Requirements Specification

November 4, 2023

DENDRO-DAWGZ

|Growing A Brighter Future|

- Team Members -

Nile Roth

Niklas Kariniemi

Zachariah Derrick

Asa Henry

- Sponsored by -

Prof. Andrew Richardson, SICCS/ECOSS

Prof. Mariah Carbone, ECOSS

Prof. George Koch, ECOSS

Austin Simonpietri, ECOSS

- Team Mentor -

Italo Santos

Version 1.2

Team Lead Signature: [illegible]

Client Signature: [illegible]

Table of Contents

1 Introduction

Dendrology is the study of trees. From Encyclopedia Britannica, one can understand the history and use of dendrology more broadly as the study of the characteristics of “woody plants” such as trees, shrubs, and liana. Dendrology has also been referred to as “forest dendrology” or “xylology”. In recent years, dendrology has been appropriated by capitalism to investigate the relationship between these plants and the goals of industrial forestry by identifying “economically useful woody plants”.

Focusing on the United States of America, forestry has been a big industry since at least the 19th century, and the indigenous people have practiced forestry for millions of years before contact. However, in the coming centuries, foresters would become increasingly alienated from the subject of their study as Capitalism led to swaths of land being cleared for industrial purposes. Approaching the 20th century, the Forest Reserves Act (1891) and the Organic Act (1897) were passed as the consequences of Capitalism became reality, and the “Society of American Foresters” (SAF) was formed in 1900 by Gifford Pinchot. The society’s mission is to safeguard the United States’ forests with science, education, and technology. Today, forest conservation efforts and attempts to learn about the life contained within forests manifest in many ways. Dendrology is one such way conservation efforts manifest.

Our client includes researchers from the Center for Ecosystem Science and Society (ECOSS) and the School of Informatics, Computing, and Cyber Systems (SICCS). Professors Andrew Richardson, Mariah Carbone, George Koch, PhD Student Austin Simonpietri are ecological researchers with an interest in studying how climate change is affecting terrestrial ecosystems. Studying how a changing climate affects Earth’s ecosystems, and how the life within the ecosystems respond is an integral piece in understanding the relationship and role different trees interact with the planet’s climate.

2 Problem Statement

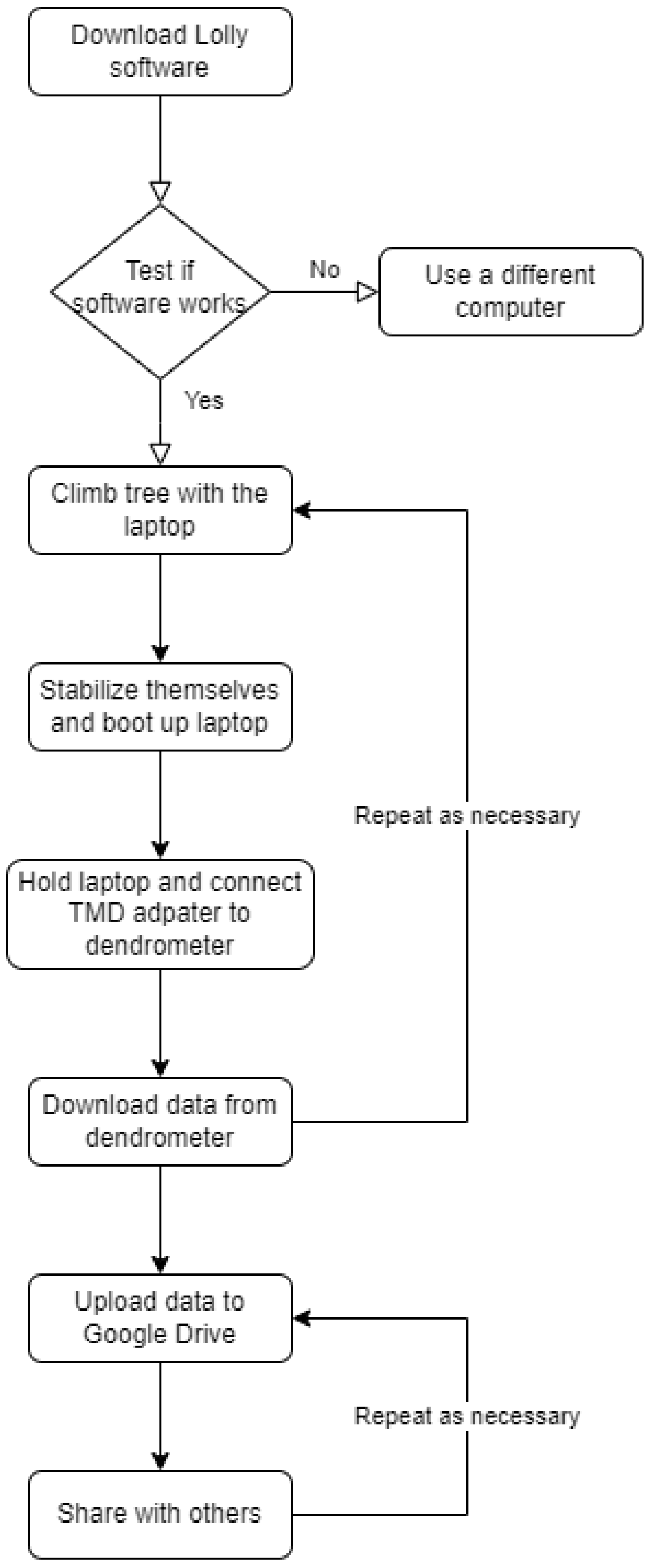
Our sponsors depend on these tree growth instruments for several areas of their research, and must use TOMST's software - the only system available - to read in data and view it. TOMST is a company that specializes in durable data-logging devices, such as security guard readers, dendrometers, and soil moisture content readers. This software works well for certain use cases and does the basics, but there are several main components in this system that introduce frustration, equipment risks, and potentially even danger for our sponsors and other users of the application. Our system intends to solve all of these issues, and provide a better experience for our clients and other end users.

