SE 4G06: Software Requirements Specification
UnderTree

Team 22, Capstoners
Palanichamy Veerash
Kannammalil Kevin
Qureshi Eesha
Ahmed Faiq

April 1, 2023

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
October 4th	1.0	Introduction added by Eesha
October 4th	1.1	Functional requirements started by Faiq and Veerash
October 4th	1.2	Nonfunctional requirements started by Kevin
October 5th	1.3	Project Issues added by Faiq and Veerash
October 5th	1.4	Functional requirements finished and revised
October 5th	1.5	Non functional requirements finished and revised
October 5th	1.6	Appendix added
October 5th	1.7	Traceability matrix added
March 31st	1.8	Fixed a requirement that does not apply

1 Changes to template

aThe template was changed to include separate sections for main requirements and requirements for undesired events in the functional requirements section to create a better distinction between them and make the template easier to follow. Additionally sections to make a distinction between likely and unlikely changes in the functional requirements and rationale for specific requirements were also added. Lastly, additional sections to include the traceability matrix and phase in plan were added as they were missing from this template.

2 Project Drivers

2.1 The Purpose of the Project

The purpose of this project is to create a collaborative **LaTex** editing and compiling software that supports version control with **Git** integration. The software will allow for multiple members of a team to view, edit and compile the same**tex**files at the same time while seeing the changes made by others in real time. This is especially useful for teams preparing project documentation where multiple members need to contribute to the documents in parallel, however it is widely intended for the development of any **LaTex** document by any number of contributors for any reason.

The **LaTex** editor will support version control management through **Git** integration so that the version history of any document created will be easily accessible. This is especially useful for documents that are likely to change over time due to revisions, such as software documentation and release notes

2.2 The Stakeholders

2.2.1 The Client

The client of this software is Dr.Spencer Smith who has requested the development of this project for the software engineering Capstone course in order to solve a real world problem. The clients also include enterprises and companies that are seeking a standardized documentation software to prepare internal documents across teams who possess **LaTex** proficiency.

2.2.2 The Customers

The customers are the purchasers and users of this software. This includes individuals who wish to install the software for their personal use, students who wish to use the software for individual or team projects, companies and teams who wish to purchase a license and anyone else interesting in using the**tex**editor.
Customers also include researchers, developers, and other secondary customers of the software such as employees of a company that chooses to make use of the software for their teams. Such employees will be interested in the usability and features of the product. Researchers and developers who are interested in the expansion of the document editor market or the development of such softwares are included in the stakeholders.

2.2.3 Other Stakeholders

Other stakeholders also include the primary developers and testers of this software, namely Faiq Ahmed, Eesha Qureshi, Veerash Palanichamy, and Kevin Kannammallil as their software engineering skills are a resource invested in this software and they are concerned in the outcome of this project.

2.3 Mandated Constraints

MC1. Project must be completed by April 2023
Rationale: Project must adhere to Capstone 4G06 timeline

2.4 Naming Conventions and Terminology

- **LaTex**:A document preparation system and language
- **Tex**: A shorthand for **LaTex**
- **Git**: A version control software
- **GitHub** A propriety hosting software for **git** version control
- **Commit**: A snapshot or the creation of a snapshot of a project that is stored in a commit history log and can be viewed at any time
- **Repository**: Central location where all **git** version history of data is stored

- **PDF**: A printable document file
- **API**: An interface between two services
- **JSON Web Tokens**: Compact tokenized methods of verifiable data transfer that use JavaScript string object notation

2.5 Relevant Facts and Assumptions

1. Users are proficient with **LaTex**
2. Users are proficient with **Git**
3. The system will validate that a user is logged in
4. The system will create a **Git** repository for new projects
5. Users will be operating on a Chrome, Firefox or Safari web browser

