Software Design Report

21 February 2020

Version 2

Sponsors

Dr. Kiona Ogle

Dr. Michael Fell

Mentor

Isaac Shaffer

TreeViz

Riley McWilliams

Qi Han

Haitian Tang

Daniel Rustrum

Alex Bentley

Contents

1. Introduction 2

1.1 Background

1.2 Problem

1.3 Solution

1.4 Purpose of this Document

2. Implementation Overview 4

3. Architectural Overview 5

3.1 Architecture Diagram

3.2 Brief Responsibility of Each Module

3.3 Features of Each Module

3.4 The Communication Between Each Component

4. Module and Interface Descriptions 7

4.1 Front-end Website

4.2 Tree Visualization

4.3 Model Manager

5. Implementation Plan 13

6. Conclusion 14

7. Appendix A: Spring Schedule 15

1. Introduction

1.1 Background

Climate change is a natural cycle of the Earth that impacts many ecosystems. However, due to the recent and rapid growth in industrialism, climate change is now being more affected by artificial causes than natural ones. A severe increase in the amount of carbon dioxide (CO2) emitted into the atmosphere by human-made technology is dangerously speeding up the effects of climate change on the planet.

Trees play an extremely important role in Earth's climate behavior. They cover around 50% of Earth's land area and contain upwards of 90% of the global vegetation carbon. This means that a lot of CO2 that humans emit is being taken in by trees and used for their growth. Observing tree growth patterns can lead to a better understanding of how trees and climate interact with each other. These observations can also be used to predict the effects of climate change in the future.

To better understand how certain factors affect trees, Dr. Kiona Ogle and Dr. Michael Fell of Ogle Labs developed a simulation that shows a tree's growth over time. The simulation is called the Allometrically Constrained Growth and Carbon Allocation (ACGCA) model. It uses over 30 input parameters to run the simulation, which calculates the state of the tree over time. The output of the model contains useful information such as the tree's height, trunk radius, the carbon in the leaves and trunk, etc. Currently, the model is purely used for research by Dr. Ogle, Dr. Fell, and Ogle lab associates, and is not being used for profit. However, they want to expand the use of the model to anyone with an internet connection, for free, allowing everyday people to learn about tree growth from it.

1.2 Problem

The issue is that the ACGCA model currently has a very small user base. This means that very few people are using the model to learn about tree growth. The specific problems that the current iteration of the model has are as follows.

- Not available online

  The **ACGCA** model is not available online. This limits the use to only those acquainted with Dr. Ogle or Dr. Fell.

- Unfriendly user input

  The model's input is command line based, which requires the user to have programming experience to run it.

- Requires knowledge in biology

  The user needs to know what each of the 30+ input parameters are, which requires experience in tree biology.

- Unfriendly output

  The model only outputs raw numerical data, which can be useful for biologists, but no one else. It is not visually appealing nor informative to non-researchers.

- User information

  Our clients want a way to keep track of what kind of people are using their model. They want to know information about the user such as if they are a student, teacher, researcher, etc and their general location.

- Low budget

  Our clients do not have a grant for this project and must pay for everything themselves.

The combination of these factors creates a user environment that is unfriendly and requires knowledge in specific fields. The fact that the model is not available online also contributes heavily to its low usage.

1.3 Solution

Dr. Ogle and Dr. Fell want to expand their audience so that anyone can utilize and learn from the ACGCA model. More specifically, they envision the model being used in a classroom setting where students can experiment with the inputs. This way, students can learn how certain factors affect tree growth. While students are the primary demographic of focus, researchers will also be considered during development as the original demographic of the model.

Our clients also want a way to track how the model is being used. They are interested in knowing if more students versus researchers versus everyday people are using it. They are also interested in the general location of the users. While mandatory, this information will only be used for analytical purposes and will not infringe on anyone's privacy.

Team TreeViz is tasked with making the model more user-friendly. To do this, we will create a website that will be hosted online for anyone to use. This will be a good way to get it into the hands of students and educators, or anyone else that wants to use it. The website will run the model and significantly increase the accessibility to it by overhauling the input and output. Specific features that will solve the project's problems are as follows.

- Available online

  The website will be hosted online so that anyone with internet access can use it. A ReST API will handle the passing of inputs and outputs of the model between the user and a server.

- User-friendly input

  The 30+ input parameters will be made up of text boxes and sliders instead of a command line. This will allow anyone to use it, rather than only researchers.

- Help boxes

  Each parameter on the input page will have a help box that explains the parameter in more detail. This will allow non-biologists to get a better understanding of what they do.

- Tree visualization

  The output will be a rendered tree visualization. This will be significantly more intuitive than the raw data and will allow many more users to learn from the model. The raw data will also be available to those who wish to use it.

- User surveys

  Users will submit a survey with some basic information so that our clients can see how the model is being used. This will help our clients develop the product further, allowing the prioritization of certain user demographics for future updates.

