UTS: ENGINEERING & INFORMATION TECHNOLOGY

SUBJECT NUMBER & NAME 41113 Software Development Studio	**NAME OF STUDENT(s)** (PRINT CLEARLY) *Luke Herron* *Sanghyeon Park* *Bhavya Khemlani* *Chi-Tai Nguyen* *Mitchell Lee* *Husam Ajaj* *SURNAME* *FIRST NAME*	**STUDENT ID(s)** 24494116 13980154 24445076 14252593 13248202 24517880

STUDENT EMAIL **Luke.w.herrron@student.uts.edu.au** **Sanghyeon.Park-4@student.uts.edu.au** **Bhavya.khemlani@student.uts.edu.au** **Chi-tai.nguyen@student.uts.edu.au** **Mitchell.Lee@student.uts.edu.au** **HusamMohammed.Ajaj-1@student.uts.edu.au**	**STUDENT CONTACT NUMBER** **0450-040-948** **0478-101-095** **0478 949 133** **0460 846 370** **0416-684-820** **0422-273-775**

NAME OF TUTOR **Grace Billiris**	**TUTORIAL GROUP** **03**	**DUE DATE** **13/11/2023**

ASSESSMENT ITEM NUMBER & TITLE
Assessment task 3: Product Assessment

I acknowledge that if AI or another nonrecoverable source was used to generate materials for background research and self-study in producing this assignment, I have checked and verified the accuracy and integrity of the information used.
I confirm that I have read, understood and followed the guidelines for assignment submission and presentation on page 2 of this cover sheet.
I confirm that I have read, understood and followed the advice in the Subject Outline about assessment requirements.
I understand that if this assignment is submitted after the due date it may incur a penalty for lateness unless I have previously had an extension of time approved and have attached the written confirmation of this extension.

Declaration of originality: The work contained in this assignment, other than that specifically attributed to another source, is that of the author(s) and has not been previously submitted for assessment. I have rewritten any material provided by AI or other nonrecoverable sources and where appropriate acknowledged their contribution. I understand that, should this declaration be found to be false, disciplinary action could be taken and penalties imposed in accordance with University policy and rules. In the statement below, I have indicated the extent to which I have collaborated with others, whom I have named.

No content generated by AI technologies or other sources has been presented as my own work and I have rewritten any text provided by AI or other sources in my own words.

Statement of collaboration:

L.H.
S.P.
B.K.
H.A
M.L

Signature of student(s) ____________________ **Date** _____**13/11/2023**_____

Software Requirements Specification

GUI and Software Backend for Simulating Cryogenic Wiring in Dilution Refrigerators

Version	0.4.0
Author(s)	Mitchell Lee, Bhavya Khemlani, Luke Herron, Husam Ajaj, Sanghyeon Park, Chi-Tai Nguyen

Table of Contents

1. Document Management

1.1 Revision History

Date	Version	Description of Change	Author
23/08/2023	0.1.0	INITIAL DRAFT	Luke Herron
30/08/2023	0.1.1	Introduction	Luke Herron
6/09/2023	0.1.2	Functional & Non-Functional Requirements	Luke Herron & Calvin & Bhavya
13/09/2023	0.1.3	Project Scope, User Case Narratives & User Case Story	Calvin & Luke Herron
18/10/2023	0.2.0	FINAL DRAFT	Luke Herron & Bhavya Khemlani
10/11/2023	0.3.0	FINAL DRAFT – with edits	Bhavya Khemlani
12/11/2023	0.4.0	FINAL REPORT	Luke Herron

2. Introduction

2.1 Document Purpose

The primary purpose of this Software Requirements document is to serve as a foundational guide that outlines the exact specifications, functionality, and expectations for the Cryogenics Wiring software project that we've collaborated on with stakeholders from CirQuS UTS. This vital documentation will endeavour to be a pivotal communication tool that will be provided to our client as a source and proof of work for the project. This SRD further attempts to create a shared understanding of the project's objectives, functionality, and constraints. It also acts as a reference point for assessing project progress and for informed, collaborative decision-making through the Agile development process of Scrum. Resultantly, this SRD is a vital output of our project that will aim to bridge the gap between our developed product and the process/guidelines that have been followed throughout this semester-long initiative.

2.2 Project Purpose

To develop a GUI and software backend for simulating cryogenic wiring in dilution refrigerators. This design aims to identify the optimal wiring configuration for conducting experiments in their dilution refrigerators. The current design, however, lacks a user-friendly interface, making it essentially inoperable for new users that are unfamiliar with its backend operations.

As a result, the CirQus UTS research group intends to address this challenge through developing an intuitive web-based graphical user interface (GUI). This new GUI will enable new and pre-existing users to easily perform similar analyses on their own dilution refrigerator cryostats. By leaning towards a user-friendly design, the software will become more accessible and allow a broader range of users to make use of its powerful functionality and carry out informed experiments regarding their cryostat wiring configurations.

2.3 Project Scope

The figure below depicts a high-level use case diagram that illustrates the key actions or use cases that the actors can perform in the system and is to be used to provide an overview of the systems functionality.

The system serves as a web-based tool for CirQuS UTS' software, designed to determine the best wiring layout for quantum experiments conducted within their dilution refrigerator cryostats, which operate at extremely low temperatures in the millikelvin range. Users can utilize the application to input their own custom or predefined wiring and component information, which can be used to create a configuration. Components can be adjusted or added as required. The system can receive cooling power measurements for the refrigerators to create a model for heat load. Graphs displaying noise and heat data are automatically generated and can be customized to highlight specific relationships. Additionally, this generated heat load information can be exported in CSV format.

In addition, the software can be independently downloaded and interacted with by power users for advanced manipulation and functionality.

The two primary users of this system are the Cryostat user and the Power user. The power user will be able to access the simulation functions independently of the GUI.

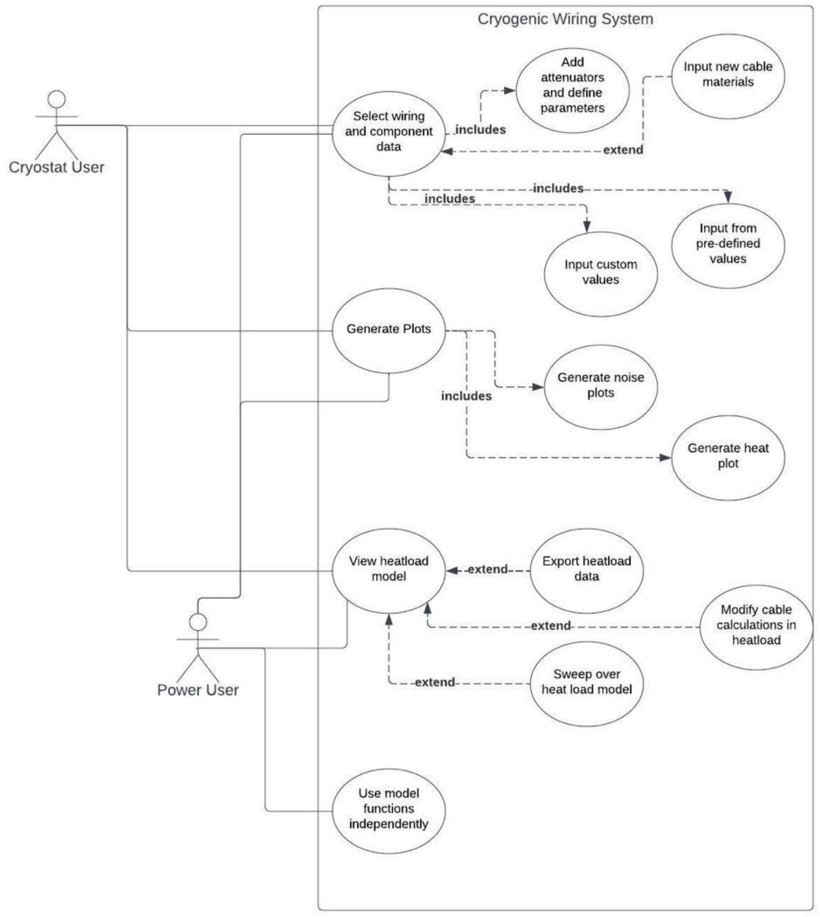

Cryogenic Wiring System
Cryostat User
Power User
Select wiring and component data
Add attenuators and define parameters
Input new cable materials
includes
extend
includes
includes
Input custom values
Input from pre-defined values
Generate Plots
Generate noise plots
Generate heat plot
includes
View heatload model
Export heatload data
extend
Modify cable calculations in heatload
extend
Sweep over heat load model
extend
Use model functions independently

