Underbranch

by

The Tater Tots

Ben Knobloch, Lauren Kibalo, Marcos Morales, Noah Spina

Stevens.edu

March 6, 2025

Ben Knobloch, Lauren Kibalo, Marcos Morales, Noah Spina
Stevens.edu

Underbranch

The Tater Tots
Ben Knobloch, Lauren Kibalo, Marcos Morales, Noah Spina
Stevens.edu

This document provides the requirements and design details of our project Underbranch. The following table (Table 1) should be updated by authors whenever major changes are made to the architecture design or new components are added. Add updates to the top of the table. Most recent changes to the document should be seen first and the oldest last.

Table 1: Document Update History

Date	**Updates**
02/27/2025	BCK: • Research portable version of Docker • Auto-versioning on GitHub LSK: • Updated Weekly Report 19 (2.1) • Update Github Kanban board MAM: • Added CRC cards NJS: • Added Development Requirement 2 (See $reqkDev_2$) • Created Development Use Case Description (See UC_5) • Added activity diagram for use case 5 (See dsnActivity5 (*Figure* 8.9)) • Updated references between development requirements, development use case, and activity diagram

Table 1: Document Update History

Date	Updates
02/20/2025	BCK: • Fixed bug where the system would not open docker images when needed LSK: • Updated Weekly Report 18 (2.2) • Worked on refining our current use case MAM: • Fixed bug where the system would not open docker images when needed NJS: • Removed two use case diagrams that were out of the projects scope • Added a development chapter (See 10) • Renamed chapter files to include the chapter number in the beginning
02/13/2025	BCK: • Updated Weekly Report 17 (2.3) • Fixed issues with the build environment of Underbranch that prevented correct launching and made improvements to the application launch workflow LSK: • Updated Weekly Report 17 (2.3) • Implement remodeled Use Case Diagram (See dsnUseCase3 (*Figure* 6.3)) MAM: • Updated Weekly Report 17 (2.3) • Fixed some app compilation errors that prevented desktop Underbranch from doing anything NJS: • Updated Weekly Report 17 (2.3) • Updated documentation with consistent, clear, and concise naming and descriptions for figures, diagrams, use cases, and requirements • Updated Use Case Diagram to reflect our project more accurately (See dsnUseCase1 (*Figure* 8.4)) • Added a requirement for using Underbranch to install Overleaf (See $reqkFunctional_5$)

Table 1: Document Update History

Date	Updates
02/06/2025	BCK: • Updated Weekly Report 16 (2.4) LSK: • Updated Weekly Report 16 (2.4) MAM: • Updated Weekly Report 16 (2.4) NJS: • Updated Weekly Report 16 (2.4) • Added diagrams of Overleaf's manual installation process
01/30/2025	BCK: • Updated Weekly Report 15 (2.5) LSK: • Updated Weekly Report 15 (2.5) MAM: • Updated Weekly Report 15 (2.5) NJS: • Updated Weekly Report 15 (2.5)
01/21/2024	BCK: • Updated Weekly Report 14 (2.6) LSK: • Updated Weekly Report 14 (2.6) MAM: • Updated Weekly Report 14 (2.6) NJS: • Updated Weekly Report 14 (2.6)
12/19/2024	BCK: • Updated Preliminary Demo (9) LSK: • Updated Preliminary Demo (9) MAM: • Updated Preliminary Demo (9) NJS: • Updated Preliminary Demo (9)

Table 1: Document Update History

Date	Updates
12/12/2024	BCK: • Updated Weekly Report 13 (2.7) LSK: • Updated Weekly Report 13 (2.7) MAM: • Updated Weekly Report 13 (2.7) NJS: • Updated Weekly Report 13 (2.7)
12/05/2024	BCK: • Updated Weekly Report 12 (2.8) LSK: • Updated Weekly Report 12 (2.8) MAM: • Updated Weekly Report 12 (2.8) NJS: • Updated Weekly Report 12 (2.8)
11/21/2024	BCK: • Updated Weekly Report 11 (2.9) LSK: • Updated Weekly Report 11 (2.9) MAM: • Updated System Architecture in Preliminary Design (8) NJS: • Updated Weekly Report 11 (2.9)
11/14/2024	BCK: • Updated Preliminary Design (8) LSK: • Updated Weekly Report 10 (2.10) MAM: • Updated Preliminary Design with deployment diagram (8) • Successfully used ElectronJS to run Overleaf with a Docker container NJS: • Updated Weekly Report 10 (2.10) • Added Activity Diagrams to Chapter 6 (6)

Table 1: Document Update History

Date	Updates
11/07/2024	BCK: • Updated Weekly Report 9 (2.11) LSK: • Updated Weekly Report 9 (2.11) MAM: • Updated Weekly Report 9 (2.11) NJS: • Updated Weekly Report 9 (2.11)
10/31/2024	BCK: • Updated Weekly Report 8 (2.12) LSK: • Updated Weekly Report 8 (2.12) MAM: • Updated Weekly Report 8 (2.12) NJS: • Updated Weekly Report 8 (2.12)
10/23/2024	BCK: • Added requirements to Chapter 5. • Added use case overviews and specifications to Chapter 6 and created Use Case Diagram 3. • Ran a local version of Overleaf. LSK: • Updated Stakeholders Chapter (See 7) MAM: • Updated Weekly Report 7 (2.13) • Added use cases to Chapter 6 NJS: • Added requirements to Chapter 5. • Added use cases to Chapter 6 • Updated Weekly Report 7 (2.13)

Table 1: Document Update History

Date	Updates
10/17/2024	BCK: • Updated Chapter 5. • Worked on compiling TeXnic Center for Windows 11. LSK: • Updated the ”Retrospective,” ”Looking Forward,” and ”Action Items” sections of Weekly Report (See 2.14) MAM: • Researched and attempted to compile TeXnicCenter NJS: • Updated the ”Action Items” sections of Weekly Report (See 2.14)
10/08/2024	BCK: • Updated the ”Action Items” section of Weekly Report (See 2.15) • Added functional requirements (See 5.2) • Added system requirements (See 5.3) LSK: • Updated the ”Retrospective” and ”Looking Forward” sections of Weekly Report (See 2.15) MAM: • Updated the ”Functional Requirements” and ”System (Contraints) Requirements” sections of the Requirements (See 5) NJS: • Updated the ”System Requirements,” ”Quality Requirements,” and ”Business Requirements” sections in User Requirements (See 5) • Updated Key Concepts in Chapter 5 (See 5)

Table 1: Document Update History

Date	Updates
10/01/2024	BCK: • Updated the ”Software”, ”Hardware”, ”Roles and Responsibilities”, ”Method”, ”Communication Plan”, ”Testing Policy/Plan”, ”Risks”, and ”Assumptions” sections of Development Plan (See 4) LSK: • Updated the ”Action Items” section of Weekly Report 4 (See 2.16) • Updated the ”Roles and Responsibilities” and ”Method” sections of Development Plan (See 4) MAM: • Updated ”Hardware”, ”Virtual and Real Workspace”, ”Heartbeat Meetings”, ”Status Meetings”, ”Issues Meetings”, ”IRB Protocol”, ”Required Resource and Budget”, and ”Documentation Plan” (See 4) NJS: • Updated the ”Introduction”, ”Roles and Responsibilities”, ”Software”, ”Hardware”, ”Timeline and Milestones”, and ”Risks” sections of the Development Plan (See 4) • Updated ”Action Items” in Weekly Report 4 (See 2.16)
09/26/2024	BCK: • Updated the ”Action Items” section of Weekly Report 3 (See 2.17) LSK: • Updated the ”Issues, Risks, and Mitigation” section of Weekly Report 2 (See 2.17) MAM: • Updated the ”Looking Forward” section of Weekly Report 3 (See 2.17) NJS: • Updated the ”Retrospective” section of Weekly Report 3 (See 2.17)
09/19/2024	BCK: • Created the ”Retrospective” section of Weekly Report 2 (See 2.18) • Created the ”Action Items” section of Weekly Report 2 (See 2.18) • Created the ”Issues, Risks, and Mitigation” section of Weekly Report 2 (See 2.18) LSK: • Organized naming structure throughout document • Updated the ”Retrospective” section of Weekly Report 2 (See 2.18) MAM: • Added names to project title NJS: • Updated the ”Looking Forward” section of Weekly Report 2 (See 2.18)

Table 1: Document Update History

Date	Updates
09/17/2024	BCK: • Created Team Declaration: Mission Statement (See 3.2) • Updated Team Declaration: Key Constraints (See 3.4) LSK: • Created Team Declaration: Team Overview (See 3.1) • Updated Team Declaration: Key Constraints (See 3.4) MAM: • Created Team Declaration Chapter (Chapter 3). • Updated Team Declaration: Key Drivers (See 3.3) NJS: • Updated Team Declaration: Key Drivers (See 3.3) • Fixed spacing in Team Declaration (See 3)
09/12/2024	BCK: • Updated ”Action Items” and ”Issues, Risks, and Mitigation” in Weekly Report 1 (See 2.19) LSK: • Updated ”What We Did the Past Week” and ”Issues, Risks, and Mitigation” in Weekly Report 1 (See 2.19) MAM: • Updated ”What We Did the Past Week” in Weekly Report 1 (See 2.19) NJS: • Added Stevens Logo to front page
09/11/2024	NJS: • Updated ”What We Did the Past Week” in Weekly Report 1 (See 2.19)
09/10/2024	BCK: • Updated Document Update History. LSK: • Created group project manual. MAM: • Imported introduction template • Created weekly report 1 (See 2.19) NJS: • Removed excess pages from manual and cleaned up document

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

– Ben Knobloch, Lauren Kibalo, Marcos Morales, Noah Spina

1.1 Benjamin Knobloch

Hi! My name is Ben Knobloch, and I am a 4/4 Software Engineering major. I'm experienced in a variety of languages and environments, but my favorite languages are Java and Python. I'm the News Editor for the campus paper, Vice President for Engineers Without Borders SIT, and a member of the Honor Board. In my free time, I enjoy photography, reading, and playing board games with my friends. This summer, I interned at the DOE National Laboratory in Los Alamos, New Mexico and traveled to Peru for Engineers Without Borders, where I got to pet a lot of alpacas.

1.2 Lauren Kibalo

Hello, my name is Lauren Kibalo and I am a 4/4 Software Engineering major with a computer science minor. At Stevens, my involvements include Software Engineering Club, Peer Leader, Tour Guide, Teaching Assistant for Introduction to Programming and Logarithmic Thinking, Society of Women Engineers, Women's Club Volleyball, Greek life, and Student Advisory Boards. In my free time, I enjoy being active and trying new things!

1.3 Marcos Morales

Hi, my name is Marcos Morales and I am a 4/4 software engineering major here at Stevens. My favorite programming languages are JavaScript and Python because of the convenient syntax. As for my interests outside of programming, I enjoy anything Star Wars and Nintendo related. Last year I moved to Hoboken after commuting to Stevens for 2 years from New York so I'm now able to hangout with friends and be more involved in extracurricular activities. Something new I did over the summer was that I went kayaking for the first time.

1.4 Noah Spina

Hello, my name is Noah Spina and I am a highly motivated 4/5 Software Engineer pursuing a Minor in Computer Science and a Masters in Computer Science. Specifically, I am interested in Machine Learning and hope to develop ML models to solve difficult problems. At Stevens, I was the Vice President of Communications at Sigma Phi Epsilon during 2024 and am currently the Founder and the past President of The Pickleball Club on campus. Other hobbies of mine include playing video games, playing pickleball, and cooking. Some fun facts about me: I have 3 dogs and 4 cats and my favorite color is white.

Chapter 2

Weekly Reports

2.1 Week Report 19 (2/27/2025)

2.1.1 Retrospective

This week the team reviewed the feedback received from our presentation last week. We appreciated our mostly positive feedback, but want to continue improving our diagrams. Additionally, we looked into a portable version of Docker so that our application would not need to rely on Docker Desktop. There was not much success on this front, but we will continue to research the issue.

2.1.2 Looking forward

Next week, we will work on the continual improvement of our diagrams and focusing on installer scripts. Specifically, we plan to improve the Developer chapter ((See 10) of our documentation by creating more diagrams that focus on the developer. Finally, we want to ensure that our installer scripts work effectively with no bugs and provide some improvements to the current versions.

2.1.3 Action Items

Item	Priority Points
Resolve Overleaf compilation issues for all team members	5
Continue development of visual installer	8
Implement portable version of Docker	11
Refine additional use cases for development and downloading packages	5

Table 2.1: Week 17 Action Items

2.1.4 Issues, Risks, and Mitigation

Risks:

1. Ongoing compilation issues

Mitigation:

1. Troubleshoot compilation process
2. Document the compilation process thoroughly

2.1.5 Image of Current Tasks

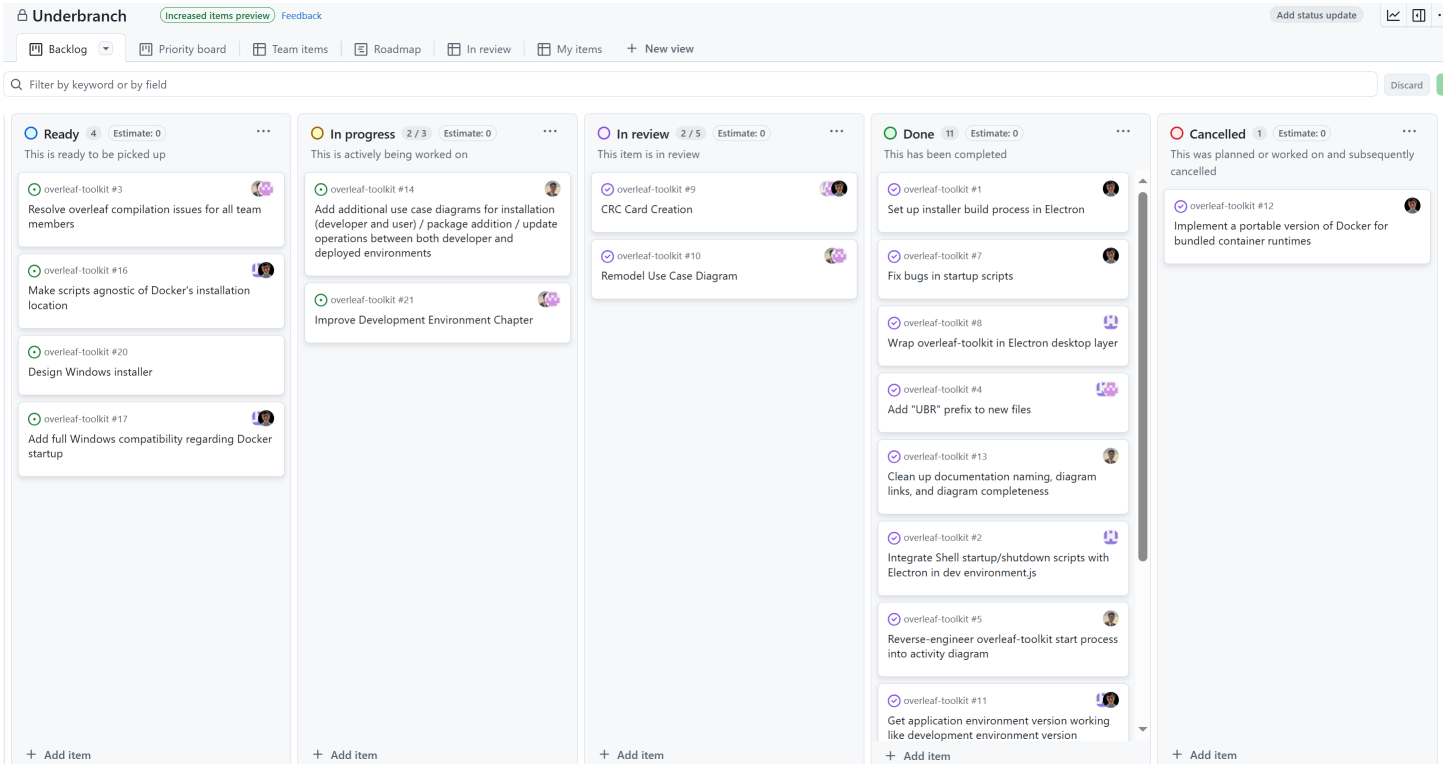

Figure 2.1: Our Kanban Board as of Week 19.

2.1.6 Current Diagram

The diagram we worked on this week is a new activity diagram for setting up the development environment for Underbranch.

See dsnActivity5 (*Figure* 8.9)

2.2 Week Report 18 (2/20/2025)

2.2.1 Retrospective

This week the team focused on creating a presentation to update our professors and peers on our current progress. We were able to showcase our current use cases, diagrams, CRC cards, and a working demo of the desktop application. Additionally, the demo included the built app being able to create and start the images.

2.2.2 Looking forward

Next week, we want to continue working on improving our use case diagrams, ensuring that all steps are well documented and align with the overall system architecture. Additionally, we want to continue addressing the desktop application's inability to access the Docker path in certain environments. Resolving this will be a critical step toward ensuring deployment across different operating systems without needing the user to perform an additional manual process.