- Local desktop development

  To avoid costly solutions to our clients, the biggest change we can make is developing for a local server.

With this solution, we will give our clients the means to broaden their audience. This way, more people can use the ACGCA model to learn about tree growth. The website may be used in labs for researchers to conduct experiments and study climate change. It could also be used in a classroom where students are learning together about how trees grow. Team TreeViz is confident in our ability to deliver the specified product and help our clients get their model into the hands of more people.

1.4 Purpose of this Document

The purpose of this document is to lay out a blueprint for the design of the completed product. It will explicitly describe the main modules of the product as well as their roles and responsibilities. It will also lay out the implementation process with a timeline of expected milestones of completion. The goal of this document is to have a design specification that is thorough enough so that, theoretically, any development team can build the product.

2. Implementation Overview

The product we are developing aims to allow students or researchers to view the process of a trees' growth. Our clients have already created a model to show researchers tree growth and now they want us to make the interface more user-friendly. This will be completed in three parts: input, model manager and output. First, the input process will have to be more streamlined. We will create a web application with an easy-to-use graphical interface. Instead of entering input through a command line, users will enter all of the necessary data in text boxes and drop-down menus. Second, the input will be passed to the model manager to be computed by the ACGCA model. Third, the model's output will be passed to tree visualization to visualize a rendered tree.

To build this product, we chose the technologies listed below:

- VueJS is a web framework and library that allows for easier development by using real time web page updating and component based development. VueJS is becoming more popular every year and will be continually supported.

- Amazon Web Service (AWS) is a hosting environment that will act as the middleman between the model and the user interface. With AWS, we can implement our product without using ReST API.

- ThreeJS is a web framework that is useful for converting the ACGCA model's output into 3D rendered trees.

- Firebase is a database and authentication service that will give our clients a way to easily access users and their data.

AWS will host the website as well as perform as the model manager. When the website gets the input from the user, it will send a JSON object to the model manager. The model manager will convert this JSON object into parameters to run the ACGCA model. Then it will get the output from the model and convert it into a separate JSON object. The output will be sent to ThreeJS, which will convert it into a rendered tree.

Firebase communicates with the survey and login pages. It is used to store information from user surveys, storing them in a database.It also is used to authenticate user information for logining in.

3. Architectural Overview

Now that we know what technologies are going to be used to create the product we need to know how each of those technologies will work together. The architectural design of the product will show how the system will work and what components of the design will work with each other. We are using a client-server model structure. Our user will use their browser as the "clients" and initiate requests to our web server.

3.1 Architecture Diagram

The whole project consists of three main modules which are Front-end website, model manager and visualization. When users want to use our website, they need to complete the survey first. They need to use the authentication process, so that they don't need to take the survey again when they use our project the second time. The data they enter into the survey will be sent to firebase. After users take the survey they will go to the visualization page. They put in the values for each parameter and model manager will use the ReST API to take it to the ACGCA instance. Figure 1 shows the flow of how users use the website and how input gets to the visualization output. The arrows between each component or process means that either data is passing or one page leads to another.