3 Functional Requirements

3.1 The Scope of the Work and the Product

3.1.1 The Context of the Work

3.1.2 Context Diagram

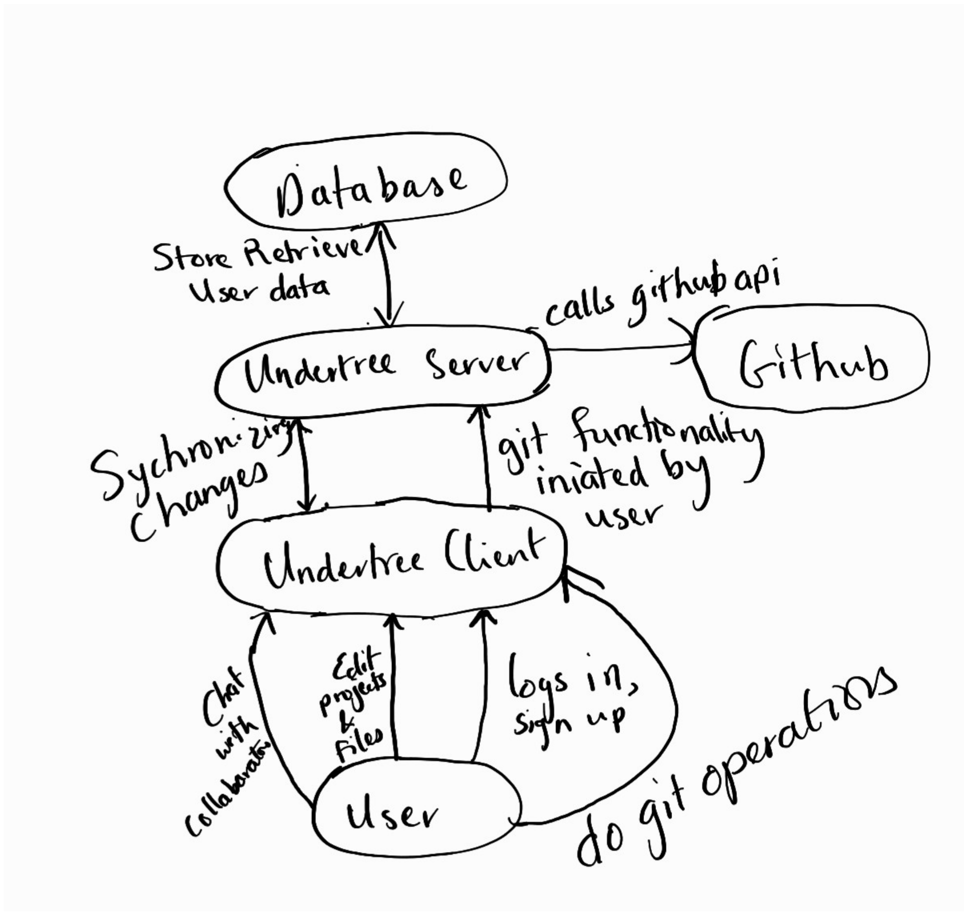

Figure 1: Context Diagram of UnderTree

3.1.3 Work Partitioning

Event Name	**Input/Output**	**Summary**
1. Log in	**Input:** **GitHub** Username, **GitHub** Password **Output:** New access permissions related to the account	The user will need to log in with their **GitHub** account to get access to their private projects and be able to edit them with collaborators.
2. Create project	**Input:** Project name and collaborators **Output:** Access to the new project and a new **Git** repository for the project	Users will need to create a project to work on the **LaTex** files.
3. Import Project	**Input:** Select **Git** repository **Output:** A new project with the data from the repository	The user may want to import existing repositories
4. Delete Project	**Input:** Mouse click to delete project and mouse click to confirm delete **Output:** Remove the project from the user and all collaborators accounts	The user may want to remove a project from their accounts
5. Edit Project	**Input:** Collaborator's Id to add to project **Output:** New collaborator in project	Users may want to add more collaborators to the project.
6. View all projects	**Input:** Open project menu **Output:** Display a list of all projects and their names	User will want to see a list of all their existing projects
7. Create a file	**Input:** File name **Output:** New file in project	This allows user to add files to a project
8. Delete a file	**Input:** Click delete button **Output:** Remove file from project	Users may wish to remove existing files from a project.