Figure 1: Use Case Diagram for the proposed system

2.3.1 In Scope

The following table outlines the items that are considered in scope for the Cryogenic Wiring GUI project:

Ref.	Item	Description
S001	Select wiring and component data from supplied parameter values.	Users will be able to determine wiring and component data by selecting a diverse range of pre-supplied parameter values resulting in a seamless and simplified process of specifying technical configurations.
S002	Upload wiring and component data in a standardised form.	This will provide the flexibility to define wiring and component data using custom parameter values, allowing users to tailor the configurations according to their specific requirements, beyond the pre-supplied options.
S003	Creating GUI graphs.	Using d3.JS for graph rendering. This will aid users in assessing the alignment between their expectations and the chosen wiring configuration, based on the inputted parameter values.
S004	Polishing Data	Polishing user input and data visualisation outputs

S005	Updating Backend	Updating some backend features for changes in the supplied Python analysis package.
S006	Documentation	Documentation on how to use the tool.

2.3.2 Out of Scope

Conversely, the table below outline the few objectives that are not made available in our final project:

Ref.	Item	Description
O001	Deployable	We don't host the product.
O002	Enable users to program in new input types.	Offers the flexibility to incorporate unforeseen inputs into the simulation even after deployment. This ensures that the system remains adaptable and future-proof, allowing users to incorporate novel or customized inputs as necessary, enhancing the versatility and longevity of the software.
O003	Export visualisations as image.	This will enable users to save generated visualisations, providing them with the convenience of referring to them at a later time without the need to rerun the simulation.

2.4 Assumptions

Ref.	Item	Description
AS001	Python API	Accurate, Stable.
AS002	Stable Internet	User has a stable internet connection.
AS003	Browser	User has a compatible browser.
AS004	Language	Users can read English.
AS005	Updated	User is running up-to-date software.
AS006	Software	User has the right software installed.
AS007	Device	User has a compatible device.

2.5 Risks

Outlined in the table below are core risks associated with this project and their related ratings. This matrix is a combination of likelihood (possible, likely, rare, etc.), consequence (major, minor, etc.) and overall rating (high, moderate, very high, etc.) – this will improve the visibility and foresight of potential project risks before they occur.

RID	Description	Likelihood (Possible, Likely etc.)	Consequence (Major, Minor etc.)	Rating (Very High, High, Moderate etc.)	Mitigation Strategy
R001	Python backend malfunctioning or returning significant server errors	Rare	Major	High	Regular server resets are a viable mitigation strategy to ensure the python backend availability is maintained
R002	Server Abuse	Unlikely	Major	High	Permissions are to be managed and assigned carefully and with all potential risks considered
R003	Bad data	Rare	Minor	Low	Update parameter filters and ensure data input is validated and

					cleansed wherever possible
R004	Scope Creep	Possible	Minor	Moderate	Limit scope initially once project requirements have been identified
R005	Team members leaving	Unlikely	Major	Low	It is unlikely this issue is to be encountered throughout our project however it could be managed by reassigning roles and re-delegating responsibilities

2.6 Issues

IID	Description	Responsible	Status	Date	Resolution
I001	Which plots to use from thesis	Chi-Tai	Complete	30/08/2023	10/09/2023
I002	Client provided faulty and incomplete code.	Mitchell	Complete	04/09/2023	20/09/2023
I003	Which sweeping method to be used.	Calvin	Complete	30/08/2023	5/09/2023
I004	How to run test code	Luke	Complete	12/10/2023	12/10/2023

2.7 Constraints

CID	Item	Description
C001	Python backend	Preference for Python-based backend due to its versatility, extensive library support, and ease of integration with various data processing and visualization tools.
C002	Web Browser	Must be using a browser interface to ensure broad accessibility and ease of use, with a focus on creating a platform-independent, user-friendly web application that can be accessed from various devices.
C003	Integrate with existing project	Try to extend the existing design to maintain consistency with current user experience and leverage existing functionalities, while ensuring new features and enhancements are seamlessly integrated without disrupting the core system.
C004	Client-Side Processing	The software may require client-side processing capabilities, such as the ability to run Python scripts in the browser for certain functionalities (e.g., sweep plotting).
C005	Data Visualization	Effective data visualization capabilities are required, including the ability to generate and manipulate various types of graphs (heat loads, noise graphs) dynamically.
C006	Offline Functionality	There should be an offline version of the application with limited functionalities compared to the online version, to cater to users without consistent internet access.
C007	Data Handling and Processing	The software should allow for a high degree of customizability and extensibility, such as creating custom cable materials or configurations.
C008	Customizability and Extensibility	Operate effectively under specified environmental conditions and adhere to any operational constraints related to the project's context.

C009	User Interface Responsiveness	The user interface must be responsive and capable of adapting to various screen sizes and devices, ensuring a consistent user experience across platforms.
C010	Cross-Browser Compatibility	The software must be compatible with multiple web browsers to ensure accessibility to a wider user base.
C011	User Interaction and Feedback	The system should provide clear and immediate feedback to user interactions, such as updates, configurations changes, or error messages.

2.8 Dependencies

The following table attempts to comprehensively outline all the dependencies associated with the CirQuS Cryogenic Wiring project covering areas of development, communication and scheduling dependencies.

DID	Item	Description
D001	Python Package	The success of our final, developed software product will be reliant on the full integration of the stakeholder supplied Python package.
D002	Regular Meetings with CirQuS	Regular stakeholder meetings will be a key dependency in the success of this project as communication with the client will allow for improved development outcomes and minimise aforementioned project risks
D003	Scheduling of particular Modules' Development	Specific modules of the software should be developed in a particular order as they depend on one another for functionality. For example, the software code for the parameter inputs should be developed before the graphing functionality can be created.

2.9 System Users and Channels

Ref.	Item	Description
A001	CirQuS	Project Manager and Product Owners
A002	Cryostat Users	Refers to individuals and groups who will make use of the web application primarily through the UI to rudimentary calculations and tasks.
A003	Power Users	Users that have advanced technical knowledge and will have direct access to the software's backend for calculations and tasks.