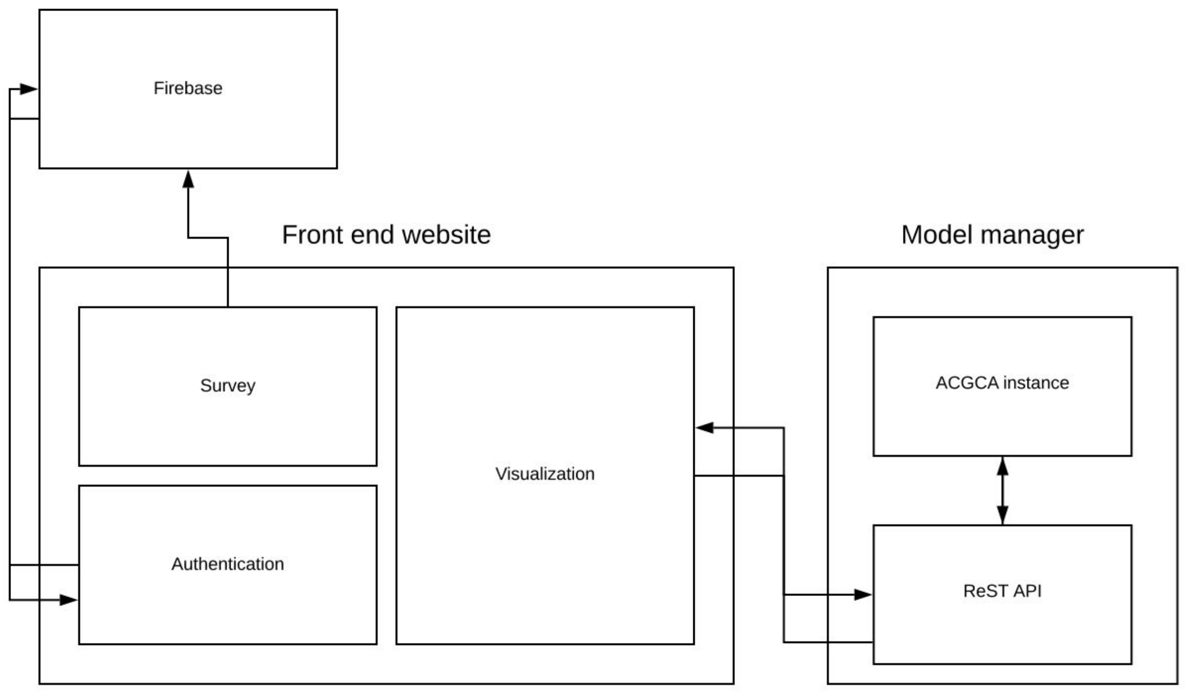


Figure 1: Architecture Diagram

3.2 Brief Responsibility of Each Module

Each module is responsible for providing certain functionality so that the whole system runs smoothly.

- Front-End Website: The website will be the environment that the user interacts with and will host a lot of the functions that make the product work. Each page should be clear and brief, so that users have an easier time using them. The front-end website includes a landing page, survey page, a login page for users, and a visualization page.

- Tree Visualization: The tree visualization is responsible for receiving the model's output from the model manager. It then takes this output and converts it into a graphical tree to be displayed to the user. There will also be 2D cross-sections of the tree, giving users a better idea of the size of it from a different perspective.

- Model Manager: The responsibility of the model manager is to connect the ACGCA model and the front-end website. This involves passing data from the survey page to the database, from the visualization page to the ACGCA model, and from the ACGCA model to the tree visualization process.
    - ACGCA Model: The ACGCA model is doing the simulation of the tree growth. It takes in different parameters about a tree and simulates the condition of the tree under a certain

condition which is defined by our user. The ACGCA model is owned and being provided to us by our clients. Therefore, it is not an independent module of our system.

3.3 Features of Each Module

- Front-end Website: With the front-end website, our user can interact with the ACGCA model. It will show users the result of the model's simulation in a visualized way. Users can also save their input or output on the visualization page. The survey page will collect the information that our clients want before users enter the visualization page.
    - Firebase: Firebase is used to store the user's survey information. It also stores the user's credentials in order to authenticate them in the future. Firebase is developed by Google.

- Tree Visualization: The tree visualization will get the result of ACGCA model from model manager and visualize it. The output of this process will clearly show the user what a tree is like under the condition that user defined with their input.

- Model Manager: The model manager passes information between each component. It needs to be able to pass the specific data to an exact component, so that each component can communicate with each other correctly.
    - ACGCA model: An ACGCA instance receives information as a JSON object and then runs the model. After the model finishes the simulation of the tree, the information about the tree will be sent to the tree visualization as a separate JSON object.

3.4 The Communication Between Each Component

The communication between the visualization modules is done through the model manager. This includes sending user input to the model manager which will run the ACGCA model using the input. The output of the ACGCA model is retrieved from the model manager and sends it to the visualization process.

4. Module and Interface Descriptions

In the section, we provide an in-depth description of each module, discussing all of the pieces necessary to build them. Accompanying these descriptions are UML diagrams, laying out the structure of the modules and how they are connected within themselves and to the other modules.

4.1 Front-end Website

The first major part of the product is the website which represents the "user-friendly platform" aspect of the project. The website is where all other aspects will be hosted and be accessible for the end user. A lot of the design work will be completed here. Each of the different aspects of the website will host different parts of the project that the client requires, such as tree visualization, login pages, etc. The website will solve the unfriendly input problem of the project by taking the complicated input parameters and changing them from command line code into text boxes, which will include text boxes to help the user understand the 30+ inputs as mentioned in section 1.3.