9. Edit file	**Input:** Text to add, remove or modify **Output:** Updated changes in the file	Adding new changes to the file should be reflected to all users
10. View file	**Input:** Click button to open file **Output:** Display all users cursors and live changes happening to the file	Users should be able to see the changes happening to the file in real time
11. View revision history	**Input:** Click button to see revision history **Output:** Display all revisions as well as users responsible for each revision	Users should be able to see **commit** history and how file was changed over time
12. **Commit** changes	**Input:** Select sections to **commit** and add a message for the **commit** **Output:** All changes to the project reflected in **Git** repository	Users should be able to reflect the changes from the project into the corresponding **Git** repository
13. Send chat message	**Input:** Type chat message and click the send button **Output:** Chat message should be sent and visible to other users	Users should be able to communicate with each other over the chat
14. View chat messages	**Input:** Open the chat **Output:** All chat messages sent by other	Users should be able to see all messages sent by other users in the project

Table 2: Work Partitioning Events and Summaries

3.1.4 Individual Product Use Cases

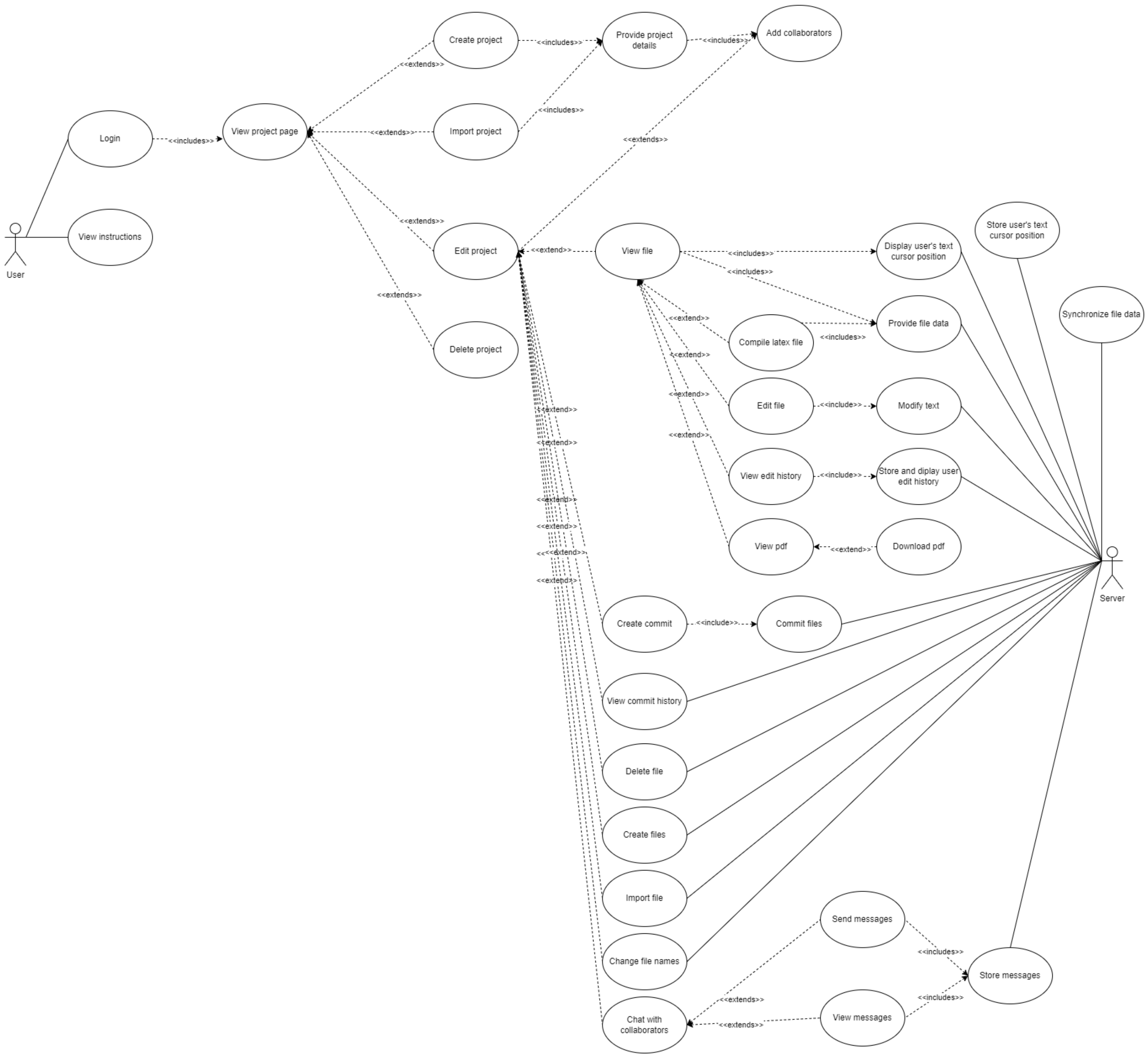

Figure 2: Use case diagram for UnderTree, Link to draw.io

3.2 Functional Requirements

3.2.1 Main Requirements

To avoid repetition in the functional requirements for all functional requirements after log in, assume that the system checks to ensure that the user is logged in and has appropriate access level.

Formal Specification Variables:
change: This is a specific datatype that stores changes made by a spefic user.
all_changes: This is an array of changes that stores all changes made in a file in order.
user_changes: This is an array of changes specific to a user in a specific file.

BE1. The user wants to view instructions

- FR1. The system must allow the user to be able to open and view instructions
- FR2. The system must allow the user to close the instructions and return to their previous task

BE2. The user wants to login to their account

- FR3. The system must allow the user to log into the system with a unique **Git** identity
- FR4. The system must validate the identity of the user

BE3. The user wants to create a project

