Biology LabView to Python

Project Alpha Prototype Report

The Cousins Photosynthesis Lab in
The School of Biological Sciences at WSU

Data Pirates

Ritik Agarwal, Zoe Parker

11/28/2022

I. Introduction

Our team has been tasked with converting an old LabView software platform to a Python application. The old platform performs data acquisition, manipulation, and presentation on inputs received from instruments in Cousins Photosynthesis Lab in The School of Biological Sciences at Washington State University. The new application will replicate the current solutions ability to collect data and make calculations and manipulations, but will also enhance the efficiency and usability. The application will take the form of a PyQt Desktop Application supported by PyQtGraph and other Python libraries. The end goal of this project is to provide an updated Python application that is more user-friendly and efficient than the current software.

I.1 Client and Mentor Information

We have one formal mentor for this project, as well as at least five additional not listed stakeholders. Our official mentor is Asaph Cousins, who leads the Cousins Photosynthesis Lab. Our five additional stakeholders are the other lab members who will be using the application for their lab work. Asaph provides us with all the information about the project, including the desired requirements and timeline of the project. The other lab members will be key in providing us feedback and user acceptance testing.

I.2 Contact Information

I.3 Company and Client Background

As mentioned above, our client is the Cousins Photosynthesis lab at Washington State University. The research in the Cousins Photosynthesis Lab focuses on better understanding how photosynthesis in terrestrial plants has adapted to historical climatic conditions and how this phenotypic variation can be leveraged to enhance the photosynthetic efficiency of key food and biofuel crop species. The lab builds and develops instruments to study natural variation in kinetics of key photosynthetic enzymes that influence the uptake of $CO_{2,}$ and leaf anatomical traits that impact water and CO_2 movement within the leaf. They also use grasses, maize and other plants to conduct their research. The essence of their research is focused on understanding traits that influence photosynthetic efficiency in relationship to water loss via transpiration and to use plant science to help sustainably meet the human population's current and future food and energy requirements.

II. Team Members & Bios

Zoe Parker is a computer science student interested in software and application development. Zoe's is most skillful in Python, C#, C/C++, Winforms and WPF. For this project, her responsibilities include developing the data calculations and data manipulation subsystems and helping with other subsystems when needed.

Ritik Agarwal is majoring in Computer Science and interested in software development. Ritik is skilled in Python, C/C++, C#, DBMS and Docker. For this project, he is the team lead and responsible for designing and developing the core features requested by the client. His main focus is on making a multi-threaded application that satisfies user experience, availability, performance, scalability, adaptability, security, and economy.

