a note until a page break or another hashtag. Thus, differentiating notes from actual documentation. Also, they requested a stretch goal: the ability to do quick commands with backslashes to bring up to-do tables and such on the fly.

1.4. Product Scope

This capstone project is aimed to be a lightweight expedition that gives the team members a glimpse of real-world software development processes. The scope being semi-detached as the project is split across two semesters, giving leeway for developmental obstacles to be quickly resolved in a low-stakes environment. The Ultimate goal is to have the finished product done at the end of the second semester.

UML Activity Diagram

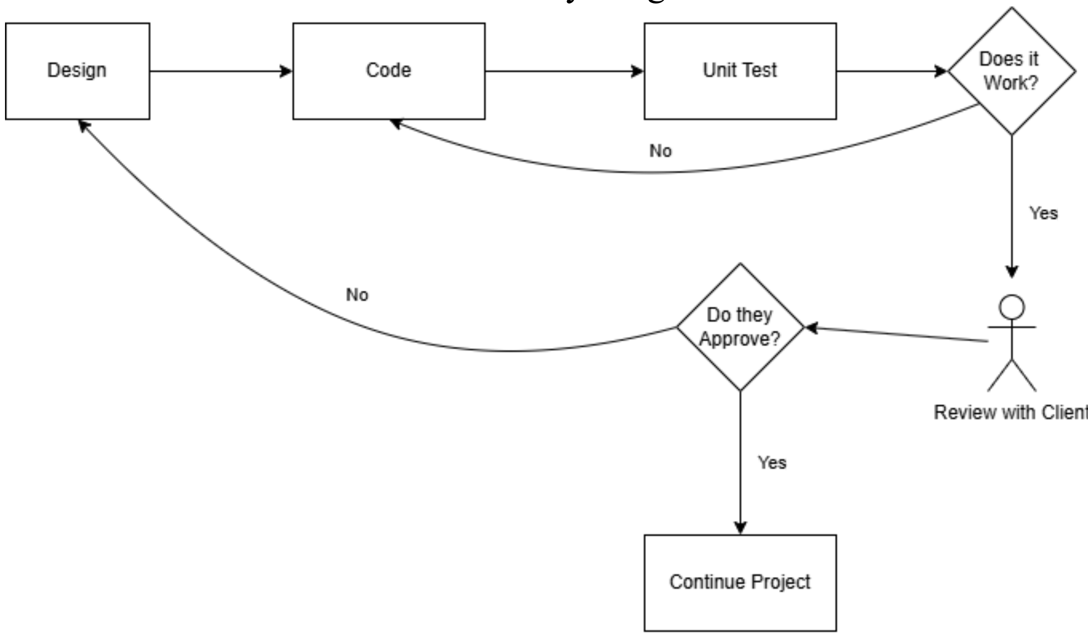

Figure 1: This diagram is a simple demonstration of our design flow we aim to use to gain user involvement from the client.