- FR5. The system must allow the user to pick the project name
- FR6. The system must allow the user to add collaborators to the project
- FR7. The system must create a **Git** repository for the project

BE4. The user wants to import a project

- FR8. The system must ask user to choose a repository to import
- FR9. The system must import all**tex**file in the repository
- FR10. The system must import all collaborators from the repository

BE5. The user wants to delete a project

- FR11. The system must remove the project data for the user
- FR12. The system must remove the project data for all collaborators

BE6. The user wants to view all their projects

- FR13. The system must display all projects that the user has created or imported
- FR14. The system must display all projects that the user is a collaborator of
- FR15. The system must display all projects in order of most recently modified to least recently

BE7. The user wants to edit a project

- FR16. The system must allow the user to add collaborators to the project
- FR17. The system must allow the user to remove collaborators from the project

BE8. The user wants to view all files in the project

- FR18. The system must display all **LaTex** files that are within a project
- FR19. The system must display all files that are in the project

BE9. The user wants to view the **LaTex** file

- FR20. The system must display all user's current text cursor position
- FR21. The system must display the file with all the changes made by different collaborators up until that current instance of time

  $\forall$ change $\cdot$ change $\in$ all_changes $|$ display(change)

  display(change): This function converts a change datatype into text and displays it to the user

- FR22. The system should highlight the **LaTex** syntax in the file accordingly
- FR23. The system should highlight any errors in spelling or grammar

BE10. The user wants to create a file

- FR24. The system must come up with a default name and extension for the file
- FR25. The system must give the user the option to change the file name and extension

- FR26. The system must add an empty file with the selected name and extension to the project

BE11. The user wants to upload a file

- FR27. The system must give the user the option to find and upload their local file
- FR28. The system must add the uploaded file with the original name and extension