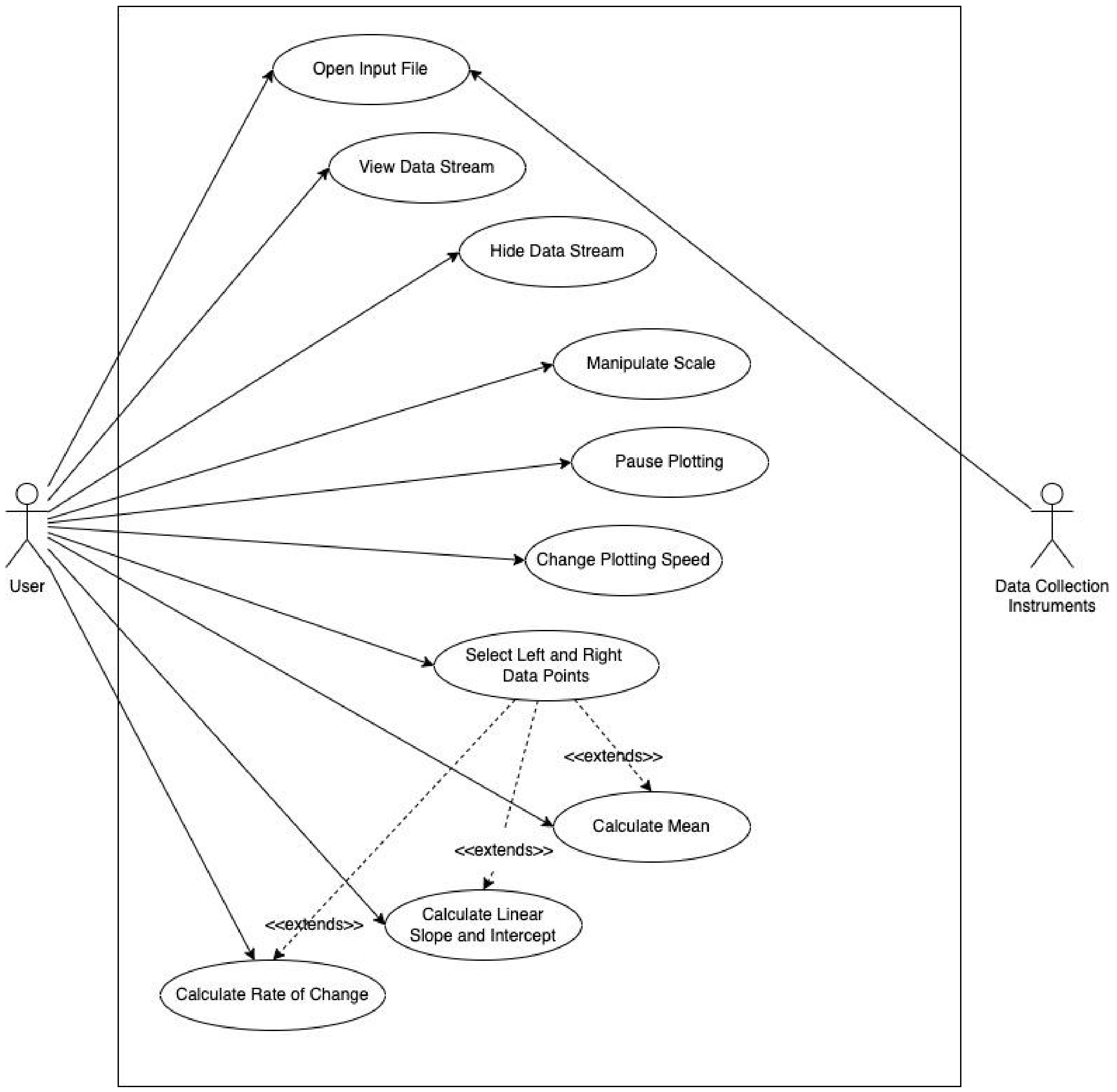
III. Project Requirements Specification

III.1. Project Stakeholders

Our first stakeholder is The Cousins Photosynthesis Lab in The School of Biological Sciences at WSU. They will use the new LabView software for conducting experiments and visualizing the data. The Cousins Photosynthesis Lab requires us to develop the old LabView Software into a new LabView Software written in Python. The researchers in the lab will get trained to use the software.

Our second stakeholder is our mentor from the class CPTS 421/423.The mentor will guide us along the project to implement the best solution. Mentor will also serve as the communicator between the Data Pirates and The Cousins Photosynthesis Lab. The mentor will also coach the agile development approach, and the project is developing without any anchors.

III.2. Use Cases



III.3. Functional Requirements

III.3.1. Data Acquisition

Acquire Data: This application must provide the ability to acquire data in the form of .csv files from a file system. If the program runs out of data or tries to read data before it has been created, it shouldn't crash.
Source: The client with Cousins Photosynthesis Lab originated this requirement. The requirement is necessary for users to use and see data.
Priority: Priority Level 0: Essential and required functionality

III.3.2. Data Plotting

Plot Inputted Data: This application must provide the ability to plot the data from the input files on a graph. This graph must be animated so data can be plotted in near-real time and this graph must plot multiple data streams at the same time.
Source: The client with Cousins Photosynthesis Lab originated this requirement. The requirement is necessary for users to use and see data.
Priority: Priority Level 0: Essential and required functionality

Ability to Select Data Streams: This application must provide the ability to select which streams of data from the input files to plot on the graph. There will be multiple streams of data in the input data, but not all are relevant, so the user must be able to choose which streams they want to plot.
Source: The client with Cousins Photosynthesis Lab originated this requirement. The requirement is desirable for users to be able to ignore unnecessary data.
Priority: Priority Level 1: Desirable functionality

Ability to Pause and Change Speed of Data Plotting: The application must provide the ability to pause the plotting animation at any time, as well as change the speed at which the animation plots the data.
Source: The client with Cousins Photosynthesis Lab originated this requirement. The requirement is necessary for users to use the application easily and efficiently.
Priority: Priority Level 0: Essential and required functionality

Plot Calculated Data: This application must provide the ability to plot data resulting from calculations performed on other data.
Source: The client with Cousins Photosynthesis Lab originated this requirement. The requirement is necessary for users to use and see calculated data.
Priority: Priority Level 0: Essential and required functionality

Mean Bars: Two movable mean bars should be included in the application. The mean of the plot between the two bars should be calculable using these two mean bars.
Source: Requirement by the stakeholder.
Priority: Priority Level 0: Essential and required functionality.