2.2.3 Action Items

Item	Priority Points
Resolve Overleaf compilation issues for all team members	5
Continue development of visual installer	8
Implement portable version of Docker	11
Refine additional use cases for development and downloading packages	5

Table 2.2: Week 17 Action Items

2.2.4 Issues, Risks, and Mitigation

Risks:

1. Ongoing compilation issues

Mitigation:

1. Troubleshoot compilation process
2. Document the compilation process thoroughly

2.2.5 Image of Current Tasks

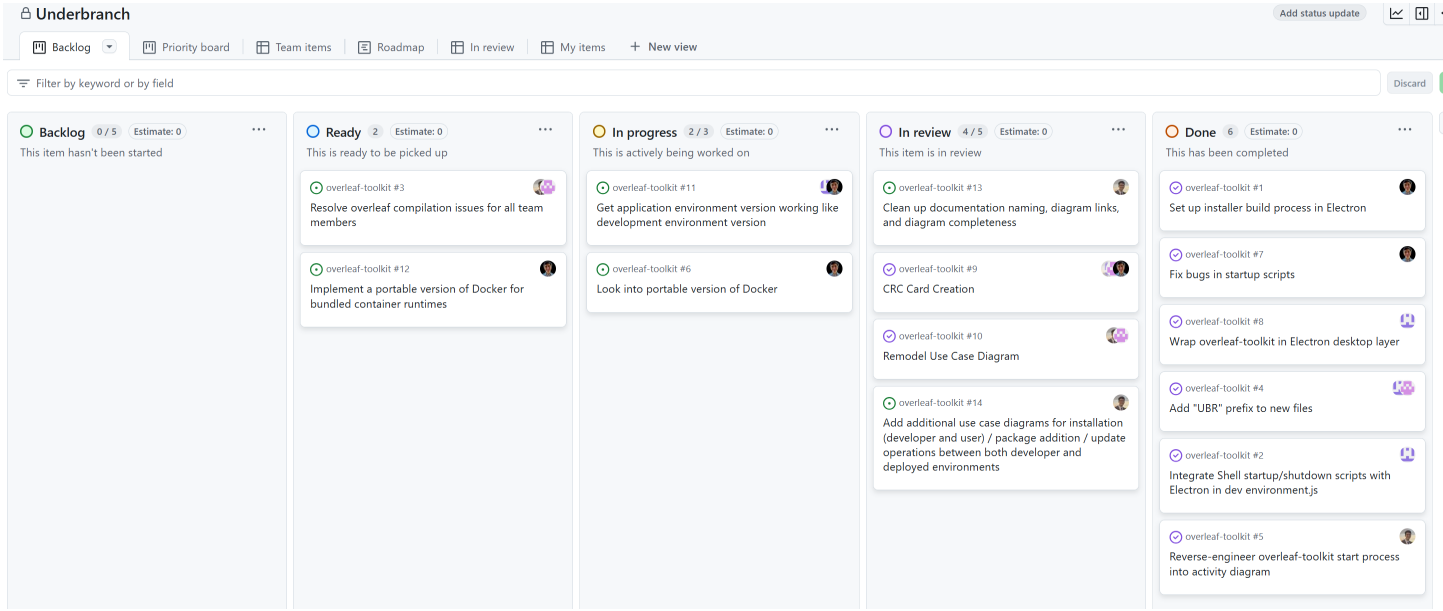

Figure 2.2: Our Kanban Board as of Week 18.

2.2.6 Current Diagram

This week we worked to refine our deployment diagram and by adding the named of Underbranch's shell scripts.

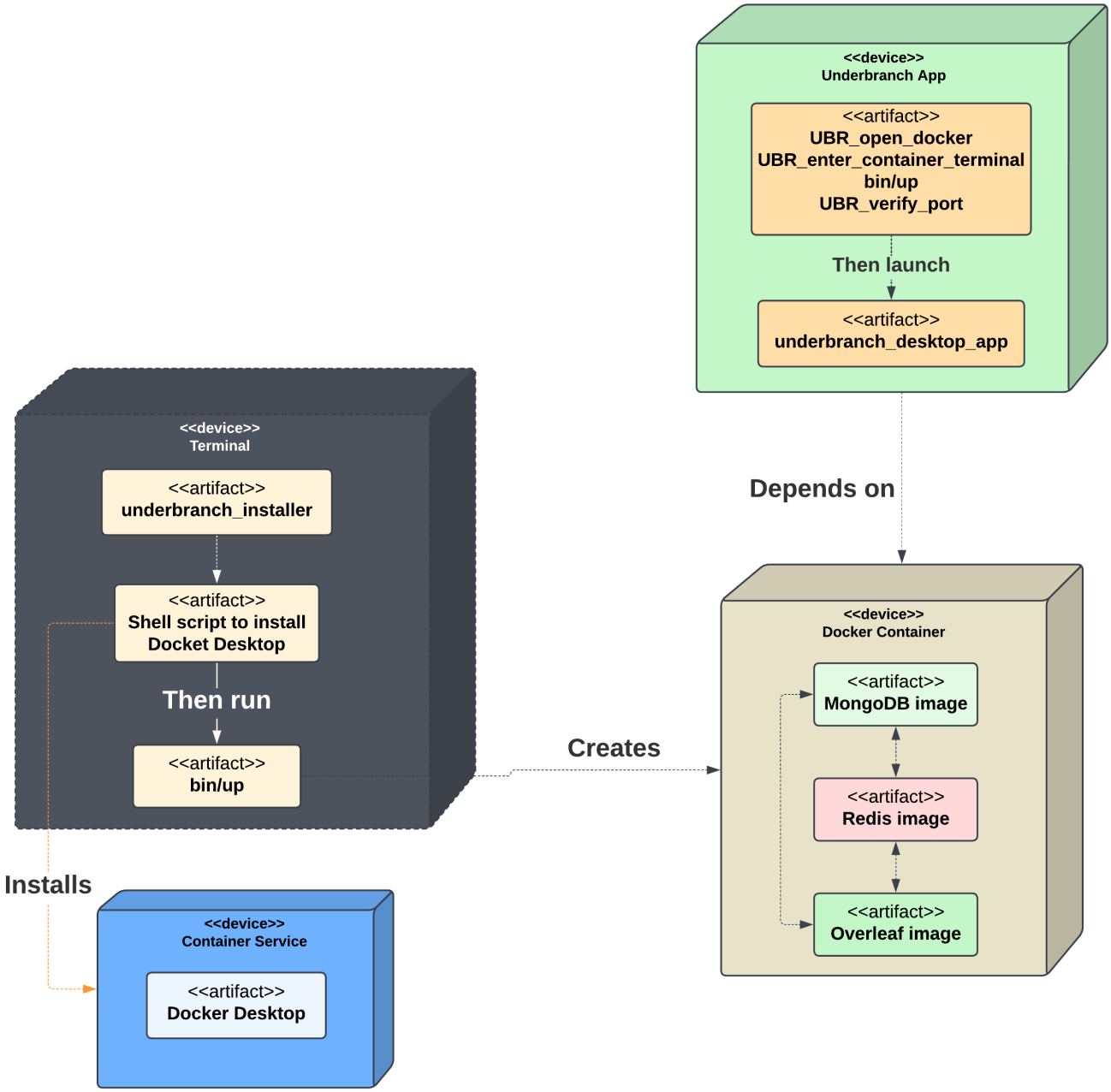

Figure 2.3: Updated deployment diagram

2.3 Week Report 17 (2/13/2025)

2.3.1 Retrospective

This week, the team made significant progress in refining our CRC cards, ensuring they accurately represent system components and interactions. We addressed inconsistencies, improving clarity for future development. Additionally, we successfully integrated our updates into the developer environment, allowing for smoother execution of the application. A major milestone was the completion of a working demo of the compiled desktop app of Underbranch During internal testing, we were able to launch and configure the Overleaf container automatically with minimal setup, demonstrating key functionalities of our tool. This achievement puts us in a strong position to refine the compiled app further based on the small issues that still need to be solved and testing in different environments.

2.3.2 Looking forward

Next week, the team will focus on preparing a presentation to showcase our progress, including the refined CRC cards and the working demo of the desktop application. Additionally, we will develop a detailed use case for the installation and implementation process, ensuring that all steps are well documented and align with the overall system architecture. This will help refine our approach to deployment and user onboarding. A major technical challenge we plan to address is the desktop application's inability to access the Docker path in certain environments with it sometimes working when the path is explicitly specified. We will investigate the root cause of this issue and explore potential solutions. Resolving this will be a critical step toward ensuring deployment across different operating systems without needing the user to perform an additional manual process.

2.3.3 Action Items

Item	Priority Points
Resolve Overleaf compilation issues for all team members	5
Continue development of visual installer	8
Implement portable version of Docker	11
Add use case for installing packages	13
Update documentation with consistent names for use cases, correct and complete links to diagrams, and clearer diagrams	15
Add additional use case diagrams for installation (developer and user) / package addition / update operations between both developer and deployed environments	8

Table 2.3: Week 17 Action Items

2.3.4 Issues, Risks, and Mitigation

Risks:

1. Ongoing compilation issues

Mitigation:

1. Troubleshoot compilation process
2. Document the compilation process thoroughly

2.3.5 Image of Current Tasks

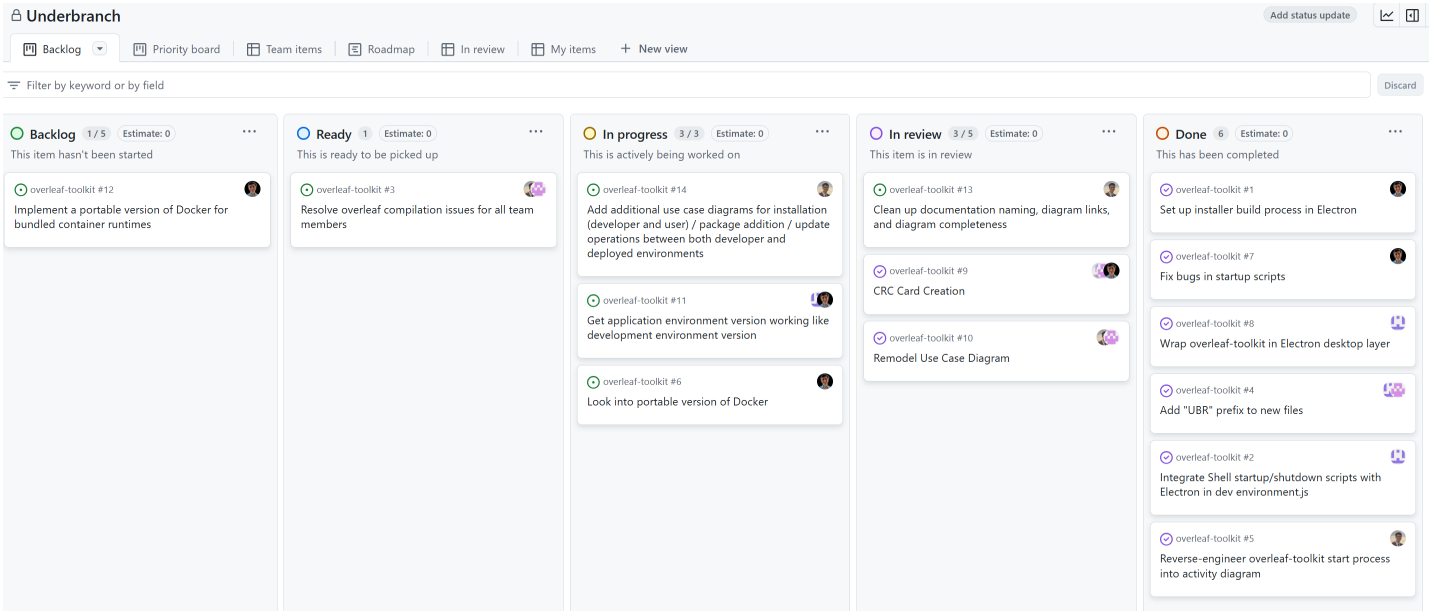

Figure 2.4: Our Kanban Board as of Week 17.

2.3.6 Current Diagram

This week we worked on use case diagram that focuses on the specifics of Underbranch installation.

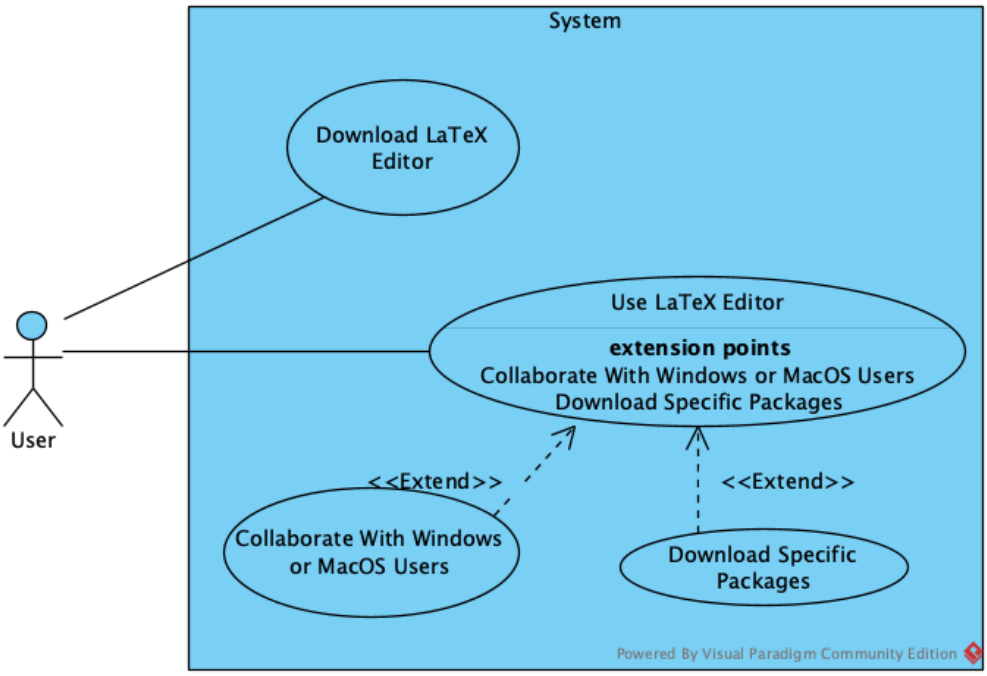

Figure 2.5: Use case diagram.

2.4 Week Report 16 (2/6/2025)

2.4.1 Retrospective

This week the team created CRC cards for the Shell scripts created by us and some of the scripts that are part of the Overleaf Toolkit. Using these scripts in a developer environment, the application automatically opens Docker and starts the Overleaf container which previously had to be done manually. We attempted to test this in a production environment, which involves compiling the project into a desktop application, but we ran into some issues due to system permissions. Additionally, the team updated existing use case diagrams to improve cohesiveness and simplicity throughout our documentation.

2.4.2 Looking forward

Next week, the team will continue refining the CRC cards to ensure clarity and accuracy in representing system components. Additionally, efforts will be directed toward getting the application fully operational in the application environment. Once stable, we will shift focus to compiling it into a desktop application for broader usability. Our goal with the initial development of a visual installer will be to streamline setup for our end users.

2.4.3 Action Items

Item	Priority Points
Resolve Overleaf compilation issues for all team members	5
Continue refining CRC cards	5
Continue development of visual installer	8
Implement portable version of Docker	11
Add use case for installing packages	13
Update documentation with consistent names for use cases, correct and complete links to diagrams, and clearer diagrams	15
Add additional use case diagrams for installation (developer and user) / package addition / update operations between both developer and deployed environments	8

Table 2.4: Week 16 Action Items

2.4.4 Issues, Risks, and Mitigation

Risks:

1. Ongoing compilation issues

Mitigation:

1. Troubleshoot compilation process
2. Document the compilation process thoroughly

2.4.5 Image of Current Tasks

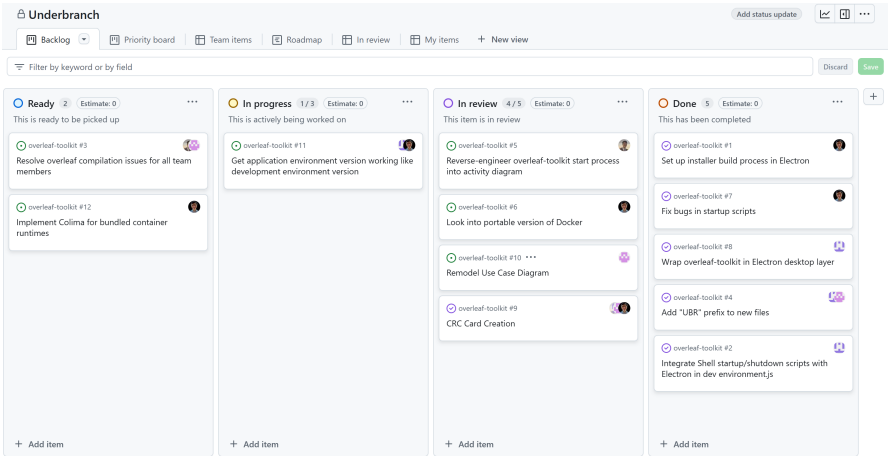

Figure 2.6: Our Kanban Board as of Week 16.

2.4.6 Current Diagram

This week, we worked on improving our use case diagrams. We updated the use case for an individual user and for multiple users interacting with different operating systems.