The typical current business flow is as follows: the end user downloads TOMST's Lolly software onto their Windows machine. The user must then test to see if the software will run on their specific device. This is because the application does not work for all Windows devices, and there is no way to predict functionality. The user climbs the tree to the dendrometer, with the Windows machine on their person. The user stabilizes themself and opens the machine, then starting the Lolly application. Assuming the Lolly application is compatible with their device, the user then holds the laptop open in one hand, and manually holds the TMD adapter to the dendrometer with the other until all the data is read in (potentially longer than a minute). Once the data is read in it is downloaded to their machine, and they can climb to the next dendrometer or get off the tree. If they want to share the data with other people, they must upload the data to Google Drive, send it over to the other person, and they must download it. This data sharing process alone has taken them over 8 hours in the past for a single expedition. To view the data in any useful way, they must run a custom script on the downloaded file, and implement any statistical analyses on Microsoft Excel or Google Sheets. The data is restricted to single dendrometers as well, which introduces considerable frustration and effort when the user wants to see data comparisons on dendrometers from the same tree, between all dendrometers on separate trees, or all trees within a site.

The typical workflow of our clients is very limited, lacking several components that could make their research safer, cheaper, and much more straightforward. Specifically, the main issues that we will address are:

- Device portability - our clients are limited to running software on Windows machines, which do not all work with the software.
- User and equipment safety - our clients must bring a laptop to the top of Redwood trees, and use a two handed device while manually holding a cable, all while at 300 feet in the air. This obviously introduces several equipment and safety risks.
- Data usefulness - the data that is downloaded is limited to single dendrometers, and is improperly formatted for typical statistical analysis.
- Data sharing - sharing data with other researchers takes considerable time and effort, and should be much quicker and easier.

Figure 2.1 | Process Workflow Diagram


Download Lolly software
Test if software works
No
Use a different computer
Yes
Climb tree with the laptop
Stabilize themselves and boot up laptop
Hold laptop and connect TMD adpater to dendrometer
Download data from dendrometer
Repeat as necessary
Upload data to Google Drive
Share with others
Repeat as necessary

3 Solution Vision

To address the issues stated above, the team envisions a mobile application to address the current safety concerns which come with collecting data from dendrometers using the currently available software. To better facilitate research and information dissemination, the team is proposing to add onto what TOMST has already developed by allowing visualization of data, allowing multiple datasets to be visualized, providing tools to run analysis on the data, and utilizing the cloud to facilitate data sharing.

The following features of our solution seek to take advantage of mobile devices to reduce the risk which comes with data collection. By targeting Android smartphones, the user can put their safety first with little cost if a smartphone is lost; additionally, the handheld formfactor needs no stabilization meaning the user can focus on keeping the connection to the dendrometer to collect data. By making use of visualization technology, the team plans to provide the client with a way to create interactive graphs from the data collected, as well as create the capability to have two or more datasets from a dendrometer. The application will make use of a database API to also give the client the option to directly upload data and graphs to a place where access can be easily managed as well as a place which is easily accessible by others.

3.1 | Specific Solution Features

- Collect and store data from a dendrometer using the FTDI API to communicate with the dendrometer;
- Make use of metadata to make data files more easily parsable, as well as add information about the dendrometer so a user can identify the dendrometer. This will allow users to easily identify the tree from which data was collected;
- Visually represent data from a dendrometer using the MPAndroidChart package to create interactive graphs;
- Ability to have two or more datasets for simultaneous visualizing and analyzing;
- Ability to upload data and visualizations to cloud for easier shareability using an API;

