Microbiology Lab Information Management and Visualization System

DESIGN DOCUMENT

Team 29
Client: Karrie Daniels
Adviser: Thomas Daniels
Brittany McPeek – Report/Documentation Manger
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: 2/23/2020 / Version: 1

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)
- Often refactors
- Simple designs and 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
    - Including from sources such as CSV and Excel
- The system should be able to abstract the data and place them into modifiable graphs
- The system should support exporting those graphs to be published into research papers
- The system should be able to maintain past data and support adding and modification with new data
- The system will be written in Python
- The system should be easy to understand and use by users with little-to-no background in programming
- The 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 (This should be the similar to the project plan)

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 act 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 manage and visualize 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 organizing, maintaining, and visualizing their data.

1.3 Operational Environment

Our end product will most likely 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. Here 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
- The data should backup to a separate location

- Function to export the data to be shared with someone else

Our non-functional requirements are:

- It should be possible for the system to be maintained by one or two people
- The system should be secure enough so that research data can't be seen by anyone else
- We must use libraries in Python for data visualization
- The 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 provide 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 graphs layout as well as select the variables inputted into 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 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	The data parser and import customization, corresponding GUI
D2	October 15th	Data backup and invalid flagging, corresponding GUI
D3	November 15th	Graphing and graphing customization, corresponding GUI
D4	November 30th	Export functionality, corresponding GUI
D5	Ongoing	Documentation for above deliverables

Table 2: A description of the deliverables for the project

ID	Description
D1	The data parser will parse all rows and columns from an excel file and assign the data to specific variables. The import customization will allow the end-user to create new data types that can be parsed and specify the properties of said data-types. Variables that the parser does not recognize will be flagged and the end-user can dictate how to handle the unknown variables.
D2	Data backup functionality will create snapshots of data imported. In addition, it will flag data that is set outside of boundaries created by end-user for a variable type. This way, the end-user can identify data and make corrections.

D3	The end user will be able to create graphs based on variables. Graphs can be automatically generated and then customized by the end-user, or a new graph type can be created. New graph types can then be set to automatically generate in the future. The layout and type graph, as well as the inputs used, can be customized. Graphs can be static or set to change as more data is inputted.
D4	Export functionality will allow end-users to export the data-set (and it's backups) as well as all graphs in a data-set.
D5	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 likely to be made using the Tkinter 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 " xlrd" 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 of it. 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.

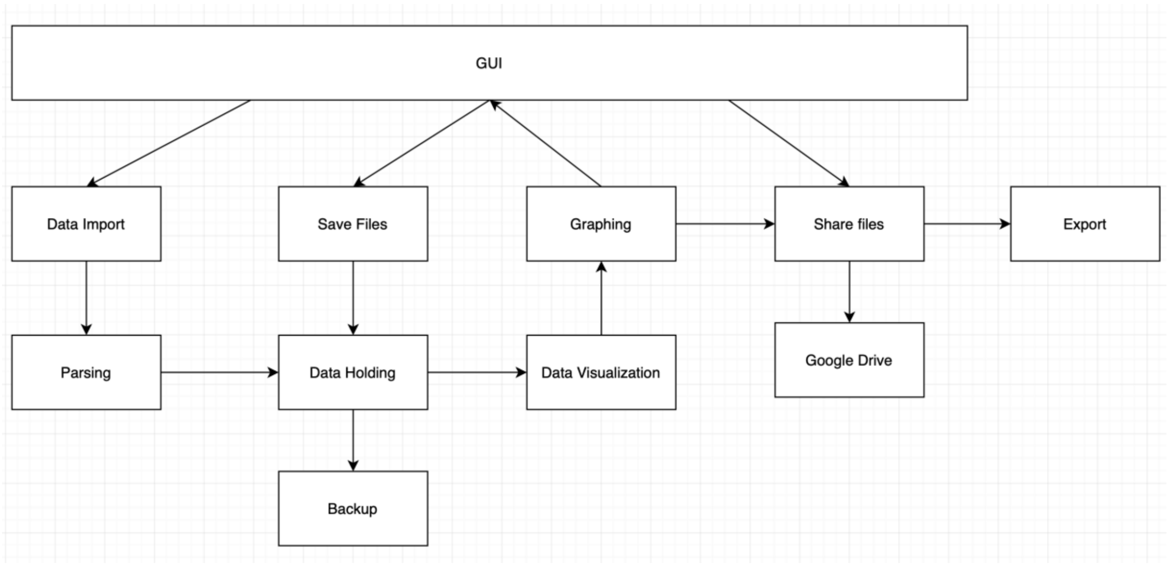
2.3 Development Process

The development processes 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, simplifying 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



3. Statement of Work

3.1 Previous Work And Literature

Include relevant background/literature review for the project

– If similar products exist in the market, describe what has already been done

– If you are following previous work, cite that and discuss the **advantages/shortcomings**

– Note that while you are not expected to "compete" with other existing products / research groups, you should be able to differentiate your project from what is available

Detail any similar products or research done on this topic previously. Please cite your sources and include them in your references. All figures must be captioned and referenced in your text.

