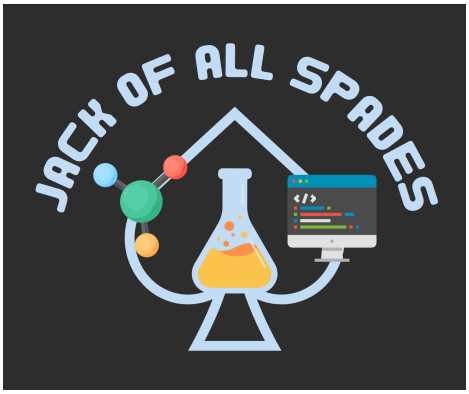

Jack of All Spades
Eric Jestel, Alex Ayer, Bryce Roy, Chris Letourneau, Conall Gouveia

System Design Document

Chemistry Instrumentation: Control, Data Acquisition and Offline Analysis
For Dr. Mitchell Bruce with the Chemistry Department at the University of Maine, Orono

11/17/2025

V1.0

Table of Contents

1 Introduction

This document presents the system requirements for Chemistry Instrumentation: Control, Data Acquisition and Offline Analysis. This project is being undertaken by a team of seniors at the University of Maine in order to fulfill the capstone requirements of a Computer Science degree. This is being undertaken in collaboration with the chemistry department at the University of Maine.

1.1 Purpose of This Document

Within this document, the system architecture, major components, and interfaces of the project are laid out. It details how all components interact, how those components satisfy our requirements, and specifies what files will be created or utilized by the program. This document will serve as a foundation during development, ensuring that we generate a functional product that aligns with the intended design.

This document was reviewed in collaboration with the client to create a common foundation for the project. It serves as a guide to the team for system development and code writing. Additionally, it allows the client and the Computer Science Department to assess the progress of the team and provides guidelines for assessing project completion.

1.2 References

Bruce, M. (2025). *Chemistry Instrumentation: Control, Data Acquisition and Offline Analysis*. ms, University of Maine.

Fowler, M. (2018). *UML distilled*.

Jestel, E., Ayer, A., Roy, B., Letourneau, C., & Gouveia, C. (2025). *System Requirements Review*. ms. University of Maine.

Jestel, E., Ayer, A., Roy, B., Letourneau, C., & Gouveia, C. (2025). *User Interface Design Document*. ms. University of Maine.

2 System Architecture

The System Architecture section provides an overview structure and design of the system. It shows how the system's components interact with the outside actors and the architectural requirements needed. This section is a visual overview of the project requirements. It illustrates how data flows between the system, and external interfaces to achieve the intended outcome, which is a functional and reliable system.

2.1 Architectural Design

This section describes the system architecture, outlining the components involved and their interactions with one another.