Ability to Manipulate Scale: This application must provide the ability to manipulate the scale of the graphs in the application. The user should be able to edit the lower and upper bound of both the vertical and horizontal axes.
Source: The client with Cousins Photosynthesis Lab originated this requirement. The requirement is desirable for users to better see and scale the data plots.
Priority: Priority Level 1: Desirable functionality

III.3.3. Data Calculations

Calculate Mean Values: This application must provide the ability to calculate the mean value between two data points on a graph.
Source: The client with Cousins Photosynthesis Lab originated this requirement. The requirement is necessary for users to manipulate and calibrate the data.
Priority: Priority Level 0: Essential and required functionality

Calculate Rates: This application must provide the ability to calculate the rate of change between two data points on a graph.
Source: The client with Cousins Photosynthesis Lab originated this requirement. The requirement is necessary for users to manipulate the data.
Priority: Priority Level 0: Essential and required functionality

Calculate Linear Equations: This application must provide the ability to calculate the slope and intercept of a linear plot.
Source: The client with Cousins Photosynthesis Lab originated this requirement. The requirement is necessary for users to derive results from the data.
Priority: Priority Level 0: Essential and required functionality

UI Utility Buttons and Bars: There should be several UI buttons and bars in the program for processing, manipulating, and graphing data. The software should function properly when these UI buttons are used.
Source: Requirement by the stakeholder.
Priority: Priority Level 0: Essential and required functionality.

III.4. Non-Functional Requirements

Easy To Use

This application should be straightforward and should not be too difficult to learn.

Self Containment

The application needs to be self-contained and simple to use. The user should have no trouble using the program after receiving training. For upcoming developments, the application should include the software documentation.

Great UI

The user should have a great UI experience when using the application. The UI should be user friendly.

Design For Change

The application has to be flexible. Future feature additions should be possible without significantly altering the present code, if necessary. Additionally, the code must adhere to accepted industry standards, be properly organized, and be simple to comprehend.

System Compatible

The hardware and software configuration in the lab at this time ought to work with the program. The application shouldn't be susceptible to both software and hardware modifications in the long run.

