Microbiology Lab Information Management and Visualization System

DESIGN DOCUMENT

Team 29

Client: Karrie Daniels
Adviser: Thomas Daniels

Brittany McPeek: Report/Documentation Manager
Benjamin Vogel: Team Manager
Rob Reinhard: Meeting Facilitator
Kyle Gansen: Meeting Scribe
Ben Alexander: Progress Manager
Samuel Jungman: Chief Engineer

Team Email: sddec20-29@iastate.edu
Team Website: https:/sddec20-29.sd.ece.iastate.edu

Revised: 4/26/2020
Version 3

Executive Summary

Development Standards & Practices Used

- Detailed Use Case diagrams and User Stories
- Simple Design
- Pair Programming
- Heavy documentation in both code and manuals, updated regularly
- Single coding standard (Will use PEP-8)
- Frequent refactoring
- Simple APIs
- Don't write code you might need in the future, but don't need yet
- Construct and maintain a user manual for non-technical clients

Summary of Requirements

- System should support importation of large amounts of data in a reasonable amount of time from sources such as CSV and Excel
- System should be able to abstract the data and place them into modifiable graphs
- System should support exporting those graphs to be published into research papers
- System should be able to maintain past data and support modifications and the addition of new data
- System will be written in Python
- System should be easy to understand and use by users with little-to-no background in programming
- System should be maintainable by 1 to 2 IT personnel

Applicable Courses from Iowa State University Curriculum

- COMS 309
- COMS 227
- COMS 228
- COMS 363

New Skills/Knowledge acquired that was not taught in courses

- Graphing and graph visualization
- Python
- Data management/backup

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

Our group would like to acknowledge and thank Thomas Daniels for his guidance, support, and technical advice throughout this project. Our group would also like to acknowledge and thank Karrie Daniels for providing us information and requirements for this project, as well as acting as a sample user of our end product. We appreciate the time and commitment these two individuals have donated towards this project.

1.2 Problem and Project Statement

Many scientists and researchers dedicate large amounts of time towards organizing, maintaining, and visualizing the data they collect. The purpose of this project is to find a solution to this problem. The solution should be able to automate the process of organizing, maintaining, and visualizing data. It is important that scientists and researchers have more time to collect and analyze their data, especially in time-sensitive experiments; thus resolving the issue of organizing, maintaining, and visualizing their data will be beneficial to scientists and researchers.

Our group proposes creating an application that allows the user to import pre-existing data from Excel and manages and visualizes the data. The application will allow users to select data elements and a type of graph/statistical analysis and visualize the resulting information in the form of a graph or another type of visual. The graphing utilities will allow the user to customize the appearance of the graphs to be created and will meet scientific publication standards. Additionally, the application will save backups of the data and allow the data and visuals to be exported and shared with another person. Our hope is to create an easy-to-use application that does not require too much maintenance and allows scientists and researchers an easier method to organize, maintain, and visualize their data.

1.3 Operational Environment

Our end product will be only software based. The software's environment is any computer that it runs on. Our end users will most likely be in a climate-controlled, indoor location which means that a computer will be able to operate without any problems.

1.4 Requirements

Our client wants to be able to use this software to log and manage data about microbiology experiments. The following is a list of functional requirements that the client wants:

- Data import from Excel
- Data modification after import
- Custom data visualization based on specified data elements
- Data sets and graphs should be saved to the file system

- Data should be backed-up in a separate location
- Function to export the data and be shared with someone else
- Creation of projects that contain multiple experiments
    - Collation of multiple graphs from similarly based experiments

Our non-functional requirements are:

- Possible for the system to be maintained by one or two people
- Secure enough so that research data can't be seen by anyone else
- Use libraries in Python for data visualization
- Data must be parsed after it is imported

1.5 Intended Users and Uses

This project's end product is intended to be used by scientists and researchers for inputting, organizing, and visualizing large amounts of data efficiently and effectively. The visualizations created by the product should meet scientific publication standards so that these visualizations can be used in published reports, papers, etc. Non-technical persons should be able to utilize the end product with ease. Many of these users are not particularly 'tech-savvy' and are not accustomed to complicated and unintuitive software. This experience places high importance in the intuition and cleanliness of the UI of our software. Furthermore, this software will not have a real maintainer or IT team after it is delivered, so the project will need to be clean and bug free to match the lack of maintenance staff.

