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 = 3 pages text (or more because space goes fast once you start using lists and forms for user feedback) + appendixes as needed.

Not included in 3 page count:

* 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.**](#_gjdgxs) **Introduction 4**

[I.1.](#_30j0zll) Project Overview 4

[I.2.](#_1fob9te) Test Objectives and Schedule 4

[I.3.](#_3znysh7) Scope 4

[**II.**](#_2et92p0) **Testing Strategy 4**

[**III.**](#_tyjcwt) **Test Plans 4**

[III.1.](#_3dy6vkm) Unit Testing 4

[III.2.](#_1t3h5sf) Integration Testing 4

[III.3.](#_4d34og8) System Testing 4

[III.3.1.](#_2s8eyo1) Functional testing: 4

[III.3.2.](#_3rdcrjn) Performance testing: 4

[III.3.3.](#_26in1rg) User Acceptance Testing 5

[**IV.**](#_lnxbz9) **Environment Requirements 5**

[**V.**](#_1ksv4uv) **Glossary 5**

[**VI.**](#_44sinio) **References 5**

# Introduction

## Project Overview

The Mass Spectrometer Interface is a series of desktop applications designed to facilitate the analysis of mass spectrometry data for researchers in the Cousins Photosynthesis Lab at Washington State University (WSU). This application streamlines data processing for plant respiration studies, enabling efficient measurement of gases like CO₂ and O₂ and their isotopes. It replaces existing proprietary software that outputs extensive, often unnecessary data, focusing instead on essential data analysis functionalities that are critical for researchers. The following are the major functionalities of our modules that require(d) testing.

* **Concentration Calculations**: Module 1 computes gas ratios, such as bicarbonate to carbon dioxide, using real-time data.
* **Isotope Ratios**: Module 3 calculates and plots the portion of CO2 molecules that are 13C(18O)2​, a combination of particular isotopes of carbon and oxygen.
* **Real-time Derivative Calculation**: Module 3 calculates and plots the derivative of the isotope proportion described above in real time.
* **Data Conversion**: To support additional instruments, Module 4 converts raw data from a second mass spectrometer to the data format accepted by Modules 1-3, enabling integration with the lab’s current workflow and maintaining a standard for future projects.
* **Multi-Instrument Data Integration**: Module 5 combines data from multiple instruments, including the LI-COR Leaf-gas Exchange System, a tunable diode laser, and a Picarro device, into one unified data stream.
* **Graphing and Data Manipulation**: Modules 1-3 dynamically plots data streams, allowing users to manipulate graph scales, set plotting speeds, and isolate data segments for analysis. Movable “mean bars” let users select specific data ranges to calculate mean values, improving analysis precision.

## Test Objectives and Schedule

Our testing approach is very integrated into our development process. Developers apply unit testing as they work on a unit, then integration testing when a component satisfies unit tests. System testing is performed at standup meetings as needed.

Our project is compartmentalized into modules, which are our main deliverables. Typically our milestones involve the completion of a module, or a module’s prototype. The following table entails our expectations for production and testing, up to sprint 5.

|  |  |  |
| --- | --- | --- |
| **Milestone/Product** | **Expected Completion** | **Level of Testing** |
| Module 1 | Sprint 1 | User Acceptance |
| Module 3 | Sprint 2 | Unit |
| Module 3 | Sprint 3 | User Acceptance |
| Module 4 Prototype | Sprint 3 | Unit |
| Module 4 | Sprint 4 | Integration |
| Module 4 | Sprint 5 | User Acceptance |
| Module 5 Prototype | Sprint 5 | Unit |

## Scope

This document provides an overview of the steps we take to test different elements of our project. This overview includes the overall flow, the unit tests for individual parts, and integration/system testing for combinations of different parts. The processes outlined are very particular to the context of our project; we’re very aware of our stakeholders and the integration plan that they prefer. Ultimately this document is intended to outline what we consider the ideal methods of testing, in order to keep development in line with those practices.

The secondary intention of the document is for posterity and external communication. Stakeholders and future developers may find themselves curious about the testing process of this project, and this document is the primary resource for that rediscovery.

