Mass Spectrometer Interface

A Desktop Application for Reading Instrument Data

Cousins Photosynthesis Lab in the School of Biological Sciences at WSU



**Team Linnaea Borealis**

****

Kyler Kupp, Erik Holtrop

**Note**: Recall that this writing assignment says:

Length = 5+ pages text + appendices as needed.   
  
Some materials do not count towards this 5 page minimum. These excluded parts include:  
 Cover page  
 table of contents  
 pictures  
 tables  
 images  
 diagrams

Posted as a single self‐contained file (no links to outside resources.)

Posted as a PDF file.

Typed single‐spaced.

Typed with black text.

Typed with #11 font size.

Typed using Arial font.

Typed with one inch margins on sides, top and bottom.

**(Please erase this page in your final document.)**

**TABLE OF CONTENTS**

[I. Introduction 3](#_Toc82521031)

[II. System Overview 3](#_Toc82521032)

[III. Architecture Design 3](#_Toc82521033)

[III.1. Overview 3](#_Toc82521034)

[III.2. Subsystem Decomposition 3](#_Toc82521035)

[I.1.1. [Subsystem Name] 4](#_Toc82521036)

[a) Description 4](#_Toc82521037)

[b) Concepts and Algorithms Generated 4](#_Toc82521038)

[c) Interface Description 4](#_Toc82521039)

[I.1.2. [Include sections III.2, III.3, etc., for other subsystems] 4](#_Toc82521040)

[IV. Data design 4](#_Toc82521041)

[V. User Interface Design 4](#_Toc82521042)

[VI. Glossary 5](#_Toc82521043)

[VII. References 5](#_Toc82521044)

[VIII. Appendices 5](#_Toc82521045)

# Introduction

This document aims to outline our project’s approach. This includes the different parts of the project modules (Architecture Design), data storage and manipulation methods (Data Design), and how the project modules will outwardly look (User Interface Design). These details are intended to elucidate objectives and methods for current and future developers, but to do so using language that can mostly still be understood by stakeholders for overview. Our project is split into a few modules. The first 3 modules offer different perspectives on Mass Spectrometer data through graphs and logs, the 4th one reformats data from a 2nd Mass Spectrometer, and the 5th one combines data from three different instruments’ data-streams.

# System Overview

The system overview contains a general description of the functionality and design of the project. The overview will only briefly describe the overall design considerations and the comprehensive explanations will be done in the sections to follow. The overview should serve as an introduction to these sections.

# Architecture Design

## Overview

This section should describe the overall architecture of your software. The architecture provides the top-level design view of a system and provides a basis for more detailed design work. This will be the initial draft of your software architecture. Next semester you will revise this draft and finalize your design.

* Provide a bird’s-eye view of your software architecture. Mention the architectural pattern you adopted in your software and briefly discuss the rationale for using the proposed architecture (i.e., why that pattern fits well for your system).
* Please refer to CptS 322, CptS 487, CptS 321, and CptS 422 materials to refresh your knowledge on system decomposition and software architectural patterns.
* Briefly describe each layer/component in the architecture and explain its responsibilities.
* Provide a block diagram (e.g., UML component diagram) that illustrates the proposed architecture. The block diagram should show all major subsystems and identify the layers/components in the architecture.

## Subsystem Decomposition

This section explains how you decomposed your system into subsystems. A subsystem typically corresponds to the amount of work that a single developer can tackle. You will show your system decomposition, identify the major subsystems, describe the assignment of functionality to each subsystem, and define the interfaces between them. When you decompose your system into subsystems, you need to consider the dependencies within and between the subsystems, i.e., cohesion and coupling measures.

* Briefly explain how you decomposed your system into subsystems.
* Discuss the rationale for the proposed decomposition in terms of cohesion and coupling.
* Redraw your architecture diagram (in section III.1) and show all the services each subsystem provides and requires (for example, UML component diagram that uses ball-and-socket notation to depict provided and required interfaces).
* For each subsystem in your architecture, include a sub-section.
* To improve clarity, you may provide multiple figures that show different parts of the architecture (illustrating services) and place each figure right before the corresponding subsection.