1.6 Assumptions and Limitations

Assumptions:

- The maximum number of users per instance of the product will be one
    - The client prefers the data to be modifiable by one person, with additions of simultaneous editors as a stretch goal. Given the size and complexity of the data, having simultaneous users modifying and graphing data from one set could prove to be a daunting task.
- The solution will not be distributed outside of Iowa State
    - Due to the specific solution the end product brings to the research department, the use of the product (at least in its early stages of development) will not be useful outside of the university.
- Python and the Python Interpreter will be used for the development
    - By the request of the client, using Python with the product should provide easy maintenance if needed by people who may be unfamiliar with the development of

the product itself. This added with powerful graphing utilities can provide a solid solution to the client's problem

- The end user will require an instruction set about the end product
    - Due to the technical knowledge of the end users, a simplified instruction manual will be required to help convey the usages and information needed to operate the solution.

Limitations:

- The end product shall be able to be maintained by 1-2 IT workers on a minimal basis (Client Requirement)
- The end product shall cost the end user no more than $200 (less than cost of current client's solution)
- The end product shall be easy to run and navigate with little-to-no programming experience (Client Requirement)
- The end product shall be based on, and released for, the ISU research department (Geographical Constraint)

1.7 Expected End Product and Deliverables

The final product will parse excel files and output graphs based on the data provided. The end-user will also be able to customize these graph layouts as well as select the variables in the graphs so the project's use extends outside of the provided data-set. Finally, the end product's components will be decoupled to allow further customization if the needs of the end-product evolve following delivery (i.e. new types of data or graphs are needed).

The design of the project will be completed by the end of April, and the end product and documentation will be finalized by the end of December 2020. Documentation code will be ongoing as we meet each date. A look below shows a list of our deliverables over the course of the next two semesters.

Table 1: Projected deadlines for the project

ID	Date	Deliverable
D1	April 30th	Framework for each component of our project will have been created.
D2	September 30th	Prototype of each component of our project.

D3	October 15th	Functional Product
D4	November 15th	Final Product
D5	November 30th	Testing
D6	Ongoing	Documentation for above deliverables

Table 2: A description of the deliverables for the project

ID	**Description**
D1	By this point, "Hello World" programs will be built on the technological stack we have chosen. In addition, the preliminary frameworks for importation, graphing, exportation, and the GUI binding them together will have been created.
D2	Essential features of each component will have been implemented. Data can be imported from an Excel file. Then, you will be able to select data to create at least one type of graph. The graph will generate, and then you will be able to adjust the graph if needed. Finally, you will be able to export a high-quality image of the graph. In addition, data will be saved.
D3	Each of the core features will be fully implemented. Every type of graph requested will be available for creation, and the importation and exportation settings will be available. More types of files will be available for importation and exportation. In addition, backup snapshots will be implemented in case data is lost. Finally, the product will be able to calculate correlation and perform T-Tests and P-Tests from the graphs.
D4	Modifications requested by our clients looking at the functional product will be taken into account, and stretch-goals, if possible, will be implemented. Bugs that occur in normal boundaries will be squashed.
D5	While testing will occur as we develop the product to ensure proper functionality in normal circumstances, these two weeks the testing will be done in attempts to "break the product." This

	way, our final product we created for the client will continue to properly function even in scenarios we didn't account for.
D6	Two types of documentation will be provided: one for the end-user, and one documenting the code behind our end-product. The end-user documentation will consist of tutorials for the end-product. The code documentation will provide a detailed look at all classes and functions making up the back-end. The documentation will also outline how these classes and functions relate to each other. This will be demonstrated by providing "under-the-hood" looks at the tutorials in the end-user documentation. Finally, the documentation will show how classes and functions can be extended.

2. Specifications and Analysis

2.1 Proposed Approach

Our proposed approach is to create a local application using Python. The PEP-8 style guide for Python Code will be used during development. The user will interact with the program through a GUI made using the PyQt5 Python library. The various functions of the application will then be split into components which interact with each other through interfaces. This should allow for independent development of the various components of the software. A figure showing how these components relate to each other is available in section 2.4.

The design addresses the functional and non-functional requirements from section 1.4 in the following ways:

Functional Requirements:

- Data import and parsing from Excel will be handled through a data import component of the software. Python has a module "pandas" which can be used to extract data from Excel spreadsheets.
- Data modification after import will be handled through a data holding component of the software. Python also has a module "xlutils" which can be used to modify an existing Excel sheet.
- Custom data visualization of the data will be handled through data visualization and graphing components of the software. The open source Python library "Plotly" is one potential library to use for these components.
- The graphs and data will be saved through a save files component of the software. Plotly allows static images of plots to be saved, and the data can be saved to new Excel files using the "xlwt" Python module.
- Data will be backed up to a separate location through a data backup component of the software. After data has been imported into the software, it can be written to a file in a separate location.
- A share files component of the software will allow for files created by the application to be uploaded to Google drive.