# Testing Strategy

The following is our loose approach to testing a particular module or feature:

1. Identify the requirement(s) involved in this module/feature. This should either come from the Requirements and Specifications Document or be added to the Requirements and Specifications Document before continuing.
2. Establish the test(s) that will be used. In other words, identify the process of using the module or feature. Document these tests in the Testing Plan Document.
3. Identify any necessary dependencies. This includes other components and input data. Include assumptions about these dependencies in the Testing Plan Document.
4. Build a representation of what acceptable results look like. This must consider our assumptions made in the previous step. For example, an Excel graph of a data acquisition: the particular data acquisition should be clarified in the previous step, with the Excel graph built off it in this step. This mockup(s) should either be included in the Testing Plan Document, or in the relevant module’s “Testing” folder with reference to it in the document.
5. Perform the test(s).
6. If the test(s) is unsuccessful, fix it if possible. If the test(s) is not successful by next standup meeting, prepare a short explanation or document explaining the issue.
7. If the test is successful, move the relevant GitHub issue to Review/QA, or from Review/QA to Done.

Ultimately, our strong connection to our primary stakeholder, Dr. Cousins, allows us to adapt our development process to a more flexible approach that handles opportunities and issues as they come up. Sometimes the requirements are vague, and the following approach may be more effective than generating more specific requirements and a mockup:

1. Implement the most obvious executions of a requirement.
2. Present those executions to the client/stakeholder(s). Receive feedback.
3. If one of the implementations is acceptable, move the relevant GitHub issue to Review/QA, or from Review/QA to Done.
4. If none of the implementations are acceptable, either return with novel implementations or revert to the primary approach, depending on team consensus.

Our delivery process is basically Continuous Delivery. Our client prefers executable files over python scripts, so a new deployment must be manually created by a team member each time. The modular nature of the project lends itself to creating a new iteration of each improved module every sprint. In this regard, our development is continuously integrated with monthly releases.

# Test Plans

Describe the plan for testing your project in the context of the following testing activities. You may include additional test activities, if necessary.

For each of the following activity, describe how the testing will be conducted. What would be the sequence of events, and how will the testing activity take place? Please refer to the CptS422 class notes for details on testing strategies.

## Unit Testing

The primary goal of unit testing is to take the smallest unit of testable software in the application, isolate it from the remainder of the code, and test it for bugs and unexpected behavior.

## Integration Testing

Integration testing detects faults that have not been detected during unit testing by focusing on small groups of components. Two or more components are integrated and tested, and when no new faults are revealed, additional components are added to the group.

## System Testing

System testing is a type of black box testing that tests all the components together, seen as a single system to identify faults with respect to the scenarios from the overall requirements specifications. Entire system is tested as per the requirements.

During system testing, several activities are performed:

## Functional testing:

Test of functional requirements (from requirements specification). The goal is to select those tests that are relevant to the user and have a high probability of uncovering a failure.

## Performance testing:

Performance tests check whether the nonfunctional requirements and additional design goals from the design document are satisfied. In stress testing, system is stressed beyond its specifications to check how and when it fails.

## User Acceptance Testing:

Acceptance testing and installation testing check the system against the project agreement. The purpose is to confirm that the system is ready for operational use. During acceptance test, end-users (customers) of the system compare the system to its initial requirements (if necessary) with help by the developers.

# Environment Requirements

Specify both the necessary and desired properties of the test environment. The specification should contain the physical characteristics of the facilities, including the hardware, communications and system software, the mode of usage (for example, stand-alone), and any other software or supplies needed to support the test. Identify special test tools needed.

# Glossary

Define technical terms used in the document.

# References

Cite your references here. Please use one style for the references. You’re welcome to choose between: IEEE and Chicago style formats. I highly recommend using scholar.google.com to help with the formatting. Seriously, scholar.google.com is an incredibly powerful tool to both find citations, and to generate well formatted citations for papers/materials you’ve already found.

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.

For the websites, give the title, author (if applicable) and the website URL. Here’s a format for it:

* http://www.easybib.com/reference/guide/apa/website