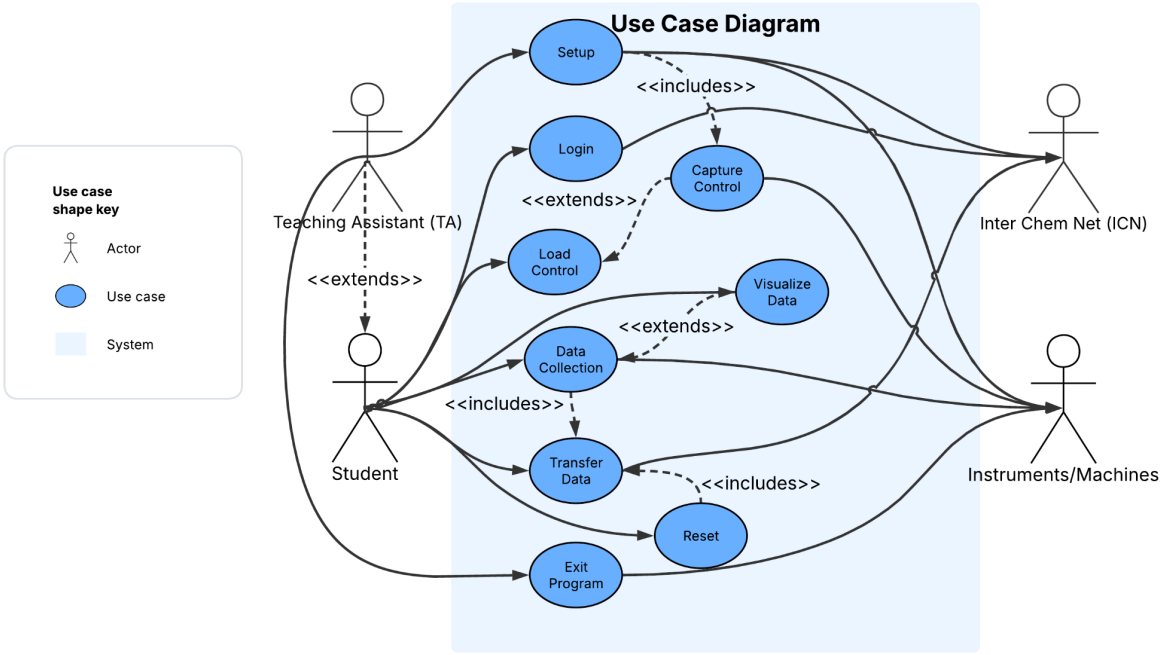

Figure 2.1.1 Use Case Diagram. This figure displays the actors of the system, the use cases, and how they interact. On the left are users, the TA, and the student, while ICN and the instruments, on the right, provide necessary services.

First, we present a use case diagram (Fig 2.1.1), displaying the actors in the system and the use cases. The user actors are the student class and the Teaching Assistant (TA) class, which extends the student user. The student can access many of the basic actions in the system, while the TA also has access to setup and shutdown procedures. The TA prepares the system once for the lab session, then multiple students use it sequentially. The TA can also perform those actions if desired. The providers of our system are the ICN Server and the instruments, both of which our system interacts with.

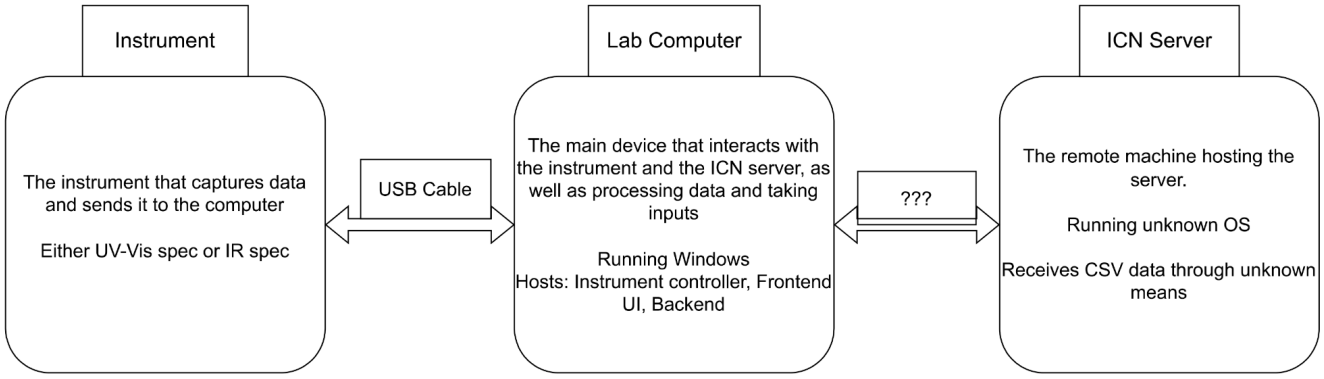

Figure 2.1.2 Logical architecture diagram. This diagram displays the hardware involved in the systems and the interactions between those components. The processes, purpose, and operating system are displayed for each where known.

With the use cases and interactions defined, we present a Logical Architecture Diagram (Fig. 2.1.2). This diagram describes the hardware necessary for the program to be utilized. We have 3 major components: **Instrument**, **Lab Computer**, and **InterChemNet (ICN) Server**. The **Instrument** is a substitutable piece of technology which captures Ultraviolet-Visible (UV-Vis) or Infrared (IR) spectra. Our program will be compatible with either possible device. The **Lab Computer**, a Windows 11 compatible device, runs the program and communicates with the **Instrument** and the **ICN Server**. The **Instrument** communicates with the **Lab Computer** over a USB connection, using specific program codes for each device. Currently, the communication mechanism for the **ICN Server** has not been defined. A meeting is scheduled with the team running the **ICN Server** to finalize the communication procedures.

