Table 1: Revision History

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

Development Plan
ProgName

Team #, Team Name
Student 1 name and macid
Student 2 name and macid
Student 3 name and macid
Student 4 name and macid

[Put your introductory blurb here. —SS]

1 Team Meeting Plan

The team will meet twice a week on Tuesday and Thursday evenings between 6:00 PM and 7:30 PM. The length of meeting time may be adjusted according to the demands of the upcoming deliverable, setbacks and points of discussion. The meetings will take place primarily in the engineering library meeting rooms. Members may join through Discord or Messenger call or chat, and some meetings may be held completely virtually when appropriate.

1.1 Rules for Agenda

The rules for the agenda for all team meetings are as follows:

1. The members will alternate being the designated note-taker and chair for each meeting
2. Meeting minutes will be taken accordingly
3. The meeting agenda will focus on planning for the next milestone, agreements for distribution of the work, discussion of any pain points or concerns from any members, and review of current progress as necessary
4. All members are expected to attend meetings either in person or virtually
5. Members should prepare for meetings by noting down their current progress, roadblocks and any concerns that have arisen prior to the meeting
6. The meeting will primarily be lead by the chair of each meeting, however the remaining members are expected to contribute to discussions and provide insight

7. Meetings should conclude with decisions being made regarding the topics discussed

2 Team Communication Plan

The team will primarily communicate in person as most meetings will occur in person. The secondary avenue of communication will be through Messenger text chat or Discord voice chat. These may be used to address critical issues as they occur outside of meeting times, or issues and concerns that may be resolved quickly and efficiently in a virtual manner.
The codebase and documentation will be managed through GitHub. Code quality reviews and tasks will assigned and communicated through GitHub comments and tasks. Members are expected to make appropriate merge and pull requests when contributing to the repository.

3 Team Member Roles

4 Workflow Plan

- How will you be using git, including branches, pull request, etc.?
- How will you be managing issues, including template issues, issue classificaiton, etc.?

5 Proof of Concept Demonstration Plan

To prepare for the proof of concept demonstration, we will be focusing firstly on getting the text editor where users can see eachother cursors and contributions in real time set up, since this will be the most difficult part of the project. Since having live edits on a single file would require the combined use of a rest api server and web sockets, and on a lower level would require concurrency to handle reads and writes, this would be the most difficult process in terms of both implementing and testing. In comparison the remaining aspect such as integrating with github and compiling the latex code can easily be done with api calls and libraries. Another complexity that may arise is from the Rust's ownership model in a more complex project since the team only has a limited amount of experience with programing in Rust.

In terms of testing, aside from the portion related to the concurrency aspect of the project, the remaining testing should be relatively straight forward and will use common and easy to use testing frameworks.

Installing libraries would be handled by Rust's cargo build tool and would not be difficult and only common libraries such as rocket and the default web-sockets library would be needed for the web server and web sockets.

Portability will also not be a concern for this project as we would be using docker to containerize all modules of the app and have a docker compose to easily run the modules. All machines that support docker should be easily able to run this project.

To demonstrate that the risk can be overcome we will focus on creating a preliminary version of the core functionality of real time multi user editing first, without paying much attention to the UI and the other remaining features. This will help us gauge how feasible this implemention is early on in the project in case we need to redefine some of the reqruirements.

If the risks are proven to be real, we can redefine the project by switching languages to Java and/or to disclude the real time editing by multiple users and instead have the updates only occur in the document once a user makes a commit. This change will allow us to use the memory management model of Java that we are more familiar with and/or remove the need of web sockets abd concurrency which is the biggest risk factor. Instead the focus for the project can then be shifted to some of the strech goals such as adding in features to make working with images and tables easier to work in latex.

6 Technology

- This project will be using Javascript, HTML, CSS for the frontend, and Rust for the backend
- We will be using Clippy (the default linter for Rust) for Rust and jslint for Javascript
- For the unit testing frameworks we will be using the default testing library that comes with Rust
- We will be using the bullt in code coverage utlity that comes with Cargo, the Rust compiler
- Basic CI/CD will be setup for running unit testing and deploying using terraform
- No performance measuring tools will be used, as we do not see a need for them yet
- We will need the basic reactjs library for the frontend, Rocket for the backend web server, and websockets library for the websockets

- For the database we will be using MongoDB, since our project does not require complex relationships
- The IDE we will be using is VSCode
- The build tool we will be using is Cargo, which is used for both compiling and testing Rust code
- We will not be using any hardware aside from a VPS (virtual private server) from digital ocean
- We will be using rustdoc for documentation of the code

7 Coding Standard

Code written should conform to the official rust style guideline, found at: https://doc.rust-lang.org/1.0.0/style/README.html and to google's JavaScript style guide, found at: https://google.github.io/styleguide/jsguide.html.
Classes and methods are to be named using *PascalCase*, and variables and parameters are to be named using *camelCase*.
The goal is to maintain readability and ease of maintenance for any other developers.

8 Project Scheduling

8.1 Scheduling software

The project schedule is organized as epics and issues in the team's Zenhub board. The epics are based of major milestones such as POC, Revision 0, Revision 1, etc. The issues are various tasks associated to these epics.

8.2 Task breakdown

The big tasks are broken down to issues which can be completed by the individual developers in 1-5 days any bigger tasks will be needed to be broken down. There will be some tasks such as the documents which will be worked on by multiple developers at once.

8.3 Task assignment

The developer assigned to each of the task will be assigned based on their proficiency related to the task, interest and available capacity. For example, one of our developer is extremely proficient at working with front end, thus will work on tasks related to front end. Some developers that are really interested in gaining experience in certain subjects will be assigned tasks related to that subject. Lastly, leftovers tasks will be assigned to developers who have leftover capacity.