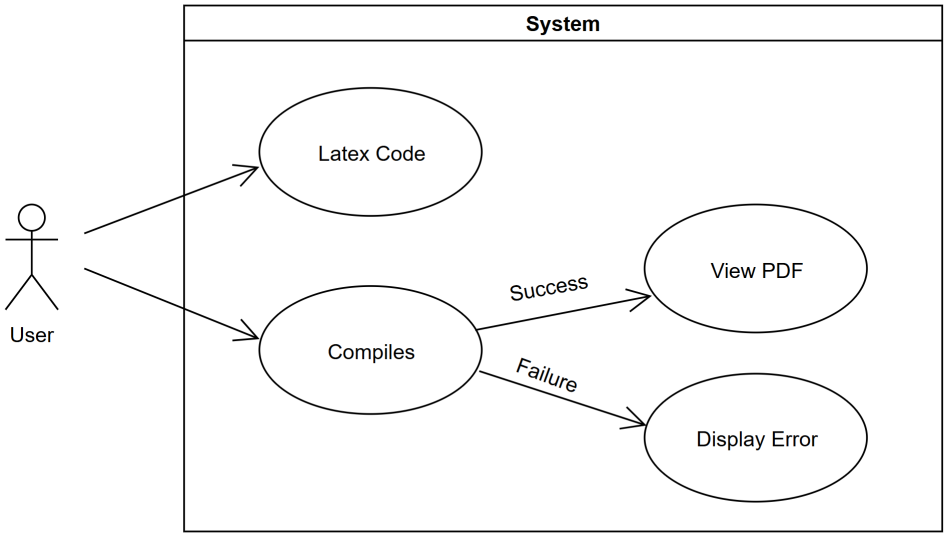

Figure 2.7: Use case diagram for 1 person use of the application (See UC_1.)

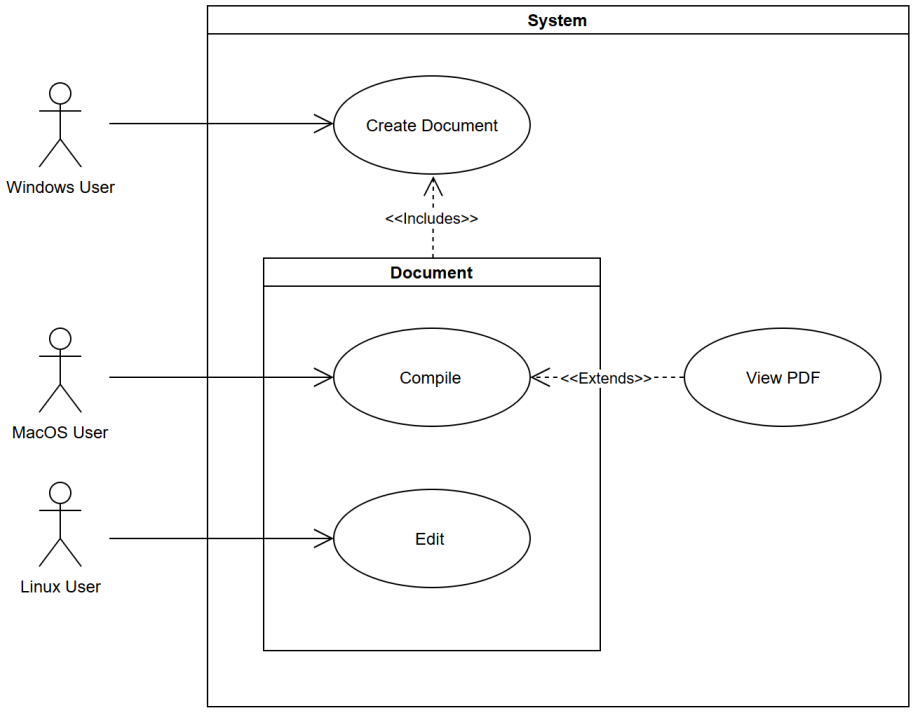

Figure 2.8: Use case diagram for multiple users interacting with the system at the same time from different operating systems (See UC_3.)

2.5 Week Report 15 (1/30/2025)

2.5.1 Retrospective

This week, we successfully completed an activity diagram outlining the Overleef-Toolkit startup process. Additionally, we explored potential alternatives to Docker for running Underbranch in different environments. Our research led us to the Colima application, which offers a lightweight, portable solution for Mac and Linux systems. While Colima presents a promising alternative to Docker, it does not support Windows, limiting its opportunity to be a universal solution.

2.5.2 Looking forward

The team will continue troubleshooting development environment issues and investigating alternative implementation strategies to address these challenges. Furthermore, we will refine our startup scripts within the Electron.js framework to ensure smoother execution. Additionally, we aim to enhance the Overleaf integration process and explore solutions that improve compatibility across all the environments of our team members.

2.5.3 Action Items

Item	Priority Points
Resolve Overleaf compilation issues for all team members	9
Begin development of visual installer	8
Establish nomenclature, including the ”UBR” prefix, for all necessary files	9
Implement portable version of Docker	15
Create CRC cards to track class responsibilities	3
Fix Use Case 4 to include ”and install”	1

Table 2.5: Week 15 Action Items

2.5.4 Issues, Risks, and Mitigation

Risks:

1. Ongoing compilation issues

Mitigation:

1. Troubleshoot compilation process
2. Document the compilation process thoroughly

2.5.5 Image of Current Tasks

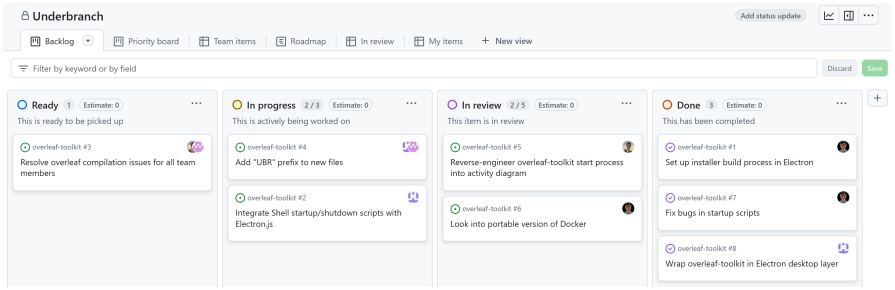

Figure 2.9: Our Kanban Board as of Week 15.

2.5.6 Current Diagram

This week, we worked on the activity diagram of the overleaf-toolkit start process.

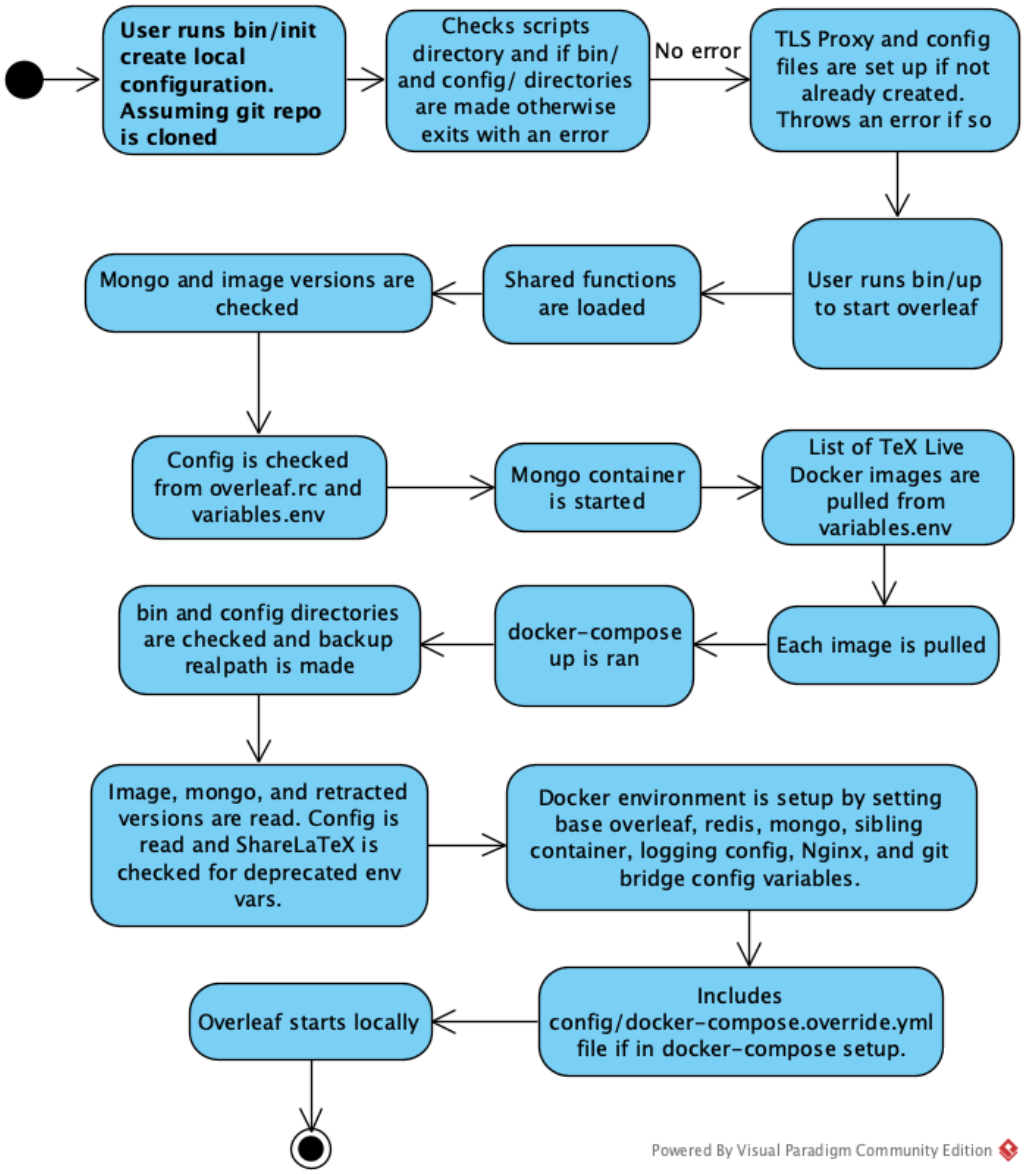

Figure 2.10: Activity diagram for Overleaf's Installation and Shell Scripts (See UC_4.)

2.6 Week Report 14 (1/21/2025)

2.6.1 Retrospective

Over the course of last semester we began planning and the preliminary implementation of Underbranch. Up to this point we have created an ElectronJs wrapper to a local instance of Overleaf hosted inside of Docker. The team also figured out how to install LaTeX packages by executing a TeX Live command inside of the sharelatex container which can be done via Docker Desktop or through using Docker commands to get inside of an existing Docker container.

This week we made progress in several things. First, there was a Docker issue preventing Docker from being used by Mac users. This was a problem by Docker and was fixed as of January 23rd.

Additionally, an activity diagram was made that shows the process of the ElectronJS installer (See Figure 8.11 in Section 8.3.3).

2.6.2 Looking forward

Our highest priority heading into the new semester is to resolve long-standing bugs preventing two of our group members from setting up a proper development environment. With regard to development objectives, the group intends to automate startup scripts within the Electron.js framework. The group will also begin development work on a visual MacOS installer to wrap these startup scripts.

2.6.3 Action Items

Item	Priority Points
Integrate Shell startup/shutdown scripts with Electron.js	8
Resolve Overleaf compilation issues for all team members	21
Continue integration of Overleaf with Electron.js	15
Begin development of visual installer	8
Add the ”UBR” prefix to new files	9
Reverse Engineer Overleaf installation into an activity diagram	2
Look into portable version of Docker	2

Table 2.6: Week 14 Action Items

2.6.4 Issues, Risks, and Mitigation

Risks:

1. Ongoing compilation issues

Mitigation:

1. Troubleshoot compilation process
2. Document the compilation process thoroughly

2.6.5 Image of Current Tasks

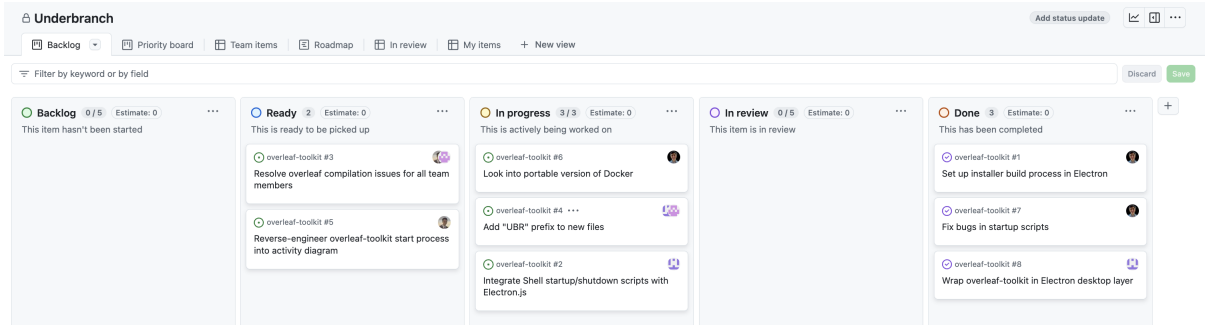

Figure 2.11: Our Kanban Board as of Week 14.

2.6.6 Current Diagram

This week, we will focus on revising the ElectronJS Installer Figure (See Figure 8.11 in Section 8.3.3). We are going to define the tasks or scripts that handle each task and be more specific with their functions.

2.7 Week Report 13 (12/12/2024)

2.7.1 Retrospective

This week, the team has been working diligently to solve compilation and runtime issues to make sure that every team member can run Overleaf. With the progress made, the team should be on the same page with the compilation by the beginning of the next semester. However, the team is currently working to address these compilation errors.

In addition, the team has been working to finalize an installer user interface and workflow. This will allow for easier visualization of the team's ideas and goals.

Last-minute exploration of the docker containers proved quite useful in understanding the system. The team was able to determine how to download Latex packages. This is done by accessing the console of the sharelatex container in docker and running a specific command to download packages. This is a massive leap in progress, as we can now download specific packages to use for the compilation of any latex project. With this, users will be able to run Underbranch locally even without Wi-Fi once the packages are installed.

2.7.2 Looking forward

Looking forward, the team will finalize the installer using ElectionJS. After that, the team will explore how to integrate package updates locally. With that done, our main focus will shift to developing the local user collaboration feature. The team has also explored ideas of running our service without Docker, but that would require an immense amount of refactoring and a deep understanding of how overleaf works.

2.7.3 Action Items

Item	Priority Points
Present Project Demo	8
Resolve Overleaf compilation issues for all team members	21
Continue integration of Overleaf with Electron.js	15

Table 2.7: Week 13 Action Items

2.7.4 Issues, Risks, and Mitigation

Risks:

1. Ongoing compilation issues
2. Understanding and utilizing ElectionJS

Mitigation:

1. Troubleshoot compilation process
2. Document the compilation process thoroughly

2.8 Week Report 12 (12/05/2024)

2.8.1 Retrospective

This week the team continued refining our approach to integrating Overleaf within an Electron.js environment. Additionally, the team has addressed the issues related to compiling Overleaf for two of our team members who are still encountering setup challenges. While Electron.js has been successfully integrated into the design for some members, the focus has shifted towards resolving these compilation issues to ensure consistency across all team members' development environments.

Additionally, the team initiated versioning at 0.1.commit-id and documented the additions to the starting scripts that fix issues with the upstream overleaf-toolkit repository. This marks the first release of Underbranch.

2.8.2 Looking forward

The team's primary focus remains on resolving the remaining compilation issues for Overleaf. Once these are addressed, we plan to proceed with the creation of a deployment diagram to better visualize the infrastructure. In addition, we will begin implementing our logo throughout the application to enhance the user interface of our application.

2.8.3 Action Items

Item	Priority Points
Resolve Overleaf compilation issues for all team members	21
Continue integration of Overleaf with Electron.js	15

Table 2.8: Week 12 Action Items

2.8.4 Issues, Risks, and Mitigation

Risks:

1. Ongoing compilation issues
2. Understanding and utilizing ElectionJS

Mitigation:

1. Troubleshoot compilation process
2. Document the compilation process thoroughly

2.9 Week Report 11 (11/21/2024)

2.9.1 Retrospective

Throughout this week, Team Underbranch made progress in planning and documentation. Multiple diagrams were created including: use Case Four (See 6.1), four activity diagrams, a component diagram, and a state diagram. The team worked on compiling and integrating ElectionJS into the design. Some progress was made, however, there are still roadblocks that the team needs to overcome.

2.9.2 Looking forward

Our highest priority is to establish a working environment, as it exists on two of our group members' machines, on all four development machines. Once that is complete, we intend to continue working within the electron.js infrastructure to implement a smooth desktop application experience that is capable of running Overleaf and its dependent services in the existing Docker environment.

2.9.3 Action Items

Item	Priority Points
Compile Overleaf for whole team	21
Experiment with electron.js	13
Complete presentation and obtain project feedback	5

Table 2.9: Week 11 Action Items

2.9.4 Issues, Risks, and Mitigation

Risks:

1. Ongoing compilation issues
2. Understanding and utilizing ElectionJS

Mitigation:

1. Document the compilation process thoroughly

2.10 Week Report 10 (11/14/2024)

2.10.1 Retrospective

This week, the team focused on creating diagrams, updating our documentation, and continuing to try to resolve compilation errors. All diagrams were able to be created in accordance with our upcoming presentation. These diagrams include a logical view, process view, development view, and physical view. Additionally, we wanted to improve our current documentation. We received feedback on how to refine our current naming structures throughout Overleaf and improve some of our previous diagrams. Finally, the group attempted to resolve some compilation errors but were unable to make much progress.

