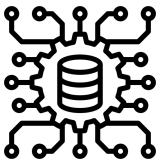
Data Manipulation Software Conversion

Prototype Report

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

Data Pirates



Ritik Agarwal, Zoe Parker

Table of Contents

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. Background and Related Work

Currently, Cousins Photosynthesis Lab is using an old LabView software that 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. Our project requires us to update the old LabView software to more efficient and user-friendly LabView software using python as the primary programming language.

Python is the best programming language for this project, as Python programming language is known for its versatility, readability, and extensible data manipulation and representation packages. The project requires an advanced understanding of Python programming language. We will be referring to the python documentation for any sorts of anchors and help. Since we are developing a desktop application - we need a GUI framework. We will be using the PyQt GUI toolkit [1]. Creating a GUI application using PyQt is an easy task due to the availability of extensive documentation. We will also use Test Driven Development to achieve fewer bugs and errors, and to produce a higher overall test coverage and, therefore, better quality of the final product.

Good software requires good coding practices and documentation. We will use Software Design Principles to develop the new LabView desktop application. We will also use UML for designing the software architecture. Our goal is to design a desktop application that is scalable, reliable, and extensible.

I.2. Project Overview

The Cousins Photosynthesis lab is focused on better understanding how photosynthesis in terrestrial plants has adapted to climate changes and how they can enhance the photosynthetic efficiency of key food and biofuel crop species. The lab builds and develops instruments to study key photosynthetic enzymes. One instrument in particular collects raw data (in the form of voltages) from different inputs at a high frequency (about every 2-3 seconds) [2]. The data is acquired in small snapshots and sent to a computer that packages it into .csv files.

The current program that accesses these files and manipulates and plots the data was created through LabView by a previous engineer. LabView is a graphical environment engineers use to develop automated research, validation, and production test systems [3]. LabView utilizes a drag and drop interface to create its programs. However, this interface has proved to be complex and hard to decipher, making it difficult for lab members to change and improve program features.

The Data Pirates team, under guidance of our team mentor and the Cousins Photosynthesis Lab, will address the shortcomings of the current LabView software by designing, building, and delivering a Python application that performs data acquisition, calculation and manipulation for the lab team. The application will replicate the current solution, but will also enhance the current functionalities and be easier to use and update.

The Data Pirates team will achieve this by first decoding and understanding the current software so it can be recreated and updated in Python. Some features that the new program should support are the ability to read input files, plot the input data in near real time, and extract key data points and calculated parameters at specific timepoints throughout data acquisition. These features are present in the current solution, but will be enhanced or improved in the new program as needed.

The Cousins Lab conducts experiments that can run anywhere from 3 to 8 hours per day, generating a lot of data, some of which is not needed. The new program will also have the ability to exclude or hide data plots that aren't needed.

The current LabView software is complex and hard to read. So, another feature for the new program is the ability for the lab team to update the program without needing much coding experience. The lab should be able to change or add features as needed on their own.

The Cousins Lab requires a Windows-like interface for the program. Thus, our team will be creating a Python desktop application. At the end of the project, our team will train the lab members on how to use the application and update the code.

In terms of tools to be used for the new desktop application, Tkinter will be used to build the GUI. Tkinter will allow us to add text, buttons, and input entries for the application. Additionally, a few Python libraries will be needed for data presentation and manipulation. The Python graphics library PyQtGraph will be used to create interactive data visualizations [4], and the numpy library will be used for data processing and calculations [5].

In order to successfully achieve these goals outlined in this overview, the team will frequently communicate and meet with The Cousins Photosynthesis Lab for project evaluations and design reviews.