3.2 | Course Flowchart for DendroDoggie

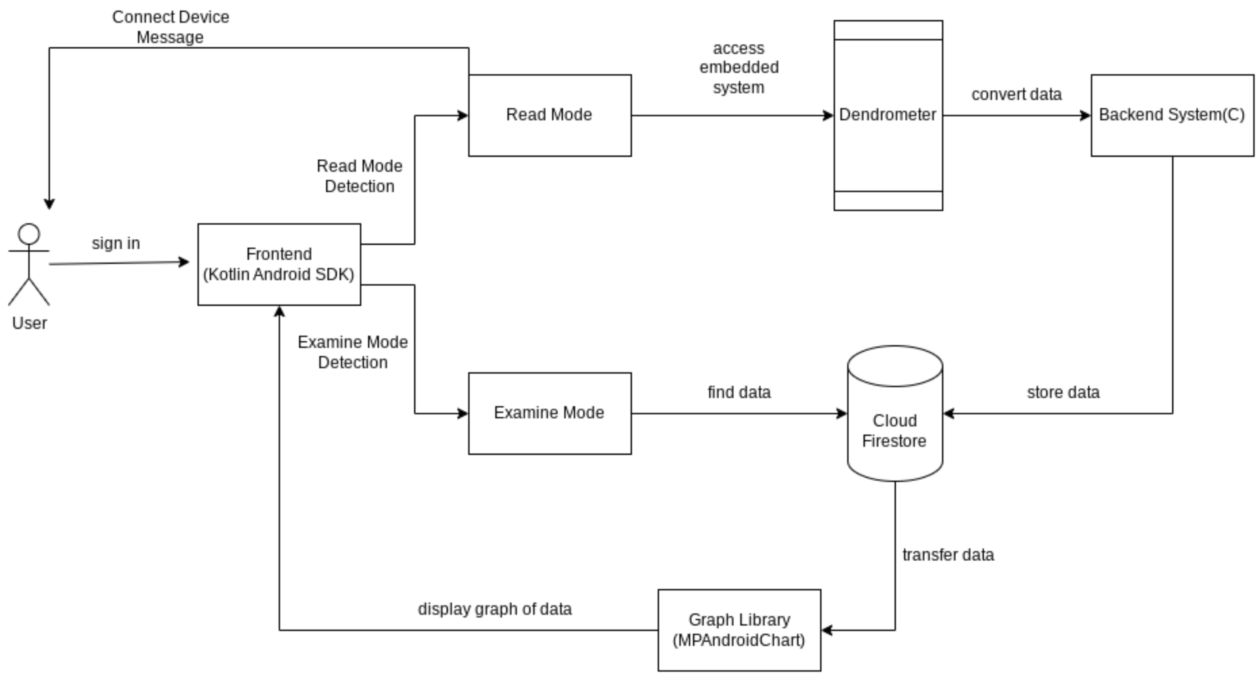


Figure 3.2: Flowchart for current vision of application use

Figure 3.2 describes the general flow for the DendroDoggie application. In either case, the user must provide credentials in order to use the application. From there, DendroDoggie provides a compact interface for a dendrologist or other professionals to collect data from a dendrometer. It also gives the user the ability to analyze data from one or more dendrometers, save, and share the visualization and data with others via a database. In the former case, the application will wait for a connection to a dendrometer, then proceed to seamlessly transition to downloading data. The application will wait for the user to give the command to disengage the collection process allowing for a seamless collection. In the later case, the application will already have data stored at a designated place in the file system. The user will be able to pick from the files stored under the application directory in the file tree, and can specify what analysis to run. When complete, the user can either export the resulting visualization to the device, or upload the data to a database.

4 Project Requirements

In order for our team to implement an application capable of reading in data from a dendrometer and displaying it to the user in a manner acceptable to our clients, we must follow the following outlined domain-level requirements. These key requirements are split up into functional requirements, non-functional requirements, and environmental requirements, and will be evaluated in the following sections.

Key Requirements

- The application will be implemented on a mobile platform, specifically Android.
- The application will contain a backend written in a low-level language, and be capable of reading in data from a TOMST point dendrometer.
- The application will implement a frontend that initiates data-reading from the dendrometer, and downloads the data to the appropriate location.
- The frontend will contain a graphical user interface capable of displaying the dendrometer data.
- The application will support metadata creation and a master file grouping system, so that several dendrometers can be tracked in the same file and by the same graphs. Additional information will also be provided with each set of data.
- The application will implement a cloud export and sharing system, to allow the users to upload data to the cloud and share with other users.

4.1 Functional Requirements

The basic product the team is aiming to produce is an Android application which can interact with a TOMST dendrometer to pull data and save the data to the device. The application should provide an interface to select 1 or more datasets and another one to create a graph (visualization) of the selected data. The user should be able to run a variety of analyses on the graph, and should also have the capability to save the resulting graph to the device. The users should additionally be able to select a window of time that they would like to view or run statistical analyses on. Lastly, the application should provide a way to upload the data to a cloud service for easy sharing.

4.1.1 | Data Visualization

Creating a visual representation of the data collected requires there to be what can be understood as a canvas onto which data signifiers (i.e. points, lines, etc.) can be drawn. Knowing what data is desired can be handled with the “data selection” function – and will be covered later

– for now we can assume there is some dataset. Depending on if the format of dendrometer data is unique or standardized, the application will first need to pull the data from the file into memory in a parsable structure. Then this structure can be queried for x- and y-axis names and values to add information to the visualized graph.