Use Case Diagram

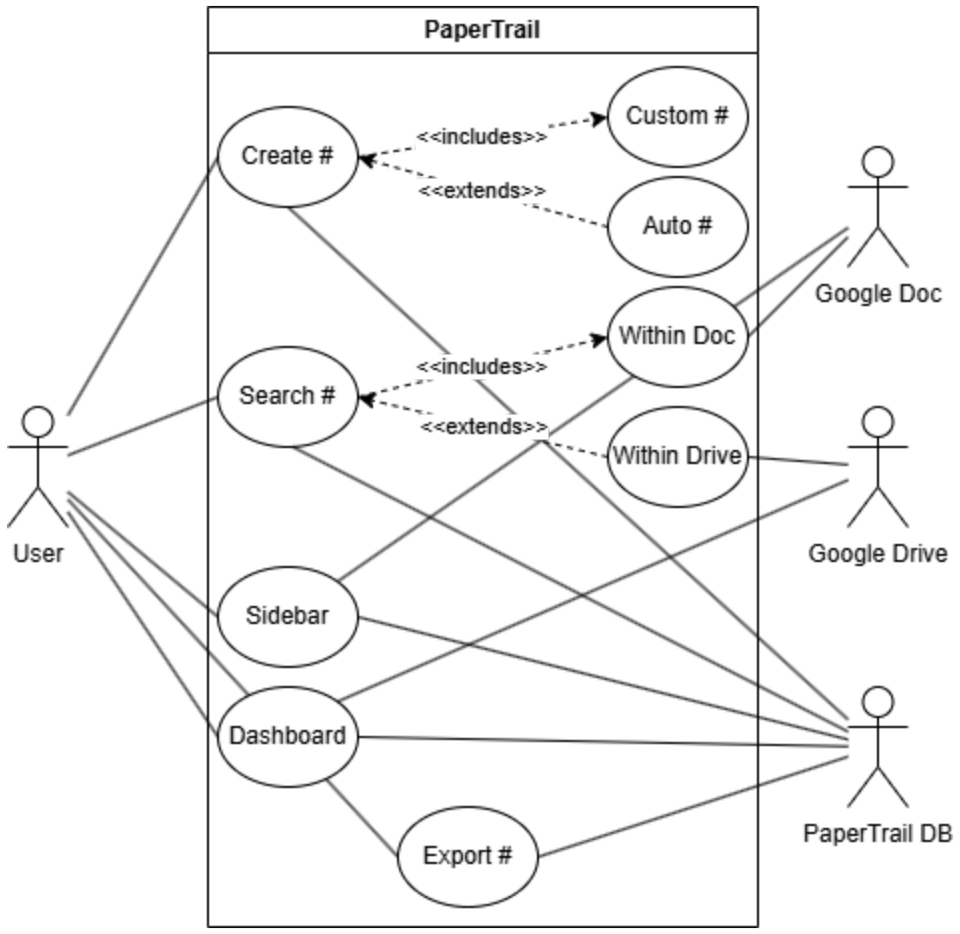

Figure 2: This diagram aims to highlight the interworking systems, and how they will react to user input from the functional requirements.

2. Functional Requirements

Each functional requirement will be in constant review as the software is designed and developed. They will let the software development team know what the software is expected to do, and when they have met the requirements specified by the client.

Number	01	
Name	Create and Categorize Nested Hashtags	
Summary	Format tag lines beginning with recognizing hashtags Ex. #Note, @Idea, $Doc	
Priority	5	
Preconditions	Authorized user defined hashtags Document is editable	
Postconditions	The Tagged lines have consistent styling and readable tag metadata	
Primary Actor	Researcher/Dr. Zihan Wu	
Secondary Actors	Google Docs API	
Trigger	User types supported hashtag at line start or selects suggested tag from UI	
Main Scenario	**Step**	**Action**
	1	User selects **#Idea**
	2	Plugin detects tag on keypress
	3	Plugin stores “tag=Idea” in metadata
	4	Entry is then added to tag index
Extensions	**Step**	**Branching Action:**
	1a	**Unknown tag:** Plugin suggests closest known tag or includes customized tag option
	1b	**User disabled auto-format:** Only metadata added no visual style
Open Issues	Should there be case sensitivity?	
Tests	1. Typing “#Note” styles the line and adds “tag=Note” 2. Typing “@Time” styles the line and adds “tag=Time” 3. Typing “#Note @Time” styles the line and add “tag=Note, tag=Time” 4. Unknown tag triggers “suggested” 5. Tag index shows new entries made	

Number	02	
Name	Search within Document by Tag	
Summary	Navigate to each entry of selected tag via UI	
Priority	4	
Preconditions	Tag exists in a document Tag is indexed At least one tag exists.	
Postconditions	Cursor transfers to selected entry, current position is then updated	
Primary Actor	Researcher/Dr. Zihan Wu	
Secondary Actors	Google Docs API	
Trigger	User selects a tag or enters search criteria into the UI.	
Main Scenario	**Step**	
	1	User picks “#Citation” in UI
	2	Plugin lists all matches with a short preview
	3	User selects desired result and cursor moves to the entry of the doc
	4	UI highlights active item
Extensions	**Step**	
	1a	**No results**: Offer to create tag or show “help” on tagging
	1b	**Large Document:** Load the results present in the document at a slow rate so as to not overload processing, and allow the user to scroll to further entries in order to find their desired tag.
Open Issues	Results autoupdate as user types	
Tests	1. Clicking results jumps to the correct line 2. Search filter reduces list in real time as the criteria entered matches preexisting options. 3. Works with 500+ entries	

Number	03	
Name	Create Custom Hashtag Template	
Summary	Define, save and share custom tags, styles, and behaviors	
Priority	4	
Preconditions	User has permission to manage templates for their document	
Postconditions	New customized tags are available in autocomplete function and UI. New tags are shared to any collaborators	
Primary Actor	Researcher or Template owner	
Secondary Actors	Google Docs API	
Trigger	User opens “Manage Tags” and selects “New Tag”	
Main Scenario	**Step**	
	1	The user enters the tag name
	2	Plugin evaluates the uniqueness
	3	Template is saved to document or team
	4	Autocomplete refreshes to include the new custom tag
Extensions	**Step**	
	1a	**Name conflict:** Suggest alternative or allow "override" mechanic
	1b	**Shared with collaborators:** Tag is stored locally in the current document only, informing the user.
Open Issues	Who is able to create team-scope tags?	
Tests	1. New custom tag appears in autocomplete suggestion 2. Shared tag is visible to other collaborators upon refresh	