2.10.2 Looking forward

The team is preparing to present on Tuesday (11/19/2024) to provide updates on our current project progress. We hope to receive valuable feedback to guide us on how to best approach the next steps in our timeline. We constructed a tentative use of ElectionJS for our project to showcase an Overleaf tool in a desktop environment. However, it currently still depends on a separate installation of Docker desktop and the Docker container must be made and running before opening the app. We want to continue experimenting with ElectionJS until we can make the installation process as simple as a ”double-click.”

2.10.3 Action Items

Item	Priority Points
Compile Overleaf for whole team	21
Experiment with electron.js	13

Table 2.10: Week 10 Action Items

2.10.4 Issues, Risks, and Mitigation

Risks:

1. Compilation issues or compatibility problems as more team members attempt to compile the toolkit locally on different systems.
2. Understanding and utilizing ElectionJS

Mitigation:

1. Document the compilation process thoroughly

2.11 Week Report 9 (11/07/2024)

2.11.1 Retrospective

This week, the team focused on running the Overleaf toolkit on our personal machines. By Tuesday (11/05/2024), half of the team successfully compiled the application locally. We then forked the Overleaf toolkit repository and implemented necessary modifications to overcome compilation errors. Additionally, we explored other existing applications designed to edit Overleaf in various ways.

2.11.2 Looking forward

With the Overleaf toolkit now successfully forked and modified, the team will decide how to proceed based on evaluating alternative solutions discovered during research. We aim to streamline our approach by leveraging existing tools or further developing a solution based on our specific project needs. Our next step would be try loading the Docker images for Overleaf into an ElectionJS project to determine if that's a viable solution for our project.

2.11.3 Action Items

Item	Priority Points
Understand Overleaf	21
Compile Overleaf toolkit for whole team	5
Understand installers	7
Experiment with electron.js	13

Table 2.11: Week 9 Action Items

2.11.4 Issues, Risks, and Mitigation

Risks:

1. Compilation issues or compatibility problems as more team members attempt to compile the toolkit locally on different systems
2. Building an installer could be technically complex

Mitigation:

1. Document the compilation process thoroughly
2. Explore existing tools or frameworks for creating installers

2.12 Week Report 8 (10/31/2024)

2.12.1 Retrospective

This week the team continued adjusting to the Overleaf source code as a basis for the project. The team worked to complete the goal of getting Overleaf locally running on personal machines. We successfully were able to compile and run it on one of our machines, so now we are working to access it on all team member machines. This would allow us to start thinking about versioning and working on creating our own product.

2.12.2 Looking forward

Now that we were able to compile and run Overleaf locally the next steps would be to create an installer for Overleaf so it runs as a desktop app instead of a web app. One of the potential solutions could be using Electron JS which is a framework which has you code in JavaScript but lets you create a desktop app installer for your program. Since Overleaf is mostly coded in JavaScript and TypeScript this could be a viable solution

2.12.3 Action Items

Item	Priority Points
Understand Overleaf	21
Continue refining and updating diagrams	11
Understand installers	7

Table 2.12: Week 8 Action Items

2.12.4 Issues, Risks, and Mitigation

Risks:

1. The existing open-source Overleaf codebase will be highly coupled and it will be difficult to add features
2. The existing codebase will be large, making it difficult to model using the techniques we have learned

Mitigation:

1. We will analyze the current codebase to get a better understanding of how Overleaf works along with the most important files
2. We will delegate learning parts of the codebase and work to find the most relevant files to update and add onto for the project

2.13 Week Report 7 (10/24/2024)

2.13.1 Retrospective

Over the past week, our team pivoted from using TeXnic Center for basing our project off of to using the source code of Overleaf. This was mainly due to TeXnic Center having an old codebase that was hard to work with. It uses C++ from 2010 which we were able to find but we couldn't find a version Visual Studio that supports 2010 C++. The team would then have to update the code to work with C++ 2022 which was proving to be significantly more difficult than anticipated. Overleaf, being the industry standard LaTeX editing tool, is a much better application to use for developing a project. The team also created use cases and use case diagrams in Chapter 6, mapping user requirements defined in Chapter 5 to each use case.

2.13.2 Looking forward

We plan to concentrate on identifying key stakeholders and finalize our architecture. Since pivoting from TeXnic Center to Overleaf is a drastic change in source code, our team will need to understand and compile Overleaf source code before developing a strong architecture and plan for our development process. Then in two weeks (11/7/24) we will start milestone 4 - 7, the development phase.

2.13.3 Action Items

Item	Priority Points
Understand and compile Overleaf	21
Create basic architecture diagrams to better understand the development needs	11
Identify key stakeholders and collect their user stories	7

Table 2.13: Week 7 Action Items

2.13.4 Issues, Risks, and Mitigation

No issues, risks, or mitigating actions at this time.

2.14 Week Report 6 (10/17/2024)

2.14.1 Retrospective

Over the past week, our team outlined a comprehensive set of requirements encompassing development, user needs, system specifications, quality standards, and business objectives. We requested feedback from our mentor, Professor Darian, to refine and enhance our existing specifications. Additionally, we are currently awaiting approval for a budget to support the use of a LaTeX editor on Mac machines.

2.14.2 Looking forward

We plan to concentrate on defining use cases that align with the requirements we've previously established. Our approach will prioritize creating use cases we believe fit our application needs and then map these use cases to our existing requirements in our requirements table (see Chapter 5). If there are requirements that to not get paired with a use case, we will most likely remove such requirement based on what we believe should be a focus of our product.

2.14.3 Action Items

Item	Priority Points
Generate use cases	13
Ensure all group members are able to compile LaTeX code on Windows	5
Present next class on project requirements	3

Table 2.14: Week 6 Action Items

2.14.4 Issues, Risks, and Mitigation

No issues, risks, or mitigating actions at this time.

2.15 Week Report 5 (10/08/2024)

2.15.1 Retrospective

After deciding to move forward with a collaborative Latex editor, the group has decided to develop Underbranch. This software will be designed specifically for Stevens users to improve Overleaf functionality. This idea was presented to our fellow peers in a group presentation, and With guidance from Professor Darian, we feel confident this could be a successful project. This week the group has worked on creating requirements and planning out the architecture more in depth.

2.15.2 Looking forward

As we progress, it is important to prioritize our tasks to ensure a structured development process. Our next steps include finalizing the requirements, designing the user interface, and establishing a timeline for implementation. We will also explore opportunities for user feedback to refine our software further. By maintaining communication and collaboration within the team, we aim to deliver a user-friendly platform that meets the needs of our Stevens community.

2.15.3 Action Items

Item	Priority Points
Finalize budget outline	2
Prioritize Group Tasks	5
Generate initial requirements	21
Generate user stories	13
Begin initial documentation and versioning	8

Table 2.15: Week 5 Action Items

2.15.4 Issues, Risks, and Mitigation

No issues, risks, or mitigating actions at this time.

2.16 Week Report 4 (10/01/2024)

2.16.1 Retrospective

After further discussion with the lead cloud engineering manager at Datacor Inc. (Amanda Weiss) for potential projects, our team's needs would not be able to be fulfilled by the company. Amanda mentioned our timeline would not work with their organization along with the complexity of the suggested projects not meeting our needs as a senior design team. We discussed this change with Professor Darian and have decided to choose a new project.

The team discussed thoroughly possible projects for us to pursue. We decided to pivot from the original transpilation idea to one that we would have more support from Stevens faculty. One idea that we considered pursing was a roommate helper that we could re-implement from a previous project, but we wanted to choose a project that was new to all team members and would introduce new technologies and learning opportunities.

2.16.2 Looking forward

The project we are interested in pursuing is a institutional collaborative LaTeX editor. We plan to create a use case where users have the opportunity to the offerings of Overleaf's paid plan for free and have it be locally compiled on an application instead of on proprietary servers. Our users would be the students and faculty of Stevens Institute of Technology to serve our client base and give them the best functionality possible.

2.16.3 Action Items

Item	Priority Points
Obtain a clear description of this Overleaf remake project from our Professor	3
Be prepared to present our project next class and be confident as a team to be successful with this initiative	5
Create a well documented development plan for the year	11

Table 2.16: Week 4 Action Items

2.16.4 Issues, Risks, and Mitigation

Our main challenge now seems to be taking on the responsibility of a new project that may require the use of technologies we are not familiar with. The team agreed to undertake an initiative that could bring a large learning curve. We have discussed using such technologies that none of the team members have used before to improve our skills as software engineers.

2.17 Week Report 3 (9/27/24)

2.17.1 Retrospective

We met with the lead cloud engineering manager at Datacor Inc., Amanda Weiss for potential projects we would work on. Amanda mentioned an API documentation tool that is a automated. We would have to build the tool and integrate it into their pipeline. Maybe every time an API is implemented, documentation is automatically created for that API. She also mentioned something about making something that stores this documentation.

It doesn't necessarily have to be built from scratch. We may need to have dependencies. We would have to map out the dependencies. She mentioned we could use a tool called "Swagger" because ideally we wouldn't write anything from scratch. We will still need to integrate the tool which will be the challenging part.

Another project was the stress testing project. She mentioned "what is the best way to stress test an application." This would involve a project plan and utilizing tools that are already in place. "We have a service that has 15 endpoints, how do we test those endpoints in a way that's scalable and that makes sense." This would definitely require more planning.

2.17.2 Looking forward

Now that we have an idea of the projects that could be worked on with Datacor, we have to decide if we want to pivot from the original transpilation idea to one of the projects Datacor has. Amanda said she will ask her team if they are ok with us taking on this project for them and will try to get back to us by the end of the week. If her team approves of the idea, then another meeting will be scheduled to get more specific requirements for the project so we can determine the complexity and have a better idea of what the finished product should look like.

2.17.3 Action Items

Item	Priority Points
Obtain a clearer description of both potential projects from Datacor Inc. representative	13
Decide whether or not to pivot from an original idea to a Datacor Inc.-supplied problem	21
Schedule a meeting with Datacor Inc. to go over the project complexity and requirements	13

Table 2.17: Week 3 Action Items

2.17.4 Issues, Risks, and Mitigation

Our main challenge remains the need to officially determine the direction of our project. Currently, we are considering a potential partnership with an external company, which introduces certain risks to our project's outcome. To mitigate these risks and prevent any loss of time and resources, we must prioritize effective communication throughout the process.

2.18 Week Report 2 (9/19/24)

2.18.1 Retrospective

This week we locked in our project selection and further refined our idea. Our current plan is to create a developer tool, most likely a plugin, that assists transpilation operations. We hope to integrate machine learning into this tool to increase its capabilities, and potentially make it language-agnostic. We are still finalizing a project idea but for now we are moving forward with transpilation.

2.18.2 Looking forward

Transpilation requires a deep knowledge of coding languages and compilation. To prepare and plan for a project of this calibre, our team will need to do extensive research before we develop a plan for how to move forward. Since this project is outside of the scope of our curriculum, we will need to make sure that a detailed plan of attack is formed before the application starts being made. In the next week, extensive research and developing an action plan will be our highest priority.

2.18.3 Action Items

Item	Priority Points
Refine business objective and user stories	21
Collect project requirements	13
Determine project scope and timeline	13
Create GitHub repository and begin versioning	5

Table 2.18: Week 2 Action Items

2.18.4 Issues, Risks, and Mitigation

Our primary issue and risk is lack of focus in our idea; While we know we want to build an auxiliary tool for developers working in transpilation, we have yet to precisely determine the scope, target use cases, or platform of our eventual product. However, that process has begun.

2.19 Week Report 1 (9/12/24)

2.19.1 Retrospective

In the past week, we formed our team and have been pursuing leads for projects. This includes researching projects that might be a good fit and reaching out to professors and connections for project ideas. After sifting through several ideas, the topic of transpilation stood out. This is the idea of translating one coding language to another. It is understood that a deep knowledge of compilers and machines are necessary for success in this project, but our team is willing to work on learning and implementing something in relation to transpilation. However, some other ideas are still being considered. Finalizing our topic idea will be our main task for the next week.

2.19.2 Looking forward

Next week, we plan to have a concrete project idea that the whole team agrees to move forward with.

2.19.3 Action Items

Item	Priority Points
Determine project topic	21
Collect project requirements	13
Determine project scope and timeline	13
Create GitHub repository	2

Table 2.19: Week 1 Action Items

2.19.4 Issues, Risks, and Mitigation

No issues, risks, or mitigating actions at this time.

Chapter 3

Team Declaration

– Ben Knobloch, Lauren Kibalo, Marcos Morales, Noah Spina

3.1 Team Overview

- Members: Ben Knobloch, Lauren Kibalo, Marcos Morales, Noah Spina
- Group Name: The Tator Tots

3.2 Mission Statement

Many organizations have large codebases written in antiquated or insecure languages. Updating these codebases is time intensive and skill intensive. However, modern automation methods like machine transpilers and large language models are unreliable or hard to verify. Therefore, a need exists for developer tools that make it easier to compare, analyze, and evaluate code translations. Equipped with our tool, companies will be able to use more modern and secure languages at lower cost and with greater confidence.

3.3 Key Drivers

Every company that has a codebase will eventually need to be updated to support modern languages and frameworks which can make their systems more reliable and efficient. A current example is with old languages like FORTRAN and Visual Basic which are not as popular as they were when they were released. Today, not many developers are familiar with these older languages which means that a senior developer who is familiar with them is needed in order to maintain them but this becomes increasingly difficult as people get older.

The older a company's codebase is, the more likely they are using languages that either have been or are in the process of being phased out. This situation also comes with security concerns because languages like C and C++ can be unsafe and contain memory safety issues. Even within popular languages, some projects stay on older versions of a modern language which can be a security concern if a vulnerability is found. These are often fixed in newer versions but that does

nothing if the codebase does not support that version. It's only practical that companies would want to transition to a safer language that does not have these issues.

There have been efforts to address the security concerns of outdated codebase with one being the TRACTOR project by the Department of Defense. The project aims to translate all C and C++ code to Rust. These large efforts would greatly benefit from a tool that could evaluate how accurate the code translation was which is where our solution becomes very relevant in ensuring that the translated code functions exactly the same as the original code

3.4 Key Constraints

Throughout the development of this project, there are limitations to be encountered such as the time the algorithm would need to produce an output. The user benefits from the speed of applications and their processing time. With the scope of this project, the team would need to prioritize the user and all their needs to ensure a successful product.

Another constraint is the complexity of the code being analyzed and translated. Past codebases could have intricate dependencies, outdated libraries, or non-standard implementations that may pose challenges to automatic analysis and conversion. The tool needs to handle these edge cases effectively, which may require additional resources than expected.

Additionally, maintaining security and data privacy is very important when dealing with company databases. Many companies handle sensitive or proprietary code, so the we create system must be built with robust security measures to ensure that their information is protected throughout the translation and analysis process.

Finally, the availability of resources to develop, maintain, and improve the system could be one of our main constraints. This project involves a deep understanding of both older and modern programming languages, which may impose a limitation regarding experience. The team will need adequate support and resources to be able to meet the goals of this project.

x

Chapter 4

Development Plan

– Ben Knobloch, Lauren Kibalo, Marcos Morales, Noah Spina

4.1 Introduction

This product will be a free, open-source, and collaborative LaTeX editor. Our goal is for institutions to have a collaborative typesetting editor that is locally compiled, free for all its members, and not reliant on any external platform or servers. Our initial client will be the students and faculty of Stevens Institute of Technology.

Enterprise members will interact with the service via a free and open-source desktop application. It will be used by the students and faculty of Stevens Institute of Technology for university-wide collaboration and content creation. Eventually this will be expanded to include all enterprise groups on an open-source basis.

4.1.1 Requirements

- Allow multiple users to connect to an document editor
- Allow for the environment to locally compile latex code
- Allow for the project to be a desktop application as to not rely on servers or 3rd party hardware

Architecture

The architecture, still being planned, will heavily depend on the users desktop to store files and compile latex. There will be a server that helps users to connect to the same document via TCP connection.

Success Criteria

Our project will be considered successful if multiple users can connect to a latex environment and compile documents in real time.

4.2 Roles and Responsibilities

These are the roles we have formalized for our software engineering team.

1. Development Lead (Ben)
2. Scrum Master (Marcos)
3. Architect (Noah)
4. Developers (Ben, Lauren, Marcos, Noah)
5. Testers (Ben, Lauren, Marcos, Noah)
6. Documentation (Ben, Lauren, Marcos, Noah)
7. Designer (Lauren, Noah)
8. User advocate (Ben)
9. Product Owner (Lauren)
10. Customer responsible for acceptance testing (Professor Darian)

4.3 Method

Agile methodologies will be adapted for this project team. This unique approach to project delivery based can fulfill specific needs and characteristics for this software project and organization. The team needs to be able to reconstruct the plan as we learn new processes throughout the execution of our overleaf adaptation.