- Use Case: User uses the application to run linear regression on data from a single dendrometer
- Actors: User, Application
- Process Flow:
    - User chooses "Visualize"
    - Application loads *Visualization* screen
    - User loads dataset from dendrometer
    - Application updates *Visualized Data* canvas with selected dataset
    - User chooses "Analyze"
    - Application presents *Analyze* menu
    - User selects "Linear Regression" algorithm
    - User chooses "Select"
    - Application run *Linear Regression* algorithm on new dataset
    - Application updates *Visualized Data* canvas with analyzed dataset
    - User chooses "Export"
    - Application presents *Export* menu
    - User enters a filename for visualization
    - User chooses "Export"
- Preconditions:
    - There is at least one dataset from a dendrometer present
    - Application has a designated directory for reading/writing data and visualization
- Postconditions:
    - Application now holds user's visualization at <path>/<filename>

4.1.2 | Data Reading

The most critical requirement our project should satisfy is a successful transfer of data. Our software must have the ability to accurately obtain and store the data file created by the dendrometer. This process is perilous due to the necessity of precise translation. When uploading data from the dendrometer, csv (comma separated values) files are produced containing raw data with low readability. A third csv file must be created by the application to present the data in an organized structure. The goal of this translation is to support the process of reading from the file and inputting to the chart. With this goal achieved, the chart creation should be seamless and require only simple file reading expressions.

- Use case: User transfers data from dendrometer onto their mobile device.
- Actors: User, Application, dendrometer, computer
- Process Flow:

- User opens application on their mobile device
- User chooses 'Create New Set'
- Application halts and waits for connection to dendrometer
- User connects dendrometer to mobile device via USB to TMD adapter cable
  - This step may require the addition of a USB to USB-C adapter depending on the user's mobile device.
- Application identifies connection and begins uploading process
- Computer generates CSV files based on dendrometer commands
- Application compiles these files into a single, more user readable CSV file

- Preconditions:
  - Dendrometer has measured data to upload
  - Adapters are fully functional with no ware that may cause faulty connection
- Postconditions:
  - Application now holds user accessible data that was downloaded from the dendrometer

4.1.3 | Data Selection

Obviously, data visualization does not work if the user cannot choose what dataset(s) they would like to visualize and run analyses on. On the other hand, the selection interface needs to be relatively effortless to navigate. Therefore, a canvas should appear over the chart when a button is clicked. On the canvas, the application will show a hierarchy of the contents of its designated directory where graphs are grouped and stored (the root of the hierarchy being the application's designated directory). As the user clicks through directories, the contents will be right justified to signify they are children of the selected item, and the most recently selected item will be highlighted.

- Use case: User loads one or more datasets for analysis and visualization
- Actors: User, Application
- Process Flow:
  - User chooses "Add Layer"
  - Application presents *Load Data* menu
  - If desired dataset is in a directory
    - User selects directory
    - User repeats until desired dataset is found
  - User selects dataset
  - User chooses "Add"
  - Application closes *Load Data* menu
  - Application updates *Visualized Data* canvas with dataset
- Preconditions:
  - There is at least one dataset from a dendrometer present
  - User has navigated to the *Visualization* screen

- Postconditions:
    - *Visualization Data* canvas is populated with the chosen dataset(s)4.1.4 | Data Storage

A big part of the program will be the ability to store data taken from a dendrometer, and store it on the cloud. More specifically, the application will take any inputted data and store it on the cloud service AWS. The data will be stored within AWS on a database called DynamoDB, which is a very fast and scalable NoSQL database. This gives the application the best possible performance when it comes to moving large amounts of data. It will also give users the ability to access their uploaded data from any device, not just the device used for uploading the data.

- Use case: User wants to store data on the cloud and not on a physical device
- Actors: User, Application, Cloud
- Process Flow:
    - User clicks the share button
    - User will need to input username and password
    - User clicks on login button
    - User selects where in the database they want to save the data
    - User clicks on share button
    - Application displays a progress bar of the upload
    - After data is uploaded user can click on the back button to return to the visualization screen
- Preconditions:
    - Data from the dendrometer, an account to link data to
- Postconditions:
    - Data will be stored on the cloud database

4.1.5 | User Authentication

To support the process of storing data, the users will have to sign in. Users will be given a couple different options to sign in. They can either use a google account or they can create an account. If they create an account that info will be stored securely on the cloud. When users sign in they will be able to see all the data that is linked to their account. If a user uploads data from a dendrometer, they will be the manager/admin of that data. These administrators can add any other users to be viewers of that dataset. This will make sharing the large amounts of data very easy and efficient.

- Use case: User links data to their account and shares data with other users
- Actors: Users, Application, Cloud Authentication
- Process Flow:
    - Application prompts user with sign in screen
    - If user already has log in or chooses to log in with google
        - User will input username/email and password
        - User will click login button

        - Application will verify if information is correct
    - If user does not have an account or does not want to use google
        - User will click create account
        - User will then input an email, username, and password
        - User will click finish creating account button
        - Application will run a verification on the email
- Preconditions: None
- Postconditions:
    - User will have access to all data linked to their account.
    - User will have an account they can link data to.

4.1.6 | Multiple Layered Graphs

An important feature of our application is the ability to layer multiple different data sets on a single graph. This allows for the comparison of multiple dendrometer readings simultaneously. This is especially useful when multiple dendrometers are installed on the same tree or in the same vicinity. Many research projects focus on studying these circumstances and therefore require these overlapping capabilities. Merging several dendrometer data files together is also an important function that should be implemented in addition to layering several files together, allowing the end user to save and combine several dendrometers permanently.