Non-Functional Requirements:

- To ensure the system could be maintained by one or two people, a local application approach was chosen. Excluding a database from the design of the software eliminates the need for database maintenance.
- The research data will be secure due to the software being a local application.
- It was requested that Python libraries be used to visualize the data, and the Python library "Plotly" is robust enough to cover the functional requirements.
- The various Excel modules mentioned in the functional requirements allow for parsing the data after it has been imported.

2.2 Design Analysis

One of the strengths of the design in theory is the modularity. Breaking the software into separate components should allow for team members to work fairly independently of each other on the software. Another strength of the design is that it allows for new functionality to be added to the software without having to overhaul the design. New functions can be implemented as new components in the software.

One potential weakness is that all processing of the data will be done locally. The speed at which this occurs will depend on the processing power of the computer running the application, whereas a client-server application could allow for processing of data on the server side.

2.3 Development Process

The development process will be an agile based development. We will use 1-2 week iterations to produce predefined goals. These goals will be written out in the form of story cards that will encapsulate use cases with diagrams to allow for easy to follow documentation and maintenance after the project is delivered. We will use a Kanban board of some kind to track the progress of story cards. The reasoning behind this process is that it will allow us constant feedback from our client as to the specifications and implementations of the project. This is a project that will require input as the design and intuition of the UI has high importance in the overall quality of the final deliverable.

2.4 Conceptual Sketch

For each element in the diagram we would have an interface that abstracts each component and thus, simplify how the elements interact. The benefits from using interfaces here is better collaboration. Each team member doesn't have to know everything about another component in order to use it, they can just use the interface.