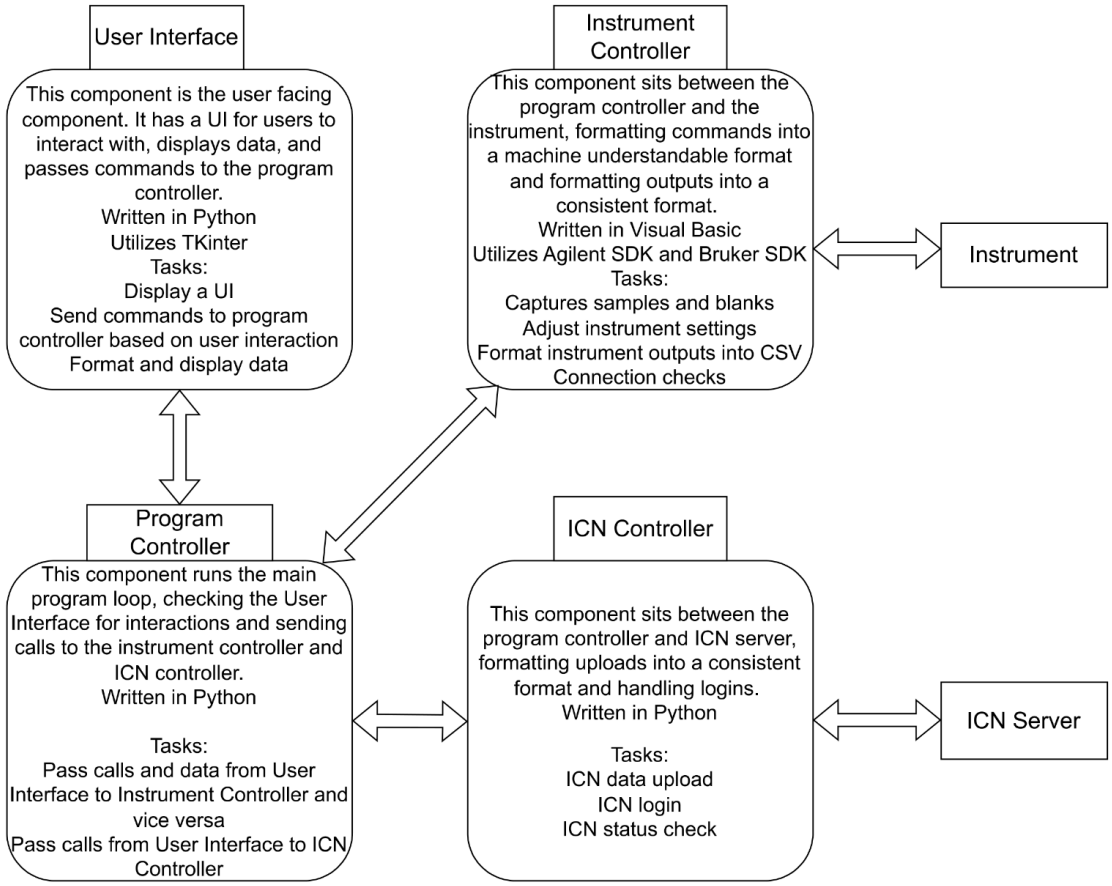

Figure 2.1.3 Technology architecture diagram. This diagram displays the software components and their interactions. The purpose, tasks, and implementation details are specified for each.

Lastly, we present a Technology Architecture Diagram (Fig. 2.1.3). In this, we explain the function of the software components and how they are connected. We have the **User Interface** (**UI**), which is the component that users see. This is where graphic elements and interactable components are held. The **UI** communicates with the **Program Controller** (**PC**), sending commands based on component interactions and receiving data and response codes. The **UI** utilizes the Python language and TKinter (an interface library for Python). Through the **UI**, users can send commands to the **PC**. The **PC** is what connects the software components together, centralizing the system. This component, written in Python, processes commands from the **UI** and forwards them to the external controllers. The first external controller is the **Instrument Controller** (**IC**), which interacts with the instrument, processing commands from the **PC**, sending commands to the instrument, receiving data and writing it into files, and sending response codes back to the **PC**. The IC is written in Visual Basic and also translates the data from the instruments into the proper format for upload to ICN. The **ICN Controller** (**ICNC**) is the other external controller. The ICNC is written in Python and communicates with the ICN server. It receives commands from the **PC**, reads data from files, uploads it to the ICN server, and sends response codes to the **PC**. It also can ping the ICN server to verify connectivity and verify usernames against the ICN server.