- Use case: User adds a new data layer to their graph
- Actors: User, Application,
- Process Flow:
    - User selects “Add Layer” button
    - Application accesses database
    - Application displays list of directories/files
    - User selects desired data set to add
    - User selects “Add To Graph” button
    - Application adds new data set to existing graph
    - Application displays multi-layered graph
- Preconditions:
    - User is logged in
    - User has an existing graph opened in the application
    - Data sets are already uploaded from the dendrometer to the Application’s database
- Postconditions:
    - Multiple data sets visible on the same charting plane.
    - Remove layer action becomes available

4.1.7 | Saving Created Graphs

Saving a graph in the application entails uploading a raw data file onto the database. This is a short and easy process that only involves the application and the database. This feature is obviously crucial to the productivity of our software, and therefore must be consistently functional. If a user is unable to save his/her work, they are unable to reference any previous findings which will be a hindrance to research. Saving capabilities will not only help to support progress, but it also creates feasibility for a graph sharing feature. There should additionally be a way to merge and save a new file with the dendrometer information from two or more dendrometer data files.

- Use case: User Saves a graph to the database
- Actors: User, Application, database
- Process Flow:
    - User selects "Save" button
        - In the case graph is new (not yet saved) button will display "Save as"
        - Application displays new dialogue box with current working directory and a text box for filename
        - User clicks through directories until at desired file path
        - User inputs desired filename
        - User selects "Save" button
    - Application uploads graph file into database
- Preconditions:
    - User is logged in
    - User has an existing graph (with data) opened in the application
- Postconditions:
    - Graph file is stored into the database

4.1.8 | Sharing Created Graphs

Once the user has one or more datasets and visualizations they wish to share, there needs to be some way to upload them to the remote database so others can view and download that information. This would be done through a two screen interface. The application needs to know which user is uploading data, what data they wish to upload, and the destination database. The first screen will ask the user to provide a username and password. When login is successful, the user will then be presented with a list of directories and files at the application's path, and a list of databases to which they can upload.

- Use case: User wants to share a visualization from two dendrometers with their colleagues
- Actors: User, Application, Database
- Process Flow:
    - User chooses "Share"

- Application loads *Share* screen
- If desired dataset is in a directory
  - User selects directory
  - User repeats until desired dataset is found
  - User selects dataset
- Otherwise, user selects dataset
- User selects "Database 1"
- User chooses "Share"
- Application presents *Upload Progress* menu
- Application starts upload process
  - User waits for upload process to complete
- Application finishes upload process
- User selects "Back"

- Preconditions:
  - One or more datasets and or one or more visualizations are present
  - User is logged in
- Postconditions:
  - A copy of the given data is stored in the cloud

4.1.9 | Deleting Created Graphs

Deleting a graph from the database is also a simple process. A user may want to delete a file in order to free up space, erase irrelevant data, and/or maintain a more organized work space. This process only requires a quick access and modification of the database. Users should have the option to delete a graph anytime their stored data is displayed to them. This includes when attempting to open and share a graph. The deletion process from the user's perspective is quick and easy, only requiring clicking a couple buttons.

- Use case: User Deletes graph from database
- Actors: User, Application, Database
- Process Flow:
  - Application displays file system
  - User selects desired file to delete
  - Application provides user with actions (one being delete)
  - User selects trash can icon (delete)
  - Application sends query to database to remove file
- Preconditions:
  - User is logged in
  - Database is displayed from an attempt to open or share a file
  - Database contains at least one file
- Postconditions:

- ○ Deleted file no longer exists in database

4.2 Non-Functional Requirements

4.2.1 | Swift Rendering Times

Users desire a program with minimal stand-by time. Little to no time should be spent waiting for a response from our application. Charts should be rendered quickly despite a large data set. For massive sets, we may require the use of parallel programming. We can use multiple threads to input data into our chart simultaneously. Any wait time on any front of our application should be preceded with some sort of message informing the user of what they are waiting for.

4.2.2 | Data Stored Securely

One of the most important things to consider when implementing user-specific data storage is the security of the data in every step of the process. Users should be confident that their data is secure and protected both in the cloud and in transit, and our application must ensure this safety. Thankfully, Google Firebase - the cloud provider we are using with our application - implements several contemporary approaches to data security. All of Firebase’s services are certified by ISO and SOC security compliance standards, which shows they are considered secure by official sources. In transit from the user’s device to the cloud database, data is encrypted using HTTPS. When data is at rest in the cloud database, the data is also encrypted using the AES-256 encryption algorithm.

4.2.3 | Simple and Aesthetic User Interface

Our application will hold an architecture that is easily navigable and understandably functional. Users should know the functionality of all of our features without having to learn through testing. To accomplish this, all buttons and other means of I/O in our application should be descriptively labeled. We plan on creating a ‘FAQ’ or ‘help’ page in order to further improve user clarity. Along with a simple user interface, we also commit to providing a clean and appealing aesthetic to our application. This will require strict attention to color theory and CSS during implementation. Our webpage will hold an organized format that is pleasing to the eye, and interactively logical.

4.2.4 | Interactive Graphs

Being able to visualize data is very important to understand relationships between factors such as temperature and tree growth. To enhance the capabilities of data analysis, the application will offer the user ability to adjust the view and to zero in on points of interest. This will require some understanding of gestures so that screen input has an effect on the graph. In addition, the team would need to convene with the client to decide on a simple and easy gesture for zooming in and out of an area on the graph.