BE12. The user wants to delete a **LaTex** file

- FR29. The system must ask the user to confirm their decision
- FR30. If the user confirms to delete the file, the system must remove the file from the project

BE13. The user wants to change a file's name

- FR31. The system must take the new file name as input
- FR32. The system must update the file name accordingly
- FR33. The system must record this name change alongside the user who made this change

BE14. The user wants to edit **LaTex** file

- FR34. The system must allow the user to insert, delete or modify text in the **LaTex** file currently being accessed

  edit(string: text) $\rightarrow$ all_changes || createChange(text, user)
  remove(string: text) $\rightarrow \exists$ i $\cdot$ all_changes[i].text = text | all_changes := all_changes[i-1] || all_changes[i+1]

  edit(string): This function is triggered by the user writing or modifying text
  createChange(string, string): This function is used to create a change data type for a specific user. This function returns a change.
  remove(string): This function is triggered by the user when deleting text

FR35. The system must (save all changes instead of record this edit might be better) record this edit with date, time and the user who made this change

BE15. The user wants to compile the **LaTex** file currently being accessed

FR36. The system must compile the **LaTex** file being currently accessed

FR37. The system should display any errors during the compilation

FR38. The system must save the compiled **LaTex** file as a PDF with the same name

BE16. The user wants to view the compiled **LaTex** file

FR39. The system must display the most recently compiled PDF for the selected file in the project

BE17. The user wants to see the revision history of all the files in the project

FR40. The system must display all confirmed/**committed** changes in the project

FR41. The system must display the identity of the user that **committed** the change

BE18. The user wants to chat with collaborators

FR42. The system must allow the user to open the chat

FR43. The system must allow the user to send a message in the chat

FR44. The system must allow the user to see all previous messages sent by other collaborators

BE19. The user wants to **commit** changes in the project

FR45. The system must ask the user to add a message describing the **commit**

FR46. The system must allow the user to pick changes that they have made in a file that they would like to **commit**, this includes newly created and deleted files:

select(user_changes) $\rightarrow$ user_changes $\subset$ all_changes

select([]change): This function is triggered whenever the user selects a change.

FR47. The system must push the selected changes to the to the corresponding **Git** repository along with the user defined message:

$\forall$ change $\cdot$ change $\in$ user_changes | push(change)

push(change): This function is responsible for syncing the changes with the **Git** repository.

3.2.2 Undesired Event Handling

UE1. If the user tries to access projects or files without logging in, the user should be redirected back to the log in screen.

UE2. If the user tries to access a file they do not have permission to access, the system must display an error stating that they do not have the appropriate permissions to the user.

UE3. If the user access the application when the server is not responding, the system must display an error stating that the server is currently unavailable to the user.

3.2.3 Likely and Unlikely changes

Likely Changes	FR9, FR10, FR15, FR19, FR33, FR40, FR41
Unlikely Changes	FR1, FR2, FR3, FR4, FR5, FR6, FR7, FR8, FR11, FR12, FR13, FR14, FR16, FR17, FR18, FR20, FR21, FR22, FR23, FR24, FR25, FR26, FR27, FR28, FR29, FR30, FR31, FR32, FR34, FR35, FR36, FR37, FR38, FR39, FR42, FR43, FR44, FR45, FR46, FR47

3.2.4 Rationale and Clarifications

- FR3: We require a log in using **Git** (specifically **GitHub**) so we can ask them for access a **GitHub API** token from them. This is necessary for the **Git** integration. The unique identities will be based on the **GitHub** account we get from the **API**.

- FR8: We must give an option to the user to import existing project that already has documentation. This way they can edit projects that they may have already created in the past.
- FR11: We only want to remove the project data from our app and not delete the project from **GitHub**. Deleting the project from **GitHub** is irreversible and should be done by the user directly from **GitHub** if they wish to do so. However deleting the project from our application does not prevent it from being reimported again later and does not have severe consequences.
- FR20: We want to display the text cursor of each collaberater that is currently working on the document, this way other users can see which part each person is working on in real time.
- FR21: We want users to be able to see what the other users are working on. This improves collaboration since multiple users can work on a task simultaneously and edit each others work without having to **commit**.
- FR35: We store all changes along with the details about the user, date and time instead of just storing it as a plain text file so we can use this information to allow users to select their sections for the **commits**.