Accurate

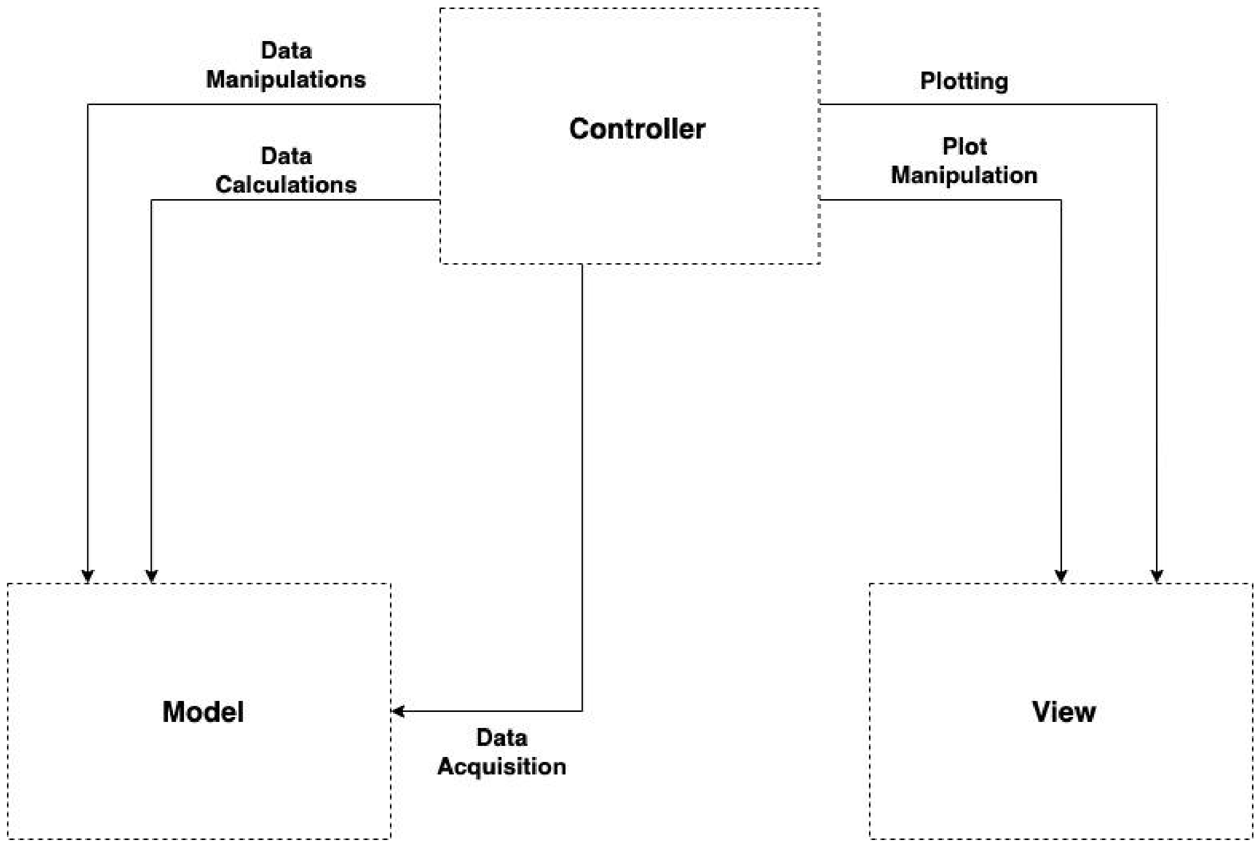
This application will deal with important data that needs to be correctly plotted and calculated on. The application should accurately plot and calculate data.

IV. Software Design

IV.1. Architecture Design

IV.1.1. Overview

The Data Pirates team has decided on a layered architectural pattern for this application. The team came to this conclusion with the justification that this application is meant to be a desktop application. For this application we need at least a front end presentation layer that the user will see and interact with and an application logic layer for core functionalities and calculations. We felt that a layered architecture was the best pattern to use in this context. Below is a diagram detailing a general view of the applications architecture.



IV.1.2. Subsystem Decomposition

IV.1.2.1 Plotting and Plot Manipulation

a) Description

The Plotting and Plot Manipulation subsystem is responsible for handling the visualization and manipulation of plots. The Plotting part of the subsystem will read data input from various data streams and plot it on an animated graph. The Plot Manipulation part of the subsystem will handle any manipulations to the data plot settings. This includes scaling the axes, pausing the plot animation, speeding up the plot animation, and selecting data points from the plot.

b) Concepts and Algorithms Generated

The Plotting subsystem will consist of 1 major Plot class. The class will handle all the matplotlib plotting functionalities for visualizing and manipulating the plots.

c) Interface Description

<u>Services Provided:</u>

1. Service Name: Plot
   Service provided to: Presentation Layer

Description: The Plot service will allow the current state of the data to be plotted. With animating a plot of data, each data point is plotted one at a time in quick succession like frames of a video. The Plot service allows the presentation layer to plot one of these frames.

2. Service Name: Animate
   Service provided to: Presentation Layer
   Description: The Animate service allows the presentation layer to plot the input data in real-time, instead of just plotting all the data at once.

3. Service Name: Scale Axis
   Service provided to: Presentation Layer
   Description: The Scale Axis service allows the user to scale and manipulate the axes of the plot. The bounds of the axes can be increased or decreased, stretching or compressing the axes.

4. Service Name: Pause Animation
   Service provided to: Presentation Layer
   Description: The Pause Animation service allows the user to pause the animation of the plot.

5. Service Name: Change Animation Speed
   Service provided to: Presentation Layer
   Description: The Change Animation Speed service allows the user to change the speed of the plot animation without crashing the program.

6. Service Name: Collect Data Points
   Service provided to: Logic Layer
   Description: The Collect Data Points service allows the user to select data points from the plot. The data points are then sent to the Logic Layer to make calculations.