Number	04	
Name	Autocomplete Hashtag	
Summary	When user types a symbol ‘#’ or partial identifier ‘@id’ (for idea) plugin opens a suggestion list of known tags including default, custom, and shared. Selecting a suggestion inserts tag, applies style, and records metadata.	
Priority	5	
Preconditions	Plugin is installed Document is open and editable Tag index (default, custom, shared) is open and available for document	
Postconditions	Chosen tag is inserted at cursor position with consistent styling Tag metadata (tag=Idea, tag=Note) is stored and indexed for search and summary features.	
Primary Actor	Researcher or Template owner	
Secondary Actors	Google Docs API	
Trigger	User opens “Manage Tags” and selects “New Tag”	
Main Scenario	**Step**	
	1	User types ‘#’ and starts entering wanted tag
	2	Plugin then detects and shows suggestions of what user may be referring to.
	3	The suggestion is prioritized by what the plugin believes is most relevant (ex. #No being #Note)
	4	User clicks desired autocomplete tag (or perhaps by using tab similar to other google autocompletion)
	5	Plugin inserts tag style and adds metadata
		The autocomplete suggestion list closes and focus goes back to current document
Extensions	**Step**	
	1a	**If no matches:** Present the “create new tag” option
	1b	**User cancels:** Close autocomplete suggestion.
Open Issues	1. Ranking the autocomplete suggestions a. Ex. frequency of used tag/most recent	
Tests	1. Halfway typing a tag ‘#No’ shows ‘#Note’ as top suggestion 2. Selecting a suggestion inserts its style and metadata 3. Pressing ‘Esc’key closes autocomplete suggestion	

Number	05	
Name	Search/Filter by Documents containing Tags	
Summary	When a user types in the tag in Google Drive, it should retrieve the documents containing that tag.	
Priority	5	
Preconditions	Plugin is installed Documents with tags exist User is in Google Drive	
Postconditions	A list of Google documents that contain the queried tags are presented to the user.	
Primary Actor	Researcher	
Secondary Actors	Google Docs/Google Drive	
Trigger	User clicks on search bar within Google Drive	
Main Scenario	**Step**	
	1	When a user is in Google Drive, they click on the search bar
	2	The user then enters the tag(s) they want to search for
	3	The system then retrieves those documents and presents them to the user
Open Issues		
Tests	1. Search for Documents with tags that are known to exist 2. Search for Documents with tags that don't appear	

Number	06	
Name	Sidebar Suggestions in Google Docs	
Summary	Display a toggleable non intrusive sidebar that suggests related tags, recent /related notes or other auto detected elements like dates and names based on the current document content.	
Priority	3	
Preconditions	Plugin is installed Document is open and editable Document has data and tags for analysis	
Postconditions	Sidebar updates dynamically	
Primary Actor	Researcher or Student	
Secondary Actors	Google Docs API	
Trigger	User activates side-bar	
Main Scenario	**Step**	
	1	User activates side-bar
	2	Plugin displays tags with dates
	3	Sidebar lists suggestions and displays related documents if applicable
	4	User selects a suggestion
	5	Plugin inserts a suggestion
Extensions	**Step**	
	1a	Toggleable - User can toggle the sidebar on/off easily
	1b	User can disable autodetection
Open Issues		
Tests	1. Selecting a suggestion inserts tag correctly 2. Sidebar correctly displays related notes 3. Sidebar refreshes in real time	

Number	07	
Name	Google Drive Dashboard View	
Summary	Toggleable dashboard in Google Drive that lists all hashtags across documents, allows browsing by tag, and shows previews of related notes.	
Priority	4	
Preconditions	User is in Google Drive Plugin installed Data available	
Postconditions	Dashboard displays filtered results; clicking a note preview opens the document.	
Primary Actor	Researcher/Student	
Secondary Actors	Google Docs API	
Trigger	User accesses the dashboard via the plugin menu in Google Drive.	
Main Scenario	**Step**	
	1	User opens dashboard
	2	Plugin aggregates tags from accessible documents
	3	User filters by tag
	4	Dashboard shows notes previews
	5	User clicks on document preview and is sent to that document
Open Issues		
Tests	1. Filtering by tag shows only matching documents. 2. Preview accurately reflects note content. 3. Handles 100+ documents without lag.	