2.2 Decomposition Description

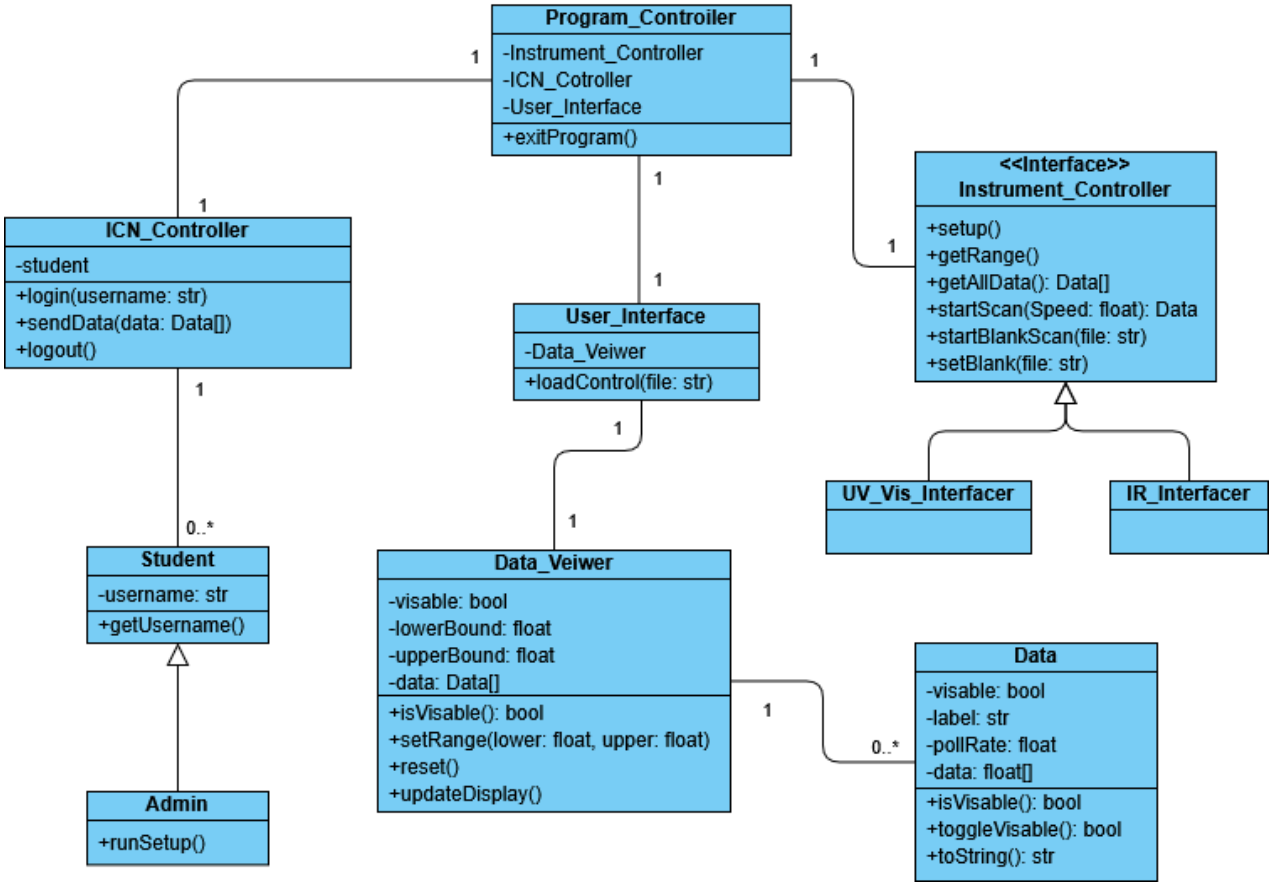

Figure 2.2 UML Class Diagram. This figure shows the class breakdown of the system

For this system, we will be following an object-oriented approach to construct the application. There are three major aspects that our system will need to achieve. First, we need a procedure to interface with the instruments from within our system. Then we need to have an easy-to-use interface for the user to interact with. Finally, we need to be able to connect to ICN so that we can upload the data for future use. By splitting each of these three aspects into their own class, we can focus on making each component work individually.

The Instrument Controller's purpose is to translate a user's actions into instructions for the instrument to follow. Given how we will be making two separate applications, one for each instrument, we plan on reusing most of the application between the two programs. To facilitate this, we will be making the Instrument Controller a module. We will develop the application using the UV-Vis Controller first, then develop the IR Controller afterwards to create the second application. Ideally, we will be able to utilize the same User Interface and ICN Controller modules during our development. The Instrument Controller will translate the data from the instrument into the format that the ICN server uses.