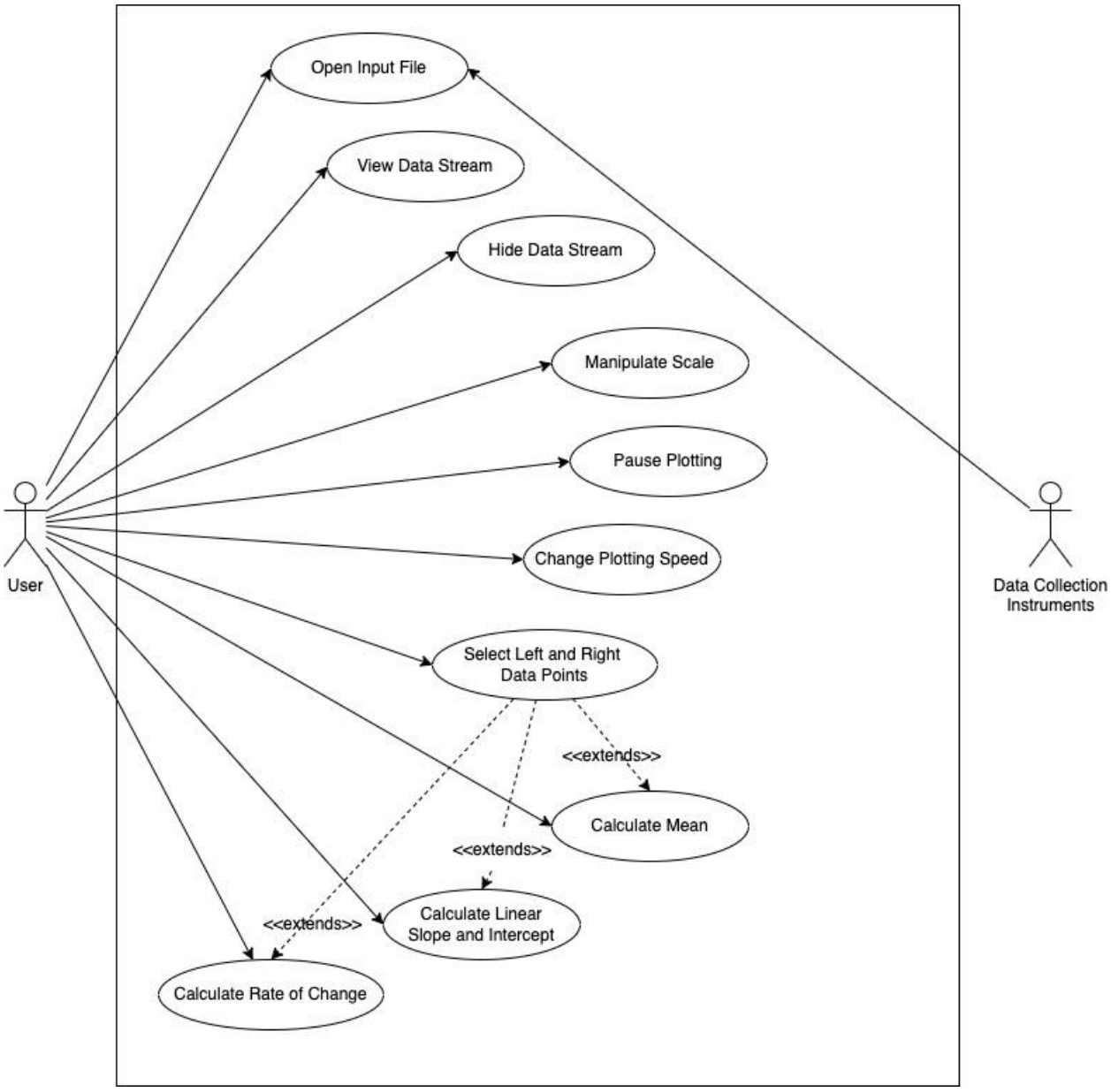
I.3. Client and Stakeholder Identification and Preferences

Our client 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.

II. System Requirements Specification

II.1. Use Cases



II.2. Functional Requirements

II.2.1. 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

II.2.2. 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

II.2.3. 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.

II.3. 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.

III. System Evolution

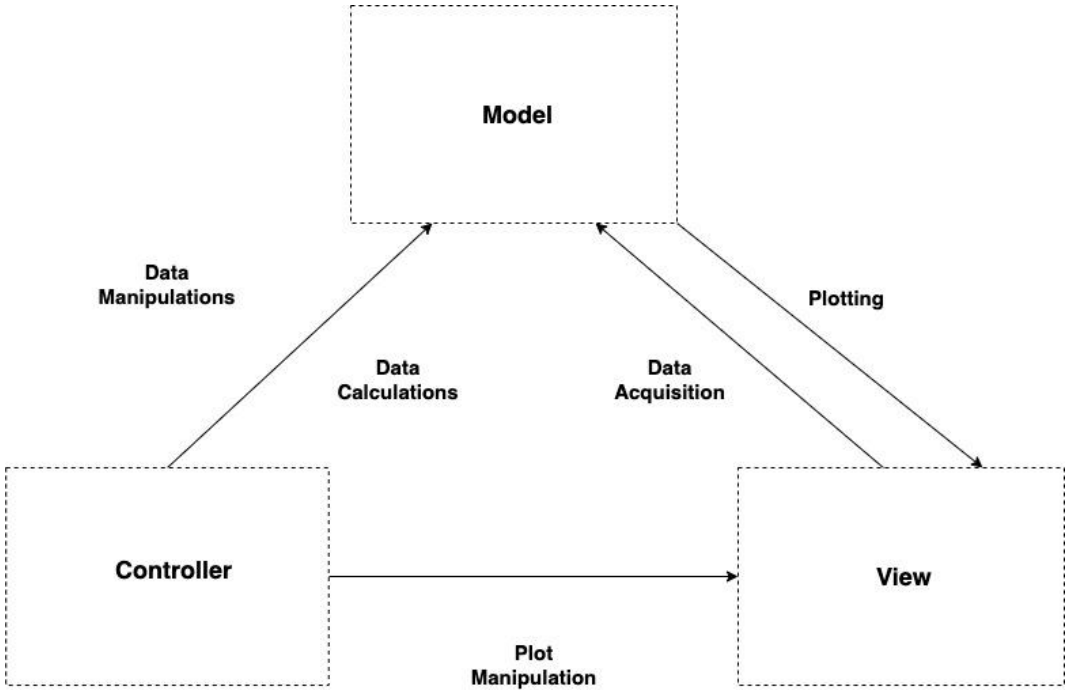
The Data Pirates team is creating the software while taking into account any modifications to both the hardware and the software. The application's compatibility with the lab's present software and hardware setups is the current objective. However, the environment in which the program will be used must have some software defaults. Hardware modifications will not have an impact on these requirements. The program will make use of Python 3.10.7, the most recent version. Python 3.6.0 must be installed at the very least on the system. Additionally, the system needs the installed versions of Tkinter, Matplotlib, and any additional programs required for development. An executable that installs the required software and drivers for the program will be made available. The team will also provide documentation which will include all the steps necessary to make the application work in the system. The team will also offer documentation that details every step required to implement the application in the system.