4 Non-functional Requirements

4.1 Look and Feel Requirements

4.1.1 Appearance Requirements

NFR1. The system shall use a simple and minimal design
Fit Criterion: Users should give atleast a $SIMPLICITY\_RATING$ on simplicity and overall design

4.1.2 Style Requirements

NFR2. The system will use consistent styling and theme
Fit Criterion: The design should use no more than $NUM\_COLOURS$ different main colors

NFR3. Text cursors for different users should be distinct colors

4.2 Usability and Humanity Requirements

4.2.1 Ease of Use Requirements

NFR4. The system shall be easy to use for a beginner who already has experience in **LaTex**
Fit Criterion: New users shall be able to comfortably use the features of the system within $LEARNING\_TIME$ mins

4.2.2 Personalization and Internationalization Requirements

NFR5. The system shall allow the user to choose between visual themes like light mode and dark mode

4.2.3 Learning Requirements

NFR6. The system shall be used by users who are comfortable with **LaTex** or are looking to learn it

NFR7. The instructions provided by the system should be should be easy to understand
Fit Criterion: New users will be surveyed on the difficulty, and on average users should give $EASINESS\_RATING$ on ease of use

4.2.4 Understandability and Politeness Requirements

NFR8. The system shall use words and symbols that are intuitive and naturally understandable

NFR9. The system shall abstract the details from the users that are unnecessary

4.2.5 Accessibility Requirements

N/A

4.3 Performance Requirements

4.3.1 Speed and Latency Requirements

NFR10. The system shall provide a response to the user in a reasonable time to avoid interrupting the user's flow

Fit Criterion: New users will be surveyed on the responsiveness of the app $RESPONSIVENESS\_RATING$ on ease of use

NFR11. The system shall be able to compile the user's **LaTex** file in less than $COMPILE\_TIME$ seconds

4.3.2 Safety-Critical Requirements

N/A

4.3.3 Precision or Accuracy Requirements

N/A

4.3.4 Reliability and Availability Requirements

NFR12. The system shall have an up time of at least $UPTIME$%

4.3.5 Robustness or Fault-Tolerance Requirements

N/A

4.3.6 Capacity Requirements

NFR13. The system will store all the data in a remote database

4.3.7 Scalability or Extensibility Requirements

NFR14. The system can be vertically scaled with the hardware of the environment it is running in

NFR15. The system can be horizontally scaled with the number of servers running

4.3.8 Longevity Requirements

NFR16. The system shall be developed to be easily maintainable in the long term

4.4 Operational and Environmental Requirements

4.4.1 Expected Physical Environment

NFR17. The system should be used on a user's preferred work device

NFR18. The system will work best with the usual peripheral devices like a keyboard

4.4.2 Requirements for Interfacing with Adjacent Systems

NFR19. The system should work as expected when interfacing with adjacent systems due to it's collaborative design

4.4.3 Productization Requirements

N/A

4.4.4 Release Requirements

NFR20. The system will have it's final iteration and release on March 20th 2023

4.5 Maintainability and Support Requirements

4.5.1 Maintenance Requirements

NFR21. The system will be fully documented on how to use it

NFR22. The system will have diagrams regarding the architecture and code

4.5.2 Supportability Requirements

NFR23. The system is supported to run on all chromium based Browser, Firefox, and Safari
Fit Criterion: Users will be able to see all functionality and run them as expected on common web browsers like Google Chrome and Safari

4.5.3 Adaptability Requirements

N/A

4.6 Security Requirements

4.6.1 Access Requirements

NFR24. The system's code will only be view-able to the public through the Git repository