4.2.5 | System Reliability

Having an application that is consistently functional and available is an important factor. A user should be able to enter the application without the occurrence of any issues or crashes. A user should also be capable of using any of the features the app offers without fail. If a user wants to load data from the database, they should be able to without any issues arising. Success should also be assured when a user wants to share any data. A user should also be able to trust the analysis done on any of the data. The algorithms should be fully functional and always provide correct results. All of these features included in the application should be working 99.9% of the time to help create an enjoyable and useful experience for the user.

4.3 Environmental Requirements

4.3.1 | Restriction to Android Devices

Our mobile application will be limited to the Android platform due to the Apple ecosystem and its general requirements surrounding connected accessories. In order for a connected device to be able to be used, it must fall into a strict program laid out by the company, known as the MFI (Made for iPhone) program. The company making the accessory must apply for this program and follow their additional regulations in order for any iOS device to work with the accessory. These regulations generally involve changing the circuitry, accessory software, and/or production process. TOMST has made it clear to us that they do not plan on joining the MFI program in the foreseeable future, leaving us with just Android.

4.3.2 | Restriction to C and Java FTDI Libraries

The TMD-USB, Temperature Measure Device, adapter that our application will be communicating with utilizes Future Technology Devices International Limited (FTDI) chips to communicate with the dendrometer. In order to read in data, we need a library to work with these chips - the only official library existing in C. An unofficial library exists in Java, and while it is certainly usable, the C library is generally the preferred option. Without using either of these libraries we would be forced to implement our own FTDI communication functions using standard USB libraries in the chosen language, which would be too long and unnecessary. Therefore we are restricted to using either Java or C to implement our FTDI chip communication.

4.3.3 | Restriction to USB TMD Adapter

Communication with the dendrometer is only possible through TOMST's TMD adapter. This is because there is a specific software flow that initiates and performs the communication that is set up through their adapter. Trying to use a separate two-pin connector would not work, because the translation of data and reading of commands is done through the TMD adapter. This means that we are restricted to using just TOMST hardware for all of our dendrometer communication, and would not be able to use any other accessory (like the Apple iButton).

4.3.4 | Restriction to Wired Connection

Communicating with the dendrometer can only happen through a wired connection, through the TMD adapter. Bluetooth, radio, or another wireless communication method would make the functionality of the data-logging instrument much better, due to the fact that users could access the device and data from the ground. Unfortunately wireless functionality is not supported by the dendrometer. TOMST has explained to us that bluetooth and other wireless communications are not being considered in the foreseeable future, so we will have to disregard this path and stick to wired connections.

5 Potential Risks

Potential risks exist in all imperfect systems. The consequence and likelihood of occurrence are two factors that determine the relevance of a given risk. How could a user be affected given a certain risk becomes reality? How likely is it that this risk becomes reality? These are questions to consider in order to appropriately determine the level of concern a risk should initiate. Our goal is to ensure the risks prevalent in our application have a minimal amount of consequence. When applicable, we should also guarantee that an alternative solution exists to possibly diminish a given risk. Our results of the risk assessment can be seen in Table 5.1.

Corrupted Data

An obvious risk with any data manipulating system is corrupted data. This risk is very relevant to our application due to the fact that we are reading, manipulating and organizing inputted data. This risk is initially present during the transfer of data from the dendrometer to the android tablet. A faulty connection to the device could potentially lead to the corruption of data. After the data is obtained, we translate the dendrometer csv files and organize it to provide our users with useful statistics. We will use the MPAndroidChart library to build charts and graphs from the translated data set. It is important that these charts perfectly reflect the outputted data from the dendrometers. If the data is skewed anywhere in this manipulation process, we are creating charts with incorrect points and therefore misinforming our users. If our application is not consistent with providing completely accurate statistics, then we have yet to produce something of use nor benefit. We should also consider the fact that the possibility of corrupted data exists in multiple different instances. For these reasons it is very important that we take precautions when implementing in order to diminish this risk. Data corruption is a *Level Two* severity due to its significant consequences and its high possibility of occurrence.

MPAndroidChart not supportive of large data sets

Dendrometers are sometimes set up in trees for years before any data is collected. Knowing these devices generally create measurements every 15 minutes, it is likely for a dataset to be extensively large. Our charting library should have the ability to support large data sets like these. A seamless rendering of charts should take place no matter the size of the set. Unfortunately, according to a post on Reddit, MPAndroidChart struggles to rapidly produce “extensively large” data sets. Because we don’t know the exact size limitations of this library (thousands vs millions of data points), we are disregarding the risk and committing to its use. We are allowing this because AChartEngine will work as a substitute library, and it is known to handle large amounts of data with ease. The use of this library may hinder our chart’s aesthetic potential, but it will assure the deletion of this risk. MPAndroidChart will most likely always display a graph to the user, but may struggle with rendering speed and quality. For these reasons, extensively large data sets can identify as either *Level Three* or *Level Four* severity, depending on their impact to our program. Because we have a backup plan available we conclude this task to be a severity *Level Four.*