### [Subsystem Name]

Include the following sub-sections for each subsystem.

#### Description

Describe the subsystem and identify its responsibilities.

#### Concepts and Algorithms Generated

Discuss the concepts, algorithms or solutions generated and considered for this subsystem. Report the selected solution and explain the solution selection process. Include any special considerations and/or trade-offs considered for the solution approach you have chosen.

#### Interface Description

Provide a description of the subsystem interface. Explain the provided services in detail and give the names of the required services.

Services Provided:

1. Service name:

Service provided to: [list the receiving subsystems here]

Description: [Describe what the service is and what it does. Provide its input and output values. Briefly describe the major functions that the service provides.]

Services Required:

Names of the required services and the subsystems that provide them.

### [Include sections III.2, III.3, etc., for other subsystems]

# Data design

This section covers the different ways our project stores data, both while it’s running, and when the data is stored for later. This overview will be a fairly technical overview, meant mostly for developers. Each subsection covers a different data structure used in the project.

## IV.1 Shared Singleton

A shared singleton is a design pattern that forces only one instance of an object to exist throughout the whole program. Although this design pattern does not usually describe a data structure, in modules 1-3, most important data structures are members of a shared singleton, named sharedData. These members include fileList (subsection IV.2) and dataPoints (subsection IV.3)

## IV.2 Basic List

In modules 1-3, we use a basic list to store the names of files that are read from. This list is named fileList and is stored in the sharedData shared singleton (subsection IV.1).

## IV.3 Dictionary

In modules 1-3, we use a Python dictionary to store the mass spectrometer data, named dataPoints. The keys are time points, and the values are tuples of isotope masses. This dictionary is stored in the sharedData shared singleton (subsection IV.1).

## IV.4 Log Table

In modules 1-3, there are certain data points that can be pulled out from the graph, or otherwise calculated from the data. These can be saved into a table built into the UI. These logs are then stored directly in the PyQT UI component QTableWidget.

## IV.5 CSV Export

Modules 1-3 can export logged data from the log table (subsection IV.4) into a CSV file, which matches the format of the table.

## IV.6 Pandas DataFrame

Module 4 uses a DataFrame from the Pandas Python library to store data from Mass Spectrometer 2. This is very similar to a dictionary approach (subsection IV.3), but effectively replaces the need for unique keys with ordering and more importantly allows for the use of Pandas methods like to\_csv at a small performance cost.

## IV.7 CSV Series

Module 4 exports the DataFrames (subsection IV.6) as CSV’s with ~8 rows each, for use in modules 1-3. Each row contains a time signature and isotope masses. These CSV’s are named with numbers in order of their time signatures.

# User Interface Design

[You may skip this section if your project doesn’t have a GUI component] – but! If the tools is ever to be used by humans (even just starting and stopping it), there’s some form of user interface design. It can be very simple, but it does exist. Make sure you document how you expect people to use your product, even if it’s just:

* Installation
* Configuration file edits
* Launch daemon by running command [x]

Provide a detailed description of user interface. The information in this section should be accompanied with proper images showing how exactly you vision the interface to be like (for example mock-ups). Make sure to mention which use cases in your “Requirements Specification” document will utilize these interfaces for user interaction.

# Glossary

Define technical terms used in the document.

# References

(Dutoit, 2010), 3rd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice Hall, 2010.

Cite your references here.

For the papers you cite give the authors, the title of the article, the journal name, journal volume number, date of publication and inclusive page numbers. Giving only the URL for the journal is not appropriate. You should use either IEEE or Chicago style formatting for your citations

For the websites, give the title, author (if applicable) and the website URL.

# VIII. Appendices

Any larger images, charts, or external materials should be put into appendices. These are attached at the end of the document, so the main materials are kept closer together and the overall flow of the document is preserved. If you include 4 pages of spreadsheets in the middle of a section, it makes it very difficult to track the flow of your presentation. Instead, those sheets go in Appendix [X] and are referred to by the earlier document.

You may have as many appendices as you need for the document to make sense.