IV. System Overview

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

Services Provided:

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.

Services Required:

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

Services Provided:

2. 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.

Services Required:

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

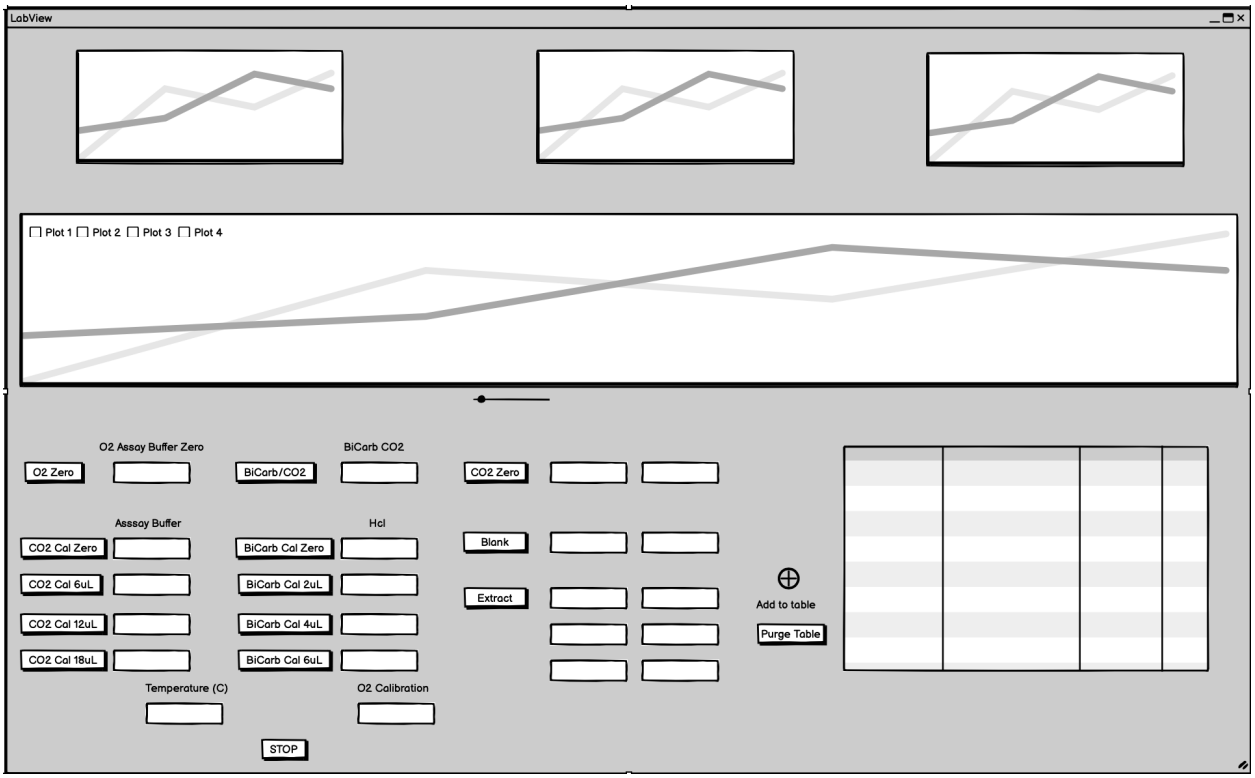
IV.2. Data Design

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.



There are several UI components in the desktop application. The screen's graphs ought to be interactive for the user. The user may compute the necessary numbers and then plot them on the graph using a variety of buttons. The interactive graphs can be used to do calculations. The pace of the plot can be adjusted using the slider beneath the graph.

V. 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.

Tkinter: An open source, portable graphical user interface (GUI) library designed for use in Python scripts

Matplotlib: A plotting library for the Python programming language. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits.

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.

V. Test Plan

V.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.2 Scope

This document discusses our plans for testing the Data Manipulation software. The scope of this document covers our plans for broad testing strategies and guidelines, as well as what resources and frameworks we will be using to test our code. This document will not cover the specific tests we will write or how they will be structured.

V.3 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.4 Test Plans

V.4.1 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.4.2 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.4.3 System Testing

V.4.3.1 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.4.3.2 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.4.3.3 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.4.4 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.

VI. 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].