Figure 1: Conceptual sketch of the application and interactions between modules

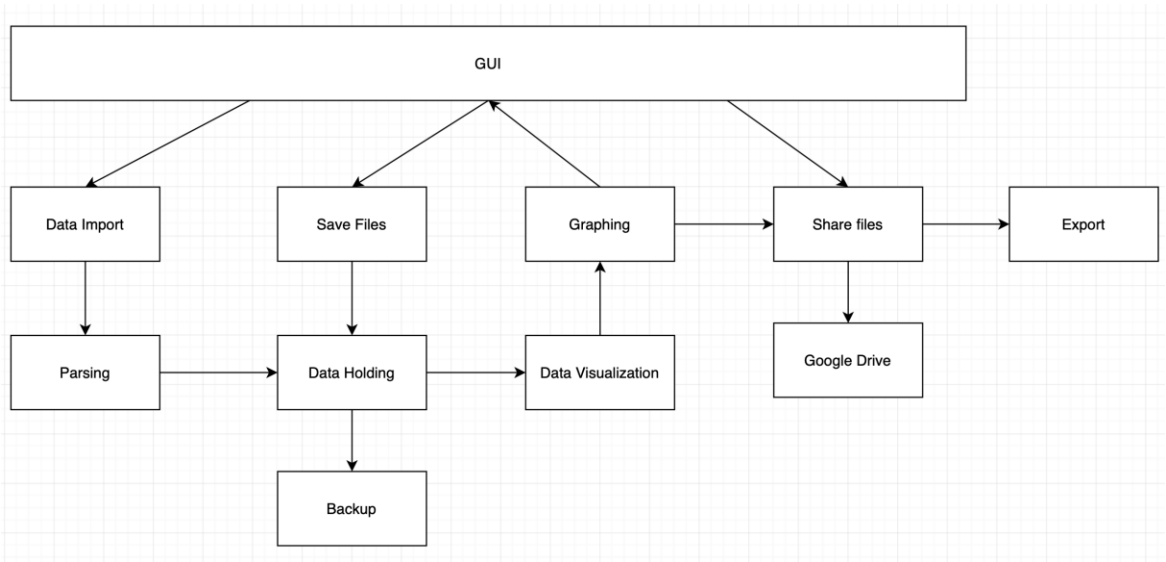
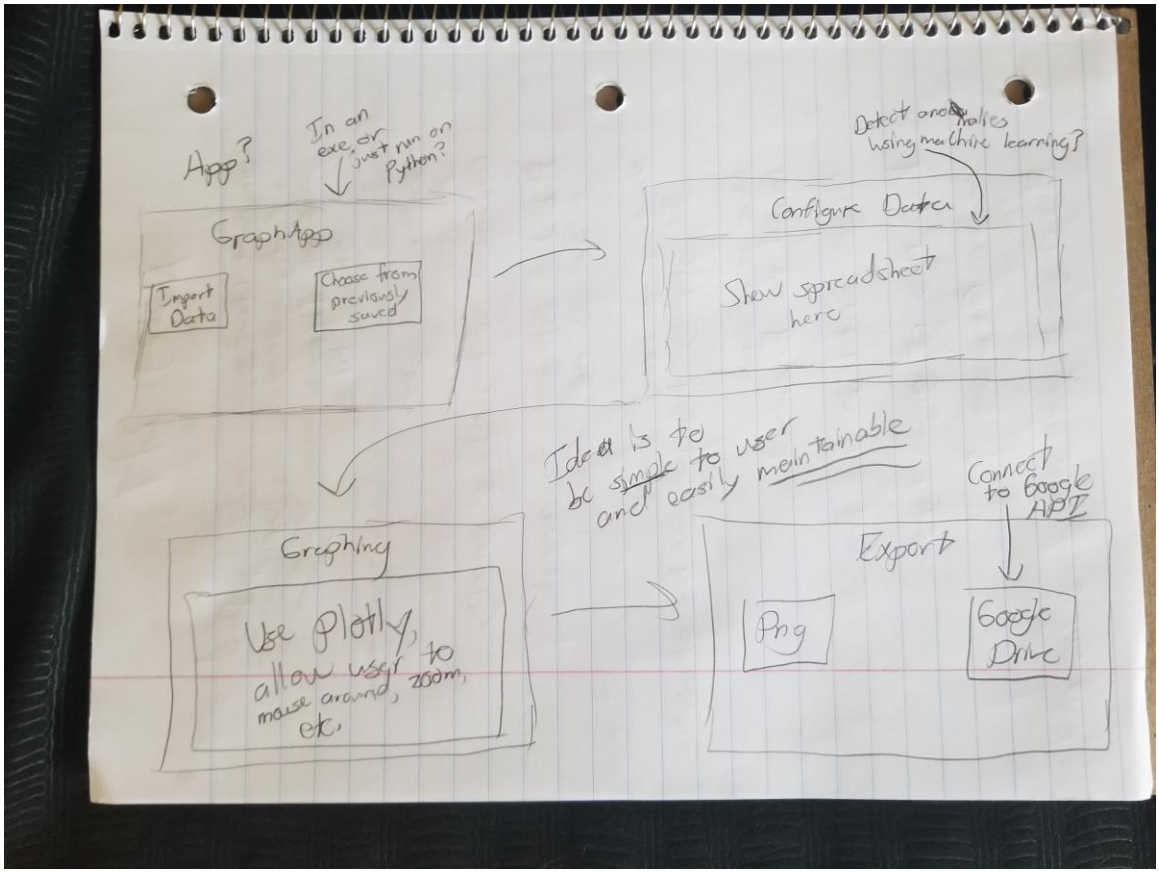


Figure 2: Conceptual sketch of the user interaction of the UI



3. Statement of Work

3.1 Previous Work And Literature

The previous work we are basing ourselves off of is the solution our client is currently using, GraphPad. This technology is designed specifically to take data and organize it similarly to a spreadsheet and then provide graphing utilities to help visualize the data. GraphPad itself does not operate data entry as a spreadsheet, but a more specialized version where they offer special data tables that can be catered to how the client wishes to organize.

One of the current major downsides to this technology is the price. GraphPad can be extremely expensive on a yearly subscription, especially when more than one person needs to have a license for it [1]. The other downside that the client has told us is the lack of options for a robust suite of graphs. Currently, GraphPad only can visualize with bar graphs, which the client may use, but would rather work with more varied graphs such as scatter dot plot graphs and correlation graphs.

3.2 Technology Considerations

The possible solution we are bringing to the table would provide a free variant to GraphPad. It would handle data in a similar way since that was valued by the client, but also utilize the power of Plotly (a Python library specialized with advanced graphing utilities) to provide a wider variety of graphs to use.

One strength of this solution would be the ability to use a free, open-source library on top of a free, open-source language and interpreter, which will be able to drive costs down on the development, which can transfer over to the user. Python has a massive amount of packages that can be easily installed and incorporated into our project to help us not only provide a robust product for the client, but relieve a lot of the behind the scenes work for the developers (for example, importing and parsing large amounts of data). Another strength of this solution is due to the nature of Python and our structure, we will be able to make an end product that should be much easier to maintain and fix for people who are not as experienced in software development. By making the front end and the back end as accessible as possible for our end clients, we can ensure that the product will be able to live beyond the Senior Design class.