Services Required:

1. Service Name: Send Data to DataCalculations
   Service Provided From: DataCalulations

IV.1.2.3 Data Manipulation

a) Description

The Data Manipulation subsystem is responsible for handling any manipulations to the input data. For example, the input data may contain a data stream that the user doesn't want to use, therefore, they should be able to manipulate the data so the data stream isn't included. Any alteration to a data set should be considered a data manipulation.

b) Concepts and Algorithms Generated

The Data Manipulation subsystem will include all of the tools needed for manipulating data. It will be implemented through a DataManipulation class. The class will consist of methods to change and alter data before and after it has been plotted.

c) Interface Description

Services Provided:

1. Service Name: Sort Input Data
   Service provided to: Data Layer
   Description: The Sort Input Data service allows the Data Layer to sort the input data that was read in from file into separate data streams.

Services Required:

1. Service Name: Receive Data from Data Acquisition
   Service Provided From: Data Acquisition

IV.1.2.4 Data Calculation

a) Description

The Data Calculation subsystem is responsible for handling any calculation that must be performed on data. This may include, but is not limited to, mean value of two data points, rate/slope between two data points, or linear intercepts. The calculations will be outputted as values, as well as new plot points, for the user to see.

b) Concepts and Algorithms Generated

The Data Calculation subsystem will include all the tools needed for performing calculations on data. It will be implemented through a DataCalculation class. Most of the needed operations are accessible through the standard Python library and Python Numpy library. The class will consist of methods that will utilize the functionalities of these libraries.

c) Interface Description

Services Provided:

1. Service Name: Calculate Mean
   Service provided to: Presentation Layer
   Description: The Calculate Mean service allows the logic layer to calculate the mean value from two points of data. This result is sent to the Presentation Layer to be printed and/or plotted.

2. Service Name: Calculate Rate
   Service provided to: Presentation Layer

Description: The Calculate Mean service allows the logic layer to calculate the rate of change value between two points of data. This result is sent to the Presentation Layer to be printed and/or plotted.

<u>Services Required:</u>

1. Service Name: Send Data to Plotting and Plot Manipulations
   Service Provided From: Plotting and Plot Manipulations

IV.1.2.5 Data Acquisition

a) Description

The Data Acquisition subsystem is responsible for acquiring data from the provided input data files. This includes reading in data from .csv files so it can be used for data manipulation and calculations. This consists of reading from a large number of .csv files.

b) Concepts and Algorithms Generated

The Data Acquisition subsystem will include all the tools needed to acquire the needed data from the input files. This will be implemented through a DataAcquisition class. This class will consist of methods to read and store data coming from the input files.

c) Interface Description

<u>Services Provided:</u>

1. Service Name: Acquire Input Data
   Service provided to: Data Layer
   Description: The Acquire Input Data service allows the Data Layer to read in input from .csv files.