3. Functional Requirements

3.1 Functional Requirements - User Stories

ID	Initial Requirement	User Story	Acceptance Criteria
US001	Upload wiring and component data in a standardised form.	As a cryostat user, I want to upload my own wiring and component parameter data, so that I can evaluate the expected performance of certain metrics for that particular wiring configuration.	Given that the inputted values are in a standardized format, When the user has filled every required parameter field, Then ensure the expected performance of each metric for that particular wiring configuration is displayed on the GUI.
US002	Select wiring and component parameter data from lists of pre-supplied parameter values.	As a cryostat user, I want to select parameter values from a list of pre-supplied wiring and component data, So that I can evaluate the expected performance of certain metrics for that particular wiring configuration.	Given that the inputted values already exists within the pre-supplied wiring and component data, When the user has filled every required parameter field, Then ensure the expected performance of each metric for that particular wiring configuration is displayed on the GUI.
US003	Enter measured cooling power measurements for their own fridge and generate a custom heat load model via data fitting.	As a cryostat user, I want to enter cooling power measurements for my own fridge, So that I can view a custom heat load model via data fitting.	Given the power user wants to assess a new set of power measurements for their fridge When the user appropriately inputs this data Then the interface should produce a heat load model for their desired configuration.
US004	Individual sessions can be stored on the database and retrieved for later re-evaluation and updates	As a poweruser, I want to be able to generate data from previous sessions So that I don’t have to manually input data for my fridge every time I access the interface	Given the user interfaces via the website or API When they request to modify previously input data Then the database should supply the previous values and data supplied by th
US005	The user should be able to easily export generated heat load data for external use/reference	As a poweruser, I want to easily export heat load data generated in the project, So that I can view and reference the data externally for later use	Given the user wants to export the data When they request this from the project Then the data should be output in a standardised and readable format, for ease of recognition and reuse
US006	The user should be able to sweep over the heat load model data, and display these sweeps visually	As a poweruser, I want to specify specific parameter values to sweep the heat load model data So that I can highlight specific interactions and relationships in the produced heat load model	Given the user wants to sweep the data. When they input all the required parameters. Then the solution should produce charts/plots from the parameters.
US007	New cable materials should be available to supply and save for use in later fridge configurations.	As a poweruser, I want to supply my own cable materials So that I can use these new materials in the wiring configuration.	Given the interface allows for new cable materials to be generated When the user supplies the relevant information to add a new cable material Then this new material should be stored in the database for later use in creating the wiring configuration
US008	New cable presets should be available to determine and save for use in later fridge configurations.	As a power user, I want to create new cable presets So that I can use new cable presets in the wiring configuration.	Given the interface allows for new cable presets to be generated When the user supplies the relevant information to add a new cable preset Then this new preset should be stored in the database for later use in creating the wiring configuration.
US009	The user should be able to supply code to modify how a cable is	As a power user, I want to input my own cable material function So that I can specify how the cable material is	Given the input code is valid When the code executes to calculate the contribution of the cable material

	calculated in the heat load model.	considered in the heat load model calculation.	Then the output heat load model data should be appropriately changed to reflect the updated calculation
US010	Customized Cooling Solutions	As a user, I want the GUI to intuitively data fit for best practice so that a custom heat model can be generated	Given the inputs are viable/optimized. When the user submits the form. Then the model will produce a configuration that most suits the input parameters.
US011	Accessibility	As a user, I want the GUI to be web-based so that I have access to it through online mediums	Given the web-based attributes of the user [reference to ERD]. When the user opens the application. Then the application should access the internet database.
US012	Plot types	As a user, I want to be able to generate the relevant plots that best represents the input parameters.	Given the paper received from the client [**reference**]. When the model gathers parameters. Then the solution should produce plots from sections 4.2, 4.3, 4.4.

3.2 User Story Narratives

Use Case ID	US001 Upload Own Parameter Values
User Story	As a cryostat user, I want to upload my own wiring and component parameter data, so that I can evaluate the expected performance of certain metrics for that particular wiring configuration.
Goal	To allow the cryostat user to upload unique parameter values to test their fridge configuration against, if not already considered in the program
Priority	High
Actors	Cryostat User
Pre-conditions	The input parameter values are valid and in a standardised format
Post-conditions	N.A.
Trigger	When selecting parameter values for the wires and its components, upon clicking the “Input Parameter” field, a pop-up window will appear to input values in a standardised format
Main Flow	1. User selects parameter data for wiring configuration 2. Selects “Input Parameter” field 3. Fills out pop-up window for new parameter value 4. Applies new parameter to configuration 5. The GUI appropriately updates to reflect these parameter values
Exceptions	Error handling for input parameter values, not within the valid standardised format
Includes/Extends/Inherits	N.A.
Supporting Information	N.A.
Non-functional Requirements	Should have no limitations for amount of different input component data as well as should be automatically stored in backend.

Use Case ID	US002 Select Pre-Supplied Parameter Values
User Story	As a cryostat user, I want to select parameter values from a list of pre-supplied wiring and component data, so that I can evaluate the expected performance of certain metrics for that particular wiring configuration.
Goal	To allow the user to select from a range of pre-evaluated parameter data to calibrate against expected results
Priority	High
Actors	Cryostat User
Pre-conditions	The pre-supplied list of parameter values is valid
Post-conditions	N.A.
Trigger	The user selects a parameter from the drop-down list to apply to the wiring or component data
Main Flow	1. User selects either a blank or pre-determined fridge configuration and moves to next tab 2. User selects drop-down for "Select Parameter Values" 3. User inputs valid parameter values in the required fields and applies to the current configuration 4. The GUI appropriately updates to reflect these input values
Exceptions	Error handling for input parameter values, not within the valid standardised format
Includes/Extends/Inherits	N.A.
Supporting Information	N.A.
Non-functional Requirements	User can swap between parameter values infinitely, with continual updates to outputs accordingly

Use Case ID	US003 Backend Operation Without GUI
User Story	As a power user, I want to operate the backend of the software independently of the GUI, so that I can perform more powerful analyses
Goal	The user can interact directly with the backend data to analyse wiring configurations without use of the GUI
Priority	High
Actors	Power User
Pre-conditions	API integration is properly established to allow direct access to backend configuration without interaction with GUI
Post-conditions	N.A.
Trigger	The user downloads the original program without any GUI implementation. The python code is run to generate plots and graphs. In addition, custom functions can be made.
Main Flow	5. User downloads the Python files from GitHub/GitLab 6. User opens the relevant coding files in their environment of choice 7. User writes their custom data/functions into the file directly 8. Graphs and plots are generated when the code is run
Exceptions	Can create unwanted errors if the code isn't written properly
Includes/Extends/Inherits	N.A.
Supporting Information	N.A.

Non-functional Requirements	Custom functions can be implemented by the user. The program should be accessible publicly on GitHub/GitLab

Use Case ID	US004
User Story	As a power user, I want to have a clear API for using the backend so that I can simply automate or integrate the solution independently of the GUI.
Goal	Enable power users to automate tasks and integrate the solution seamlessly using a clear API.
Priority	High
Actors	Power User
Pre-conditions	The solution backend is operational.
Post-conditions	Successful integration or automation by the power user.
Trigger	The power user initiates integration or automation tasks.
Main Flow	1. Power user accesses the clear API documentation. 2. Power user integrates or automates tasks according to their requirements.
Exceptions	1. API documentation is unclear or incomplete. 2. Integration or automation encounters errors.
Includes/Extends/Inherits	N.A.
Supporting Information	API documentation detailing available endpoints and functionalities.
Non-functional Requirements	API response time should be minimal for efficient automation.

Use Case ID	US005
User Story	As a user, I want the solution on as many platforms as possible so that I can access the software easily on a platform that is already available to me.
Goal	Maximize software accessibility by supporting a wide range of platforms.
Priority	Medium
Actors	User
Pre-conditions	N.A.
Post-conditions	The solution is available and accessible on multiple platforms.
Trigger	User seeks access to the software on a specific platform.
Main Flow	1. User identifies the desired platform for software access. 2. User installs or accesses the software on the chosen platform.