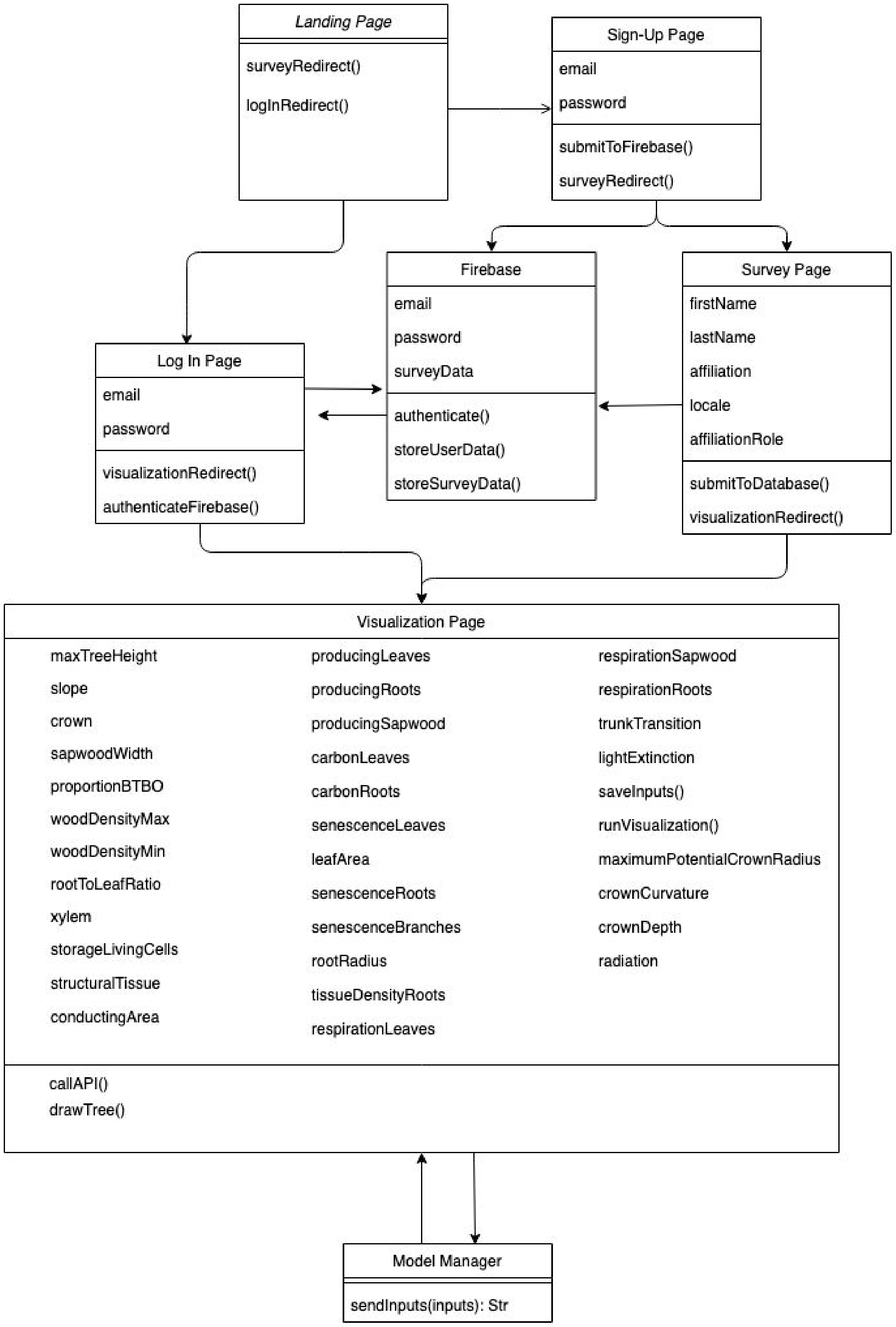


Figure 2: Website Sequence Diagram

The website module of the project will have several aspects. Figure 2 above shows the path the user will follow starting with the landing page. The aspects are as follows.

- The landing page

  The landing page will be the first thing that the user sees when they load the website. It will have sign-up and login redirect buttons that load different pages.

- The sign-up page

  The sign up page will consist of two text boxes for the user's email and password for them to manually sign up using whatever email they want. Their information will be sent to Firebase for future authentication. After the user signs up they will be redirected to the survey page.

- The survey page

  The survey page will consist of the user information that our clients' want to store to keep track of. The fields include first name, last name, affiliation, affiliation role, and locale. This information will be stored in Firebase's database. This page will issue a warning if the submit button is pressed but the survey is not filled out. If the survey is completely filled out then the data is sent to the database.

- Login page

  The login page is where a returning user can enter their email and password or use the Google authentication button to be redirected to the visualization page without needing to fill out the survey again. Firebase will use the user's information to authenticate their credentials. If the user is not found in the system they will be alerted that they do not have an account.

- Visualization page

  The visualization page is the main part of the project and will be the most demanding page. There will be up to 36 inputs for the visualization itself so there will be a lot of parameters to keep track of and pass to the ACGCA model through the ReST API. The "run visualization" button will send the inputs through the ReST API to the ACGCA model manager. Then, the model manager will return the outputs to be deciphered by the visualization software, which will render the tree in-page. There will also be an option for the user to save the inputs they entered as a JSON file in order to duplicate previous results.

4.2 Tree Visualization

The second module of the product is the tree visualization. It represents the "visualizing" aspect of the project title. This module is responsible for taking the output from the ACGCA model and visualizing it as a rendered tree for the user. It will be embedded in the front-end website, on the visualization page, so that users can get a quick render of a tree that is created from their entered input parameters.

A visualization will solve the unfriendly output problem of the project by making it more understandable and appealing. It will also help with the biology knowledge requirement problem by actually showing the user a tree, which is more intuitive than raw data.