A downside to this possible solution is the nature of Python itself. Because Python runs on an interpreter, creating an EXE or application will be harder. There are packages and libraries to help with the process, but it just doesn't work as naturally as some other languages or solutions might. Another downside is the current solution is a local solution. The client, while making it a low-priority stretch goal, wanted to have simultaneous users. By keeping, modifying, and visualizing the data on a local machine, it will make a solution with simultaneous users difficult if not impossible.

3.3 Task Decomposition

Our tasks can be decomposed into the modules similar to the structure of our solution in Figure 1:

1. Graphical User Interface
    a. Styling

i. Create a visually appealing front end that also shows all relevant data

ii. User should be able to edit styles to their own liking

b. Layout

i. Layout should be easily understandable by the end user

ii. Graphing should be the primary focus of the layout

2. Data Import and Parsing

a. The client should be able to import CSV or EXCEL files to be analyzed and graphed

b. The solution should be able to parse data from the files and sort them into data structures for better visualization options

3. Data Visualization/Graphing

a. Create a system that integrates with Plotly to create and show graphs to the end user

b. The system should be able to handle the following types of graphs and computations: dots plots, scatter plots, bar graphs, t-test plots, p-value computation, and correlation computation

c. The graphs should be customizable in regards to the types of graphs, which variables are graphed, what color schemes are used, and what data point shapes are used

4. Saving Files

a. The user should be able to save a current file within the application

b. The system should store backups of the imported data in a hidden folder for data recovery

5. Sharing Files

a. Create an export tool that can either share to Google Docs through an external API or save it as a picture to their local machine.

3.4 Possible Risks And Risk Management

Unfamiliarity with both Python and some of the needed packages will become a possible risk for the group, though we are already taking steps to remedy the problem with research and practice before beginning on developing the actual solution.

In order to avoid most of the risks that come from developing this solution, we will be communicating with one other and the client frequently in order to sort out misunderstandings

and confusion as fast as possible. By doing so, we are able to mitigate risks faster and more effectively as they come up.

3.5 Project Proposed Milestones and Evaluation Criteria

The key milestones in our proposed project are the framework, prototype, the functional product, and final product stages.

The framework stage is a key milestone because it will require us to build a theoretical model to implement. Since each of us are focusing on individual components of the project, the frameworks section is also critical because it will require us to have discussions on how data is passed between each component. The test for our frameworks are not programmatic, the test is that the frameworks we have built handle a consistent type of data and could work together.

The prototype stage is a key milestone because it will be the first time a product is delivered. By implementing core-features with the frameworks laid out in the previous stage, we will have our first demonstration of the product's components not only working, but working together. Testing will be done by passing in data through each component in a few key scenarios.

The functional product stage is a key milestone because it will be an implementation of all features we have been asked to deliver. Testing for these phases are done similar to the prototype stage. Using the same data, we test the features of that component, and then test the passing of the data to other components.

Finally, the final product stage is a key milestone because it will be a final tailoring of what we presented in the functional product stage for our client. Feedback will be taken into consideration, and stretch goals, if possible will be met. Testing is now for small adjustments, and similar to the functional product stage.

3.6 Project Tracking Procedures

Our group is currently using Trello to manage and track progress for this and next semester. We also meet with the client regularly to discuss our progress and gain feedback so we know what portions of the project need correction and to narrow our next steps.