Exceptions	Software is not compatible with the chosen platform.
Includes/Extends/Inherits	N.A.
Supporting Information	N.A.
Non-functional Requirements	Ensure the software's responsiveness and performance across various platforms.

Use Case ID	US006
User Story	As a researcher, I want the solution to have parameter sweeping so that I can run simulations repeatedly using different values of the parameter/s of choice.
Goal	Facilitate researchers in running simulations with varied parameter values for comprehensive experimentation.
Priority	High
Actors	Researcher
Pre-conditions	The solution is installed and operational.
Post-conditions	Successful execution of simulations with different parameter values.
Trigger	Researcher initiates simulation tasks with varied parameter values.
Main Flow	1. Researcher accesses the simulation module. 2. Researcher configures parameters for the simulation. 3. Researcher initiates the simulation with the selected parameter values.
Exceptions	1. Simulation module encounters errors during configuration. 2. The simulation fails to execute with the specified parameters.
Includes/Extends/Inherits	N.A.
Supporting Information	Simulation module documentation detailing available parameters and configurations.
Non-functional Requirements	Ensure the simulation execution time is optimized for efficiency.

Use Case ID	US007 Make and save new cable materials
User Story	As a poweruser, I want to supply my own cable materials So that I can use these new materials in the wiring configuration.
Goal	To allow the end user to a dynamically modifiable service that ensures a wide variety of cable material use cases are catered for
Priority	High
Actors	Cryostat User
Pre-conditions	The input parameter values are valid and in a standardised format
Post-conditions	N.A.

Trigger	When selecting cable materials, the interface will accept a range of custom values through its open-ended data input format
Main Flow	1. User selects parameter data for wiring configuration 2. Selects “Input Material” field 3. Fills out pop-up window for new material value 4. Applies new material to configuration 5. The GUI appropriately updates to reflect these parameter values
Exceptions	Error handling for input parameter values, not within the valid standardised format
Includes/Extends/Inherits	N.A.
Supporting Information	N.A.
Non-functional Requirements	Should have no limitations for amount of different material component data as well as should be automatically stored in backend

Use Case ID	US008 New cable pre-sets available and saved for later use
User Story	As a power user, I want to create new cable presets So that I can use new cable presets in the wiring configuration.
Goal	To allow the cryostat user with a functional and user-friendly way to load presets for cables and streamline the data input process
Priority	Medium
Actors	Cryostat User
Pre-conditions	The preset values are valid and in a standardised format
Post-conditions	N.A.
Trigger	When selecting parameter values for the wires and its components, upon clicking the “Input Parameter” field, a pop-up window will appear to input values in a standardised format
Main Flow	1. User selects parameter data for wiring configuration 2. Selects “Input Parameter” field 3. Fills out pop-up window for new parameter value 4. Applies new parameter to configuration 5. The GUI appropriately updates to reflect these parameter values 6. These values can be saved as presets for later use through a button that triggers a backend ‘save’ functionality
Exceptions	Error handling for input parameter values, not within the valid standardised format
Includes/Extends/Inherits	N.A.
Supporting Information	N.A.
Non-functional Requirements	Should have no limitations for the number of different presets that can be stored in the backend of the software product.

Use Case ID	US009 User-supplied code able to modify cable calculations

User Story	As a power user, I want to input my own cable material function So that I can specify how the cable material is considered in the heat load model calculation
Goal	To allow the cryostat user to dynamically configure the software solution's calculations to ensure the heat load model is configured to their research
Priority	High
Actors	Cryostat User
Pre-conditions	The cable material function is valid and can be reflected within the given parameters
Post-conditions	N.A.
Trigger	Before selecting parameter values for the wires and its components, upon clicking the "Configure Calculations" field, a pop-up window will appear to input values in a standardised format
Main Flow	1. User selects a 'Configure Calculations' field 2. Fills out pop-up window for new calculation values 3. Applies new parameter to calculation configuration 4. The GUI appropriately updates to reflect these new changes
Exceptions	Error handling for configured calculations, not within the valid standardised format
Includes/Extends/Inherits	N.A.
Supporting Information	N.A.
Non-functional Requirements	Should have no limitations for types of configured calculations

Use Case ID	US010 Dynamically Updating Heat Load Model Visual Representation
User Story	As a power user, I want to have my generated heat loads be visually demonstrated and dynamically update upon variable change, so that I can manipulate the associated graphs in real time to highlight relevant data/outputs
Goal	Produce dynamically updating visual representations of generated heat load models
Priority	Medium
Actors	Power User
Pre-conditions	Calculated heat load model can be accurately graphed, and is dynamically updated upon parameter change
Post-conditions	N.A.
Trigger	Upon completion of inputting parameter data and generating heat load model, graphical representation is produced. This then updates upon any relevant variable change
Main Flow	1. User inputs valid heat load model data parameters 2. GUI displays graphical representation of this model 3. User updates parameter values for this heat load model 4. The graph dynamically updates to reflect these changes in real time
Exceptions	Parameter change to non-valid input value does not change the graph, preventing invalid input
Includes/Extends/Inherits	N.A.

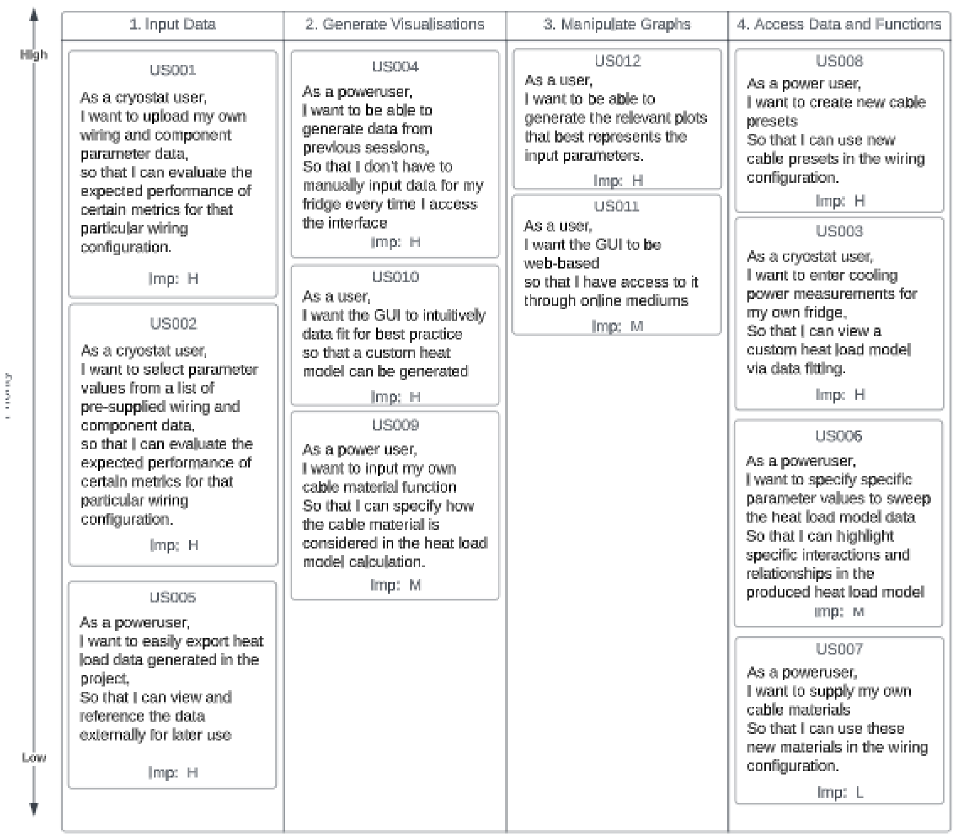
Supporting Information	N.A.
Non-functional Requirements	Updates to graphical representation should be immediate upon parameter update confirmation. Should be no lockout to amount of graph updates in a session.