<u>Services Required:</u>

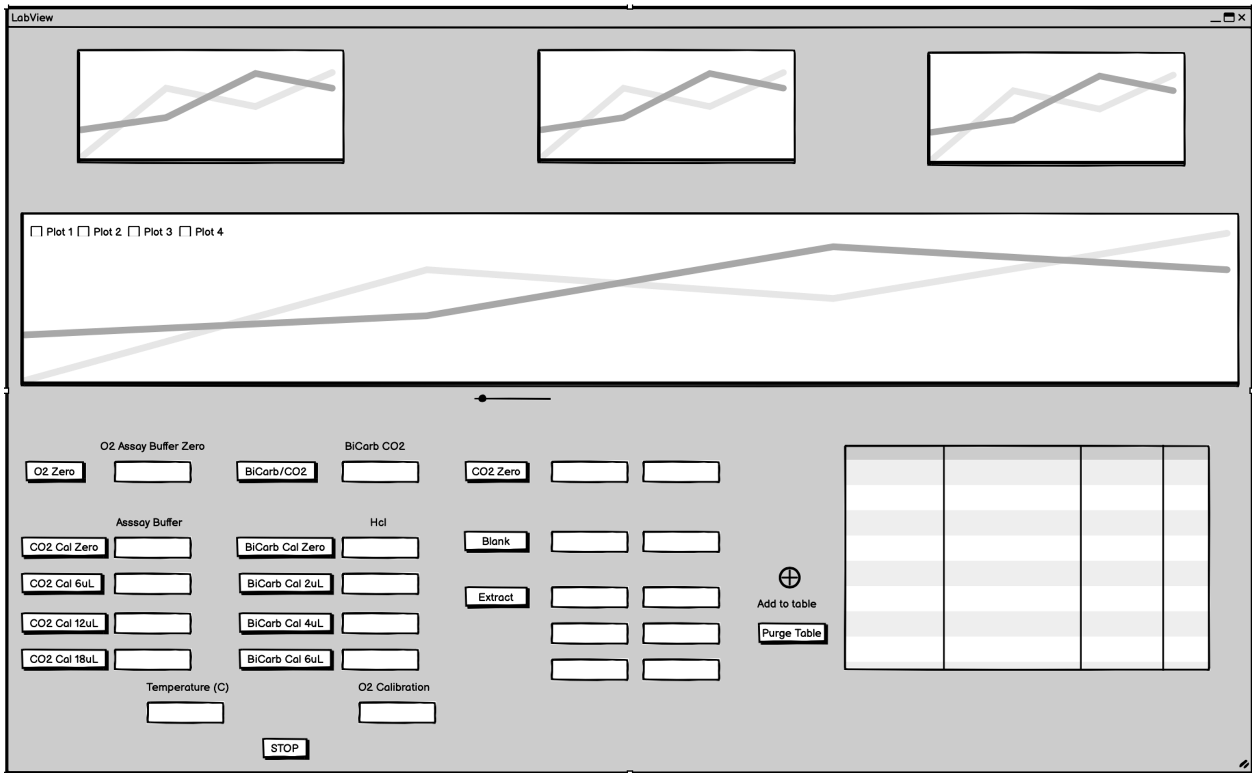
1. Service Name: Get Input Files
   Service Provided From: Presentation Layer

IV.2. Data Design

The data used in the application is stored on a local computer as .csv files and is read by the application for plotting, manipulation and calculations. Once the application is closed, neither the input data nor the calculated data needs to be stored anywhere. Thus, our application will only use an Application Temporary Data Structure. This data structure is a class that will be used to store and handle data while it is being actively used by our application. This data structure takes the form of a class called DataStream. The DataStream class contains all the plotting data for a single stream of data. The input files that application plots from have multiple streams of data to be plotted, so we need to separate and organize these streams. The application will temporarily store the input streams into their own DataStream class while plotting and calculations occur. Once the application is closed, these objects will be deleted.

IV.3. User Interface Design

The Data Pirates team has created a partial UI for the data acquisition and manipulation desktop application that is similar to the Cousin's Photosynthesis Lab's current LabView software. The UI will have an executable file which will download/update all the necessary packages required to run the application. The user will have to run the file before the application can be launched. After running the installation executable, the user should be able to run the application on the system.



V. Test Case Specifications and Results

V.1. Testing Overview

V.1.1 Test Objectives and Schedule

The Data Pirates team will use testing to gain confidence in the integrity and accuracy of our application. Our goal is to write tests that will help us find bugs and determine the greatest points of failure in our system. Testing will also help us address and improve our systems weak points.

In order to carry these objectives our team will make use of the Pytest testing framework and the testing and mocking functionalities in the Python standard library. We will utilize these frameworks for writing unit and integration tests. Our development pipeline will be created using GitHub’s CI tool.

For testing, we plan to have two main deliverables: Documentation of our tests and results, and our GitHub repository containing our tests and CI pipeline. We will follow somewhat of an agile testing strategy in which tests will be written concurrently with the code by the same developer that is writing the code. This process would look like: Developer writes code, same developer writes unit tests for that code, writes integration tests, integrates the code with CI, performs system testing, then lastly, performs demo and user acceptance testing.

V.1.2 Testing Strategy

1. Identify the requirements to be tested. All test cases shall be derived using the current Software Requirements Specification.

2. Identify which particular test(s) will be used to test each module.

3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.

4. Identify the expected results for each test.

5. Document the test case configuration, test data, and expected results.

6. Perform the test(s).

7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the revised Test Plan document.