Figure 3 is a UML diagram of the tree visualization module. There is only one component with a list of parameters and functions for it. The box on the left represents the model manager. The model manager uses the ACGCA model to process the user's input. Then the input is passed to the tree visualization component to be rendered for the user.

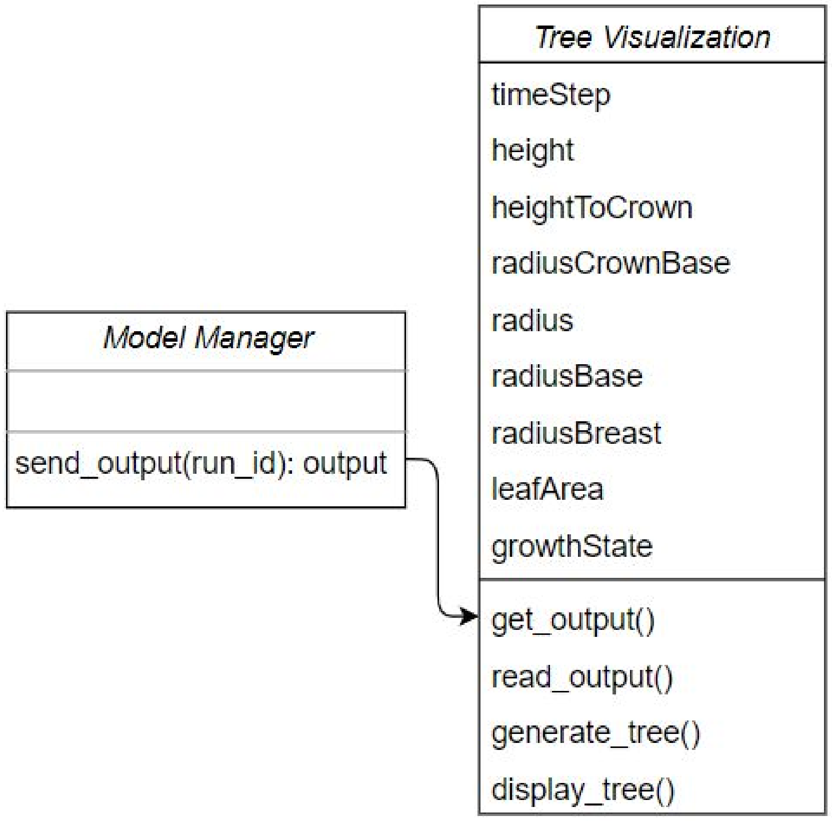


Figure 3: Visualization Software Diagram

The tree visualization module will provide several services. These are as follows.

- Retrieve and read the output of the ACGCA model

  The model's output will be passed to the tree visualization as a JSON. It will be parsed for its contents, so the data can progress to the generation step.

- Generate a tree based on the output of the ACGCA model

  Once the data is gathered, it needs to be turned into a tree. The data that can be visualized is listed in Figure 3, above. More detailed descriptions of each variable are listed below.
  - **timeStep**: The point in the tree's life that the current output represents.
  - **height**: The total height of the tree.
  - **heightToCrown**: The height of the tree to where the first branches are.

    - **radiusCrownBase**: The radius of the crown at its base.
    - **radius**: The radius of the tree's trunk.
    - **radiusBase**: The radius at the base of the tree.
    - **radiusBreast**: The radius of the tree at breast height (3.37m).
    - **leafArea**: The area of leaf coverage on the tree.
    - **growthState**: The state of the tree. (healthy, static, dead, etc.)

- Generate geometric parts of the tree for more detailed information

  The visualization will also have an option to view geometric cross-sections of the tree. This will allow users to get a better idea for parameters such as the radius of the trunk.

- Render the tree and display it on the website for the user to see

  Once a tree is generated, it will be rendered on the visualization page of the website. This will allow the user to visually see how their inputs can affect a tree's growth over time.

4.3 Model Manager

The third module of the product is the Model Manager. It represents the "tree growth" aspect of the project title. This module takes the input sent by the user (elaborated in Front-end Website Module), gives it to an instance of a wrapped ACGCA model, processes the input, and sends the output to the appropriate user. The output is used in the Tree Visualization Module.

First, the user's machine sends input from the user interface to the ReST API. That request is validated in the Validate Request function. From there, there are 3 functions that the user interface can use: Send Input or Get Input. If the user submits input on the visualization page, then the Send Input function is triggered. The input gets stored into the Input Queue and the user receives a unique ID referenced as a run ID to get the results of that run. If the user has a run ID, then the Get Output function is triggered and the run ID is validated by the Validate Run ID function. The Validate Run ID function either gets the correlated output or returns an empty output if the model output is not ready yet.