The ICN Controller will connect to the ICN Server to upload data and validate login credentials. By separating these aspects into their own class, we can test parts of the application without needing to connect to ICN. Because the data is already formatted by the Instrument Controller, it can be uploaded directly.

The User Interface is the user facing component. It contains a data viewer that will allow the user to visualize the recorded data. All inputs are entered here and passed to the program controller. The user will be able to enter credentials, take samples, and access advanced options from this component.

3 Persistent Data Design

This section of the System Design Document describes what data the system will handle, what files it must interact with, and any database information. It will detail the required database interactions for the system and the output of each instrument, as well as the format of the output files and how they need to be reformatted for database entry.

3.1 Database Descriptions

Account information and data storage for this system will be handled in collaboration with a third-party actor, AVibe. Their software development team will be responsible for storing the data and ensuring it gets uploaded properly to ICN. This system is solely responsible for properly formatting the data from the spectra and sending it to AVibe.

A meeting with AVibe is being planned, but has not been finalized. This meeting is necessary to define the methods used to interact with AVibe's servers and set up testing protocols. Until such a meeting has occurred, the specific methods for data transfer cannot be defined. However, given the compartmentalized nature of the planned system, development will focus on the user interface, instrument controller, and program controller.

3.2 File Descriptions

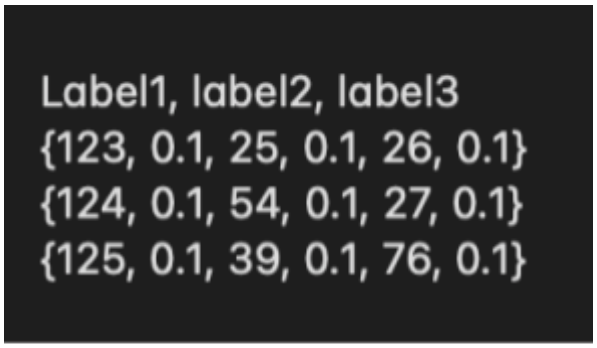

Figure 3.1.1: Example CSV Output. This figure shows the output file from the spectral analysis machines.

The system will use two separate file types, with one corresponding to raw machine output and the other to the formatted data that is sent to AVibe. Currently, both spectral analysis machines output a comma-separated values (CSV) file containing the data collected from the spectra. Figure 3.1.1 shows the format of these CSV files. Each CSV file has the labels of each sample collected. Each "label" is one of the samples that was named by the user. Each row of comma-separated values coincides with one label. Taking the first row as an example, 123 is the first wavelength measured in nanometers(nm), with a reading of 0.1

absorbance. Similarly, 25 would be the second wavelength measured in nm, also recording 0.1 absorbance. All elements in the first row correspond to Label1 and represent 1 sample.

Figure 3.2.1: Formatted JSON. This figure shows a JSON that is ready to be sent to AVibe for database entry.

The ICN only accepts JavaScript Object Notation (JSON) files for uploaded files. It does not accept CSV files, meaning the system must reformat the CSV output into the proper accepted format. To keep formatting consistent between different machines, the instrument controller will store files locally on the machine as JSON files. The format to be utilized is shown in Figure 3.2.1. Each JSON file has three fields: **label**, **data**, and **type**. The **label** field is a string that contains the name of the sample and corresponds to 1 label in the CSV (See figure 3.1). This sample name shall not be more than twenty characters. The **data** field contains an array of 2-element arrays. Each of the inner elements contains two floating data points representing the wavelength and absorbance values collected from the sample spectra. From the CSV file, this maps [A1,A2,B1,B2,C1,C2] to [[A1,A2], [B1, B2], [C1, C2]]. The **type** field is representative of the type of data being sent. This field enables the data to be correctly represented, as both instruments will have similar-looking data, but differ in the type label.

In the example CSV in Figure 3.1.1, there are three labels, or samples, being exported in a single CSV. This would need to be converted to 3 separate JSON files, each of which can contain at most 1 sample. While a multi-entry CSV is a possible output from the instrument, our system will handle each sample taken independently. Thus, we will never need to convert one CSV file to multiple JSON files. However, we will write the code in a way that handles such a situation to allow for future development of the system.