Number	08	
Name	Export Tagged Notes	
Summary	Allow users to export tagged notes (e.g., all #ToDo items) to PDF, or another Google Doc for sharing or backup.	
Priority	3	
Preconditions	User is in Google Drive Plugin installed Data available	
Postconditions	Export file generated and downloadable.	
Primary Actor	Researcher/Student	
Secondary Actors	Google Docs API	
Trigger	User selects the export option from the UI.	
Main Scenario	**Step**	
	1	User chooses tag and format
	2	Plugin compiles matching notes
	3	Generates and downloads files
Open Issues		
Tests	1. Export matches all tagged content 2. File opens correctly in target format 3. Handles 100+ notes	

3. Non-Functional Requirements

The non-functional requirements will measure how well the software performs under various stress tests. The tests will consist of recording the responsiveness and stability of the software over a period of time, as well as ensuring that no cybersecurity flaws are being violated. Throughout the development process, we aim to keep these requirements in mind during our design and testing phases.

NFR #	Related FR #	Priority (1–5)	Requirement Description	Verification / Test Method
NFR-01 Performance	FR-01: Create and Categorize	5	The system shall tag and format lines within 2 seconds after the user enters a hashtag (e.g., #Note or #Idea) in a document up to 50 pages.	**Test-01:** Time plugin response for 20 consecutive tag entries in a 50-page doc; verify formatting completes ≤ 2 seconds each.
NFR-02 Accuracy & Consistency	FR-01: Create and Categorize	4	All tagged lines shall display consistent font style, color, and metadata formatting across all users and sessions.	**Test-02:** Apply identical tags across 3 user accounts; confirm uniform style and metadata consistency.
NFR-03 Performance / Responsiveness	FR-02: Search within Document by Tag	5	The plugin shall display tag search results within 1 second for up to 500 tagged entries in a single document.	**Test-03:** Measure latency during 10 search queries in a test doc; verify ≤ 1 second response.
NFR-04 Usability	FR-02: Search within Document by Tag	4	Search results shall highlight matches clearly and maintain navigation state even after scrolling or editing.	**Test-04:** User testing with 5 participants; confirm navigation and highlight persist after edits.

NFR-05 Usability / Customization	FR-03: Create Custom Hashtag Template	3	The “Manage Tags” interface shall allow a new tag to be created and saved with ≤ 3 user actions.	**Test-05:** Observe test users creating a new tag; verify average ≤ 3 user actions.
NFR-06 Authentication	FR-03: Create Custom Hashtag Template	4	Only authenticated users shall be able to create or modify shared tag templates Unauthorized users shall be denied access.	**Test-06:** Attempt tag creation without authentication; verify system denies request and logs event.
NFR-07 Predictive Performance	FR-04: Autocomplete Hashtag	5	Autocomplete suggestions shall appear within 0.5 seconds of typing “#” or partial text, ranking most-used tags first.	**Test-07:** Measure response time for 10 random hashtag queries; verify ≤ 0.5 second average delay.
NFR-08 Usability / Learnability	FR-04: Autocomplete Hashtag	4	The autocomplete menu shall be intuitive, allowing users to select suggestions using keyboard input (Tab/Enter).	**Test-08:** Observe 5 users performing tag insertions; confirm both selection methods function correctly.
NFR-09 Integration & Reliability	FR-05: Search/Filter by Documents Containing Tags	4	The plugin shall interface reliably with Google Drive API and retrieve matching documents with ≥ 99% success rate.	**Test-09:** Conduct 100 Drive search operations; verify ≥ 99 results returned correctly.
NFR-10 Security / Privacy	FR-05: Search/Filter by Documents Containing Tags	5	Search queries and tag metadata shall never be transmitted outside of the user’s authenticated Google environment.	**Test-10:** Perform network packet inspection; verify no external API calls beyond Google endpoints.

NFR-11 (Performance)	FR-06: Sidebar Suggestions in Google Docs	3	The sidebar shall refresh contextual suggestions within 3 seconds after document edits exceeding 100 words.	**Test-11:** Edit 100+ words and measure sidebar update time ≤ 3 sec.
NFR-12 (Usability / Non-Intrusiveness)	FR-06: Sidebar Suggestions in Google Docs	4	Sidebar shall occupy ≤ 25% of screen width and be dismissible via toggle button at all times.	**Test-12:** Confirm layout constraint and visibility toggle in Chrome, Edge, Firefox.
NFR-13 (Maintainability)	FR-07: Collaborative Tag Sharing (if applicable)	3	The plugin code shall use modular structures and be fully documented to support new feature updates in < 2 hours of dev time.	**Test-13:** Developer inspection confirming code modules and documentation completeness.
NFR-14 (Compliance / External)	FR-08: Integration & Deployment	4	The plugin shall comply with Google Workspace publishing standards and FERPA data guidelines.	**Test-14:** Review package for compliance checklist and approval verification.