Use Case ID	US011 Export Generated Heat Load Model Data
User Story	As a power user, I want to easily export heat load data generated in the project, so that I can view and reference the data externally for later use
Goal	User is able to export heat load data into a standardised format (i.e., csv file) for later reference
Priority	Medium
Actors	Power User
Pre-conditions	Generated data should be in a standardised format so that it can be easily exported to appropriate external editors
Post-conditions	N.A.
Trigger	Upon heat load model data generation, the user can select to export the data in a specific file type
Main Flow	1. The user interacts with the web interface or API to generate a heat load model. 2. They then select to export the generated data. 3. They will then select the file type under which this will then be exported. 4. An associated file is then created with the relevant heat load model data.
Exceptions	Exported data can only be in a standardised format, and only in specified file types.
Includes/Extends/Inherits	N.A.
Supporting Information	N.A.
Non-functional Requirements	Export of data should be reasonably fast. Data should be in a readable, standardised format

Use Case ID	US012 Sweep Heat Load Model Data
User Story	As a power user, I want to specify specific parameter values to sweep the heat load model data, so that I can highlight specific interactions and relationships in the produced heat load model.
Goal	The user can sweep over the heat load model and produce associated graphs in accordance with these sweeps.
Priority	High
Actors	Power User
Pre-conditions	All fridge configuration data must be loaded and validated within the browser prior to execution of sweep calculations.
Post-conditions	N.A.
Trigger	Upon completing the definition of a sweep pattern (through user defined python code) a graphical representation is generated.

Main Flow	1. User defines sweep parameters through python code with references to desired data points and expected sweep range. 2. User executes sweep. 3. API executes user's python code to get necessary parameters. 4. API performs sweep calculations. 5. Produced data is tabulated and graphed for the user
Exceptions	User defined sweeps can only make use of limited variables, functions, are lengths. Any invalid definitions will result in a graph not being generated.
Includes/Extends/Inherits	N.A.
Supporting Information	N.A.
Non-functional Requirements	Sweep data definitions should allow for standard python functions and common libraries (e.g. numpy or pandas).

3.3 User Story Map



1. Input Data

US001
As a cryostat user,
I want to upload my own wiring and component parameter data,
so that I can evaluate the expected performance of certain metrics for that particular wiring configuration.
Imp: H

US002
As a cryostat user,
I want to select parameter values from a list of pre-supplied wiring and component data,
so that I can evaluate the expected performance of certain metrics for that particular wiring configuration.
Imp: H

US005
As a poweruser,
I want to easily export heat load data generated in the project,
So that I can view and reference the data externally for later use
Imp: H

2. Generate Visualisations

US004
As a poweruser,
I want to be able to generate data from previous sessions,
So that I don't have to manually input data for my fridge every time I access the interface
Imp: H

US010
As a user,
I want the GUI to intuitively data fit for best practice
so that a custom heat model can be generated
Imp: H

US009
As a power user,
I want to input my own cable material function
So that I can specify how the cable material is considered in the heat load model calculation.
Imp: M

3. Manipulate Graphs

US012
As a user,
I want to be able to generate the relevant plots that best represents the input parameters.
Imp: H

US011
As a user,
I want the GUI to be web-based
so that I have access to it through online mediums
Imp: M

4. Access Data and Functions

US008
As a power user,
I want to create new cable presets
So that I can use new cable presets in the wiring configuration.
Imp: H

US003
As a cryostat user,
I want to enter cooling power measurements for my own fridge,
So that I can view a custom heat load model via data fitting.
Imp: H

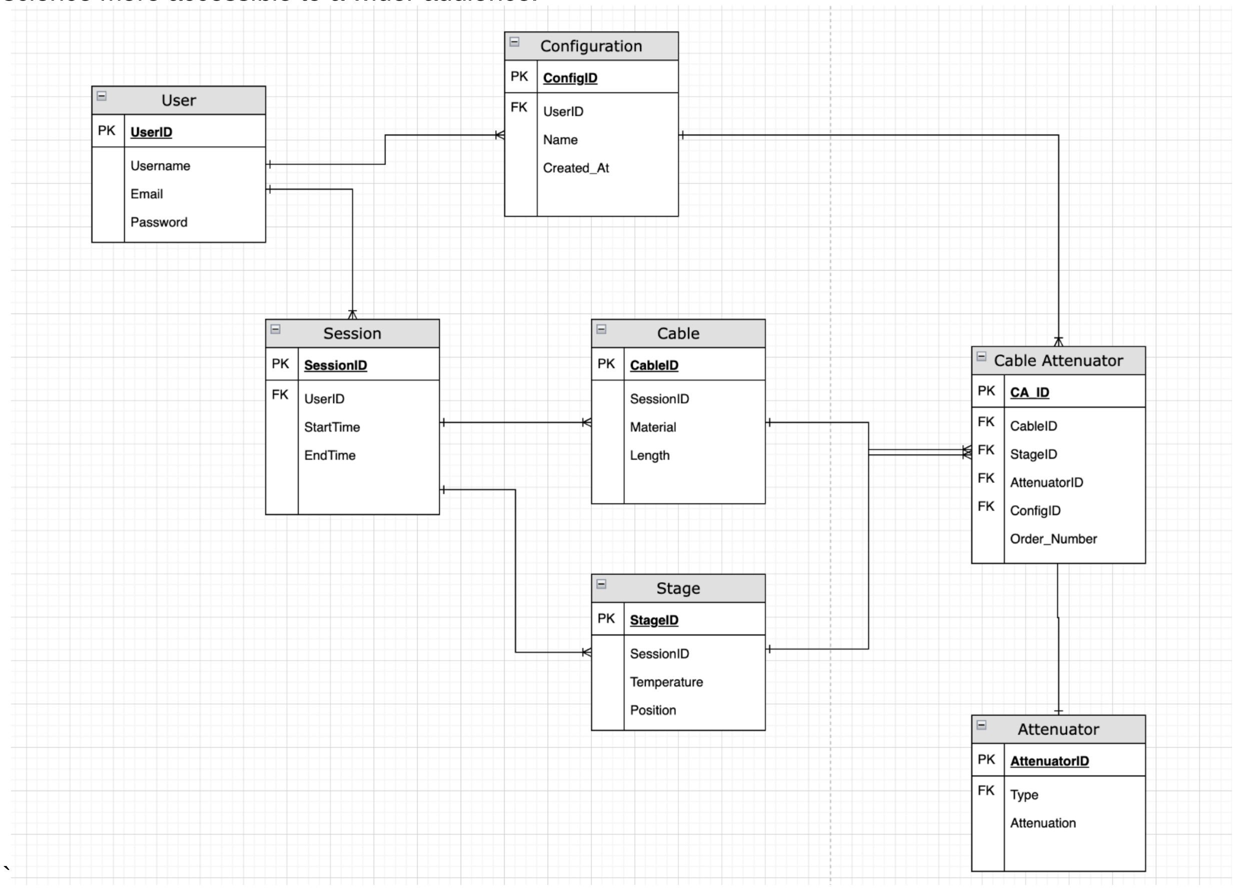
US006
As a poweruser,
I want to specify specific parameter values to sweep the heat load model data
So that I can highlight specific interactions and relationships in the produced heat load model
Imp: M

US007
As a poweruser,
I want to supply my own cable materials
So that I can use these new materials in the wiring configuration.
Imp: L

4. Data Requirements

4.1 Conceptual Data Model

The CirQuS UTS project aims to simplify the complex task of cryogenic wiring by offering a user-friendly software tool. To make this tool effective and easy to use, we've designed a clear and flexible Information Architecture. This setup manages different types of data, like user details and wiring configurations, and explains how they're related to each other. We use an Entity-Relationship Diagram (ERD) to visualize these connections and a Data Dictionary to describe what each type of data means. This approach makes the system easier to understand, manage, and extend, aligning with our goal to make advanced cryogenic science more accessible to a wider audience.