The ACGCA model instance is the combination of the ACGCA Wrapper and the ACGCA model itself. The model instance has its own set of operations that it can do within the ReST API. It has access to the Get Input and Send Output functions. The ACGCA instance goes through a cycle of choosing the Get Input function, doing some internal processing then choosing the Send Output function. The Get Input function simply grabs the input from the Input Queue and returns it. The Send Output function sends the results of the Run_ACGCA_Model to the ReST API which then is put into the Output Collection.

The internal process of the ACGCA instance is also a simple cycle of Request Input, Validate Input, Send Errored Output or Run ACGCA Model then Send Processed Output. The Request Input function requests the input from the ReST API. The Validate Input function validates if the input provided is formatted correctly, then either gives it to the Run ACGCA Model function or the Send Errored Output function. The Run ACGCA Model object processes the input and sends the output to the Send Processed Output function. The Send Errored Output function and the Send Processed Output function both send the output they have to the ReST API.

Even though each function has only a single operation they rely on each other to fully operate.

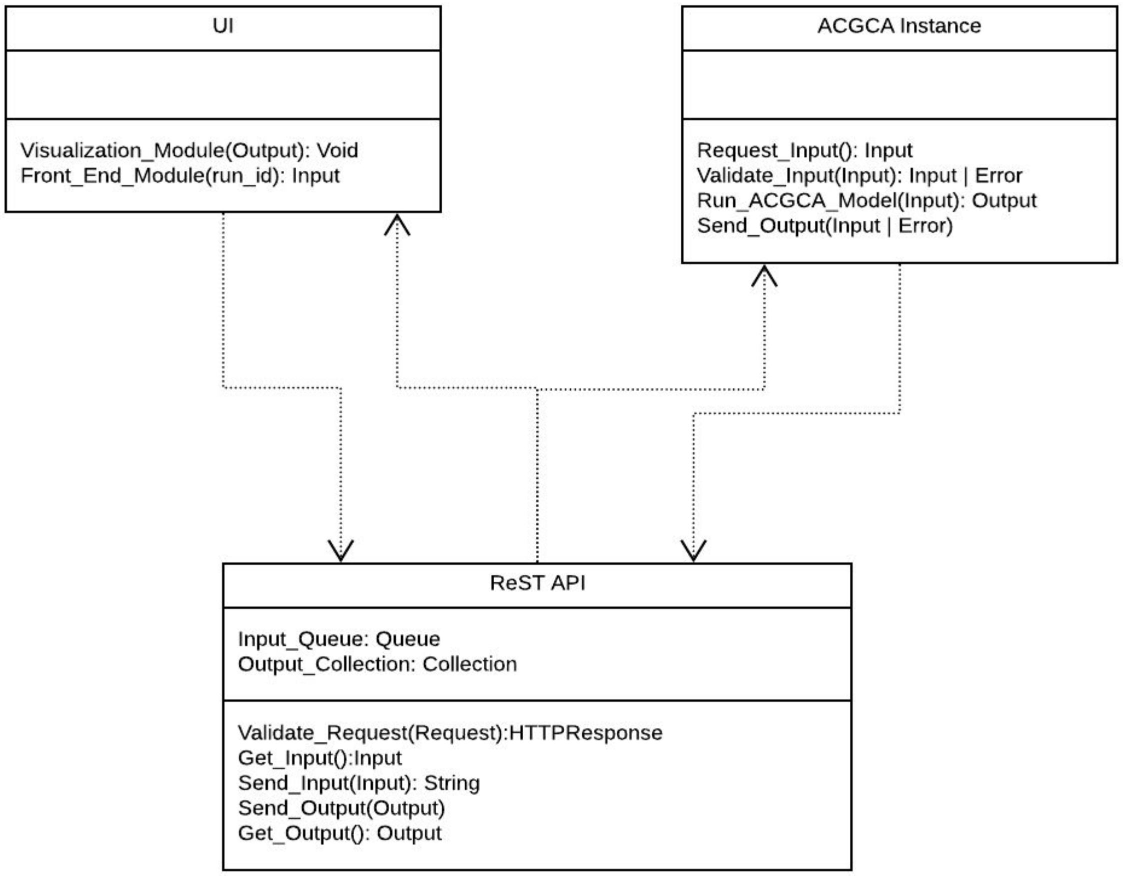

UI
Visualization_Module(Output): Void
Front_End_Module(run_id): Input
ACGCA Instance
Request_Input(): Input
Validate_Input(Input): Input | Error
Run_ACGCA_Model(Input): Output
Send_Output(Input | Error)
ReST API
Input_Queue: Queue
Output_Collection: Collection
Validate_Request(Request):HTTPResponse
Get_Input():Input
Send_Input(Input): String
Send_Output(Output)
Get_Output(): Output

Figure 4: Model Manager Diagram

The model manager module will provide several services. These are as follows.

- Route an input to an instance of the ACGCA model.
  - The input is put into a queue for the ACGCA instance to request later.
  - When an instance of the model is ready it requests the next available input from the queue.
  - The user is provided with a run ID used to request the output.