8. Successful unit testing is required before the unit is eligible for component integration/system testing.

9. Unsuccessful testing requires a bug form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.

10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

V.1.3 Test Plans

V.1.3.1 Unit Testing

Our team will follow a standard approach for unit testing. Unit testing will require all non-trivial methods of a unit to have at least two tests. The first will test a valid use of the method, and the second will test an invalid use of the method. A method is trivial if it has an empty body or contains only trivial operations such as simply assigning a value to a variable, or adds two numbers. Additionally, depending on the relevance and complexity of a unit, we may develop additional tests for that component. A method is relevant if it is used a lot, or if broken, would result in a significant decrease in integrity in the application. A method is complex if it has many potential uses or many execution paths. Developers should communicate with the developer responsible for a unit if they feel the unit needs more testing.

V.1.3.2 Integration Testing

Given the multitude of operations possible in the application, integration testing is quite a bit more complex than unit testing. Our team will identify the major application operations or functionalities from the requirements specification from which we will develop integration call graphs and test classes. To test these operations, we will use top-down integration testing in Python. Since our team has familiarity with unit testing in Python, but not with integration testing in Python, we will be integrating on a per class basis, rather than per method in order to reduce complexity of writing tests. However, if an operation has a greater relevance or complexity, the team will consider more detailed testing.

V.1.3.3 System Testing

V.1.3.3.1 Functional Testing

For our functional testing plan, we will rely mostly on our Requirements Specification section of this document, which specifies the functional requirements of the project. Functional testing measures the quality of the functional components of the system from the user's viewpoint. Thus, our team will focus on testing the functional requirements that are most relevant to the user, such as acquiring data, plotting data, selecting data points, data calculations, etc. Our functional tests will consist of one functional requirement, the logical steps taken to carry out the function, and the result of the test. The functional requirement and steps will be formulated before the test is executed and will be documented. All the test documentation will be compiled in a single document that will be updated in the event of restructuring that changes how a function is carried out. This document will also contain a results section, which will record the event of a failed test and all possible information about why the test failed. For any errors or failed tests, the developer who found the error must create an issue in the team repository. The developer responsible for the error will be assigned the issue and must rectify the bug.

V.1.3.3.2 Performance Testing

For our performance testing plan, as in functional testing, we will rely mostly on our Requirements Specification section of this document. Specifically, we will create a test for each of the non-functional requirements in the section. However, these tests will not be as specific as functional testing, and may rely more on developer judgment. Our performance tests will consist of one non-functional requirement, the steps to be tested or the criteria to be satisfied, and the results of the test. Similar to functional testing, there will be a document detailing the tests and their content, however, there will also be more developer comments and judgements. If the developer who is testing feels the system fails to satisfy a requirement, they must determine what needs to be changed in order for the requirement to be satisfied.

V.1.3.3.3 User Acceptance Testing

For user acceptance testing, we plan to carry testing out multiple times (almost weekly) before the end of the project. Everytime we meet with our client, we will demo the project progress to the client. This demo serves as an opportunity for the client to evaluate the application. The developers will always come prepared with the list of requirements they are demoing, as well as a list of requirements yet to be implemented. These requirements will be based upon the Requirements Specification section of this document. With this list, the client will not have to worry about remembering every requirement, and can get a clear idea of the progress of the project. This will also help when discussing the requirements the client feels are most important to start implementing next. As the project progresses, these demos will become less of the developer showing the application, and more of the client using the application. One of the end goals of the project is for the client to have the ability to easily use and adapt the application. Thus, transitioning to the client using the application is very important. As with the other forms of testing, if a bug is found or a new feature is requested while user testing, one of the developers will create an issue in the team repository for it. The developer assigned to the issue will rectify the bug or create the new feature.

V.2. Environment Requirements

Our testing environment will require some outside tools. For unit testing, we will make use of the Pytest software testing framework. Integration testing will also make use of the Pytest framework. Pytest is a testing framework that is useful for small, readable tests, but can also scale to support complex functional and integration testing for applications [6]. No outside tools will be needed for mocking in the integration testing because the Python standard library has its own mock functionality. Our team will utilize GitHub Actions for Continuous Integration [7] to ensure our tests are run and pass before code is merged.

V.3. Test Results

// we’re still working on setting up our CI pipeline and have no testing results at this time

VI. Projects and Tools used

PyQt	A Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in. We used this to make the GUI for the application
PyQtGraph	A pure-python graphics and GUI library built on PyQt / PySide and numpy. It is intended for use in mathematics / scientific / engineering applications. We used this to make data plots in the application.
Qt Designer	The Qt tool for designing and building graphical user interfaces (GUIs) with Qt Widgets. We used this to create the ui mockup.