1. User: Represents users who can log into the system. Each user has a unique identifier (`UserID`), along with `Username`, `Email`, and a hashed `Password`.

2. Session: Contains session-related information for users. The session is identified by a unique `SessionID` and linked to a user by `UserID`. It also has `StartTime` and `EndTime` fields to keep track of the session duration.

3. Configuration: Represents the specific wiring configurations created by users. Each configuration is identified by a unique `ConfigID` and linked to a user by `UserID`. Other details like the `Name` and `Created_At` timestamp are also stored.

4. Cable: Stores information about the cables used in configurations. Each cable is identified by a unique `CableID` and optionally linked to a session (`SessionID`). Attributes include the `Material` and `Length` of the cable.

5. Stage: Contains details about the stages within the cryostat. Each stage is identified by a unique `StageID` and optionally linked to a session (`SessionID`). The `Temperature` and `Position` of each stage are stored.

6. Attenuator: Stores information about attenuators. Each attenuator is identified by a unique `AttenuatorID`. Attributes include `Type` and `Attenuation`.

7. Cable_Attenuators: This is a complex entity that represents the relationships between cables, attenuators, and stages within a configuration. It has a unique identifier (`CA_ID`) and references to `CableID`, `StageID`, `AttenuatorID`, and `ConfigID`. An additional field, `Order_Number`, helps in maintaining the structure.

The design aims for flexibility, enabling users to create and manage configurations while also allowing for the sharing and private session-based usage of cables and stages. It should also support complex configurations involving multiple cables, stages, and optional attenuators.

4.2 Data Dictionary

User

Field Name	Format	Length	Description
UserID	Integer	32-bit	Unique identifier for each user.
Username	String	50 characters	Username chosen by the user.
Email	String	255 characters	Email address of the user.
Password	String	255 characters	Hashed password for user authentication.

Session

Field Name	Format	Length	Description
SessionID	Integer	32-bit	Unique identifier for each session.
UserID	Integer	32-bit	Foreign key that links to User.
StartTime	Timestamp	N/A	Start time of the session.
EndTime	Timestamp	N/A	End time of the session.

Configuration

Field Name	Format	Length	Description
ConfigID	Integer	32-bit	Unique identifier for each configuration.
UserID	Integer	32-bit	Foreign key that links to User.
Name	String	255 characters	Name or title of the configuration.
Created_At	Timestamp	N/A	The date and time when the configuration was created.

Cable

Field Name	Format	Length	Description
CableID	Integer	32-bit	Unique identifier for each cable.
SessionID	Integer	32-bit	Foreign key that links to Session for private cables.
Material	String	50 characters	Material of the cable.
Length	Float	N/A	Length of the cable in meters.

Stage

Field Name	Format	Length	Description
StageID	Integer	32-bit	Unique identifier for each stage.
SessionID	Integer	32-bit	Foreign key that links to Session for private stages.
Temperature	Float	N/A	Temperature of the stage in millikelvin.
Position	Integer	32-bit	Position of the stage within the configuration.

Attenuator

Field Name	Format	Length	Description
AttenuatorID	Integer	32-bit	Unique identifier for each attenuator.
Type	String	50 characters	Type of the attenuator.
Attenuation	Float	N/A	Attenuation value.

Cable Attenuator

Field Name	Format	Length	Description
CA_ID	Integer	32-bit	Unique identifier for each cable-attenuator relationship.
CableID	Integer	32-bit	Foreign key that links to Cable.
StageID	Integer	32-bit	Foreign key that links to Stage.
AttenuatorID	Integer	32-bit	Foreign key that links to Attenuator (optional).
ConfigID	Integer	32-bit	Foreign key that links to Configuration.

Order_Number	Integer	32-bit	Order of the cable within the stage for structuring.

5. Non-functional requirements

5.1 Non-Functional Requirements (User Stories)

ID	Initial Requirement	User Story	Acceptance Criteria
US013	The software backend should be well structured and designed to be accessible independently of the GUI, so that it can potentially be released directly to "power users" if requested.	As a power user, I want to operate the backend of the software independently of the GUI, So that I can perform more powerful analyses.	functionality does not currently exist, When the user has requested to access the software's backend, Then ensure the backend of the software can be accessed and operated independent of the GUI.
US014	Provide a pictorial representation of the wiring configuration as users select parameters.	As a cryostat user, I want to view a pictorial representation of the wiring configuration as I select parameter values, So that I can visually confirm whether the wiring design is aligned with my expectations based on the parameter values that I inputted.	Given that the parameter value that was inputted is valid and in a standardized format, When the user requests to proceed with the simulation, Then ensure the visualisation of the wiring configuration performance of each metric for that particular wiring configuration is displayed on the GUI.
US015	The project should be packaged using a docker container for ease of install and portability between devices.	As a power user, I want to be able to run the API and Database using a Docker container So that it is easily installable across multiple devices	Given the user wants to install the project on a new device When they attempt to do so Then they should only have to install one package, with all the dependencies and requirements being installed automatically also
US016	The user should be able to interface with the API to generate data for their fridge simulations, without interacting with the GUI.	As a power user, I want to interface with the API directly and independently So that I can effectively make use of the simulations without a web interface.	Given the user interfaces with the API directly When they perform calculations utilising the GraphQL infrastructure interface Then ensure all necessary actions are appropriately accessible and generate the relevant data.
US017	Previously generated heat load models should be stored client-side easy re-access	As a power user, I want to be able to store and retrieve heat load model data So that I can easily retrieve this data for ease of analysis and comparison against other models	Given the user wants to store a heat load model When the user stores this data and tries to later retrieve it Then the data should be appropriately reinstated, with all supplied input values and their corresponding outputs accessible
US018	Heat loads should be visually represented, with dynamic generation upon variable change and removal	As a power user, I want to have my generated heatloads be visually demonstrated and dynamically update upon variable change So that I can manipulate the associated graphs in real time to highlight relevant data/outputs	Given the user has input valid heat load model data When they request graphs associated with this data Then this should be appropriately demonstrated in graphs, that update in real time to reflect any variable changes requested by the user.

5.2 User Interface Requirements (UXD)

The user interface (UI) of the CirQuS Wiring Project software should prioritize usability, clarity, and efficiency. The design must cater to the needs of both novice and experienced users, ensuring intuitive navigation and interaction. The UI should facilitate the core functionalities of the software, such as data input, simulation, and visualization.

1. Layout and Design (UXD01):

- Aesthetic and Clarity: The interface should have a clean, modern aesthetic with a logical layout, prioritizing ease of use and clarity.
- Consistency: Elements like color schemes, typography, and button styles should be consistent throughout the application.

2. Navigation (UXD02):

- Intuitive Navigation: Menus and navigation tools should be intuitive, allowing users to easily find and access different sections of the software.
- Breadcrumb Trails: Implement breadcrumb trails for complex menus or settings to help users keep track of their location within the application.