Unable to Obtain Data from Firebase

There are multiple different scenarios in which the application may struggle to obtain data from the database that is on Firebase. One of these scenarios could be that the user trying to obtain the data is experiencing network issues. In order to access a cloud service like a database on Firebase, you will need access to the internet in order to speak to the database. Therefore, if a user is having network issues they will not be able to obtain data from the database. A user will be able to upload data because Firebase will save the data on the device temporarily and wait until the user gets a network connection, but this isn't the same for obtaining data. Another possible scenario would be that Firebase itself is down. Firebase is managed by Google, so the likelihood this system goes down is slim, but still possible. In this case, Firebase and its services would not be accessible to anyone. This is something that we cannot control since we do not manage the infrastructure for Firebase. Like any cloud service provider out there, there are limits and quotas for cloud service. Firebase sets a certain quota on requests to the service that developers must follow. If an application using the Firebase database makes too many requests, there will be a temporary stop on accessing the data. To fix this, you would have to contact Firebase and increase the quota. Some of the scenarios can be avoided, but for the most part there are not many precautions we can take. The consequences of this risk are not very impactful to the application as obtaining data from Firebase would only temporarily stop working. This leads us to give this risk a severity of *Level Three*.

Table 5.1 | Risk Severity Assessment

Risk	Severity	Description	Mitigation
Corrupt Data	2	Data must hold its original value throughout the processes of transferring, translating, and plotting. If the data is skewed anywhere in these processes, we are creating charts with incorrect statistics and therefore have yet to produce something of use or benefit.	We can ensure minimal potential of corrupted data by testing multiple edge cases. Our Program will be able to identify circumstances that increase the probability of this risk such as a faulty connection, production of garbage characters, and anything unusual about our translated data file.

Datasets too large for charting library	4	Due to the high frequency of measurement, It is likely for a dataset generated by these dendrometers to be extensively large. Our charting library should have the ability to support large data sets like these. A seamless rendering of charts should take place no matter the size of the set.	AChartEngine will work as a substitute library, as it is known to handle large amounts of data with ease.
Unable to obtain data from Firebase	3	There are multiple scenarios where a user won't be able to obtain data from Firebase. The most probable scenarios are when a user has no access to the internet, or the application reaches the quota set by Firebase. These scenarios don't pose that large of a threat, as for the most part not being able to obtain data will be temporary and fixed relatively quickly.	We can make sure we have a quota that will fit the needs of our application. We also have a backup cloud database, AWS DynamoDB, we can use if Firebase ends up causing a lot of problems.

.

6 Project Plan

In this section we will go over what our plan is to complete the project. Below you can find a Gantt chart [Figure 6.1] that displays when we plan to complete each milestone.

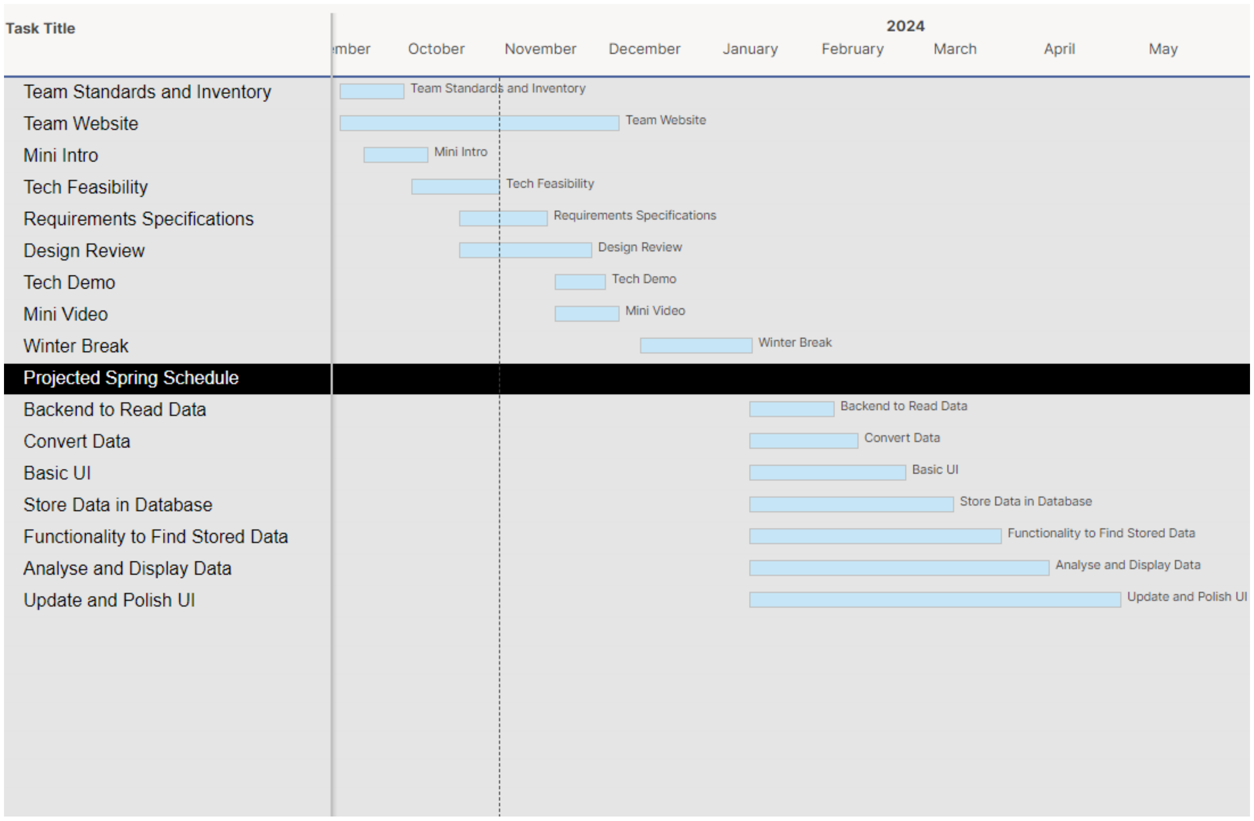
From 1/16/24 to 02/09/24, the first milestone we want to reach is implementing some backend code that is able to read in the data from the dendrometer. The data from this will need to be converted, which is the next milestone we want to complete by 02/16/24. More specifically, we want the data from the dendrometer to be easy to use and understand. Once we get the data sorted, we can start implementing the next milestone which is creating a basic UI and have it done by 03/01/24. Our initial goal is to create a UI that helps a user download data from a dendrometer. This would include things like a display for if a dendrometer is connected or not, or a button that when clicked would download the data.