4.3.1 Software

1. Languages and Platforms

   Parallels virtual environment software

   C++20 (ISO/IEC 14882:2020)

   JavaScript ECMAScript 2024 (NodeJS WebSocket)

   PostgreSQL 17.0

2. Operating systems

   Windows

   MacOS

   Linux

3. Software packages/libraries used with release/version number

4. Code conventions

   Official C++20 coding standards

   REST

4.3.2 Hardware

1. Development and Test Hardware:

   MacBook Pro 14-inch (November 2023)

   M1 MacBook Air 13-inch (2021)

   Windows Surface Book (2021)

   MacBook Pro 16-inch (2019)

2. Target/Deployment Hardware

   All personal machines running modern versions of Windows, MacOS, or Linux

4.3.3 Review Process

1. Architecture, usability, design, security, privacy and code reviews
2. Informal methods of individually reviewing a new version of the code
3. Every team member will be responsible for reviewing versions and ensuring that the quality planned for is met after each sprint
4. Code reviews will be performed by the team member who is reviewing the pull request

4.3.4 Build Plan

1. GitHub version control and project management
2. GitHub Actions continuous integration
3. Weekly builds
4. Weekly build deadlines
5. Multiplicity of builds
6. Regression test process – see test plan

4.4 Virtual and Real Workspace

We will be utilizing the software engineering lab and university library for our primary physical workspaces on campus. Virtually, we will use the team's Overleaf document to keep track of all progress.

4.5 Communication Plan

4.5.1 Heartbeat Meetings

Heartbeat meetings will take place during our scheduled senior design class on Tuesdays and Thursdays. We would mainly be discussing items that need to be worked on and small updates on items that are in progress.

4.5.2 Status Meetings

Status meetings will take place biweekly in place of the heartbeat meeting on Tuesday or Thursday. Here we will go into more detail on the progress that has been made since the previous status meeting. Professor Darian would most likely be attending these meetings as well. For any issues these will be discussed in a separate meeting.

4.5.3 Issues Meetings

If anyone on the team encounters an issue with development, they would message the team group chat we have where it can briefly be addressed. Depending on the complexity of the issue the rest of the team may need some time to research and assist mitigating the issue. If the issue cannot be solved with these methods then someone with more experience in this field (Professor Darian) will be contacted for further assistance.

4.6 Timeline and Milestones

4.6.1 Milestone 1 - 10/3/24

Have a Development Plan laid out for each milestone

4.6.2 Milestone 2 - 10/24/24

Plan development with UML diagrams, making sure to focus on class, object, and package diagrams.

4.6.3 Milestone 3 - 11/7/24

Set up environment and DevOps workflows. Have a clear understanding of team roles for each milestone. Establish Database for documents to be stored. Finish UML planning.

4.6.4 Milestone 4 - 11/21/24

Figure out and implement a system where a user can add LaTeX code and compile it locally.

4.6.5 Milestone 5 - 12/12/24

Establish API calls for one user to push data to the document. Make sure it works for one user locally before implementing support for multiple users.

4.6.6 Milestone 6 - 1/31/24

Establish TCP connection for multiple users to edit a document.

4.6.7 Milestone 7 - 2/21/24

Configure multiple API calls to happen when different users write latex.

4.6.8 Milestone 8 - 3/14/24

Test and fix any errors.

4.6.9 Milestone 9 - 4/7/24

Polish product.

4.7 Testing Policy/Plan

We will adopt a continuous integration scheme and write regular tests to verify the code base upon each commit. We plan to update unit tests on a regular biweekly basis, system tests on a monthly basis, and regression tests on an as-needed basis when major changes or additions are made to the system.

Changes to the code base will not be approved unless they pass all unit, system, and regression tests, or if any failed tests have been shown to be flawed or insufficient.

4.8 Risks

1. Our product will require network engineering that goes beyond the experiences of any of our project members.

   Mitigation: Our team will conduct extensive individual research and learning.

2. Our product may require paid services.

   Mitigation: We will leverage Stevens equipment for testing and initial deployment if needed.

4.9 Assumptions

1. We assume that Stevens Institute of Technology will dedicate the resources to maintain our solution for a lifetime of at least five years so that the service will experience wide acceptance and adoption among Stevens students and faculty.

2. We assume that the most important attributes of a LaTeX editor for academic faculty and students is the ability to edit collaboratively and access personal content without connecting to proprietary services.

4.10 Distribution List

Professor Muresan would receive this document weekly which will always include a new weekly report.

4.11 IRB Protocol

Our project does not require an IRB application.

4.12 Required Resource and Budget

Since we will be using Overleaf, we will not need a budget at this time.

4.13 Documentation Plan

At a minimum this document will be updated weekly through the weekly report. However once development kicks off it will end up being updated more frequently Many years ago we had much too much documentation, now we have precious little – this must change.

Chapter 5

Requirements

– Ben Knobloch, Lauren Kibalo, Marcos Morales, Noah Spina

5.1 Stakeholders

1. Software Developers
2. Researchers
3. Professors
4. Students

5.1.1 Customers

1. Software Developers
2. Researchers
3. Professors
4. Students

5.1.2 Sponsors

1. Stevens Institute of Technology
2. Darian Muresan

5.1.3 Engineering and Technical Persons

Noah Spina, Lauren Kibalo, Ben Knobloch, Marcos Morales

5.1.4 Regulators

Stevens Institute of Technology

5.1.5 Third Parties

1. Overleaf

5.1.6 Competitors

Systems which provide similar functions.

1. TeXnicCenter
2. Overleaf
3. Google Docs
4. Microsoft Word

5.2 Key Concepts

List the key concepts and their definitions that relate to your project. These concepts and terms should be used consistently throughout this document and in your project. This can point out to the Glossary chapter.

1. LaTeX
2. TCP Connection
3. Simultaneous Document Editing

5.3 Development Requirements

Table 5.1: Development Requirements Table

Requirement	**Priority**	**Use Case(s)**
Development Requirement 1 (reqfFirstDevRequirement) *The developers shall understand the existing source code of Overleaf to continue the development process*	Must Have	UC_5

Table 5.1: Development Requirements Table

Requirement	**Priority**	**Use Case(s)**
Development Requirement 2 (reqfSecondDevRequirement) *Users seeking to develop Underbranch further shall understand how to setup their development environment.*	Must Have	UC_5

5.4 User Requirements

The following user requirements are numbered and linked to use-cases and user-stories.

Table 5.2: Functional Requirements Table

Requirement	**Priority**	**Use Case(s)**
Functional Requirement 1 (reqFirstFunctionalRequirement) *The system shall link LaTeX code to the germane location of the PDF document and vice versa.*	Should Have	UC_1
Functional Requirement 2 (reqSecondFunctionalRequirement) *The system shall allow multiple users should be able to edit a single document simultaneously.*	Must Have	UC_2
Functional Requirement 3 (reqThirdFunctionalRequirement) *The system shall locally compile and display a LaTeX document.*	Must Have	UC_1
Functional Requirement 4 (reqFourthFunctionalRequirement) *The system shall only be updating the LaTeX code in real time without recompiling the code. Users can recompile by pressing a button to avoid slowdowns while editing.*	Must Have	UC_1

Table 5.2: Functional Requirements Table

Requirement	**Priority**	**Use Case(s)**
Functional Requirement 5 (reqFifthFunctionalRequirement) *The system shall install a local version of Overleaf with minimal user interaction unless needed.*	Must Have	UC_1

5.5 System (Constraints) Requirements

More detailed descriptions of the services and constraints from the perspective of the system to

Table 5.3: Constraints Requirements Table

Constraint	**Priority**	**Use Case(s)**
Constraint Requirement 1 (reqFirstSystemConstraint) *Users shall be able to edit a document simultaneously.*	Must Have	UC_2, UC_3
Constraint Requirement 2 (reqSecondSystemConstraint) *System shall be fully functional on the user's hardware.*	Must Have	UC_1, UC_3, UC_4
Constraint Requirement 3 (reqThirdSystemConstraint) *The system shall operate on the host computer's subnet*	Must Have	UC_1, UC_3, UC_4
Constraint Requirement 4 (reqFourthSystemConstraint) *The system shall operate on the Windows and MacOS operating systems*	Must Have	UC_1, UC_3, UC_4

5.6 Non-functional (Quality) Requirements

Any important non-functional requirements for the project. These include any performance and quality requirements (i.e. numbers, such as bandwidth, time, speed, rates, etc.).

Table 5.4: Quality Requirements Table

Requirement	**Priority**	**Use Case(s)**
Quality Requirement 1 (reqfFirstQualityRequirement) *The system shall handle multiple users editing changes in real time without noticeable lag (<5 seconds of un-updated edits on any user's computer).*	Must Have	UC_2
Quality Requirement 2 (reqfSecondQualityRequirement) *The system shall not be blocked by firewalls when users attempt to connect to the same document.*	Should Have	UC_2

5.7 Domain (Business) Requirements

Table 5.5: Business Requirements Table

Requirement	**Priority**	**Use Case(s)**
Business Requirement 1 (reqfFirstBusinessRequirement) *The application shall be free and offer the same or higher quality as other leading document editors.*	Must Have	UC_1
Business Requirement 2 (reqfSecondBusinessRequirement) *The application shall allow for anyone with a windows or mac computer to use it.*	Must Have	UC_1, UC_4

Chapter 6

Use Cases

– Noah Spina, Lauren Kibalo, Ben Knobloch, and Marcos Morales

This chapter presents the use case diagrams that satisfy the requirements. Detailed description of each use case should be given in separate chapters for each use case. In this sample script, they are all left in this chapter. Move them as needed.

6.1 Table of Use Cases

To prevent gold-plating all use cases must be mapped to at least one requirement. If there are any use cases that don't have a requirement, then the use case should be removed. A table to keep track of this mapping is shown next.

Table 6.1: Use Cases Table

Use Case	Requirements	Name and Description
UC_1	$reqkFunctional_1$, $reqkFunctional_3$, $reqkFunctional_4$, $reqkConstraint_2$, $reqkConstraint_3$, $reqkConstraint_4$, $reqkBusiness_1$	ucLaTeXToPDF describes how a user should be able to seamlessly develop, view, and change between LaTeX and the pdf file it compiles.
UC_2	$reqkFunctional_2$, $reqkConstraint_1$, $reqkQuality_1$, $reqkQuality_2$	ucMultipleUsers describes how multiple users should be able to edit the same document in real time.
UC_3	$reqkConstraint_1$, $reqkConstraint_2$, $reqkConstraint_3$, $reqkConstraint_4$, $reqkBusiness_2$	ucDiverseOperatingSystems describes how a group of five users with mixed operating systems running on laptops should be able to work collaboratively on a project.
UC_4	$reqkFunctional_5$, $reqkBusiness_2$, $reqkConstraint_2$, $reqkConstraint_3$, $reqkConstraint_4$	ucInstallingUnderbranch A user clicks a button to install Underbranch.
UC_5	$reqkDev_1$, $reqkDev_2$	ucDevelopUnderbranch A user seeks to develop and add to Underbranch

6.2 Use Case Diagrams

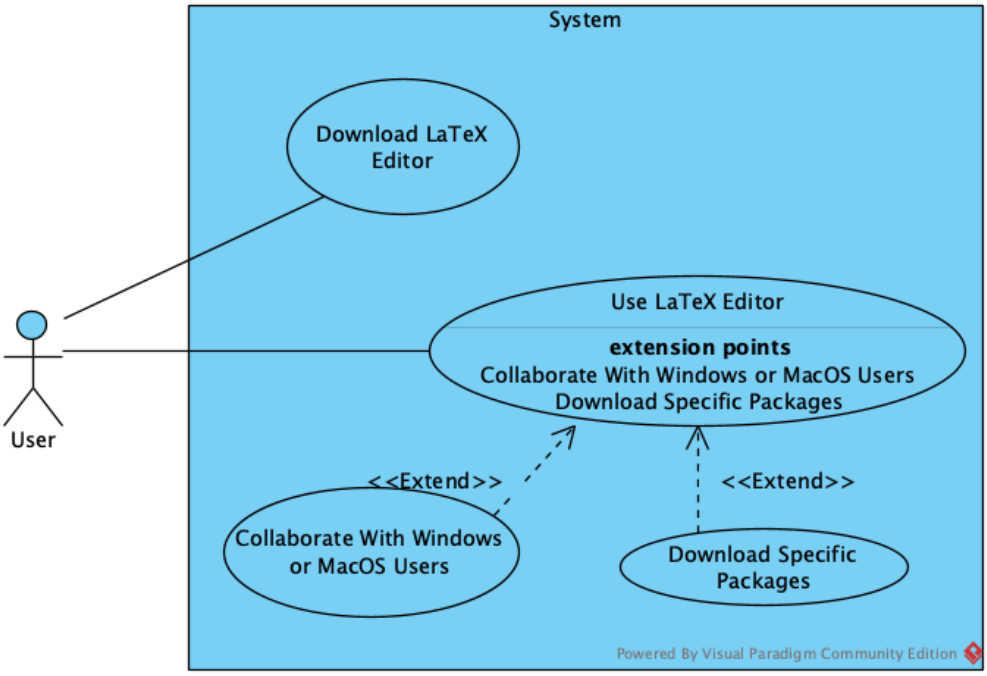

Figure 6.1: Use case diagram

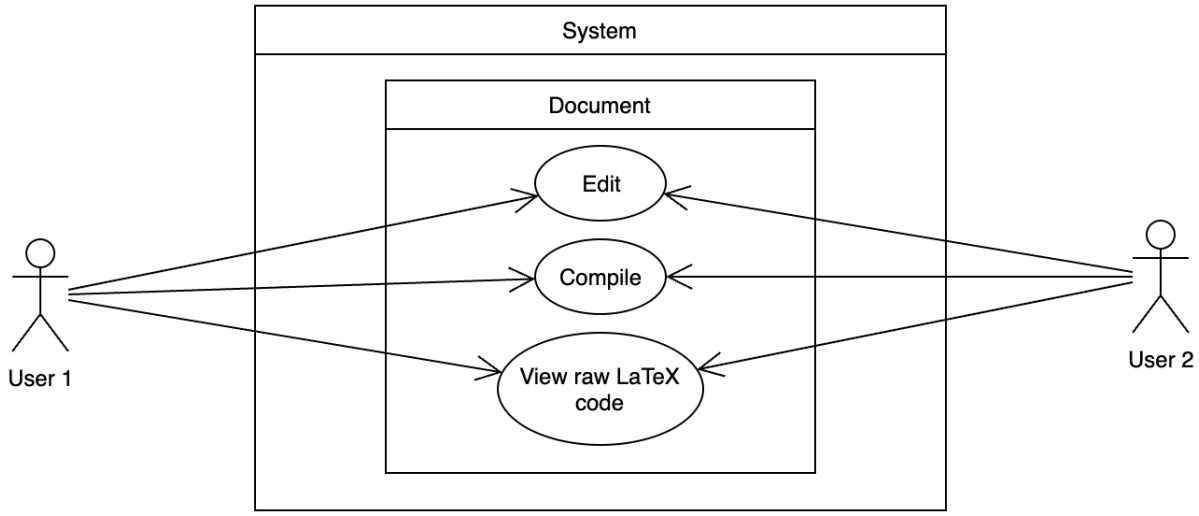

Figure 6.2: Use case diagram for multiple users interacting with a document at the same time (See UC_2.)

Table 6.2: Use Case First

Use Case 1 (ucLaTeXToPDF) *A user types LaTeX code and renders the typed text in a PDF viewer.*
Diagrams: Figure dsnUseCase1 (*Figure* 8.4), dsnActivity1 (*Figure* 8.6)
Brief description: The user has the ability to type LaTeX code and compile it into a PDF document.
Primary actors: User
Secondary actors: None.
Preconditions: 1. Document contains LaTeX code that can be compiled
Main flow: 1. User types LaTeX code 2. User presses the compile button 2.1. If the code has no syntax errors then a PDF of the resulting document is shown 2.2. If not then an error is displayed describing what went wrong
Postconditions: None.
Alternative flows: None.

Table 6.3: Simultaneous editing

Use Case 2 (ucMultipleUsers) *Multiple users should be able to edit the same document in real time.*
Diagrams: Figure dsnUseCase2 (*Figure* 6.2), dsnActivity1 (*Figure* 8.6)
Brief description: One user makes an edit to a collaborative document, with the edit consisting of adding text, deleting text, pasting text, or other text editor actions. The system should reproduce the changes on the upstream version of the document and then on any instances of the document on other users' machines. All editing conflicts should be reconciled according to the time of the edit.
Primary actors: User
Secondary actors: Synchronous users
Preconditions: 1. A user has created and shared a collaborative document with at least one other user.
Main flow: 1. A user has launched a collaborative document. 2. A second user has launched a collaborative document. 3. If the first user makes an edit to the document then 3.1. The master version of the document updates. 3.2. The second user's version of the document updates and reconciles the changes with any simultaneous and contradictory changes.
Postconditions: None.
Alternative flows: None.