3. Accessibility (UXD03):

- Compliance with Standards: The UI must comply with accessibility standards, such as WCAG, to ensure it is usable by people with disabilities.
- Responsive Design: The interface should be responsive, functioning effectively across different devices and screen sizes.

4. Interaction and Feedback (UXD04):

- User Feedback: The system should provide immediate and clear feedback in response to user actions, such as confirming successful operations or displaying error messages.
- Help and Support: Include tooltips, help icons, or a FAQ section to guide users through complex features or settings.

5. Data Visualization (UXD05):

- Graphical Representations: Implement clear and effective graphical representations for data, such as wiring diagrams or simulation results.
- Customization: Allow users to customize visual elements of graphs or charts, like colors or data points, to suit their preferences.

6. User Input and Forms (UXD06):

- Input Validation: Include input validation to prevent errors and ensure data integrity.
- Ease of Data Entry: Design forms and data entry points to be user-friendly, minimizing the effort required to input or modify data.

7. Compatibility and Performance (UXD07):

- Browser Compatibility: Ensure compatibility with major web browsers to maximize user accessibility.
- Performance Optimization: The UI should be optimized for performance, minimizing load times and ensuring smooth interactions.

8. Customization and User Preferences (UXD08):

- User Customization: Provide options for users to customize the UI according to their preferences, like theme selection or layout adjustments.
- Save User States: The system should remember user states and preferences between sessions.

6. Bibliography

Books and Academic Journals:

- Cooper, Alan. "About Face: The Essentials of Interaction Design." Wiley, 4th edition, 2014.
- Nielsen, Jakob, and Loranger, Hoa. "Prioritizing Web Usability." New Riders, 2006.
- Norman, Don. "The Design of Everyday Things." Basic Books, Revised and Expanded Edition, 2013.

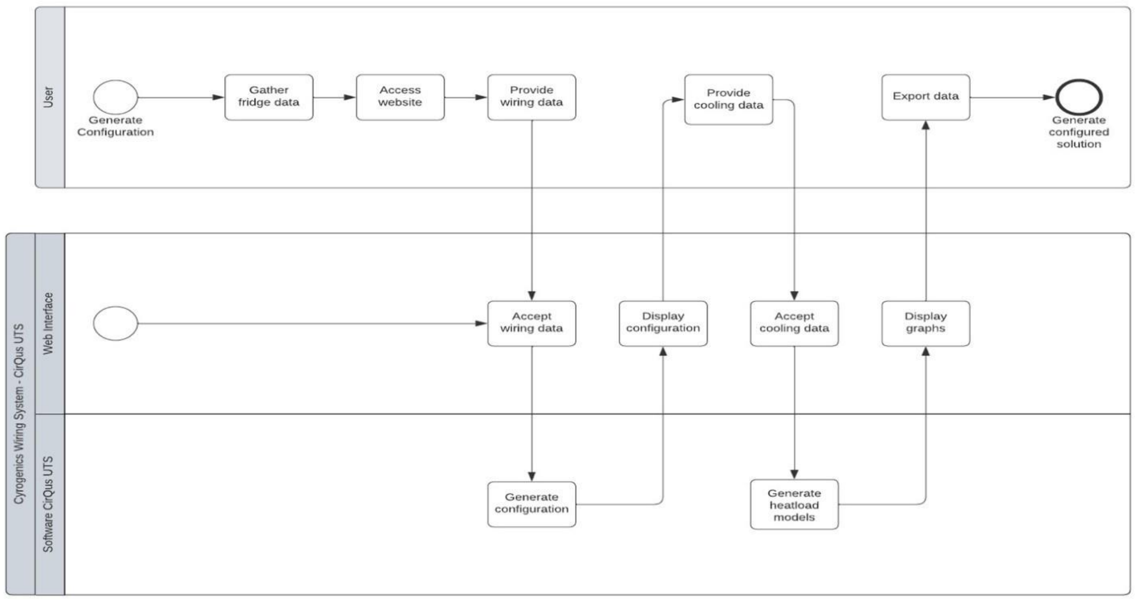
Online Resources and Articles:

- "Web Content Accessibility Guidelines (WCAG) 2.1." W3C, 2018. https://www.w3.org/TR/WCAG21/
- "Python Documentation." Python Software Foundation. https://docs.python.org/3/