NFR25. The system restricts it's editing privileges for the code to only the maintainers
Fit Criterion: Users will be able to view the source code on the **GitHub** repository page

NFR26. Users can only access projects that they are authorized to do so

NFR27. The system will retrieve the minimum required data needed for the user

NFR28. The system will provide appropriate errors to communicate system issues to user

NFR29. The system will provide necessary confirmations to crucial changes

4.6.2 Integrity Requirements

NFR30. The system will not manipulate or modify any of the user's data that is stored on it

NFR31. The system will protect itself from intentional abuse

NFR32. The system will back up the data in the database frequently

NFR33. The system will retain user's projects even after being deleted for a few days

NFR34. The system will store the editor data locally on the client's device

NFR35. The system will implement strict measures in the back end to prevent unintentional behaviour

4.6.3 Privacy Requirements

NFR36. The system will require the user to create an account

NFR37. The system will not use the user's personal information for anything than what is required and consented to by the user

NFR38. The system will store all user credentials securely

NFR39. The system will renew API keys regularly

NFR40. The system will re-authenticate the user when required

4.6.4 Audit Requirements

N/A

4.6.5 Immunity Requirements

N/A

4.7 Cultural Requirements

4.7.1 Cultural Requirements

N/A

4.8 Legal Requirements

4.8.1 Compliance Requirements

NFR41. The system is not accountable for the legality of the data that is being stored

NFR42. The system is not responsible for the loss of any data

NFR43. The system will not use any copyrighted information

4.8.2 Standards Requirements

NFR44. The system shall follow the IEEE standards

4.9 Health and Safety Requirements

N/A

This section is not in the original Volere template, but health and safety are issues that should be considered for every engineering project.

5 Traceability Matrix

Table 3: **Traceability Matrix between FR & NFR**

	NFR3	NFR7	NFR11	NFR36	NFR37
FR1		X			
FR3				X	X
FR4				X	X
FR20	X				
FR36			X		

6 Project Issues

6.1 Open Issues

N/A

6.2 Off-the-Shelf Solutions

The most popular off the self solution and a competitor to this project would be overleaf which is an open-source product. Although, Overleaf does its job as a **LaTex** editor that allows collaboration and shows real time edits of all users, it has poor **Git** integration and lacks proper version control. A lot of collaborative **LaTex** projects are hosted using **Git**, but do not have access to a live editor. Overleaf also lacks some commonly used **LaTex** features. UnderTree is going to be a collaborative real time text editor designed with **Git** support and users in mind.

6.3 New Problems

N/A

6.4 Tasks

As this project is part of the course Software Engineering 4G06, it will follow the product life cycle based on the deliverable outline of that course.
Detailed tasks can be found in the ZenHub board that can be found in the project's **GitHub** repository.

6.5 Migration to the New Product

N/A

6.6 Risks

The biggest risk for this project that we should be concerned about is the processing speed of editing the file. Since multiple people can edit the same document at the same time, a correct concurrency implementation is required to ensure that the edits are smooth and do not freeze access for other users.

6.7 Costs

The only cost associated with this project is the cost for renting the server that the application would be hosted on, as well as a domain name. For a fixed price digital ocean server this cost would be 20$/month with the domain name costing an additional 10$/year.

6.8 User Documentation and Training

There is very little documentation that would be required on our part as the user is only required to be able to know **LaTex** to be able to use this product. However, we would have documentation on the same web applications that would point to existing resources that already exist to learn about **LaTex** from.