- Process the input into output using the ACGCA model.
  - This is achieved by a wrapper for the ACGCA model.
  - Inputs that are formatted incorrectly are filtered out and return an error response.
  - Once a valid input is found, the input is run through the ACGCA model and the output from the model is returned to a distributor.

- Route the output to the user that requested the result
    - This is done by a distributor mentioned earlier.
    - The output is put into a collection for the user interface to request. However, the output has a lifetime period. When this period is over the output is deleted and the user will have to go through the process again.
    - If the user interface requests the output using the a valid run ID mentioned earlier, then the distributor sends the output to the user.

- Provide the means for the user to send the input
    - This is achieved using a ReST API. The ReST API validates the request and the source it comes from. For example, the instance has a set of actions that only they can perform, so they have to send an API key to the ReST API to validate that the instance is indeed an instance and not something else.

5. Implementation Plan

In this section, we are going to provide a design-centric implementation timeline for our project. We designed a gantt chart to indicate our plan for implementing our product, shown in Appendix A.

Our team is developing a user-friendly interface to visualize tree growth for our clients. We divided the product into two parts, frontend (which includes the website and visualization) and backend (which includes the AWS environment and model manager). Daniel and Tang are assigned to backend while Riley, Alex and Han are assigned to frontend.

The first task is the host environment setup. This is the basis of our product where we can combine all our work together. As we mentioned in the implementation overview, we are going to use AWS as our platform to combine our work. Daniel has already created and shared an AWS account for developmental use. When it is time to deliver the final product, we will transfer the account to our clients.

The front end structure has already been implemented with most of the pages completed. The visualization page needs the most work while the rest of the website will be refined to our clients' approval.

For backend, Tang is working on calling the ACGCA model's functions from a dll file via python. The user input will be sent from the website to the model manager as a JSON object. After calling the model's functions, Tang will convert them into this JSON object to be passed along. The model's output will also need to be converted to a JSON object to be passed to the tree visualization module. Daniel is working on setting up the AWS server so it can send and receive data properly. Once that is complete, the next step is to get the model running, so that it can pass

All the tasks above are expected to be done before spring break. After spring break, all of us will be focusing on user testing and improvement.

6. Conclusion

TreeViz's clients, Dr. Ogle and Dr. Fell, built the ACGCA model to calculate the state of a tree based on numerous growth conditions. It is a complex system that can be used to study how certain factors affect tree growth over time. Due to a strong link between tree growth and climate change, this research can also be used to study and predict climate change and its effects on Earth. While this is the original motivation of constructing the model, our clients want to add to it by expanding their audience to everyday people. This way, students, teachers, or anyone else can learn about tree growth.

Unfortunately, the model is command-line based and requires specific knowledge in tree biology to utilize it, making it not very user-friendly. It is also not available online, restricting the use of it to only those acquainted with our clients. Another key problem is that the output of the model is purely data-based, which is useful for researchers, but not everyday people.