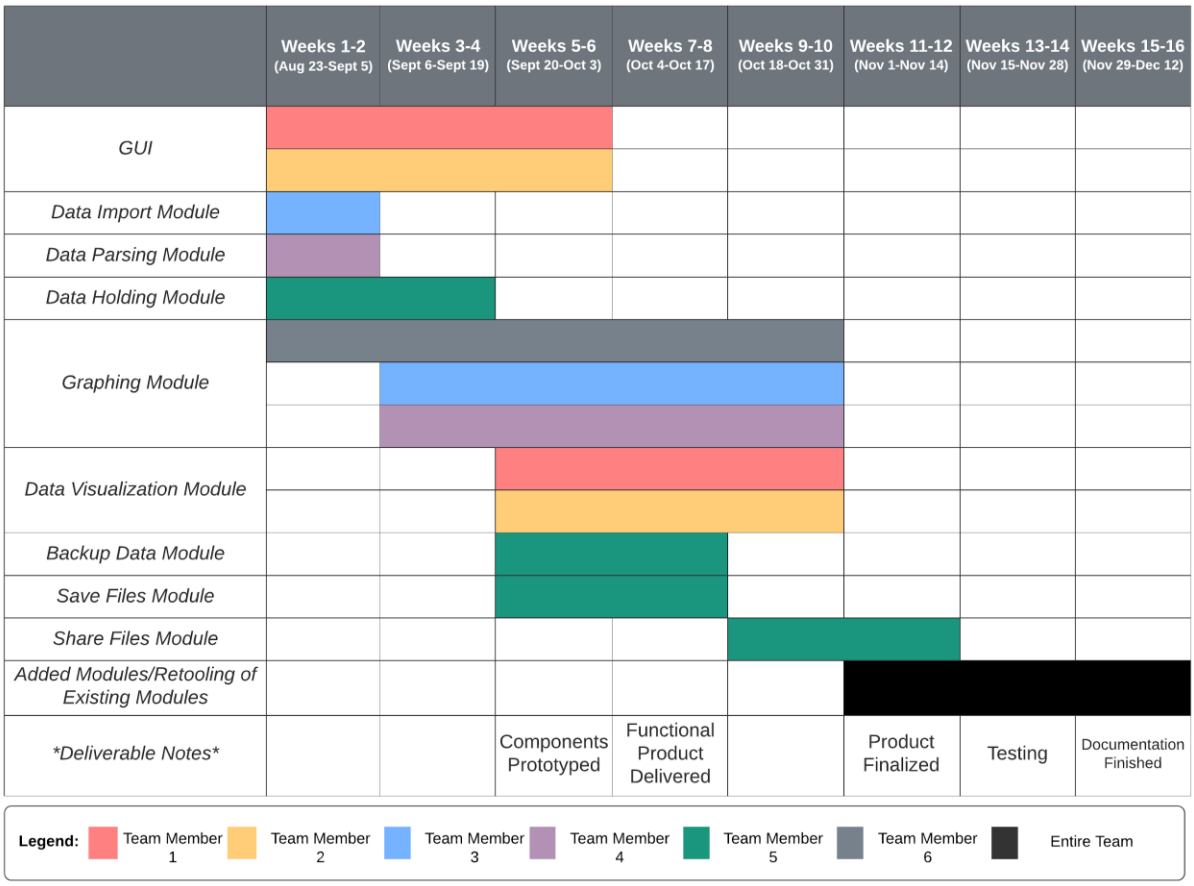
3.7 Expected Results and Validation

The desired outcome is a program that encompasses all of the tasks layed out in 3.3. Each of these tasks must behave properly within themselves as well as interact properly with one another. This can be verified at a **high level** using full integration tests in which individuals separate from our team will be asked to perform various tasks in a black box testing fashion that incorporate all of our key milestones and features that have been laid out in this document. It should be noted that this testing will occur only after rigorous Moq testing and unit testing on a low level as well as small scale integration testing at various points of development.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

Figure 3: Project Timeline



Work on the project on this proposed timeline begins with the GUI, data import module, data parsing module, data holding module, and graph module. These components set the groundwork for the project. For the future modules to be functional in the scope of the program, it is necessary that these are finished first. The GUI is vital for this project as it is the only way for a user to interact with the modules, so two team members are set aside for work on it for the first six weeks of the fall.

After significant progress has been made on the GUI and data can be imported into the program, the majority of the team is scheduled to work on the graph and data visualization modules. Because these are the key features of the programs, the graphing and data visualization modules will have five of the team's six members dedicated to working on them. After the product has been finalized, rigorous testing will be performed on the product. Testing will have already occured throughout development, but this portion of the testing will be done to ensure the finished product is airtight.

4.2 Feasibility Assessment

The project will be a standalone executable with its own files kept in the program location. It will encapsulate and allow for all of the main features previously discussed in this document. However, there are some foreseen challenges with this predicted project that have been laid out in the following paragraph as well as some preliminary plans to relive these challenges and improve the feasibility of this project.

The first main challenge will be the formatting and creation of the graphs themselves. Graphs will require some statistical knowledge and practice to be able to generate the graphs from the data. There will need to be some team research to accomplish this aspect of the project. The second is the variety of Python libraries that will be needed to complete this project. Due to the immensity of Python and its libraries, our group has varying experience with Python and certain libraries and will need to conduct experiments and research to learn how to develop the project using these libraries. These present themselves as the main foreseen challenges of our project which is why this project seems to be well within the realm of feasible.