4 Requirements Matrix

This section aligns system components with functional requirements (tagged as UC-0XX). Additionally, a brief description accompanies each functional requirement, explaining how those requirements are satisfied.

Requirement	Satisfied by	Explanation
UC-001 : Setup	Program_Controller, Instrument_Controller, ICN_Controller	Initializes the program, checks instrument and network connections, and displays the login screen. Triggered by the Lab Manager starting the software.
UC-002 : Capture control sample	Instrument_Controller, User_Interface	Captures and stores the blank sample locally; the interface handles prompts, confirmations, and instrument error messages.
UC-003 : Login	ICN_Controller, User_Interface	Authenticates the user through ICN and displays login results or error messages on the interface.
UC-004 : Load control sample	User_Interface, Instrument_Controller	The user interface contains the UI for load control sample, allowing the student to choose a preexisting option. Additionally, they can capture a new sample using the instrument controller.
UC-005 : Data collection	Instrument_Controller, Program_Controller	Captures sample data through the instrument, saves it locally, and automatically starts the data transfer process.
UC-006 : Visualize data	Data_viewer	Once the data is collected, it is passed to the data viewer, where it is graphed. The graph parameters can be changed.
UC-007 : Transfer data	ICN Controller, Program_Controller	Sends collected data to the ICN Server, marks files as sent, and reports upload results or errors.
UC-008 : Reset	Program_Controller, User_interface	Clears session data, calls UC-007, logs out the user, clears all UI changes, and returns the system to its default state.
UC-009 : Exit program	Program_Controller	Verifies that all data has been uploaded, deletes temporary files, and closes the application.

Table 1. Requirements Matrix. This table describes the requirements, which component satisfies them, and how that component does so.

Appendix A - Agreement Between Customer and Contractor

This document outlines the system design that is agreed upon by the customer and the development team for the Chemistry Instrumentation project. By signing and dating in the specified fields, both parties agree on the information outlined within the contents of this document.

In the event that any information within this document changes, the software team is liable to meet with the client and review any alterations. If both parties agree to the new information, new signatures and dates will be recorded for the new version.

Mitchell Bruce		**12/11/2025**
Customer Name	Customer Signature	Date
Alexander Ayer		**11/17/2025**
Team Member	Team Member Signature	Date
Conall Gouveia		**11/17/2025**
Team Member	Team Member Signature	Date
Eric Jestel	/s/ Eric Jestel	**11/17/2025**
Team Member	Team Member Signature	Date
Chris Letourneau	/s/ Chris Letourneau	**11/17/2025**
Team Member	Team Member Signature	Date
Bryce Roy		**11/17/2025**
Team Member	Team Member Signature	Date

Appendix B – Team Review Sign-off

Below are the signatures of all the team members working on the chemistry instrumentation project. By signing this document, all team members agree to its contents and format. Along with signatures, any member can provide comments in the provided space to list any concerns or disagreement that they may have with the information covered within this document.

Alexander Ayer		**11/17/2025**	N/A
Print Name	Signature	Date	Comments
Eric Jestel	/s/ Eric Jestel	**11/17/2025**	N/A
Print Name	Signature	Date	Comments
Chris Letourneau	/s/ Chris Letourneau	**11/17/2025**	N/A
Print Name	Signature	Date	Comments
Conall Gouveia		**11/17/2025**	N/A
Print Name	Signature	Date	Comments
Bryce Roy		**11/17/2025**	N/A
Print Name	Signature	Date	Comments

Additional Comments:

Appendix C – Document Contributions

Below is a list of contributions each member made to the writing of the System Design Document.

Eric Jestel: Document setup, title page, Section 1, Figs. 2.1.2, 2.1.3. Writing in Sections 2.1, 4. Collaboration on section 2.2. Revisions on Section 2.2, 3.

Alex Ayer: Wrote appendix A, and B. Writing in section 3.1, 3.2, created figure 3.1.1 and 3.2.1

Bryce Roy: Wrote Section 2.1 with Conall, made figure 2.1.1, corrected formatting errors in document, reviewed Section 2.2.

Chris L: Formatted Section 2.1, Wrote Section 2.2, Created Figure 2.2, Wrote Reference List, Did the Rubric Review and Final Formatting Pass

Conall Gouveia: Edited Requirements Matrix, Reviewed all diagrams, Edited section 2.1, reviewed whole document