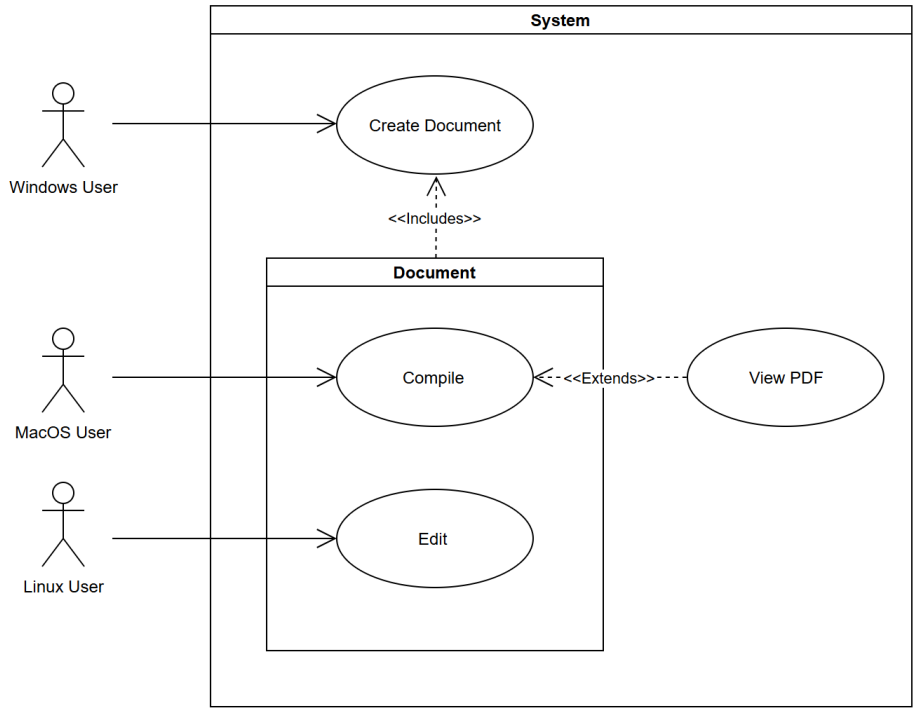

Figure 6.3: Use case diagram for multiple users interacting with the system at the same time from different operating systems (See UC_3.)

Table 6.4: Simultaneous editing

Use Case 3 (ucDiverseOperatingSystems) *A group of more than two users with mixed operating systems running on laptops should be able to create and work collaboratively on a project.*
Diagrams: Figure dsnUseCase3 (*Figure* 6.3), dsnActivity3 (*Figure* 8.7)
Brief description: Multiple users with machines running different operating systems should be able to create, edit, and view documents collaboratively without interruption.
Primary actors: Windows User MacOS User Linux User
Secondary actors: None.
Preconditions: 1. Application is installed on all operating systems.
Main flow: 1. A Windows user creates a new document. 2. A Linux user edits the document. 3. A MacOS user compiles and then views the document.
Postconditions: None.
Alternative flows: None.

Table 6.5: Installing Underbranch

Use Case 4 (ucInstallingUnderbranch) *A user clicks a button to install Underbranch on their local computer/desktop.*
Diagrams: Figure dsnActivity4 (*Figure* 8.8), dsnActivity5 (*Figure* 8.9)
Brief description: A user clicks a button to install Underbranch and, after the install has finished, can use it without any additional steps.
Primary actors: User
Secondary actors: None.
Preconditions: 1. None
Main flow: 1. A user clicks the install button online. 2. The Underbranch installer starts and installs Underbranch 3. User can open Underbranch and use it
Postconditions: None.
Alternative flows: If the user already has Underbranch installed, Underbranch will open instead of installing twice.

Table 6.6: Developing Underbranch

Use Case 5 (ucDevelopUnderbranch) *The user sets up their development environment to further work on Underbranch.*
Diagrams: dsnActivity5 (*Figure* 8.9)
Brief description: The user sets up their development environment to further work on Underbranch.
Primary actors: User
Secondary actors: None.
Preconditions: None
Main flow: 1. Clone the Underbranch Github Repository (https://github.com/beknobloch/overleaf-toolkit) 2. Open repository in IDE of your choice 3. Run *npm install* 4. Run *npm run build*
Postconditions: None.
Alternative flows: If the user has built the application before (i.e. If youve ran *npm run build*), then the user must run *rm -rf node_modules dist* before every build after the first one.

6.3 Architecture Diagrams

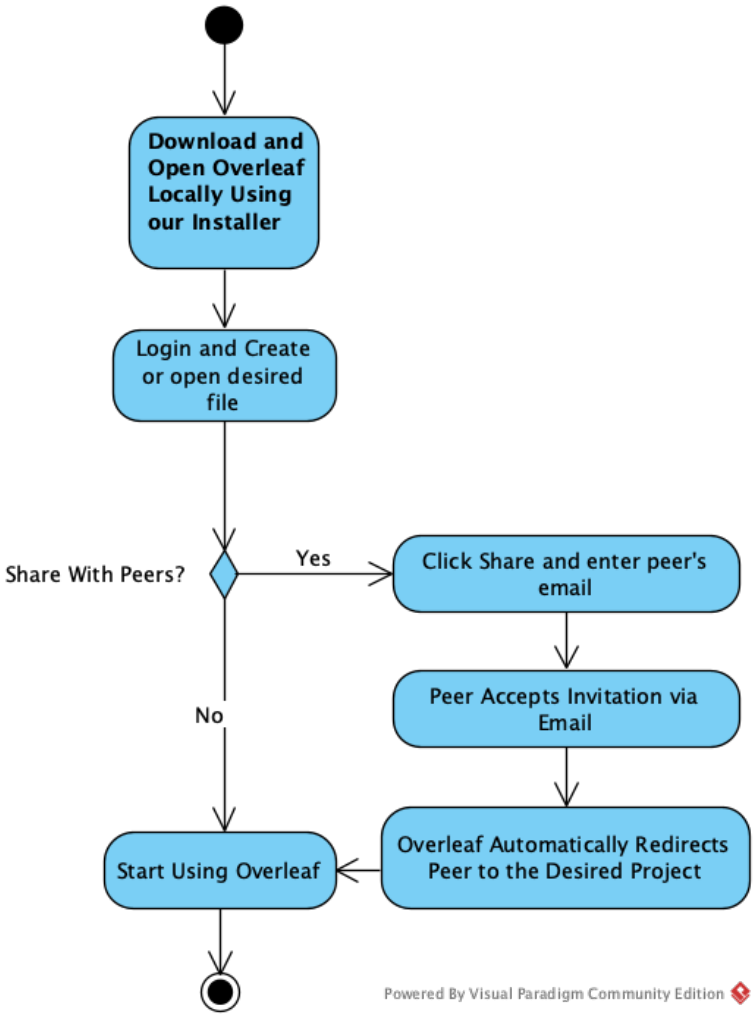

Figure 6.4: Activity diagram for compilating and collaborating on an Underbranch document (See UC_1 and UC_2.)

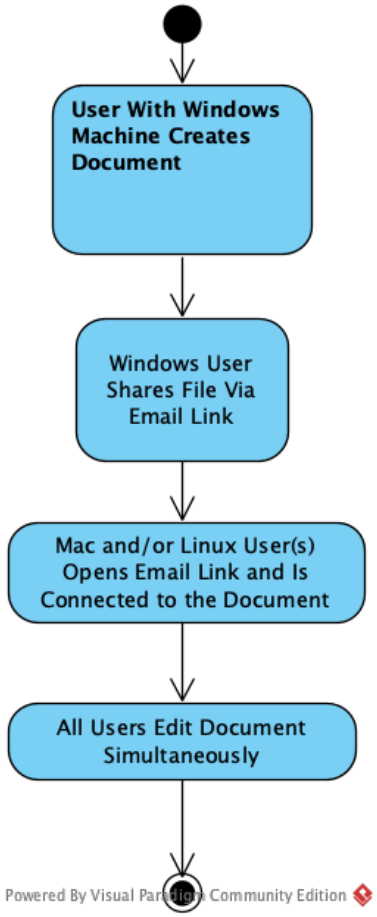

Figure 6.5: Activity diagram for multiple operating systems using Underbranch (See UC_3.)

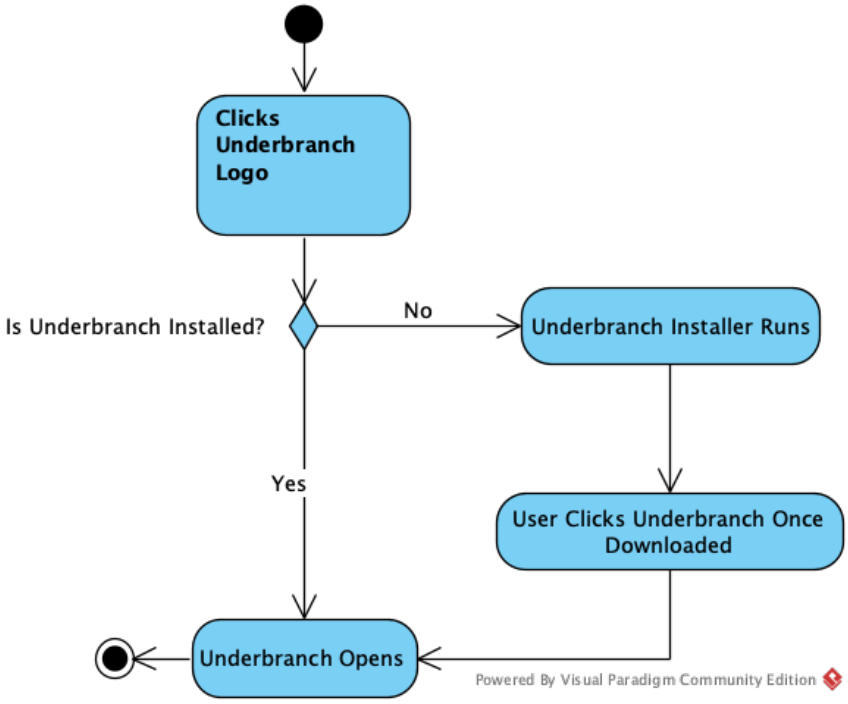

Figure 6.6: Activity diagram for Underbranch installation (See UC_4.)

Chapter 7

Stakeholders

– Ben Knobloch, Lauren Kibalo, Marcos Morales, Noah Spina

7.1 Overview

This section identifies key stakeholders affected by this project. Each stakeholder group plays a vital role in shaping requirements, providing insights, and ensuring the system meets the needs of its users.

7.2 Software Developers

1. Role
    - Responsible for building and maintaining the application
2. Needs
    - Clear technical requirements
    - Access to user feedback
3. Contributions
    - Provide insights on feasibility and possible technical challenges
    - Suggest improvements for development practices

7.3 Researchers

1. Role
    - Users who rely on the application for documenting their research findings
2. Needs
    - Features for documentation
    - Collaboration tools to work with others effectively

3. Contributions
    - Offer requirements based on use cases
    - Highlight functionality that improves research documentation

7.4 Students

1. Role
    - Primary users who will use the application for assignments and projects
2. Needs
    - Usable interface that helps learning LaTeX without hard learning curves
    - Features that support collaboration with peers on group projects
3. Contributions
    - Share feedback on usability and feature effectiveness
    - Identify gaps in functionality based on academic requirements

7.5 Customers

1. Role
    - Those that may purchase or adopt the application for institutional uses
2. Needs
    - Assurance of scalability and support for multiple users
    - Clear productivity and efficiency gains
3. Contributions
    - Provide insights into market requirements and competitive features
    - Suggest pricing and licensing options

7.6 Summary

By understanding the diverse needs and contributions of each stakeholder group, this project can align its development efforts to ensure the LaTeX application effectively addresses current challenges and enhances user experiences. This stakeholder analysis will help guide the requirements process and inform design decisions throughout the project.

Chapter 8

Preliminary Design

– Ben Knobloch, Lauren Kibalo, Marcos Morales, Noah Spina

8.1 Introduction

The project aims to develop a free, open-source, and collaborative LaTeX editor that allows users to collaboratively create and edit LaTeX documents without reliance on external platforms or servers. The system will enable users to compile LaTeX code locally on their machines, ensuring privacy, security, and performance. Our initial target audience will be the students and faculty of Stevens Institute of Technology, with the potential for future expansion to other academic institutions and enterprise groups.

The preliminary design for this project includes the architecture for a collaborative document editing environment, where users can locally compile LaTeX code. It will enable multiple users to edit documents simultaneously, with local compilation of LaTeX code and file storage on each user's desktop. The design will include basic user interface components, a system architecture overview, and the necessary infrastructure to support collaborative editing.

8.2 UI Design

8.2.1 User Persona

Define the persona of your users. You may have more than one user person for capturing different types of user roles for your system.

1. Student
    - Overview: Undergraduate student in engineering
    - Goal: Needs an easily usable LaTeX editor for documenting collaborative group work, preferably on a local platform
    - Pain: Frustrated with only having 2 free users editing at a time
2. Faculty / Researcher
    - Overview: Professor in engineering
    - Goal: Needs a collaborative environment for creating lecture notes, research papers, and student group assignment submissions

- Pain: Prefers a system where students can collaborate in groups without external dependencies

8.2.2 User Interface Design

Our user interface design will mimic Overleaf in utility but add a streamlined installation and launching experience that will improve speed and usability compared to the Overleaf Toolkit.

Figure 8.1: Mockup of installing Underbranch from the OS's application system

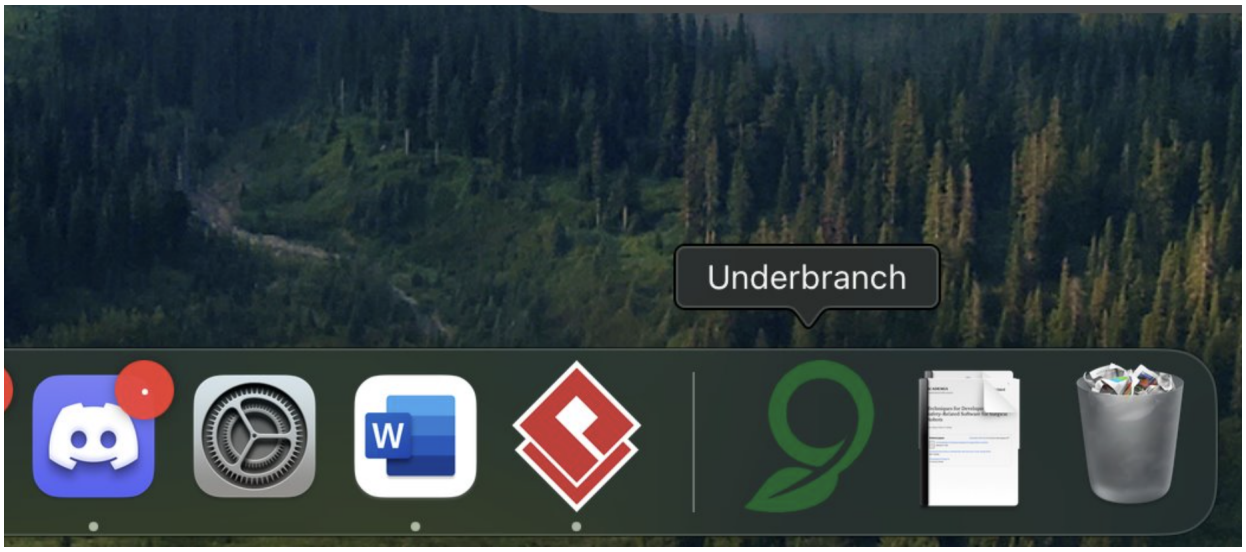

Figure 8.2: Mockup of launching Underbranch from the OS's application system (pictured here as MacOS with a stand-in logo)

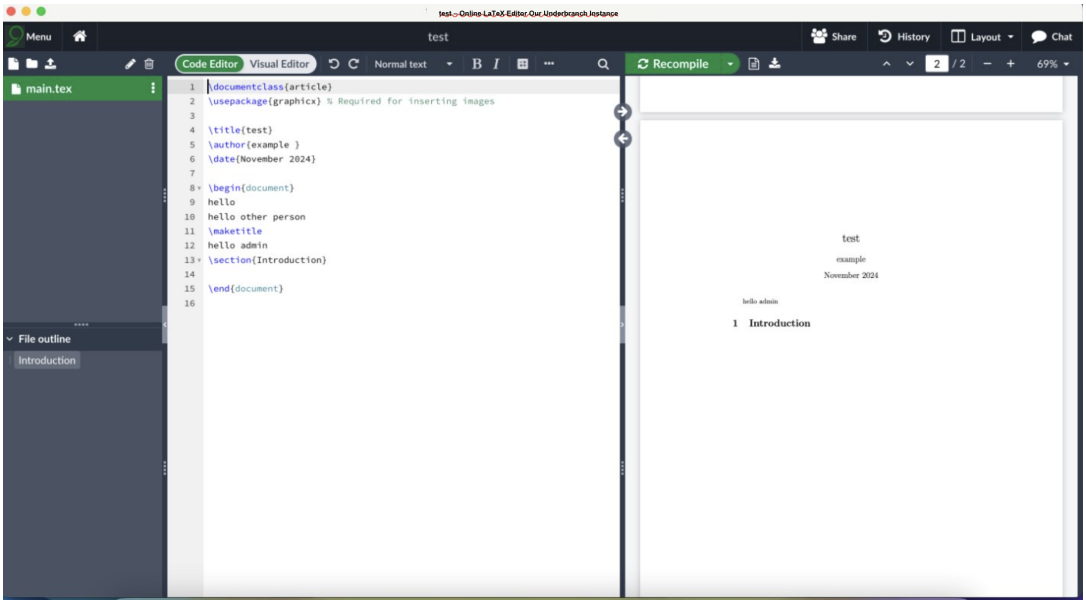

Figure 8.3: Mockup of the Underbranch UI in-use

8.3 System Architecture

This section describes the design of the Underbranch system using the 4+1 view.