4.3 Personnel Effort Requirements

Table 3: Personal Effort Requirements

Task	**Description**	**Time Estimate**	**Difficulty Estimate** (1 = Easy, 5 = very hard)
GUI Styling	Create a desktop application that is aesthetically appealing and intuitive.	15 hours over 2 weeks	2
GUI Layout	The layout should be easy to navigate and the app functions easy to understand	15 hours over 2 weeks	2
Data Import	Import data into the app from an excel or csv file	10 hours over 2 weeks	1
Parsing	Analyze the imported data to find and sort relevant variables and measurements	25 hours over 3 weeks	3

Data Holding	The parsed data should be temporarily kept in the program to be used later	5 hours over 1 week	2
Saving Files	Parsed data will be saved to the local storage in a csv file	10 hours over 2 weeks	1
Backup	Versions of program files will be backed up in the file system once every set time interval	15 hours over 2 weeks	3
Data Visualization	Use Plotly to create various kinds of graphs. User can customize data inputs, colors and shapes of data points	40 hours over 4-5 weeks	4
Sharing Files	Export files to Google Drive through API to share with other users	20 hours over 2 weeks	2

4.4 Other Resource Requirements

This project will be completed entirely using Python and Python libraries such as Plotly. This means that the project will not require any physical materials or equipment, other than the computers we use to complete the project. The end product, an application, should not take up much space, and will mostly likely end up around 10MB. Furthermore, the application will not require a very powerful computer to run. This means no additional equipment will be required for our client to use the product since the client has computers available to them in their lab.

4.5 Financial Requirements

This project will only require the use of Python and some Python libraries. A Python install is completely free, and the libraries we will use, such as Plotly, are free and open-source. We also plan on using our own PCs to complete the project, so there is no additional cost in that regard. The client will not need to buy any special equipment to use our end product, as their computers in their lab will suffice. Therefore our project will not require any financial resources.

5. Testing and Implementation

5.1 Interface Specifications

Our project is primarily software in nature with no real hardware to interface with. However, there is some software interfacing that will need to be tested, both from our own creation and from the libraries we opted to utilize for this project.

The first software library we will be interfacing with is Plotly, a graphing based module which will be the engine of our graph generation. We will need to do rigorous input and data manipulation testing with the module to make sure that it can handle the large and complex amounts of data that will be inputted to the application at one time, as well as provide enough options and flexibility to let the user define their own constraints on the graph being produced.

The other software library we will be interfacing with is pandas, a data analysis and manipulation tool for Python, and will be our primary module behind data importation. This interface should be able to handle wrong files, large files, different types of files (as discussed above and in Lightning Talk 3), while also providing a unified and correct output so as to keep the logic between importing data and manipulating/graphing the data as simple as possible.

We will also be utilizing the Google Drive API to export graphing, which will require us to test not only the API, but the module we will be writing to connect the user to the API, which includes providing boundary and exception testing within our module and its connection to other modules within the system.

5.2 Hardware and software

Since our project does not include any hardware, this section will focus on only software we will leverage for our testing.

- PyCharm and Python's unit testing: PyCharm (a python IDE) and Python's own unit testing library will provide a sufficient enough software platform to create and run robust unit testing required for each of the modules within our project
- behave[2] is a Behavior Driven Development module within Python that will be a powerful tool within integration testing. It takes documents written in Gherkin[3] (which is a plain-english language made up of Given-When-Then statements) and translates them into tests to run. This will allow us to create easy to read and understandable tests for anyone not familiar with the project which will be vital as we are not the ones to maintain the code after we graduate.

5.3 Functional Testing

Functional testing will play a critical role in determining the success of our project. The following is an overview of the types of different functional tests that will be conducted for this project.

5.3.1 Unit Tests

For unit testing, the acceptance criteria is that ALL tests pass in order for the module to be considered "valid" and be integrated to the main branch of the repo and included in the project

NOTE: The testing plan will evolve and expand over time as things become more stable. This is the testing plan we have currently with the modules we have planned.

Table 4: Unit Testing Plan

Module Under Unit Test	**Testing Plan**
Data Importing (this will be done for each individual file module)	Test for "gold value" (normal behavior) Test when given no file path (NULL value) Test when given an invalid path Test when given a path to wrong type of file Test when given an empty file Test when given a directory Test when data within file is invalid
Data Visualization (Graphing, Plotly)	Test for "gold value" (normal behavior) Test when given a NULL value Test when given a non-supported type (not a pandas data type) Test when data has "none" type Test when conflicting plots are selected
Data Export	Test for "gold value" (normal behavior) Test for when given a NULL value Test for when given a non-Plotly graph Test for when user asks for a non-supported file type to export to Test for when user asks for an upscaled

	resolution Test for when user provides invalid Google Drive credentials

5.3.2 Integration Tests

For integration testing, we will utilize behave and the power it has to test the interactions between the two modules. The project itself will not be considered a stable release until it has passed ALL the integration tests.