4. User Interface

See User Interface Design Document for PaperTrail to understand the user interface functionality and layout.

5. Deliverables

Upon completion, all deliverables will be presented to the client for approval in order to gain user involvement with the client. Deliverables will have hard and digital copies for file maintenance and to track project progress. Hard copies will be distributed to each team member and the client, while digital copies will be stored in the GitHub repository.

There will be hard copies of the following materials:

- Systems Requirement Specification (10/29/25)
- System Design Document (11/17/25)
- User Interface Design Document (12/03/25)
- Critical Design Review Document (12/19/25)
- Code Inspection Report (TBD)
- Administrator Manual (TBD)
- User Guide (TBD)
- Final Project Report (TBD)

There will be digital copies of the following materials:

- Systems Requirement Specification (10/29/25)
- System Design Document (11/17/25)
- User Interface Design Document (12/03/25)
- Critical Design Review Document (12/19/25)
- Code Inspection Report (TBD)
- Administrator Manual (TBD)
- User Guide (TBD)
- Final Project Report (TBD)
- All source code (TBD)
- The executable program (TBD)
- Any other software required for installation and execution of the delivered program. (TBD)

6. Open Issues

User Interface:

The user interface of the plugin is still to be finalized. After the first meeting with the client, they stated they wanted a minimal, nonintrusive UI. The plan moving forward is to work with the client using UI mockups and prototypes in hopes of gaining a better understanding of precisely what the client is looking for. This issue will continue to become a higher priority as we get closer to creating and finalizing the User Interface Design Document.

Appendix A – Agreement Between Customer and Contractor

This document represents a formal agreement between the PaperTrail Development Team and Dr. Zihan Wu. The agreement confirms that both parties understand and accept the scope, objectives, and deliverables of the *PaperTrail Project*, which aims to create a document tracking and workflow automation system integrated with Google Docs and Google Drive. Both parties agree to collaborate throughout the project lifecycle, with the development team responsible for implementing the system according to the specifications outlined in this Software Requirements Specification (SRS) document, and the customer providing timely feedback and testing support.

In the event of future changes to this document, all revisions must be proposed in writing by either party. Any modification will require a review meeting between the customer and the contractor, and mutual written approval before updates are considered valid. Updated versions of the SRS will be labeled clearly with version numbers and dates to ensure traceability.

Signatures

Name (Typed)	Signature	Date
Dr. Zihan Wu (Client)	Zihan Wu	Oct 29, 2025
Vasu Patel	[illegible]	
Ben Yandell	[illegible]	Oct 29 2025
Robert Kulow	[illegible]	
Arius Ahmad	[illegible]	10/29/25
Anthony Veilleux	[illegible]	

Appendix B – Team Review Sign-off

All members of the PaperTrail Development Team have thoroughly reviewed this Software Requirements Specification (SRS) document and agree that it accurately represents the project's goals, scope, and requirements. Each team member confirms that the content and format of this document meet the expectations of both the team and the customer. While minor editorial or stylistic differences may exist, there are no major points of disagreement among the team regarding the material presented herein.

Team Member Signatures

Name (Typed)	Signature	Date
Vasu Patel	[illegible]	OCT 29 2025
Ben Yandell	[illegible]	Oct 29 2025
Robert Kulow	[illegible]	Oct 29 2025
Arius Ahmad	[illegible]	10/29/25
Anthony Veilleux	[illegible]	OCT 29 2025

Appendix C – Document Contributions

The following table identifies each team member's contributions to the creation of this Software Requirements Specification document. Each member has contributed to writing, editing, and reviewing the document. The percentages represent an estimate of each member's contribution to the total effort.

Team Member	Contributions	Percentage of Work
Vasu Patel	Completed Section 3, Appendix A, Appendix B, and Appendix C. And reviewed the document for final Submission.	20%
Ben Yandell	Added most Functional Requirements.	10%
Robert Kulow	Oversaw document progression, completed Section 1: Introduction, Section 4: User Interface, and Section 5: Deliverables.	30%
Arius Ahmad	Completed Section 6: Open issues. Made small grammatical corrections to descriptions throughout the document. Fixed formatting issues across document to improve organization.	20%
Anthony Veilleux	Added several functional requirements and made grammatical and consistency corrections	20%