8.3.1 Scenarios/Use Cases

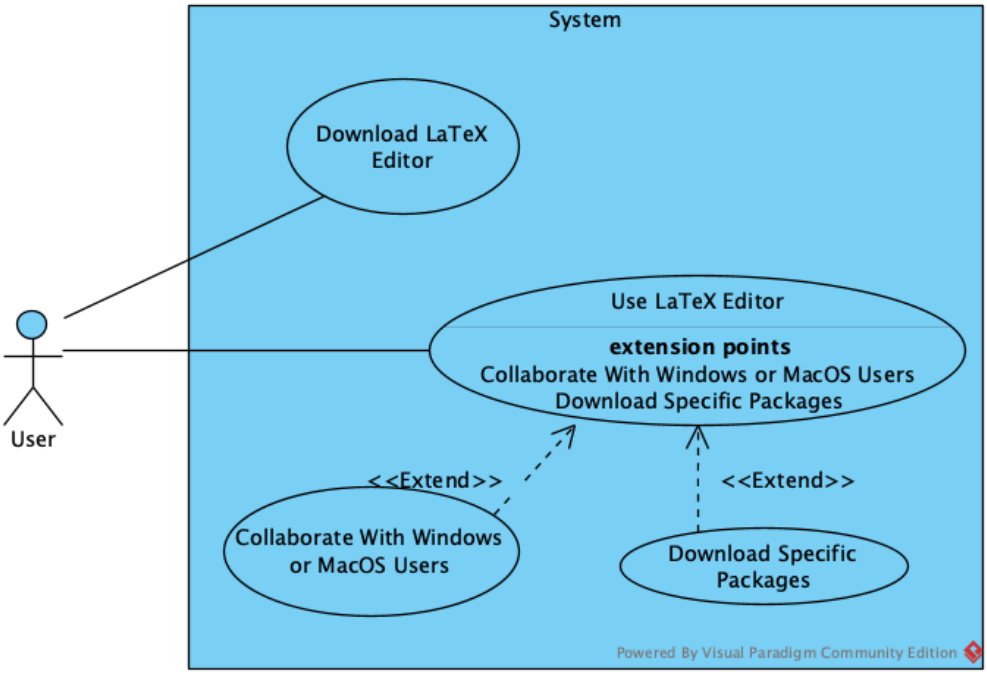

Figure 8.4: Use case diagram.

8.3.2 Logical View

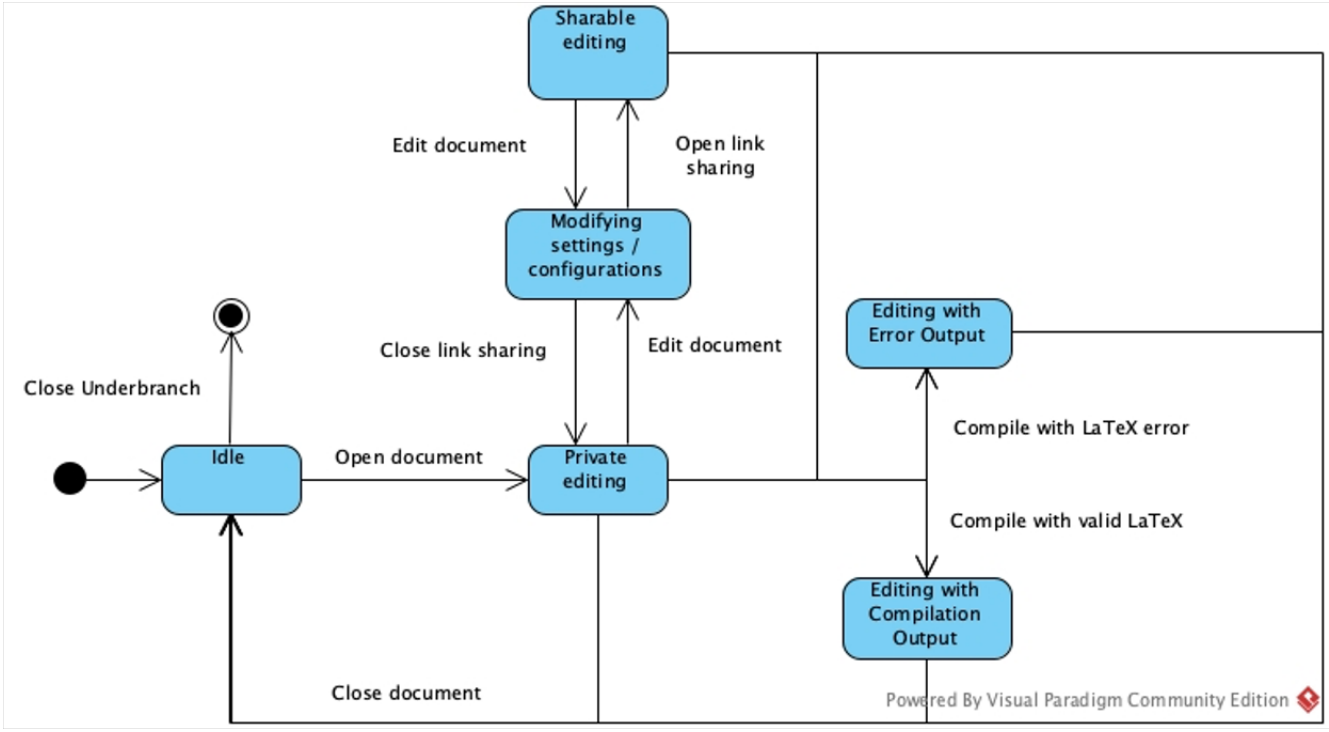

Figure 8.5: State Diagram of Underbranch Application

8.3.3 Process View

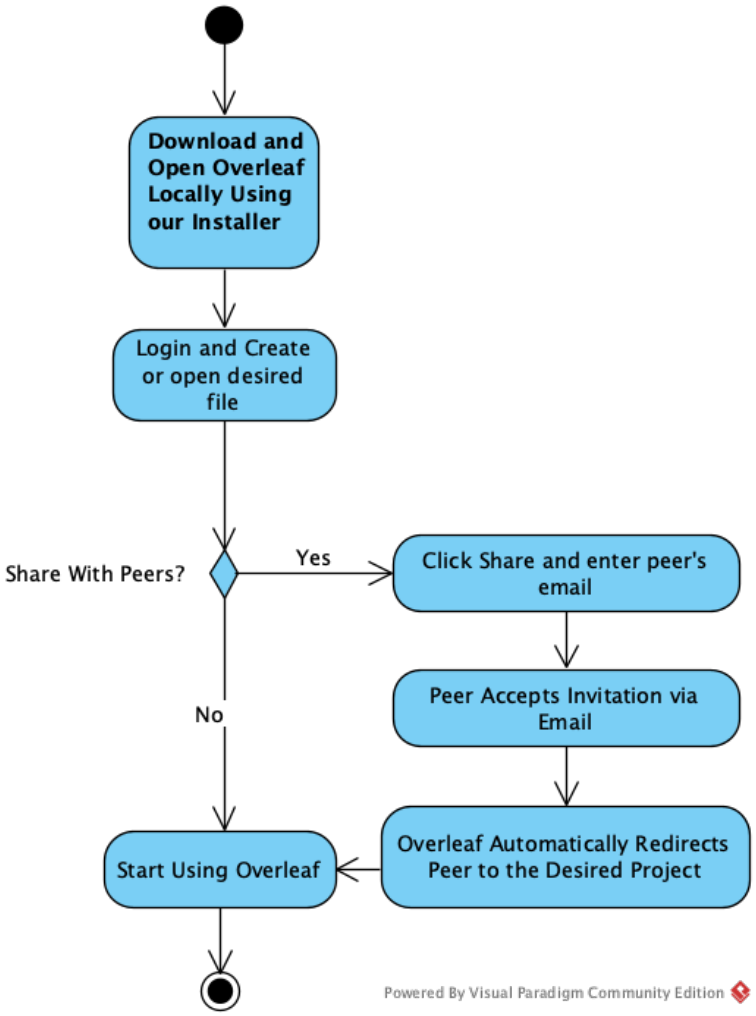

Figure 8.6: Activity diagram for compiling and collaborating on an Underbranch document (See UC_1 and UC_2.)

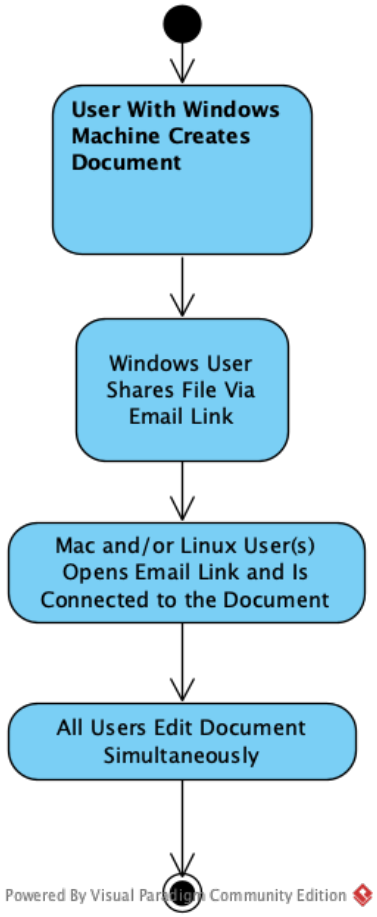

Figure 8.7: Activity diagram for multiple operating systems using Underbranch (See UC_3.)

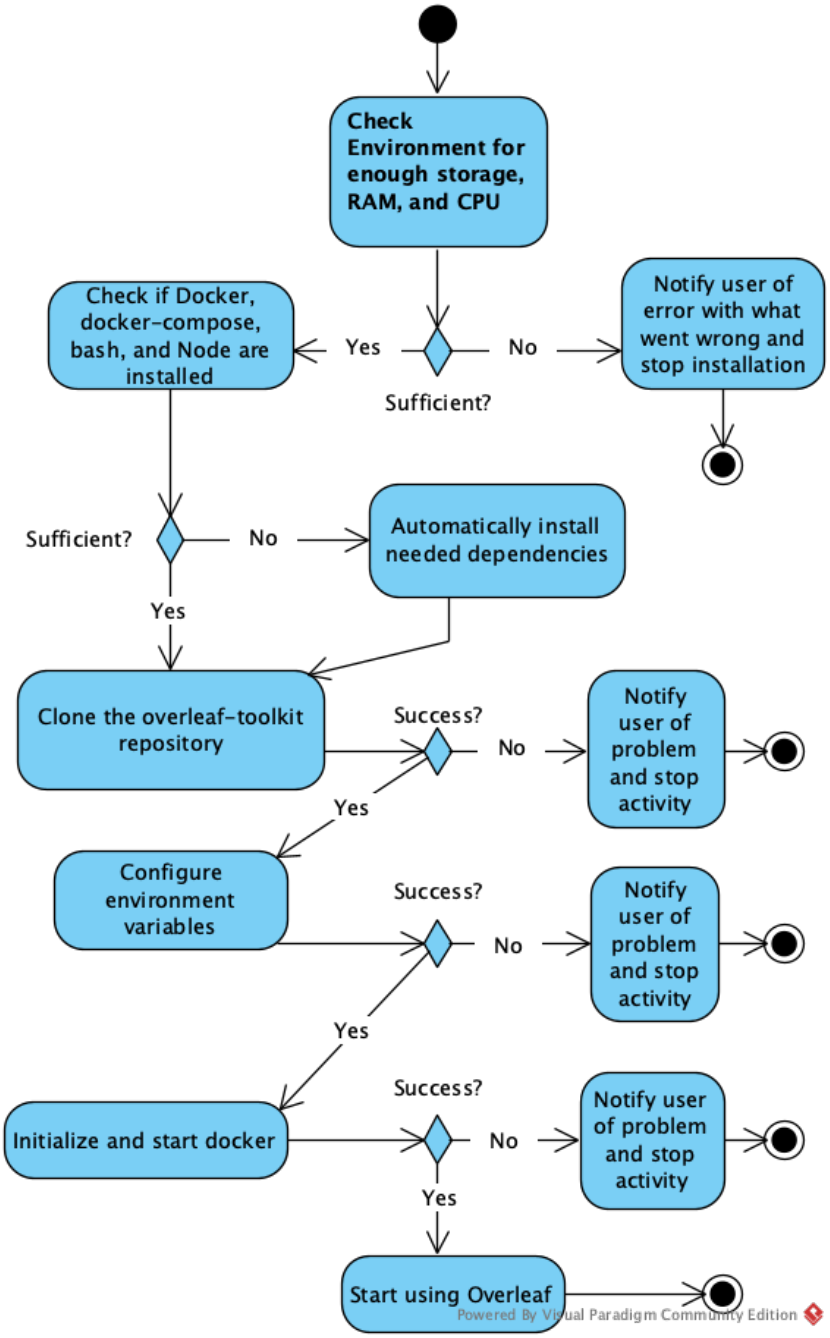

Figure 8.8: Activity diagram for Underbranch installation (See UC_4.)

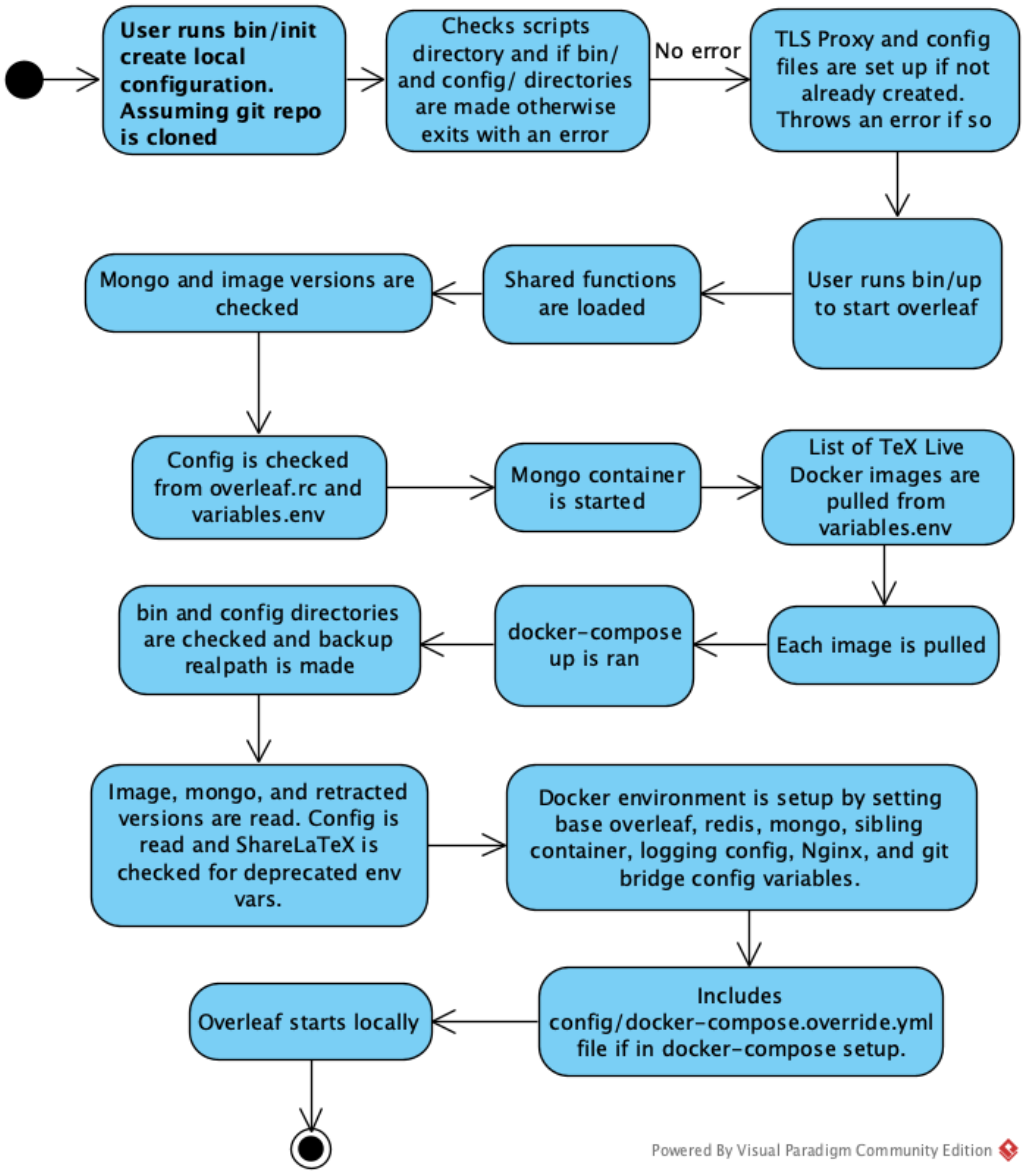

Figure 8.9: Activity diagram for Overleaf's manual Installation and Shell Scripts (See UC_4.)