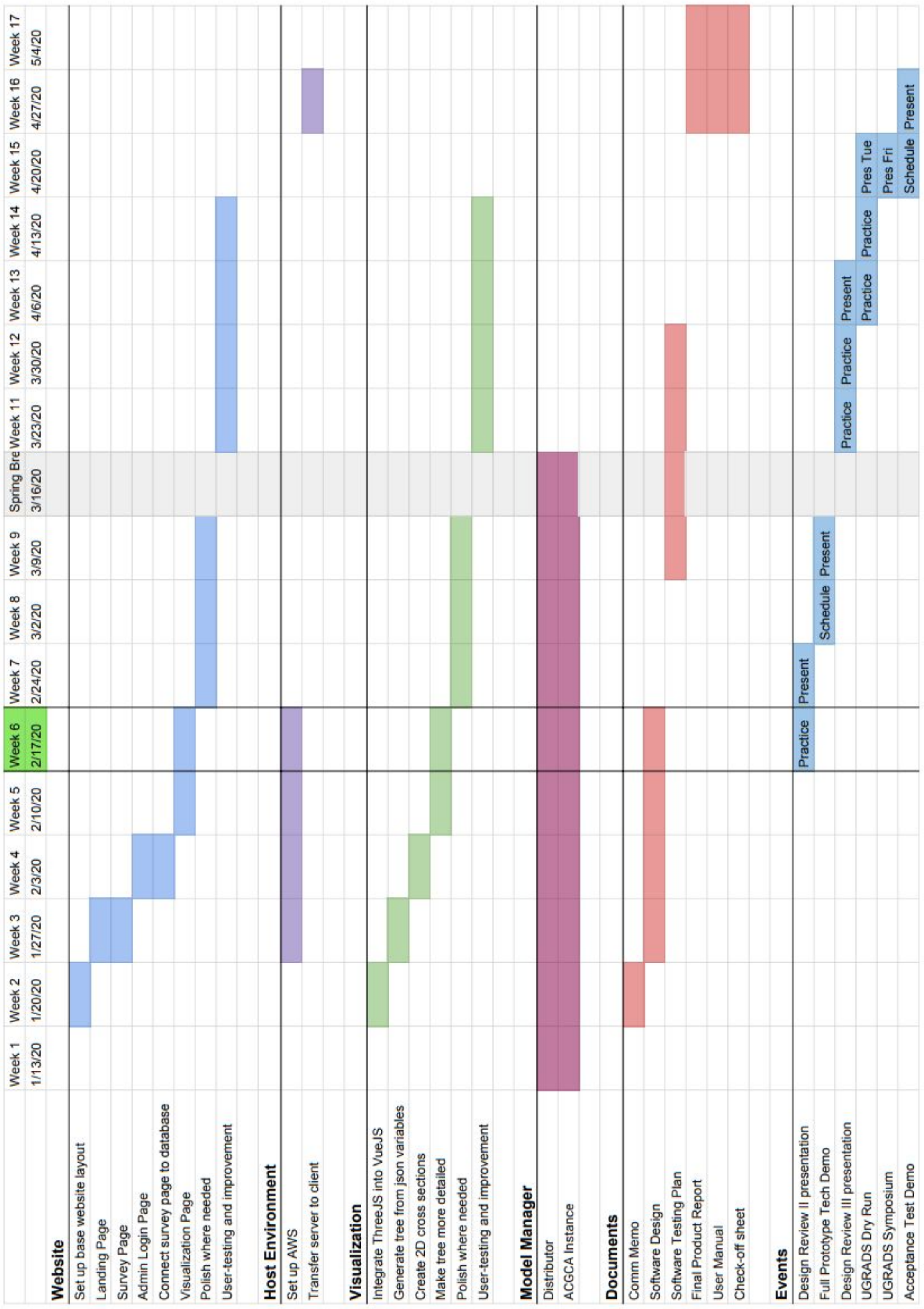
Our solution is to develop a website that hosts the connection between users and the ACGCA model. This will help our clients obtain the product that they envision. The user will be able to access the website from any computer and be able to adjust the various inputs that affect the visual output of the tree. Connecting the model to a visualization software will create an appeal for students to interact with the model and enjoy learning about tree growth.

TreeViz is now building and assembling all of the necessary pieces to deliver an effective product to our clients. So far, we already have the base website laid out with some pages developed in more detail. We also have the tree visualization software integrated into the website and ready for further development. The model manager is in progress and will soon be connected to the website, allowing the other modules to access the ACGCA model's functionality.

The next step is to get all of the pieces connected to each other. Then we will develop the functional details, making sure everything works together well. Finally, we will polish the website and make it more user-friendly via user-testing and feedback.

With this document, we now have a clear blueprint for what needs to be developed. This will allow us to plan which pieces to work on from week to week, and give us room to adjust where needed. Through continual, weekly meetings with our clients to update them on progress, TreeViz can effectively create the product they want. We are all excited to build a user-friendly platform for visualizing tree growth and are confident in our abilities to do so.

7. Appendix A: Spring Schedule



	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Spring Bre	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17
	1/13/20	1/20/20	1/27/20	2/3/20	2/10/20	2/17/20	2/24/20	3/2/20	3/9/20	3/16/20	3/23/20	3/30/20	4/6/20	4/13/20	4/20/20	4/27/20	5/4/20
Website																	
Set up base website layout		🟦															
Landing Page			🟦														
Survey Page			🟦														
Admin Login Page				🟦													
Connect survey page to database				🟦													
Visualization Page					🟦	🟦											
Polish where needed							🟦	🟦	🟦								
User-testing and improvement											🟦	🟦	🟦	🟦			
Host Environment																	
Set up AWS			🟪	🟪	🟪	🟪											
Transfer server to client																🟪	
Visualization																	
Integrate ThreeJS into VueJS		🟩															
Generate tree from json variables			🟩														
Create 2D cross sections				🟩													
Make tree more detailed					🟩	🟩											
Polish where needed							🟩	🟩	🟩								
User-testing and improvement											🟩	🟩	🟩	🟩			
Model Manager																	
Distributor	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪							
ACGCA Instance	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪							
Documents																	
Comm Memo		🟥															
Software Design			🟥	🟥	🟥	🟥											
Software Testing Plan									🟥	🟥	🟥	🟥					
Final Product Report																🟥	🟥
User Manual																🟥	🟥
Check-off sheet																🟥	🟥
Events																	
Design Review II presentation						Practice	Present										
Full Prototype Tech Demo								Schedule	Present								
Design Review III presentation											Practice	Practice	Present				
UGRADS Dry Run													Practice	Practice	Pres Tue		
UGRADS Symposium															Pres Fri		
Acceptance Test Demo															Schedule	Present	