An example of an integration test using behave and Gherkin is as follows:

Given the user provides a valid .csv file

And has selected the “bar graph” option

When the user clicks “export to .png”

Then the system will create a .png file for the user

This will give us readability and testability between multiple modules.

5.4 Non-Functional Testing

While non-functional testing is not of the highest priority for us in our first semester of the project, it is still on our minds. For compatibility, we have a desire to make sure that the program we develop is not only easy to download and use, but also can be used across multiple platforms. While Windows is the primary target, and having the system run natively with Python, we run into errors when we start requiring the end-user to download and install Python for the sole use of our program. In order to test these certain issues such as performance, usability, and compatibility, we will be heavily relying on field testing with the stakeholders to make adjustments based on what they need. Because the field of research our stakeholders are in is relatively foreign to the team, we will try to accommodate the standards that we know about, but will be looking for feedback in order to make sure our product matches the desires of the users and fits our non-functional requirements.

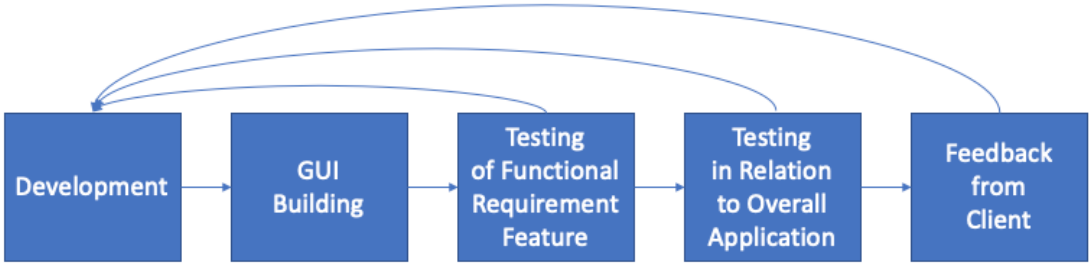
Part of our non-functional requirements stated above in the design document will be covered through our use of the unit testing and integration testing which will help reduce the burden on the stakeholders to provide large amounts of feedback to the team.

5.5 Process

Each functional and nonfunctional requirement feature in Section 2 undergoes the same testing process. In each iteration, code is written for that requirement. The code should be independent on other functional requirements besides what is passed in as parameters. It is also independent of the GUI. After this, the building of a GUI interface for these functions is performed. Once the GUI is completed, testing begins to ensure quality of the implementation of a feature of that functional

requirement begins, as well as testing of the feature working with other features and requirements. Tests that fail are rewritten, and the process begins again. After testing is completed, the feature is presented to the client, and feedback is implemented into the design. Once there are no revisions, the implementation of the feature is complete.

Figure 4: Flow Chart of Testing



5.6 Results

While the plan for testing has been set out in the few sections preceding this, it has not yet been implemented for the project and thus there are no results from it at the moment. A simple demo has been created for the software and no testing has occurred for it outside of simple verification that the demo software compiles and runs as expected.

6. Closing Material

6.1 Conclusion

Throughout this first semester of our project, the team has been working towards creating a design that meets the requirements of our client. We needed to design an application that allows the user to import pre-existing data from Excel and manages and visualizes the data. The design also needed to consider that the user must be able to customize the data variables used to make-up the graphs as well as the appearance of the graphs. Additionally, the design needed to address the need for saving backups of the data and allowing the data and visuals to be exported and shared with another person.

The design we have come up with will result in an easy-to-use application that does not require much maintenance, and allows our target client demographic, scientists and researchers, an easier method to organize, maintain, and visualize their data. We have worked on and experimented with the base components of the application to guarantee our design is feasible and meets our client's needs. A video demonstration of these initially completed basic components can be found in the Final Presentation Slides on our team's website. The video shows that the completion of these components went smoothly and indicates that our design is feasible and satisfactory. This design was the best choice for our project and we will continue to implement it next semester. We believe that by the end of next semester, we will have completed an application that confirms the exceptionality of our design choice and satisfies our client immensely.

6.2 References

[1]"How to Buy Prism," *GraphPad*. [Online]. Available: https://www.graphpad.com/how-to-buy/. [Accessed: 26-Apr-2020].

[2]"Welcome to behave!," *Welcome to behave! - behave 1.2.7.dev1 documentation*. [Online]. Available: https://behave.readthedocs.io/en/latest/. [Accessed: 18-Apr-2020].

[3]"Cucumber," *Introduction - Cucumber Documentation*, 2019. [Online]. Available: https://cucumber.netlify.app/docs/guides/overview/. [Accessed: 18-Apr-2020].