After completing these first three milestones we will have a good foundation to build off. At this point we can focus on connecting the application to the Firebase database. We want to implement functionality that takes the data downloaded from the dendrometer and stores it in the Firebase database, Cloud Firestore by 03/15/24.

After adding this functionality, we can move onto the next functionality which will be finding and gathering previously stored data, and we plan to complete it by 03/29/24 . We want users to be able to see what data they have stored previously on the database. Included in this milestone will be adding user authentication. With this milestone completed we can add analysis and display of the data stored, which is the next milestone. Users should be able to do a multitude of different analyses on the data. To do this, they will have the ability to change what kind of graph the data is using. The plan is to have this done by 04/12/24.

After completing all of these milestones we can start working on the final one. Which is updating and polishing the UI. The goal here is to make sure the app looks good and is easy to use. This final milestone we want to have done by 05/03/24.

Figure 6.1 | Projected Development Plan



7 Conclusion

This team was formed to create a mobile version of an existing software which enables a better understanding of trees and the health of our forests. Currently, our client uses an application built specifically for Windows devices. Sometimes the application does not work, but more importantly our client is forced to scale trees, including the giant Red Wood, and stabilize a laptop while ensuring they are positioned appropriately in the tree while holding a device to the dendrometer. To add insult to injury, the current software does not have very much to support visualizing and running analysis on the collected data. Our goal is to create a mobile application which can alleviate these pain points, to an acceptable degree at least.

This document outlines the product the team plans to produce for Professors Andrew Richardson, Mariah Carbone, George Koch, and PhD Student Austin Simonpietri. Beyond the benefits to safety this application will afford, the application must have a way to visualize data from the dendrometers and run analysis, and allow to pull multiple datasets into one graph. The other most crucial function is the ability to push data up to the cloud for easy accession by colleagues and other personnel with a vested interest in insight gained from the data.

In the discussion throughout this document, the team identified risks which could pose a threat to the usefulness of this application. The document covers the potential for data to be corrupted, the potential for our chosen visualization software to not work for large datasets, and the potential for circumstances where we are unable to communicate with Firebase. The team is confident in our ability to implement safeguards which will identify corrupted data during collection, and create a solution to ensure the client gets the correct information from the dendrometer. As far as graphing, MPAndroidChart has been in active development for 7 years (since 2016), so many of the surface level, data corrupting bugs should be patched; furthermore, the team is confident in our ability to find techniques needed to protect data while running analysis. There are multiple scenarios where a user won't be able to obtain data from Firebase. The most probable scenarios are when a user has no access to the internet, or when the application reaches the quota set by Firebase. Fortunately these scenarios don't pose that large of a threat because the inability to obtain data will be temporary and fixed relatively quickly.

Overall, we are satisfied with the current state of our planning phase. Proceeding the completion and approval of this document, we will begin implementing a prototype for our application. Because of our collaboration with the TOMST company, an ideal release date has been scheduled for testing in early 2024. Proceeding customer experimentation, we will make any necessary modifications and have a version release in Spring 2024, ideally around March. We are confident that our pace is more than sufficient to have a fully-functional version ready to release by this time.

9 References

1. Encyclopædia Britannica, inc. (2023, October 24). *Forestry*. Encyclopædia Britannica. https://www.britannica.com/science/forestry
2. Northern Arizona University. (2021, March 22). *About*. The Richardson Lab. https://richardson-lab.nau.edu/
3. Petruzzello, M. (2018, August 20). dendrology. Encyclopedia Britannica. https://www.britannica.com/science/dendrology
4. Society of American Foresters. (n.d.). *Our Mission and History*. History. https://www.eforester.org/Main/About/History/Main/About/History.aspx?hkey=f112ee86-0f07-4cca-b342-b9d4bca0f535