Languages Used
Python

VII. Description of Final Prototype

VII.1 Plotting

VII.1.1 PyQtGraph Plotting

VII.2 Data Calculations

VII.3 Front End User Interface

VIII. Product Delivery Status

This project will be delivered to our client at Cousins Photosynthesis Lab on Sunday April 30th before midnight. We plan to deploy the application on the client's machine at the beginning of April, and will use the time until April 30th to test, debug, and make any necessary changes to the application. In addition we will use the time in April to teach the members of the lab how to use and modify the application, as well as finish up this and any other required documents.

Included in our deliverables will be the following:

- Code Repository
- Readme/how-to docs
- Diagrams
- This document
- A list of Python resources and documentation

Although we will be installing Python and any necessary libraries–as well as testing the application ourselves–on the client's machine, we will provide an installation script and a readme on how to install packages and run the application. In addition, the how-to documents will describe how to operate the application, as well as how to change or modify the application to the users' needs.

IX. Conclusions and Future Work

IX.1. Limitations and Recommendations

IX.2. Future Work

X. Acknowledgements

We thank Asaph Cousins and the team at Cousins Photosynthesis Lab.

XI. Glossary

Python: A high-level, general-purpose programming language, often used to build websites and software, automate tasks, and conduct data analysis.

LabView: A graphical programming environment used by engineers to develop automated research, validation, and production test systems.

PyQt: A Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in.

PyQtGraph: a pure-python graphics and GUI library built on PyQt / PySide and numpy. It is intended for use in mathematics / scientific / engineering applications.

Numpy: A library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Continuous Integration (CI): a DevOps software development practice where developers regularly merge their code changes into a central repository

XII. References

1. Real Python, “Python and pyqt: Building a GUI desktop calculator,” *Real Python*, 23-Sep-2022. [Online]. Available: https://realpython.com/python-pyqt-gui-calculator/. [Accessed: 29-Oct-2022].
2. A. Cousins, “Biology-LabViewToPython.” .
3. “What is LabView? Graphical Programming for Test & Measurement,” *NI*. [Online]. Available: https://www.ni.com/en-us/shop/labview.html. [Accessed: 20-Sep-2022].
4. “Scientific Graphics and GUI library for Python,” *PyQtGraph*. [Online]. Available: https://www.pyqtgraph.org/. [Accessed: 29-Oct-2022].
5. *NumPy*. [Online]. Available: https://numpy.org/. [Accessed: 10-Oct-2022].
6. “Helps you write better programs¶,” *pytest*. [Online]. Available: https://docs.pytest.org/en/7.2.x/. [Accessed: 28-Oct-2022].
7. “About continuous integration,” *GitHub Docs*. [Online]. Available: https://docs.github.com/en/actions/automating-builds-and-tests/about-continuous-integration. [Accessed: 28-Oct-2022].

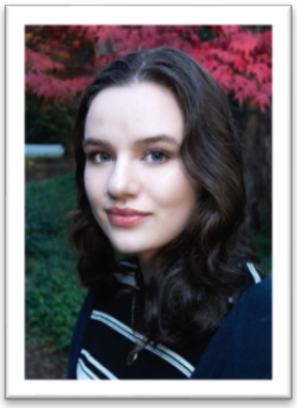
XIII. Appendix A - Team Information

Ritik Agarwal

Zoe Parker

Email:
ritik.agarwal@wsu.edu

Email:
zoe.parker@wsu.edu



XIV. Appendix B Example Testing Strategy Reporting

XV. Appendix C - Project Management

Our team's weekly schedule consisted of one meeting with our client. The team would meet weekly before the client meeting, and discuss after to plan for the week. The team also communicated regularly on Discord. Our client meetings served as a time to demo the progress we had made on the project that week, get feedback on our work, and discuss any issues we faced. A typical client meeting would consist of:

1. Demo current progress and issues
2. Questions about current work
3. Discussion of issues and solutions
4. Discussion of future work

Overall this model worked well for us. Weekly meetings kept the client up to date with our progress and allowed us to get a lot of feedback. Our meetings as a team before client meetings served as a check in for each other on our progress. Then our discussions after the client meetings served as sprint planning. We would discuss what issues we could finish and close, and what needed to be done during the next week.