6.9 Waiting Room

There are several more additions that can be added given extra development time, but the main ones from highest priority are:

- Ability to copy & paste images into the **LaTex** file
- Simplify creating tables in **LaTex**
- Ability to import word document which will be automatically converted to **LaTex**

6.10 Ideas for Solutions

N/A

7 Phase In Plan

1. Implement text editor design - Oct 28th 2022
2. Implement live edit feature for allowing real time collaboration - Nov 11th 2022
3. Implement **LaTex** compiler - Nov 14th 2022
4. Implement login capability - Nov 18th 2022
5. Implement **commit** capability - Dec 2nd 2022
6. Implement revision history - Dec 16th 2022
7. Implement download and upload files capability - Jan 6th 2022
8. Implement import and export project capability - Jan 20th 2022
9. Implement chat capability - Feb 10th 2022

8 Appendix

8.1 Knowledge Required

1. The team would need to be knowledgeable about the basics of front-end development. This includes knowledge of creating template with HTML, making complex and responsive designs with CSS, and using JavaScript to add interactivity to the web application. Additionally, to assist with the development speed of this project, the team should be knowledgeable about using front-end frameworks like React and CSS frameworks like Bulma to rapidly design the website.

   To be able to learn these technologies we can either look at tutorials on YouTube or use the documentation that is provided online for all these resources.

   Faiq: To be able to learn about these technologies, I think the best approach is looking at tutorials on YouTube. This is the preferred method since it helps apply the knowledge to a web development project that can easily be generalized to our own project as well.

   Eesha: The approach to gain working proficiency with these technologies is to consult documentation and practise developing simple components or side projects. This will help gain familiarity with the necessary core concepts and best practises.

   Kevin: I'm a big fan of tutorials that are on YouTube to learn stuff, especially for work regarding the front-end. Since that helps visualize what exactly I'm trying to create and how to do it. This will refresh my mind on best practices for front-end work.

2. All members of the team would need to be knowledgeable about software development in Rust. Since Rust is a relatively new programming language with a unique memory management system, members of the team would not be very familiar with being able to design the necessary software patterns in this language. The team would need to familiarize themselves with this language and the software design patterns that are the best fit for it.

To be able to learn Rust, we can use the Rust book provided online by the Rust foundation, or look at tutorials on YouTube.

Faiq: To be able to learn more about Rust, the best approach for me would be to look through the Rust book. This book is relatively short and goes over all the main topics of the language. It also contains several examples of full applications which would help up with applying the knowledge to our own project. Finally the book is also targeted towards people who are already experienced with programming and wish to learn Rust.

Veerash: For the best approach to learn Rust would be to first read through the book for the basic feel and syntax of Rust. Then during implementation, I would watch YouTube tutorials how other people implement similar features in Rust which would better explain their reasoning for the implementation choices they make. Overall, these resources will always be revisited during implementation.

Kevin: My preferred method of learning is to get into the book. I like to figure out how exactly the technology works and what happens under the hood. This gives me a deeper understanding and refines my approach to coding in Rust with the best practices. One of the key benefits of the book is the project section. One of the best ways for me to learn is to delve right in and get into a project. I'm confident that through this method, I'll be able to pick up the language and be comfortable working with it on the capstone project.

3. All members of the team would need to be knowledgeable about network communication protocols such as HTTP and Web Sockets. These protocols are needed to enable interaction between the front-end and back end, as well as integration with the database. Additionally web sockets would be essential for implementing the live editing features as well as the chat feature for this project.

   To be able to learn about these protocols, we can either read articles on how these protocols work, watch YouTube tutorials or look for projects and repositories that use these protocols.

   Faiq: To be able to learn about these protocols, I will be reading up on articles on how to use these protocols and how to implement them with a project. Since the project might be language specific and would not give me a full understanding of the topics.

Veerash: To learn about these protocols I will watch YouTube tutorials on network communication protocols. This is because I learn the best by watching other people, thus it complements my learning style. I will also look for other projects and repositories to further learn how others are implementing these protocols.

8.2 Symbolic Parameters

The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

$EASINESS\_RATING = 7/10$
$RESPONSIVENESS\_RATING = 7/10$
$COMPILE\_TIME = 60$
$LEARNING\_TIME = 30$
$SIMPLICITY\_RATING = 8/10$
$NUM\_COLOURS = 4$
$UPTIME = 99$