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. The other downside that the client has leveraged to 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

Highlight the strengths, weakness, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

The possible solution we are bringing to the table would provide a free (or relatively inexpensive) 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 491 and 492.

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

In order to solve the problem at hand, it helps to decompose it into multiple tasks and to understand interdependence among tasks.

Our tasks can be decomposed into the modules similarly to the structure of our solution in {INSERT NAME OF FIGURE HERE}:

1. GUI
    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
    a. The client should be able to import CSV or EXCEL files to be analyzed and graphed
3. Parsing
    a. The solution should be able to parse data from the files and sort them into data structures for better visualization options
4. Data Holding
    a. Create dynamic data structures to contain the data for graphing uses
5. Saving Files
    a. The user should be able to save a current file within the application
6. Backup
    a. The system should store backups of the imported data in a hidden folder for data recovery
7. Data Visualization/Graphing
    a. Create a system that integrates with Plotly to create and show graphs to the end user
8. Share 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

Include any concerns or details that may slow or hinder your plan as it is now. These may include anything to do with costs, materials, equipment, knowledge of area, accuracy issues, etc.

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 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 working closely to communicate to each other and the client 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

What are some key milestones in your proposed project? Consider developing task-wise milestones. What tests will your group perform to confirm it works?

3.6 Project Tracking Procedures

What will your group use to track progress throughout the course of this and next semester?

3.7 Expected Results and Validation

What is the desired outcome?

How will you confirm that your solutions work at a **High level**?

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

• A realistic, well-planned schedule is an essential component of every well-planned project

• Most scheduling errors occur as the result of either not properly identifying all of the necessary activities (tasks and/or subtasks) or not properly estimating the amount of effort required to correctly complete the activity

• A detailed schedule is needed as a part of the plan:

– Start with a Gantt chart showing the tasks (that you developed in 3.3) and associated subtasks versus the proposed project calendar. The Gantt chart shall be referenced and summarized in the text.

– Annotate the Gantt chart with when each project deliverable will be delivered

• Completely compatible with an Agile development cycle if that's your thing

How would you plan for the project to be completed in two semesters? Represent with appropriate charts and tables or other means.

Make sure to include at least a couple paragraphs discussing the timeline and why it is being proposed. Include details that distinguish between design details for present project version and later stages of project.

4.2 Feasibility Assessment

Realistic projection of what the project will be. State foreseen challenges of the project.

4.3 Personnel Effort Requirements

Include a detailed estimate in the form of a table accompanied by a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be based on the projected effort required to perform the task correctly and not just "X" hours per week for the number of weeks that the task is active

4.4 Other Resource Requirements

Identify the other resources aside from financial, such as parts and materials that are required to conduct the project.

4.5 Financial Requirements

If relevant, include the total financial resources required to conduct the project.

5. Testing and Implementation

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or a software library

Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project:

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study for functional and non-functional requirements)
2. Define the individual items to be tested
3. Define, design, and develop the actual test cases
4. Determine the anticipated test results for each test case 5. Perform the actual tests
6. Evaluate the actual test results
7. Make the necessary changes to the product being tested 8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you've determined.

5.1 Interface Specifications

– Discuss any hardware/software interfacing that you are working on for testing your project

5.2 Hardware and software

– Indicate any hardware and/or software used in the testing phase

– Provide brief, simple introductions for each to explain the usefulness of each

5.3 Functional Testing

Examples include unit, integration, system, acceptance testing

5.4 Non-Functional Testing

Testing for performance, security, usability, compatibility

5.5 Process

– Explain how each method indicated in Section 2 was tested

– Flow diagram of the process if applicable (should be for most projects)

5.6 Results

– List and explain any and all results obtained so far during the testing phase

- – Include failures and successes

- – Explain what you learned and how you are planning to change it as you progress with your project
- – If you are including figures, please include captions and cite it in the text

  • This part will likely need to be refined in your 492 semester where the majority of the implementation and testing work will take place

-**Modeling and Simulation**: This could be logic analyzation, waveform outputs, block testing. 3D model renders, modeling graphs.

-List the **implementation Issues and Challenges**.

6. Closing Material

6.1 Conclusion

Summarize the work you have done so far. Briefly re-iterate your goals. Then, re-iterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

6.2 References

This will likely be different than in project plan, since these will be technical references versus related work / market survey references. Do professional citation style(ex. IEEE).

6.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar that does not directly pertain to the problem but helps support it, include that here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc. PCB testing issues etc. Software bugs etc.