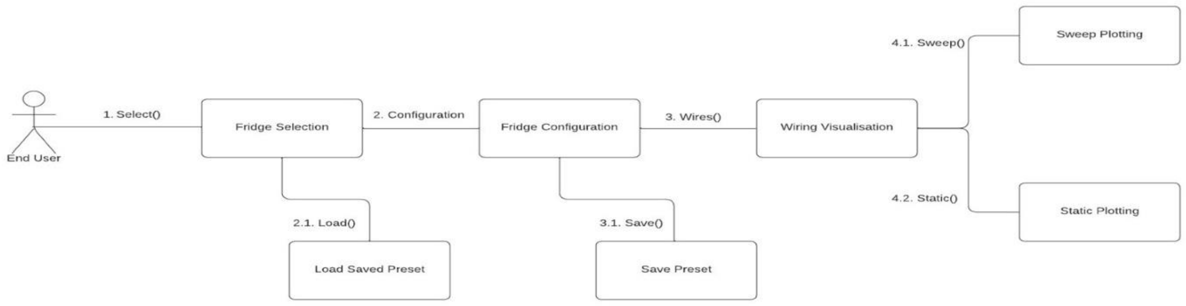
7. Appendices

7.1 Process Maps

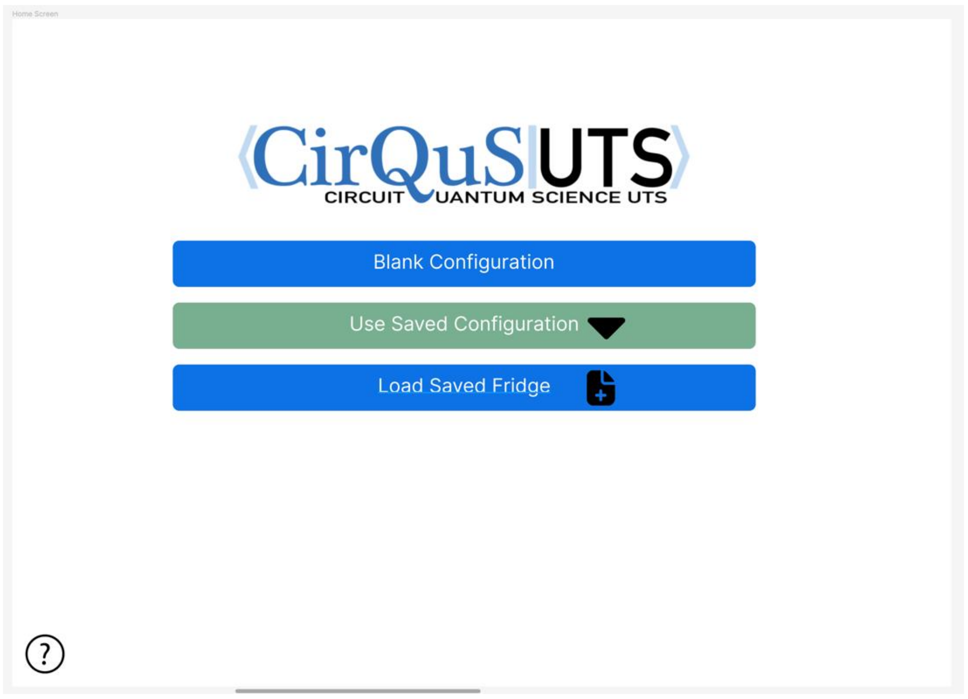
7.1.1 Business Process Model Diagram



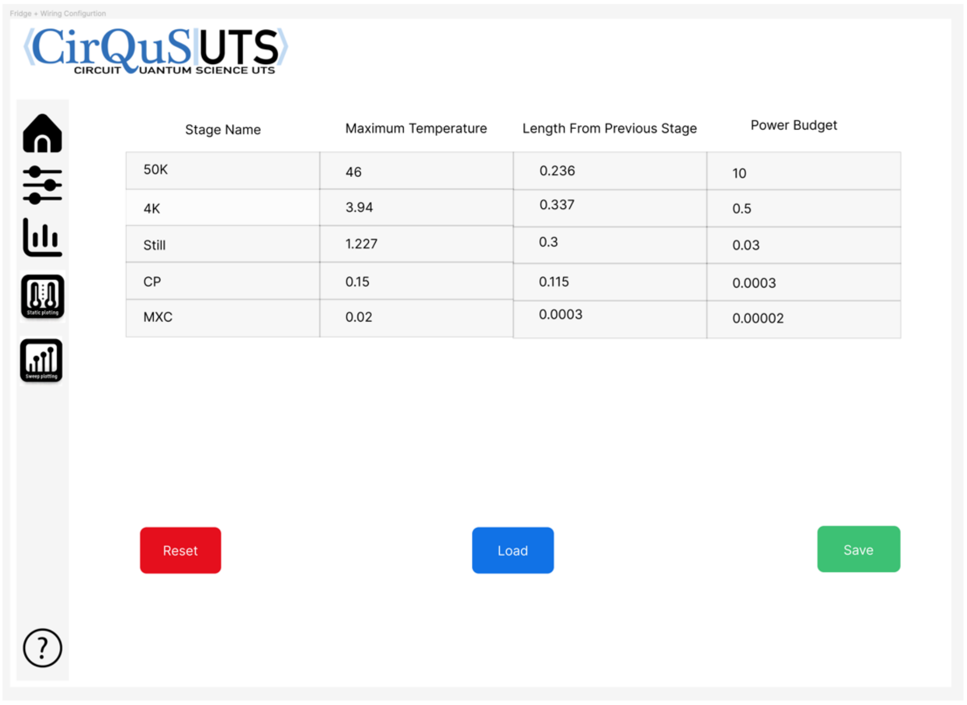
7.1.2 Communication Diagram



7.2 UI Wireframe/Design

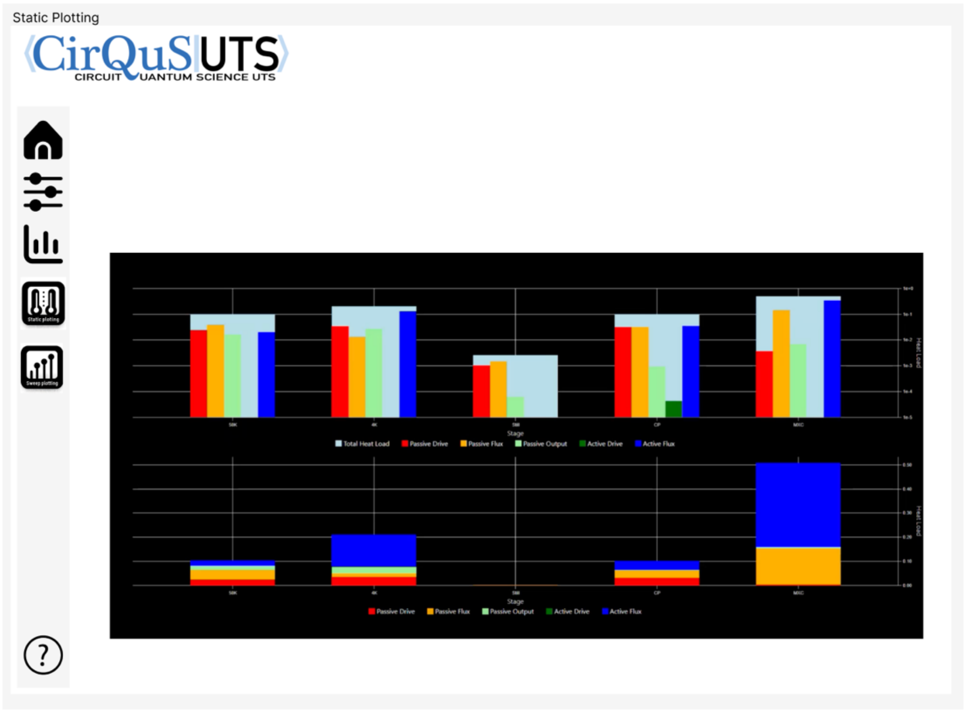


Appendix A: Main Screen

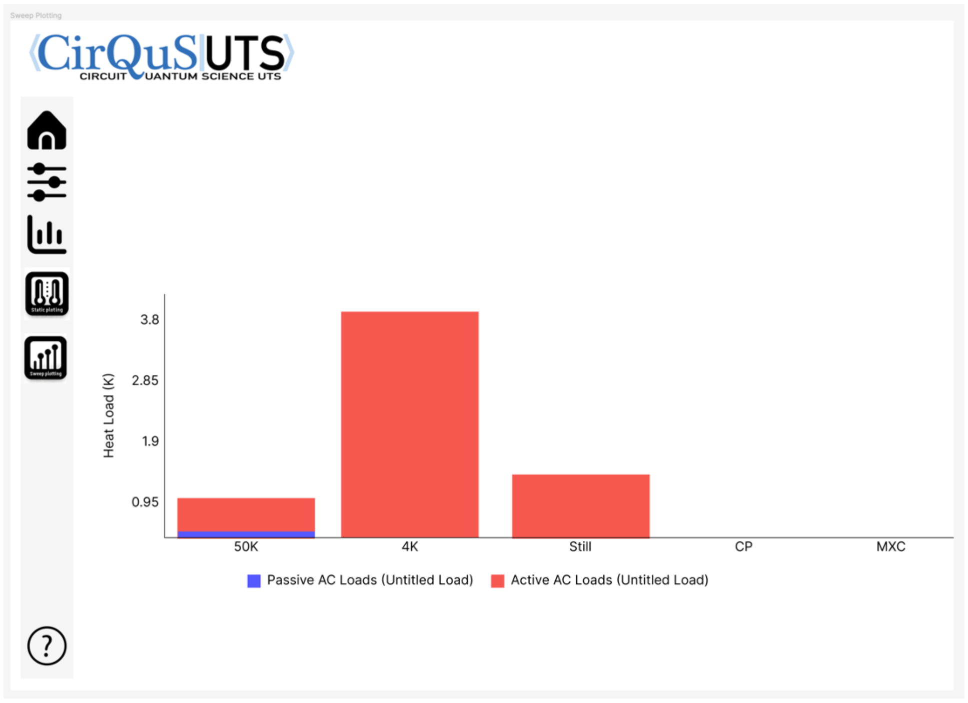


Stage Name	Maximum Temperature	Length From Previous Stage	Power Budget
50K	46	0.236	10
4K	3.94	0.337	0.5
Still	1.227	0.3	0.03
CP	0.15	0.115	0.0003
MXC	0.02	0.0003	0.00002

Appendix B: Fridge + Wiring Configuration



Appendix C: Static Plotting


Sweep Plotting
CirQuS UTS
CIRCUIT QUANTUM SCIENCE UTS
Static plotting
Sweep plotting
Heat Load (K)
3.8
2.85
1.9
0.95
50K
4K
Still
CP
MXC
Passive AC Loads (Untitled Load)
Active AC Loads (Untitled Load)