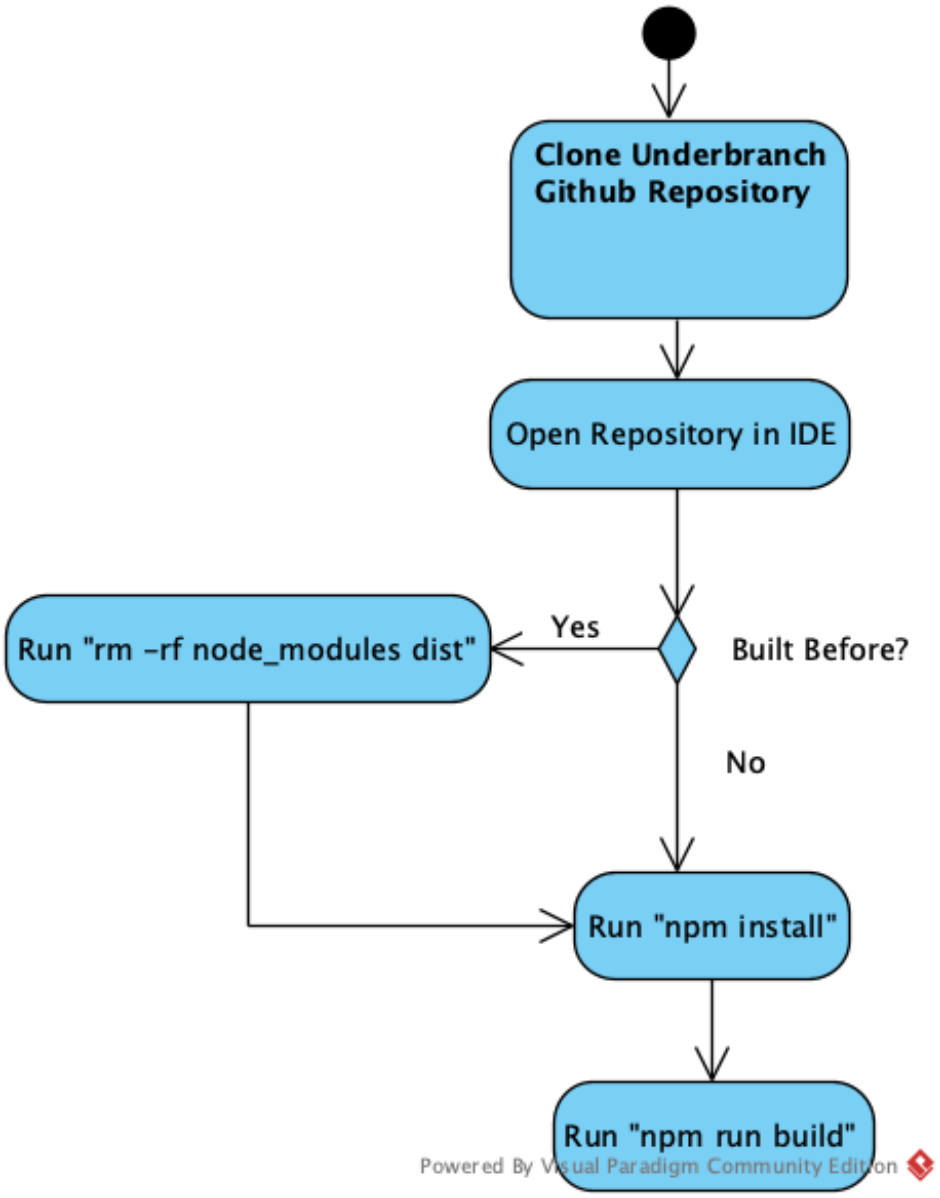

Figure 8.10: Activity diagram for setting up the development environment for Underbranch (See UC_5.)

8.3.4 Development View

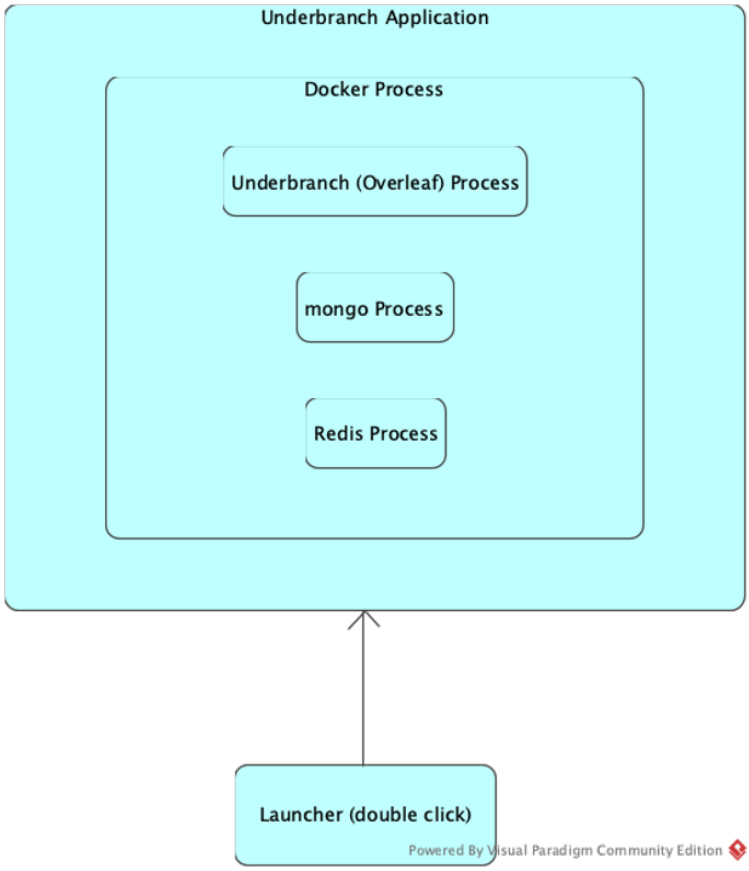

Figure 8.11: Development View of Underbranch Application

8.3.5 Physical View

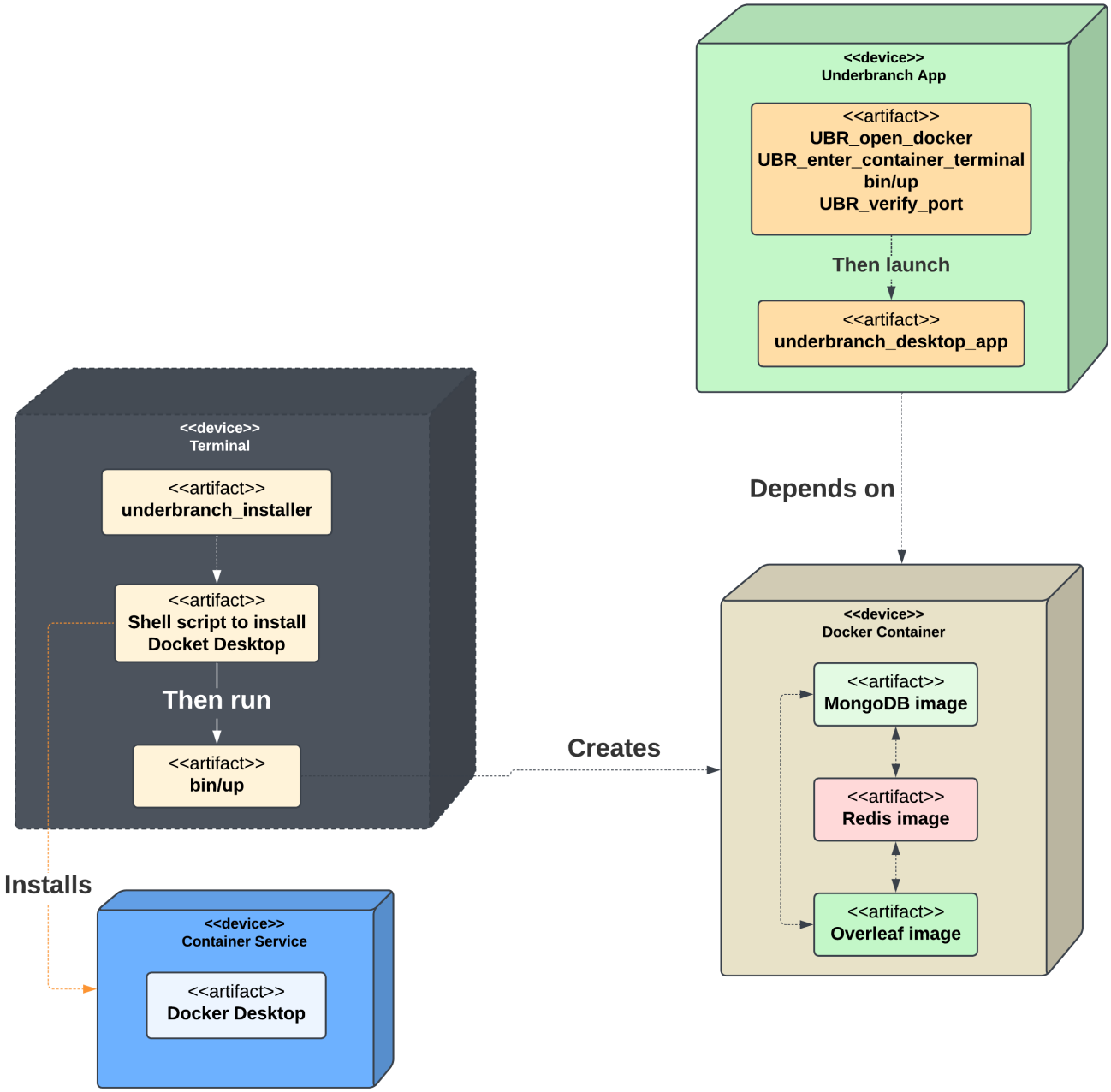

Figure 8.12: Deployment Diagram for Installation Process

Chapter 9

Prototype Demo

– Ben Knobloch, Lauren Kibalo, Marcos Morales, Noah Spina

9.1 Current Prototype

Currently we have an ElectionJS desktop app that loads the application running on http://localhost/launchpad. For this to work, the Overleaf Docker container needs to have been created and running on the user's machine. If these conditions are met, the desktop app will then open the Overleaf sign in page and the user can proceed to create and run an Overleaf project just as they would on Overleaf.com. You can find a recording of our demo here.

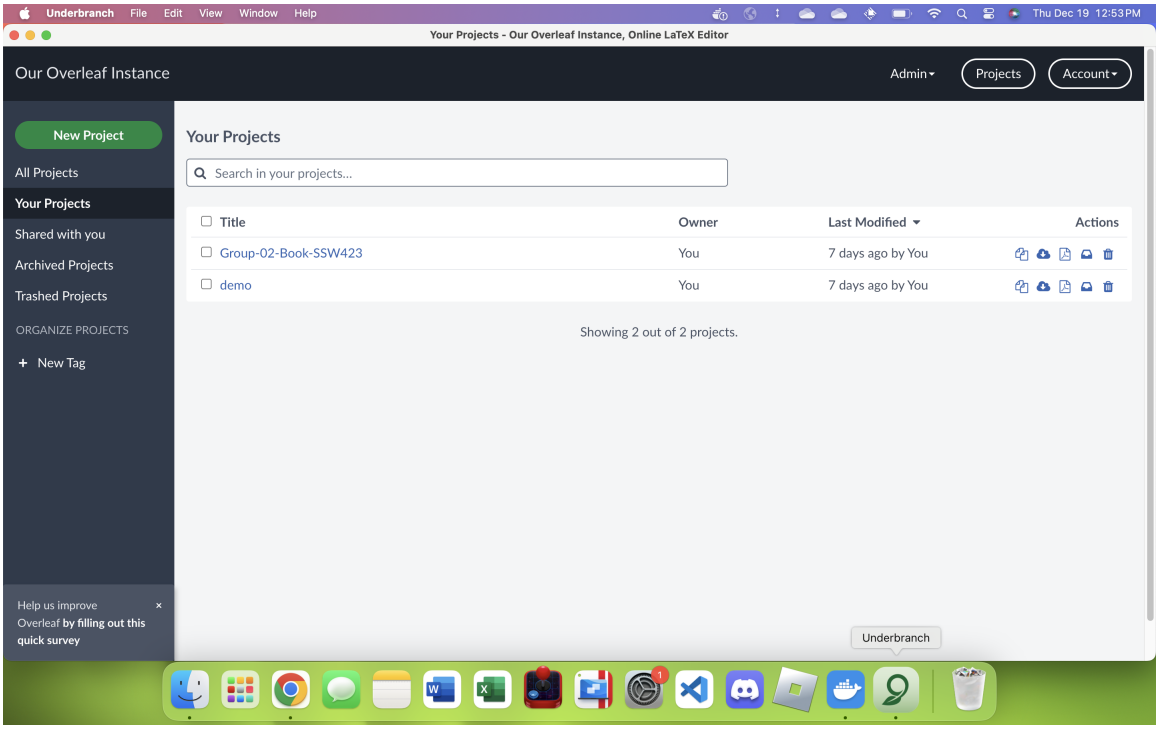

Figure 9.1: Prototype Demo of Underbranch Desktop Application

9.2 LaTeX Package Installation

In the days leading up to the demo presentation, the team discovered how to install LaTeX packages in a local instance of Overleaf and how these packages are retrieved by the system. LaTeX packages are managed by a service called TeX Live. On Overleaf.com, LaTeX packages are automatically retrieved from an Overleaf server that contains some version of TeX Live that automatically updates existing packages and retrieves new packages. When using a local instance of Overleaf through the Overleaf Toolkit, a lighter TeX Live with very few packages is used in order to make the installation faster. The Overleaf toolkit does not connect to a TeX Live server by default which presents the problem of how can LaTeX packages be installed. Within the Docker container there are 3 images with one of them being called "Sharelatex". By going inside the Sharelatex container either via command or Docker Desktop, you can interact with the Tex Live package manager, TeX Live Manager. Using the tlmgr command you can update TeX Live and install/update LaTeX packages.

9.3 Installer

In support of our primary goal of increasing the ease and simplicity of installing Underbranch compared to Overleaf, the team designed a streamlined visual installation process. A core part of this process is the visual installer, for which a mockup was created. The two purposes of this installer are to verify the installation of the Docker daemon on the user's machine (or provide a simple instruction set to carry out the installation if verification fails) and to then automate the installation of Underbranch, including its Docker image dependencies.

In practice it doesn't make sense for the user to install every package since that can take over an hour and it is unlikely that all 4000+ packages will be used. As part of the installer, we could have the user specify which packages they want installed and this could be done during or after the initial Underbranch installation.

Figure 9.2: Mockup of the Underbranch visual installer

9.4 Next Steps

The next steps will focus on collecting user feedback to improve the current demo and refine our user needs. The team plans to conduct user testing to improve functionality along with the overall experience of our stakeholders. We will prioritize the most impactful changes based on the feedback we gather. In addition, the team will focus on testing the application to ensure optimal performance. We want to focus on the installer process to maintain efficiency through automated package installation. By combining feedback and testing, we plan to deliver a reliable prototype that meets all the expectations of Underbranch users. Below you can see an itemized list of our next steps:

1. Finalize Installer Plan
2. Automated Package Installation for Local Compilation
3. Automate startup/shutdown of Docker container by the desktop app
4. Develop and refine installer
5. Collaboration Feature
6. User Acceptance Testing

Chapter 10

Development Environment Setup

– Ben Knobloch, Lauren Kibalo, Marcos Morales, Noah Spina

10.1 Cloning the Repository

To set up your development evnironment, you'll first need to clone the current Github repository at this link (https://github.com/beknobloch/overleaf-toolkit). Make sure to open the repository in an IDE of your choice.

Now, feel free to make edits as you wish. Files with the UBR prefix are ones pertaining to Underbranch. main.js is the other important file developers will use. Every other file is a clone of the overleaf-toolkit library.

10.2 Running Underbranch

To run Underbranch, developers will need to run 3 commands in the terminal.

1 - (if you've never built the project) See the following command :

```
$ rm -rf node_modules dist
```

2 - See the following command :

```
$ npm install
```

3 - See the following command :

```
$ npm run build
```

After the third command, Underbranch should open in a local window on your computer.

Glossary

ElectionJS An open source framework that is used to develop desktop apps in JavaScript . 19, 21, 22, 23, 24, 73

LaTeX A software system for typesetting documents. Its markup language allows for more specialized symbols and intelligent documents than standard word editors.. 43, 44, 48, 74

Must Have This defines the first highest priority requirement. All of the tasks, requirements, or anything that is marked this way are build in the current version. 43, 44, 45, 46

Overleaf The industry standard open-source LaTeX editing environment and compiler. . 25, 26, 41, 43, 73

Sharelatex Container running when using Underbranch locally. Allows you to access TeX Live for package installation. . 74

Should Have This defines the second highest priority requirement. The system should implement all of the tasks, requirements, or anything that is marked this way, but if resources are limited, it can be left out of the current version. Build in next version. 44, 46

Simultaneous Document Editing Allowing two or more contributors to edit the latex code of a document in real time. Then being able to compile the document when ready. . 43

TCP Connection connection between two devices or applications that allows for two-way communication. 43

TeX Live Manager Package manager that is bundled with TeX Live and is usually abbreviated as ”tlmgr” . 74

TeX Live Software that allows a user to work with LaTeX code and provides the LaTeX package manager: TeX Live Manager . 74, 77

TeXnic Center An existing open-source LaTeX editing environment and compiler. . 26

Visual Studio An integrated development environment (IDE) that is a popular choice for developing in C++ . 26

Bibliography

Index